iJRASET
International Journal For Research in Applied Science and Engineering Technology



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 11 **Issue:** V **Month of publication:** May 2023

DOI: https://doi.org/10.22214/ijraset.2023.51597

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 11 Issue V May 2023- Available at www.ijraset.com

Design of Power Efficient Posit Multiplier using Compressor Based Adder

Mr. N. Senathipathi[1], R. Rasiha[2], R. Sadhurya[3], S. Sangeetha[4]
[1]*Assistant professor,* [2, 3, 4]*UG Scholar, Electronics and Communication Engineering, P.A. College of engineering and technology, Pollachi, Tamilnadu*

Abstract: ***Posit number system has been used in many applications, especially the deep learning. Because of how well its non-uniform number distribution aligns with deep learning's data distribution, deep learning's training process can be sped up. The hardware multiplier is typically built with the widest mantissa bit-width available due to the flexibility of posit numbers' bit-width. Such multiplier designs consume a lot of power since the mantissa bit-width is not necessarily the maximum value. This is especially true when the mantissa bit-width is tiny. The mantissa multiplier is still built to have the widest bit-width feasible, but it is broken into numerous smaller multipliers. At run-time, just the necessary tiny multipliers are turned on. The regime bit-width, which can be used to determine the mantissa bit-width, controls those smaller multipliers. This design technique is applied to 8-bit, 16-bit, and 32-bit posit formats.***

Keywords: posit number system component; formatting; style; styling; insert (key words)

I. INTRODUCTION

A new datatype called Posit was created to completely supplant IEEE Standard 754 floating-point numbers. (floats). Plots do not require interval arithmetic or variable size, in contrast to previous iterations of universal number (unum) arithmetic. operands; similar to fractions, they round if a response is not precise. Larger dynamic range, better accuracy, better closure, bitwise identical results across systems, simpler hardware, and easier exception management are just a few of the compelling benefits they offer over floats. Posits never go over or under their limits, and "Not-a-Number " (NaN) denotes an action rather than a bit sequence. An IEEE float FPU requires more hardware than a posit processing unit. Furthermore, its uneven data distribution blends well with the uneven data distribution of some uses, including deep learning. In deep learning algorithms, the 8-bit or 16-bit posit formats are frequently employed. In some scientific computation uses, the 64-bit floating-point format is replaced by the 32-bit posit format. A posit number is defined as Posit (nb,es), where nb is the overall bit width and es is the exponent bit width. Sign (s), regime (rg), exponent (exp), and mantissa are its four constituents. (frac). The bit-width of the component varies. For various values, the regime bit-width changes.

value = (−1)s × useedrg × 2exp × (1 + frac)

where used = 2^2res

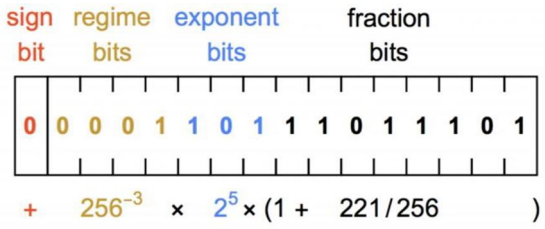


Figure 1: Posit number system

In contrast to the traditional floating-point format, each component of the posit format, as shown in Fig. 1, has a dynamic bit breadth (aside from the single-bit sign). The format is always used for the sign and the protocol. Only when the sign and regime do not take up all of the bit positions do the leftover exponent and fraction part become visible. Since nb is the overall bit-width of the posit format and es is the exponent bit-width, the mantissa's bit-width (which includes the implicit bit) can range from 1 bit to (nb es)-bit. The mantissa does not always require a (nb es)-bit multiplier because the actual bit-width of the mantissa is not always the maximum number. Power or energy is lost when using a (nb es)-bit multiplier for a tiny bit-width mantissa.

©IJRASET: All Rights are Reserved

In this article, we demonstrate how a 16-bit posit multiplier, in which the mantissa (fraction) multiplier is divided into several smaller multipliers, can be successfully implemented. At runtime, only the necessary factors are applied. So, to increase power efficiency, we effectively create a low-power posit multiplier architecture. To increase the power efficiency of the related component or device, this architecture can also be used in multipliers other than posit multipliers.

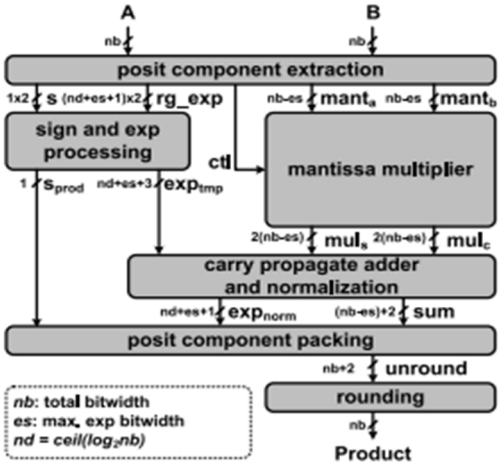


Figure 2: Datapath of the posit multiplier

II. EXISTING METHOD

The n-bit adder built from n one-bit full adders is known as a ripple carry adder, because of the way the carry is computed. Each full adder inputs a Cin, which is the Cout of the preceding adder. This kind of adder is called a ripple carry adder, since each carry bit "ripples" to the next full adder. It is shown in figure 3.

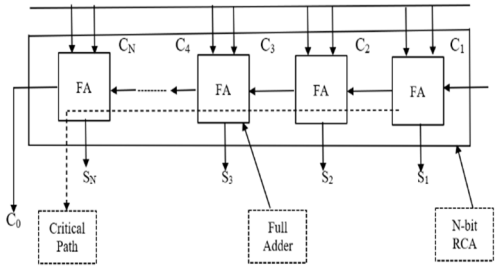


Figure 3: 4-Bit ripple carry adder

A ripple carry adder is a digital circuit that produces the arithmetic sum of two binary numbers. It can be constructed with full adders connected in cascaded with the carry output from each full adder connected to the carry input of the next full adder in the chain. The interconnection of four full adder (FA) circuits to provide a 4-bit ripple carry adder. But here, it will give the output for whatever the input bit sequences with some delay. As per the digital circuits if the circuit gives output with delay won't be preferable. This can be overcome by a carry look-ahead adder circuit.

The 4-bit-carry-skip adder consists of a *n*-bit-carry-ripple-chain, a *n*-input AND-gate and one multiplexer. It is shown in figure 4.

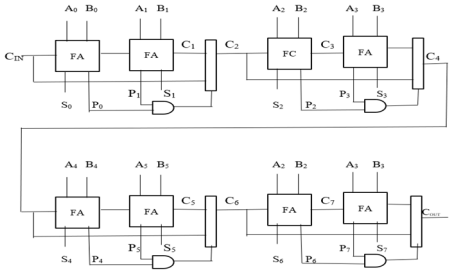


Figure 4: 4-Bit carry skip adder

Each propagate bit, that is provided by the carry-ripple-chain is connected to the *n*-input AND-gate. The resulting bit is used as the select bit of a multiplexer that switches either the last carry-bit or the carry-in to the carry-out signal This greatly reduces the latency of the adder through its critical path, since the carry bit for each block can now "skip" over blocks with a *group* propagate signal set to logic 1.

The carry-select adder generally consists of ripple-carry adders and a multiplexer. Adding two n-bit numbers with a carry-select adder is done with two adders (Therefore two ripple-carry adders), in order to perform the calculation twice, one time with the assumption of the carry-in being zero and the other assuming it will be one. After the two results are calculated, the correct sum, as well as the correct carry-out, is then selected with the multiplexer once the correct carry-in is known. It is shown in figure 5.

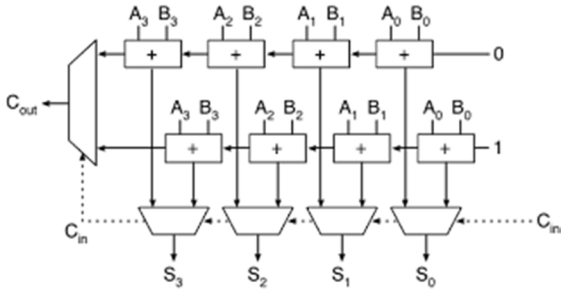


Figure 5: 4-Bit carry select adder

A carry select adder is an arithmetic combinational logic circuit which adds two N-bit binary numbers and outputs their N-bit binary sum and a 1-bit carry.

A carry look-ahead adder definition is it is the faster circuit in performing binary addition by using the concepts of Carry Generate and Carry Propagate. It is shown in figure 6.

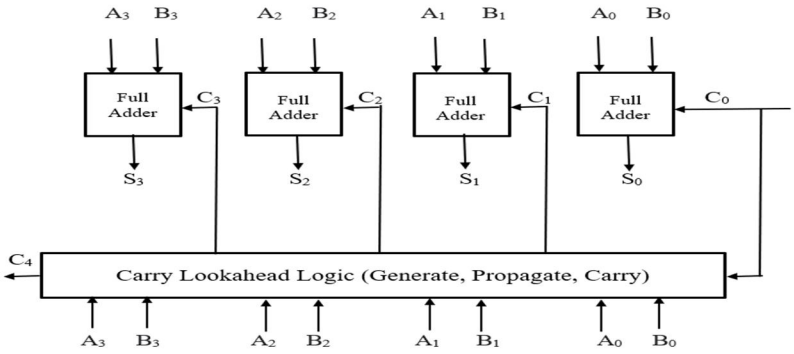


Figure 6: 4-Bit carry look adder

A CLA is termed as the successor of a ripple carry adder. A CLA circuit minimizes the propagation delay time through the implementation of complex circuitry.

A. Drawbacks Of Existing Method

Time consumption consuming under existing method is 18.79 ns. The power consumption is 24.00 watts. The proposed work overcome these results and time delay.

III. PROPOSED METHOD

One of the major speed enhancement techniques used in modern digital circuits is the ability to add numbers with minimal carry propagation. The basic idea is that three numbers can be reduced to 2, in a 3:2 compressor, by doing the addition while keeping the carries and the sum separate. This means that all of the columns can be added in parallel without relying on the result of the previous column, creating a two output & quot; adder & quot; with a time delay that is independent of the size of its inputs.1 bit 3:2 compressor is shown in figure 7.

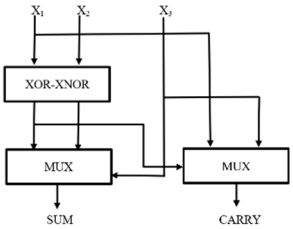


Figure 7: 3:2 Compressor

The sum and carry can be recombined in a normal addition to form the correct result. This process may seem more complicated and pointless, but the power of this technique is that any amount, number of additions can be added together in this manner. It is only the final recombination of the final carry and sum that requires a carry propagating addition. 3:2 compressor is also known as full adder. It adds three one bit binary numbers, a sum and a carry. The full adder is usually a component in a cascade of adders. The carry input for the full adder circuit is from the carry output from the cascade circuit. Carry output from full adder is fed to another full adder. It is used for low power consumption.

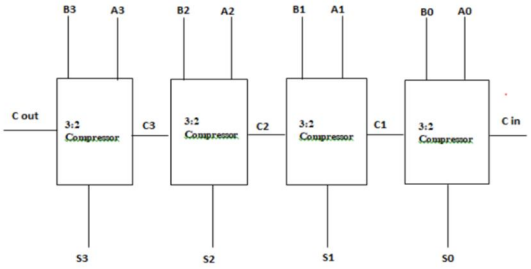


Figure 8: 4-Bit compressor adder

Time consumption consuming under proposed method is 14.19 ns. The power consumption is 20.00 watts.

IV. APPLICATION

Audio and video processing: Posit multipliers can be used in audio and video processing systems to perform multiplication operations on signals such as sound or video data. This can help improve the accuracy and fidelity of the processed signals. Machine learning: Machine learning algorithms often involve large numbers of multiplication operations, which can be computationally expensive. Using posit arithmetic and posit multipliers can help reduce the computational cost of these operations while maintaining accuracy. Cryptography: Posit arithmetic can be used in some cryptographic algorithms, such as elliptic curve cryptography. Using posit multipliers can help improve the efficiency and security of these algorithms. Scientific computing: Posit arithmetic and posit multipliers can be used in scientific computing applications where high precision and accuracy are required. This can include simulations, modeling, and other numerical analysis tasks.

V. RESULT AND ANALYSIS

The proposed and existing designs are modeled in Verilog HDL. These Verilog HDL models are simulated/verified using the Xilinx ISE simulator.

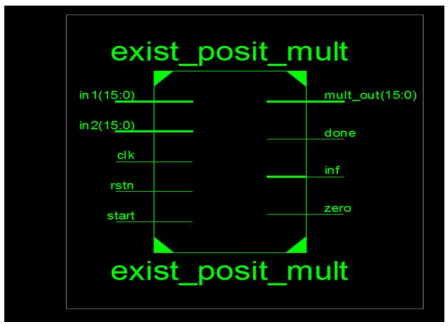


Figure 9: RTL Schematic of existing posit multiplier

```
Device utilization summary:
---------------------------

Selected Device : 6slx100tcsg484-2


Slice Logic Utilization:
 Number of Slice Registers:             136  out of  126576     0%
 Number of Slice LUTs:                  546  out of  63288      0%
    Number used as Logic:               546  out of  63288      0%

Slice Logic Distribution:
 Number of LUT Flip Flop pairs used:    577
   Number with an unused Flip Flop:     441  out of    577     76%
   Number with an unused LUT:            31  out of    577      5%
   Number of fully used LUT-FF pairs:   105  out of    577     18%
   Number of unique control sets:        10

IO Utilization:
 Number of IOs:                         54
 Number of bonded IOBs:                 54  out of    296     18%

Specific Feature Utilization:
 Number of BUFG/BUFGCTRLs:               1  out of     16      6%
 Number of DSP48A1s:                     1  out of    180      0%
```

Figure 10: Design summary of existing posit multiplier

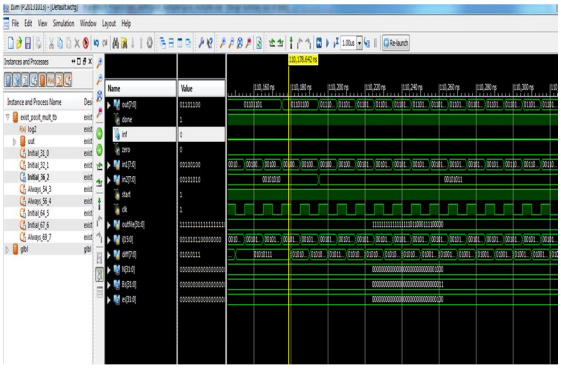


Figure 11: Simulation of existing posit multiplier

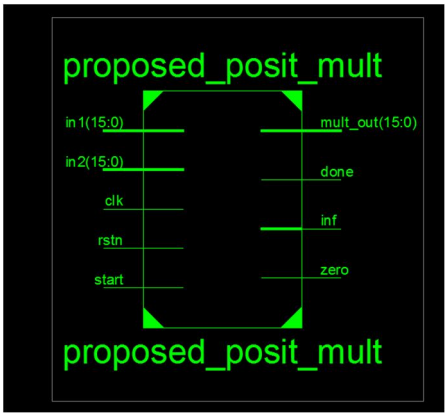


Figure 12: RTL Schematic of proposed posit multiplier

```
Device utilization summary:
---------------------------

Selected Device : 6slx100tcsg484-2


Slice Logic Utilization:
 Number of Slice Registers:             135  out of  126576     0%
 Number of Slice LUTs:                  532  out of  63288      0%
    Number used as Logic:               532  out of  63288      0%

Slice Logic Distribution:
 Number of LUT Flip Flop pairs used:    554
   Number with an unused Flip Flop:     419  out of    554     75%
   Number with an unused LUT:            22  out of    554      3%
   Number of fully used LUT-FF pairs:   113  out of    554     20%
   Number of unique control sets:        10

IO Utilization:
 Number of IOs:                         54
 Number of bonded IOBs:                 54  out of    296     18%

Specific Feature Utilization:
 Number of BUFG/BUFGCTRLs:               1  out of     16      6%
 Number of DSP48A1s:                     1  out of    180      0%
```

Figure 13: Design summary of proposed posit multiplier

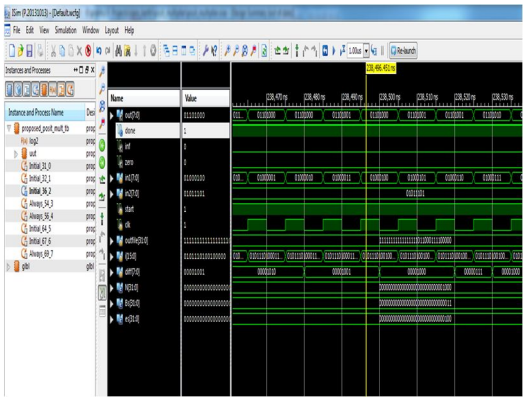


Figure 14: Simulation of proposed posit multiplier

Performance Comparision

S.no	Parameters	Adder based posit multiplier	Compressor based posit multiplier
1	LUT's	546	532
2	Delay in ns	18.791	14.192
3	Power in Watts	24.00	21.00

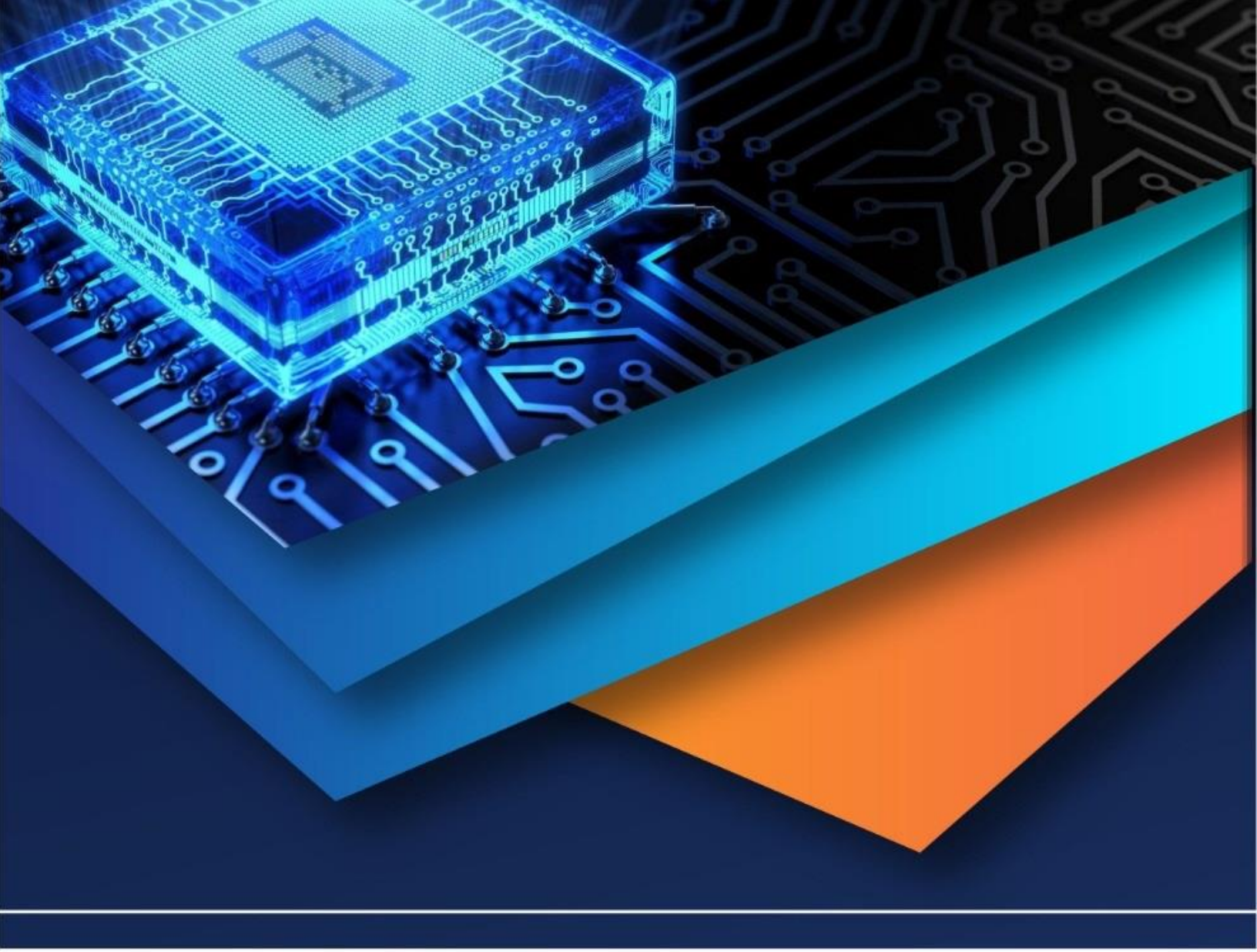
VI. CONCLUSION

The idea proposed in the paper is a 32-bit Posit multiplier architecture with power efficiency. Intrigued by the idea of reconstructing the multiplier unit for mantissa into smaller parts, because of the whole mantissa unit is not used entirely all the time, we have built the hardware. To limit the power consumption, we use only the necessary potion of the multiplier. Our method is evaluated for 16-bit multiplier, whereas we can extend the work for 8-bit and 32-bit posit multipliers using the same technique. For futuristic purposes, more power reduction techniques for multiplier architecture can be developed. The work need not be necessarily limited to multipliers alone. Future works can be deployed also for Posits Adder or Posits Multiply Accumulate functions.

REFERENCES

[1] J. Johnson, “Rethinking floating point for deep learning,” CoRR, Nov. 2018.
[2] R. Chaurasiya et al., “Parameterized posit arithmetic hardware generator,” in Proc. IEEE 36th Int. Conf. Comput. Design (ICCD), Orlando, FL, USA, Oct. 2018.
[3] M. K. Jaiswal and H.-K. So, “Architecture generator for type-3 unum posit adder/subtractor,” in Proc. IEEE Int. Symp. Circuits Syst. (ISCAS), Florence, Italy,May 2018.
[4] 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.
[5] 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,” Dec. 2018.
[6] M. Klöwer, P. D. Düben, and T. N. Palmer, “Posits as an alternative to floats for weather and climate models,” in Proc. Conf. Next Gener. Arithmetic, Mar. 2019.
[7] H. Zhang, J. He, and S.-B. Ko, “Efficient posit multiply-accumulate unit generator for deep learning applications,” in Proc. IEEE Int. Symp. Circuits Syst. (ISCAS), Sapporo, Japan, May 2019.
[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.
[9] Uguen, Yohann, Luc Forget, and Florent de Dinechin. "Evaluating the hardware cost of the posit number system." 2019 29th International Conference on Field Programmable Logic and Applications (FPL). IEEE, 2019.
[10] Murillo, R., Del Barrio, A. A., Botella, G., Kim, M. S., Kim, H., & Bagherzadeh, N. (2021). PLAM: A posit logarithm-approximate multiplier. IEEE Transactions on Emerging Topics in Computing.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089 (24*7 Support on Whatsapp)