Trading Performance, Power, and Area on Low-Precision Posit MAC Units for CNN Training

Luís Crespo, Pedro Tomás, Nuno Roma, and Nuno Neves
INESC-ID, Instituto Superior Técnico, Universidade de Lisboa, Lisboa, Portugal
luis.miguel.crespo@tecnico.ulisboa.pt, pedro.tomas@inesc-id.pt, nuno.roma@inesc-id.pt, nuno.neves@inesc-id.pt

***Abstract*—The recently proposed Posit number system has been regarded as a particularly well-suited floating-point format to optimize the throughput and efficiency of low-precision computations in convolutional neural network (CNN) applications. In particular, the Posit format offers a balance between decimal accuracy and dynamic range, which results in a distribution of values that seems particularly interesting for deep learning applications. However, the adoption of the Posit still raises some concerns regarding hardware complexity, particularly when accounting for the overheads associated with the quire exact accumulator. Accordingly, this paper presents a holistic study on the model accuracy, performance, power, and area trade-offs when adopting low-precision Posit multiply-accumulate (MAC) units for the training of CNNs. In particular, 28nm ASIC implementations of a reference Posit MAC unit architecture demonstrate that the quire accounts for over 70% of the area and power utilization, and the obtained CNN training results showed that its use is only strictly required when considering mixed low-precision configurations. As a result, reducing the size of the quire results in an average reduction of area and power by 57% and 47%, without imposing visible training accuracy losses.**

***Index Terms*—Posit Number System, Quire Structure, Low-precision Arithmetic, Convolutional Neural Networks, Deep Learning**

I. Introduction

The recently proposed Posit format [1] has been gaining some momentum in the Deep Learning (DL) domain. While it was initially proposed as a direct replacement to the IEEE-754 standard, its benefits have been most evident in low-precision arithmetic scenarios, particularly in the training and inference phases of Convolutional Neural Networks (CNNs) [2]–[6]. By design, the Posit number system offers a unique trade-off between dynamic range and decimal precision by including a new regime field. Due to their inherent balance, Posits are particularly suited to represent numbers near zero (in magnitude). Additionally, the Posit standard [7] features an optional exact accumulator structure (quire) for fused operations.

However, the implementation of the quire introduces a significant increase in logic complexity when designing Posit arithmetic units [8]–[12]. In fact, when considering the deployment of FMA/MAC units, typically used in tensor-like accelerators [5], [13], [14], it is necessary to account for the overheads associated not only with the decoding/encoding but also with the quire. Although the quire format has been updated in recent Posit format revisions [7], [15], it still poses a significant concern since its size grows linearly by a factor of the precision and exponentially with the exponent size.

In this respect, while a significant body of work has already demonstrated the different Posit configuration trade-offs for CNNs training [3]–[6], most studies do not account for the hardware cost of implementing Posit arithmetic units. In fact, most previous studies have been circumscribed to mathematical evaluations with Posit emulation libraries [6], not account for the quire and its overheads [5] or have a limited scope [16].

Accordingly, this paper consists of a comprehensive study of the Posit numbering system, comprising its different formats and characteristics, its impact on the training of neural networks, and how this relates to hardware requirements. Hence, instead of aiming our objectives to the comparison of the Posit and IEEE-754 formats, we focused our attention on the evaluation of different Posit configurations.

With the premise that the quire is one of the most critical structures (both in terms of result accuracy and hardware overhead), the presented study focuses on evaluating its use and impact with low-precision Posit units, by attempting to answer the following key questions:

Q1 – Accuracy: What is the impact in the trained model accuracy of adopting different Posit configurations?
Q2 – Quire Format: How does the use of a quire influence the accuracy of the trained model?
Q3 – Hardware Cost: Are the potential gains worth the hardware overhead associated with the quire?
Q4 – Mixed-Precision: Can the performance, power, and area of Posit MAC units be fine-tuned by adopting different Posit configurations in different CNN stages?
Q5 – Alternatives: Can these overheads be mitigated by using accumulator structures alternative to the quire without significant impacts on the model accuracy?

To answer these questions, the presented study makes use of the recently proposed PositNN framework [6] to emulate the training phase of LeNet-5 and CifarNet CNN models. The conducted evaluations adopted different Posit configurations, quire formats, and alternative structures. The obtained accuracy results were paired with the resulting performance, power,

This work was partially supported by national funds through Fundação para a Ciência e a Tecnologia (FCT) under projects UIDB/50021/2020, PCIF/MPG/0051/2018, 2022.06780.PTDC, 2022.11626.BD, and from the European High Performance Computing Joint Undertaking (JU) under Framework Partnership Agreement No 800928 and Specific Grant Agreement No 101036168 (EPI SGA2). The JU receives support from the European Union's Horizon 2020 research and innovation programme and from Croatia, France, Germany, Greece, Italy, Netherlands, Portugal, Spain, Sweden, and Switzerland.

and area metrics, obtained from implementations of reference Posit FMA/MAC units using 28nm ASIC technology.

The observed results show that the use of the quire is fundamental in CNN training for mixed-precision configurations. However, it still accounts for 82% of the total chip area and 72% of the power consumption in an 8-bit Posit MAC unit. Consequently, a new scaled accumulator structure is proposed, showcasing a viable alternative to the standard quire. This is achieved by reducing the size of the fixed-point format quire and by pairing it with a scale factor component. When compared with ASIC implementations of a reference Posit MAC with the standard quire, the proposed structure provides an average reduction of area and power by 57% and 47%, respectively, with no significant impact on the model accuracy.

In accordance, this paper introduces the following contributions and features:

- An in-depth study on the Posit system when training CNNs. The study evaluates the model accuracy, performance, power, and area trade-offs of adopting low-precision Posit arithmetic units, by considering multiple Posit configurations, with and without quire, as well as mixed-precision scenarios.
- Definition of a new scaled accumulator structure, alternative to the quire, that reduces its size to mitigate hardware requirements while maintaining the model accuracy.
- Extension of the PositNN framework [6] and the underlying Universal library [17] to support the scaled accumulator and alternative quire formats.
- Architecture RTL design and ASIC implementation of the proposed scaled accumulator structure in a reference Posit FMA/MAC unit.[1]

II. Background

The Posit numbering system [1] was designed as an alternative numbering format to the IEEE-754 floating-point standard [20] (henceforth simply referred to as floats). The Posit format defines a parameterization pair $<n, es>$, where n represents the precision (i.e., bit-width) and es denotes the maximum exponent size. The general structure of an n-bit posit with es exponent bits is depicted in Eq. 1.

$$\underbrace{\overbrace{s}^{sign}\ \overbrace{r\,r\,\ldots\,\bar{r}}^{regime}\ \overbrace{e_0\,e_1\,\ldots\,e_{es-1}}^{exponent}\ \overbrace{f_0\,f_1\,f_2\,\ldots}^{fraction}}_{n\,bits} \tag{1}$$

Similarly to floats, the Posit structure includes the *sign*, *exponent*, and *fraction* fields, but with an additional field called *regime*. Whenever the sign bit corresponds to a negative number, it is necessary to take the 2's complement before decoding the remaining fields. Moreover, the regime is a variable-sized field, whose encoded value (k) is given by the run-length (m) of '0' or '1' bits:

$$k = \begin{cases} m-1 & \text{, if } r = 1 \\ -m & \text{, otherwise.} \end{cases} \tag{2}$$

[1]https://github.com/hpc-ulisboa/Posit-FMA-Units/

As a consequence of this variable-sized regime, the exponent and fraction bit-widths are unknown before decoding the regime. The calculated k together with the exponent configuration and the value encoded in the exponent field (exp) defines the scaling factor (sf) for the encoded posit (equivalent to the exponent in the IEEE-754 format), which is given by:

$$sf = exp + k2^{es} \tag{3}$$

Accordingly, a posit number value is given by:

$$(-1)^{sign} \times 2^{sf} \times 1.f \tag{4}$$

The Posit format has single encodings for zero (000...0) and Not-a-Real (NaR) to represent all mathematical exceptions (100...0). Moreover, posits do not underflow to 0 neither overflow to infinity. Instead, they saturate to the minimum and maximum representable number, respectively.

The Posit format makes use of an exact accumulator structure (*quire*), based on the Kulisch accumulator [21]. It is used to store sums of products without rounding, avoiding any accuracy loss. The quire is a fixed-point 2's complement value (see Fig. 1) composed by 4 fields: sign, carry guard (cg), integer (int) and fraction ($frac$); and its size is given by:

$$\text{quire size} = 1 + cg + 2^{es+2} \times (n-2) \tag{5}$$

Although the Posit numbering system supports an arbitrary parameterization of the precision and of the exponent size, there are standardized configurations. In particular, the Posit standard has two main releases, differing on the hyperparameter es and the carry guard (cg) size. While an older release from 2018 (Release 4.3) [15] defines $es = \log_2(n) - 3$ for the most commonly adopted precisions with 8, 16, and 32 bits (to correspond to the same dynamic ranges used in IEEE-754 floating-point arithmetic) and $cg = n-1$, the latest release from 2021 (Release 4.12) [7] sets a fixed $es = 2$ (providing low-precision posits a much wider dynamic range) and $cg = 31$ in all configurations (allowing billions of accumulations).

Table I depicts the recommended Posit and quire configurations and their characteristics according to the respective standard release, 4.3 [15] and 4.12 [7]. It can be observed that increasing es provides a larger dynamic range at the cost of fraction bits (accuracy). The latest revision (4.12) was proposed as a direct consequence of the conclusion from several studies indicating that, in DL applications, $es = 2$ configurations provide better results for low-precision posits [3]–[6], [22]–[24]. However, since in the older release [15] any increase of es imposed a significant increase of the quire size for low-precision posits, the latest revision [7] fixed the cg to 31 bits for all configurations, providing an accumulation capacity of up to $2^{31} - 1$ values.

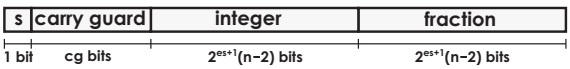

Fig. 1. Binary quire format adopted in Posit numbering system.

TABLE I
GENERAL CHARACTERISTICS OF THE POSIT FORMAT AND ITS QUIRE ACCUMULATOR ACCORDING TO THE TWO LATEST POSIT STANDARD RELEASES (REL. 4.3 [15] AND REL. 4.12 [7]), MATCHED WITH THE IEEE-754 STANDARD AND FP8 [18], [19] COUNTERPARTS.

# Bits	Exp. Size	Fraction Length	Dynamic Range		Quire				
					size		*cg* size		int/frac size
			minpos	maxpos	Rel. 4.3 [15]	Rel. 4.12 [7]	Rel. 4.3 [15]	Rel. 4.12 [7]	
Float (FP8)	5	2	1.5×10^{-5}	5.7×10^{4}					
Posit<*8*, *es*>	**0**	0 to 5	1.5×10^{-2}	64	**32**	56	**7**	31	12
	1	0 to 4	2.4×10^{-4}	4.1×10^{3}	56	80	7	31	24
	2	0 to 3	6.0×10^{-8}	1.7×10^{8}	104	**128**	7	**31**	48
	3	0 to 2	3.6×10^{-15}	2.8×10^{14}	200	224	7	31	96
Float (FP16)	5	10	6.0×10^{-8}	6.6×10^{4}					
Posit<*16*, *es*>	0	0 to 13	6.1×10^{-5}	1.6×10^{4}	72	88	15	31	28
	1	0 to 12	3.7×10^{-9}	2.7×10^{8}	**128**	144	**15**	31	56
	2	0 to 11	1.4×10^{-17}	7.2×10^{16}	240	**256**	15	**31**	**112**
	3	0 to 10	1.9×10^{-34}	5.2×10^{33}	464	480	15	31	224
Float (FP32)	8	23	1.4×10^{-45}	3.4×10^{38}					
Posit<*32*, *es*>	0	0 to 29	9.3×10^{-10}	1.1×10^{9}	152		31		60
	1	0 to 28	8.7×10^{-19}	1.2×10^{18}	272		31		120
	2	0 to 27	7.5×10^{-37}	1.3×10^{36}	**512**		**31**		240
	3	0 to 26	5.7×10^{-73}	1.8×10^{72}	992		31		480

III. RELATED WORK

The advantages of the Posit format for different application domains have attracted the interest of the research community, with several hardware implementations and software platforms being proposed. The benefits of the Posit system have been particularly studied in the training and inference of CNNs.

A. Posit Arithmetic Units

Recent works include support for fundamental arithmetic operators (adder/subtractor and multiplier) [9], [25]–[27], FMA operations [28]–[31] and quire accumulation [4], [9], [11], [14], [32], [33]. In what concerns MAC architectures, Charmichael et al. [4] proposed a low-precision 8-bit Posit unit for CNN inference. Zhang et al. [28] defined a parameterized Posit FMA unit generator. Similarly, Murillo et al. [32] proposed a set of algorithms that allow generating synthesizable VHDL using FloPoCo [34]. Alternative Posit architectures have also been recently explored. In particular, Neves et al. proposed a Posit MAC unit with dynamically configurable exponent size [33] and a reconfigurable tensor unit with a 4x4 array of 64-bit vectorized Posit MAC units [14].

B. Software Platforms

While off-the-shelf Posit hardware is not readily available, a great body of work has been devoted to developing emulation libraries and frameworks to evaluate the Posit number system. The most prominent emulation libraries include Universal [17], a C++ template library that supports any arbitrary precision posit and quires; SoftPosit [35], a C library (endorsed by the Posit creators) with limited posit configuration support; cppPosit [36], a C++ header-only library with support for many posit variants, defined by template parameters. Machine learning frameworks have also been proposed that due to the current hardware limitations make use of the mentioned emulation libraries. Examples include PositNN [6], a framework based on PyTorch's C++ API (Libtorch) capable of end-to-end training and inference with any configuration; and Deep PeNSieve [2], an extension to TensorFlow.

To tackle the lack of standard availability of Posit arithmetic, ISA and compiler back-end support are introduced by Xposit [10], offering a Posit RISC-V extension.

C. Posit Number System Studies

Most studies that apply Posit to DNNs only address the inference stage [4], [16], [37], [38], which is often easier and less sensitive to errors than the training phase. In particular, Nakahara et al. [16] study reducing the size of the quire. More recent studies explored the use of Posits in the training of DNNs [2], [3], [5], [6], where performance improvements and energy savings are more compelling. One of the first studies [3] managed to train an FCNN on a simple binary classification problem with a minimum of 12-bit Posit, while smaller formats showed irregular convergence. Later, Lu et al. [5] trained CNNs using 8 and 16-bit posits in a mixed-precision setup, but still relying on the IEEE-754 format in their hardware prototype. Murillo et al. [2] trained CNNs using their Deep PeNSieve framework. They managed to converge CNNs trained with 32 and 16-bit Posit, but failed to converge using Posit<8 , 0>. More recently, Raposo et al. [2] trained CNNs using as low as Posit<8 , 2> in a mixed precision setup.

Despite their success, most studies have been circumscribed to mathematical evaluations, not relating them with the respective hardware requirements of Posit arithmetic units.

IV. CNN TRAINING WITH LOW-PRECISION POSIT

The following paragraphs present two studies. First, a brief mathematical analysis of the Posit number system, with particular emphasis on the decimal accuracy that is offered in low-precision Posit configurations. Then, it is evaluated the training accuracy of several datasets with the considered Posit configurations. This is done by also assessing different quire

formats and by evaluating the viability of a new alternative accumulating structures at the quire.

A. Mathematical Analysis

By following a similar approach as in [1], the representation capabilities of the considered Posit configurations (and float counterparts) were measured through an exhaustive analysis of their dynamic range and decimal accuracy:

$$\text{Decimal Accuracy} = -\log_{10}\left|\log_{10}\left(\frac{x_{computed}}{x_{exact}}\right)\right| \quad (6)$$

The decimal accuracies of the considered Posit and float formats are plotted in Fig. 2, when considering 32 (A), and 8 bits (B). The obtained plots clearly demonstrate the tapered behavior of the Posit format, when compared to the representation offered by floats. In particular, posits provide a higher accuracy for numbers represented with a near-zero scale factor (i.e., around values $\{-1,+1\}$), thus defining an area that is usually referred to as *golden zone* [8]. As the dynamic range increases in magnitude, the accuracy gradually decreases, by following a pyramid-like structure. Floats, on the other hand, maintain a constant accuracy, with an exception in the subnormal range, which presents a tapered accuracy.

When considering 32-bit precisions (see Fig. 2.A), there is a visible trade-off relating decimal accuracy and dynamic range between posits and floats. While posits offer higher accuracy for numbers represented with a near-zero scale factor, when the dynamic range increases they are surpassed by floats, until they close off at zero or infinity. Conversely, by reducing the decimal accuracy, posits can reach a much larger dynamic range before saturating. However, this trade-off is much less accentuated in lower precisions. For 8-bit configurations (see Fig. 2.B), although posits still show higher accuracy for low-magnitude numbers, the same relation is not so evident for high-magnitude numbers due to the sparsity of the values.

For low-precision, it is clear that the Posit format presents mathematical representation benefits when compared to floats, especially for low-magnitude numbers. On the one hand, the weight parameters of CNNs usually follow a normal distribution, with most of the values centered around zero (in magnitude) [6]. This observation suggests that posits may provide a higher CNN model accuracy than floats, whose distribution is sparser. On the other hand, posits are also potentially well-suited to mitigate the vanishing gradient problem, usually visible when training CNNs with low-precision formats. This problem occurs when gradients become smaller while the model converges, often resulting in weight updates so small that their decimal accuracy is lost.Posits can handle weight updates for longer (due to the higher decimal accuracy) and potentially result in more accurate models at lower precisions.

B. CNN Training Study

Presently, there is still no standard hardware (and respective software) with native support for Posit. Consequently, we were faced with the need to use a framework that emulates their mathematical behavior in software (PositNN [6]). Contrarily

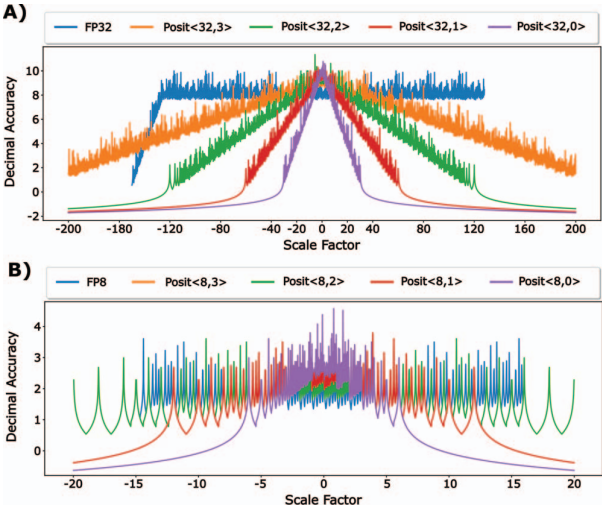

Fig. 2. Comparison between the decimal accuracy of A) 32-bit posits and 32-bit floats and B) 8-bit posits and an 8-bit float variation for different Posit exponent configurations. Notice that in the B) plot, the Posit<8,3> is not visible since it overlaps with FP8 in the [-5, 5] range.

to other mainstream tools (such as PyTorch and TensorFlow) that do not support Posit arithmetic, the PositNN framework allows to run the training and inference phases of DL models with arbitrary Posit precision and exponent configurations.

Accordingly, the following CNN models were considered in this study, by evaluating the trained model accuracy:

- A 2-layer FCNN trained with MNIST for 10 epochs;
- A LeNet-5 model trained with MNIST and Fashion MNIST for 10 epochs;
- A CifarNet variation (with $\sim$ 0.5 million parameters) trained with CIFAR-10 and CIFAR-100 for 20 epochs.

The choice and complexity of the adopted models (and number of training epochs) is limited by the emulation time of the posit arithmetic by the PositNN library, which can take several weeks for each configuration. Furthermore, emulating more complex workloads and models is an unbearable effort, both from the run-time perspective and from the matter of support in the available frameworks (e.g., ResNet requires BatchNorm2d which is not available in existing tools).

All the considered models were trained with the Posit<n,es> configurations presented in Table I, by considering $n = \{6,8,10,12,16\}$ and $es = \{0,1,2,3\}$. To detect potential accuracy losses, all models were also trained with 32-bit floats, the de facto representation used to train machine-learning models. Posit configurations with n>16 are not presented here since all tests with such configurations showed a similar performance to the float case. In an attempt to answer the questions defined in Section I regarding the utilization of the quire, three experimental setups were defined, by considering: *i)* not using a quire; *ii)* using a quire format ($cg = 31$) according to Release 4.12 [7]; and *iii)* using a quire format ($cg = n-1$) according to Release 4.3 [15]. From the insights gathered from these setups, a new accumulator structure alternative to the quire was also defined and evaluated.

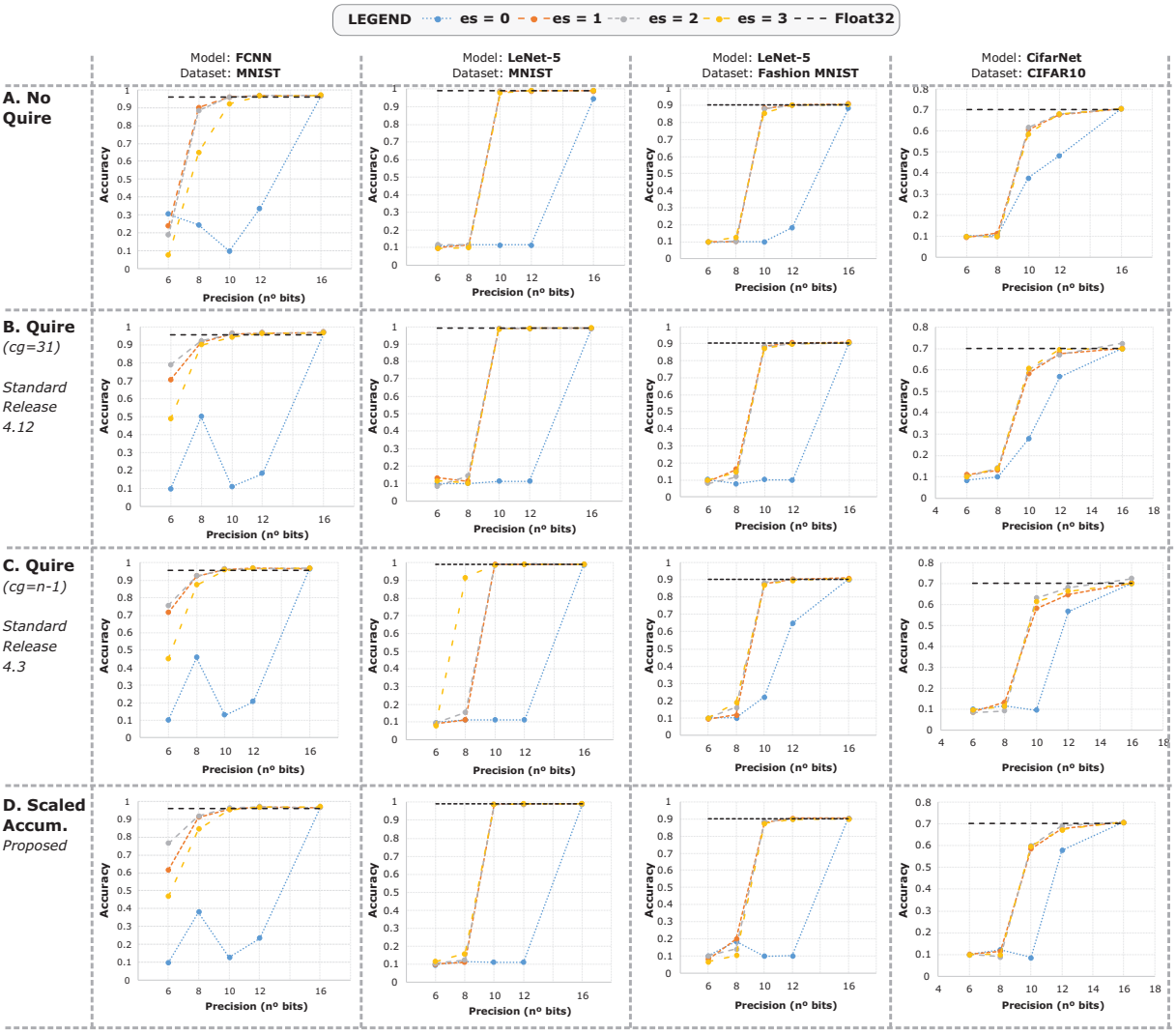

Fig. 3. Model and respective dataset accuracy obtained from the training of CNNs with posit configurations (A) without using a quire, (B) using a quire format ($cg = 31$) according to Release 4.12 [7], (C) using a quire format ($cg = n - 1$) according to Release 4.3 [15], and (D) using a new accumulator structure alternative to the quire. As reference, 32-bit floats are used in every model and respective dataset to detect potential accuracy losses.

Without Quire: Fig. 3.A presents the attained model accuracy when considering the several posit configurations without the use of quire. It can be observed that posits with $es = 0$ have more difficulty to converge, presenting worse results when compared to the other exponent configurations. This is a direct result of the reduced dynamic range of low-precision configurations. Conversely, when considering $es = 3$, a higher dynamic range is attained, but at the cost of a fraction accuracy loss. This also leads to a slightly lower model accuracy when compared to the configurations with $es = 0$ and $es = 1$, especially in the FCNN model with precisions lower than 10 bits. In fact, as the model complexity increases, the required precision to successfully train the models also increases. This can be observed by comparing the FCNN and the LeNet-5 models. While the first reaches an accuracy similar to the reference 32-bit float with 8-bit posits (and maximum accuracy with 10 bits), the LeNet-5 model does not converge with precisions lower than 10 bits. However, it also attains a similar accuracy to the float reference with 10-bits posits. This is further confirmed when observing the accuracy of the CifarNet model, which progressively approaches the maximum accuracy with 10 and 12-bit precisions, reaching it at 16 bits.

With Quire *($cg = 31$ – Standard Release 4.12):* Fig. 3.B presents the obtained model accuracy when using the quire format with $cg = 31$, defined in the most recent Posit standard. Similarly to the setup without quire, the models have difficulty converging with $es = 0$, but overcome this issue when the exponent increases. However, a significant difference from the previous case is now observed with respect to the convergence and model accuracy: similar performance to the FP32 reference is now attained with much narrower precisions. In particular, the FCNN model is able to converge with only 6-bit

TABLE II
EVALUATION OF MIXED LOW-PRECISION CONFIGURATIONS FOR CNN TRAINING. ALL MODELS USE A POSIT<*16, 2*> CONFIGURATION ON THE LOSS AND OPTIMIZER STAGES. AS REFERENCE, 32-BIT FLOATS ARE USED TO DETECT POTENTIAL ACCURACY LOSSES.

Format	Setup	LeNet5		CifarNet			
		MNIST	Fashion MNIST	CIFAR-10		CIFAR-100	
		Accuracy	Accuracy	Top-1	Top-3	Top-1	Top-5
Float (FP32)		**99.19%**	**90.35%**	**70.25%**	**92.35%**	**35.95%**	**65.59%**
Mixed16 Posit<8 , 2>	No Quire	98.32%	83.88%	15.72%	44.22%	1.00%	5.00%
	Quire ($cg = 31$)	99.17%	90.89%	71.11%	92.45%	35.61%	66.88%
	Scaled Accum.	99.16%	89.97%	70.96%	92.56%	34.45%	65.78%
Mixed16 Posit<8 , 1>	No Quire	47.54%	10.00%	10.00%	30.00%	1.00%	5.00%
	Quire ($cg = 31$)	11.35%	10.00%	70.70%	92.34%	36.72%	67.00%
	Scaled Accum.	11.35%	10.00%	71.86%	92.58%	37.10%	67.24%
Mixed16 Posit<6 , 2>	No Quire	9.80%	10.00%	10.00%	30.00%	1.00%	5.00%
	Quire ($cg = 31$)	98.83%	87.52%	62.04%	87.87%	1.00%	5.00%
	Scaled Accum.	98.79%	88.07%	60.66%	86.77%	1.00%	5.00%
Mixed16 Posit<6 , 1>	No Quire	9.80%	10.00%	10.00%	30.00%	1.00%	5.00%
	Quire ($cg = 31$)	11.35%	10.00%	12.57%	36.62%	1.00%	5.00%
	Scaled Accum.	11.35%	10.00%	10.00%	30.00%	1.00%	5.00%

posits (although with a lower accuracy), attaining an accuracy close to the reference with only 8-bit posits. However, for the remaining (more complex) models, the quire did not introduce significant differences. In fact, the LeNet-5 model still requires 10-bit posits to attain convergence and the CifarNet model shows no improvements for 8 and 10-bit posits.

According to these observations, two conclusions can be attained: *i)* this quire configuration ($cg = 31$) is only relevant for low-precision training; and *ii)* for higher precisions, the models converge and attain an accuracy similar to the FP32 reference, independently of the use of a quire.

With Quire *($cg = n - 1$ – Standard Release 4.3):* To provide further insights on the utilization of the quire, the CNN models were also trained using the quire format from the previous Posit standard. In this release, the quire makes use of a variable carry guard size, proportional to the Posit precision ($cg = n - 1$). This results in smaller quire sizes for low-precision configurations, allowing to reduce the hardware costs on dedicated units. Fig. 3.C highlights this fact by showing that, in general, there is no significant difference in the attained accuracy in relation to the previous setup.

Proposed Scaled Accumulator*:* The main objective of the quire structure is to allow repeated accumulations with full accuracy and overflow protection. However, it was verified that not only CNN models can tolerate certain accuracy losses, but they also rarely require large quires. To overcome such compromise, the typical quire structure was replaced by a scaled accumulator, attained by fixing the carry guard size to 7 bits (henceforward denoted as *accumulation guard*), reducing the integer and fraction fields to the $es = 0$ size, and pairing with a scale factor to correctly represent the values. Hence, by considering Eq. 5, the proposed scaled accumulator is set at:

$$\text{accumulator size} = 4n \tag{7}$$

$$\text{scale field size} = \log_2(n) + es + 2 \tag{8}$$

It should be highlighted that such proposed scaled accumulator is not designed to provide higher accuracy. Instead, it aims a decrease the implementation cost of traditional quire structures required for CNN training (as it will be shown in section VI).

The required changes were introduced in the software library that emulates posits (Universal library [17]) to evaluate its results for CNN training. Fig. 3 (row D) presents the obtained results for the considered models. It can be observed that there is no significant difference when compared to the standard quires (see rows B and C). However, there is an added benefit for the FCNN model with 6-bit posits, achieved with both scaled accumulator and standard quire.

C. CNN Training with Mixed-Precision Arithmetic

As it was shown in previous studies [5], [6], even narrower numerical formats can be used if different precisions are used across the different training stages, namely at the forward pass, calculus of the loss function, optimizer and backward pass.

Hence, to further evaluate the importance of the quire in posit units, the LeNet-5 and CifarNet models were trained in a mixed precision configuration, by considering 6- and 8-bit posits in the forward and backward passes, and 16-bit posits in the optimizer and loss stages (as they achieve float-like accuracy in all the considered models and exponent configurations). Additionally, to determine the best exponent configuration in the forward and backward passes, these models were trained with exponent sizes $es = 1$ and $es = 2$. The training results for the MNIST, fashion MNIST, CIFAR-10 and CIFAR-100 datasets are presented in Table II. To detect potential accuracy losses, all models were also trained with 32-bit floats.

It can be observed that the Mixed16 Posit<8 , 2> configuration converges in all models when using the quire and the scaled accumulator with an accuracy similar to the 32-bits float reference (apart from small variations). For the same configuration, although the LeNet-5 model converges in both datasets (MNIST and Fashion MNIST) – even without

using the quire – a slight accuracy decrease is observed. On the other hand, on the CifarNet model, the quire or scaled accumulator are fundamental to attain accurate results, as there is a significant decrease in accuracy for CIFAR-10 and a convergence failure for CIFAR-100. In contrast, for the Mixed16 Posit<8 , 1> configuration, accurate results are only attained for the CifarNet model when using the quire or the proposed scaled accumulator.

When trying to reduce the precision to 6 bits, it was observed that the Mixed16 Posit<6 , 2> configuration was still able to converge with similar accuracy when using the quire and scaled accumulator (with the exception of the CIFAR-100 dataset), although with some accuracy losses depending on the model and dataset complexity. In contrast, all trained models failed to converge when the quire was not used. For the Mixed16 Posit<6 , 1> configuration, all setups failed to converge due to the insufficient dynamic range.

Finally, several conclusions can be taken when considering the posit configuration, quire and scaled accumulator:

- the Posit<8 , 2> configuration is the recommended setup when training simple models in mixed precision configuration; however, Posit<8 , 1> and Posit<6 , 2> can also be used depending on the model and requirements;
- the use of the quire is essential when training CNN models with low-precision posits;
- the proposed scaled accumulator (devised to reduce hardware overheads) did not affect the results, achieving the same accuracy as the standard quire.

V. Posit FMA/MAC Architecture

To complement the study presented in the previous section, and evaluate the hardware overheads associated with the use of a quire structure, it is first necessary to define a reference architecture of a Posit arithmetic unit. As such, this section presents a generic Posit FMA/MAC unit architecture (depicted in Fig. 4.A), derived from recent state-of-the-art units [4], [9], [11], [14], [32], [33]. The devised datapath was implemented with a parameterizable RTL template that allows generating different designs with different pairs of precision and exponent configurations. The utilization of a quire is optional, which can also be configured to follow any of the considered Posit standard releases and the corresponding parameters, by adjusting the carry guard size (see Eq. 5). Finally, the proposed scaled accumulator structure was also implemented and integrated into the reference Posit FMA/MAC architecture by performing the necessary modifications to the datapath.

A. Reference Posit FMA/MAC Architecture

The developed Posit FMA/MAC unit was implemented with a 5-stage pipelined architecture. The datapath supports addition, multiplication, fused multiply-add and multiply-accumulate operations. As such, it follows the classical pipeline stage distribution: *i) decode*; *ii) multiply*; *iii) quire* (add/accumulate); *iv) normalize*; and *v) encode*. The following paragraphs describe each stage in detail.

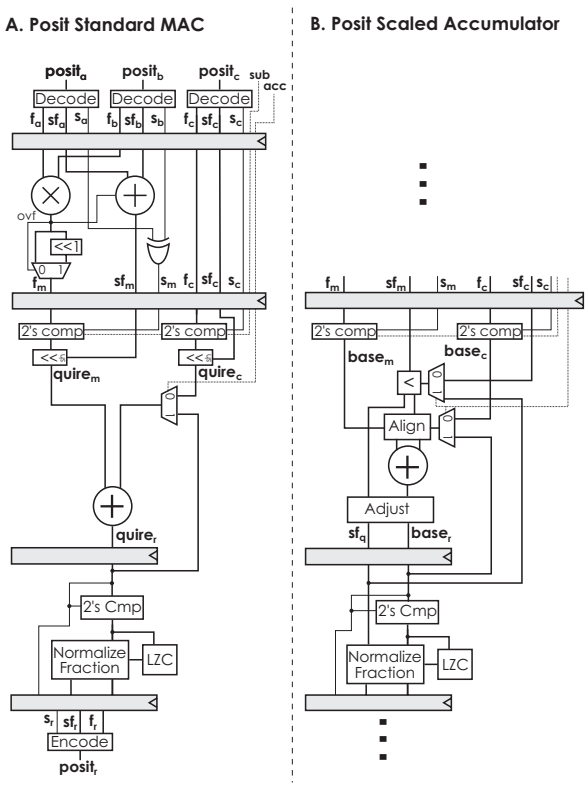

Fig. 4. Reference FMA/MAC architecture for the (A) standard quire and (B) scaled accumulator.

Decode: The Decode stage comprises three equivalent decoding modules (see Fig. 4.A), one for each operand. As a consequence, each decode stage translates the input posit value to the corresponding sign (s), scale factor (sf) and fraction (f) fields, according to the processing schemes defined in the literature [4], [25]–[27], whose internal structure is depicted in Fig. 5.A. The process starts by taking the 2's complement of the input value, according to the sign bit. Next, the regime run-length is decoded by means of a leading-zero counter (LZC) (if it starts with '1' the value is first inverted). Then, k is calculated and the regime is left-shifted out according to the obtained zero count, leaving the exponent and fraction. The k value is then concatenated with the exponent to obtain sf. Finally, a '1' bit is added to the fraction to obtain f.

Multiply: The Multiply stage (see Fig. 4.A) performs the multiplication of the decoded posit_a and posit_b operands, while propagating the decoded posit_c to the next stage. Multiplication is performed by using the conventional floating-point scheme, by performing a XOR between the signs, an addition of the scale factors, and a multiplication of the fractions.

Quire Arithmetic: The Quire stage (see Fig. 4.A) starts by converting the multiplication result and the third operand (posit_c) to the quire format, which is accomplished by *i)* taking the fraction 2's complement, according to the sign and operation (*sub* signal); and *ii)* shifting the value according to the scale factor. With the values in the quire format, addition or accumulation with a previously stored quire value is performed

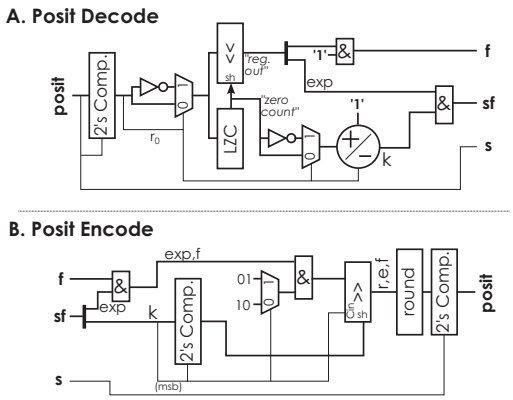

Fig. 5. Posit (A) decoding and (B) encoding modules of the reference FMA/MAC architecture.

(depending on the *acc* signal).

Normalization: The Normalize stage (see Fig. 4.A) is responsible for re-normalizing the obtained quire and extracting the s, sf, and f signals. First, the sign vector is extracted from the most significant bit of the quire, which allows converting the quire to an unsigned value through a 2's complement. Next, the number of shift positions required to normalize the quire is obtained with a LZC. The obtained zero count is used by a left shifter to align the unsigned quire vector. Finally, the scale factor is obtained by adding the obtained zero count and an offset (due to the quire conversion).

Encode: The Encode stage (see Fig. 5.B) converts the s, sf, and f signals of the output value back to the Posit format, according to the schemes defined in the literature [4], [25]–[27]. The process starts by detaching the regime value (k) and the exponent (e) from the scale factor (sf), according to the es configuration. Then, the k value 2's complement is taken and the regime is shifted together with the exponent and fraction, according to k and its sign. The resulting binary value is then rounded and the 2's complement is taken according to s.

B. Alternative Scaled Accumulator

As referred in section IV-B, to mitigate the hardware overheads associated with the use of the quire, an alternative scaled accumulator structure was proposed. Accordingly, the following paragraphs describe its structure and the corresponding modifications (see Fig. 4.B) to the base architecture.

Scaled Accumulator Architecture: The study presented in Section IV showed it is possible to rely on an alternative structure to reduce the logic complexity and attempt to mitigate the quire hardware overheads, while maintaining the training accuracy of CNN models.

The proposed new binary format for the quire (defined in Fig. 6 and Eq. 5) maintains a representation with a 2's complement fixed-point `base` value with size $4n$, obtained by fusing the quire carry guard and the integer fields into a fixed-size 7-bit *accumulation guard* (ag) field. Since the devised fixed-point value precision is not enough to represent the entire dynamic range of the corresponding Posit configuration, the value is paired with a *scale factor* of size $\log_2(n) + es + 2$.

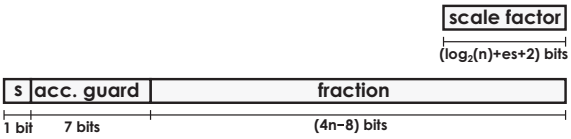

Fig. 6. Binary scaled accumulator format.

The introduced ag field results in an immediate reduction in logic complexity, by simply limiting the maximum number of accumulations that maintain overflow protection. As an example, a Posit<*8, 2*> configuration maintains a scaled accumulator with a total of 32 bits, instead of the corresponding 104 bits (release 4.3 [15]) or 128 bits (release 4.12 [7]) of the original quire. Furthermore, with the introduction of the associated scale factor, the new structure still allows a high number of accumulations without significant accuracy loss.

Alternative Accumulator Stage: To deploy the proposed scaled accumulator in the reference Posit MAC architecture, the Quire stage was replaced with the new Accumulator stage (see Fig. 4.B). Similarly to the original Quire stage, the input operands (posit_c and the result from the Multiply stage) are first complemented and sign-extended and the fraction most significant bit is placed in the accumulator guard least significant bit, to obtain the *base* and *scale factor* values. Then, the registered accumulator value and the propagated operand are selected (with the *acc* signal) and the corresponding *scale factor* values are compared. The *base* values are then aligned according to their *scale factor*. Specifically, the operand with the smaller *scale factor* is shifted by the *scale factor* difference, while the operand with the greater *scale factor* remains unchanged. After being aligned, the *base* values are added. For overflow protection purposes, the accumulator guard's most significant bit is checked for potential overflows, adjusting the accumulator *base* and *scale factor* values accordingly.

Normalization Stage Modifications: Similarly to the previous stage, the original Normalize stage was also slightly modified. The sign is extracted from the *base* most significant bit element, and the value is converted to an unsigned value. Next, the *base* is normalized to extract the Posit fraction and its magnitude (obtained with a LZC as in the original Normalization stage). The obtained magnitude is added to the accumulator *scale factor* to obtain the final Posit scale factor.

VI. Evaluation

To correlate the CNN training results with the corresponding hardware requirements, RTL descriptions of the Posit configurations listed in Table I and quire setups described in Section IV-B were implemented according to the reference Posit FMA/MAC architecture (section V). Specifically, the arithmetic units for each Posit configuration were synthesized *i)* without the use of the quire – *NQ*; *ii)* using the quire from the latest Posit standard release 4.12 [7] – *Q*; *iii)* using the quire from the previous Posit standard release 4.3 [15] – *QO*; and *iv)* using the proposed scaled accumulator – *SA*. All RTL descriptions were synthesized by targeting the 28nm UMC standard cell technology under typical operating conditions

TABLE III
COMPARISON OF THE REFERENCE UNITS WITH STATE-OF-THE-ART.

Unit/ Design	Pipeline Stages	ASIC Tech.	Area (μm^2)	Power (mW)	Delay (ns)
8-bit MAC (Q)	5	28 nm	7288	26.58	0.65
8-bit MAC (SA)	5	28 nm	1584	5.70	0.78
8-bit FMA (NQ)	5	28 nm	1613	6.97	0.50
8-bit MAC [27]	3	45 nm	4346	4.47	3.01
8-bit FMA [28]	0	28 nm	1400	1.08	1.00
16-bit MAC (Q)	5	28 nm	17062	48.30	0.78
16-bit MAC (SA)	5	28 nm	5705	17.60	0.91
16-bit FMA (NQ)	5	28 nm	5028	22.12	0.69
16-bit MAC [27]	3	45 nm	10258	10.91	5.99
16-bit FMA [28]	0	28 nm	3700	3.19	1.3

(1.05 V, 25° C). Chip area and power estimation results were obtained with Cadence Genus 19.11.

To ensure a reliable evaluation of the hardware requirements, the defined FMA/MAC architecture was first compared with current state-of-the-art solutions. Accordingly, selected 8- and 16-bit architectures with $es = 2$ (for the NQ, Q, and SA setups) were synthesized and matched with equivalent units from the literature. In particular, 8- and 16-bit versions of the MAC unit described in [27] and the FMA unit presented in [28] were considered for this validation. Table III presents the corresponding synthesis results.

The obtained area, power, and delay results show that the designed reference Posit FMA/MAC architectures present equivalent values close to those from the state-of-the-art solutions, in turn validating the use of the reference architectures as a baseline for the remainder of this study.

Moreover, it is also possible to measure the exact overheads associated with the use of the quire, by directly comparing the area and power values obtained for Q and NQ setups. When comparing both setups, it is possible to ascertain that the quire structure accounts for 79% of the total chip area and 61% of the power consumption of the Posit arithmetic unit (on average). Conversely, when comparing a reference MAC unit (Q setup) with the proposed scaled accumulator (SA), it can be observed a significant reduction of area and power by 57% and 47% (on average), respectively, only at the cost of a slight increase in delay. While these results already demonstrate some of the benefits of the proposed structure, a more in-depth analysis is required by correlating them with results obtained from the presented CNN training study.

A. Area and Power Trade-Offs

From the CNN training study presented in section IV, it was observed that Posit configurations with $es = 1$ and $es = 2$ result in the highest model accuracy for all the considered precisions. Accordingly, to complement the performed studies, reference Posit FMA/MAC architectures for each setup (NQ, Q, QO, and SA) were implemented and synthesized for $n = \{6, 8, 10, 12, 16\}$ and $es = \{1, 2\}$. To ensure fair area and power comparisons, all synthesis runs were performed by targeting an operating frequency of 1 GHz. The obtained results are presented in Fig. 7A and Fig. 7B.

The first observation is the relation between the exponent size and the resulting area and power, for the different quire configurations. In particular, the plots from Fig. 7.A and Fig. 7.B show that for the specific case that does not use quire (NQ), when the exponent size increases the total area and power of the unit slightly decreases. This is mainly motivated by the reduction of the size of the multiplier, since the fraction bits are fewer when the exponent size increases (see Table I). Conversely, when using a quire (Q and QO), the power and area increase when the exponent size increases. This is a direct result of the exponential growth in the quire size depending on the exponent (see Eq. 5). These results clearly highlight the significant overheads associated with the use of the quire. Furthermore, similarly to the setups that do not use quire (NQ), the proposed scaled accumulator (SA) area and power consumption decrease when the exponent size increases. This is due to the fact that the size of the base value is fixed, independently of the exponent size (see Eq. 7).

Relevant insights can also be obtained when analyzing the quire formats from each Posit standard release (Q and QO). As it could be expected, the latest Posit release (4.12 [7]) incurs in the utilization of more chip area and higher power consumption when compared to release 4.3 [15] (QO). This is mainly due to the increase of the size of the quire for the same Posit configuration in the latest release. On the other hand, it is also clearly visible that the proposed scaled accumulator (SA) significantly reduces the chip area and power consumption, when compared to the original quire setups (Q and QO). Furthermore, as the precision increases, the observed differences also increase, further showing the benefits of the proposed structure.

To further highlight the observed hardware results, area and power efficiency studies were also conducted by considering performance-per-area (GFLOPS/mm^2) and performance-per-watt (GFLOPS/W) metrics, respectively. The calculated metrics are presented in Fig. 7.C and Fig. 7.D. The obtained results confirm the previous observations regarding the use of the quire, with Q and QO representing the less efficient setups, both in terms of area and power. Conversely, the proposed scaled accumulator allows a visible efficiency increase.

B. Accuracy Trade-Offs in CNN Training

To conclude the present study, it is important to correlate the observations gathered from the CNN training evaluation, with the hardware trade-offs discussed above.

As it was observed in Section IV-C, mixed-precision configurations allow accurate training of CNNs. This enables reducing the size of the operands, in turn potentially reducing power consumption (see Fig. 7.B) and the total operand bandwidth from registers/memory, or improved throughput by exploring data-level parallelism (using freed area to deploy more units). However, under such configurations, it was observed that the use of a quire is critical to maintaining model accuracy at low precisions. On the other hand, according to the observed area and power trade-offs, both standard quire definitions impose a significant power and hardware cost (see Fig. 7).

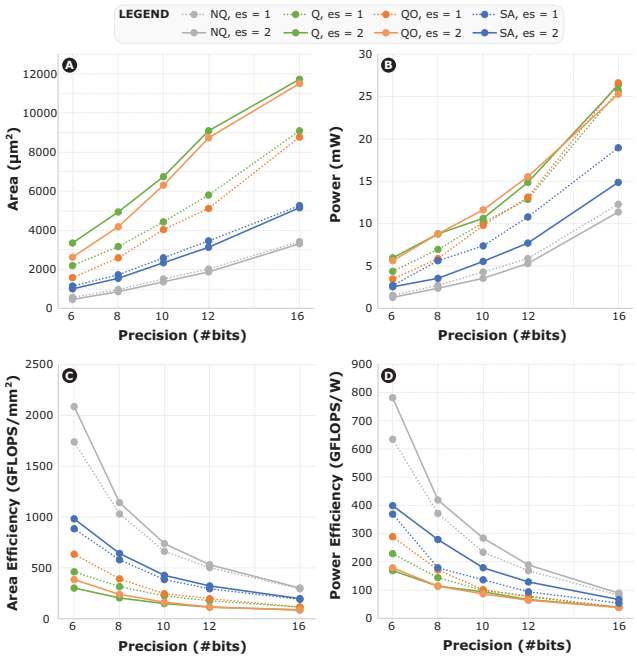

Fig. 7. Evaluation of the several reference architectures for $es = 1$ and $es = 2$ for different setups: NQ (No Quire), Q (Quire, with $cg = 31$), QO (Quire Old, with $cg = n - 1$), and SA (Scaled Accumulator). The subplots represent: A) chip area; B) power consumption; C) area efficiency; and D) power efficiency.

The mixed-precision study also concluded that even the smaller quire format (QO) has more decimal precision than what is actually necessary to achieve good model accuracy, and that a better compromise between accuracy and logic complexity can be achieved by relaxing the quire properties (see Table II). This is clearly visible when considering that the proposed scaled accumulator did not impose accuracy losses when training CNN models. However, the corresponding Posit unit setups (SA) presented much less chip area (over 70% reduction for 8-bit posits) and lower power consumption, when compared to the original quire setups (Q and QO). Also, as the precision increases, such relations also increase, further showing the benefits of reducing the quire.

C. Considerations on Mixed-precision CNN Training

While mixed-precision CNN training resulted in an overall increase in the attained model accuracies (see Section IV-C), the use of different Posit configurations in different layers conventionally requires the deployment of multiple Posit units with different configurations (e.g., 8 and 16-bit units, see Table II). This requires a careful balance between the characteristics of the deployed units to minimize the total hardware cost (see Fig.7). On custom solutions, this is a natural arrangement, such as by relying on a tensor-like processor [5], [13], [14] to perform the forward and backward passes with low-precision (e.g., Posit<8 , 2>) and a controller computing the loss and optimizer steps with higher precision (e.g., Posit<16 , 2>).

Naturally, more general-purpose solutions will require extending the datapath to consider a mixed-precision configuration. Even when considering the use of the proposed scaled accumulator, two separate Posit<16 , 2> and Posit<8 , 2> units would impose a 20% chip area and 24% power consumption increases when compared to using a single Posit<16 , 2> unit. Although it is out of the scope of this work, several solutions in related research areas could be explored to tackle this problem.

Instead of relying on the combination of multiple units, variable-precision units [11], [14], [39], [40] allow varying the computing precision by introducing dynamic datapaths that can operate in different precisions. This solution not only allows reducing the necessary chip area but it also allows parts of the circuit to be turned off when the operand precision is lowered. Alternatively, these solutions can also make a more efficient use of computing resources by reusing the unused resources in lower precisions to deploy efficient vectorization mechanisms [11], [39] and increase the computing throughput.

At this respect, the above-mentioned solutions are already being applied on transprecision computing, which relies on the principle that different applications (and phases) have different precision requirements [41]. To exploit this feature, researchers carefully analyze application phases and define optimized combinations of arithmetic units with different precisions (e.g., [18], [26]) to increase performance and energy efficiency.

VII. Conclusions

This paper presented an holistic study regarding the utilization of the Posit format for low-precision CNN training. This was done by observing the impact of different precision and exponent size configurations on the trained model accuracy, while correlating them with the hardware costs associated with the corresponding implementation of Posit FMA/MAC units. The conducted experiments were mainly focused on assessing the utilization of the Posit quire structure, and evaluating the model accuracy and hardware trade-offs associated with its use. To do so, a training accuracy study was initially performed with known CNN models, by considering the use of several quire format configurations and an alternative scaled accumulator structure, along with a mixture of low-precision Posit format configurations. The obtained results were then correlated with the associated area and power trade-offs, obtained from 28nm ASIC implementations of reference Posit FMA/MAC unit architectures for the considered setups.

The performed studies were successful in answering a set of pre-established key questions. In particular, the observed results showed that low-precision Posit formats are well-suited for CNN training (**Q1**), specifically when using different precision configurations with different CNN layers (**Q4**). It was also observed that the use of a quire only brings clear benefits to the attained model accuracy in mixed-precision scenarios (**Q2** and **Q4**), particularly when considering that this structure accounts for 79% of the total chip area and 61% of the power consumption of a Posit MAC unit (**Q3**). According to these observations, it was proposed a new scaled accumulator structure alternative to the quire, offering an average reduction of area and power by 57% and 47%, respectively, with no significant impact on the CNN model training accuracy (**Q5**).

REFERENCES

[1] J. L. Gustafson and I. T. Yonemoto, "Beating floating point at its own game: Posit arithmetic," *Supercomputing Frontiers and Innovations*, vol. 4, no. 2, pp. 71–86, 2017.

[2] R. Murillo, A. A. Del Barrio, and G. Botella, "Deep pensieve: A deep learning framework based on the posit number system," *Digital Signal Processing*, vol. 102, p. 102762, 2020.

[3] R. M. Montero, A. A. Del Barrio, and G. Botella, "Template-based posit multiplication for training and inferring in neural networks," *arXiv preprint arXiv:1907.04091*, 2019.

[4] 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 *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1421–1426, IEEE, 2019.

[5] J. Lu, C. Fang, M. Xu, J. Lin, and Z. Wang, "Evaluations on deep neural networks training using posit number system," *IEEE Transactions on Computers*, vol. 70, no. 2, pp. 174–187, 2020.

[6] G. Raposo, P. Tomás, and N. Roma, "Positnn: Training Deep Neural Networks with Mixed Low-Precision Posit," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 7908–7912, IEEE, 2021.

[7] P. W. Group, "Posit Standard Documentation," *Release 4.12-draft*, Jul. 2021.

[8] F. De Dinechin, L. Forget, J.-M. Muller, and Y. Uguen, "Posits: the good, the bad and the ugly," in *Proceedings of the Conference for Next Generation Arithmetic 2019*, pp. 1–10, 2019.

[9] L. Forget, Y. Uguen, and F. De Dinechin, "Hardware cost evaluation of the posit number system," in *Compas'2019 - Conférence d'informatique en Parallélisme, Architecture et Système*, pp. 1–7, Jun 2019.

[10] D. Mallasén, R. Murillo, A. A. Del Barrio, G. Botella, L. Piñuel, and M. Prieto-Matias, "Percival: Open-source posit risc-v core with quire capability," *IEEE Transactions on Emerging Topics in Computing*, vol. 10, no. 3, pp. 1241–1252, 2022.

[11] L. Crespo, P. Tomás, N. Roma, and N. Neves, "Unified posit/ieee-754 vector mac unit for transprecision computing," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 69, no. 5, pp. 2478–2482, 2022.

[12] N. N. Sharma, R. Jain, M. M. Pokkuluri, S. B. Patkar, R. Leupers, R. S. Nikhil, and F. Merchant, "Clarinet: A quire-enabled risc-v-based framework for posit arithmetic empiricism," *Journal of Systems Architecture*, vol. 135, p. 102801, 2023.

[13] N. P. Jouppi *et al.*, "In-datacenter performance analysis of a tensor processing unit," in *2017 ACM/IEEE 44th Annual International Symposium on Computer Architecture (ISCA)*, pp. 1–12, IEEE, 2017.

[14] N. Neves, P. Tomás, and N. Roma, "Reconfigurable Stream-based Tensor Unit with Variable-Precision Posit Arithmetic," in *2020 IEEE 31st International Conference on Application-specific Systems, Architectures and Processors (ASAP)*, pp. 149–156, IEEE, 2020.

[15] P. W. Group, "Posit Standard Documentation," *Release 4.3-draft*, Jun. 2018.

[16] Y. Nakahara, Y. Masuda, M. Kiyama, M. Amagasaki, and M. Iida, "A posit based multiply-accumulate unit with small quire size for deep neural networks," *IPSJ Transactions on System LSI Design Methodology*, vol. 15, pp. 16–19, 2022.

[17] E. T. L. Omtzigt, P. Gottschling, M. Seligman, and W. Zorn, "Universal Numbers Library: design and implementation of a high-performance reproducible number systems library," *arXiv:2012.11011*, 2020.

[18] G. Tagliavini, S. Mach, D. Rossi, A. Marongiu, and L. Benini, "A transprecision floating-point platform for ultra-low power computing," in *2018 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1051–1056, IEEE, 2018.

[19] S. Mach, D. Rossi, G. Tagliavini, A. Marongiu, and L. Benini, "A transprecision floating-point architecture for energy-efficient embedded computing," in *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1–5, 2018.

[20] "IEEE Standard for Floating-Point Arithmetic," *IEEE Std 754-2019 (Revision of IEEE 754-2008)*, pp. 1–84, 2019.

[21] U. Kulisch, *Computer arithmetic and validity*. de Gruyter, 2013.

[22] N. Wang, J. Choi, D. Brand, C.-Y. Chen, and K. Gopalakrishnan, "Training deep neural networks with 8-bit floating point numbers," in *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pp. 7686–7695, 2018.

[23] X. Sun, N. Wang, C.-Y. Chen, J. Ni, A. Agrawal, X. Cui, S. Venkataramani, K. El Maghraoui, V. V. Srinivasan, and K. Gopalakrishnan, "Ultra-low precision 4-bit training of deep neural networks," *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[24] S. Shen, Z. Dong, J. Ye, L. Ma, Z. Yao, A. Gholami, M. W. Mahoney, and K. Keutzer, "Q-bert: Hessian based ultra low precision quantization of bert," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, pp. 8815–8821, 2020.

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

[26] R. Chaurasiya, J. Gustafson, R. Shrestha, J. Neudorfer, S. Nambiar, K. Niyogi, F. Merchant, and R. Leupers, "Parameterized posit arithmetic hardware generator," in *2018 IEEE 36th International Conference on Computer Design (ICCD)*, pp. 334–341, IEEE, 2018.

[27] R. Murillo, A. A. Del Barrio, and G. Botella, "Customized posit adders and multipliers using the flopoco core generator," in *2020 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1–5, IEEE, 2020.

[28] H. Zhang, J. He, and S.-B. Ko, "Efficient posit multiply-accumulate unit generator for deep learning applications," in *2019 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1–5, IEEE, 2019.

[29] M. Arunkumar, S. G. Bhairathi, and H. G. Hayatnagarkar, "Perc: Posit enhanced rocket chip," in *Proceedings of Fourth Workshop on Computer Architecture Research with RISC-V (CARRV 2020)*, 2020.

[30] S. Tiwari, N. Gala, C. Rebeiro, and V. Kamakoti, "Peri: A configurable posit enabled risc-v core," *ACM Transactions on Architecture and Code Optimization (TACO)*, vol. 18, no. 3, pp. 1–26, 2021.

[31] S. Jean, A. Raveendran, A. D. Selvakumar, G. Kaur, S. G. Dharani, S. G. Pattanshetty, and V. Desalphine, "P-fma: A novel parameterized posit fused multiply-accumulate arithmetic processor," in *2021 34th International Conference on VLSI Design and 2021 20th International Conference on Embedded Systems (VLSID)*, pp. 282–287, IEEE, 2021.

[32] R. Murillo, D. Mallasén, A. A. Del Barrio, and G. Botella, "Energy-efficient mac units for fused posit arithmetic," in *2021 IEEE 39th International Conference on Computer Design (ICCD)*, pp. 138–145, IEEE, 2021.

[33] N. Neves, P. Tomás, and N. Roma, "Dynamic Fused Multiply-Accumulate Posit Unit with Variable Exponent Size for Low-Precision DSP Applications," in *2020 IEEE Workshop on Signal Processing Systems (SiPS)*, pp. 1–6, IEEE, 2020.

[34] F. De Dinechin and B. Pasca, "Designing custom arithmetic data paths with flopoco," *IEEE Design & Test of Computers*, vol. 28, no. 4, pp. 18–27, 2011.

[35] S. H. Leong, "Softposit," Mar. 2020. https://gitlab.com/cerlane/SoftPosit.

[36] E. Ruffaldi and F. Rossi, "cppposit," 2020. https://github.com/federicorossifr/cppposit.

[37] H. F. Langroudi, Z. Carmichael, J. L. Gustafson, and D. Kudithipudi, "Positnn framework: Tapered precision deep learning inference for the edge," in *2019 IEEE Space Computing Conference (SCC)*, pp. 53–59, IEEE, 2019.

[38] 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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pp. 726–727, 2020.

[39] H. Zhang, D. Chen, and S.-B. Ko, "Efficient multiple-precision floating-point fused multiply-add with mixed-precision support," *IEEE Transactions on Computers*, vol. 68, no. 7, pp. 1035–1048, 2019.

[40] H. Zhang and S.-B. Ko, "Efficient multiple-precision posit multiplier," in *2021 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1–5, IEEE, 2021.

[41] A. C. I. Malossi, M. Schaffner, A. Molnos, L. Gammaitoni, G. Tagliavini, A. Emerson, A. Tomás, D. S. Nikolopoulos, E. Flamand, and N. Wehn, "The transprecision computing paradigm: Concept, design, and applications," in *2018 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1105–1110, IEEE, 2018.