IJIRSET
International Journal of Innovative Research in
SCIENCE | ENGINEERING | TECHNOLOGY

e-ISSN: 2319-8753 | p-ISSN: 2347-6710

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN SCIENCE | ENGINEERING | TECHNOLOGY

Volume 10, Issue 12, December 2021

INTERNATIONAL STANDARD SERIAL NUMBER INDIA

Impact Factor: 7.569

9940 572 462 | 6381 907 438 | ijirset@gmail.com | www.ijirset.com

| e-ISSN: 2319-8753, p-ISSN: 2347-6710| www.ijirset.com | Impact Factor: 7.569|

|| Volume 10, Issue 12, December 2021 ||

| DOI:10.15680/IJIRSET.2021.1012136 |

Design of an Efficient Posit Arithmetic Generator for Deep Learning Applications

Borra Sai Dinesh [1], B.T. Krishna [2]

P.G. Scholar, Department of Electronics and Communication Engineering, University College of Engineering Kakinada, JNTU Kakinada, Andhra Pradesh, India [1]

Professor, Department of Electronics and Communication Engineering, University College of Engineering Kakinada, JNTU Kakinada, Andhra Pradesh, India [2]

ABSTRACT: With its capabilities of a better dynamic range and much more accurate outcomes than of IEEE Floats, the Posit number system has emerged as a promising alternative replacement for said IEEE 754-2008 Floating-point number system. Adders and multipliers are crucial in Deep Learning applications, and the Posit number system, itself with non-uniform data representation, yields greater benefits. For numbers with an exponent and a mantissa, this Posit number system is extremely beneficial. A new Posit MAC unit is implemented and compared to prior designs in this study. Using the Xilinx ISE tool and the Verilog HDL programming language, the Posit Arithmetic Generator featuring MAC architecture is designed and implemented.

KEYWORDS: Posit Number System, IEEE 754-2008 Floating Point Numbers, Deep Learning, Multiply and Accumulate (MAC) Architecture.

I. INTRODUCTION

The first to introduce the notion of a posit number system is [1]. It is expected to be used as a substitute for the IEEE 754-2008 floating point formats [2]. With its broader dynamic range and enhanced precision, it has shown itself to be an effective substitute for said IEEE standard in a wide range of applications. They do have a lot of benefits over floats, such like improvement of closure, bitwise identical results across several platforms, simpler hardware, and easier exception handling. Posits never overflow or underflow to zero, and "Not-a-Number" (NAN) implies an action instead of just a bit pattern. With the benefits of memory and computations, a small bit width posits format may match diverse applications. In some scientific computational scenarios, the 32-bit posit format can override the regular 64-bit format. Furthermore, its non-uniform data distribution is capable of interacting with the data distribution of a wide range of applications, including deep learning. Deep learning systems commonly employ the 8-bit or 16-bit posit formats. In its representation style, this Posit Number System involves four components: Sign, Regime, Exponent, and Mantissa. In contrast to the floating point format, the bit-width of the each element in the posit format (illustrated in Fig.1) is constantly dynamic.

"Posit (nb,es)" that's how a Posit number is expressed in short, where nb is the total bit-width and es is the exponent bit-width. The bit widths of the component bits are not consistent in design. The bit-width of the regime changes according on the application. When the sign and regime components occupy all bit spaces, the exponent and mantissa components are not present in the format. The mantissa and exponent components will occupy the remaining bit spaces if sign and regime do not occupy all the bit spaces in the maximum possible bit-width of posit format. The bit width of the mantissa (fraction) can go from 1-bit to (nb-es)-bit. A number in posits format has the following value:

$$\text{Value} = (-1)^{s} \times (\text{useed})^{rg} \times 2^{exp} \times (1 + frac)$$

Where, useed = 2^2^es.

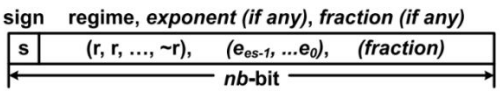

Figure 1: Format of Posit Number System.

Adder and multiplier operations contribute a significant role in the computations that take place in deep learning. Individual block power and area utilization are diminished, resulting in a decline in overall values. The

| e-ISSN: 2319-8753, p-ISSN: 2347-6710| www.ijirset.com | Impact Factor: 7.569|

|| Volume 10, Issue 12, December 2021 ||

| DOI:10.15680/IJIRSET.2021.1012136 |

multiplier and adder in the MAC unit share components through an accumulator register, where the output of the previously multiplied inputs is stored in the register and is added to the next product of multiplier and the register is stored with a new output.

II. METHODOLOGY

Figure 2 illustrates the parameterized data path of the posit multiplier block in the posit arithmetic generator. Posit component extraction, mantissa multiplier, final adder and normalization, posit component packing, and rounding are all forms the critical chain of the data path. Additionally, the sign and exponent are handled independently. The extraction of sign, regime, exponent, and mantissa components from the input data is not as simple as it is with the data in floating point representation. If it is a negative number, it is complemented in the initial stages. After then, the regime component is extracted. A run of ones or zeros is followed by a single zero or one bit respectively in the regime section.

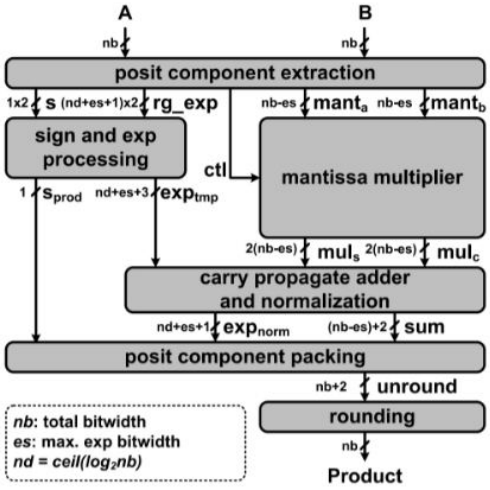

Figure 2: Data path of Posit Multiplier in Arithmetic generator.

The number of leading bits is detected using a leading zero detector (LZD) and a leading one detector (LOD). If the leading elements are ones, then value of rg equals to one less than the count; that is count -1. Else if the leading elements are zeroes, then value of rg equals to the count. Additionally a complementor, COMP is required to perform some conversions of positive count into a negative rg value based on context. Furthermore, the regime bit-width shift_rg of one more than the count that is count + 1 is constructed, allowing the regime to be eliminated and the exponent and mantissa to be acquired via a shifting operation. The bit width of a regime varies from 2-bit to (nb-1)-bit. The final extracted mantissa is (nb-es)-bit to handle all instances. A (nb-es)-bit mantissa multiplier is utilised in a posit multiplier or multiply-accumulate unit architecture. The mantissa multiplier is a modified (nb-es)-bit radix-4 Booth multiplier [7]. The block to multiply the mantissas of both the operands is built for the greatest feasible bit-width, but it is fragmented into several multipliers with lesser bit-width so that only the smaller multipliers with lesser bit-width that are necessary at a given run-time are activated. The smaller multipliers of a mantissa multiplier are controlled by a control signal, which is determined by the bit-width of regime, which may be used to calculate the mantissa bit-width. The partial product bits at the smaller multipliers are divided into several horizontal and vertical blocks and are enabled at a specific runtime with respect to bit-width of multiplier and multiplicand.

The Multiply and Accumulate Architecture is used to create the proposed Posit Arithmetic generator (MAC). Its design is based on a typical MAC architectures that are previously have been designed using IEEE floating-point; it can also be viewed as an elaborated form from fused multiply and adder units. All data path bit widths are adjustable so that multiple designs with varied nb and es values may be generated. The phases of operation of a Posit Arithmetic Generator employing MAC are outlined below. It starts with extracting Posit components from the supplied input data and ends with merely rounding the output once, rather than rounding it at each stage.

| e-ISSN: 2319-8753, p-ISSN: 2347-6710| www.ijirset.com | Impact Factor: 7.569|

|| Volume 10, Issue 12, December 2021 ||

| DOI:10.15680/IJIRSET.2021.1012136 |

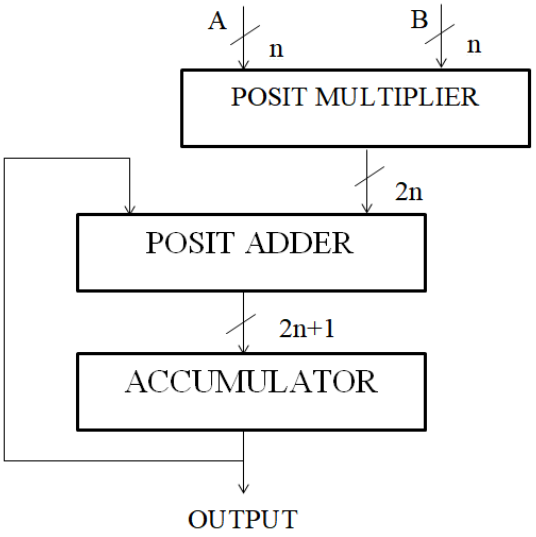

Figure 3: Theoretical Block Diagram of Posit Arithmetic Generator using MAC Unit.

- **Extraction of Posit Component:**
  Of the process in posit component extraction t he sign bit is analyzed first to see if the other sections require complementing. The complementing is done only if the number represented in posits format is negative. The regime value is then determined using the description in [1]. The operand will then be moved to the left to eradicate the regime bits. After shifting, exponent will be the most significant es-bit, and mantissa will be the remaining bit prefixed with implicit bit. The regime component and the exponent component are also integrated in this module. To obtain the effective exponent, the regime value rg is first left shifted with es-bit and then combined with exponent. This effective exponent will be used to transfer control in later phases.

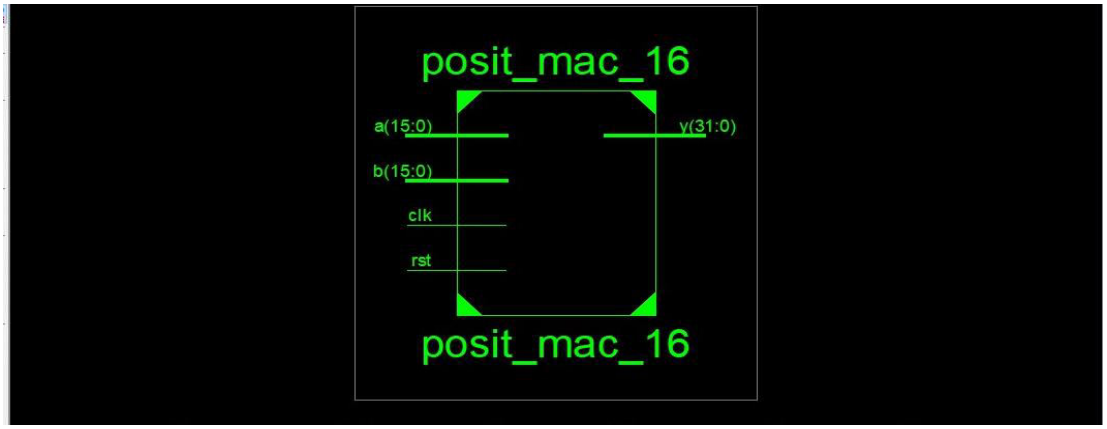

Figure 4: Logic Symbol of 16 bit Posit Arithmetic Generator.

| e-ISSN: 2319-8753, p-ISSN: 2347-6710| www.ijirset.com | Impact Factor: 7.569|

|| Volume 10, Issue 12, December 2021 ||

| DOI:10.15680/IJIRSET.2021.1012136 |

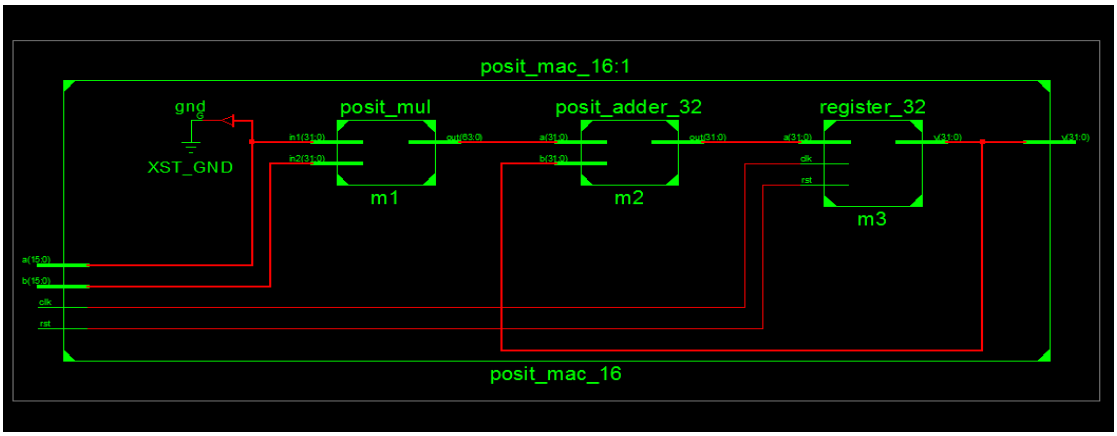

Figure 5: RTL Schematic of Posit Arithmetic Generator.

- **Mantissa Multiplication:**
  The Mantissa multiplier block is an unsigned multiplier of bit-width nb-es. The radix-4 modified Booth multiplication method is being used to implement multiplication operation on mantissas of both components. As of horizontal and vertical blocks, the mantissa multiplier is fragmented into numerous areas of partial products.

- **Addition:**
  The product of the mantissa multiplier will be merged with the 2(nb-es)-bit-width preceding output value which is of least significant using the carry save adder. A carry propagate adder then adds up the resultant total vector and carry vector. An incrementer to add the most significant (nb-es)+2-bit positions is used to align the preceding output value . The lower order adder's output carry determines whether or not the higher portion has to be incremented. The adder result is formed by combining the two component outcomes. If the outcome is negative, it must be complemented.

- **Normalization:**
  A two-stage shifter is used to execute the normalization shifter, with a constant shifter then followed by a dynamic shifter. If the operands' exponent difference expdiff is less than 2, a constant shifting with a quantity (nb-es)+1 is done initially, accompanied by a dynamic shifting with an amount counted by LZA (Leading Zero Anticipator). However, a dynamic shifter with an amount equal to the alignment amount is utilised instead of a constant shifter. A 1-bit right shift is applied if overflow occurs at any case. The exponent will be adjusted in accordance with the amount of normalizing shifting.

- **Processing and Rounding of Output:**
  The result is rounded in the end, once all of the necessary computations have been done and processed. All of the posit components are aggregated and blended in the output of the adder and normalizer. The components regime and exponent respectively are extracted by splitting the resultant grouped exponent into two parts. In addition, an nb-bit regime vector is formed. The exponent will be prefixed to the resultant mantissa portion, followed by the regime vector. After that, a shifting right operation is made to combine the vector based on the regime value. A 3(nb-es)+2-bit vector will arise as a result from the addition operation. This will be normalized before being reduced to the least significant 2(nb-es)+1-bit. Lastly, the round to the closest is performed according to the regime bit-width, and the output is represented in the general format of the posit number system.

III. RESULTS AND ANALYSIS

The Posit Arithmetic Generator with Multiply and Accumulate Unit is engineered to help utilising the posit number format in deep learning applications as easier as it is possible with its capability. Posit Arithmetic Generator's Verilog codes with various exponent bit-widths have been created. The area and power utilization of any combination of parameters among the variations possible by the various number systems are decreasing as

| e-ISSN: 2319-8753, p-ISSN: 2347-6710| www.ijirset.com | Impact Factor: 7.569|

|| Volume 10, Issue 12, December 2021 ||

| DOI:10.15680/IJIRSET.2021.1012136 |

long as the exponent bit-width goes on increasing. It is so that since the greater the exponent bit-width, the smaller the mantissa bit-width will be. As a result, both the mantissa multiplier and the final adder's bit widths will be shortened.

Bit-Width of "es"	Area (um^2)	Power (mW)
1-bit	3750	3.3
2-bit	3700	3.2
3-bit	3600	3.1
4-bit	3500	2.7
5-bit	3450	2.5

Table 1: 16-bit Posit Arithmetic Generator's area and power metrics with various exponent bit-widths.

The majority of posit MAC and floating-point MAC's core processes seem to be the same. Nonetheless, posit has a difficult input process for obtaining each component from the posit format, as well as a difficult output process for packing the resulting component back into the posit format. When compared to floating-point designs, this results in increased latency and higher resource usage with the same overall bit-width. In deep learning, moreover, 8-bit posit may reach 32-bit IEEE floating-point performance with considerably greater hardware efficiency, precision, and data range.

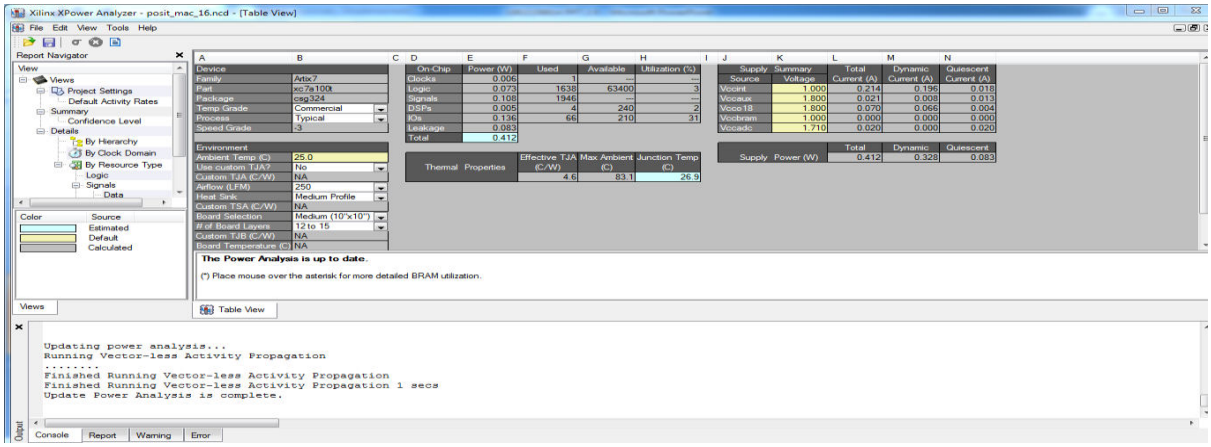

Figure 6: Power Analysis of 16-bit Posit Arithmetic Generator.

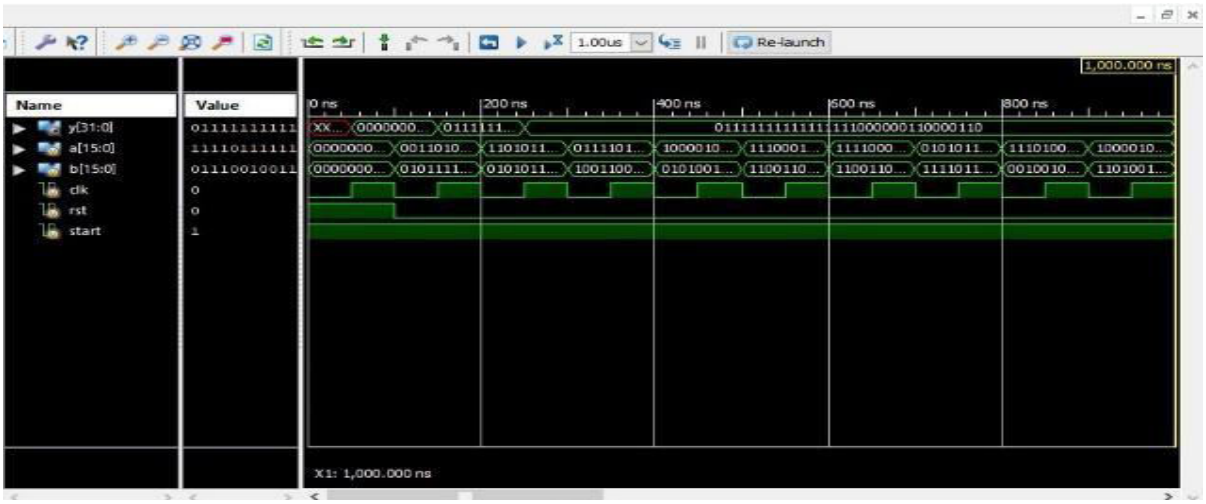

Figure 7: Simulation result of a 16-bit Posit Arithmetic Generator.

| e-ISSN: 2319-8753, p-ISSN: 2347-6710| www.ijirset.com | Impact Factor: 7.569|

|| Volume 10, Issue 12, December 2021 ||

| DOI:10.15680/IJIRSET.2021.1012136 |

IV. CONCLUSION

Due to the added control signal provided for the mantissa multiplier, the architecture with posit number system has a somewhat longer delay than other models of MAC units with normal floating point number formats and others. This also results in a somewhat bigger surface area physically or virtually. The adder contributes very little to the total power usage. Hence, t he power reduction in technological improvements is mostly attributed to the mantissa multiplier module, as the control signal simply enables a needed portion of the mantissa multiplier at a certain run time based on the total bit-width and exponent bit-width. A Posit Arithmetic Generator with MAC Unit, on the other hand, can result in a greater accuracy and better dynamic range environment for Deep Learning applications by trading benefits over time consumption. Pipelining approach may be applied to the posit MAC design in order to boost the throughput of the posit MAC unit in real world applications.

REFERENCES

[1] J. L. Gustafson and I. Yonemoto, “Beating floating point at its owngame: Posit arithmetic,” Supercomput. Front. Innovat. Int. J., vol. 4, no. 2, pp. 71–86, Jun. 2017.
[2] IEEE Standard for Floating-Point Arithmetic, IEEE Standard 754-2008, Aug. 23, 2008, pp. 1–70.
[3] Z. Carmichael, S. H. F. Langroudi, C. Khazanov, J. Lillie, J. L. Gustafson, and D. Kudithipudi, “Deep positron: A deep neural network using the posit number system,” CoRR, vol. abs/1812.01762, pp. 1–6, Dec. 2018.
[4] J. Johnson, “Rethinking floating point for deep learning,” CoRR,vol. abs/1811.01721, pp. 1–8, Nov. 2018.
[5] M. K. Jaiswal and H.-K. So, “Architecture generator for type-3 unum posits adder/subtractor,” in Proc. IEEE Int. Symp. Circuits Syst. (ISCAS), Florence, Italy, May 2018, pp. 1–5.
[6] Podobas and S. Matsuoka, “Hardware implementation of POSITs and their application in FPGAs,” in Proc. IEEE Int. Parallel Distrib. Process. Symp. Workshops (IPDPSW), Vancouver, BC, Canada, May 2018, pp. 138–145.
[7] A. D. Booth, “A signed binary multiplication technique,” Quart. J. Mech. Appl. Math., vol. 4, no. 2, pp. 236–240, 1951.
[8] Z. Carmichael, H. F. Langroudi, C. Khazanov, J. Lillie, J. L. Gustafson, and D. Kudithipudi, “Performance-efficiency trade-off of low-precision numerical formats in deep neural networks,” in Proc. Conf. Next Gener. Arithmetic, Mar. 2019, pp. 1–9.
[9] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” NATURE, vol.521, pp. 436–444, May 2015.
[10] M. S. Schmookler and K. J. Nowka, “Leading zero anticipation and detection-a comparison of methods,” in Proceedings 15th IEEE Symposium on Computer Arithmetic. ARITH-15 2001, June 2001, pp. 7–12.

INNO SPACE
SJIF Scientific Journal Impact Factor

Impact Factor:
7.569

doi cross ref

ISSN INTERNATIONAL STANDARD SERIAL NUMBER INDIA

निस्केयर NISCAIR

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN SCIENCE | ENGINEERING | TECHNOLOGY

9940 572 462 6381 907 438 ijirset@gmail.com

www.ijirset.com

Scan to save the contact details