**Flosit: Float/Posit Coarchitecture Exploiting Value Location in Neural Network**

*Yu-Liang Chen, Ren-Xiang Liu and Tsung-Chu Huang*

Department of Electronics Engineering, National Changhua University of Education, Changhua, Taiwan

*Abstract*—Posit is the most significant universal number format for compansating for IEEE 754 in both high dynamic-range computing and high precision neural network. However, it suffers considerable overhead and time penalty. In this paper, we firstly exploited the value locality of computing data and develop the switching Float/Posit algorithm to save systematic computing time. Secondly, we develop the combined switching architecture, called the Flosit for selecting proper operations efficiently. Thirdly, the Flosit architecture is integrated to further reduce the area and power costs. From prelimianary simulations, the proposed Flosit can automatically have both advantages in speed of Floating-Point and high-dynamic range of Posit.

# Introduction

Consumer Electronics have been utilized and developed with AI techniques. Computing for AI training has increased exponentially, doubling every 3.43 months on average [1], where IEEE 754 has become the most significant standard for computing [2]. Fig.(1) shows a n=32 or 64-bit floating-point (FP) number format, where 1-bit sign s=±1, *e*-bit exponent *E*=*U*-2*e-1*+1 and *m*-bit mantissa *M*, e=8/11 for single/double precision. Fraction *f*=1.+*M*×2*-m* with format *1.M*. The number will be , *eg*., π=3.14=0\_10000000\_10010010000111111011011.

(a)  (b) 

1. (a) FP and (b) Posit Formats.

With the need of high resolution and high dynamic-range (DR) computing, Posit has been the most efficient type-3 Unum as shown in Fig.1(b) [3-7]. Given a proper exponent size (*es=p* bits) with a value *e*, a Posit consists of a sign bit *s*, a *r*-bit regime *k* and *m*-bit mantissa and will be , *eg*.,π=3.14=0\_10\_01\_100100100001111110110101010.

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |

Figure 2. Basic arithmetic architectures of Floats/Posits

Fig.(2) shows the basic adder/subtractor/multiplier in FP/Posit formats. There are many references on error analysis, cost reduction, and performance improvement on Posit operations. The research teams of Posit and Unum have listed dozens of projects and papers on various arithmetic in various aspects [8]. Recently, the author in [9] summarized various implementation improvements [10-13]. However, there is still no integration on switching FP/Posit to improve systematic computing performance. In this paper we thus propose the integrated Flosit architecture in Sec. II. The performance is estimated in Section III before Section IV concludes this paper.

# Proposed Flosit Architecture

*A. Value Locality*

Locality of data/instructions includes reference and value localities. As Fig.3(a) shows, reference locality is well-known in a memory caching technique for getting both merits of the huge capacity of the main memory and the high speed of the cache. Value locality has also found by 2020 [13]. But it has been limited to programs and only exploited for approximation [14]. In the batch normalization (BN) are normalized to fit a new dynamic range (DR) for approximate quantization as shown in Fig.3(b). We found that when the computing values are mapped to a binary value, {within\_range, out\_of\_range}, the locality will be extremely high. Actually, that is because 99.99% of data values are in the scope of floating-point range. However, the computing systems may be down due to any possible out\_of\_range. Fig.3(c) shows the observation of a node in a neural network during all operation cycles, where illustrates most localities for floats and some for posits.

(a)(b)(c)

1. (a) Ref. (b) val. locality for approx. and (c) proposed switching.

*B. Flosit Algorithm and Combined Architecture*

Fig.4(a) shows a node in a computation model, where the FP and Posit operators are combined and selected by a flag as shown in Fig.4(b). This is the first paper that proposes to combine and switch both FP and Posit in a system. The basic concept is that we would rather use the Posit format for risky operations. Therefore, it will take a longer time to switch back to FP mode. In Fig.4(c), an up/down counter is triggered by the exception signals and compared with a given threshold to control the flag.

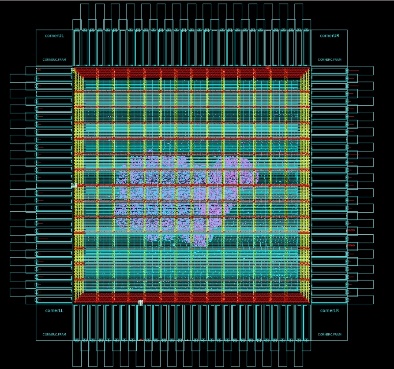
(a)  (b) (c)

1. (a) A network (b) combined-type Flosit (c) adaptive flag.

*C. Integrated Architecture*

Actually, parts of some elements and even modules can be shareable in the combined Flosit operators. They can be reduced by further integration. In this preliminary work, without loss of generality, only the major mantissa operator are integrated as illustrated in Fig.5(a). Note that the bypass, denormalization and normalization modules may not the same for FP and Posit so that the integration rate could not so high.

(a) 

(b) 

1. (a) Basic integrated Flosit operator, (b) Layout of Fig.6b.

Fig.6(a) and (b) separately show the integrated block diagrams of the adder and multiplier for the Flosit system.

(a) 

(b) 

1. Integrated Flosit (a) adder and (b) multiplier..

# Preliiminary Estimation

Since Adders and Multipliers are the most popular operators in most neural networks, only FP, Posit, combined and integrated types of adder and multiplier cells are designed in a 0.18m CMOS technology. Fig.5(b) shows one layout of the experimented Flosit multiplier cell. Table I shows the comparison of their areas, powers and critical-path times. We find that the area of a Posit cell are about 2.4 times of a FP cell with an about 1.6 times of critical path. The combined type should originally have a slightly larger area than their sum. However, because Silicon Compiler has the optimization capability, the area is reduced a bit after combination. The critical path of the combined type is obviously that of the FP with a selector.

1. Comparison of Arithmetic Cells.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Operator | Adder | | | Multiplier | | |
| Cost | area() | power | time | area() | power | time |
| Float32 | 50,419 | 1.84 | 10.30 | 71,289 | 3.97 | 9.40 |
| Posit32 | 119,720 | 2.75 | 17.28 | 176,942 | 5.18 | 15.83 |
| Combined Flosit | 153,334 | 3.46 | 17.39 | 246,415 | 6.99 | 15.91 |
| Integrated Flosit | 131,760 | 2.76 | 17.29 | 215,652 | 5.99 | 15.79 |

From the last row, the integrated adder and multiplier cells are separately reduced by 14% and 12.5% in area.

Taking advantage of value locality, the systematic computing performance can be improved to the level of the FP, although the combined and integrated operator cells almost suffer the combined area cost and the maximum critical path. In Table II, the average area, average power and average time of a FP multiplier cell is normalized as 1. Although the FP system can have the lowest cost and highest speed, it may be down due to a high out-of-range exception rate. When the proposed switching system is applied, the system can have the similar performance of the FP system without any exception. Moreover, the switching can have the merit of the Posit system with a potentially high precision performance.

1. Comparison of Systematic Views.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Normalized Cost | area | power | time | exception rate |
| Float32 | 1.0 | 1.0 | 1.0 | 1.0 |
| Posit32 | 2.5 | 1.3 | 2.3 | 1.7 |
| Combined Flosit | 3.5 | 1.8 | **1.0** | **1.7** |

# Conclusions

In this work, we have exploited the value locality of computing data in the precision computation system. We have also developed the switching Float/Posit algorithm to save systematic computing time. Additionally, as for the operating cells we have proposed the combined and the integrated types of designs. From the same systematic simulations using experimental results of operation cells, the proposed approach can have almost the same speed of the floating-point system and the precision and dependability of the Posit number system.

##### References

1. OpenAI. AI and Compute. https://openai.com/blog/ai-and-compute. May 16, 2018.
2. IEEE Computer Society (2019-07-22). IEEE Standard for Floating-Point Arithmetic. IEEE STD 754-2019. IEEE. pp. 1–84. doi: 10.1109/ IEEESTD. 2019. 8766229. ISBN 978-1-5044-5924-2.
3. Tichy, Walter F. (April 2016). "The End of (Numeric) Error: An interview with John L. Gustafson". Ubiquity – Information Everywhere. Association for Computing Machinery (ACM). 2016 (April): 1–14. doi:10.1145/2913029.
4. John L. Gustafson and I. Yonemoto. (February 2017) Beyond Floating Point: Next Generation Computer Arithmetic. [Online]. Available: https://www.youtube.com/ watch?v=aP0Y1uAA-2Y
5. Posit Working Group (2024-2-1). Standard for Posit™ Arithmetic (2022). https://posithub.org/docs/posit\_standard-2.pdf, March 2, 2022.
6. Posit Working Group (2024-2-1). Introduction to Posits using SoftPosit-Python. <https://posithub.org/docs/PositTutorial_Part1.html>
7. M. K. Jaiswal and H. K. . -H. So, "PACoGen: A Hardware Posit Arithmetic Core Generator," in IEEE Access, vol. 7, pp. 74586-74601, 2019, doi: 10.1109/ACCESS.2019.2920936
8. Posit Working Group (2024-2-1) Survey of Posit Hardware and Software Development Efforts, July 2019. [https://posithub.org/docs/PDS/Posit EffortsSurvey.html](https://posithub.org/docs/PDS/Posit%20EffortsSurvey.html)
9. Chien Chen Lai. Design of Neural Network Processing Unit Based on Modified Posit Format and Block Floating-Point Processing Technique. Master thesis, National Sun Yat-Sen University, Taiwan, Feb. 2023.
10. 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.
11. R. Chaurasiya et al., "Parameterized posit arithmetic hardware generator," 2018 IEEE 36th International Conference on Computer Design (ICCD), 2018, pp. 334-341.
12. Z. Carmichael, H. F. Langroudi, C. Khazanov, J. Lillie, J. L. Gustafson and D. Kudithipudi, "Deep positron: A deep neural network using the posit number system," 2019 Design, Automation & Test in Europe Conference & Exhibition (DATE), 2019, pp. 1421-1426.
13. J. Lu, C. Fang, M. Xu, J. Lin and Z. Wang, "Evaluations on Deep Neural Networks Training Using Posit Number System," in IEEE Transactions on Computers, vol. 70, no. 2, pp. 174-187, 2000.
14. R. Singh, G. S. Ravi, M. Lipasti and J. S. Miguel, "Value Locality Based Approximation With ODIN," in IEEE Computer Architecture Letters, vol. 19, no. 2, pp. 88-91, 1 July-Dec. 2020, doi: 10.1109/LCA.2020.3002542.