IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received June 18, 2021, accepted July 8, 2021, date of publication July 20, 2021, date of current version July 29, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3098730

ExPAN(N)D: Exploring Posits for Efficient Artificial Neural Network Design in FPGA-Based Systems

SURESH NAMBI[1], SALIM ULLAH[1], SIVA SATYENDRA SAHOO[1], ADITYA LOHANA[1], FARHAD MERCHANT[2], (Member, IEEE), AND AKASH KUMAR[1], (Senior Member, IEEE)

[1]Chair of Processor Design, Center for Advancing Electronics Dresden (cfaed), TU Dresden, 01062 Dresden, Germany
[2]Institute for Communication Technologies and Embedded Systems, RWTH Aachen, 52056 Aachen, Germany

Corresponding authors: Salim Ullah (salim.ullah@tu-dresden.de) and Siva Satyendra Sahoo (siva_satyendra.sahoo@tu-dresden.de)

This work was supported by the German Research Foundation (DFG) funded Project "Runtime Reconfigurable Approximate Architecture" (ReAp) under Grant 380524764.

ABSTRACT The high computational complexity, memory footprints, and energy requirements of machine learning models, such as Artificial Neural Networks (ANNs), hinder their deployment on resource-constrained embedded systems. Most state-of-the-art works have considered this problem by proposing various low bit-width data representation schemes and optimized arithmetic operators' implementations. To further elevate the implementation gains offered by these individual techniques, there is a need to cross-examine and combine these techniques' unique features. This paper presents ExPAN(N)D, a framework to analyze and ingather the efficacy of the *Posit* number representation scheme and the efficiency of *fixed-point* arithmetic implementations for ANNs. The Posit scheme offers a better dynamic range and higher precision for various applications than IEEE 754 single-precision floating-point format. However, due to the dynamic nature of the various fields of the Posit scheme, the corresponding arithmetic circuits have higher critical path delay and resource requirements than the single-precision-based arithmetic units. Towards this end, we propose a novel *Posit to fixed-point converter* for enabling high-performance and energy-efficient hardware implementations for ANNs with minimal drop in the output accuracy. We also propose a modified Posit-based representation to store the trained parameters of a network. With the proposed Posit to fixed-point converter-based designs, we provide multiple design points with varying accuracy-performance trade-offs for an ANN. For instance, compared to the lowest power dissipating Posit-only accelerator design, one of our proposed designs results in 80% and 48% reduction in power dissipation and LUT utilization respectively, with marginal increase in classification error for Imagenet dataset classification using VGG-16.

INDEX TERMS Computer arithmetic, deep neural networks, energy efficient computing, posits, FPGA, high-level synthesis.

I. INTRODUCTION

Machine learning algorithms have become an essential factor in various modern applications, such as scene perception and image classification [1]–[3]. Over the past few years, these algorithms have mainly relied on the performance of modern computing systems to support the increasing complexity of the algorithms. For example, the massively parallel architectures, such as Graphics Processing Units (GPUs), and cloud-based computing have been traditionally used to train these algorithms. However, to utilize these trained machine learning models on resource-constrained embedded systems, the computational complexity and storage requirements of these algorithms must be reduced.

Many recent works have considered this problem to define various optimization techniques to reduce the complexity of machine learning models, such as Artificial Neural Networks (ANN). For example, the techniques used in [5] and [6] have employed the sparsity of Deep Neural Networks (DNN) to reduce the total number of trained parameters. The works in [7], [8] and [9] have explored other number representation techniques, such as *bfloat16*, *Posit* and *Fixed Point (FxP)*,

The associate editor coordinating the review of this manuscript and approving it for publication was Hongli Dong.

to overcome the storage requirements of single-precision IEEE-754 Floating Point (FP32). Depending on the configuration used, each of these number representation techniques provides different dynamic range to represent the parameters (weights and biases) of a network. For example, Fig. 1(a) shows the FP32-based distribution of the pre-trained weights of the Conv2_1 layer of VGG16 DNN [4]. The pre-trained weights have a dynamic range between -0.3 to $+0.3$, with most of the weights clustered around '0'. To reduce the memory footprint of the weights and associated computational complexity, Fig. 1(b) represents the distribution using an 8-bit fixed point linear quantization scheme, referred to as FxP8. The FxP8 provides a set of 256 uniformly distributed discrete values, which generates an average relative error of 0.295 in the quantized weights. To reduce the quantization-induced errors, Fig. 1(c) shows the trained parameters using an 8-bit Posit scheme. The Posit representation maps the FP32 weights better due to denser clustering of values around 0, resulting in an average relative error of 0.052 in the quantized weights. Therefore, it is imperative to define number representation schemes (or quantization methods), which can maintain FP32-based machine learning models' accuracy within a desired limit while reducing their corresponding computational complexity and storage requirements.

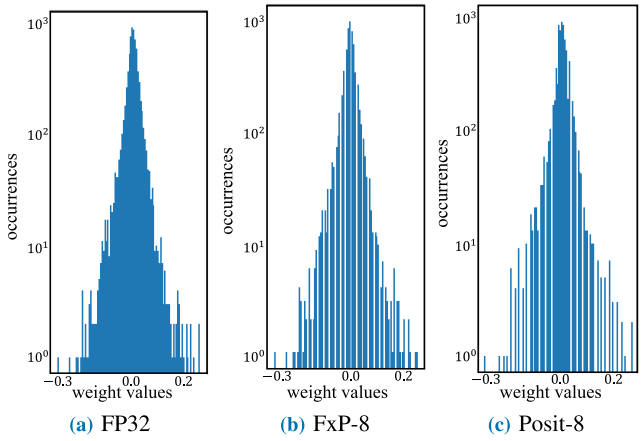


Fig. 1. Distribution of pre-trained weights of Conv2_1 layer of VGG16 [4]. (a) Single-precision floating-point, (b) 8-bit linear fixed-point quantization: average absolute relative quantization-induced error = 0.295 (c) Posit (8, 2)-based quantization: average absolute relative quantization-induced error = 0.052.

The various number representation schemes (quantization methods) result in varying performance overheads of their associated arithmetic hardware. For example, Fig. 2 shows the comparison of the effect of using different quantization methods across multiple performance aspects – behavioral (error in the quantization of weights), computational (critical path delay of a Multiply and ACcumulate (MAC) unit), and memory requirements (weights' storage) in the Conv2_1 layer of pre-trained VGG16. The hardware implementation results have been obtained by implementing each technique on the Xilinx UltraScale Field Programmable Gate Array (FPGA) using Vivado HLS 2018.2. For a fair

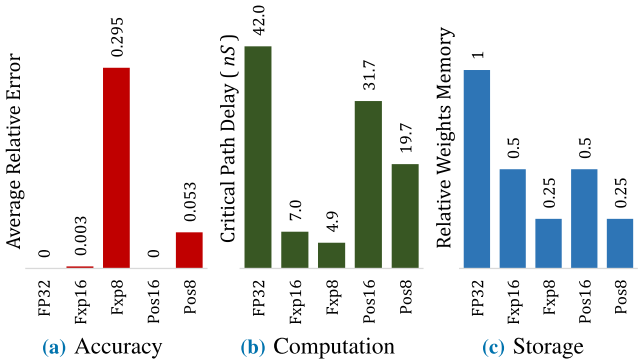


Fig. 2. Accuracy and performance comparison of various schemes for numbers representation for the Conv2_1 layer of pre-trained VGG16 [4]: (a) Average absolute relative error with respect to FP32-based parameters, (b) critical path delay, (c) normalized memory footprints.

comparison, the critical path delay (CPD) is obtained from MAC units implemented using 6-input lookup tables (LUTs) and with a latency of a single cycle. Modern FPGAs, such as Xilinx UltraScale, also host DSP blocks for performing MAC operations. However, as shown by our previous work in [10], an FPGA has a limited number of DSP blocks, and it is always advantageous to have LUT-based MAC units along with the DSP blocks. Further, the DSP blocks' fixed location can also result in creating extra routing for an implementation. As shown by our results, higher bit-widths for the quantization schemes significantly reduce quantization-induced errors. The FP32 implementation has the highest memory footprint with the worst CPD of 42*ns*. The Posit schemes provide better coverage of the FP32-based pre-trained parameters than the corresponding FxP-based schemes. However, the FxP-based arithmetics' simplicity results in significantly reducing the CPD of the MAC units when compared with the corresponding Posit schemes.

Most state-of-the-art works do not consider application-specific optimizations to the quantization methods. For instance the Posit related works focus on representing the whole range of real numbers, $(-\infty, \infty)$, rather than the actual range of the parameters in the application. Similarly, many related works consider each quantization method in isolation and do not attempt to leverage the best features of multiple methods. To this end, we propose *ExPAN(N)D* framework for Exploring the joint use of Posit and FxP representations for Designing efficient ANNs. The major contributions in this paper are as follows.

Contributions:

1) We propose a reduced bit-length Posit-based representation that improves the encoding efficiency of weight normalized ANNs to reduce the communication and storage costs. Using our proposed representation for each N-bit Posit number within the reduced weight normalized range, we only store $N-1$ bits.

2) We propose a novel arithmetic hardware design, referred to as Posit to Fixed Point (PoFx), that aims to combine the best of both Posit and FxP number representations.

The proposed hardware unit offers resource-efficient and low-latency conversion of Posit-based numbers to FxP-based numbers to leverage the lower computation overheads of fixed-point arithmetic. For example, compared to compared to 8-bit FxP, an 8-bit PoFx-based MAC has at-most 15.5% resource overhead (with PoFx(7, 2)) and provides upto 46% reduction in the storage requirement (with PoFx(6, 0)) of a network's parameters in an accelerator.

3) Framework for Behavioral Analysis: We provide a high-level framework for the efficient and thorough exploration of various quantization schemes to satisfy the accuracy constraints of a DNN. The proposed framework explores the limitations and the interplay of various quantization schemes, such as FxP to Posit to FxP, to minimize the quantization-induced errors. The framework prunes the non-optimal quantization configurations by analyzing the quantization induced-errors in (a) parameters of individual layers, (b) output activations of each layer using quantized weights, and (c) final output of the network. For example, our framework explores various N-bit Posit configurations to achieve output accuracy comparable to an M-bit FxP-based quantization, where $N < M$.

4) We explore the impact of using the proposed hardware designs in a fully-connected layer. Specifically we use an automated design flow, using state-of-the-art High Level Synthesis (HLS) tools, to explore storage-computation trade-offs in the design of FPGA-based accelerators for ANNs. For example, compared to an 8-bit FxP-based accelerator, the PoFx-based accelerator(PoFx(6, 2)) provides up to 27% and 13% reductions in the power and resource requirements of the accelerator with a cost of 0.32% additional classification error.

The rest of the paper is organized as follows. In Section II, we provide the relevant background and brief overview of related work. The system model used for the evaluation of the proposed methods is presented in Section III. In Section IV, we explain the proposed methodology for exploring the use of Posit representation for ANNs, along with the proposed hardware designs. In Section V, we discuss the results from the experimental evaluation of the different components of the proposed methodology at multiple design-levels. Finally, we conclude the article in Section VI with a summary and a discussion on the scope for related future research.

II. BACKGROUND AND RELATED WORKS

A. POSIT NUMBER SYSTEM

The IEEE 754-2008 compliant floating-point (floats)-based arithmetic has become ubiquitous in modern-day computing and is deeply embedded in compilers and low-level software routines. However, the floats have several limitations, such as non-identical results across systems, redundant/wasted bit patterns, and a limited dynamic range. The Posit number scheme overcomes these limitations by offering a better dynamic range and portability across various computing platforms [11]. Fig. 3 shows the various fields (*sign*, *regime*, *exponent* and *fraction*) of the Posit number scheme.

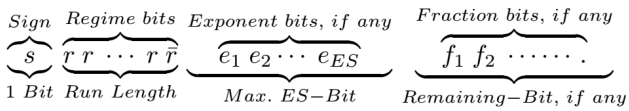


Fig. 3. Posit number representation.

A Posit configuration is characterized by its total bit length (N) and the number of bits reserved for exponent (ES). Utilizing the four fields of the Posit scheme, Eq. 1 defines the computation of a Posit value. The *regime* field, in Fig. 3, is utilized to compute the value of k in Eq. 1. The *regime* field is terminated when an inverted bit ($\bar{r}$) is encountered, and the associated value of k is determined by the number of identical bits (m); if the identical bits are a string of $0s$, then $k = -m$; if they are a string of $1s$, then $k = m - 1$. Next, the *exponent* (e) and *fraction* values (f) are determined using the remaining bits. The utilization of *regime* field provides a better dynamic range to Posit number scheme. For example, the authors in [12] have reported that for some applications, the n-bit floats can be replaced by m-bit Posit-based numbers (where $m < n$) to achieve comparable output accuracy. With an appropriate configuration of exponent size and total bit-width, a posit number can be formed to act as an IEEE 754-2008 compliant floating-point number. However, posit arithmetic supports only one rounding mode that is *round to nearest, ties to even*.

$$Posit\ value = s * (2^{2^{ES}})^k * 2^e * 1.f \tag{1}$$

Compared to the floats and fixed-point number representation schemes, Posit requires more computational resources. In the following section, we summarize the state-of-the-art works related to hardware implementation of Posit-based arithmetic circuits.

B. POSIT ARITHMETIC HARDWARE

The major challenges faced while developing an efficient hardware implementation for Posit arithmetic involve– (1) handling run-time length variation in individual Posit fields, (2) extraction of Posit components to facilitate further manipulation and, (3) implementation of rounding algorithms as proposed in the Posit standard. TABLE 1 presents an overview of the state-of-the-art work related to Posit-based arithmetic and highlights our proposed framework's key focus. These works are summarized below.

The authors in [12] tackle run-time varying field length by developing hardware arithmetic architectures for conversion from Posit to floating point and vice-versa. The work in [15] proposes a tool to generate pipelined Posit operators to be used as a drop-in replacement in processing units. In [13], authors present the architecture of a parameterized Posit arithmetic unit to generate posit adders and multipliers of any bit-width. Similarly, PACoGen [14] employs a three-stage process which involves Posit data extraction, core arithmetic processing and Posit construction to perform parameterized Posit arithmetic including multiplication and division.

TABLE 1. Posit-based hardware developments at a glance.

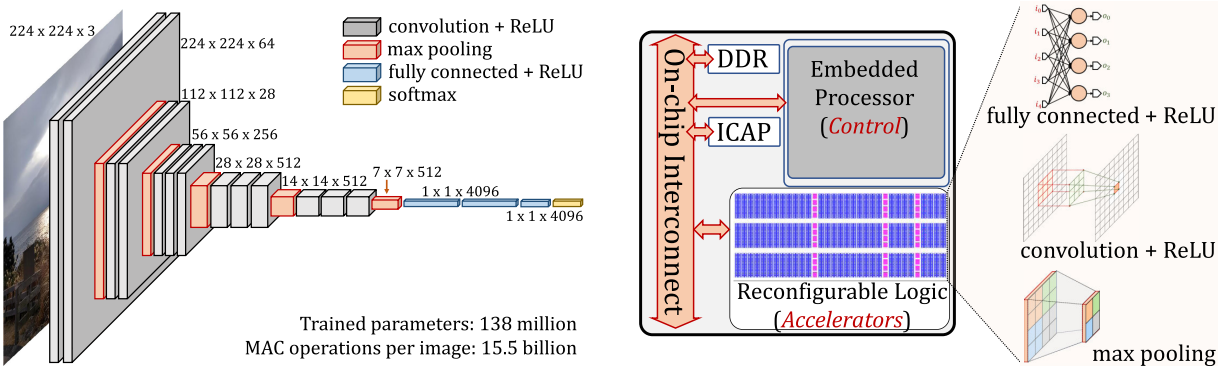
Related Work	Main Objective	Degrees of Freedom	Posit-based Arithmetic	FxP-based Arithmetic	ANN-specific Optimizations	Energy-Aware	Open Source
Chaurasiya, et.al [12]	Posit Arithmetic Unit Generator	Computation	✓	✗	✗	✗	✗
Jaiswal, et. al [13]	Posit Arithmetic on FPGA	Computation	✓	✗	✗	✗	✓
Jaiswal, et. al [14]	Posit Arithmetic Architectures	Computation	✓	✗	✗	✗	✓
Podobas, et.al [15]	Posit-based hardware implementation	Computation	✓	✗	✗	✗	✗
Carmichael, et.al [16]	Posit-based DNNs	Computation, Communication	✓	✗	✓	✗	✗
Langroudi, et.al [17]	Posit-based DNNs with adpative dynamic range	Computation, Communication	✓	✗	✓	✗	✗
Cococcioni, et.al [18]	Posit-based DNNs for image processing	Computation	✓	✗	✓	✗	✗
Murillo, et.al [19]	Posit-based Deep Learning Framework	Computation	✓	✗	✓	✗	✓
Langroudi, et.al [20]	Posit-based Deep Neural Networks Inference	Computation	✓	✗	✓	✗	✗
Langroudi, et.al [21]	Posit vs Fixed point for Deep Learning Inference	Computation	✓	✓	✓	✗	✗
Zhang, et.al [22]	Posit-based Multiply and Accumulate Unit	Computation	✓	✗	✗	✗	✗
ExPAN(N)D	Posit based DNNs	Storage, Computation, Communication	✓	✓	✓	✓	✓

It proposes improvements in Posit data extraction methodology and a pipelined architecture for Posit (N = 32, ES = 6). Posit arithmetic has also been integrated into Clarinet [23] which is a RISC-V ISA based processor that supports the use of a Posit arithmetic core. However, the RISC-V implementations are not capable of handling large-scale applications.

C. ARITHMETIC HARDWARE FOR ANN INFERENCE

A plethora of recent works have considered different quantization schemes to reduce the memory footprints and computational complexity of DNNs for resource-constrained embedded systems and edge devices for IoT. These techniques can be categorized into (a) in-training quantization, and (b) post-training quantization schemes. For example, the techniques proposed in [24]–[27] have considered various fixed-point schemes for in-training quantization. The in-training quantization schemes can overcome most of the quantization-induced errors. However, these techniques cannot be utilized for the quantization of the parameters of pre-trained DNNs. For example, for the quantization of pre-trained DNNs, [9], [28]–[30] have proposed different schemes. The techniques presented in [29], [30] have focused on the utilization of logarithmic data representations to avoid the computationally expensive multiplication operations. However, some recent works, such as [31]–[33] have utilized fixed-point quantization schemes to employ the well-explored high-performance and energy-efficient approximate adders and multipliers. The utilization of approximate arithmetic units [10], [34]–[36] provides another degree of freedom for achieving the accuracy, performance, and energy constraints of DNNs for IoT. For example, the authors of [31] have utilized the library of approximate multipliers [35] to provide approximate accelerators for reduced-precision DNNs. Some recent works have also explored the utilization of Posit numbers for training and inference phases of ANN. For example, the work in [18] has used ARM scalable vector extension SIMD engine to present vectorized extensions for the cppPosit C++ posit arithmetic library. The authors of [16] have proposed an exact multiply and accumulate (EMAC) for implementing the MAC operations in ANN. Their results show that the Posit-based representation of networks' parameters performs better than fixed-point-based representation in retaining the output accuracy of ANN. However, the Posit-based EMACs have significantly higher resource utilization and energy-delay product (EDP) than the fixed-point-based MAC operations. In [22], the authors have also proposed a parametrized Posit MAC generator to produce the HDL code of a Posit MAC unit. However, they do not present the efficacy of their proposed design in any real-world application. In [20], the authors have also used the EDP metric to compare their proposed Posit-based framework with the FP32- and FxP-based implementations; the FxP-based implementations always produce lower EDP values than the corresponding Posit-based designs. Further, they do not report the overall resource utilization of their presented designs. The work in [21] and [17] have considered Posits for storing the trained weights of ANN and then utilizing the FP32-based operations to compute output values.

Currently, the Posit numerical scheme's utilization in implementing accelerators for various applications is hampered by the unavailability of resource-optimized and energy-efficient Posit arithmetic units. In our proposed work, we aim to leverage the useful storage capability of Posit by modifying the Posit number representation to store numbers within the sub-normal region and the compute efficiency of FxP-based arithmetic by implementing a PoFx converter.



(a) Application (b) Architecture

Fig. 4. System model.

III. SYSTEM MODEL

A. APPLICATION MODEL

The hardware designs proposed in our current work can be used for any arbitrary application that needs to communicate and/or store a large number of parameters. However, in this article, we limit our exploration to ANNs. Fig. 4(a) shows one of the more widely used ANN—the VGG16 [4]—in research. As shown in the figure, VGG16 is composed of 16 layers of 4 different types—*convolutional*, *max pooling*, *fully connected* and *softmax*. Although we use the VGG16 as the application for evaluating our proposed methodology, the methods are applicable to any arbitrary ANN as most networks are composed of a subset of these types of layers. Fig. 4(a) also shows the dimension of the parameters that are used in each of the layers. Using accelerators for inference usually involves communicating and storing these large number of trained parameters—138 million for VGG16. Consequently, the quantization methods used for the parameters can influence the corresponding storage and communication overheads. Similarly, given the large number of MAC operations involved in the inference of a single input—15.5 billion for VGG16—the speed and power dissipation of the MAC unit determines the throughput and energy consumption of ANN inference.

B. ARCHITECTURE MODEL

Fig. 4(b) shows the architecture model used in this article. As shown in the figure, we assume an FPGA-based System-on-Chip (SoC) as the hardware platform. It contains an embedded processor along with reconfigurable logic similar to the Zynq EPP [37]. We assume that the accelerators for different types of layers of an ANN are executed on the reconfigurable logic and can implement the proposed hardware designs. For any accelerator, we assume that the parameters of the corresponding layer are fetched from the main memory through streaming interfaces with the on-chip AXI interconnect [38]. Similarly the input and output activations are transferred from and to the main memory using AXI streaming interfaces as well. Hardware platforms based on the Zynq EPP, such as the Ultra96-V2 [39], are being widely marketed as edge processing devices for Internet of Things (IoT).

IV. DESIGN METHODOLOGY

The top-level view of ExPAN(N)D is shown in Fig. 5. The *Hardware design* and characterization of the MAC units for various quantization schemes forms the central theme around which the other two methods—*Behavioral analysis* and *Accelerator design*—are implemented. *Behavioral analysis* enables the estimation of quantization-induced errors in a given ANN using the proposed hardware designs. Similarly, *Accelerator design* allows the designer to estimate the performance-resource trade-offs resulting from implementing various quantization schemes in an accelerator for a given layer of the ANN. The results from each of the three methods can be used to constraint the search space in the design of an efficient ANN using successive design space pruning. However, the implementation of an effective design space exploration (DSE) methodology is beyond the scope of this article.

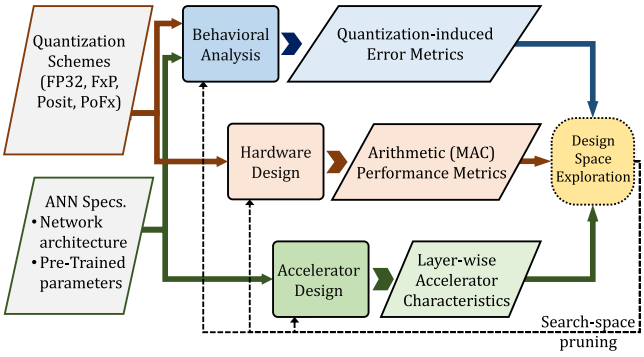


Fig. 5. Proposed design methodology.

A. HARDWARE DESIGN

1) NORMALIZED POSIT REPRESENTATION

The Posit representation is inherently designed to encode numbers in the range $(-\infty, \infty)$. However, due to their tapered accuracy, numbers near ± 1 have better accuracy in comparison to extremely small or large numbers [11]. The improved

dynamic range of Posit numbers also helps map weight normalized FP32 values better as illustrated in Section I. Thus, low-precision Posit numbers perform better than an equivalent linear fixed-point representation during the quantization of normalized ANN weights. While processing weight normalized numbers, sub-optimal utilization of all possible Posit bit-patterns leads to half of them being unused as all of the weight normalised values lie within the range $[-1, +1)$. This can translate to communication and storage overheads, as more than required bits are being transferred around. Similarly, a higher number of bits, than that required for storing the information, are processed during each computation. Hence, we propose *normalized Posit*—an alternative representation based on Posits which preserves its encoding efficiency, hardware realization and tapered accuracy while doubling the usable bit pattern values (x) within the normalized range ($-1 \leq x < +1$). This normalized Posit representation is a logical subset of Posits that is customized for the quantization and storage of weight normalized FP32 values.. For example, TABLE 2 shows all the possible bit-patterns and their equivalent real values for a Posit configuration of $N = 4$, $ES = 0$. The highlighted rows in the table show the bit-patterns which represent normalized numbers. It is evident that the two leading bits of the Posit representation are identical when the bit pattern denotes a normalized number; we leverage this finding to drop the leading Posit bit in our proposed normalized Posit representation.

This Posit representation helps us encode N-bit Posit functionality within the normalized range with $N - 1$ bits. This leads to a reduction in storage requirement while still being able to reuse existing Posit arithmetic hardware by replicating the leading bit near the processing unit. However, existing hardware implementations are not optimized to perform normalized Posit-only arithmetic. Existing implementations do not take complete advantage of the benefits arising as a consequence of the potentially unidirectional nature of bit shifts required to extract normalized Posits. Thus to leverage the aforementioned benefits we propose a novel parameterized Posit-to-FxP converter, *PoFx*. The optimized PoFx conversion hardware helps us use lower bit-width normalized Posit representation to effectively quantize and store weight normalized FP32 values in memory while providing FxP converted values near the processing elements to facilitate compute efficient ANN inference with minimal conversion overhead.

2) PoFx: NORMALIZED POSIT TO FIXED-POINT CONVERTER

Most Posit-based computations require a decode stage to extract the value before arithmetic operations as Posit bit-patterns cannot be directly operated upon. Currently, Posit-based arithmetic relies heavily on extraction of Posit numbers to a floating point like representation before operating on them, which leads to increased resource utilization. Instead, we design a novel resource-efficient parameterized PoFx converter which facilitates the use of existing resource-efficient FxP arithmetic optimizations.

TABLE 2. Posit(N = 4, ES = 0) to normalized Posit representation.

Posit	s	k	f	Value	ExPAN(N)D
0000	0	-3	0	0	**000**
0001	0	-2	0	0.25	**001**
0010	0	-1	0	0.5	**010**
0011	0	-1	0.5	0.75	**011**
0100	0	0	0	1	-
0101	0	0	0.5	1.5	-
0110	0	1	0	2	-
0111	0	2	0	4	-
1000	1	-3	0	NaR	-
1001	1	2	0	-4	-
1010	1	1	0	-2	-
1011	1	0	0.5	-1.5	-
1100	1	0	0	-1	**100**
1101	1	-1	0.5	-0.75	**101**
1110	1	-1	0	-0.5	**110**
1111	1	-2	0	-0.25	**111**

The proposed PoFx conversion algorithm is an intuitive technique which effectively converts a Posit to a FxP number developed from the way Posits are decoded. Taking the example of the Posit($N = 4, ES = 0$) bit-patterns in TABLE 2, we illustrate how we use minimal resources during conversion to a FxP format after Posit field-extraction by working at the bit level. The key to developing this algorithm rests on recognizing that the fraction field extracted from the Posit representation is identical to that required in the FxP output. Thus, once the data in the Posit bit pattern is extracted into its components s, k, e and f; the posit value only requires us to set a bit and store the extracted fraction bits to its right followed by a final bit shift determined by the equation $2^{ES} * k + e$. This equation can be implemented by adding the e value to the bit-sequence obtained by appending k to ES number of zero bits as illustrated in Fig. 6. The sign-bit along with the shifted bit sequence gives us the Posit representation in sign-magnitude FxP format.

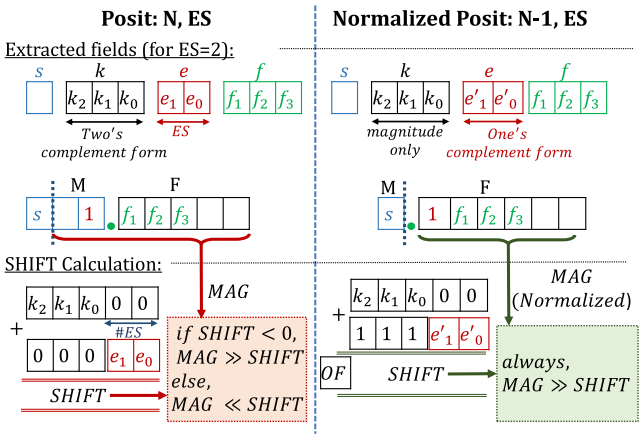


Fig. 6. Comparing shifting operations in Posit and normalized Posit representations.

PoFx conversion algorithm for manipulation at the bit-level which converts Posit representation, Posit(N, ES) to fixed-point representation FxP(M, F) where M is the total

FxP Output length and F is the fraction length in the FxP output is summarized in Algorithm 1. *Stage A* (comprising of *Stages A1*, *A2* and *A3*) stores the sign bit and prepares the Posit bits for subsequent extraction. *Stage B1* implements an optimized algorithm to evaluate the number of contiguous 1's. *Stage B2* performs bit manipulations to ascertain location of exponent and fraction bits and subsequently extracts them. All the loop indices are carefully evaluated based on the constraints arising from the Posit representation. *Stage C* performs the bit shift calculation and *Stage D* implements the bit shifts. The final *Stage E* is optional depending on the application and involves the conversion of sign-magnitude format to two's complement.

The proposed PoFx can be adapted to perform normalized PoFx conversion which leads to lower resource utilization and improved performance in ANNs. This is primarily due to the simplification of *Stage C* and *Stage D* in comparison to traditional implementations such as PACoGEN [14] as in this case the shifts are unidirectional, that is towards the right, making the value smaller. For normalized Posits we set $F = M - 1$ as all bits except for one sign bit would be used to store fraction bits since we can only store values in the normalized $[-1, 1)$ range. The first bit is replicated within *Stage A* followed by simplified extraction in *Stage B1* as the regime bit would always begin with zero thus K would store only magnitude. We use an optimized algorithm to evaluate the modified shift equation $2^{ES} * K - E$ in *Stage C* which is illustrated in Fig. 6. We store 1 after the assumed decimal point in normalized PoFx extraction and thus always need to right shift one time less. This is achieved implicitly by adding the one's complement of E to $2^{ES} * K$; further we will set the overflow flag (OF) if the required number of shifts exceeds the width of the *MAG* field. *Stage D* is replaced with a standalone right bit-shifter while *Stage E* remains unchanged.

The five stages in our proposed design can be pipelined to further improve the throughput of the PoFx converter as there are no feedback paths between the stages, thus eliminating data hazards. We note that though normalized Posit representation can represent the value -1, the normalized PoFx cannot extract the same due to its implicit storage in sign-magnitude format. For the rest of the article, the term PoFx will be used to denote the normalized PoFx. Similarly, Posit(N, ES) and Posit($N-1$, ES) will be used to denote Posit and normalized Posit respectively.

3) MAC UNIT WITH PoFx CONVERTER

The PoFx converter can be used for any application that can benefit from storing a large number of parameters efficiently. As a special case for ANNs, we integrate the normalized PoFx into MAC units to facilitate the use of our proposed optimizations for improving low-precision ANN inference. Fig. 7 shows the schematic of a parameterized PoFx converter based MAC along with *ReLU* activation function. As shown in the figure, the weights/biases are assumed to be stored/communicated as Posit($N-1$, ES) numbers. These values are then converted to their corresponding M-bit FxP

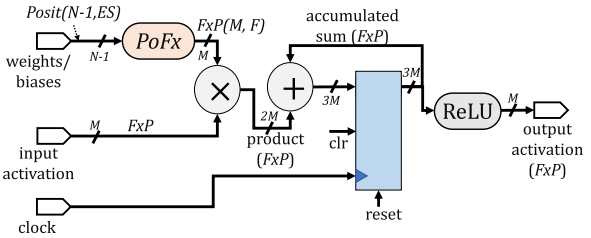


Fig. 7. MAC unit (with ReLU activation) using PoFx.

representations and multiplied with the M-bit input activation values. To accommodate the overflows resulting from the accumulation of a large number of $2M$-bit values and facilitate the evaluation of our architecture, we have chosen a $3M$-bit adder for accumulation across all configurations. After accumulating all the values, for a single node in a layer of an ANN, we pass the $3M$-bit result to the activation function.

It can be noted that the PoFx-based MAC unit allows the designer to represent the weights/biases with a fewer number of bits while still being able to implement different kinds of FxP-based arithmetic optimizations, such as precision-scaling, approximations, etc in the processing element. However, the effect of such a reduced bit representation on the ANN's behavior, and the corresponding reduction in the computer and communication/storage overheads of the associated accelerators for each layer needs to be estimated. The next two sub-sections provide the details of our contributions regarding these aspects of designing a PoFx-based ANN.

B. BEHAVIORAL ANALYSIS

To evaluate the impact of various quantization schemes on the output accuracy of a DNN, we have utilized TensorFlow for implementing a high-level behavioral framework, as shown in Fig. 8. It evaluates each quantization scheme's efficacy by analyzing its impact on (a) accuracy of the quantized parameters, (b) errors generated in the output activations of each layer due to quantized parameters, and (c) the accuracy of the final output of the quantized DNN compared to FP32-based output. The multi-level analysis of the quantization induced errors helps in the early elimination of the infeasible configurations. For this work, we have considered various configurations of the FxP-based linear quantization and Posit-based representations, denoted by the Quantization Schemes in Fig. 8. However, our proposed framework is generic and allows the integration of other types of quantization schemes. In this work, we aim to utilize the Posit scheme to decrease the storage and communication overheads and employ FxP-based arithmetic to reduce the overall computational complexity of the DNNs. Our proposed workflow performs a thorough analysis of the inter-conversions of these schemes to evaluate the impact of the available quantization step sizes and the dynamic ranges offered by each scheme on the final output accuracy of the DNN. For example, path ① converts a trained parameter from FP32 with comparatively higher precision into a low bit-width Posit scheme

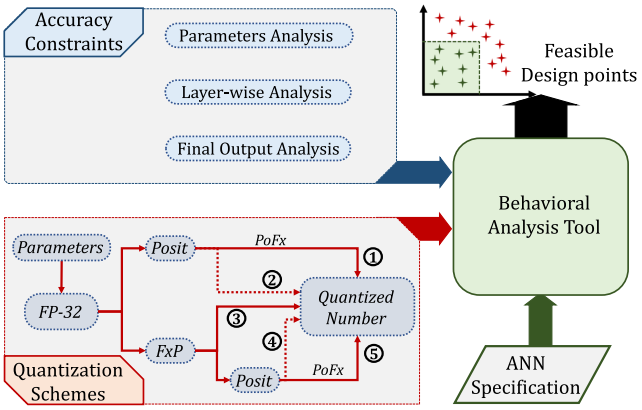


Fig. 8. Framework for behavioral analysis.

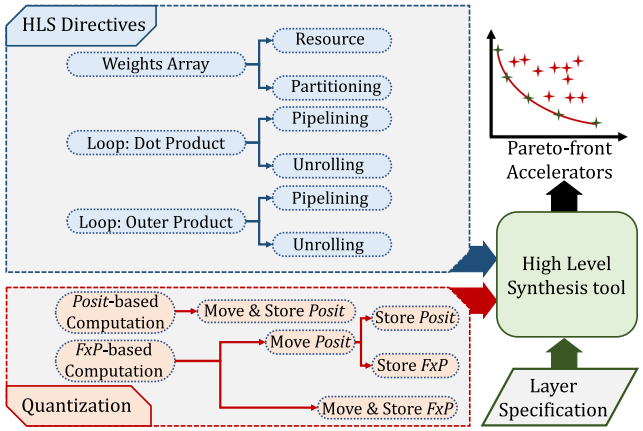


Fig. 9. HLS-based accelerator design.

(for storage) having lower-precision and then utilizes PoFx to perform FxP-based arithmetic. However, path ⑤ first quantizes the FP32 number using FxP-based representation (having lower-precision) and then utilizes a low bit-width Posit scheme to store the quantized numbers. The Posit-induced quantization errors in both paths will be different. Similarly, the Posit-based arithmetic using paths ② and ④ will have different quantization-induced errors. As shown by the classification accuracy results in Section V, the utilization of each of these schemes has a distinct impact on the final output accuracy. After providing the description of an ANN and the various quantization schemes, the proposed framework provides quantization configurations fulfilling the desired accuracy constraints. These selected configurations are then used by our proposed *Accelerator Design* tool flow to compute their respective performance metrics.

C. ACCELERATOR DESIGN

The HLS-based design flow, shown in Fig. 9, is used for evaluating the associated trade-offs between computation overhead and communication/storage gains offered by the PoFx-based MAC units. The design choices tree originating from HLS directives shows the various degrees of freedom (not exhaustive) associated with the design of an accelerator for a fully-connected layer. We assume a weight-stationary [40] design, where a set of weights for a subset of the artificial neurons in the layer are transferred once to the hardware accelerator. Subsequently, each input activation vector is transferred and the corresponding output activation of each neuron is computed. Therefore, the computation of each output activation vector can be seen as the multiplication of a matrix (*weights*) by a vector (*input activations*). Consequently, HLS directives of *pipelining* and *loop unrolling* can be applied to the computation of the *Dot Product* (evaluation of the output activation of each node) and the *Outer Product* (evaluation of all output activations) for obtaining designs with varying performance and resource utilization. Similarly, the type of resources allocated for the weights matrix, BRAMs or LUTRAMs, and the associated array-partitioning choices can affect the accelerator characteristics.

The design decisions associated with the quantization schemes are integrated into the HLS-based flow. The computation mode, Posit- or FxP-based, and the associated bit-widths impact the accelerator performance considerably. The proposed accelerator allows the designer to send and store the weights in Posit($N-1, ES$) or FxP format. If the weights are moved and stored as Posit ($N-1, ES$), the MAC units need to have the PoFx unit integrated into it (similar to Fig. 7). However, if the weights are moved as Posit($N-1, ES$) and stored as FxP (using PoFx), the MAC units do not require the run-time conversion during each computation. However, this approach increases the storage requirements compared to storing as Posit($N-1, ES$). It must be noted that the joint exploration across HLS directives and quantization schemes is necessary for a good estimation of accelerator characteristics. Performance improvement using HLS directives usually involves replicating compute and memory resources which are in turn dependent upon the choices related to the quantization schemes.

V. EXPERIMENTS AND RESULTS

A. EXPERIMENT SETUP

The proposed PoFx converter and the associated computer arithmetic blocks were implemented using Verilog HDL. Python-based scripts were used for automating the generation of the parameterized designs. SmallPosit HDL repository [41] was used for generating the Posit-based arithmetic designs [42]. The hardware designs were characterized using Xilinx Vivado Design Suite. For the calculation of the dynamic power of all implementations, Vivado Simulator and Power Analyzer tools have been utilized. All designs have been implemented on Xilinx Zynq UltraScale+ MPSoC (xczu3eg-sbva484-1-e device). The behavioral analysis was achieved using Python-based implementations and used TensorFlow [43] for estimation of various quantization induced error metrics. Xilinx Vivado HLS 18.3 was used as the High-level Synthesis tool for accelerator design. While the results for the behavioral analysis correspond to the

Algorithm 1 *Posit (N,ES) to FxP (M,F)*

Require: $N,\ ES,\ M,\ F$

▷ N: Input Posit Bit Length
▷ ES: Maximum Exponent Bit Length
▷ M: FxP Output Length
▷ F: Fraction length in FxP Output

► A1: Extract Sign Component to FxP Output

$S = POSIT\ [N-1]$
$MAG[F] = 1$ ▷ Set Leading Bit

► A2: Implement conditional Two's Complement

if $POSIT[N-1] == 1$ **then**
    $POSIT[N-2:0] = (!\ POSIT[N-2:0]) + 1$

► A3: Implement Modified Leading Zero Detector

if $POSIT[N-2] == 0$ **then**
    $P[N-2:0] =!\ POSIT[N-2:0]$

▷ To avoid LOD by inversion of bit sequence

$LZD[N-2] = P[N-2]$ ▷ Always 1
for $(i = N-3;\ i >= 0;\ i--)$ **do**
    $LZD[i] = LZD[i+1]\ \&\ P[i]$

► B1: Evaluate Regime Value

$V = \#\ 1's\ In\ LZD$
if $POSIT[N-2] == 0$ **then**
    $K = -V$
else
    $K = V - 1$

► B2: Extract Exponent and Fraction Fields

$E : [e_{ES-1}, \ldots, e_1, e_0] = \vec{0}$
for $(i = N-4;\ i >= 0;\ i--)$ **do**
    $EXT[i] =\ !(LZD[i+1]\ |\ LZD[i])$
$ST[N-4] = EXT[N-4]$
for $(i = N-5;\ i >= 0;\ i = i--)$ **do**
    $ST[i] = EXT[i+1] \oplus EXT[i]$

▷ To Generate Silhouette ST for Extraction

$switch = N - 4 - ES$
for $(i = 0;\ i <= N-4;\ i++)$ **do**
    $set = 0$
    for $(j = 0;\ j <= i;\ j++)$ **do**
        $set = set|(ST[N-4-i+j]\ \&\ POSIT[j])$
    if $i <= switch$ **then**
        $MAG[F-1-switch+i] = set$
    else
        $E[i-1-switch] = set$

► C: Shift Calculation

$SHIFT = 2^{ES} * K + E$

▷ $SHIFT$ register size $= \lceil log_2(M) \rceil$

► D: Bit Shift Implementation

MAG $\ll$ SHIFT ▷ -ve Value = Right Shift

► E: Sign Magnitude to Two's Complement Block

experiments using VGG16 as the test application, all the proposed methods can be used for any arbitrary application.

B. HARDWARE DESIGN

1) NORMALIZED PoFx

We analyze the impact of varying output bit-width (M) of PoFx converter on the overall performance of PoFx for a given configuration of Posit. Fig. 10 presents the results of the analysis for Posit ($N-1=5$, $ES=1$) configuration.[1] The variation in M, for a *fixed* Posit configuration, has an insignificant impact on the converter's CPD. For a specific value of $\lceil log_2(M) \rceil$, the overall LUT utilization also remains relatively unchanged.[2] For example, the total number of utilized LUTs by PoFx for $M=9$ is approximately 2.3 times the total number of utilized LUTs for $M=8$. The total number of utilized LUTs also directly affects the dynamic power consumption of the PoFx. The minor variations in the Power metric of the PoFx is a result of the optimizations performed by the synthesis tool. Compared to the resource utilization of traditional Posit-based arithmetic units (discussed in the following sections), the PoFx has an insignificant contribution to the overall resource utilization of FxP-based arithmetic units.

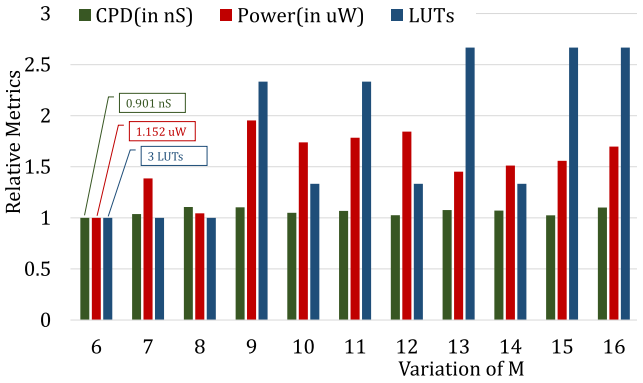


Fig. 10. Performance metrics of PoFx with varying *M* for Posit($N-1=5$, $ES=1$) for 2^N random input combinations.

Fig. 11 compares the impact of various Posit configurations, varying $N-1$ and ES, on the performance metrics of PoFx for a fixed bit-width (M) of the output. The critical path delay follows an increasing trend with an increase in the values of ES and $N-1$. This trend is primarily due to an increase in the logic required for Posit extraction due to increased variability in the individual field length. The designs with $ES=0$ have minimum resource utilization. The absence of the exponent field results in significant simplification of the overall extraction circuit. However, the designs with $ES \in \{2, 3\}$ have comparably higher resource utilization. A similar

[1] Similar results are obtained for other Posit configurations.
[2] As described in Algorithm 1, the $\lceil log_2(M) \rceil$ is used to calculate the size of the *shift* register for computing the corresponding FxP value.

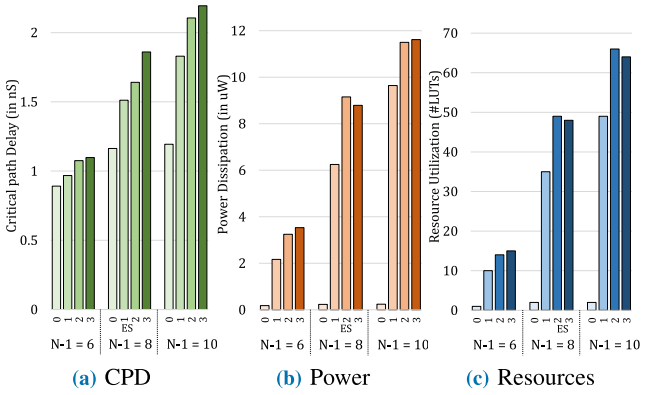


Fig. 11. Variation in the hardware performance metrics of PoFx with varying values of ES and $N-1$ for Posit($N-1, ES$) representation being converted to FxP($M=16, F=15$).

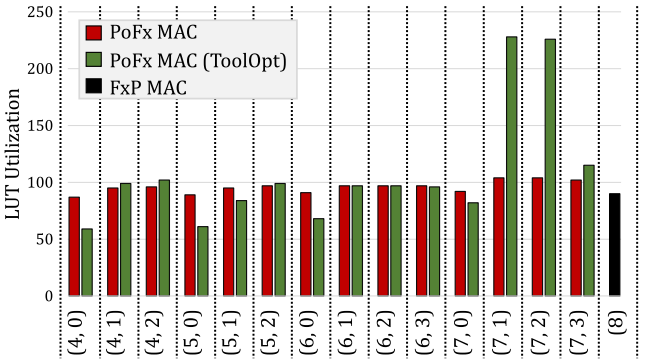


Fig. 12. Effect of synthesis tool's optimization across component blocks for PoFx-based 8-bit MAC implementations.

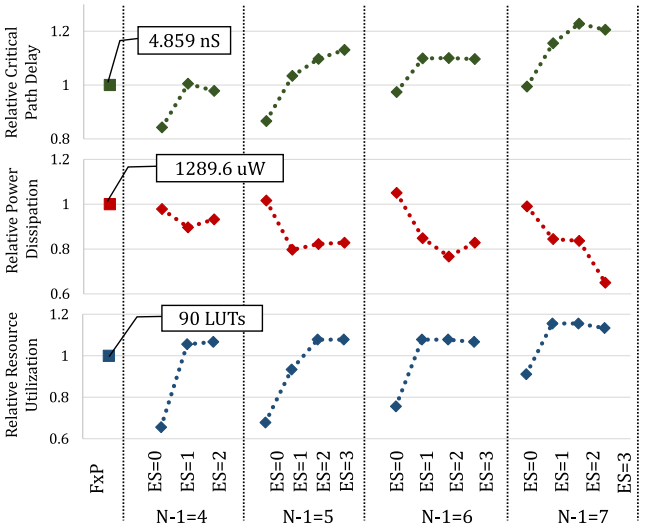


Fig. 13. Relative hardware performance metrics of PoFx-based MAC units with varying values of ES and $N-1$ for Posit($N-1, ES$) inputs to 8-bit FxP MAC.

trend is also observed for the dynamic power consumption of the PoFx for various Posit configurations.

2) MAC DESIGN ANALYSIS

The proposed PoFx allows the utilization of resource-efficient and high-performance FxP-based arithmetic units for Posit number systems. To evaluate the efficacy of the proposed approach and estimate the associated overheads of the PoFx, we compare PoFx-based 8-bit MAC units[3] with a traditional FxP-based MAC unit. Moreover, for a more thorough exploration of the PoFx-based designs, we have synthesized two types of designs—one that allows the synthesis tool to optimize across the constituent blocks (converters, multipliers, and adders) and the other that performs optimization for the constituent blocks separately. Fig. 12 shows the effect of the synthesis tool's optimization (*ToolOpt*) for all the PoFx-based 8-bit MAC designs. Such optimizations result in reduced resource utilization in many cases. In most other instances, the increase in LUT utilization is not significant. However, for PoFx($N-1=7, ES=1$) and PoFx($N-1=7, ES=2$), it results in more than 100% increase.

The results of comparisons across multiple design metrics for various configurations of Posit are presented in Fig. 13. It should be noted that the data shown in Fig. 13 (and Fig. 14) corresponds to the design with the better metrics among the ToolOpt and non-ToolOpt versions. The critical path delay and resource utilization of the MAC follow a gradually rising trend with both N and ES values. It can be noted that in a few cases, especially for $ES=0$, the PoFx-based MAC provides better performance across critical path delay, power dissipation, and LUT utilization than the FxP-only MAC. For $ES=0$, the Posit scheme's dynamic range is limited, and the PoFx does not utilize the complete dynamic range of the FxP. The limited number of unique FxP values, after conversion, allows the synthesis tool to optimize the overall design of PoFx-based MAC to improve the associated performance metrics. The power metrics do not follow a well-defined trend as they are generated based on the bit switches required to obtain the correct bit-sequence as the output. Compared to the FxP-only MAC, we report worst-case overheads of 22.8%, 5.0% and 15.5% for critical path delay, power dissipation, and LUT Utilization, respectively. Similar trends are observed in Fig. 14, which compares the same performance metrics for a 16-bit FxP MAC.

To further evaluate the efficacy of PoFx-based MAC design, we compare it with FxP-only MAC, Posit-only MAC, and Posit-based 3-input Fused Multiply Add (FMA) [42].

Fig. 15 and Fig. 16 show the comparison of the power-delay-product (PDP) and the LUT utilization of these designs for 8- and 16-bit designs, respectively. Posit-only MAC, which has been implemented by using a standalone N-bit Posit adder and N-bit Posit Multiplier, has significantly higher PDP and LUT utilization as a result of the extraction and packaging of Posits between stages. The Posit-based FMA, though optimized, requires more hardware resources for implementation. It can be observed that the PoFx-based

[3]As shown in Fig. 7, an M-bit FxP-based MAC includes a $M \times M$ multiplier and a $3M$-bit adder.

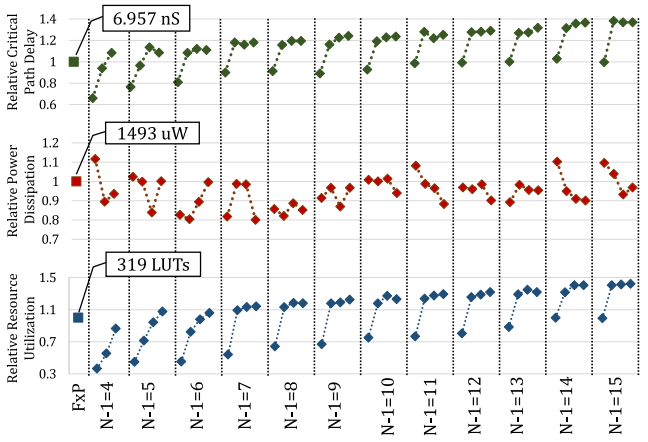


Fig. 14. **Relative hardware performance metrics of PoFx-based MAC units with varying values of *ES* and $N-1$ for Posit($N-1$, *ES*) inputs to 16-bit FxP MAC. *ES* $\in \{0, 1, 2, 3\}$ for all cases, except for $N-1=4$ where *ES* $\in \{0, 1, 2\}$.**

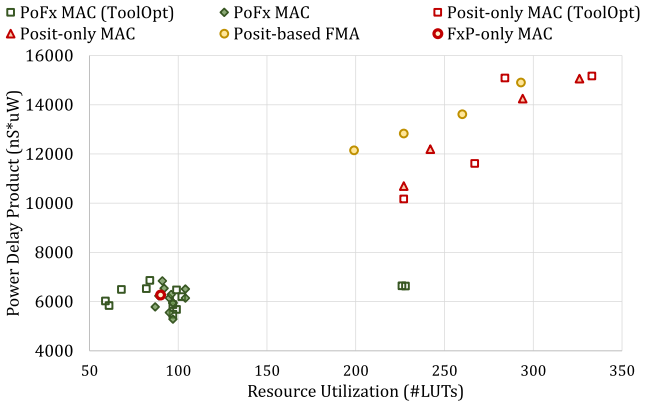


Fig. 15. **Comparison of various 8-bit MAC implementations: for Posit($N-1$, *ES*) $N-1 \in [4\ ..\ 7]$ *and* *ES* $\in \{0, 1, 2\}$.**

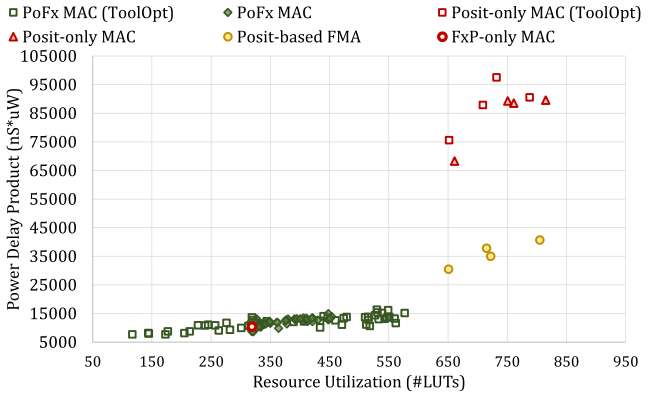


Fig. 16. **Comparison of various 16-bit MAC implementations: for Posit($N-1$, *ES*) $N-1 \in [4\ ..\ 15]$ *and* *ES* $\in \{0, 1, 2, 3\}$.**

MAC designs fall closely within the range of FxP-only MAC. Further, the Posit-only MAC and Posit-based FMA designs generate an N-bit output whereas, the proposed design generates a more precise $3N$-bit output once extracted. This can lead to lower inter-layer losses in ANNs as we can ascertain the type of rounding mechanism at the output based on the network to retain as much precision as possible before transferring the value to the next stage.

C. BEHAVIORAL ANALYSIS

We have considered DNNs as a test case to show the impact of various number representation schemes on the output accuracy of high-level applications. For this work, we have used a pre-trained VGG16 [4] network for the classification of the ImageNet dataset [44]. The VGG16 network mainly consists of 13 convolution layers and 3 fully connected layers. The very large number of the network's trained parameters, 138 million, makes it a sound candidate for evaluating efficiency of various quantization schemes. The single-precision FP32-based Top-1 and Top-5 percentage output classification accuracy of the 50000 validation images in the ImageNet dataset is 69.72% and 89.09%, respectively. Our proposed TensorFlow-based framework performs a multi-level analysis to identify possible quantization configurations fulfilling the output accuracy requirements of the network.

1) WEIGHTS QUANTIZATION ERROR ANALYSIS

In the first step, our framework quantizes the parameters (weights and biases) of all layers and filters out the configurations having large quantization-induced errors. For example, Fig. 17 shows the average absolute and the maximum quantization-induced errors in the weights of the Conv2_1 layer of the VGG16 network using different configurations of Posit and FxP schemes. The 8-bit FxP produces an average absolute error of 0.002. For smaller values of N, Posit schemes produce more errors than the FxP-based scheme in the quantized weights. However, for 7-bit and 8-bit Posit schemes, the average absolute errors are reduced to 0.002 and 0.001 only. We also evaluate the interconversions[4] of various schemes to identify feasible configurations for PoFx-based hardware. For example, the Posit($N-1=7, ES=2$)$\rightarrow$ 8-bit FxP scheme produces an average absolute error of 0.003, whereas the 8-bit FxP$\rightarrow$ Posit($N-1=7, ES=2$)$\rightarrow$ 8-bit FxP generates an average error of 0.002 only. Fig. 17 also reveals that Posit($N-1=3, ES=2$)-based configurations can be eliminated in the first step due to large quantization induced-errors. We have performed a similar analysis for all layers of the VGG-16 network by exploring all combinations of Posit(N, ES) where $N \in \{4, 5, 6, 7, 8\}$ and $ES \in \{0, 1, 2, 3\}$, and 8-bit FxP. The analysis identifies the quantization schemes producing the minimum average absolute error and the maximum absolute error for each layer of the network. For each N-bit Posit scheme, the quantized parameters are analyzed to identify the values of ES inducing minimum quantization errors.

In our current work we focus only on the quantization of weights and biases. The use of a specific quantized representation of the weights and biases will require the use of a compatible MAC design for inference. Hence, we performed

[4]As shown in Fig. 8

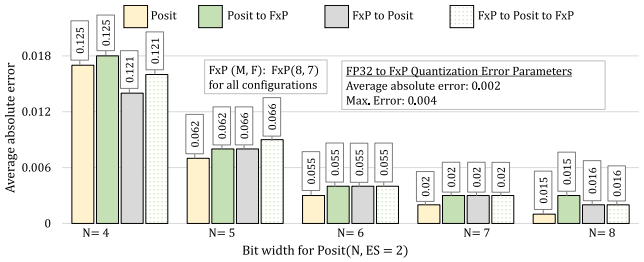


Fig. 17. Error analysis of various quantization schemes for Conv2_1 layer of pre-trained VGG16 [4]. *ES* values are kept 2 for all configurations. Maximum absolute quantization-induced error of each configuration is shown above the corresponding average relative error bars.

a joint analysis of the performance of the various MAC designs and the errors induced in the parameters by the corresponding quantization scheme. The various MAC designs are grouped under three categories – PoFx-based, Posit-based (that includes both multiply and adder combination and FMA-based designs) and FxP-based. For the PoFx-based and Posit-based designs, lower bit-width input designs were also considered. For example, for 8-bit quantization, N was varied from 5 to 8. Similarly, for 16-bit quantization, N was varied from 5 to 16. TABLE 3 shows the Pareto analysis results for 8- and 16-bit MACs with the three objectives – PDP, average quantization-induced error and the LUT utilization. We report the number of dominating points for each of the three types of quantization schemes used for the parameters of each layer of VGG16. As shown in the table, using PoFx-based designs contribute significantly to the number of points on the Pareto-front for 8-bit precision. We also report the percentage increase in the Pareto-front hypervolume due to the usage of PoFx-based designs over the collection of Posit and FxP-based designs only. As seen in the table, using PoFx-based designs we report up to 173% increase in the hypervolume for 8-bits precision. Fig. 18 shows the dominating and dominated points for each of the three categories in the corresponding design space for 8-bit precision MACs for the first layer (Conv1_1) of VGG16. It can be observed that the Posit- and FxP-based designs contribute one point each to the resulting Pareto-front, compared to 9 PoFx-based points.[5]

The improvements for 16-bit precision are lower compared to 8-bits. However, as shown in TABLE 4, if we also consider the bits-width of the parameters as a design objective in the analysis, we report consistent improvements using PoFx-based designs for both 8- and 16-bits precision. Since the number of input bits is an indicator of the communication power dissipation (and energy consumption) for moving weights, using PoFx-based quantization can result in reducing the overall power dissipation during DNN inference.

2) OUTPUT ACTIVATION ERROR ANALYSIS

In the second step of behavioral analysis, our framework utilizes the quantized parameters to evaluate each configuration's impact on the output activations of each layer.

[5]Since we have used a 2D plot for showing the pareto-front for 3 objectives, some dominating points appear as dominated in Fig. 18.

TABLE 3. Pareto analysis of MAC designs with weights quantization error. Objectives: PDP, Average Error, #LUTs.

VGG16 Layer	Number of points on Pareto front						% Improvement in hypevolume due to PoFx-based MACs	
MAC Type	PoFx-based		Posit-based		FxP-based			
Max Bits	8	16	8	16	8	16	8	16
conv1_1	9	7	1	5	1	0	173	74
conv1_2	8	4	3	11	1	0	125	2
conv2_1	8	4	3	9	1	0	121	2
conv2_2	8	3	2	8	1	0	119	1
conv3_1	8	3	2	8	1	0	115	1
conv3_2	8	1	4	10	1	0	109	0
conv3_3	8	1	4	10	1	0	109	0
conv4_1	8	1	6	8	1	0	104	0
conv4_2	7	1	6	8	1	0	97	0
conv4_3	7	1	6	8	1	0	98	0
conv5_1	7	1	6	8	1	0	102	0
conv5_2	8	1	6	8	1	0	102	0
conv5_3	7	1	6	8	1	0	102	0
fc6	7	0	5	8	1	0	67	0
fc7	7	0	5	8	1	0	86	0
fc8	8	1	6	8	1	0	104	0

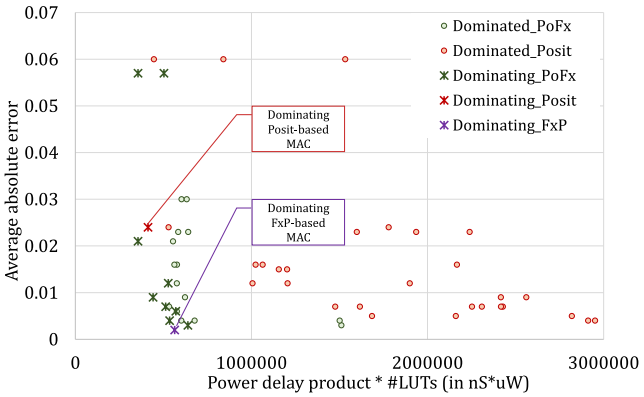


Fig. 18. Pareto analysis of 8-bit MAC design along with errors induced in quantization of weights for Conv1_1 layer of pre-trained VGG16.

TABLE 4. Pareto analysis of MAC designs with weights quantization error. Objectives: CPD, Power, Average Error, #LUTs, #Bit-width of parameters.

VGG16 Layer	Number of points on Pareto front						% Improvement in hypevolume due to PoFx-based MACs	
MAC Type	PoFx-based		Posit-based		FxP-based			
Max Bits	8	16	8	16	8	16	8	16
conv1_1	21	31	10	41	0	0	40	74
conv1_2	20	32	11	23	0	0	27	50
conv2_1	20	31	10	23	0	0	27	48
conv2_2	17	30	11	18	0	0	27	48
conv3_1	17	29	10	17	0	0	27	47
conv3_2	17	26	12	18	0	0	27	46
conv3_3	17	26	12	18	0	0	26	46
conv4_1	17	26	10	16	0	0	27	46
conv4_2	17	26	10	16	0	0	27	46
conv4_3	17	26	10	16	0	0	27	46
conv5_1	17	26	10	16	0	0	27	46
conv5_2	17	26	10	16	0	0	27	46
conv5_3	17	26	10	16	0	0	27	46
fc6	17	25	11	15	0	0	28	46
fc7	17	26	11	15	0	0	27	46
fc8	17	26	10	16	0	0	27	46

The computation of the output activation involves using a MAC design that is compatible with the chosen quantization scheme. Similar to the analysis presented in Fig. 18 for the errors induced in the parameters, Fig. 19 shows the design space while considering the errors in the output activations for the first layer— Conv1_1— of VGG16. The 3D scatter plot

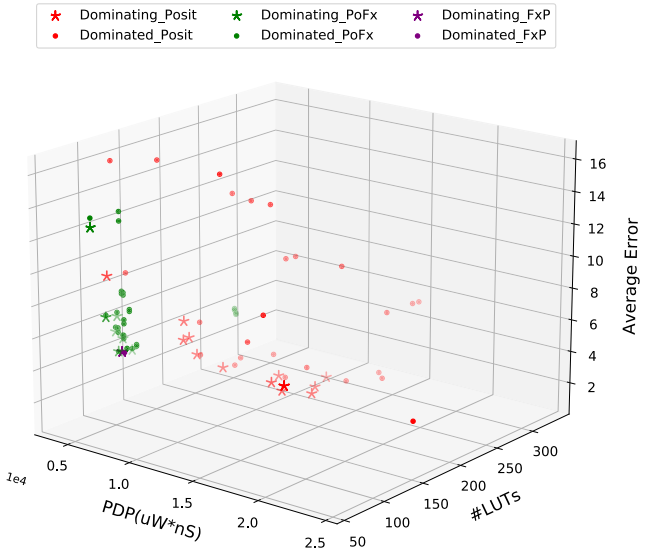


Fig. 19. Pareto analysis of 8-bit MAC design along with errors induced in output activations for Conv1_1 layer of pre-trained VGG16.

shows the various design points corresponding to the three categories of MAC designs, PoFx-, Posit- and FxP-based. It can be observed from Fig. 19 that the PoFx and FxP-based designs' contribution to the Pareto-front is mainly due to better hardware performance— lower PDP and reduced number of utilized LUTs. Similarly, Posit-based designs' contribution is mainly due to lower average error, albeit at high hardware costs. The resulting Pareto-front in Fig. 19 has 7, 13 and 1 points from PoFx, Posit and FxP-based designs respectively, with 12.4% improvement in the hypervolume over the collection of only Posit- and FxP-based designs. It must be noted that since we focus on the quantization schemes for only the parameters, during the behavioral analysis, the input activations for each of the layers are kept at FP32 precision. After computing the output activations, they are quantized using the configuration employed to quantize the respective parameters. This lets us evaluate the impact of the proposed methods and designs while other aspects are kept unchanged.

3) CLASSIFICATION ERROR ANALYSIS

Finally, the behavioral analysis involves estimating the impact of the proposed methods on the classification accuracy. TABLE 5 shows the percentage Top-1 and Top-5 classification accuracies of the ImageNet validation dataset [44] using different quantization schemes. For this experiment, the activations have FP32 precision, and the parameters (weights and biases) are quantized using various 8-bit schemes. For comparison, we also show the classification accuracy using 7-bit and 16-bit FxP-based quantization techniques. The FxP-16 and Posit($N = 8, ES = 2$) produce similar classification results by reducing the final output accuracy by only 0.06 and 0.07, respectively when compared with FP32-based results. The FxP-8 based configuration reduces the Top-1 and Top-5 classification accuracy by 5.01 and 2.83, respectively. However, the FxP-7-based quantization

TABLE 5. Classification accuracy of VGG-16 network [4] on ImageNet dataset [44] with quantization of weights and biases using different schemes of fixed-point, Posit and PoFx.

Configuration		Configuration Parameters		Top-1 [%]	Top-5 [%]
		N	ES		
FP32		-	-	69.72	89.09
FxP-16		-	-	69.66	89.02
FxP-8		-	-	64.71	86.26
FxP-7		-	-	10.94	26.10
Posit		7	1	68.88	88.50
Posit		8	1	69.59	89.00
Posit		6	2	66.32	86.99
Posit		7	2	68.77	88.54
Posit		8	2	69.65	89.00
Posit		6	3	64.86	86.04
Posit		7	3	68.02	87.97
Posit		8	3	69.43	88.86
PoFx (N-1, ES)	Posit_FxP	6	1	46.05	71.12
	Posit_FxP	7	1	11.13	26.08
	Posit_FxP	5	2	43.59	69.08
	Posit_FxP	6	2	11.96	27.22
	Posit_FxP	7	2	1.92	6.31
	Posit_FxP	5	3	41.37	66.99
	Posit_FxP	6	3	11.67	26.84
	Posit_FxP	7	3	1.79	6.11
PoFx (N-1, ES)	FxP_Posit_FxP	6	1	64.38	85.94
	FxP_Posit_FxP	7	1	64.48	86.15
	FxP_Posit_FxP	5	2	58.27	81.99
	FxP_Posit_FxP	6	2	64.36	85.99
	FxP_Posit_FxP	7	2	64.40	86.08
	FxP_Posit_FxP	5	3	57.13	81.13
	FxP_Posit_FxP	6	3	62.67	84.62
	FxP_Posit_FxP	7	3	64.45	86.15

significantly drops the final classification accuracy. For the PoFx-based schemes, we consider the normalized PoFx technique and utilize Posit(N-1, ES) configurations for N-bit Posit numbers. TABLE 5 reveals that the direct conversion of Posit numbers to FxP scheme (Posit-FxP) significantly diminishes the final output accuracy. However, utilizing FxP→ Posit→ FxP based conversion, the PoFx has an insignificant impact on the final output. For example, compared to the FxP-8 based results, the FxP-8→ Posit(N-1 = 6, ES = 2)→ FxP-8 decreases the Top-1 and Top-5 classification accuracy by only 0.35 and 0.26.

TABLE 6 shows the joint analysis of the ImageNet dataset classification accuracy and the corresponding MAC designs for a subset of the configurations. It contains only those configurations from TABLE 5 that have comparable accuracy and having feasible hardware designs. For instance, arithmetic blocks for Posit($N = 6, ES = 3$) could not be generated using SmallPosit HDL [41]. Similarly, as shown in TABLE 5 the Posit-FxP modes have much lower accuracy than similar configurations for FxP→ Posit→ FxP, while requiring the same PoFx-based MAC, and are hence omitted from the analysis. The PDP and LUT utilization values for each configuration in TABLE 6 are obtained from the lowest PDP design for that configuration. The PDP and LUT metrics shown in the table correspond to values relative to the maximum shown in the table's top row.

The highest value of PDP and LUT utilization occurs for the configurations Posit($N = 8, ES = 1$) and FxP-16 respectively. The highest and lowest values of the performance

TABLE 6. Joint analysis of classification accuracy and MAC hardware characteristics of fixed-point, Posit and PoFx-based designs.

Configuration	N	ES	Top-1 [%]	Top-5 [%]	Relative MAC Metrics	
					PDP [Maximum: 13616 uW*nS]	LUTs [Maximum: 319]
FxP	16	-	69.66	89.02	0.763	1.000
	8	-	64.71	86.26	0.475	0.282
Posit (N,ES)	7	1	68.88	88.5	0.578	0.671
	8	1	69.59	**89**	**1.000**	0.815
	6	2	**66.32**	**86.99**	**0.441**	**0.555**
	7	2	68.77	88.54	0.550	0.618
	8	2	**69.65**	**89**	0.853	**0.837**
	7	3	68.02	87.97	0.469	0.567
	8	3	69.43	88.86	0.747	0.712
PoFx (N-1,ES)	6	1	64.38	85.94	0.432	**0.304**
	7	1	**64.48**	**86.15**	0.451	0.326
	5	2	58.27	81.99	0.417	0.310
	6	2	64.36	85.99	**0.388**	**0.304**
	7	2	64.4	86.08	**0.478**	0.326
	5	3	**57.13**	**81.13**	0.446	**0.304**
	6	3	62.67	84.62	0.418	**0.304**
	7	3	64.45	86.15	0.413	**0.361**

metrics for each of the two categories – Posit and PoFx are highlighted in bold text in TABLE 6. It can be observed that the Posit configuration for the highest Top-1 accuracy, Posit($N = 8, ES = 2$), corresponds to the MAC design with highest LUT utilization. Similarly the Posit configuration with highest Top-5 accuracy, Posit($N = 8, ES = 1$) (and Posit($N = 8, ES = 2$)), corresponds to highest (and relatively *higher*) PDP value. The Posit configuration with the lowest accuracy, Posit($N = 6, ES = 2$) corresponds to the design with lowest PDP and LUT utilization among Posit-based MACs.

Similar correlations were also observed in the case of PoFx-based designs. Designs with higher PDP usually result in better accuracy. Compared to FxP-8 based designs the PoFx($N - 1 = 7, ES = 1$) achieves similar accuracy with lower PDP (≈5%) and slightly higher LUT overhead (≈15.5%). Similarly, PoFx($N - 1 = 6, ES = 2$) achieves comparable accuracy with even lower PDP (≈18%) and less LUT overheads (≈8%). Additionally, these PoFx-based designs requires less bits for representing the parameters of a network. This can result in lower communication and storage overheads in the accelerator design for each layer of the network.

D. ACCELERATOR-LEVEL DESIGN ANALYSIS

The advantages of using the PoFx-based arithmetic operators can be seen clearly in the design of accelerators. As we shall see in the experiment results, the proposed PoFx approach results in large reductions in the computing overheads with very little cost to accuracy as compared to Posit- and FxP-based accelerators. In order to estimate the system-level impact of using the proposed PoFx methodology, we integrated the candidate solutions in the design of an accelerator for a fully-connected layer of a DNN. The accelerator was designed using C++ and synthesized using Xilinx's Vivado HLS. To keep the design generic, we implemented a matrix-vector multiplication. The matrix represents the weights of a fully-connected layer, while the vector represents a single input activation. One thousand input activations were used to estimate the switching activity in order to compute the power dissipation. The implemented accelerator uses ReLU activation function.

1) ACCELERATOR RESOURCE REQUIREMENTS

As was shown in Fig. 9, the accelerator design using HLS involves using various micro-architectural optimizations to generate designs with power-performance-area trade-offs. To provide a fair comparison, we used the same microarchitecture design choices for the Posit-, PoFx- and FxP-based design variants. We used loop unrolling for the inner product (dot product) and the outer loop of the matrix multiplication. Further, we employed LUTRAMs for storing the local arrays, with adequate partitioning to support parallel execution obtained by loop unrolling. In order to compare the effect of using Posit-based, PoFx-based and FxP-based MAC units, we implemented the following four variants of the accelerator:

1) *Posit*: The accelerator stores and computes all operations in Posit(N, ES) format.
2) *PoFx(Move)*: The weights are moved to the accelerator in normalized PoFx($N - 1, ES$) representation, converted to FxP and stored as FxP($M = 8$) numbers. During computations, the FxP($M = 8$) weights are fetched from local memory and used directly for arithmetic.
3) *PoFx(Move & Store)*: The weights are moved from main memory and stored in local memory in normalized PoFx($N - 1, ES$) format. During computation, the weights are fetched from local memory, converted to FxP($M = 8$) and used in the computation of the output activation values.
4) *FxP(8)*: The weights are moved from main memory to accelerator and stored in the local memory of the accelerator as FxP($M = 8$) numbers. Similar to PoFx(Move), the computation stage does not involve any conversions between number representations.

Fig. 20 shows the accelerators' relative resource requirements for the implementation of the four designs with varying configurations of Posit(N, ES) and PoFx($N - 1, ES$) for $ES = 0$. The accelerators designed for Fig. 20 correspond to a weight matrix of size 64×10. It can be observed that the LUT utilization of Posit is much higher than both FxP(8) and PoFx-based designs in all cases. This can be attributed to the high resource costs of the Posit arithmetic blocks. Similarly the RegFF utilization of PoFx(Move & Store) is lower than that of PoFx(Move) designs for all cases. Additionally, lower LUTRAM utilization is observed in PoFx(Move & Store) than Posit-based and FxP(8) designs in most cases. For instance, compared to the Posit($N = 7, ES = 0$) and FxP(8), we report ≈46% reduction in LUTRAMs utilization with the PoFx($N - 1 = 6, ES = 0$) design. Therefore, the proposed PoFx-based designs results in reduction in the accelerator's overall resource consumption. Fig. 21 shows

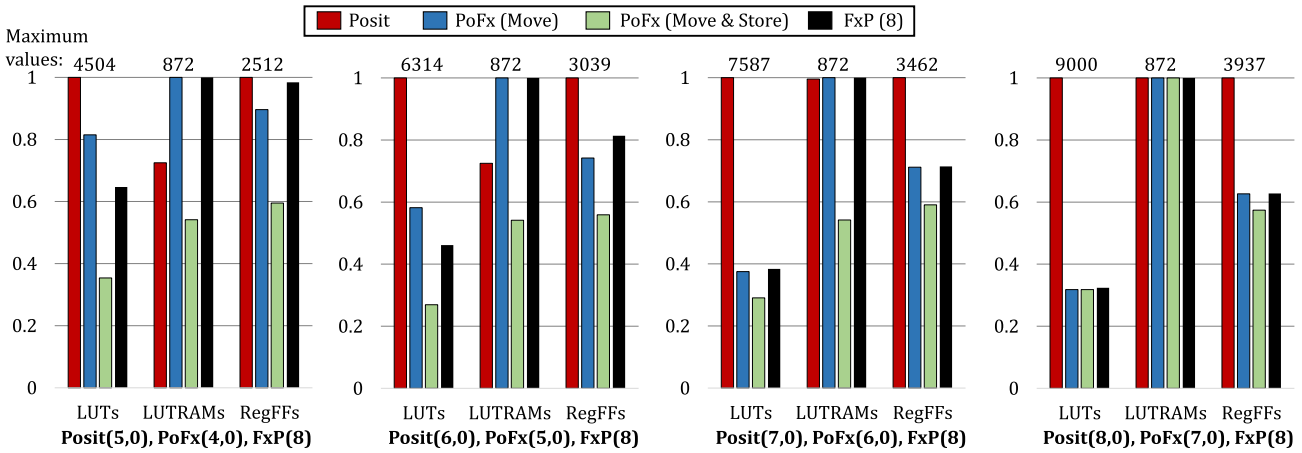


Fig. 20. Variation in the relative resource utilization of LUTRAM-based accelerator implemented with varying Posit and PoFx designs compared to FxP8-based designs.

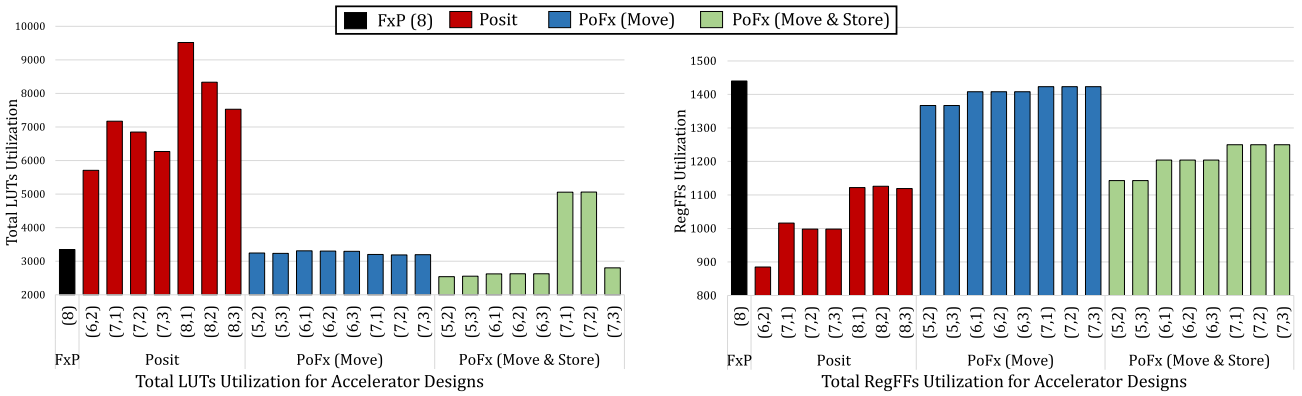


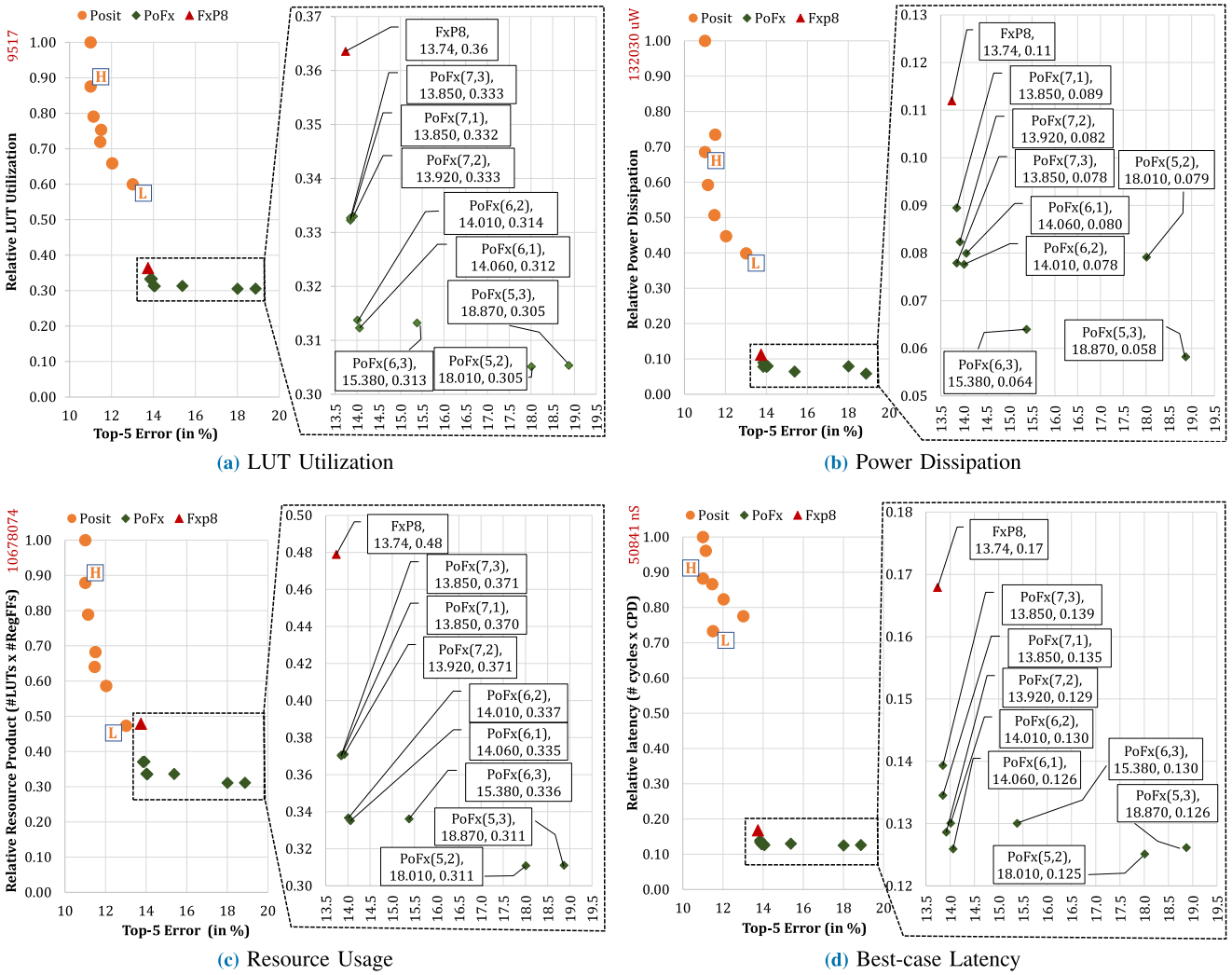
Fig. 21. Variation in the relative resource utilization of LUTRAM-based accelerator implemented with varying Posit and PoFx designs compared to FxP8-based designs.

the resource utilization for the accelerators corresponding to the configurations shown in TABLE 6. The PoFx-based accelerator designs show considerable lower LUT utilization than Posit-only designs. The high LUT utilization for two configurations can be attributed to the high LUT utilization of the PoFx-based MAC units (Fig. 12) reported by using the synthesis tools cross-optimization. The Posit-only designs report lower RefFF utilization than both FxP and PoFx-based designs.

2) ANALYZING PERFORMANCE-ACCURACY TRADE-OFFS

To demonstrate the effectiveness of the proposed PoFx-based designs, Fig. 22 plots the ImageNet dataset classification accuracy using VGG-16 network for FxP8, Posit, and PoFx(More & Store) along with various performance metrics of an accelerator implementing those designs. The accelerators designed for Fig. 22 correspond to a weight matrix of size 32×10. Each sub-figure in Fig. 22 shows the plot with all the designs on the left and a zoomed-in plot to compare with FxP8-based and PoFx-based designs. The design points shown in the plot correspond to the configurations shown in TABLE 6 (except Fxp-16). The horizontal axis of the plots shows the Top-5 classification error (in %) for the ImageNet dataset and the vertical axis slows the relative performance metric. The maximum value each of the performance metrics (corresponding to 1.00) is shown in red along the vertical axis. As can be seen across all the sub-figures, the PoFx- and FxP8-based accelerator designs show considerably better performance (lower values on the vertical axis) compared to Posit-based designs. This improved performance is obtained at the cost of slightly higher classification error.

Fig. 22(a) shows the impact of using fixed-point operators with reduced computational complexity on the accelerators' resource utilization (LUTs). The dominating (Pareto) Posit-based designs with the highest and lowest LUT utilization are highlighted in the figure as **H** (Posit(8, 2)) and **L** (Posit(6, 2)), respectively. As seen in the figure, the FxP8-based design results in around 2.74% and 0.73% more error than **H** and **L** designs, respectively. However, the FxP8-based design results in 4874 and 2248 less LUT usage than **H** and **L** implementations, respectively. If we consider the PoFx-based designs in the zoomed-in portion, the PoFx(7, 1)-based



(a) LUT Utilization (b) Power Dissipation

(c) Resource Usage (d) Best-case Latency

Fig. 22. Top-5 percentage errors in ImagNet dataset classification using VGG-16 v/s the performance of a sample accelerator implementing a fully-connected layer. The PoFx-based designs correspond to the PoFx(*Move & Store*) design variants.

design has an additional 0.11% error but uses 298 fewer LUTs compared to FxP8. Similarly, PoFx(6, 1)-based design adds only 0.32% additional error but uses 488 fewer LUTs than FxP8-based implementation. The other PoFx-based design points provide different error-area trade-offs. The lower LUT utilization of these PoFx-based design points, compared to the FxP8-based implementation, can be attributed to reduced storage requirements that provide resource utilization benefits in addition to amortizing the conversion overheads of each PoFx-based MAC unit.

The benefits of using PoFx-based designs in terms of power dissipation are reported in Fig. 22(b). The dominating Posit-based points with the highest and lowest power dissipation are shown as **H** (Posit(8, 2)) and **L** (Posit(6, 2)) respectively. The FxP8-based design shows nearly 75.71mW and 37.85mW lower power than **H** and **L** designs, respectively. The lower power dissipation is at the cost of 2.74% and 0.73% higher classification error. The PoFx-based designs report even further lower power dissipation. Designs using PoFx(7, 3), PoFx(6, 3) and PoFx(5, 3) report 4.49mW, 6.33mW and 7.09mW lower power than FxP8 with 0.11%, 1.64% and 5.13% higher error respectively. The higher power dissipation of the Posit-based MAC units gets exacerbated in the accelerator, with routing power accounting for a considerable portion of the total power dissipation.

Similar to LUT utilization and power dissipation, Fig. 22(c) and Fig. 22(d) show the accelerator's total resource utilization and *best-case*[6] latency, respectively, for various Posit and PoFx-based designs. The dominating Posit-based designs with the highest and lowest accelerator performance metrics are shown as **H** and **L**, respectively. For resource utilization, **H** and **L** correspond to Posit(8, 2) and Posit(6, 2) respectively. Similarly, for best-case latency, the points marked **H** and **L** refer to Posit(8, 1) and Posit(7, 1) respectively. Similar to Fig. 22(a) and Fig. 22(b), the FxP8-based design shows better performance than Posit-based designs with a slight reduction in the classification accuracy.

[6]The best-case latency refers to the latency corresponding to the CPD of the design.

The PoFx-based designs provide further design points that provide novel accuracy-performance trade-offs. The lower latency of FxP8- and PoFx-based designs can be attributed to their much lower CPD than Posit-based designs.

VI. CONCLUSION

To implement machine learning applications on resource- and energy-constrained embedded systems with limited computational power, it is imperative to consider the unique features of various optimization techniques together. This paper proposes the *ExPAN(N)D* framework for analyzing and combining the number representation efficacy of the Posit scheme and the resource- and compute-efficiency of FxP-based schemes. ExPAN(N)D utilizes a modified and novel representation of Posit numbers systems to represent the trained parameters of DNNs. Using the proposed scheme, we use $N-1$ bits for an N-bit Posit configuration to reduce the storage requirements. For performing arithmetic operations on trained parameters, stored in Posit format, ExPAN(N)D proposes and utilizes a resource-efficient Posit to FxP converter PoFx. Using PoFx, all arithmetic operations are performed using FxP-based arithmetic operators.

Compared to the lowest power consuming Posit-based accelerator implementation, Posit(6, 2), our proposed PoFx(6, 2)-based accelerator design results in 80% lower power dissipation with an additional 1.05% additional classification error. Compared to FxP8-based design, the PoFx(6, 2) design results in 27% lower power dissipation at the cost of 0.32% additional classification error. Similarly, the PoFx(6, 2)-based accelerator implementation results in 13% and 48% lower LUT utilization compared to FxP8- and Posit(6, 2)-based designs. ExPAN(N)D utilizes a TensorFlow-based behavioral framework to evaluate the impact of different quantization configurations on the final output accuracy of ANNs. We intend to extend the proposed framework by incorporating other networks' optimization techniques such as approximate arithmetic operators and various other quantization schemes.

REFERENCES

[1] T. Young, D. Hazarika, S. Poria, and E. Cambria, "Recent trends in deep learning based natural language processing," *CoRR*, vol. abs/1708.02709, pp. 1–32, Aug. 2017. [Online]. Available: http://arxiv.org/abs/1708.02709

[2] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification," in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2015, pp. 1026–1034.

[3] L. Deng, G. Hinton, and B. Kingsbury, "New types of deep neural network learning for speech recognition and related applications: An overview," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, May 2013, pp. 8599–8603.

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

[5] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding," 2015, *arXiv:1510.00149*. [Online]. Available: http://arxiv.org/abs/1510.00149

[6] B. Liu, M. Wang, H. Foroosh, M. Tappen, and M. Penksy, "Sparse convolutional neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 806–814.

[7] N. Burgess, J. Milanovic, N. Stephens, K. Monachopoulos, and D. Mansell, "Bfloat16 processing for neural networks," in *Proc. IEEE 26th Symp. Comput. Arithmetic (ARITH)*, Jun. 2019, pp. 88–91.

[8] H. F. Langroudi, Z. Carmichael, D. Pastuch, and D. Kudithipudi, "Cheetah: Mixed low-precision hardware & software co-design framework for DNNs on the edge," 2019, *arXiv:1908.02386*. [Online]. Available: https://arxiv.org/abs/1908.02386

[9] D. D. Lin, S. S. Talathi, and V. S. Annapureddy, "Fixed point quantization of deep convolutional networks," in *Proc. 33rd Int. Conf. Int. Conf. Mach. Learn.*, vol. 48, Jun. 2016, pp. 2849–2858.

[10] S. Ullah, H. Schmidl, S. S. Sahoo, S. Rehman, and A. Kumar, "Area-optimized accurate and approximate softcore signed multiplier architectures," *IEEE Trans. Comput.*, vol. 70, no. 3, pp. 384–392, Mar. 2021.

[11] J. L. Gustafson and I. T. Yonemoto, "Beating floating point at its own game: Posit arithmetic," *Supercomput. Frontiers Innov.*, vol. 4, no. 2, pp. 71–86, Jun. 2017, doi: 10.14529/jsfi170206.

[12] R. Chaurasiya, J. Gustafson, R. Shrestha, J. Neudorfer, S. Nambiar, K. Niyogi, F. Merchant, and R. Leupers, "Parameterized posit arithmetic hardware generator," in *Proc. IEEE 36th Int. Conf. Comput. Design (ICCD)*, Oct. 2018, pp. 334–341.

[13] M. K. Jaiswal and H. K.-H. So, "Universal number posit arithmetic generator on FPGA," in *Proc. Design, Automat. Test Eur. Conf. Exhib. (DATE)*, Mar. 2018, pp. 1159–1162.

[14] M. K. Jaiswal and H. K.-H. So, "PACoGen: A hardware posit arithmetic core generator," *IEEE Access*, vol. 7, pp. 74586–74601, 2019.

[15] A. Podobas and S. Matsuoka, "Hardware implementation of POSITs and their application in FPGAs," in *Proc. IEEE Int. Parallel Distrib. Process. Symp. Workshops (IPDPSW)*, May 2018, pp. 138–145.

[16] Z. Carmichael, H. F. Langroudi, C. Khazanov, J. Lillie, J. L. Gustafson, and D. Kudithipudi, "Deep positron: A deep neural network using the posit number system," in *Proc. Design, Automat. Test Eur. Conf. Exhib. (DATE)*, Mar. 2019, pp. 1421–1426.

[17] H. F. Langroudi, V. Karia, J. L. Gustafson, and D. Kudithipudi, "Adaptive posit: Parameter aware numerical format for deep learning inference on the edge," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2020, pp. 3123–3131.

[18] M. Cococcioni, F. Rossi, E. Ruffaldi, and S. Saponara, "Fast deep neural networks for image processing using posits and ARM scalable vector extension," *J. Real-Time Image Process.*, vol. 17, no. 3, pp. 759–771, Jun. 2020.

[19] R. Murillo, A. A. Del Barrio, and G. Botella, "Deep PeNSieve: A deep learning framework based on the posit number system," *Digit. Signal Process.*, vol. 102, Jul. 2020, Art. no. 102762.

[20] H. F. Langroudi, Z. Carmichael, J. L. Gustafson, and D. Kudithipudi, "PositNN framework: Tapered precision deep learning inference for the edge," in *Proc. IEEE Space Comput. Conf. (SCC)*, Jul. 2019, pp. 53–59.

[21] S. H. F. Langroudi, T. Pandit, and D. Kudithipudi, "Deep learning inference on embedded devices: Fixed-point vs posit," in *Proc. 1st Workshop Energy Efficient Mach. Learn. Cognit. Comput. Embedded Appl. (EMC)*, Mar. 2018, pp. 19–23.

[22] H. Zhang, J. He, and S.-B. Ko, "Efficient posit multiply-accumulate unit generator for deep learning applications," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2019, pp. 1–5.

[23] R. Jain, N. Sharma, F. Merchant, S. Patkar, and R. Leupers, "CLARINET: A RISC-V based framework for posit arithmetic empiricism," 2020, *arXiv:2006.00364*. [Online]. Available: https://arxiv.org/abs/2006.00364

[24] S. Zhou, Z. Ni, X. Zhou, H. Wen, Y. Wu, and Y. Zou, "DoReFa-Net: Training low bitwidth convolutional neural networks with low bitwidth gradients," *CoRR*, vol. abs/1606.06160, pp. 1–13, Jun. 2016. [Online]. Available: http://arxiv.org/abs/1606.06160

[25] P. Gysel, J. Pimentel, M. Motamedi, and S. Ghiasi, "Ristretto: A framework for empirical study of resource-efficient inference in convolutional neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 11, pp. 5784–5789, Nov. 2018.

[26] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "XNOR-Net: ImageNet classification using binary convolutional neural networks," *CoRR*, vol. abs/1603.05279, pp. 1–17, Aug. 2016. [Online]. Available: http://arxiv.org/abs/1603.05279

[27] M. Courbariaux, Y. Bengio, and J. David, "BinaryConnect: Training deep neural networks with binary weights during propagations," *CoRR*, vol. abs/1511.00363, pp. 1–9, Nov. 2015. [Online]. Available: http://arxiv.org/abs/1511.00363

[28] M. de Prado, M. Denna, L. Benini, and N. Pazos, "QUENN: Quantization engine for low-power neural networks," *CoRR*, vol. abs/1811.05896, pp. 1–9, Nov. 2018. [Online]. Available: http://arxiv.org/abs/1811.05896
[29] S. Gupta, S. Ullah, K. Ahuja, A. Tiwari, and A. Kumar, "ALigN: A highly accurate adaptive layerwise log_2_lead quantization of pre-trained neural networks," *IEEE Access*, vol. 8, pp. 118899–118911, 2020.
[30] S. Vogel, M. Liang, A. Guntoro, W. Stechele, and G. Ascheid, "Efficient hardware acceleration of CNNs using logarithmic data representation with arbitrary log-base," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, Nov. 2018, pp. 1–8.
[31] C. De la Parra, A. Guntoro, and A. Kumar, "ProxSim: GPU-based simulation framework for cross-layer approximate DNN optimization," in *Proc. Design, Automat. Test Eur. Conf. Exhib. (DATE)*, Mar. 2020, pp. 1193–1198.
[32] V. Mrazek, S. S. Sarwar, L. Sekanina, Z. Vasicek, and K. Roy, "Design of power-efficient approximate multipliers for approximate artificial neural networks," in *Proc. 35th Int. Conf. Comput.-Aided Design (ICCAD)*. New York, NY, USA: Association for Computing Machinery, 2016, pp. 1–7, doi: 10.1145/2966986.2967021.
[33] M. S. Ansari, V. Mrazek, B. F. Cockburn, L. Sekanina, Z. Vasicek, and J. Han, "Improving the accuracy and hardware efficiency of neural networks using approximate multipliers," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 28, no. 2, pp. 317–328, Oct. 2020.
[34] B. S. Prabakaran, S. Rehman, M. A. Hanif, S. Ullah, G. Mazaheri, A. Kumar, and M. Shafique, "DeMAS: An efficient design methodology for building approximate adders for FPGA-based systems," in *Proc. Design, Automat. Test Eur. Conf. Exhib. (DATE)*, Mar. 2018, pp. 917–920.
[35] S. Ullah, S. S. Murthy, and A. Kumar, "SMApproxLib: Library of FPGA-based approximate multipliers," in *Proc. 55th ACM/ESDA/IEEE Design Automat. Conf. (DAC)*, Jun. 2018, pp. 1–6.
[36] Z. Ebrahimi, S. Ullah, and A. Kumar, "LeAp: Leading-one detection-based softcore approximate multipliers with tunable accuracy," in *Proc. 25th Asia South Pacific Design Automat. Conf. (ASP-DAC)*, Jan. 2020, pp. 605–610, doi: 10.1109/ASP-DAC47756.2020.9045171.
[37] V. Rajagopalan, V. Boppana, S. Dutta, B. Taylor, and R. Wittig, "Xilinx Zynq-7000 EPP: An extensible processing platform family," in *Proc. IEEE Hot Chips 23 Symp. (HCS)*, Aug. 2011, pp. 1–24.
[38] *Xilinx AXI Interconnect v2.1*, LogiCORE IP Product Guide, 2017.
[39] Avnet. (2019). *ULTRA96-V2*. [Online]. Available: https://www.avnet.com/opasdata/d120001/medias/docus/193/5365-pb-ultra96-v2-v4a.pdf
[40] Y.-H. Chen, J. Emer, and V. Sze, "Eyeriss: A spatial architecture for energy-efficient dataflow for convolutional neural networks," in *Proc. ACM/IEEE 43rd Annu. Int. Symp. Comput. Archit. (ISCA)*, Jun. 2016, pp. 367–379.
[41] B. Wu. (2020). *SmallPositHDL*. [Online]. Available: https://github.com/starbrilliance/SmallPositHDL
[42] F. Xiao, F. Liang, B. Wu, J. Liang, S. Cheng, and G. Zhang, "Posit arithmetic hardware implementations with the minimum cost divider and SquareRoot," *Electronics*, vol. 9, no. 10, p. 1622, Oct. 2020. [Online]. Available: https://www.mdpi.com/2079-9292/9/10/1622
[43] M. Abadi *et al.*, "TensorFlow: A system for large-scale machine learning," in *Proc. 12th USENIX Symp. Oper. Syst. Design Implement. (OSDI)*, 2016, pp. 265–283.
[44] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet large scale visual recognition challenge," *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, Dec. 2015, doi: 10.1007/s11263-015-0816-y.

SURESH NAMBI received the B.E. degree in electrical and electronics from BITS Pilani, India, in 2020. He was a Guest Researcher with the Chair of Processor Design, TU Dresden, in 2020. He is currently working as a Processor Design Engineer for an AI chip startup in India. His research at TU Dresden involved exploring quantization and approximation methods for designing energy-efficient deep neural networks for edge computing. His research interests include computer architecture, hardware for machine learning, reconfigurable computing, and digital VLSI.

SALIM ULLAH received the B.Sc. and M.Sc. degrees in computer systems engineering from the University of Engineering and Technology Peshawar, Pakistan. He is currently pursuing the Ph.D. degree with the Chair of Processor Design, Technische Universität Dresden. His current research interests include the design of approximate arithmetic units, approximate caches, and hardware accelerators for deep neural networks.

SIVA SATYENDRA SAHOO received the master's (M.Tech.) degree in the specialization electronics design technology from the Indian Institute of Science, Bengaluru, in 2012, and the Ph.D. degree in reliability in heterogeneous embedded systems from the National University of Singapore, Singapore, in 2019. He has also worked with Intel India, Bengaluru in the domain of physical design. He is currently working as a Postdoctoral Researcher with the Chair of Processor Design, TU Dresden. His research interests include embedded systems, machine learning, approximate computing, reconfigurable computing, reliability-aware computing systems, and system-level design.

ADITYA LOHANA received the B.E. degree in computer science from BITS Pilani, in 2020. He was a Guest Researcher with the Chair of Processor Design, TU Dresden, and worked on privacy-aware distributed machine learning. He is currently working as a Software Engineer with Microsoft India. His research interests include deep learning, natural language processing, and distributed systems.

FARHAD MERCHANT (Member, IEEE) received the Ph.D. degree from the Indian Institute of Science, Bengaluru, India, in 2016. His Ph.D. thesis title was "Algorithm-Architecture Co-design for Dense Linear Algebra Computations." From March 2016 to December 2016, he worked as a Postdoctoral Research Fellow with Nanyang Technological University (NTU), Singapore. In December 2016, he moved to Corporate Research with Robert Bosch, Bengaluru, as a Researcher, where he worked on numerical methods for ordinary differential equations. He joined the Institute for Communication Technologies and Embedded Systems, RWTH Aachen University, as a Postdoctoral Research Fellow with the Chair for Software for Systems on Silicon, in December 2017. He received the DAAD Fellowship during his Ph.D. He was a recipient of the HiPEAC Technology Transfer Award, in 2019.

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