A Python Framework for SPICE Circuit Simulation of In-Memory Analog Computing Circuits

Md Hasibul Amin, Mohammed Elbtity, Ramtin Zand
Department of Computer Science and Engineering, University of South Carolina, Columbia, SC 29208, USA

I. INTRODUCTION

With the increased attention to memristive-based in-memory analog computing (IMAC) architectures [1] as an alternative for energy-hungry computer systems for data-intensive applications, a tool that enables exploring their device- and circuit-level design space can significantly boost the research and development in this area. Thus, in this paper, we develop IMAC-Sim, a circuit-level simulator for the design space exploration and multi-objective optimization of IMAC architectures. IMAC-Sim is a Python-based simulation framework, which creates the SPICE netlist of the IMAC circuit based on various device- and circuit-level hyperparameters selected by the user, and automatically evaluates the accuracy, power consumption and latency of the developed circuit using a user-specified dataset. IMAC-Sim simulates the interconnect parasitic resistance and capacitance in the IMAC architectures, and is also equipped with horizontal and vertical partitioning techniques to surmount these reliability challenges [2]. In this abstract, we perform controlled experiments to exhibit some of the important capabilities of the IMAC-Sim.

II. PROPOSED IMAC-SIM FRAMEWORK

Figure 1 illustrates the structure of the IMAC-Sim framework, which includes four Python modules: 1) testIMAC, 2) mapWB, 3) mapLayer, and 4) mapIMAC. The testIMAC module runs as a parent file, which controls the deployment of DNN workloads on IMAC architectures, as well as assessing their performance and accuracy. The testIMAC module receives two sets of inputs from the user, as shown in Fig. 1. First, it takes trained weights and biases, Network Topology (\(T_N\)), Horizontal Partitioning (\(H_P\)) and Vertical Partitioning (\(V_P\)) information, and the device- and circuit-level hyperparameters to deploy the DNN on the IMAC architecture. Next, it receives test dataset (TestData), test label (TestLabel), and Number of test samples (\(N_s\)) to assess the developed IMAC circuit. Table I lists the hyperparameters of the IMAC-Sim.

The functionality of testIMAC module is demonstrated in Algorithm I. For each input sample, testIMAC first stores the target labels in an array called label. It then calls another python module, mapIMAC, which is responsible for creating the SPICE netlist of the IMAC circuit. For this purpose, mapIMAC calls another python module, mapLayer, which builds separate subcircuits for each of the layers in DNN including their interconnect parasitics and required partitioning as requested by user through \(H_P\) and \(V_P\) arrays.

mapLayer modules returns the SPICE files for all of the layer subcircuits to mapIMAC, which concatenates them to form the main IMAC SPICE file. Finally, testIMAC runs the SPICE simulation for the developed IMAC SPICE file using the input voltages generated from the test dataset, and extracts the outputs of the last layer in IMAC circuit (out) and compares them with the label to obtain the accuracy. Moreover, testIMAC measures the average power consumption and latency of the circuit across various inputs and reports them to the user.

### Table I

| Parameter | Value |
|-----------|-------|
| Transistor Technology Node | FinFET, CMOS |
| Nominal Voltages \([VDD, VSS]\) | \([R_{low}, R_{high}]\) |
| Neuron Circuit Model | \(T_N = [layer_1,...,layer_n]\) |
| Synaptic Technology | \(V_P = [v_{p1}, v_{p2},..., v_{pn-1}]\) |
| Network Topology | \(H_P = [h_{p1}, h_{p2},..., h_{pn-1}]\) |
| Vertical Partitioning | \([G_1, G_2,..., G_{n-1}]\) |
| Horizontal Partitioning | \([Width, Height]\) |
| Differential Amplifier Gains | \([t_{sampling}]\) |
Algorithm 1: IMAC-Sim Framework

Input: test dataset (TestData), test label (TestLabel), weights, biases, Network Topology (TN), No. of test samples (N), Horizontal Partitioning (HP), Vertical Partitioning (VP), R_low, R_high

1. Initialize: Error = 0, PWR = 0, TN = [L_1, L_2, ..., L_N], HP = [h_p1, h_p2, ..., h_p_n], VP = [v_p1, v_p2, ..., v_p_m]

2. for i = 1 to N do
   
   W^*, W^*, B* ≜ mapWB (weights, biases)

3. call target labels for input i

4. for j = 1 to len(TN) - 1 do
   
   if (out≠label) then
      
      Error += 1;
   
end

5. Run the SPICE simulation for IMAC circuit

6. Concatenate the layer subcircuits

7. IMAC SPICE circuit

8. IMAC-Sim supports any arbitrary value for horizontal and vertical partitions increases both accuracy and power dissipation. In another test, we increased the number of partitions to HP = [16, 8, 8] and VP = [8, 8, 1], as listed in the last row of Table III. Based on the results obtained from IMAC-Sim, this deployment scenario results in a high accuracy of 94.04% at the cost of 60% higher power dissipation. These types of trade-offs are important information that can be provided to developers by IMAC-Sim framework.

B. Effect of Memristive Device Technology

We investigate the impact of memristive device technology on the performance of the IMAC architecture using four resistive technologies: MRAM [4], RRAM [5], CBRAM [6], and PCM [7]. Here, R_m and R_off values for different devices are changed for each run, while HP and VP are fixed to [13,4,3] and [4,3,1] respectively. Results listed in Table IV show that PCM-based IMAC architecture can achieve a high accuracy of 96.66%, while consuming significantly less power compared to other technologies. This can be justified by the larger resistance of the PCM devices.

REFERENCES

[1] M. Elbiby et al., “An in-memory analog computing co-processor for energy-efficient cnn inference on mobile devices,” in 2021 IEEE Computer Society Annual Symposium on VLSI (ISVLSI), 2021, pp. 188–193.

[2] M. H. Amin, M. Elbiby, and R. Zand, “Interconnect parasitics and partitioning in fully-analog in-memory computing architectures,” in 2022 IEEE International Symposium on Circuits and Systems (ISCAS), 2022.

[3] M. H. Amin et al., “Mram-based analog sigmoid function for in-memory computing,” in The Great Lakes Symposium on VLSI, 2022, pp. 319–323.

[4] R. Zand, A. Roohi, and R. F. DeMara, “Fundamentals, modeling, and application of magnetic tunnel junctions,” Nanoscale Devices: Physics, Modeling, and Their Application, p. 337, 2018.

[5] C. Li et al., “Analogue signal and image processing with large memristor crossbars,” Nature Electronics, vol. 1, no. 1, p. 52, 2018.

[6] Y. Shi et al., “Neuroinspired unsupervised learning and pruning with subquantum cbram arrays,” Nature communications, 2018.

[7] G. W. Burr et al., “Experimental demonstration and tolerancing of a large-scale neural network (165 000 synapses) using phase-change memory as the synaptic weight element,” IEEE Transactions on Electron Devices, vol. 62, no. 11, pp. 3498–3507, 2015.