ReFloat: Low-Cost Floating-Point Processing in ReRAM for Accelerating Iterative Linear Solvers

Linghao Song†, Fan Chen‡, Hai Li§, Yiran Chen§

†*University of California, Los Angeles* ‡*Indiana University Bloomington* §*Duke University*

ABSTRACT

Resistive random access memory (ReRAM) is a promising technology that can perform low-cost and in-situ matrix-vector multiplication (MVM) in analog domain. Scientific computing requires high-precision floating-point (FP) processing. However, performing floating-point computation in ReRAM is challenging because of high hardware cost and execution time due to the large FP value range. In this work we present REFLOAT, a data format and an accelerator architecture, for low-cost and high-performance floating-point processing in ReRAM for iterative linear solvers. REFLOAT matches the ReRAM crossbar hardware and represents a block of FP values with reduced bits and an optimized exponent base for a high range of dynamic representation. Thus, REFLOAT achieves less ReRAM crossbar consumption and fewer processing cycles and overcomes the noncovergence issue in a prior work. The evaluation on the SuiteSparse matrices shows REFLOAT achieves 5.02× to 84.28× improvement in terms of solver time compared to a state-of-the-art ReRAM based accelerator.

1. INTRODUCTION

Scientific computing models a complex system with partial differential equations (PDEs), and in practice, the PDEs are converted to a linear system $A\mathbf{x} = \mathbf{b}$, and then solved through an iterative solver. The floating-point (FP) sparse matrix-vector multiplication (SpMV) is the key computation kernel. With the approaching the end of Moore's Law [7], ReRAM is considered as a promising candidate for implementing processing-in-memory (PIM) accelerators [1,3,6] that can provide orders of magnitude improvement of computing efficiency.

ReRAM-based processing engines for matrix-vector multiplication are fixed-point hardware in nature due to the fact that the matrix and the vector are respectively represented in *discrete* conductance states and voltage levels [8]. Many accelerators [1,3,6] are built with ReRAM and achieve reasonably good classification accuracy in machine learning, but they are for fixed-point processing.

It is non-trival to support floating-point processing in ReRAM. Take 64-bit double-precision number as an example, each floating-point number consists of a 1-bit sign (s), an 11-bit exponent (e), and a 52-bit fraction (f). The value is interpreted as $(-1)^s \times (1.b_{51}b_{50}...b_0) \times 2^{(e-1023)}$, yielding a dynamic data range from $\pm 2.2 \times 10^{-308}$ to $\pm 1.8 \times 10^{308}$. If ReRAM is used to support floating-point MVM operation, a large number of crossbars will be provisioned for fraction alignment, resulting in very high hardware cost (illustrated the problem in Section 3). To reduce the overhead, Feinberg *et al.* [2] proposes to truncate the higher bits in exponents, e.g., using the low 6 bits or module 64 of the exponent to represent each original value, while keeping the number of fraction bits the unchanged (52 bits). This ad-hoc solution *does not ensure the convergence* of iterative solves. Moreover, it may unnecessarily incur the hardware and execution time cost for the full precision of fractions.

This work is published at SC 2023 [5]. The lead author Linghao Song was a Ph.D. student at Duke University at the time this work was performed, and the preliminary version of this work appears as Chapter 5 of Linghao's doctoral dissertation [4].

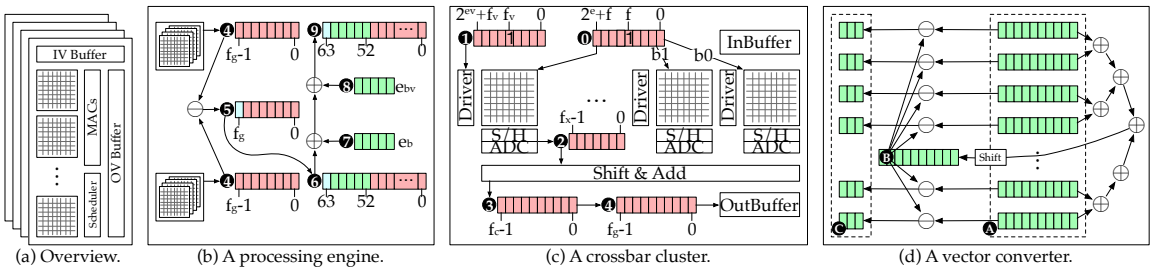
2. KEY INSIGHTS

The key insight of our solution is the *exponent value locality* among the elements in a matrix block. If we consider the whole matrix, the exponent values can span a wide range, e.g., up to 11 for a matrix, but the range is smaller within a block, e.g., at most 7 for the same matrix. It naturally motivates the idea of choosing an *exponent base* e_b for all exponents in a block, and storing only the *offsets* from e_b. For a matrix block, while the absolute exponent values can be large, the variation is not. For most blocks, by choosing a proper e_b, the offset values are much smaller than the absolute exponent values, thus reducing the number of bits required.

Hardware Cost and Performance Analysis. To map the FP matrix to ReRAM accelerators, we need C ReRAM crssbars. $C = 4 \times (2^{e_M} + f_M + 1)$, where e_M and f_M respectively is the bit length for the exponent and fraction of a matrix. The hardware cost increases *exponentially* with e_M while linearly with f_M. To process a FP multiplication in ReRAM, it takes T cycles. T is calculated as $T = (2^{e_v} + f_v + 1) + (2^{e_M} + f_M + 1) - 1$, where e_v and f_v respectively is the bit length for the exponent and fraction of a vector. We can observe that the computation latency increases *exponentially* with both e_v and e_M, while linearly with f_v and f_M.

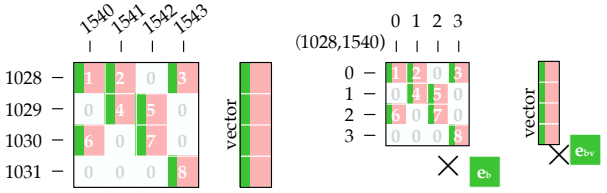
Truncation and Non-convergence. The design of the state-of-the-art ReRAM-based accelerator [2] for floating-point SpMV is driven exactly by the conclusion of our analysis—reducing the number of bits for exponent. However, this solution adopts an ad-hoc approach that simply truncates a number of high order bits in exponent. Specifically, with the low 6 bits of exponent, it uses module 64 of the exponent to represent each original value. Unfortunately, bit truncation may lead to significantly slower convergence and, more importantly, *non-cvonvergence*. Thus, the solution proposed in [2] may break the correctness of the iterative solver.

Opportunity: Value Locality While the number of exponent bits has a major impact on the hardware cost and computation cycles, sufficient accuracy of exponents is needed to ensure convergence. We leverage an intuitive observation of matrix element values—*exponent value locality*—to significantly the number of bits for exponents while keeping enough accuracy. In the real-word data matrix, the values of closely located



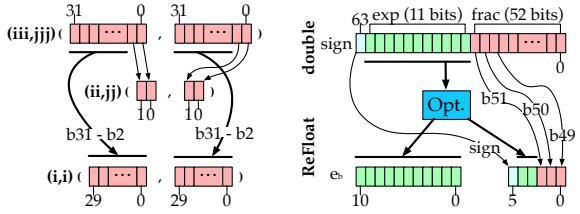
(a) Overview. (b) A processing engine. (c) A crossbar cluster. (d) A vector converter.

Figure 1: (a) the accelerator architecture overview. Architectures of (b) a processing engine for floating-point MVM on a matrix block, (c) a crossbar cluster for fixed-point MVM, and (d) a vector converter.



(a) A block in full precision. (b) A block in ReFloat.

Figure 2: Comparison of a matrix block (a) in original full precision format and (b) in REFLOAT format.



(a) Index conversion. (b) Value conversion.

Figure 3: The conversion of index and value in floating-point format to REFLOAT format.

elements are not too far from each other. The values are around a common base value.

Based on these insights, we propose REFLOAT, a principled approach with a novel data format and an accelerator architecture to reduce FP processing cost in ReRAM. In REFLOAT, we reduce bit length for the exponent with the exponent offsets from a base, rather than truncating exponents, are processed by a flexible and fine-grained FP number representation.

In REFLOAT, we naturally control the accuracy by the number of bits e allocated for the offsets, which is less than the number of exponent bits necessary to represent the offsets precisely. When an offset is larger (smaller) than the largest (smallest) offset representable by e bits, the largest (smallest) value of e bits is used for the offset. With e-bit exponent offset, the range of exponent values is $[e_b - 2^{(e-1)} + 1, e_b + 2^{(e-1)} - 1]$. Intuitively, given e and e_b, this system can precisely represent the exponent values that fall into a "window" around e_b, while the "size of the window" is determined by $2^{(e-1)}$. Then, selecting e_b becomes an optimization problem that minimizes the difference between the exponents of the original matrix block and the exponents with e_b and e-bit offsets.

3. MAIN CONTRIBUTIONS AND RESULTS

A Novel Data Format. We propose a novel data format for low-cost floating-point processing in ReRAM for scientific computing. We present the conversion method from default IEEE floating-point format to the proposed data format and the computation method with the proposed data format.

An Accelerator Architecture. We present an accelerator architecture to support the proposed data format. Existing computing platforms can only emulate the functionality of REFLOAT data format, but a hardware architecture is needed to fully amplify the processing efficiency.

We open-sourced the implementation of REFLOAT and the work [5] has been awarded three ACM artifact badges.

Key Results. Our experiments show that REFLOAT achieves a speedup of 5.02× to 84.28× compared with a state-of-the-art ReRAM-based accelerator [2] for scientific computing even with the assumption that the accelerator [2] functions the same as FP64 solvers.. For the 12 matrices evaluated in iterative solvers, only 3 bits for exponent and 8 or 16 bits for fraction are sufficient to ensure convergence. In comparison, [2] uses 6 bits for exponent and 51 bits for fraction without guaranteeing convergence.

Key Contributions.

- We present the insights on leveraging data locality in real-word matrix for the opportunities to compress data representation and process in low hardware cost in ReRAM.
- We propose REFLOAT, a data format with the conversion method (shown in Figure 3) and the computation for floating-point processing in ReRAM which consumes less number of crossbars (hardware resource) and less cycles.
- We present an accelerator architecture (shown in Figure 1) to support the proposed REFLOAT data format for full potential.

REFERENCES

[1] Ping Chi et al. Prime: A novel processing-in-memory architecture for neural network computation in reram-based main memory. In *ISCA*, 2016.

[2] Ben Feinberg et al. Enabling scientific computing on memristive accelerators. In *ISCA*, 2018.

[3] Ali Shafiee et al. Isaac: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars. In *ISCA*. IEEE, 2016.

[4] Linghao Song. *Accelerator Architectures for Deep Learning and Graph Processing*. Duke University, 2020.

[5] Linghao Song, Fan Chen, Hai Li, and Yiran Chen. Refloat: Low-cost floating-point processing in reram for accelerating iterative linear solvers. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–15, 2023.

[6] Linghao Song et al. Pipelayer: A pipelined reram-based accelerator for deep learning. In *HPCA*. IEEE, 2017.

[7] M Mitchell Waldrop. The chips are down for moore's law. *Nature News*, 530(7589):144, 2016.

[8] H-S Philip Wong et al. Metal–oxide rram. *Proc. of IEEE*, 2012.