Efficient Hardware Architecture for Posit Addition/Subtraction

Authors:
S. Susheel Ujwal
Dr. Srinivas Boppu
Dr. Debapratim Ghosh

Contents

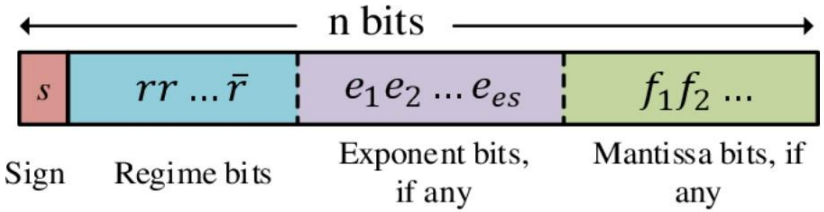
- Overview on Posits
- Posit format and decoding example
- Posits Vs Floats
- Flowchart of proposed implementation
- Comparison of Existing design and Proposed design
- Flowchart of Verification
- Accuracy comparison
- Results
- Conclusion
- References

Overview on Posits

- Floating point can support a much wider range of values than Fixed point, with the ability to represent very small numbers and very large numbers.

- Posits are designed as a direct drop-in replacement for IEEE Standard 754 floating-point numbers (floats) because of their compelling advantages over floats like,
  - Larger dynamic range.
  - Simpler exception handling.
    - It defines only one ZERO, one Infinity and no Not-a-Number (NaN).By not including NaN, with a M-bit FP mantissa, posit saves on 2*(2^M-1)invalid (NaN) combinations, 32 bit numbers have 16,777,214 combinations.
  - Tapered accuracy
  - Higher accuracy when compared to same bit width floats.

Posit Format



- **Regime bits** are included in total exponent calculation, It has an array of bits with the same bit value (0/1) and it is terminated by the opposite bit value.

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

- **Exponent bits** are included in the total exponent calculation, its length is ES bits. It is Stored in unsigned integer format.
- **Mantissa bits** are fraction bits; They are decoded as 1.f1f2. . . ..
- Decoded Posit value:

$$sign * 2^{2^{Es} * k + e} * 1.f$$

Posit decoding examples

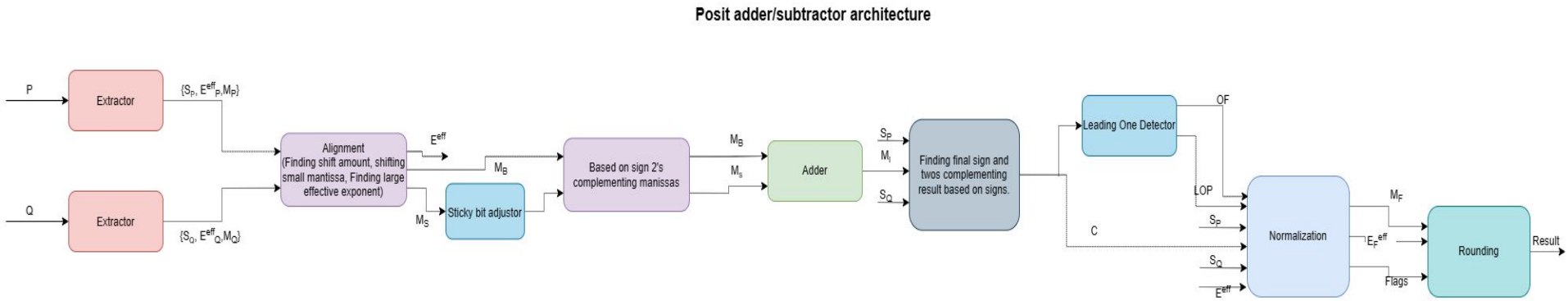
- For N_{width} = 8, Es=2

$WithES = 2:$	Sign Regime	Exp	Mantissa
0_0001_11_1	$= +(2^{(2^2)})^{-3}$	$*2^3$	$*(1+\frac{1.0}{2})$
0_110_10_11	$= +(2^{(2^2)})^{1}$	$*2^2$	$*(1+\frac{3.0}{4})$

Posits Vs Floats

Float	Posit
Dynamic range of 16 bit: 6*10^(-8)- 7*10^(4)	Dynamic range of 16 bit with ES=1: 4*10^(-9)- 3*10^(8)
2046 NaN Combinations for 16 bit.	No NaN Combinations.
There is a difference between +0,-0.	There is no difference between +0,-0.
Not a Tapered accuracy.	Tapered accuracy, because of this near the +1, -1 accuracy is more when compared to away from +1, -1.
Need only 16 bits for storing 16 point floats.	Need 18 number of bits for storing decoded posits.
Might get different results across systems due to the different rounding schemes.	Bitwise identical results across systems.
Lower accuracy compared to Posits of the same bit length.	Higher accuracy compared to Floats of the same bit length.
Might Overflow to +inf, -inf, underflow to zero.	Will not Overflow to inf but rounded to maxpos,will not underflow to zero but rounded to minpos.
No need for extra hardware for decoding and encoding.	Need extra hardware for decoding and encoding.

Flowchart of proposed implementation

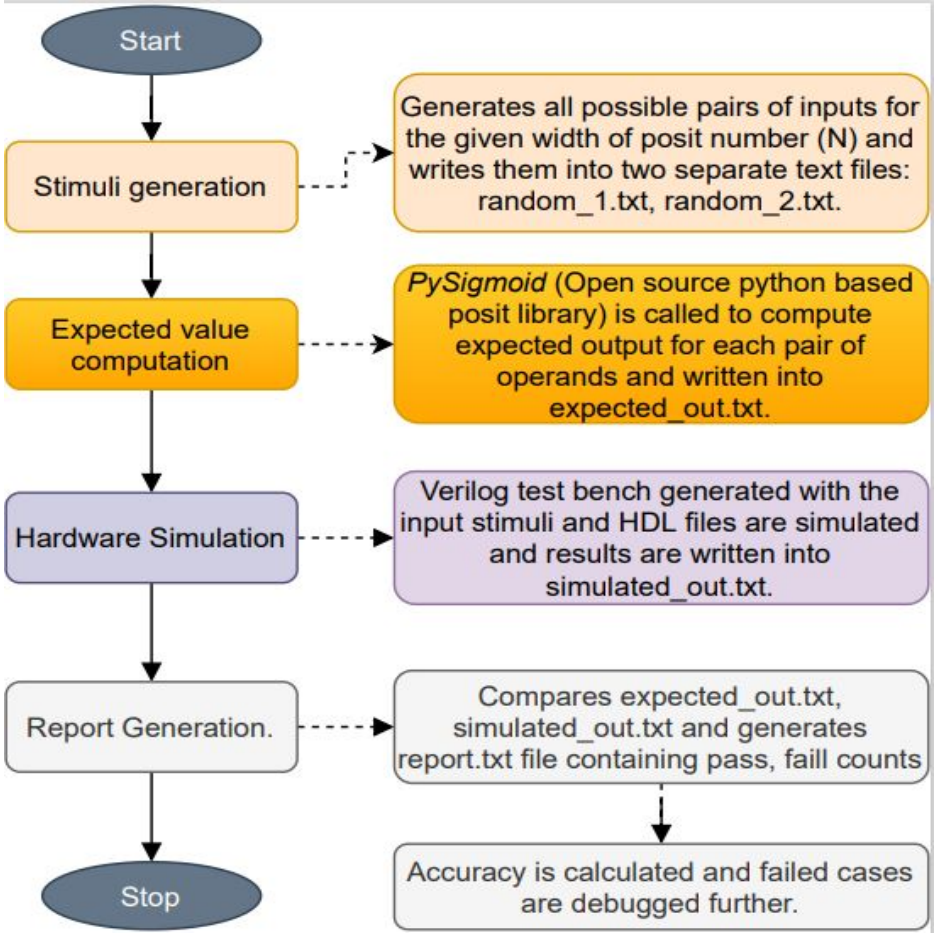


Comparison of proposed and existing design

- The main difference between the proposed design and the existing designs is, that the regime and exponent are treated as a single quantity and compute the intermediate values, whereas the proposed design treats the Regime, and exponent separately and computes the intermediate values.

- The proposed design uses a single adder in the arithmetic stage whereas existing designs use two adders in the arithmetic stage.

Flowchart of verification



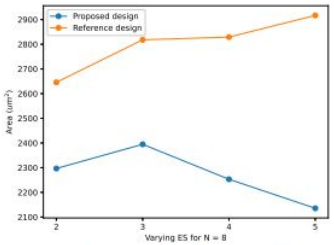
Accuracy Comparison

N	ES	No.of Test cases(2^(N)*2^(N))	Accuracy(%) (Reference design)[2]	Accuracy(%) (Proposed design)
8	1	65536	94.55	100
8	2	65536	95.27	100
8	3	65536	93.27	100
9	1	262144	95.52	100
9	2	262144	97.79	100
9	3	262144	98.91	100
9	4	262144	99.27	100
10	1	1048576	95.44	100
10	2	1048576	97.71	100
10	3	1048576	98.99	100
10	4	1048576	99.46	100
10	5	1048576	99.62	100

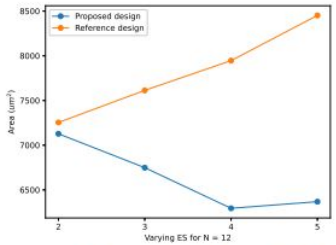
Reasons for accuracy improvement

- Guard, round, and sticky (GRS) bits are appended to the LSB of the mantissa in the proposed design to achieve accurate results.
- In the Alignment stage, the Sticky bit of the Small mantissa is adjusted where this Small mantissa is right-shifted based on the shift.
- The LSB of the result mantissa is adjusted in the Normalization stage depending on this Sticky bit.
- When the vector padded by the regime, exponent, and mantissa is right-shifted in the rounding stage, the Sticky bit is adjusted and Guard and round bits are selected depending on the shifted section.
- The proposed design uses parameters like regime maximum (Rmax) and regime minimum (Rmin).

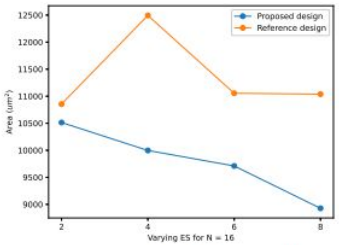
Synthesis Results



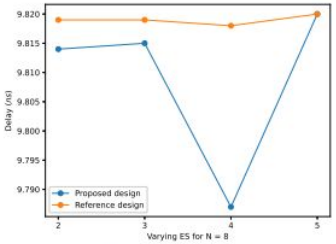
(a) Area for $N = 8$, varying ES



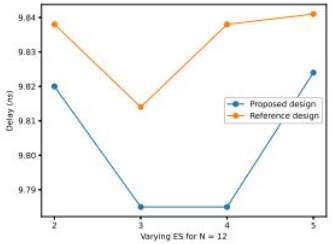
(b) Area for $N = 12$, varying ES



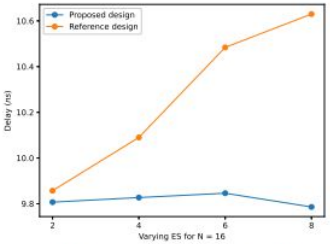
(c) Area for $N = 16$, varying ES



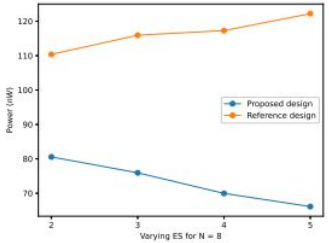
(d) Delay for $N = 8$, varying ES



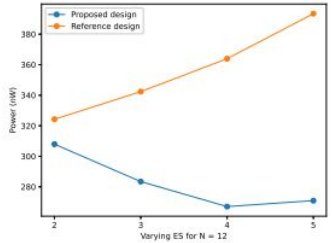
(e) Delay for $N = 12$, varying ES



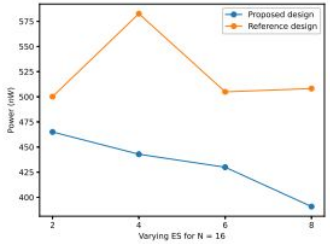
(f) Delay for $N = 16$, varying ES



(g) Power for $N = 8$, varying ES



(h) Power for $N = 12$, varying ES



(i) Power for $N = 16$, varying ES

Conclusion

- This work proposed an efficient hardware architecture for posit addition/subtraction with parameterized word size N and exponent size ES.
- Designs were generated for different values of N and ES; subsequently, they were synthesized using Cadence's 45nm standard cell library.
- It was observed that the proposed design is efficient in performance metrics such as area, delay, and leakage power compared to the available reference design. Furthermore, the proposed design is 100% accurate, on average 15% area, and 23% leakage power efficient while having a similar critical path delay when compared to the recent designs proposed in the literature.

References

[1] Gustafson and Yonemoto, “Beating Floating Point at Its Own Game: Posit Arithmetic,”Supercomput. Front. Innov.: Int. J., vol. 4, no. 2, p. 71–86,Jun. 2017. [Online]. Available: https://doi.org/10.14529/jsfi170206

[2] M.K.Jaiswal and Hayden K.-H.So, “Pacogen: A hardware posit arithmetic core generator,”IEEE Access, vol. 7, pp. 74 586–74 601, 2019

[3] https://github.com/mightymercado/PySigmoid

[4] M. K. Jaiswal and H. K.-H. So, “Universal number posit arithmetic generator on fpga,” in 2018 Design, Automation Test in Europe Conference Exhibition (DATE). IEEE, 2018, pp. 1159–1162.

[5] M. K. Jaiswal and H. K.-H. So, “Architecture generator for type-3 unum posit adder/subtractor,” in 2018 IEEE International Symposium on Circuits and Systems (ISCAS). IEEE, 2018, pp. 1–5.

[6] J. Hou, Y. Zhu, S. Du, and S. Song, “Enhancing accuracy and dynamic range of scientific data analytics by implementing posit arithmetic on fpga,” Journal of Signal Processing Systems, vol. 91, no. 10, pp. 1137–1148, 2019.

[7] 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). IEEE, 2018, pp. 334– 341.

[8] . Murillo, A. A. Del Barrio, and G. Botella, “Customized posit adders and multipliers using the flopoco core generator,” in 2020 IEEE International Symposium on Circuits and Systems (ISCAS). IEEE, 2020, pp. 1–5

[9] D. Shekhawat, A. Jangir, and J. G. Pandey, “A hardware generator for posit arithmetic and its fpga prototyping,” in 2021 25th International Symposium on VLSI Design and Test (VDAT). IEEE, 2021,

Thank You