International Journal of Engineering Technology and Management Sciences
Website: ijetms.in Issue: 3 Volume No.7 May - June – 2023
DOI:10.46647/ijetms.2023.v07i04.026 ISSN: 2581-4621

IMPLEMENTATION OF HIGH PERFORMANCE POSIT-MULTIPLIER

Vinotheni M S[1], Karthika K[2]
[1]Teaching Fellow - Electronics Engineering, Anna University-MIT campus ,Chennai.
[2]Assistant Professor- Department of ECE- Mohamed Sathak A J College of Engineeing, Chennai.
Corresponding Author Orcid ID : 1. https://orcid.org/0000-0002-0550-8175
2. https://orcid.org/0000-0002-2749-6128

ABSTRACT
To represent real numbers, practically all computer systems now employ IEEE-754 floating point. Posit has recently been offered as an alternative to IEEE-754 floating point because it provides higher accuracy and a wider dynamic range. The use of not-a-numbers (NaN's) is one of the most common , having too many of them wastes valuable bit patterns in floating point format. As an alternative to floating point, a system known as "Universal Numbers" or UNUMs was developed. There are three variations of this system, but in terms of hardware compatibility, Type III(POSIT) is the best substitute for floating point. By employing only one bit pattern, this approach overcomes the NaN problem associated with floating point format. An IEEE float FPU requires 22.2% more circuitry than a posit processing unit. The proposed posit multiplier consumes 28.5% less power compared to corresponding IEEE Floating point format. The number of posit operations per second(POPS) handled by a device can be ubstantially larger than the number of FLOPS due to lower power consumption. In this paper, High performance Posit multiplier is implemented and compared with normal Floating point multiplier.
Keywords—Floating point, NaN, UNUMs, Posit Multiplier

1. INTRODUCTION

Non-integer numbers must be represented in a format that digital processors can understand. Floating point, a binary version of scientific notation, has been the standard for decades.Over the years, this format has had only minor changes, such as various bit widths or new operations such as fused operations. The format was developed by a group in the mid-1980s, taking into account current technologies. That has altered dramatically while the format has remained unchanged, allowing for a plethora of optimizations.While there are a variety of different formats for particular applications, such as fixed point, there are few that challenge fixed point as a standard.
As an alternative, Universal Numbers (Unums) were proposed. They were created as two distinct sorts at first. The first was developed as a superset of floating point, allowing for higher range and accuracy while requiring even more technology than floating point.The second type relies on positional bit patterns rather than actual data transfer. This allows for highly quick computations, but it relies on look-up tables, limiting the size of operations. As a result, a third Unum type called Posits was born.
This study compares a floating point implementation of the Posit number system against a hardware implementation. Both floating point and posit systems will be evaluated for area, power, and other factors.
The goal of designing and testing both a floating point and posit unit is to demonstrate a side-by-side comparison so that the costs and benefits of each may be understood.The following is a list of the research tasks [1] to gain an understanding of floating point merits and problems [2] to gain an understanding of posit merits and problems [3] to design floating point and posit arithmetic cores [4] Fully test and analyze both cores and Compare analysis results and discuss.

2. BACKGROUND

Figure 2.1 shows the 3 parts of IEEE-754 floating point representation: sign, exponent and mantissa. Single precision floats have 1 sign bit, 8 exponent bits and 23 mantissa bits. For more details we refer the reader to see the reference papers of [2], [9].

Sign	Exponent	Mantissa

Fig 2.1 Floating Point Bit Pattern

A generic Posit format consists of a sign bit, one or multiple regime bits, optional exponent bits and optional fraction bits (Fig. 2.2). The sign bit defines the number as being either positive (0) or negative (1). After the sign bit there is a dynamic number of sequential 1s or 0s r followed by a terminating bit of the other sign r, this is the regime. The number of bits for the exponent and fraction is also dynamic, and might be zero if required.

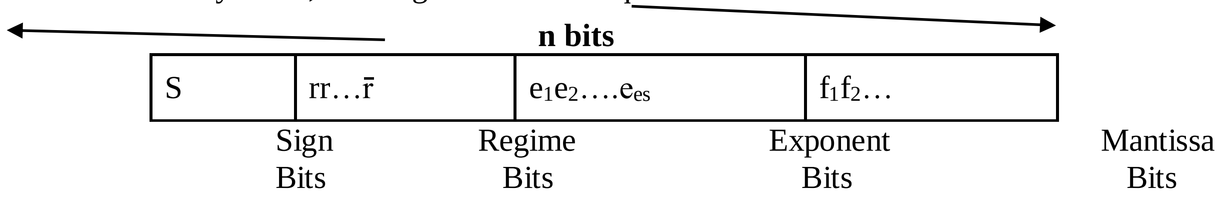


Fig 2.2 Posit Bit Pattern

The regime is a sequence of m r-bits, where r ∈ (0, 1) followed by a $\bar{r}$ of the other sign to terminate the sequence. Let R be a regime consisting of m r-bits. If r is 1, the value represented by R is the positive value of (m - 1). If r is 0, the value is instead the negative value -m. Thus, regimes with a leading 0 represent a negative value and regimes with a leading 1 represent a positive value (table 2.1).

Table 2.1 Posit Regime Decoding

Binary	0000	0001	001x	01xx	10xx	110x	1110	1111
Numerical meaning, k	−4	−3	−2	−1	0	1	2	3

The regime denotes a scale factor of useed k, where useed is the unit of measurement.The regime is then combined to a scale factor useed^R, where useed = 2^{2es} and es is the number of bits in the exponent E.

Table 2.2 Posit value of used

es	0	1	2	3	4
useed	2	2^2=4	4^2=16	16^2=256	256^2=65536

The next bits (color-coded blue) are the exponent e, regarded as an unsigned integer. There is no bias as there is for floats; they represent scaling by 2 e. There can be up to (es) exponent bits, depending on how many bits remain to the right of the regime. This is a compact way of expressing tapered accuracy.Like in IEEE 754 the Posit format can have a variable number of exponent bits. However, there is no predefined rule that states how many exponent bits there must be. A Posit exponent may consist of up to es number of bits, forming the decimal value e. The remaining bits after the exponent make up for the fraction f. Like IEEE 754 the fraction is normalized, and the leading 1 is always omitted in the encoding.

2. LITERATURE SURVEY

[1]This paper proposes open- source hardware Posit Arithmetic Core Generator(PACoGen) for the lately developed universal number posit number system, along with a set of pipelined architectures. The posit number system is composed of a run- time varying exponent element, which is defined by a composition of varying length ‘‘ regime- bit ’’ and ‘‘ exponent- bit ’’(with a maximum size of ES bits, the exponent size). This in effect also makes the fraction part to vary at run- time in size and position. In this view, this paper targets algorithmic development and general HDL creators(PACoGen) for basic posit arithmetic.. The PACoGen can deliver the Verilog HDL code respective posit arithmetic for any given posit word range(N) and exponent size(ES), as defined under the posit number system. further, pipelined infrastructures of 32- bit posit with 6- bit exponent size are also proposed and bandied for addition/ deduction, addition, and division computation. The proposed posit computation infrastructures are demonstrated on the Virtex- 7(xc7vx330t- 3ffg1157) FPGA device as well as Nangate 15 nm ASIC platform.

[2] In this paper, a power effective posit multiplier configuration is proposed.Posit number system has been used as an alternative to IEEE floating- point number system in numerous operations, especially the recent popular deep learning. Among all the affiliated computation operations, addition is one of the most frequent operations used in operations. Still, due to the bit- range flexibility of posit arithmetic, the tackle multiplier is generally designed with the maximum possible mantissa bit- range. The mantissa multiplier is still designed for the maximum possible bit- range, the whole multiplier is divided into multiple lower multipliers. This design approach is applied to 8- bit, 16- bit, and 32- bit posit formats in this brief and an average of 16 power reduction can be achieved with negligible area and timing overhead.

[3] This paper focuses on the design and implementation of a high-speed floating-point multiply-accumulator (MAC) using field-programmable gate arrays (FPGAs).The authors aim to develop an efficient and high-performance MAC unit that can perform floating-point multiplication and accumulation operations quickly.The paper begins with an introduction to the importance of MAC units and the need for high-speed implementations. The proposed design in the paper focuses on reducing the critical path delay, which directly impacts the speed of the MAC unit. The authors present a novel architecture that employs pipelining and parallelism techniques to improve the performance. They used a Xilinx Virtex-7 FPGA for implementation and performed various tests to measure the speed, area utilization, and power consumption of their MAC unit. The results demonstrate that the proposed design outperforms previous designs in terms of speed while maintaining reasonable area utilization and power efficiency.

[4]This work presents the design and algorithm of a novel parameterized and pipelined Posit fused multiply and accumulate (P-FMA) unit.The P-FMA unit is a fused functional unit that performs addition, subtraction, multiplication, and multiply-and-accumulate operations. The design incorporates a 5-stage pipeline and is parameterized to allow flexibility. The P-FMA unit has also been synthesized on an FPGA and compared with other contemporary Posit FMAs. It exhibits a lesser critical path and comparable area utilization. The accuracy of the Posit FMA unit is evaluated by estimating Pi (π) using the Chudnovsky algorithm. The Pi estimates from Posit (64, es=1 to 6) are compared with an IEEE 754 FPU (double precision) based Pi estimate. The results show that the P-FMA, when integrated with other Posit units, enables computing Pi to 16-digit accuracy, surpassing the 15-digit accuracy of IEEE Double-Precision floats achieved with Y-Cruncher.This unit has been thoroughly verified and synthesized on an FPGA, showcasing its potential for high-performance arithmetic computations.

[5] This paper proposes an iterative approach to reduce hardware resources in a posit multiplier. It dynamically adjusts the number of truncated bits in the fraction component based on the number of regime bits to leverage the features of the posit format. This paper presents architectures for a fast parser and packer. By utilizing the posit format, the proposed design achieves a dynamic range approximately 60 decades wider than a single-precision floating-point multiplier design. The proposed iterative approach reduces the number of lookup tables by 44% while achieving a 51% higher maximum frequency. Moreover, the design allows for a fine balance between latency and accuracy at runtime.

3. POSIT MULTIPLIER

A. Multiplication Algorithm (introduction)

The posit multiplier arithmetic has three main processing sections, similar to the posit adder arithmetic: posit extraction, core arithmetic, and posit creation. The posit multiplier generator's algorithmic computational flow. The extraction of posit data has been completed. It includes the operands complemented form (XIN1, XIN2, for negative posit), sign bits (S1, S2), regime check bits (RC1, RC2), absolute regime sequence values (R1, R2), exponent values (E1, E2), mantissas (M1, M2), infinity & zero check bits, and infinity & zero check bits (Inf1, Inf2, Z1, Z2).
The multiplication Algorithm involves two blocks:

- Input Extraction Block
- Posit Multiplication Main Block

B. Input Extraction Block

The extraction of components in hardware arithmetic units is more difficult than in floating-point format. The circuit for extracting each component of a posit number.If the number is negative, it is complemented first. The regime is then extracted first. A succession of ones (zeroes) is followed by a single zero (one) bit in the regime section.

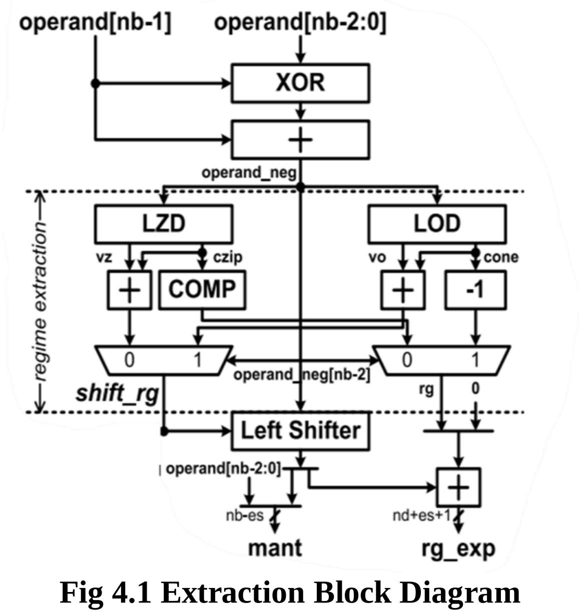


Fig 4.1 Extraction Block Diagram

C. Posit Multiplication Main Block

The suggested posit multiplier's parameterized Datapath is demonstrated. Posit component extraction, mantissa multiplier, final adder and normalization, posit component packing, and rounding are all part of the critical route.Furthermore, the sign and exponent are treated individually. The mantissa multiplier is a modified Booth multiplier with (nb - es) bits of radix 4.The bit-width of the mantissa in posit format changes according to the value. As a result, a (nb - es)-bit multiplier is not always required for the mantissa.
To count the number of leading bits, a leading zero detector (LZD) and a leading one detector (LOD) are utilized. If leading ones are found, rg equals count-1. Otherwise, rg is count, and to

convert a positive count to a negative rg number, a complementer, COMP, is required.In addition, the count+1 regime bit-width shift rg is constructed, allowing the regime to be eliminated and the exponent and mantissa to be acquired using a shifting operation.

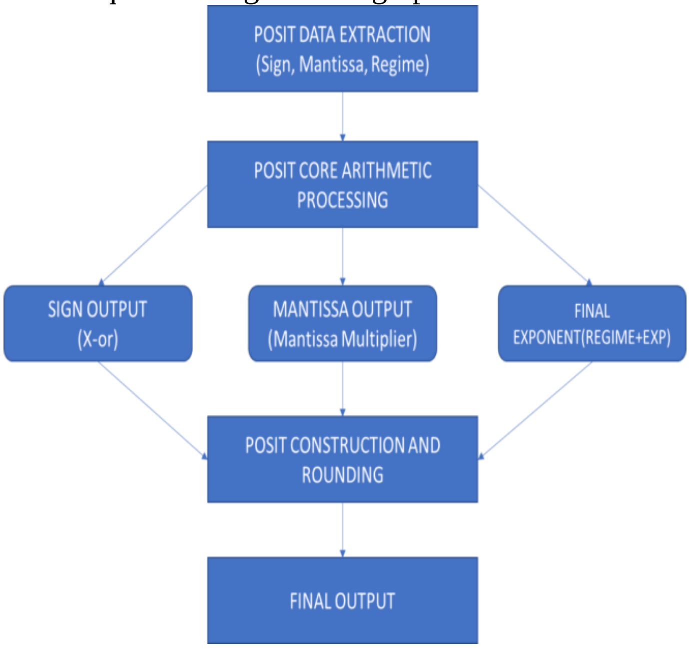

POSIT DATA EXTRACTION (Sign, Mantissa, Regime)
POSIT CORE ARITHMETIC PROCESSING
SIGN OUTPUT (X-or)
MANTISSA OUTPUT (Mantissa Multiplier)
FINAL EXPONENT(REGIME+EXP)
POSIT CONSTRUCTION AND ROUNDING
FINAL OUTPUT

Fig 4.2 Block Diagram Of Posit Multiplier

The bit-width of the 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 will be utilized in a posit multiplier or multiply-accumulate unit architecture.

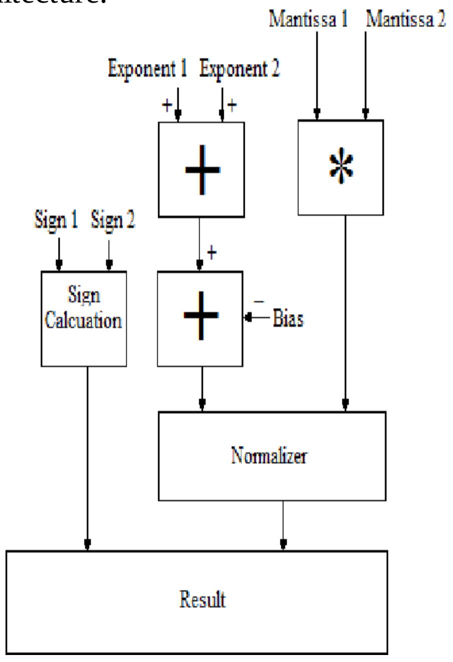


Fig 4.3 Block Diagram Of Floating Point Multiplier

D. Posit Multiplier Algorithm

The computational flow for the multiplication of Posit number is given below :

1: GIVEN:
2: N: Posit Word Size
3: ES: Posit Exponent Field Size
4: RS: log2 (N) (Posit Regime Value Store Space Bit Size)
5: Input Operands: IN1, IN2
6: Posit Data Extraction:

```
7: IN1 → XIN1, S1, RC1, R1, E1, M1, Inf1, Z1
8: IN2 → XIN2, S2, RC2, R2, E2, M2, Inf2, Z2
9: Z ← Z1&Z2 Inf ← Inf 1|Inf 2
```

Core multiplication arithmetic is performed on the extracted components. A 1- bit xor operation among input sign bits is required for the sign-bit computation (Line 12). A (N-ES-2)X(N-ES-2) integer multiplier (Line 14) is required for mantissa multiplication, and the output MSB bit is checked for any mantissa multiplication overflow (Line 15). For correct normalizing, the mantissa is left shifted by one bit for mantissa multiplication overflow (Line 16) , and the final exponent computation is incremented by one bit (Line 20).

```
10: Posit Core Multiplier Arithmetic Processing:
11: Sign Processing:
12: S ← S1 xor S2
13: Mantissa Multiplication Processing:
14: M ← M1×M2 (Mantissa Multiplication
15: Movf ← M[MSB] (Check Mantissa Overflow)
16: M ← Movf ? M : M << 1 (1-bit Mantissa Shifting for overflow)
17: Final EXPONENT (E_O) and REGIME (R_O) Processing:
18: RG1 ← RC1 ? R1 : -R1 (Effective regime-1 value)
19: RG2 ← RC2 ? R2 : -R2 (Effective regime-2 value)
20: Exp_O[ES + RS + 1:0]←{RG1, E1} + {RG2, E2} + Movf (Total Exponent value) 21:
Exp_ON[ES + RS:0] = Exp_O[ES + RS + 1] ? - Exp_O : Exp_O
( Absolute Total Exponent Value)
22: E_O[ES-1:0] = (Exp_O[ES + RS + 1] & (|Exp_ON[ES1:0])) ?
Exp_O[ES-1:0] : Exp_ON[ES-1:0] (Exponent Output)
23: R_O[RS:0] = !(Exp_O[ES + RS + 1]) | (Exp_O[ES + RS + 1] &
( |Exp_ON[ES-1:0])) ? Exp_ON[ES + RS:ES] + 1'b1 : Exp_ON[ES + RS:ES]
( Absolute Regime Value)
```

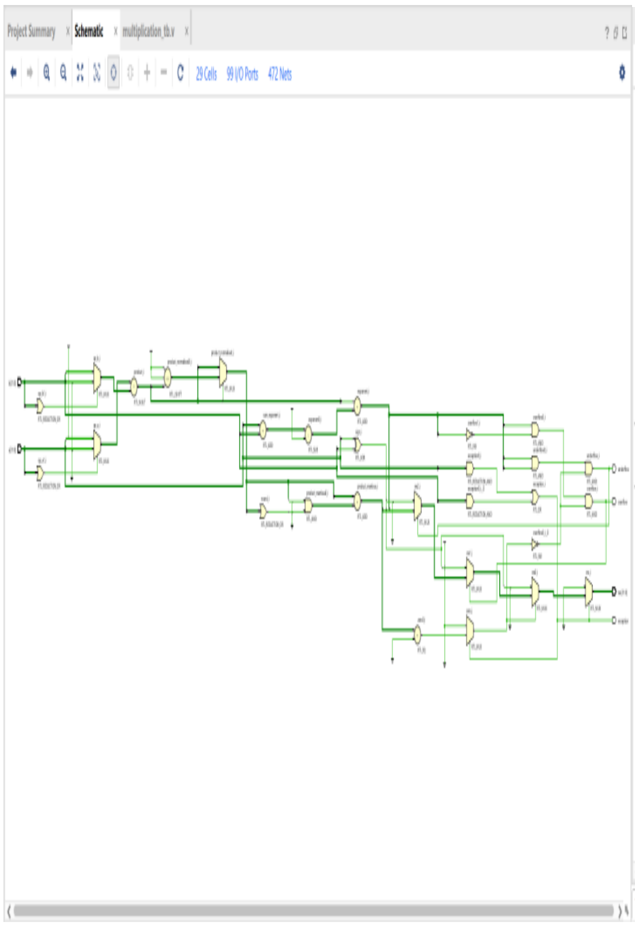
The real regime values (RG1 and RG2) are first obtained utilizing respective regime check bits and absolute regime value for the exponent computation (Lines 18-19). These regime values are combined with their corresponding exponents (E1, E2) to yield effective exponent values for each operand, which are then added to the Movf bit (mantissa overflow) to yield the total output exponent value, Exp_O. (Line 20). Lines 21-23 compute the ES-bit exponent output E_O and absolute regime output value R_O using Algorithm-6 and the total output exponent value Exp_O.

```
24:Posit Construction, Rounding and Final Processing:
25: S, R_O, Exp_O, E_O, M
26: Posit Data Composition:
27: REGIME, EXPONENT and MANTISSA Packing:
28: REM ←N+1 Bits Regime Sequence z }| { N{!Exp_O[MSB]},
Exp_O[MSB], Exponent − Mantissa z }| { E_O, M[(N-ES-2) bits MSBs],
GRS−Bits z }| { M[Next 2 bits],|(M[:0])
29: REM ← REM >> R_O
30: Rounding: Round to nearest even
31: IF (R_O < N-ES-2)
32: ULP_add = G.(R + S) + L.G.(!(R + S))
33: REM ← REM + {(N-1)'b0,ULP_add}
34: REM ← (LS == 0) ? REM : 2's Complement of REM
35: Final Output:
36: Combine LS with MSB (N − 1) bit of rounded REM
37: Discharge Output while considering Exceptions
```

The posit creation, rounding, and final processing are all done after the fundamental arithmetic processing.

4. RESULTS AND DISCUSSION

4.1 RTL DESIGN



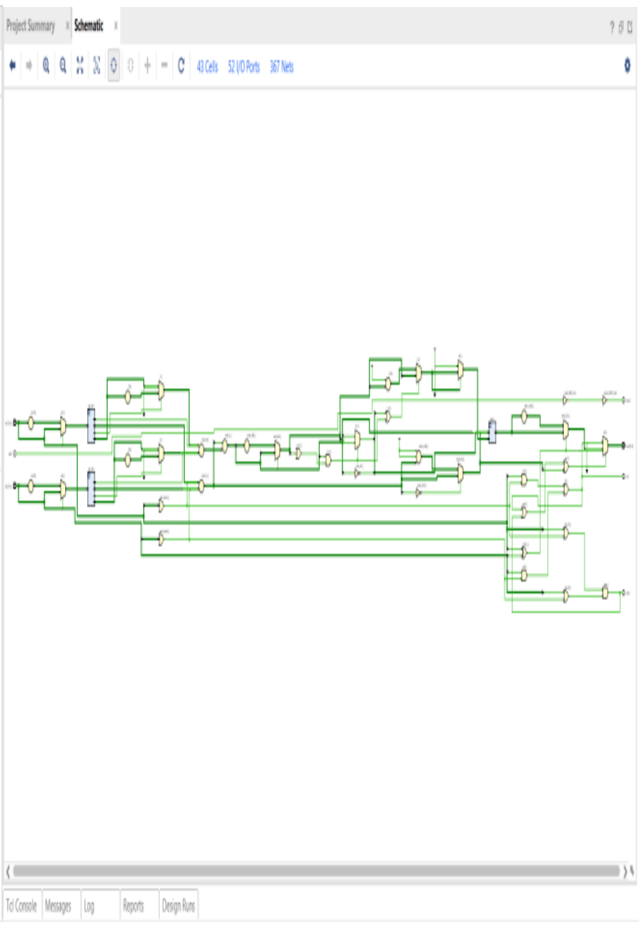
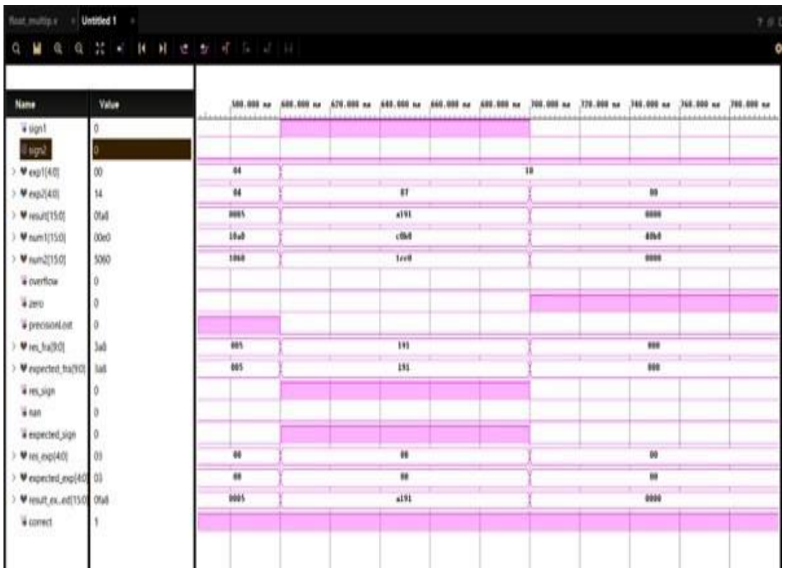


Fig 4.1: Floating Point Multiplier RTL **Fig 4.2: Posit Multiplier RTL**

4.2 MULTIPLICATION OUTPUT

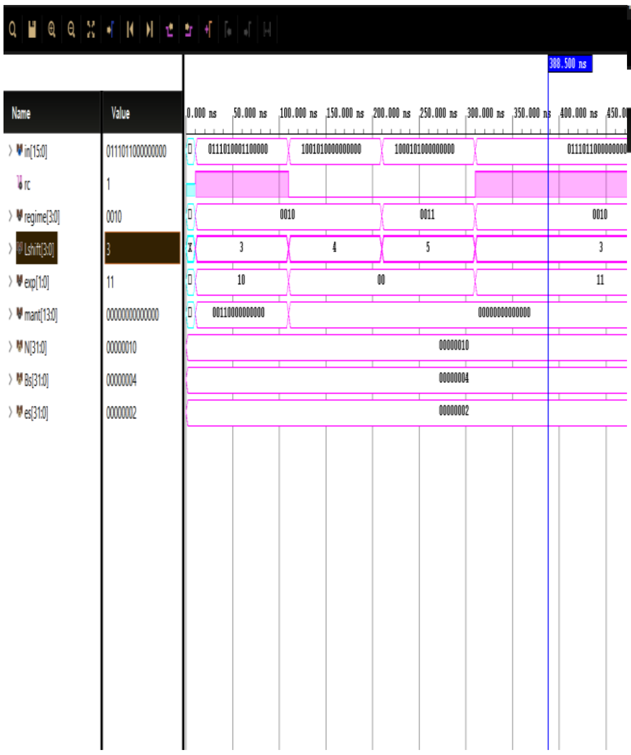
4.2.1 FLOATING MULTIPLIER OUTPUT



Name	Value
sign1	0
sign2	0
exp1[4:0]	00
exp2[4:0]	14
result[15:0]	0fa8
num1[15:0]	00e0
num2[15:0]	5060
overflow	0
zero	0
precisionLost	0
res_fra[9:0]	3a8
expected_fra[9:0]	3a8
res_sign	0
nan	0
expected_sign	0
res_exp[4:0]	03
expected_exp[4:0]	03
result_ex..ed[15:0]	0fa8
correct	1

Fig 4.3: Floating Point Multiplier Output (16 bits)

4.2.2 POSIT MULTIPLIER OUTPUT



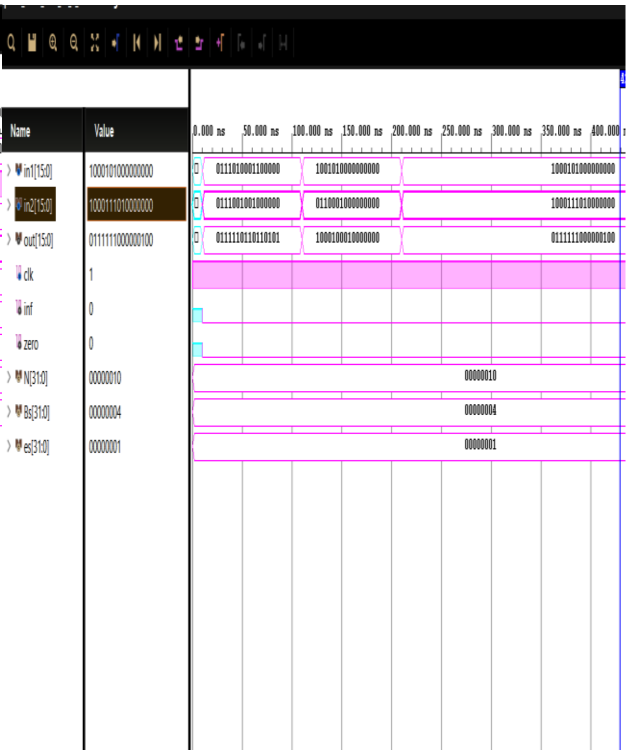
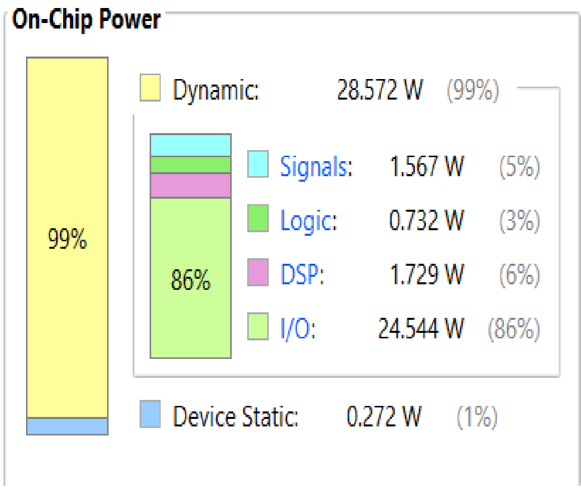


Fig 4.4 : Data Extraction Output (16 bits)

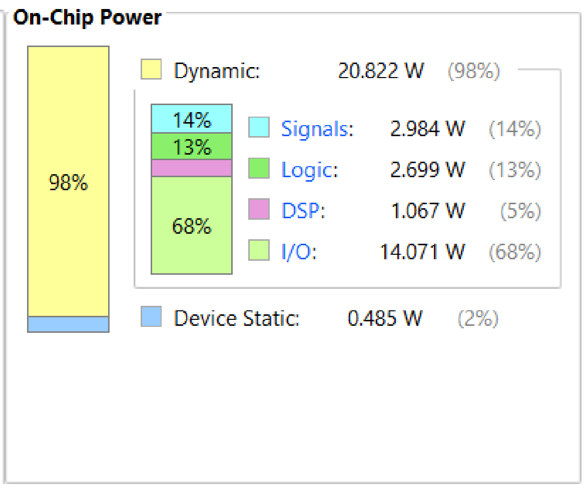
Fig 4.5: Posit Multiplication Output (16 bits)

4.3 POWER OUTPUT



On-Chip Power

Dynamic:	28.572 W	(99%)
Signals:	1.567 W	(5%)
Logic:	0.732 W	(3%)
DSP:	1.729 W	(6%)
I/O:	24.544 W	(86%)
Device Static:	0.272 W	(1%)



On-Chip Power

Dynamic:	20.822 W	(98%)
Signals:	2.984 W	(14%)
Logic:	2.699 W	(13%)
DSP:	1.067 W	(5%)
I/O:	14.071 W	(68%)
Device Static:	0.485 W	(2%)

Fig 4.6 : Floating PointPower Output

Fig 4.7: Posit Power Output

Posit is a recent development in numerical computing and has shown some significant benefit over IEEE-754 floating point standard. Posits outperformed floats at their own game: doing calculations by guesswork and rounding errors. Posits feature better closure, a wider dynamic range, and higher precision. They can be utilized to generate more accurate results using the same amount of bits as floats, or (what may be even more convincing) an equally sound response with less information. Given that present-day systems With limited bandwidth, employing smaller operands results in faster processing and less power use. Posits avoid the main flaw of the IEEE 754 float definition by producing bitwise-reproducible results across computing platforms. The circuitry required for quick hardware is decreased by the simpler and more elegant design of posits compared to floats. Floats may already be commonplace, but posits may soon render them obsolete.

4.4 COMPARATIVE ANALYSIS

Table 5.1 Comparative Analysis Of Ieee 754 And Posit Format

PARAMETER	Floating point - IEEE 754 Format	Flaoting Point - Posit Format
No. of bonded IOBs (Available -300)	259	259
No. of slice LUTs used (Available -134600)	14235	8814
Total on chip Power(W)	28.57	20.82
Delay(ns)	22.08	21.12

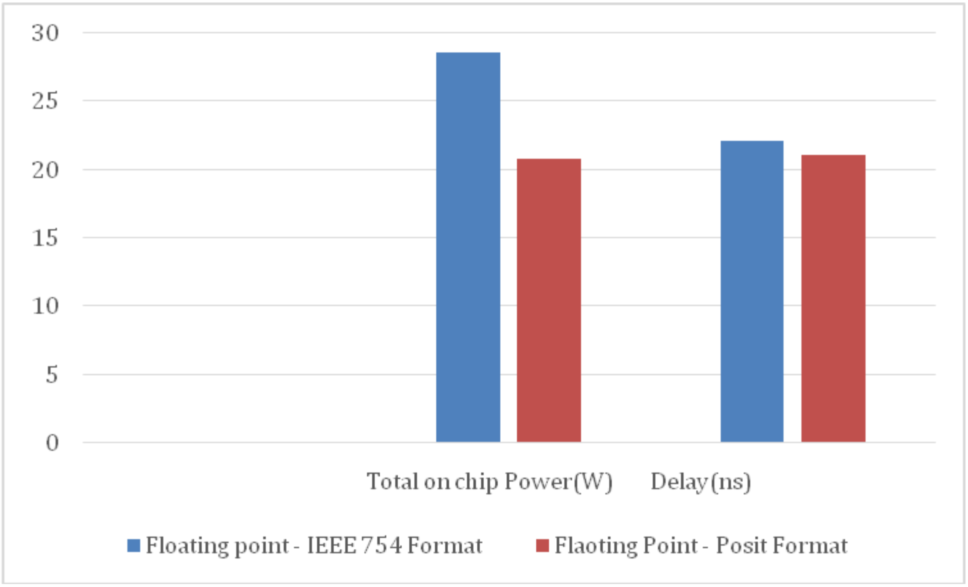


Fig 4.9 Power And Delay Analysis of IEEE 754 And Posit Format

The comparative analysis of area, power and delay of Floating point IEEE 754 and Posit Format is shown in Fig 5.9. Compare to IEEE 754 Standard, the implemented design of posit multiplier have reduced area, power and delay.

5 .CONCLUSION AND FUTURE SCOPE

The implemented design is having total power consumption of 21.307 W and a generic Floating point multiplier is having a power consumption of 28.844 W, therefore a 26.13% of power reduction is achieved. The area is also reduced. For the same number of inputs Posits can achieve higher accuracy with less power and area, which leads to its more efficient usage in the deep learning process and Graphical processing units. The future plan is to implement this in an FPGA and use it in the general processing where the Floating point system is in use currently. It would be interesting to see Posits used in a custom processor core built specifically for neural nets and deep learning applications.

Future works can be implementing a complete architecture of ALU or micro architecture using complete POSIT architecture. The data processing for neural nets are very computationally heavy and rely primarily on Floating Point units. Some alternatives have been used such as a variable precision FPU. There has been some work using Posits which performed significantly better than a comparable fixed-point computational unit or when using a dedicated Posit multiply-accumulator unit. We have implemented the Posit multiplier by extracting the input using the extraction block. The regime and the exponent are processed in the exponent processing block, the mantissa is multiplied using the multiplier. Finally, all the components are packed and the final output is given as output. By comparing the floating point and posit multiplier, we are seeing significant reduction in the power and area.

References

[1] M. K. Jaiswal and H. K. -. So, "PACoGen: A Hardware Posit Arithmetic Core Generator," in IEEE Access, vol. 7 , pp. 74586-74601 , 2019 , doi:10.1109 /ACCESS.2019.2920936.
[2] H. Zhang and S. -B. Ko, "Design of Power Efficient Posit Multiplier,"in IEEE Transactions on Circuits and Systems II: Express Briefs , vol.67 , no. 5 , pp. 861-865 , May 2020, doi: 10.1109/TCSII.2020.2980531.
[3] B. Zhou, G. Wang, G. Jie, Q. Liu and Z. Wang, "A High-Speed Floating-Point Multiply-Accumulator Based on FPGAs," in IEEE Transactions on Very Large Scale Integration (VLSI) Systems , vol. 29,no. 10, pp. 1782-1789,Oct.2021, doi: 10.1109/TVLSI.2021.3105268.
[4] S. Jean et al., "P-FMA: A Novel Parameterized Posit Fused Multiply-Accumulate Arithmetic Processor," 2021 34th International Conference on VLSI Design and 2021 20th International Conference on Embedded Systems (VLSID) , 2021, pp. 282-287, doi: 10.1109/VLSID51830.2021.00053.
[5] C. J. Norris and S. Kim, "An Approximate and Iterative Posit Multiplier Architecture for FPGAs," 2021 IEEE International Symposium on Circuits and Systems (ISCAS),2021,pp.1-5,doi:10.1109/ISCAS51556.2021.9401158.
[6] V. Gohil, S. Walia, J. Mekie and M. Awasthi, "Fixed-Posit: A Floating-Point Representation for Error-Resilient Applications," in IEEE Transactions on Circuits and Systems II: Express Briefs, vol. 68 , no.
10 , pp. 3341-3345 , Oct. 2021 , doi: 10.1109/TCSII.2021.3072217.
[7] Research Website for POSIT development https://posithub.org/
[8] Decimal to Posit converter https://posithub.org/widget/plookup
[9] Beating Floating Point at its Own Game: Posit Arithmetic 1. Background: Type I and Type II Unums
[10] Wagner, Matt, "Posits: An Alternative to Floating Point Calculations" (2020). Thesis. Rochester Institute of Technology.
[11] “Beating Floating Point at its Own Game: Posit Arithmetic” John L. Gustafson1 , Isaac Yonemoto2
[12] Beyond Floating Point: Next-Generation Computer Arithmetic. Speaker: John L. Gustafson, National University of Singapore. https://youtu.be/aP0Y1uAA-2Y .
[13] Vo Ngoc Mai Anh; Hoang Kim Ngoc Anh; Vo Nhat Huy; Huynh Gia Huy; Minh Ly. "Improve Productivity and Quality Using Lean Six Sigma: A Case Study". *International Research Journal on Advanced Science Hub*, 5, 03, 2023, 71-83. doi: 10.47392/irjash.2023.016
[14] Swathi Buragadda; Siva Kalyani Pendum V P; Dulla Krishna Kavya; Shaik Shaheda Khanam. "Multi Disease Classification System Based on Symptoms using The Blended

Approach". *International Research Journal on Advanced Science Hub*, 5, 03, 2023, 84-90. doi: 10.47392/irjash.2023.017

[15] Susanta Saha; Sohini Mondal. "An in-depth analysis of the Entertainment Preferences before and after Covid-19 among Engineering Students of West Bengal". *International Research Journal on Advanced Science Hub*, 5, 03, 2023, 91-102. doi: 10.47392/irjash.2023.018

[16] Pavan A C; Lakshmi S; M.T. Somashekara. "An Improved Method for Reconstruction and Enhancing Dark Images based on CLAHE". *International Research Journal on Advanced Science Hub*, 5, 02, 2023, 40-46. doi: 10.47392/irjash.2023.011

[17] Pravin T, M. Subramanian, R. Ranjith, Clarifying the phenomenon of Ultrasonic Assisted Electric discharge machining, "Journal of the Indian Chemical Society", Volume 99, Issue 10, 2022, 100705, ISSN 0019-4522, Doi: 10.1016/j.jics.2022.100705

[18] R. Devi Priya, R. Sivaraj, Ajith Abraham, T. Pravin, P. Sivasankar and N. Anitha. "MultiObjective Particle Swarm Optimization Based Preprocessing of Multi-Class Extremely Imbalanced Datasets". International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems Vol. 30, No. 05, pp. 735-755 (2022). Doi: 10.1142/S0218488522500209

[19] T. Pravin, C. Somu, R. Rajavel, M. Subramanian, P. Prince Reynold, Integrated Taguchi cum grey relational experimental analysis technique (GREAT) for optimization and material characterization of FSP surface composites on AA6061 aluminium alloys, Materials Today: Proceedings, Volume 33, Part 8, 2020, Pages 5156-5161, ISSN 2214-7853, https://doi.org/10.1016/j.matpr.2020.02.863.

[20] Rajashekhar, V., Pravin, T., Thiruppathi, K.: A review on droplet deposition manufacturing a rapid prototyping technique. Int. J. Manuf. Technol. Manage. 33(5), 362–383 (2019) https://doi.org/10.1504/IJMTM.2019.103277

[21] V.S. Rajashekhar; T. Pravin; K. Thiruppathi , "Control of a snake robot with 3R joint mechanism", International Journal of Mechanisms and Robotic Systems (IJMRS), Vol. 4, No. 3, 2018. Doi: 10.1504/IJMRS.2018.10017186

[22] Subha S; Sathiaseelan J G R. "The Enhanced Anomaly Deduction Techniques for Detecting Redundant Data in IoT". *International Research Journal on Advanced Science Hub*, 5, 02, 2023, 47-54. doi: 10.47392/irjash.2023.012

[23] Nguyen Kieu Viet Que; Nguyen Thi Mai Huong; Huynh Tam Hai; Vo Dang Nhat Huy; Le Dang Quynh Nhu; Minh Duc Ly. "Implement Industrial 4.0 into process improvement: A Case Study in Zero Defect Manufacturing". *International Research Journal on Advanced Science Hub*, 5, 02, 2023, 55-70. doi: 10.47392/irjash.2023.013

[24] Ayush Kumar Bar; Avijit Kumar Chaudhuri. "Emotica.AI - A Customer feedback system using AI". *International Research Journal on Advanced Science Hub*, 5, 03, 2023, 103-110. doi: 10.47392/irjash.2023.019

[25] Rajarshi Samaddar; Aikyam Ghosh; Sounak Dey Sarkar; Mainak Das; Avijit Chakrabarty. "IoT & Cloud-based Smart Attendance Management System using RFID". *International Research Journal on Advanced Science Hub*, 5, 03, 2023, 111-118. doi: 10.47392/irjash.2023.020