Unified Posit/IEEE-754 Vector MAC Unit for Transprecision Computing

***Abstract*—Transprecision computing targets energy-efficiency with multiple floating-point modules with different precisions to suit application requirements. Variable-precision architectures aim at making a more efficient hardware resource utilization, but they often rely on the IEEE-754 standard, without low-precision arithmetic support. Alternatively, the Posit format is particularly well-suited for low-precision arithmetic. However, for higher precisions, hardware requirements become prohibitive. Accordingly, this paper proposes a new unified Posit/IEEE-754 Vector Multiply-Accumulate (VMAC) unit, comprising a vectorized variable-precision datapath with shared support for the Posit and IEEE-754 formats. A 28nm ASIC implementation resulted in 50% less area and 2.9× less power consumption than typical transprecision setups.**

***Index Terms*—Floating-point arithmetic, Posit, IEEE-754, Variable-Precision, SIMD**

I. Introduction

Transprecision computing [1] has received a gradually increasing attention as a viable paradigm to cope with ever increasing performance and energy efficiency demands in modern computing systems. It is set on the principle that different applications have different precision requirements (e.g., while some physics simulations require higher than 64-bit precisions [2], deep learning applications sustain lower precisions with as little as 4 bits [3]), and that, as recent studies have shown [4–13], by lowering floating-point (FP) precision it is possible to gain straightforward acceleration and efficiency.

However, most transprecision hardware solutions [14] attempt to support different precisions by instantiating multiple arithmetic modules. This leads to an increased chip area and a waste of resources [15]. To tackle this issue, recent variable-precision arithmetic units [15–17] introduce dynamic datapaths that can operate in different precisions with the same hardware resources. To do so, they deploy a higher precision arithmetic logic (e.g., 32-bit) and allow parts of the circuit to be turned off to lower the operand precision (e.g., to 8-bit or lower [3]). While this approach provides for significant area reductions and enables straightforward Single-Instruction Multiple-Data (SIMD) capabilities [15], existing solutions are often limited by their adoption of the IEEE-754 standard [16], whose lowest supported precision is only 16 bits.

Alternatively, some recent solutions [15, 17] adopt the Posit format [6], mainly since it allows parameterizable *precision* and dynamic range (*exponent size*). The Posit format is also interesting for fused operations, since it adopts an exact accumulator structure (quire) with enough precision to avoid overflow and accuracy losses [18]. While Posit-based implementations traditionally define and fix its parameters at design-time [9, 11, 12, 19, 20], it has been shown that it is possible to support runtime-configurable exponent sizes with minimal hardware overheads [13]. This allows making use of the entire representable dynamic range for a given posit precision by specifying the exponent size of the input values. In turn, it also provides the possibility to encode a larger dynamic range, capable of supporting (within the same hardware) both values with high decimal precisions and very large magnitude.

Nevertheless, while these features make posits well suited for low-precision and transprecision computing, the overheads associated with the quire becomes prohibitive when the precision increases [19, 20]. Hence, for a general-purpose context, it is desirable to maintain compatibility with the standard IEEE-754 format, as it still is the most established FP format.

This paper proposes a new Posit/IEEE-754 Vector Multiply-Accumulate (VMAC) unit for transprecision computing. Besides combining variable-precision arithmetic and SIMD capabilities, it takes a step further from existing solutions by deploying a unified support for the IEEE-754 and Posit formats. It introduces the following contributions and features:

- an efficient variable-precision FP multiply-accumulate (MAC) 32-bit architecture for transprecision computing;
- a unified FP arithmetic architecture compatible with both the IEEE-754 and the Posit formats with support for inter-format operation and conversion, which is also compatible with the existing RISC-V Vector (RVV) [21] and recently proposed RISC-V Posit extensions [22, 23];
- a fully vectorized datapath to efficiently make use of the released hardware resources in low-precision scenarios;
- SIMD decoding/encoding modules with shared support for FP vectors encoded with *i)* dynamic posit formats with configurable exponent size; *ii)* IEEE-754 standard and low-precision non-standard formats; and *iii)* multiple scalar and vector element precisions (including 32/16/8-bit scalars and 2x16/4x8-bit vectors).

Finally, when implemented in a 28nm ASIC technology, the proposed VMAC results in 50% less area and 2.9× less power to achieve the same multiple-precision functionality when compared with typical transprecision architecture topologies [14], while supporting a unique unified FP format.

II. Background

A. IEEE-754 Standard

The IEEE-754 standard [24] defines a FP number with sign (S), biased exponent (E) and mantissa (M), with value:

$$(-1)^S \times 2^{E-bias} \times 1.M, \tag{1}$$

where *bias* is the exponent bias value. Although the standard defines formats for half- (16-bit), single- (32-bit), and doubleprecision (64-bit), it does not define a low-precision 8-bit format. However, since the proposed architecture supports 8-bit posits, for comparison purposes, an 8-bit floating point format is adopted with 4 exponent bits and 3 mantissa bits.

B. Posit Number System

The posit number system is defined by the pair $<n, es>$, where n represents the word size (*precision*) and es is the maximum *exponent* size. Eq. 2 depicts the Posit encoding:

$$\underbrace{\overbrace{s}^{sign}\quad \overbrace{r\,r\ldots\bar{r}}^{regime}\quad \overbrace{e_0\,e_1\ldots e_{es-1}}^{exponent}\quad \overbrace{f_0\,f_1\,f_2\ldots}^{fraction}}_{n\,bits} \tag{2}$$

Similarly to floats, posits include the sign, exponent, and fraction, with an additional field called regime. Contrarily to floats, whenever the sign bit corresponds to a negative number, it is necessary to take the 2's complement before decoding the remaining fields. The regime is a variable-sized field, whose encoded value (k) is given by the run-length of '0' or '1' bits.

Together with the exponent field, the k encoded value in the regime represents a scale factor of the represented value, equivalent to the exponent in floats. As a consequence of the variable-sized regime, the exponent and fraction contents are unknown before decoding the regime. In fact, depending on the run length, they can be partly (or fully) left out of the binary encoding. Hence, a posit number value is given by:

$$(-1)^{sign} \times 2^{exp+k2^{es}} \times 1.f \tag{3}$$

The Posit format has a single encoding for zero (000...0) and a single Not-a-Real (NaR) mathematical exception (100...0).

Additionally, it makes use of a 2's complement fixed-point accumulator (*quire*) based on the Kulisch accumulator, used to store sums of products of posits without rounding and accuracy loss. Naturally, the quire has a considerable hardware overhead. It is composed by 4 fields: sign, carry guard (cg), integer (int) and fraction (frac); and its size is given by:

$$quire\ size = 1 + cg + 2^{es+2} \times (n-2) \tag{4}$$

Hence, the quire must be carefully dimensioned as it grows exponentially with the exponent size and precision [20].

III. Posit/IEEE-754 VMAC Architecture

A. Overview

The proposed VMAC architecture takes a step further from existing multiple-precision arithmetic units, not only by combining variable-precision arithmetic and dynamic vectorization capabilities, but also by providing an unified support for the Posit and IEEE-754 FP formats. Accordingly, it features:

$\boxed{1}$ **Posit-based Variable-Precision Structure:** All modules of the 32-bit posit fused MAC datapath are designed to easily allow an adaptation of their arithmetic precision (at runtime), supporting 32, 16, or 8-bit operations (as illustrated in Fig. 1.A). To mitigate the hardware overheads associated with the quire, the proposed unit only provides an exact accumulation for low-precision scenarios with standard 8-bit posits ($es = 2$) [18], by using a quire of 128-bits (as opposed to 512 bits for 32-bit posit accumulation). Hence, a scale factor value is paired with the quire to ensure the correct representation of the accumulations for all the supported precisions.

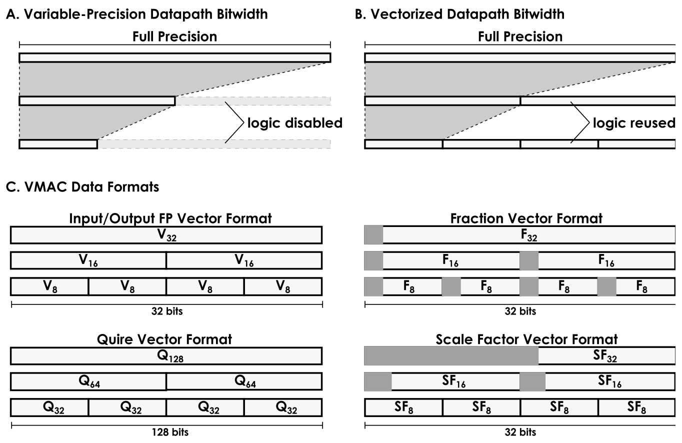

Fig. 1. Proposed VMAC (A) variable-precision and (B) vector datapath configuration schemes, together with the (C) encoded/decoded FP and quire vector data formats. Grey areas represent unused bits.

$\boxed{2}$ **Dynamic Vectorization:** All arithmetic modules are fully vectorized and configurable at runtime to support 1x32-bit, 2x16-bit, and 4x8-bit vector operations using the same hardware (see Fig. 1.B). Hence, the resources released when precision is reduced provide support for parallel computations, offering increased throughput. To support vectorization, the 32-bit input vectors are decoded into three unified vector formats that gather the sign (s), scaling factor (or exponent - sf), and fraction (f) of each vector element, according to the $(-1)^s \times 2^{sf} \times 1.f$ generic exponential format (see Fig. 1.C).

$\boxed{3}$ **Variable-Exponent Posit Configuration:** Posit exponent size can be defined at runtime (instead of being fixed at design-time), allowing most of the dynamic range for a given precision to be representable. Since the quire is already paired with a scaling factor (see Fig. 1.C), the arithmetic logic can support dynamic ranges larger than those that can be represented by the quire. Accordingly, it is only necessary to include a set of shifters to decode/encode the posit format according to the configured exponent size (described below).

$\boxed{4}$ **FP Format Unification:** While the Posit and IEEE-754 formats are fundamentally different in their representation, after decoded, both represent a FP number in the generic exponential format. As such, the logic to perform multiplication and addition/subtraction is virtually the same. Conversely, to add IEEE-754 support in a Posit base architecture, it is only necessary to add minimal decoding/encoding logic and detection for mathematical exceptions (nonexistent in the Posit format). For 8-bit precision operations, the 8-bit minifloat variant was adopted to match the equivalent Posit precision.

$\boxed{5}$ **Inter-format Operations and Conversions:** The introduced unified FP format allows the proposed unit to perform inter-format operations between equivalent Posit and IEEE-754 precisions. Since the unit's internal representation is compatible with both formats, it is only necessary to decode each operand according to their specific format (controlled by dedicated configuration signals - see below). Similarly, the

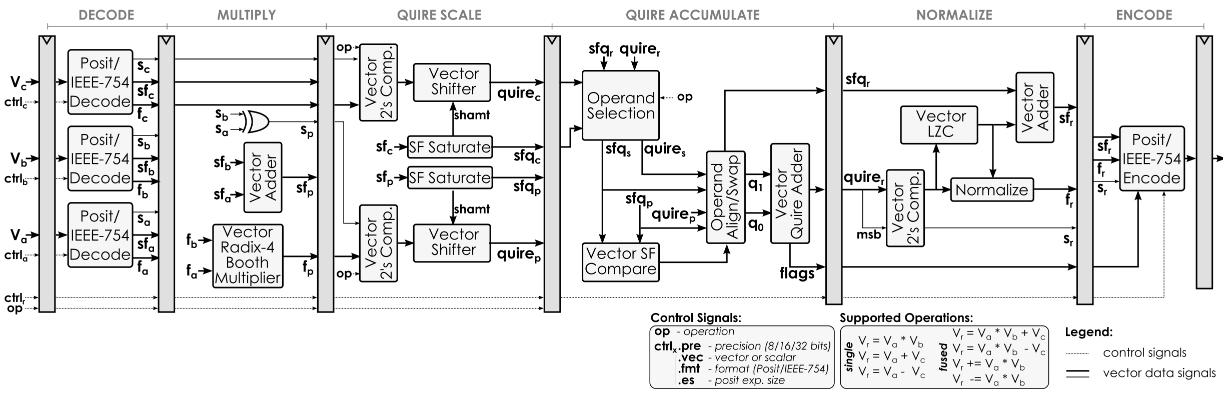

Fig. 2. Proposed Posit/IEEE-754 VMAC unit architecture diagram.

format of the output can also be configured independently of the input formats, enabling straightforward format conversions.

B. Proposed Architecture

The proposed VMAC unit (depicted in Fig. 2) comprises a fully pipelined architecture, supporting vector variable-precision FP addition, subtraction, and multiplication, together with fused multiply-add and multiply-accumulate operations. The unit deploys a 32-bit SIMD datapath with unified support for Posit and IEEE-754 FP formats, implemented by a 6-stage pipeline : *i)* `Decode`; *ii)* `Multiply`; *iii)* `Quire Scale`; *iv)* `Quire Accumulate`; *v)* `Normalize`; and *vi)* `Encode`. The unit accepts three input vector operands (V_a, V_b and V_c), and outputs one result vector (V_r), and is capable of operating with 32/16/8-bit scalar values or with 2x16/4x8-bit vectors.

The following paragraphs detail each of the pipeline stages.

Unified Decode: The `Decode` stage comprises three equivalent vectorized decoding modules (one for each input value - see Fig. 3.A), each containing the necessary logic to decode either the Posit and IEEE-754 formats, to their s, sf, and f fields. The FP format and precision are selected according to a set of control signals paired with the input value (see Fig. 2).

For the IEEE-754 format, the three fields are extracted and a bias is subtracted from the exponent value, according to the configured precision. Conversely, for the Posit format, the 2's complement is applied to 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 zero count, leaving the exponent and fraction. This value is then shifted again by es, to split the exponent and the fraction. The k value is then shift-aligned according to es and concatenated with the exponent to obtain sf. Finally, a '1' bit is added to the fraction to obtain f.

Multiplication: The `Multiply` stage implements a variable-precision vector FP multiplier, operating the decoded V_a and V_b values while propagating V_c to the next stage. To do so, the product of the fractions is performed with a 4×4 structure of 8-bit radix-4 Booth multipliers, generating 16 partial products in carry-save format, accumulated in a Wallace tree-like structure, resulting in a 64-bit value. Variable-precision and/or vectorization are applied by only enabling the required multipliers. The scale factor vectors are added

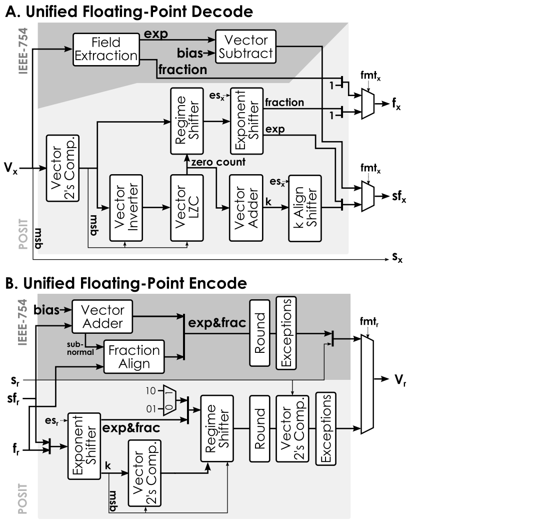

Fig. 3. Unified FP (A) decoding and (B) encoding modules, providing simultaneous support for Posit and IEEE-754 vector formats.

in a vectorized carry-lookahead adder, capable of breaking its carry-chain to perform lower-precision parallel additions. The sign vector results from a bitwise XOR of the input vectors.

Quire Scale and Accumulate: To mitigate the critical path associated with the quire processing structure, the `Quire` module is subdivided into two pipeline stages: `Scale` and `Accumulate`. The operands (V_c and the product from the previous stage) are first converted to a 128-bit fixed-point quire format vector (in accordance with the features discussed in Section III-A), paired with a scale factor vector (see Fig. 1.C).

The conversion is done in the `Scale` stage, by first taking the 2's complement of the fraction vectors and sign-extending them according to the precision. Next, the fraction is aligned to the quire fixed-point format, with a vectorized barrel shifter. Given that the quire size is limited to constrain hardware resources, the shifting amount is calculated from the scale factor dynamic range with the aid of a *saturation module* (see Fig. 2), which saturates the shifting amount (adjusting the scale factor accordingly) whenever the fixed-point value overflows.

The `Accumulate` stage is responsible for implementing the quire arithmetic logic, either by adding the values obtained from the previous stage, or by accumulating the saved quire value with one of such values. As such, the required operands are first selected and then aligned according to their scale factors. This is done by typical FP alignment logic, with the aid of a vectorized right barrel shifter, while any discarded bits are condensed in a sticky vector. Finally, the mathematical exception flags are generated (both for the Posit and IEEE-754 formats), and the result is saved in the pipeline registers.

Normalization: The `Normalize` stage is responsible for re-normalizing the quire and extracting the s, sf, and f vectors. First, the sign vector is extracted from the MSB of each quire vector element, which allows converting the quire to an unsigned value. Next, the number of shift positions required to normalize the quire is obtained with a vectorized LZC. The obtained zero count is used by a vectorized left shifter to align the unsigned quire vector. Any discarded bits are condensed in a sticky vector. Finally, the scale factor is obtained by adding the quire scale factor and the obtained zero count.

Unified Encode: Finally, the `Encode` stage provides the necessary logic for encoding the output vectors to Posit and IEEE-754 formats (see Fig. 3.B). The logic is fully vectorized and translates the s, sf, and f vectors of the result to the selected FP format vector. For the IEEE-754 format, the bias is added to the scale factor (according to the precision) and the resulting value is verified, adjusting the fraction for subnormal numbers. Afterwards, the fields of each vector element are concatenated and the fraction is rounded. The output result is selected between the rounded result, zero, infinity or canonical NaN, according to the flags generated by previous stages.

For the Posit format, sf and f are first concatenated and then right shifted, according to es, to obtain k. The k value's 2's complement is taken and the regime is shifted-in to sf and f, according to k's sign. The resulting binary value is then rounded and the 2's complement is taken according to s.

IV. Implementation Results

The 32-bit 6-stage pipeline architecture of the proposed Unified Posit/IEEE-754 VMAC unit was described in RTL and synthesized using a 28nm UMC standard cell technology [25], by targeting an operating frequency of 667 MHz, under typical operating conditions (1.05 V, 25° C). Chip area and power estimation results were obtained with Cadence Genus 19.11. The proposed unit was also validated with testing vectors generated with Sigmoid Numbers julia library [26] and TestFloat [27]. To establish a reference architectures, three 8, 16, and 32-bit Posit fused multiply-accumulate (MAC) architectures were implemented, all with exponent size of 2, as defined in the latest Posit standard [18]. These precisions imply the use of 128, 256, 512-bit internal quires, respectively. The proposed design was also compared with state-of-the-art dynamic and variable-precision units. Namely, with a 64-bit IEEE-754 variable-precision fused multiply-add (FMA) [16] (VFMA), a 32-bit Posit variable-precision multiplier [17] (VMULT), and a 32-bit Posit dynamic FMA [13] with configurable exponent size (DFMA). The synthesis results for the proposed VMAC and all the considered reference units are presented in Table I.

When compared to the reference 32-bit Posit MAC unit, it can be seen that despite the introduced variable-precision and unified FP functionality, the proposed VMAC only presents a 30% chip area increase, while showcasing a similar power consumption. Furthermore, although a lower operating frequency was expected as a result of the increased complexity, the critical path is still majorly mitigated by limiting the size of the VMAC quire to 128 bits (as opposed to the reference 512-bit quire). This is also evident when comparing with the DFMA [13], which also adopts a 512-bit quire. However, as opposed to the proposed VMAC, the higher flexibility of the DFMA [13] comes at the cost of a fixed-precision datapath, making it unsuited for transprecision computation.

TABLE I
Comparison of the proposed VMAC with the state-of-the-art.

Unit	Num. Bits	Pipel. Stages	ASIC Tech.	Delay (ns)	Area (μm^2)	Power (mW)
Ref. Posit Std. MAC	8	5	28 nm	0.65	7598	21
Ref. Posit Std. MAC	16	5	28 nm	0.8	17384	47
Ref. Posit Std. MAC	32	5	28 nm	0.91	39767	108
Proposed VMAC	**8/16/32**	**6**	**28 nm**	**1.5**	**51563**	**99**
Posit DFMA [13]	32	5	45 nm	1.5	112350	370
FP VFMA [16]	16/32/64	3	90 nm	1.5	180610	44
Posit VMULT [17]	8/16/32	-	90 nm	2.3	91861	64

While direct comparisons with the state-of-the-art variable-precision solutions are hardly possible due to the distinct implementation technologies (28nm vs. 90nm), it is still possible to compare the offered functionality. In particular, while VFMA [16] presents a variable-precision architecture, with SIMD capabilities, it is bound by its sole adoption of the IEEE-754 format, and cannot perform 8-bit low-precision operations. Contrarily, while also providing IEEE-754 support, the proposed VMAC leverages the Posit format to perform low-precision operations with a configurable dynamic range. Hence, the VMAC shows a much higher flexibility and is better suited for low-precision computation scenarios. Conversely, while the more recent VMULT [17] presents low-precision Posit support and variable-precision capabilities (similar to the proposed VMAC), it only implements the multiplier datapath and lacks the same flexibility of the VMAC in what concerns the configurable exponent size and IEEE-754 format support.

Finally, when considering the integration of the proposed VMAC in a typical transprecision architecture [14] to support a multiple-precision datapath, it is estimated that it requires 50% less area and 2.9× less power than an alternative combination of Posit MAC units that offers the same precision mix (4x8-bit + 2x16-bit + 1x32-bit MAC). Additionally, the VMAC offers an increased flexibility, by supporting a unified FP format.

V. Conclusions

This paper proposes a new unified Posit/IEEE-754 Vector Multiply-Accumulate (VMAC) unit architecture for transprecision computing. It not only offers a variable-precision datapath with SIMD processing capabilities, but also a unique support for both the Posit and IEEE-754 FP standards. Accordingly, it is capable of performing low- and high-precision Posit operations (with dynamic exponent size) without requiring a prohibitive chip area size, while maintaining compatibility with the standard IEEE-754 format. A 28nm ASIC implementation resulted in 50% less area and 2.9× less power when compared with a typical transprecision system topology.

References

[1] 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.

[2] M. Klöwer, P. D. Düben, and T. N. Palmer, "Posits as an alternative to floats for weather and climate models," in *Proceedings of the Conference for Next Generation Arithmetic 2019*, pp. 1–8, 2019.

[3] 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.

[4] 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.

[5] NVIDIA, "Nvidia tesla v100 gpu architecture.," *White paper. [Online]. Available: http://images.nvidia.com/content/volta-architecture/pdf/voltaarchitecture-whitepaper.pdf*, 2017.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

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

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

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

[19] 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.

[20] F. d. D. Luc Forget, Yohann Uguen, "Hardware cost evaluation of the posit number system," in *Compas'2019 - Conférence d'informatique en Parallélisme, Architecture et Système*, pp. 1–7, Jun 2019.

[21] A. Waterman and K. Asanovic, "RISC-V "V" Vector Extension," 2019.

[22] 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.

[23] R. Jain, N. Sharma, F. Merchant, S. Patkar, and R. Leupers, "Clarinet: A risc-v based framework for posit arithmetic empiricism," *arXiv preprint arXiv:2006.00364*, 2020.

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

[25] UMC, "UMC's 28nm high-k/metal gate stack HPCU." https://www.umc.com/upload/media/05_Press_Center/3_Literatures/Process_Technology/28nm_Brochure.pdf, 2021.

[26] I. Yonemoto, ""sigmoid numbers"," *[Online]. https://github.com/interplanetary-robot/SigmoidNumbers*, 2018.

[27] J. Hauser, "Berkeley testfloat.," *[Online]. Available: http://www.jhauser.us/arithmetic/TestFloat.html*, 2018.