Vector Multiply-Accumulate Unit for Transprecision Computing

Luís Crespo
Instituto Superior Técnico
Universidade de Lisboa
Lisboa, Portugal
luis.miguel.crespo@tecnico.ulisboa.pt

***Abstract*—Transprecision computing is currently viewed as a possible potential paradigm to increase performance and energy efficiency in modern computing systems, by allowing the floating-point precision to be tuned to the application requirements. However, most attempts at deploying transprecision architectures often rely on mutliple different modules to provide support for different precisions, leading to a waste of hardware resources and power. While variable-precision and vectorized architectures have been explored in the past to mitigiate this issue, they often have to rely on the IEEE-754 standard and lack support for low-precision arithmetic. To that end, the recent Posit number system presents a non-uniform encoding that is particularly well-suited for low-precision arithmetic. However, for higher precisions it often incurs in prohibitive hardware requirements. In this paper, it is proposed a new unified Posit/IEEE-754 Vector Multiply-Accumulate Unit, with variable-precision and SIMD computing capabilities. It implements a fully vectorized datapath with variable-precision arithmetic with a unique shared support for the Posit and IEEE-754 formats. A 28nm ASIC implementation results in 50% less area and 2.9× less power consumption, when compared to typical transprecision systems setups.**

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

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 application domains have different precision requirements (e.g., while physics simulations may require higher than 64-bit precisions [2], some deep learning applications may be executed with as low as 4-bit precisions [3]). To that end, several recent studies [4–13] have shown that by lowering floating-point (FP) precision (as much as allowed by the application requirements) it is possible to reduce: *i)* the chip area per arithmetic operator, allowing the released area to be used for additional computing and storage resources; and *ii)* the total memory storage requirements per operand, boosting the effective memory bandwidth, in turn, allowing an increased computing throughput.

However, most transprecision hardware solutions [14] rely on the instantiation of multiple modules to support different precisions, which leads to an increased chip area for FP arithmetic. However, even if the non-used modules are disabled when a given precision is considered, it still results in a waste of computing resources [15]. To tackle this issue, recent variable-precision arithmetic units [15–17] introduce dynamic datapaths that can operate in multiple and different precisions, while sharing 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 by as much as it is required by the application (e.g., to 8-bit or lower [3]). While this approach has been recognized to provide significant chip area reductions and to enable straightforward Single-Instruction Multiple-Data (SIMD) capabilities [16], existing solutions are often limited by their adoption of the IEEE-754 standard [15], whose lowest supported precision has as much as 16 bits.

Alternatively, some variable-precision solutions [16, 17] have been adopting the Posit format [6], mainly due to its low-precision computation capabilities and its fully parameterizable *precision* and dynamic range (*exponent size*). The Posit format is also particularly suited 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 the above-mentioned features make posits very-well suited for low-precision arithmetic and transprecision computing, the hardware overhead associated with the quire becomes prohibitive when the precision and exponent size increase [19, 20] and they still lack support in standard compilation frameworks. On the other hand, when considering the adoption of the transprecision paradigm on general-purpose computing contexts, it is desirable to maintain compatibility with the standard IEEE-754 format, as it still is the most established FP format.

In accordance, 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. The proposed unit introduces the following contributions and features:

- an efficient variable-precision FP multiply-accumulate 32-bit architecture, especially designed for transprecision computating;
- a unified FP arithmetic architecture compatible with both the IEEE-754 and the Posit formats with support for inter-format operation and conversion;
- a fully vectorized datapath to efficiently make use of the hardware resources that are released in low-precision computing 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 ultra-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 other transprecision architectures [14], while supporting a unified FP format with dynamic configuration.

II. Background

A. IEEE-754 Standard

The IEEE-754 Standard [21] defines a FP number consisting of three fields – sign (S), biased exponent (E) and mantissa (M). The value of a normal number is given by

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

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

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:

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

Similarly to floats, posits include the sign, exponent, and fraction 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, contrarily to floats. The regime comes after the sign bit and is a variable-sized field, whose encoded value (k) is given by the run length of 1s or 0s in the regime bits.

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

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

Additionally, the posit format also defines special encodings. A single representation for 0 (all 0 bits) and Not-a-Real (NaR) (1 followed by all 0 bits). The latter comprises all mathematical exceptions.

The posit format makes use of a 2's complement fixed-point accumulator (quire) based on the Kulisch accumulator. It can store sums of products of posits without rounding and accuracy loss. However, as with other long accumulators, it has a considerable hardware overhead. It is constituted by 4 fields: sign, carry guard (cg), integer (int) and fraction (frac). The quire size is given by:

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

However, when considering the use of a quire, it must be carefully dimensioned as tends to grow exponentially with the considered exponent size and precision [20].

III. Posit/IEEE-754 VMAC Architecture

A. Overview

The herein proposed VMAC architecture (depicted in Fig. 2) 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, the proposed unit features the following properties:

1 Posit-based Variable-Precision Architecture: The proposed unit features a 32-bit posit fused multiply-accumulate datapath. All modules are designed to allow reducing their arithmetic precision at runtime to alternate between 32, 16, and 8-bit operations (as illustrated in Fig. 1.A). To mitigate the hardware overheads associated with the use of a quire, the proposed unit only provides exact accumulation for low-precision scenarios with standard [18] 8-bit posits ($es = 2$) and an 128-bit quire (as opposed to 512 bits for 32-bit posit accumulation). As such, a scale factor value is paired with the quire to ensure the correct representation of accumulations for all the supported precisions.

2 Dynamic Vectorization: All arithmetic operators and logic modules are fully vectorized and configurable at runtime to support 1x32-bit, 2x16-bit, and 4x8-bit vector operations within the same hardware (see Fig. 1.B). This allows resources that are freed (when precision is reduced) to be reused for additional parallel computations, in turn offering increased throughput. To support the introduced variable-precision vectorization, input 32-bit vectors are decoded into

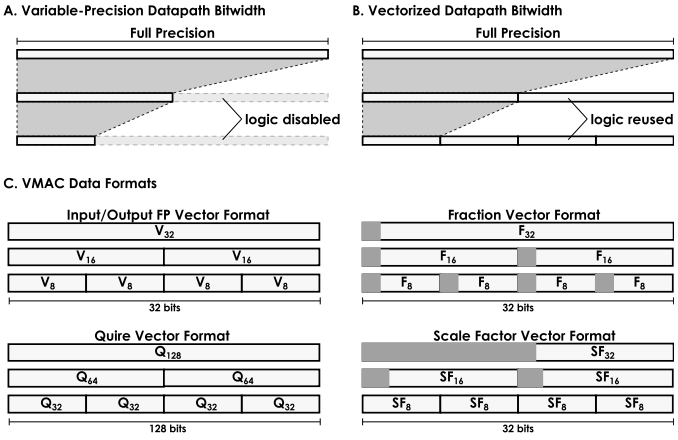

Fig. 1. Proposed VMAC (A) variable-precision and (B) vector datapath configuration schemes, together with the (C) encoded/decoded FP and quire vector data formats. Grey areas represent unused bits.

three unified vector formats that gather the sign (s), scaling factor (or exponent - sf), and fraction (f) that compose each vector element precision, according to the $(-1)^s \times 2^{sf} \times 1.f$ generic exponential format (see Fig. 1). These vectors are also paired with additional flag vectors to represent mathematical exceptions.

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 posit precision to be representable. Since, as mentioned above, the quire is already paired with a scaling factor value, the arithmetic logic can already support dynamic ranges larger than that which can be represented by the quire precision. 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).

4 FP Format Unification: While the Posit and IEEE-754 formats are fundamentally different in their representation, after being decoded, both represent a FP number in the generic exponential format. As such, the logic to perform multiplication and addition/subtraction operations is virtually the same for both formats. Conversely, to add IEEE-754 support in a Posit base architecture, it is only necessary to add minimal decoding/encoding logic and detection for IEEE-754 mathematical exceptions (not represented in the Posit format). For the particular case of 8-bit precision operations, to match the equivalent Posit precision, it is also adopted a 8-bit format (since the IEEE-754 standard does not define lower than 16-bit precisions).

5 Inter-format Operation and Conversion: 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 format of the output can also be configured independently of the input formats. As such, it is also possible to perform conversions between formats (with or without performing arithmetic operations).

B. Proposed Architecture

The proposed VMAC unit (depicted in Fig. 2) comprises a fully pipelined architecture, supporting variable-precision and vector FP addition, subtraction, and multiplication, together with fused multiply-add and multiply-accumulate operations. Accordingly, 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` + `Accumulate`); *iv)* `Normalize`; and *v)* `Encode`. The unit accepts three input 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 describe each pipeline stage in detail.

Unified Decode: The `Decode` stage comprises three equivalent vectorized decoding modules (one for each input value - see Fig. 3.A). Each decoding module contains the necessary logic for decoding FP vectors represented in the supported Posit and IEEE-754 formats, to their corresponding s, sf,and f fields. The FP, precision, and data formats are selected according to a set of control signals paired with the input value (see below).

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, it is taken the 2's complement according to the sign bit. Next, the regime run-length is decoded by means of a leading zero counter (LZC) is used (if the run-length starts with '1' the value is first inverted). Then, k is calculated and the regime is shifted out of the value, and shifted once again according to the dynamically configured exponent size. Finally, the k value is added with the exponent to obtain sf, and a '1' bit is concatenated with the fraction to obtain f.

Multiplication: The `Multiply` stage performs implements a variable-precision/vectorize FP multiplier. Multiplication is performed between the decoded V_a and V_b values while propagating V_c to the next stage. To do so, the fraction product is performed with a 4×4 structure of 8-bit radix-4 Booth multipliers, generating 16 partial products in carry-save format. These partial products are gathered through 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 scaling factor vectors are added with vectorized carry-lookahead adder, capable of breaking its carry-chain to perform lower-precision parallel additions. The sign vector is generated by performing a bitwise XOR of the input vectors.

Quire Scale and Accumulate: To mitigate the critical path associated with the Posit quire structure, the `Quire` module is subdivided into two pipeline stages, `Scale` and `Accumulate`. In this module, the operands (V_c and the

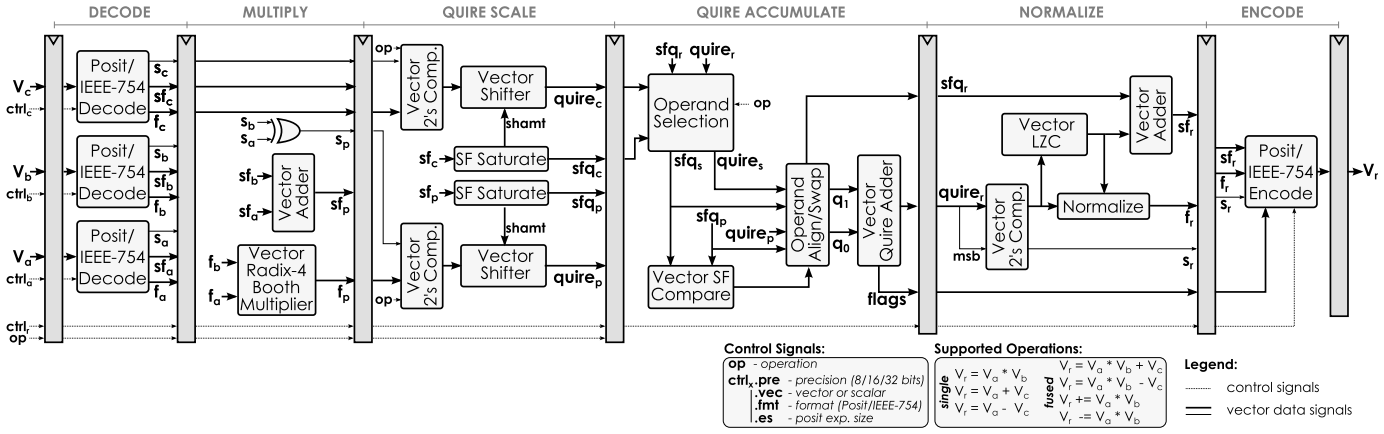

Fig. 2. Proposed Posit/IEEE-754 VMAC unit architecture diagram.

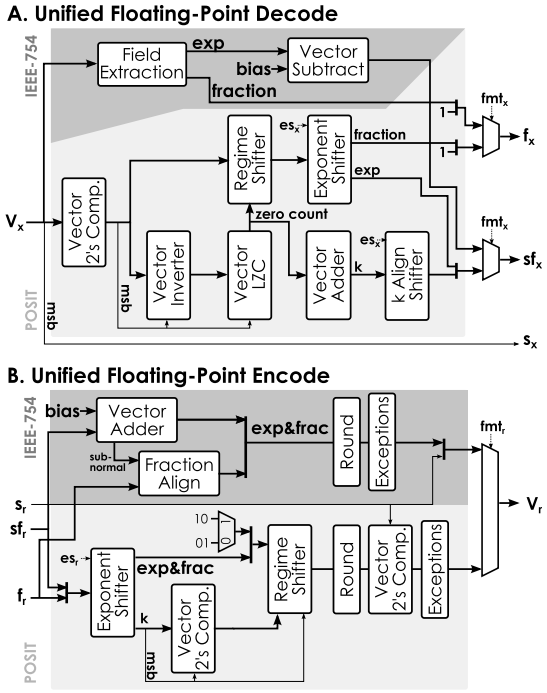

Fig. 3. Unified FP (A) decoding and (B) encoding modules, providing simultaneous support for Posit and IEEE-754 vector formats.

product from the previous stage) are first converted to a 128-bit fixed-point quire format vector, paired with a scale factor vector (see Fig. 1.C). This format was particularly dimensioned to provide a three-fold advantage over existing solutions, by: *i)* supporting higher posit exponent values without increasing the size of the quire; *ii)* keeping the hardware associated with the quire to a minimum, while still assuring exact accumulation for low-precision 16/8-bit Posit formats [18]; and *iii)* provide enough precision to allow operations over values encoded with the IEEE-754 standard.

Accordingly, in the `Scale` stage the necessary conversion is done by first taking the two'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 the aid of 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.

Subsequently, the `Accumulate` stage is responsible for implementing the quire arithmetic logic, with support for the addition of the values obtained from the previous stage, or accumulation of a saved quire value with one 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 vectors are added, mathematical exception flags are generated (both for the Posit and IEEE-754 formats), and the result 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 positions to normalize the quire is obtained with a vectorized LZC. The obtained zero count is used by a 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 zero count with an offset (due to the quire conversion).

Unified Encode: Similarly to the `Decode` stage, `Encode` stage also provides the necessary logic for encoding the proposed unit output vectors to the 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 configured FP format vector. For the IEEE-754 format, the bias corresponding to the value precision is added to the scale factor and the value is verified, adjusting the fraction for subnormal numbers. Afterwards the vector fields are concatenated and the fraction is rounded. The correct result is then 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 the k value is taken out with a vectorized barrel shifter, according to the exponent size. 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, and the sign is concatenated.

IV. Implementation Results

This section presents the main implementation results of the proposed Unified Posit/IEEE-754 VMAC unit and discusses them when compared to reference designs and solutions from the literature. A case is also made regarding its implementation in transprecision computing systems.

The 32-bit vector MAC architecture of the proposed unit was successfully implemented in RTL and synthesized for a 28nm ASIC technology, by targeting an operating frequency of 667 MHz under typical operating conditions (1.05 V, 25° C). Synthesis results for chip area and power estimation with obtained with Cadence Genus 19.11, by considering the 28nm UMC technology [22]. The functionality of all modules was verified with testing vectors generated with the help of the SigmoidNumbers julia library [23] and TestFloat [24].

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 [15]	16/32/64	3	90 nm	1.5	180610	44
Posit VMULT [17]	8/16/32	-	90 nm	2.3	91861	64

To establish a reference architecture design, three Posit fused multiply-accumulate (MAC) architectures were implemented, with 8, 16, and 32-bit precisions, all with exponent size of 2, as defined in the latest Posit standard [18]. Accordingly, the 8, 16, and 32-bit maintain 128, 256, 512-bit quires, respectively. The proposed design is also compared with state-of-the-art dynamic and variable-precision units. In particular, it is compared with a 64-bit IEEE-754 variable-precision fused multiply-add (FMA) [15] (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 units are presented in Table I.

Despite the introduced variable-precision and unified FP functionality, When compared to the reference 32-bit Posit MAC architecture, the proposed VMAC only presents a 30% chip area increase and a similar power consumption. While a lower frequency was expected as a result of the increase complexity of the circuit, 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. Despite the higher flexibility of the DFMA [13], when compared to standard architectures, as opposed to the proposed VMAC, it still presents a fixed-precision datapath and is unsuited for transprecision computation.

While direct comparisons with the state-of-the-art variable-precision solutions are hardly possible due to the differences in implementation technologies (28nm vs. 90nm), it is still possible to estimate how the proposed VMAC would match against these units. In particular, while VFMA [15] presents a variable-precision architecture, also with similar SIMD capabilities, it is bound by its sole adoption of the IEEE-754, and cannot perform 8-bit low-precision operations. Contrarily, while still providing support for the IEEE-754 standard, the proposed VMAC also leverages the Posit format to perform low-precision operations with 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 compatibility with the standard FP format.

Finally, when considering typical transprecision system architectures [14], by deploying a variable-precision datapath, the proposed VMAC requires 50% less area and 2.9× less power, when compared to a combination of Posit MAC units that offers the same precision mix (4x8-bit, 2x16-bit, 1x32-bit MAC). Additionally, the VMAC offers increased flexibility by supporting a unified FP format with dynamic configurations.

V. Conclusion

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] 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.

[16] 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.

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

[22] 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.

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

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