SIAM: Scalable In-Memory Acceleration With Mesh

Gokul Krishnan[1,+,*], Sumit K. Mandal[1,+,*], Zhenhua Zhu[2,+], Chaitali Chakrabarti[1], Jae-sun Seo[1], Yu Wang[2], Umit Y. Ogras[1], and Yu Cao[1,*]

[1]Arizona State University
[2]Tsinghua University
[*]Corresponding Authors: {gkrish19, skmandal, yu.cao}@asu.edu
[+]these authors contributed equally to this work

ABSTRACT

With the widespread use of Deep Neural Networks (DNNs), machine learning algorithms have evolved in two diverse directions: one with ever-increasing connection density for better accuracy and faster training, and the other with more compact sizing for energy efficiency. The former is more appropriate for the cloud environment, while the latter is suitable for edge devices. In this paper, we illustrate that the traditional point-to-point-based interconnection architecture is incapable of handling the high on-chip data-flow demands of modern DNNs. We propose to use the network-on-chip (NoC) architecture with different levels of complexity to manage on-chip communication for diverse set of DNNs. The contribution of this work is threefold. We first develop a new simulation framework that integrates computing elements with various interconnect topologies to benchmark circuit and system performance of DNNs. Second, we evaluate the accuracy of the DNN on the proposed architecture and provide a compensation technique to improve performance. Finally, we demonstrate that the NoC-based architecture with in-memory computing achieves up-to 6× improvement in energy-delay-area product for VGG-19 inference as compared to the state-of-the-art results for both SRAM and ReRAM-based cross-bars.

Contents

1 Introduction

SIAM is a performance bench-marking tool for in-memory computing (IMC) architectures. The tool combines both circuits level parameters and the interconnect cost to estimate the overall architecture performance (area, latency, energy, throughput). In order to achieve this, a parametric heterogeneous IMC architecture and an NoC interconnect is utilized. The tool supports a range of technologies from SRAM to FET for in-memory acceleration. Combined with a cycle accurate NoC simulator, SIAM provides a complete performance analysis for researchers. This tool uses DNN to IMC mapping, circuit-level estimation, NoC evaluation, IMC tile to NoC router mapping and scheduling.

2 Related Work

3 Tool Flow

Multiple simulators exist to evaluate the performance of DNNs on different hardware platforms[1,2]. These simulators consider different technologies, platforms and periphery circuit modeling, while providing less consideration to interconnect. In this work, we develop an in-house simulator where a circuit level performance estimator of the computing fabric is combined with a cycle-accurate simulator for the interconnect. Figure 1 shows a block-level representation of the simulator. The inputs of the

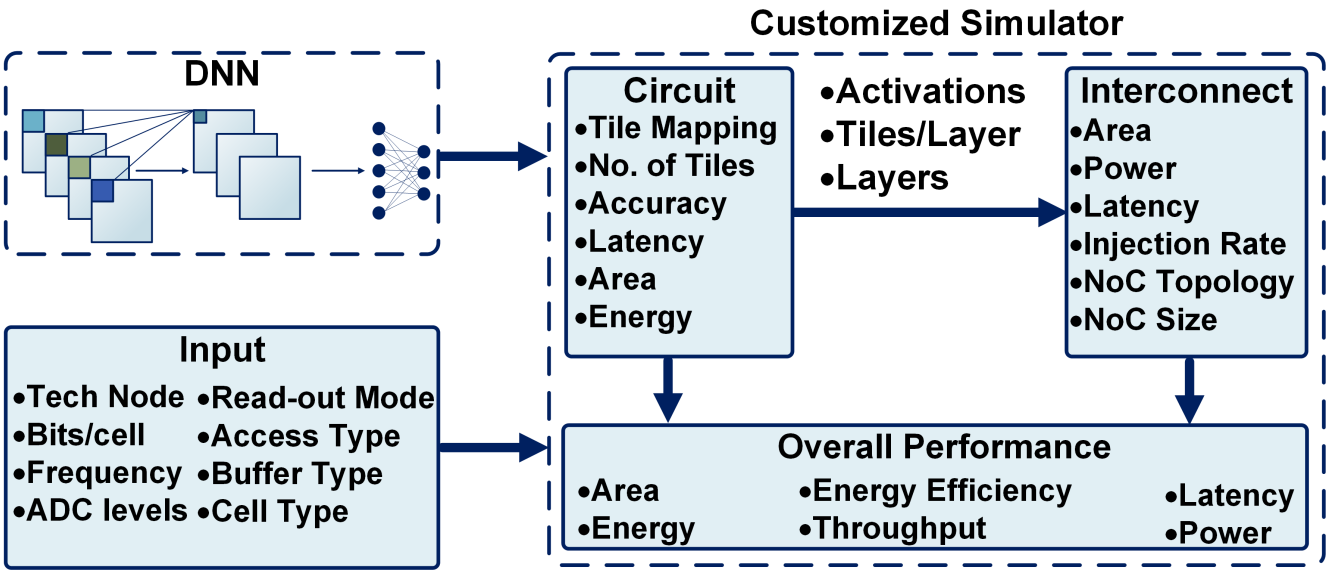

Figure 1. Block-level representation of the SIAM simulator.

simulator primarily include the DNN structure, technology node, and frequency of operation. The circuit part and interconnect part of the simulator are calibrated with NeuroSim[2] and BookSim[3] respectively. The simulator performs the mapping of the entire DNN to a multi-tiled IMC architecture by estimating the number of crossbar arrays and number of tiles per layer. The circuit simulator reports performance metrics, such as area, energy and latency of the computing logic. The interconnect performance is evaluated using the interconnect simulator. The circuit simulator outputs the number of tiles per layer, activations and number of layers. Then, we compute the injection rates for each source-destination pair in the multi-tiled architecture. The injection rate from every source to every destination is provided as input to the interconnect simulator. The interconnect simulator provides average latency and energy to complete all transactions in the DNN, and the area of the interconnect.

3.1 Instructions to Use the Tool

The following instructions will guide the user in using the tool.

- Train the network for best accuracy in any representation (FP 32, 16, 8) depending on the quantization of the weights and activations using Tensorflow.
- Create two arrays for both input activations and weights of the DNN and append them layer wise from beginning to end.
- Run the session to calculate all the values of activations and weights.
- Call the function hardware_estimation.py and pass the arguments into in the following order:
  array with activations, array with weights, precision of activations, precision of weights.
- Use the param.cpp file in SIAM folder to enter all the parameters for the design to be evaluated.
- Enter the network structure in the file Network.csv in the SIAM folder in the following order;
  input feature size, input feature size, number of input features, kernel size x, kernel size y, number of output features, if layer followed by pooling.
- Run make clean and make inside the SIAM folder with all the cpp files. This creates the binary for each of the files.
- Run the command below to provide the permission to 'booksim' binary. The user needs to need provide the permission one-time only.

```
$ chmod +x booksim
```

- Run the top file which invokes 'hardware_estimation.py'.
- The final results are available in the directory 'Final_results'.

4 Using SIAM

In order to use SIAM a set of parameters need to be set by the user under the param.cpp file in the SIAM folder. Each parameter is attached with a comment which describes the function it serves. The heterogeneous structure can be realized by using the row multiplier, col multiplier and to subtract parameters. Row multiplier depicts the number of number of rows of crossbar arrays in the tile and col multiplier depicts the number of columns of crossbar arrays in the tile. Based on these parameters the total number of crossbars in a tile is the product of both row and col multipliers.So as to enable more versatile tile structures we introduce a to subtract parameter which essentially subtracts crossbar arrays count from the tile. Hence the final number of crossbars in a tile is given by (1),

$$crossbars/tile = (rowmultiplier \times colmultiplier) - tosubtract \tag{1}$$

For multilevel cells, the parameter cellBit can be changed for different levels of the cell. Additionally, two mapping schemes are supported which are based on NeuroSim[2]. The mapping schemes are then optimized such that the utilization of the crossbar arrays is improved.

5 IMC Hardware Architecture

Area-Efficient Tile Architecture: The heterogeneous tile architecture consists of tiles having different number of CEs and PEs. Along with CEs and PEs, buffers and peripheral circuits also vary depending upon the size of the tile. Such a heterogeneous architecture improves PE array utilization, in-turn reducing area of the hardware. The tile architecture further includes non-linear activation units, I/O buffer and accumulators to manage data transfer efficiently. Each CE consists of the PE arrays, multiplexers, buffers, sense amplifier (SA) and flash ADCs. In addition, the architecture does not make use of a DAC. The proposed heterogeneous tile architecture can be used for both SRAM or ReRAM (1T1R) technologies. However, the peripheral circuits change according to the technology used.

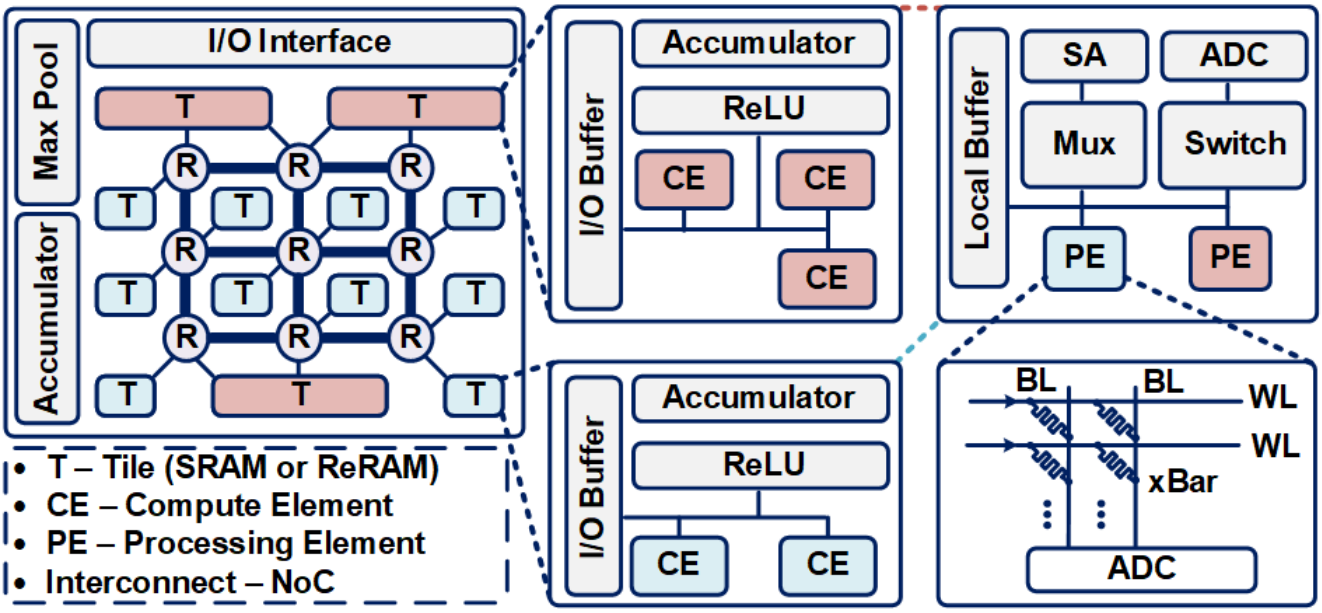

Figure 2. Heterogeneous in-memory architecture with optimized NoC.

Energy-Efficient NoC Structure: The architecture uses heterogeneous interconnects to maximize the energy-efficiency. It employs the NoC-based interconnect at the global tile level with a H-Tree interconnect at the CE level and bus interconnect at the PE level. Since the injection rate is much lower at the CE and PE level than that at the tile level, H-Tree and bus interconnect provide ample performance. At the tile level, the proposed energy-optimized mesh NoC with X–Y routing algorithm is used. We consider mesh NoC as the interconnect topology since mesh NoC is the state-of-the-art interconnect topology both in the realm of computer architecture[4] and DNN accelerators[5,6]. Based on our analysis, to achieve a balance between power and performance, the upper bound on the number of routers is set as 3 times the number of layers of the DNN. As a result of the optimization, we obtain the number of routers required for each layer of the DNNs. These routers are connected to the tiles of the corresponding layers. We also obtain the schedules for the activations from the tile to the NoC router. These schedules are implemented as a look-up table at each router of the NoC. This design choice also eliminates the overhead of router arbitration.

6 Future Work

6.1 Interconnect

- Right now, only mesh topology is used for the NoC. We plan to include the provision for user-defined NoC topology.
- In future, we plan to include user-defined parameters such as buffer size, number of virtual channels etc. for the NoC simulation.
- Through simulations, we also plan to guide the user about which NoC topology should be used for different DNN algorithms.

References

1. Dong, X. *et al.* Nvsim: A Circuit-level Performance, Energy, and Area Model for Emerging Nonvolatile Memory. *IEEE TCAD* **31**, 994–1007 (2012).

2. Chen, P.-Y. *et al.* NeuroSim: A Circuit-level Macro Model for Benchmarking Neuro-inspired Architectures in Online Learning. *IEEE TCAD* **37**, 3067–3080 (2018).

3. Jiang, N. *et al.* A Detailed and Flexible Cycle-Accurate Network-on-Chip Simulator. In *IEEE ISPASS*, 86–96 (2013).

4. Jeffers, J. *et al. Intel Xeon Phi Processor High Performance Programming: Knights Landing Edition* (Morgan Kaufmann, 2016).

5. Chen, Y.-H. *et al.* Eyeriss v2: A Flexible Accelerator for Emerging Deep Neural Networks on Mobile Devices. *IEEE JETCAS* (2019).

6. Shafiee, A. *et al.* ISAAC: A Convolutional Neural Network Accelerator with in-situ Analog Arithmetic in Crossbars. *ACM/IEEE ISCA* (2016).