PIMCoSim: HW/SW Co-Simulator for Exploring Processing In Memory Architectures

Seongmo An, Jinyoung Shin, Sangho Lee, and Seung Eun Lee*
Dept. of Electronic Engineering, Seoul National University of Science and Technology
Seoul, Republic of Korea
*seung.lee@seoultech.ac.kr

ABSTRACT

The growing intricacy of PIM architecture, coupled with extended simulation durations and the constraints of software simulators, presents challenges for PIM developers. In this paper, we propose a HW/SW co-simulator for exploring PIM architectures. The proposed co-simulator allows PIM developers to simulate various PIM architectures with a configuration file and PIM-specific instructions. Experimental results demonstrated that the proposed co-simulator is feasible and time efficient by showing operation time reduction is about 7.5 times on average.

KEYWORDS

Processing In Memory, co-simulator, time reduction

ACM Reference Format:
Seongmo An, Jinyoung Shin, Sangho Lee, and Seung Eun Lee*. 2023. PIMCoSim: HW/SW Co-Simulator for Exploring Processing In Memory Architectures. In *ACM Student Research Competition (ACM SRC '23), October 21–25, 2023, Vienna, Austria.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/XXXXXX

1 INTRODUCTION

The processing in memory (PIM) has been the subject of considerable attention and scholarly inquiry in recent years[5]. The proposition of diverse PIM architectures has led to a concomitant rise in the significance of simulation tools[4]. However, the increasing diversity and complexity of PIM architecture necessitate longer simulation times. Furthermore, debugging hardware is restricted by software simulators, so that software simulation alone has limitations in verifying hardware operations[1]. Consequently, we propose PIMCoSim, a HW/SW co-simulator for PIM architectures that enables the exploration of diverse PIM architectures. We designed an instruction set that supports PIM operations and made the processing element (PE) of PIM implemented in hardware receive the instructions. This allows developers to implement various PIM architectures on the simulator and assess the hardware attributes.

Related works for PIM simulators are mainly composed of software. In the case of PIMSim, it receives the configuration of DRAM, computing unit and performs simulations for three modes with trade-offs between speed/accuracy[3]. The Sim^{2}PIM for more flexible simulation of PIM allows for determination of design details and performance measurement during development, and guarantees independence from the host CPU, enabling faster simulation by simulating only the PIM[2]. The proposed PIMCoSim receives configurations for DRAM, SRAM, and PE and PIM-specific 32-bit application instruction codes for simulations. The co-simulator performs accelerated PIM operations by transmitting instructions to the hardware-implemented PE. In addition, the arithmetic unit included in the PE hardware is flexibly replaceable so that developers are able to measure the performance of their own PE arithmetic unit and simulate hardware characteristics.

2 SYSTEM ARCHITECTURE

Figure 1 depicts the overall structure of the proposed co-simulator system. First, when the utilizer inputs a configuration file for DRAM, SRAM, and PE into the simulator, the simulator establishes the models in the PIM library based on the configurations. Next, the utilizer is able to write the application code with PIM-specific instructions and feed it into the simulator. The PIM-specific instruction set consists of commands such as *pim_load* and *pim_store*, which exchange data with the PIM, *pim_add*, *pim_sub*, *pim_mul*, which perform binary operations, and *pim_cp*, which allows data to be copied between PEs. The commands are converted into 32-bit instructions in order by the instruction generator and delivered to the PE hardware via SPI. Each command contains information including which PE to utilize and addresses for DRAM and SRAM, so the utilizer is capable of implementing various PIM structures by establishing the PIM components through the configuration file and combining various instructions in the application code.

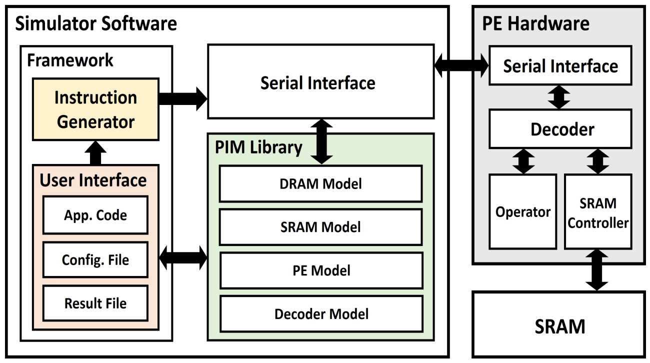


Figure 1: The overview of the proposed co-simulator system.

After the PE hardware receives an instruction, it interprets the instruction in the decoder. In case of selecting multiple PEs in the received instruction, the operation is repeated according to the number of selected PEs. Once the operation is completed, the result is transmitted to the simulator software via SPI.

When an instruction is completed, the simulator calculates and records contents such as operation results, simulation time, predicted operation time, and the number of accesses to DRAM and SRAM in the result file based on the configuration file. Upon completion of all instructions, the utilizer is able to examine the result file.

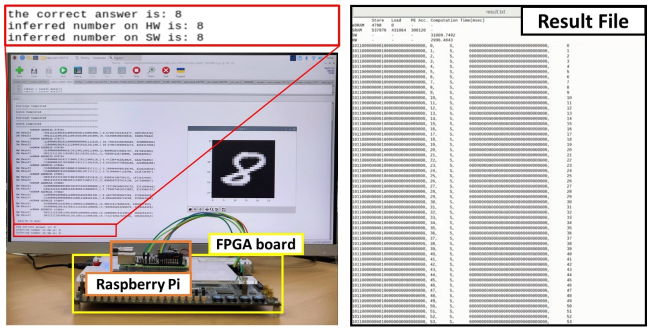
3 EXPERIMENT

As depicted in Figure 2, the co-simulator was implemented as a Python program on a Raspberry Pi, while the PE hardware was implanted on an FPGA. To verify the feasibility of the co-simulator, we also developed a software simulator that performs operations of the PIM-specific instructions. We conducted experiments by inputting instruction codes that perform LeNet-1 inference operations into both simulators, inferring a total of 150 MNIST test datasets simultaneously, and comparing the inference results. Additionally, we measured and compared the operation time for various applications to verify time reduction.

The results of the experiments are presented in Figure 3. The MNIST inference results indicate that comparable outcomes were obtained based on the similarity of the accuracy by label between both simulators, with a discrepancy of about 1.3% in the total accuracy rate. The comparison of operation time by application revealed that the co-simulator was about 12.2 times faster at maximum in 3×3 convolution and about 7.5 times faster on average.

4 CONCLUSION

This paper proposes a HW/SW co-simulator to evaluate various PIM architectures. Developers are able to simulate their own PIM architectures with a configuration file and a PIM-specific instruction set. Experimental results confirm availability and time efficiency of the co-simulator.



```
the correct answer is: 8
inferred number on HW is: 8
inferred number on SW is: 8
```

```
       Store    Load    PE Acc.  Computation Time(msec)
eDRAM  4798     0       -        -
SRAM   537876   431064  300120   -
SW     -        -       -        31009.7492
HW     -        -       -        2996.4843
10110000000010000000000000000000, 0,     5,     00000000000000000000000000000000,     0
10110000000010000000000000000000, 1,     5,     00000000000000000000000000000000,     1
10110000000010000000000000000000, 2,     5,     00000000000000000000000000000000,     2
10110000000010000000000000000000, 3,     5,     00000000000000000000000000000000,     3
10110000000010000000000000000000, 4,     5,     00000000000000000000000000000000,     4
10110000000010000000000000000000, 5,     5,     00000000000000000000000000000000,     5
10110000000010000000000000000000, 6,     5,     00000000000000000000000000000000,     6
10110000000010000000000000000000, 7,     5,     00000000000000000000000000000000,     7
10110000000010000000000000000000, 8,     5,     00000000000000000000000000000000,     8
10110000000010000000000000000000, 9,     5,     00000000000000000000000000000000,     9
10110000000010000000000000000000, 10,    5,     00000000000000000000000000000000,     10
10110000000010000000000000000000, 11,    5,     00000000000000000000000000000000,     11
10110000000010000000000000000000, 12,    5,     00000000000000000000000000000000,     12
10110000000010000000000000000000, 13,    5,     00000000000000000000000000000000,     13
10110000000010000000000000000000, 14,    5,     00000000000000000000000000000000,     14
10110000000010000000000000000000, 15,    5,     00000000000000000000000000000000,     15
10110000000010000000000000000000, 16,    5,     00000000000000000000000000000000,     16
10110000000010000000000000000000, 17,    5,     00000000000000000000000000000000,     17
10110000000010000000000000000000, 18,    5,     00000000000000000000000000000000,     18
10110000000010000000000000000000, 19,    5,     00000000000000000000000000000000,     19
10110000000010000000000000000000, 20,    5,     00000000000000000000000000000000,     20
10110000000010000000000000000000, 21,    5,     00000000000000000000000000000000,     21
10110000000010000000000000000000, 22,    5,     00000000000000000000000000000000,     22
10110000000010000000000000000000, 23,    5,     00000000000000000000000000000000,     23
10110000000010000000000000000000, 24,    5,     00000000000000000000000000000000,     24
10110000000010000000000000000000, 25,    5,     00000000000000000000000000000000,     25
10110000000010000000000000000000, 26,    5,     00000000000000000000000000000000,     26
10110000000010000000000000000000, 27,    5,     00000000000000000000000000000000,     27
10110000000010000000000000000000, 28,    5,     00000000000000000000000000000000,     28
10110000000010000000000000000000, 29,    5,     00000000000000000000000000000000,     29
10110000000010000000000000000000, 30,    5,     00000000000000000000000000000000,     30
10110000000010000000000000000000, 31,    5,     00000000000000000000000000000000,     31
10110000000010000000000000000000, 32,    5,     00000000000000000000000000000000,     32
10110000000010000000000000000000, 33,    5,     00000000000000000000000000000000,     33
10110000000010000000000000000000, 34,    5,     00000000000000000000000000000000,     34
10110000000010000000000000000000, 35,    5,     00000000000000000000000000000000,     35
10110000000010000000000000000000, 36,    5,     00000000000000000000000000000000,     36
10110000000010000000000000000000, 37,    5,     00000000000000000000000000000000,     37
10110000000010000000000000000000, 38,    5,     00000000000000000000000000000000,     38
10110000000010000000000000000000, 39,    5,     00000000000000000000000000000000,     39
10110000000010000000000000000000, 40,    5,     00000000000000000000000000000000,     40
10110000000010000000000000000000, 41,    5,     00000000000000000000000000000000,     41
10110000000010000000000000000000, 42,    5,     00000000000000000000000000000000,     42
10110000000010000000000000000000, 43,    5,     00000000000000000000000000000000,     43
10110000000010000000000000000000, 44,    5,     00000000000000000000000000000000,     44
10110000000010000000000000000000, 45,    5,     00000000000000000000000000000000,     45
10110000000010000000000000000000, 46,    5,     00000000000000000000000000000000,     46
10110000000010000000000000000000, 47,    5,     00000000000000000000000000000000,     47
10110000000010000000000000000000, 48,    5,     00000000000000000000000000000000,     48
10110000000010000000000000000000, 49,    5,     00000000000000000000000000000000,     49
10110000000010000000000000000000, 50,    5,     00000000000000000000000000000000,     50
10110000000010000000000000000000, 51,    5,     00000000000000000000000000000000,     51
10110000000010000000000000000000, 52,    5,     00000000000000000000000000000000,     52
10110000000010000000000000000000, 53,    5,     00000000000000000000000000000000,     53
```

Figure 2: Experimental environments and display of simulation results

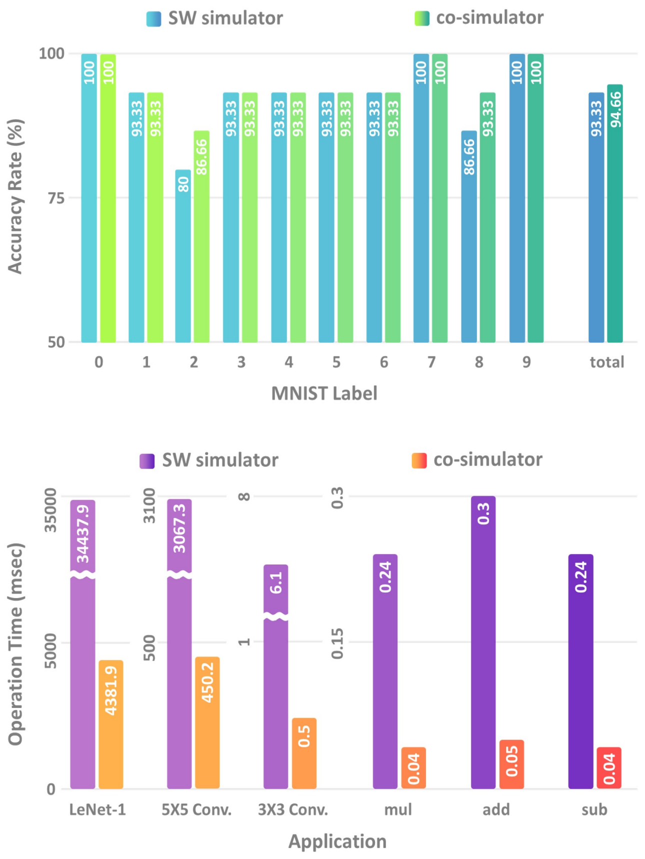


Figure 3: Experimental results

ACKNOWLEDGMENTS

This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) (2022-0-01013, Development of DRAM PIM semiconductor technology for enhanced computing function for edge).

REFERENCES

[1] Yosuke Kurimoto, Yusuke Fukutsuka, Ittetsu Taniguchi, and Hiroyuki Tomiyama. 2013. A hardware/software cosimulator for Network-on-Chip. In *2013 International SoC Design Conference (ISOCC)*. 172–175. https://doi.org/10.1109/ISOCC.2013.6863964

[2] Paulo C. Santos, Bruno E. Forlin, and Luigi Carro. 2021. Sim2PIM: A Fast Method for Simulating Host Independent & PIM Agnostic Designs. In *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. 226–231. https://doi.org/10.23919/DATE51398.2021.9474104

[3] Sheng Xu, Xiaoming Chen, Ying Wang, Yinhe Han, Xuehai Qian, and Xiaowei Li. 2019. PIMSim: A Flexible and Detailed Processing-in-Memory Simulator. *IEEE Computer Architecture Letters* 18, 1 (2019), 6–9. https://doi.org/10.1109/LCA.2018.2885752

[4] Chao Yu, Sihang Liu, and Samira Khan. 2021. MultiPIM: A Detailed and Configurable Multi-Stack Processing-In-Memory Simulator. *IEEE Computer Architecture Letters* 20, 1 (2021), 54–57. https://doi.org/10.1109/LCA.2021.3061905

[5] Changwu Zhang, Hao Sun, Shuman Li, Yaohua Wang, Haiyan Chen, and Hengzhu Liu. 2023. A Survey of Memory-Centric Energy Efficient Computer Architecture. *IEEE Transactions on Parallel and Distributed Systems* 34, 10 (2023), 2657–2670. https://doi.org/10.1109/TPDS.2023.3297595