Computation at the Source: PIM Methodology Survey and Case Studies

Olivia Fann*
Amherst College

Alston Chen*
Computer Science
Amherst College

Lillian Pentecost
Amherst College

***Abstract*—Processing in memory (PIM) architectures could significantly improve efficiency for data intensive applications, an appealing promise as neural networks and other forms of artificial intelligence are used increasingly widely. To analyze proposed architectures without the need for specialized or future hardware, researchers commonly rely on varying combinations of simulators and estimation tools, an approach which makes verifying or comparing results difficult or even impossible. Our research investigates this range of methodologies for evaluating PIM architectures, identifying prevalent tools. We find that although several tools, one being Ramulator, display few to no issues, many others come with significant errors that call their accuracy into question–in particular, CACTI, the most widely used tool of our survey, has extensive issues in the source code and produces results which are far from realistic. Through this research, we hope to identify strengths and weaknesses with PIM simulators, with an overall goal of developing a more centralized approach to evaluating PIM architectures.**

***Index Terms*—Processing in Memory, Data Movement, Simulators**

I. Introduction

With the recent rise of deep learning and increasingly data-centric applications, the bottleneck of data movement between memory and the CPU becomes increasingly costly. One category of architectures finds a solution using processing in memory (PIM), in which computation is placed in or near memory to reduce inefficient data movement (Figure 1). Much research has been conducted in recent years into potential PIM architectures, which are evaluated using simulators and other estimation tools. But considering the multitude of tools, many of which are not actively maintained, how accurate are those results? In our research, we survey the varying ways researchers have evaluated PIM architectures, looking for patterns and recurring methods. We then assess the accuracy and usability of referenced tools, conducting in-depth analyses of Ramulator and CACTI-7.

II. Methodology

In order to categorize common methodologies, we surveyed 115 papers [4] referenced in Mutlu et al.'s "Modern Primer on Processing in Memory" [3] and from recent ISPASS, ISCA, HPCA, and MICRO conferences. We tagged each paper with keywords such as 'In-House Simulator,' 'FPGA/Synthesis,' and with the specific tools referenced in their methodology. Based on our results from this survey, we identified the most commonly used tools and assessed their accuracy and current usability (maintenance, compatibility with current libraries and packages). For common and functional tools, we conducted a detailed review of inputs and outputs, testing functionality across a range of configurations, and identifying issues and inconsistencies.

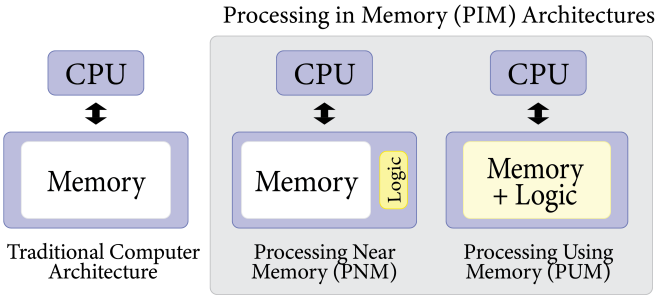

Fig. 1. High-level comparison between traditional and processing in memory computer architectures

III. Results

A. Survey Results

We observed in our survey that rather than relying on one singular tool for full-system evaluation, researchers obtained metrics from multiple separate tools targeting individual components in their architecture. Represented in Figure 2, we also identified a subset of tools which PIM researchers consistently used. Of the tools in our survey, CACTI [1], which models cache/memory, off-chip IO, and 3D stacking, had the highest number of occurrences, being used 30 times across its multiple versions. Ramulator [2], a fast and cycle-accurate DRAM simulator, was another popular tool especially recently, with 9 occurrences from 2021 to 2024. Other common tools included Gem5 and McPat.

B. Case Studies: Ramulator and CACTI 7

We found that Ramulator is actively maintained (Ramulator 2 was published in 2023), with detailed and accurate results. One shortcoming is its somewhat inflexible user interface, which fails to provide clarity into the full meaning and impact of each input. On the other hand, CACTI 7 (the most recent version) has a detailed and easily configurable interface but

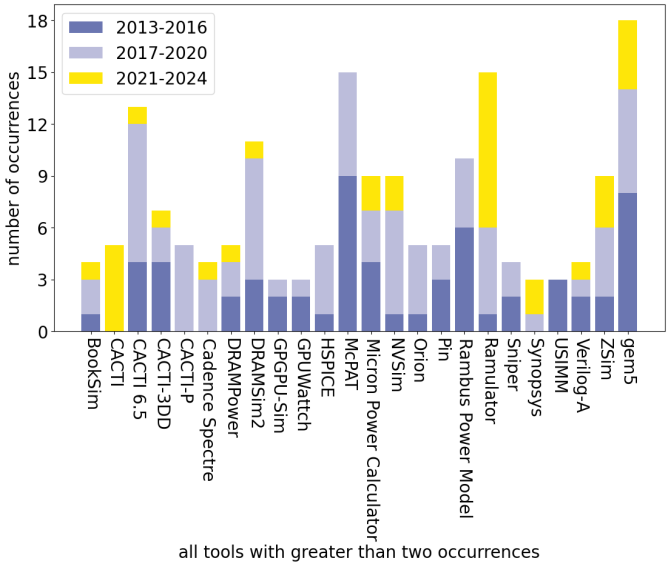

Fig. 2. Simulation tools vs. number of occurrences

contains several significant issues and inconsistencies, producing entirely inaccurate results for several metrics (Figure 3). While we were able to locate and correct the source code for several of these issues (primarily the lack of parsing for key input parameters and a faulty while loop), they raise concerns over the tool's overall accuracy. CACTI 7 also does not appear actively maintained; issues raised by users remain unresolved after several years. Without updates, it becomes progressively more outdated as the technologies used in proposed architectures become more advanced over time.

IV. Conclusion

Effective evaluation frameworks are key for designing new PIM architectures. However, researchers in the field use an extremely wide range of simulation tools and methodologies, making it extremely difficult to verify accuracy of results or compare distinct architectures. There is a clear need for a more centralized and effective framework for evaluating PIM architectures, towards which we hope this work can be a meaningful first step.

Acknowledgment

We would like to thank our lab mates for their support during this process.

Metric	Before modification	After modification
IO Area (sq.mm)	inf	2.90152
t_RCD (Row to column command delay) (ns)	1.14548e+107	7.84292
t_RAS (Row access strobe latency) (ns)	1.14548e+107	29.194
t_RC (Row cycle) (ns)	1.14548e+107	49.1003
t_CAS (Column access strobe latency)	2.29096e+107	16.9298
t_RP (Row precharge latency)	1.14548e+107	20.648
t_RRD (Row activation to row activation delay)	1.14548e+107	4.06969
Activation energy (nJ)	6.14115e+54	1.81143
Read energy (nJ)	6.14115e+54	1.11818
Write energy (nJ)	6.14115e+54	1.1182
Precharge energy (nJ)	6.14115e+54	1.67972
TSV latency overhead (ns)	1.14548e+107	0.741733
TSV energy overhead per access (nJ)	6.14115e+54	0.0481619

Fig. 3. Difference in outputs before and after modifying CACTI source code

References

[1] Norman P. Jouppi et al. "CACTI-IO: CACTI with off-chip power-area-timing models". In: *Proceedings of the International Conference on Computer-Aided Design*. ICCAD '12. San Jose, California: Association for Computing Machinery, 2012, pp. 294–301. ISBN: 9781450315739. DOI: 10.1145/2429384.2429446. URL: https://doi.org/10.1145/2429384.2429446.

[2] Yoongu Kim, Weikun Yang, and Onur Mutlu. "Ramulator: A Fast and Extensible DRAM Simulator". In: *IEEE Comput. Archit. Lett.* 15.1 (Jan. 2016), pp. 45–49. ISSN: 1556-6056. DOI: 10.1109/LCA.2015.2414456. URL: https://doi.org/10.1109/LCA.2015.2414456.

[3] Onur Mutlu et al. "A Modern Primer on Processing in Memory". In: *CoRR* abs/2012.03112 (2020). arXiv: 2012.03112. URL: https://arxiv.org/abs/2012.03112.