# Design and Implementation of POSIT Based Adder and Multiplier in Verilog HDL

*Rambabu* Sanivarapu*[1], *Mallikarjuna Rao* Y[2], *Venkataiah* C[3], *Linga Murthy* MK[4], *Laith H.* Alzubaidi[5], *Vyeshikha*[6]

[1,2]Department of Electronics & Communication Engineering, Santhiram Engineering College, Nandyal, Andhra Pradesh, 518501 INDIA
[3]Department of Electronics & Communication Engineering, Rajeev Gandhi Memorial College of Engineering and Technology, Nandyal, Andhra Pradesh, 518501 INDIA
[4]Department of Electronics & Communication Engineering, LakiReddy Bali Reddy College of Engineering, Mylavaram, Andhra Pradesh, 518501 INDIA
[5]The Islamic University, Faculty of Engineering, Najaf, Iraq
[6] Uttaranchal School of Computing Sciences, Uttaranchal University, Dehradun 248007 INDIA

**Abstract.** Due to recent developments, the POSIT number system, winch has been planned as a successor for numbers that are expressed in IEEE floating-point, which are in the focus of advances in arithmetic. Although this format claims to deliver more precise outcomes with the same bit width as ordinary floating point, the duration of the operation fluctuation during posit field identification poses a hardware design problem. The POSIT-based MAC Unit is created using Verilog HDL in this study, and the designed architecture is evaluated for good operation before being implemented on an FPGA using Xilinx Vivado.

Keywords: POSIT, MAC Unit, and VHDL

## I INTRODUCTION

POSIT is a new data type that is intended to replace IEEE Standard 754 floating-point integers directly [1][2]. POSITs do not necessitate the use of arithmetic of regular intervals or operands of varying sizes, unlike previous types of of universal number arithmetic. The following factors like increased scaling factor, increased precision, improved closure, similar bitwise results among existed platforms, less complicated hardware, and easy exception handling are just a few of the advantages they offer over floats. Posits does not overflow or underflow to infinity. In addition, "Not-a-Number" denotes a procedure rather than a bit pattern [4].

*Corresponding author: ramababu.ece@srecnandyal.edu.in

## 1.1 The Uniform Number Format(UNF)

The universal number format (UNF), which is one of the floating-point-like arithmetic styles, that was gaining traction as a substitute for IEEE 754[17]. This article goes through the posit number format in great depth. Both real numbers and real number ranges are expressed using the universal number format.

The original Type I universal is a superset of floats, just as floats are of integers. When a computation is unable to deliver a numerically accurate response and ordinary floating-point arithmetic rounding is required, these can either indicate an exact float or an open interval between adjacent floats. [5]. For accomplish this, universal numbers incorporate a "universal bit" once the fraction comes to a close that indicates if the fraction represents an exact amount or a range, depending on whether the universal bit which is similar to 0 or 1.

Components which are included in IEEE 754 floating-point are like the first field is sign field and second field is exponent and the last field is fraction bit section which is also supported by the Type I universal number format. To solve some of the shortcomings of the previous version, the Type II universal number was suggested, such is the complexity of hardware design and the fact that some values can be expressed in a variety of ways. IEEE floats are no longer compatible with this second version [15][16].

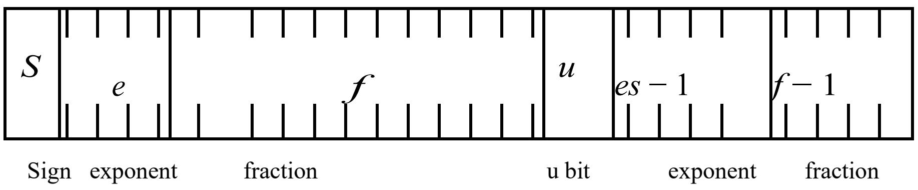


**Fig. 1**. Type I u num bit fields

To solve some of the shortcomings of the previous version, the Type II universal number was suggested, that means with the difficulty of hardware implementation and the fact that some values can be expressed in a variety of ways. IEEE floats are no longer compatible with this second version. Type II universal numbers, on the other hand, demonstrate a simple, mathematical design based on the translation of values onto one real spatial line, that was indicated by the field R = R. The basic idea is that the position where the signed (two's complement) numbers turn negative is the same point at which positive real numbers turn negative, and that point gives the value.

The concept of a Type III universal number is thus on the basis of a genuine projective line, just as it is for Type II, albeit the this format's developed system would be similar to that of IEEE 754 floating-point arithmetic based on an actual spatial line[7][8]. The reciprocals are obtained by relaxing the perfect reflection criterion, which now only applies to integer powers of 2 and 0. Because all of the integers are also of the form m 2k, where m and k are integers, there seem to be no empty intervals. A good POSIT is indeed the interval arithmetic variant of the POSIT. It's made up of two hypotheses of equal size, each terminating in a universal bit that indicates the limits [4].

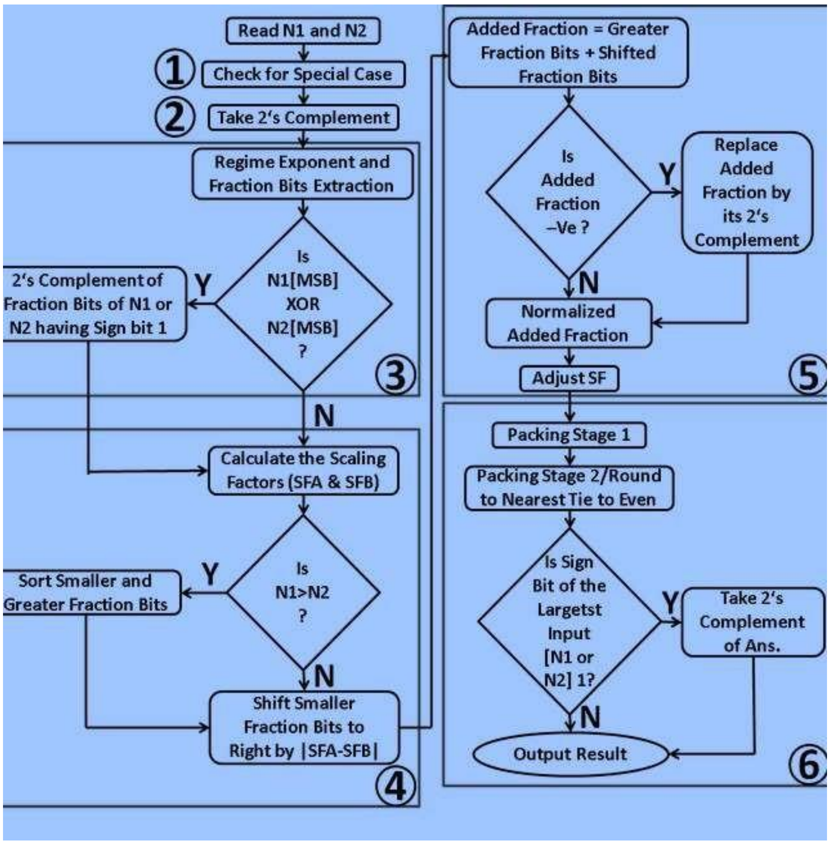
## 2 SYSTEM MODEL

### 2.1 Unit for MAC

The multiply–accumulate procedure involves computing the product of two integers and adding the result to an accumulator. While dealing on floating - point, it could be done with two or just one rounding. When done with a one round, it's termed a fused multiply–add (FMA) or fused multiply–accumulate.

### 2.2 POSIT Adder Unit

Modern computers include a specialized MAC that consists of a multiplier, an adder, and an accumulator register that records the result[10][12]. The register's output is sent back into those adder's inputs, causing the multiplier's output to be added to the register every clock cycle. Combinational multipliers need a lot of logic, but they can compute a result much faster than the shifting and adding method used by older machines. Digital signal processors were the first modern processors to include MAC units, although the approach is now widely used in general-purpose processors.



Read N1 and N2
1
Check for Special Case
2
Take 2's Complement
Regime Exponent and Fraction Bits Extraction
Is N1[MSB] XOR N2[MSB] ?
Y
2's Complement of Fraction Bits of N1 or N2 having Sign bit 1
3
N
Calculate the Scaling Factors (SFA & SFB)
Is N1>N2 ?
Y
Sort Smaller and Greater Fraction Bits
N
Shift Smaller Fraction Bits to Right by |SFA-SFB|
4
Added Fraction = Greater Fraction Bits + Shifted Fraction Bits
Is Added Fraction –Ve ?
Y
Replace Added Fraction by its 2's Complement
N
Normalized Added Fraction
Adjust SF
5
Packing Stage 1
Packing Stage 2/Round to Nearest Tie to Even
Is Sign Bit of the Largetst Input [N1 or N2] 1?
Y
Take 2's Complement of Ans.
N
Output Result
6

**Fig. 2**. MAC Unit flow chart.

## 2.3 POSIT Multiply Unit

Bits distinguish sign bits, regime bits, exponent bits, and mantissa from the two inputs. To make the original exponent bit, combine the regime bit with the exponent bit [11]. Compare the freshly created exponent bits of both inputs and add the mantissa bits if they are equal. If they aren't equal, move one of the mantissa bits to make them equal. We may now add the adjusted mantissa bits because the modified exponent matches. Normalize the extra fraction bits if necessary, then recreate the posit format. Perform the 2's complement on the reconstructed posit format if the sign bit is negative.

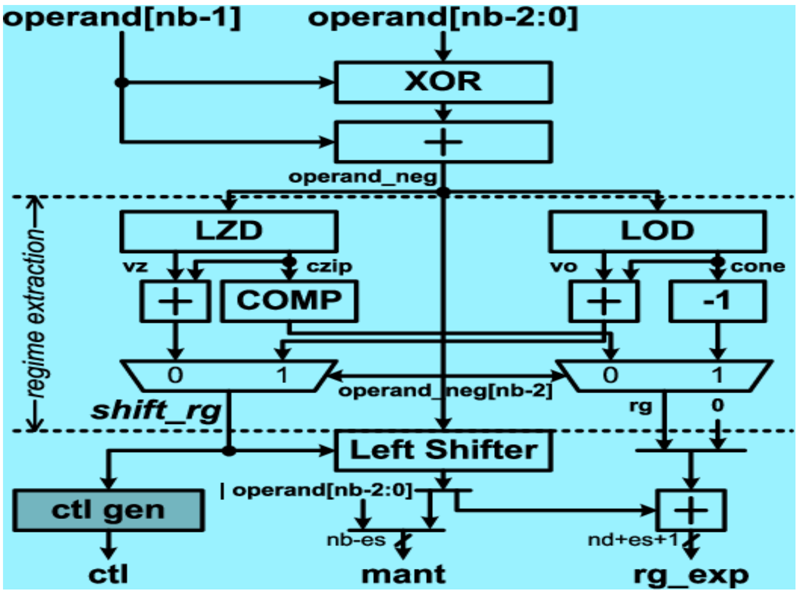


**Fig. 3**. POSIT Adder Unit.

## 3 Implementation of Hardware

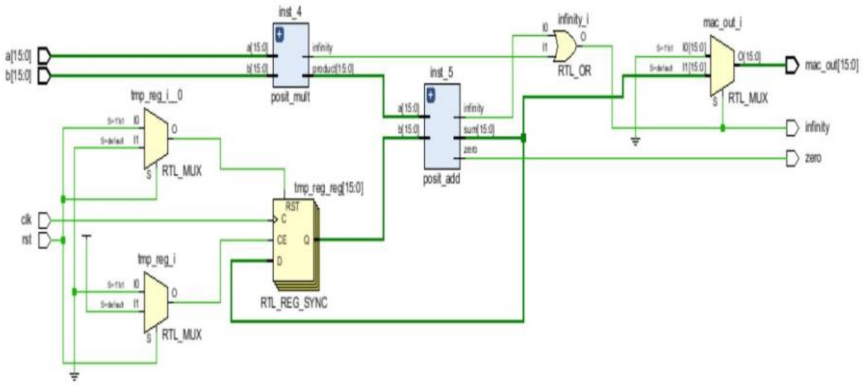


**Fig. 4**. Implementation of proposed Hardware

The posit decoder described in this architecture decodes the regime using only a leading zero detector, although others employ a leading one detector as well. This, combined with other. For certain adders and multipliers, changes lead to increased performance in term of area and energy consumption. Posits was designed to be easy to calculate on a hardware level, utilizing circuitry that was equivalent to modern floating point electronics.The main distinction between float and posit representations is that the latter includes a time-varying scaling component – the scale and accessible exponential bits. As a result, there are no specified fields in the executable format, which would be a circuit design competition. Here, we show a properly operational posit multiplier operators, as well as how this module's hardware architecture is analogous to that of floating-point arithmetic. In posit multiplication, which is virtually equivalent to floating-point multiplication, the scalability coefficients are applied, and the percentages are multiplied and reduced. When multiplying propositions, there are little differences due to the varying length of the governing field. The resultant regime's computation is not straightforward.

## 4 Discussion and Results

The results of the POSIT Multiplier are discussed in this chapter. The code is written in Verilog HDL and tested with the Vivado tool for functionality. In addition, the design was synthesized in order to obtain the report on the schematic and its use.

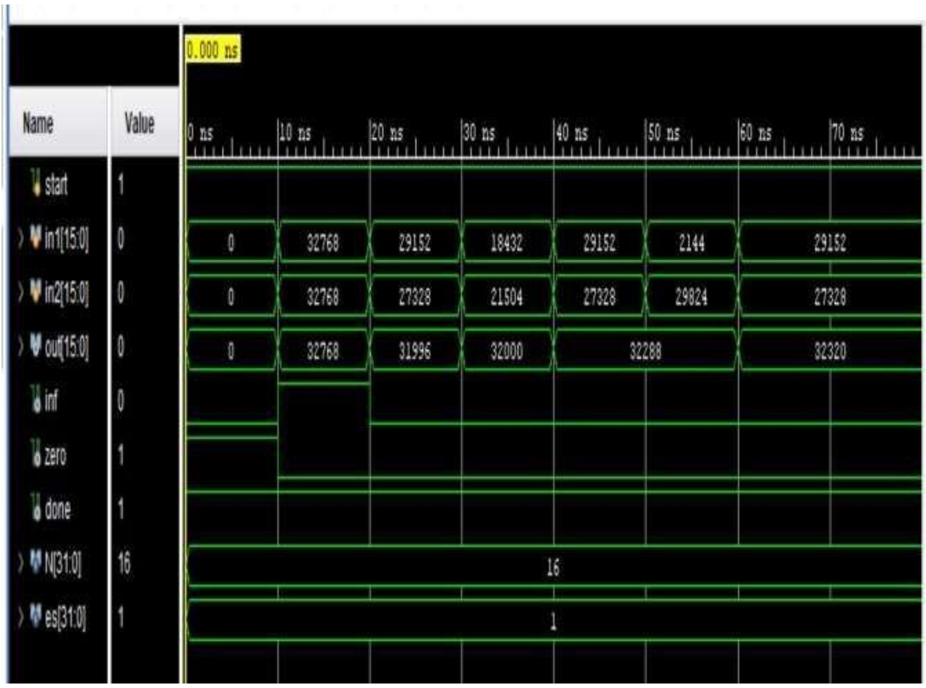
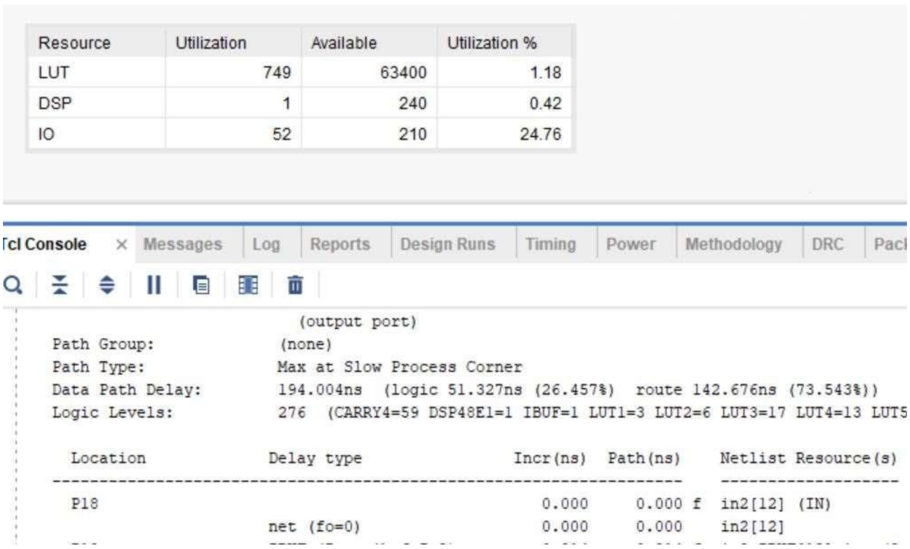


**Fig. 5**. Simulation results for POSIT MAC Unit



| Resource | Utilization | Available | Utilization % |
|---|---|---|---|
| LUT | 749 | 63400 | 1.18 |
| DSP | 1 | 240 | 0.42 |
| IO | 52 | 210 | 24.76 |

```
                            (output port)
  Path Group:             (none)
  Path Type:              Max at Slow Process Corner
  Data Path Delay:        194.004ns  (logic 51.327ns (26.457%)  route 142.676ns (73.543%))
  Logic Levels:           276  (CARRY4=59 DSP48E1=1 IBUF=1 LUT1=3 LUT2=6 LUT3=17 LUT4=13 LUT5

    Location             Delay type                Incr(ns)  Path(ns)    Netlist Resource(s)
  -------------------------------------------------------------------    -------------------
    P18                                               0.000     0.000 f  in2[12] (IN)
                         net (fo=0)                   0.000     0.000    in2[12]
```

**Fig. 6**. Utilisation Report of POSIT MAC Unit

## 5 Conclusion and Next Steps

The IEEE Standard on Floating-Point Arithmetic is often used to define floating-point integers for over thirty years. Notwithstanding this, the newly formed posit number system is seen as a direct rival to the widely accepted IEEE Series of standards. This dissertation looked at the strengths and vulnerabilities of the two computing forms to determine if the Type III universal number could be used as a fall substitute for the current IEEE Standard on Floating-Point Calculation. Furthermore, we provide a few quick notes on future study and development. The multiplier calculated inside this research will be used to design all required to form. Keep in mind that the posit demodulator is a module that all posit operators use. Furthermore, the synthesis findings revealed that the already constructed components can still be improved. As a consequence, creating a completely operational Posit Calculation Unit will be a goal in the future. Because there are currently no deep learning frameworks that support posit arithmetic, future work will include assimilating the said new layout into libraries like Tens, Flow, or Keras, allowing for testing on bigger frameworks and sets of data, as well as computing power posits on GPUs, after the first functional units on posit arithmetic become accessible**.**

## References

1. D. Goldberg, "What every computer scientist should know about floating-point arithmetic", *ACM Computing Surveys (CSUR)*, vol. 23, no. 1, pp. 5–48, Mar. 1991. DOI:10.1145/103162.103163.
2. Karthik Rao, R., Bobba, P.B., Suresh Kumar, T., Kosaraju, S., Feasibility analysis of different conducting and insulation materials used in laminated busbars, Materials Today: Proceedings, 2019, 26, pp. 3085–3089.
3. IEEE Computer Society Standards Committee and American National Standards Institute, "IEEE Standard for Binary Floating-Point Arithmetic", *ANSI/IEEE Std754-1985*. DOI: 10.1109/ieeestd.1985.82928.

4. "IEEE Standard for Floating-Point Arithmetic", *IEEE Std 754-2008*, pp. 1–70, 2008. DOI: 10.1109/ieeestd.2008.4610935.
5. Tummala, S.K., Indira Priyadarshini, T., Morphological Operations and Histogram Analysis of SEM Images using Python, Indian Journal of Engineering and Materials Sciences, 2022, 29(6), pp. 794–798
6. J. L. Gustafson, *The End of Error: Unum Computing*. CRC Press, Feb. 5, 2015,vol. 24, ISBN: 9781482239867.
7. W. Kahan and J. D. Darcy, "How Java's floating-point hurts everyone everywhere", in *ACM 1998 workshop on Java for High–Performance Network Computing*, Stanford University, 1998, pp. 1–81.
8. Suresh Kumar Tummala, Phaneendra Babu Bobba & Kosaraju Satyanarayana (2022) SEM & EDAX analysis of super capacitor, Advances in Materials and Processing Technologies, 8:sup4, 2398-2409,
9. 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, Jun. 2017. DOI: 10.14529/jsfi170206.
10. Nagarjuna, A., Suresh Kumar, T., Yogeswara Reddy, B., Udaykiran, M., Fifteen level cascaded H-bridge multilevel inverter fed induction motor, International Journal of Innovative Technology and Exploring Engineering, 2019, 8(11), pp. 640–645.
11. L. van Dam, "Enabling High Performance Posit Arithmetic Applications Using Hardware Acceleration", Master's thesis, Delft University of Technology, the Netherlands,Sep. 17, 2018, ISBN: 9789461869579.
12. Satyanarayana, K., Gopal, A.V., Babu, P.B., Design optimisation of machining parameters for turning titanium alloys with taguchi-grey method, International Journal of Machining and Machinability of Materials, 2013, 13(2-3), pp. 191–202
13. Davu, S.R., Tejavathu, R. & Tummala, S.K. EDAX analysis of poly crystalline solar cell with silicon nitride coating. Int J Interact Des Manuf (2022).
14. A. A. D. Barrio, N. Bagherzadeh, and R. Hermida, "Ultra-low-power adder stagedesign for exascale floating point units", *ACM Trans. Embed. Comput. Syst.*, vol. 13,no. 3s, 150:1–150:24, Mar. 2014. DOI: 10.1145/2567932.
15. J. L. Gustafson. (Oct. 10, 2017). Posit Arithmetic, [Online]. Available: https ://posithub.org/docs/Posits4.pdf (visited on Mar. 13, 2019).
16. J. L. Gustafson, "A Radical Approach to Computation with Real Numbers", *Supercomputing Frontiers and Innovations*, vol. 3, no. 2, pp. 38–53, Sep. 2016. DOI:10.14529/jsfi160203.
17. J. Srinivas Rao, Suresh Kumar Tummala, Narasimha Raju Kuthuri, Comparative investigation of 15 Level and 17 level cascaded h-bridge MLI with cross h-bridge MLI fed permanent magnet synchronous motor, Indonesian Journal of Electrical Engineering and Computer Science, 21(2), pp: 723-734, (2020)
18. Posit Working Group. (Jun. 23, 2018). Posit Standard Documentation, [Online].Available: https://posithub.org/docs/posit_standard.pdf (visited on Apr. 30,2019).
19. Tummala, S.K., Kosaraju, S. & Bobba, P.B. Optimized power generation in solar using carbon substrate for reduced greenhouse gas effect. Appl Nanosci 12, 1537–1543 (2022).
20. R. Munafo. (2018). Survey of Floating-Point Formats, [Online]. Available: http://www.mrob.com/pub/math/floatformats.html (visited on Feb. 9, 2019).
21. Y.Mallikarjuna Rao, M.V.Subramanyam and K. Satyaprasad, "QoS based Mobility management algorithms for Wireless mess networks", Journal of scientific and Industrial research, volume. 77, pp. 203-207, 2018.

22. Y.Mallikarjuna Rao, M.V.Subramanyam and K. Satyaprasad, "Cluster based Mobility management algorithms for Wireless mesh networks", International Journal of Communication systems (Wiley), Vol. 31, no.11, pp.1-14, 2018.
23. Y.Mallikarjuna Rao, M.V.Subramanyam and K. Satyaprasad, "Cluster based hybrid routing protocol for Wireless mesh networks", Wireless personal communications – An international journal (Springer), Vol. 103, no.4, pp. 3009-3023, 2018.