Performance Comparisons Between Resistive and Static RAM for Machine Learning Models

Ramesh Sah, Hong-Ying Lin, and Oluwafemi Ajeigbe

Introduction

Machine learning systems are increasingly becoming an integral part of many domains.

Running and accelerating machine learning models on the chip is an important area of research.

Compute-in-memory (CIM) technologies provide one solution toward accelerating machine learning models.

In CIM paradigm, multiply-accumulate operations are done in-memory or near-memory to remove the memory bandwidth bottleneck.

Objectives

1. To study different CIM technologies such as SRAM and ReRAM and highlight their advantages and disadvantages.

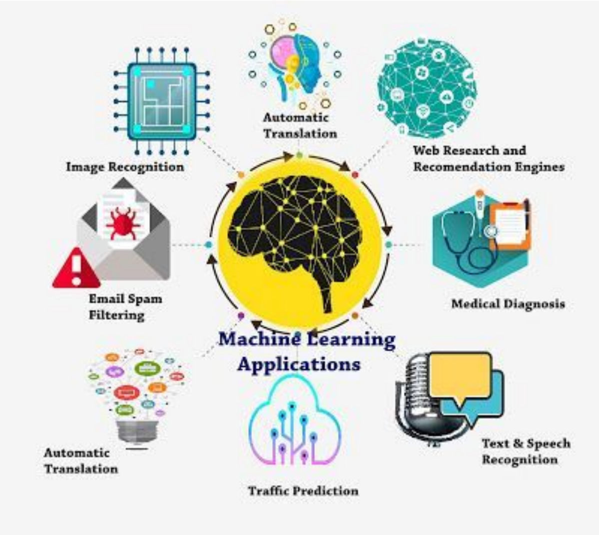
2. Compare the efficacies of SRAM and ReRAM for accelerating machine learning models from the design point-view.

3. Present analysis report with metrics such as latency, energy, and throughput.

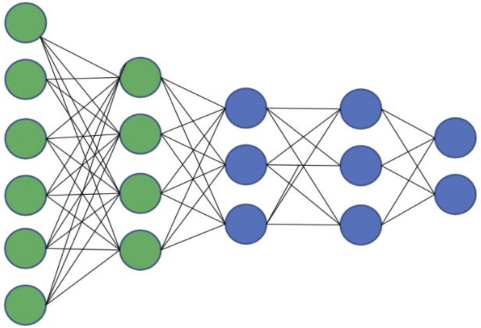
System Components

1. Machine Learning Algorithms
   a. **VGG8 and DenseNet40**

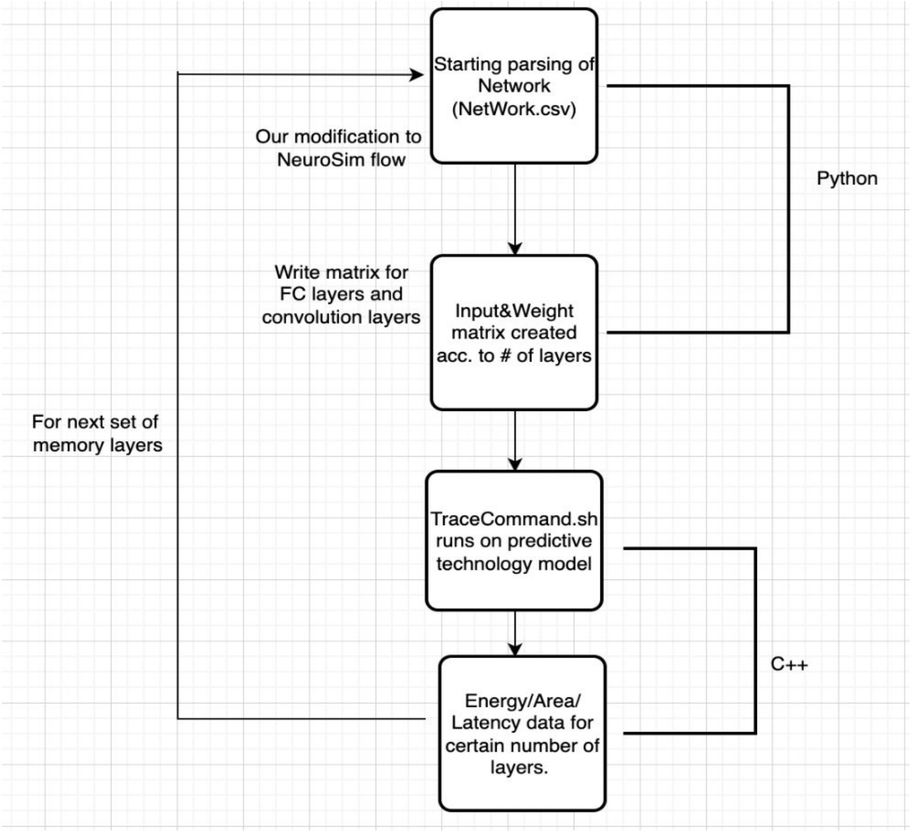
2. Neurosim : An integrated framework to emulate deep neural network inference performance on hardware accelerator based on near or in memory compute technologies.



Procedure



Neural Network



Starting parsing of Network (NetWork.csv)

Our modification to NeuroSim flow

Python

Write matrix for FC layers and convolution layers

Input&Weight matrix created acc. to # of layers

For next set of memory layers

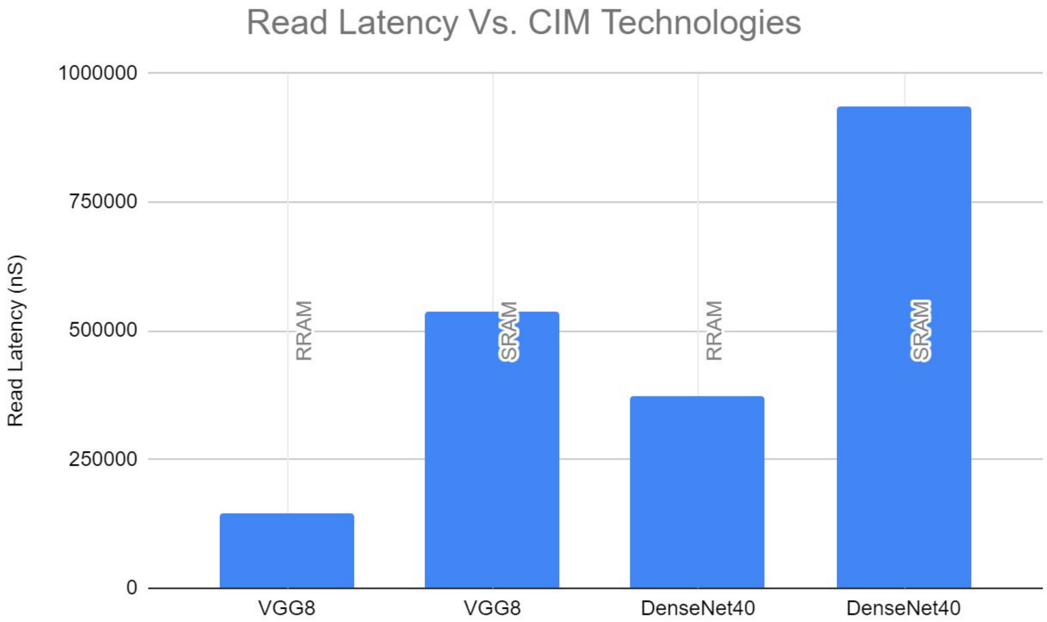
TraceCommand.sh runs on predictive technology model

C++

Energy/Area/ Latency data for certain number of layers.

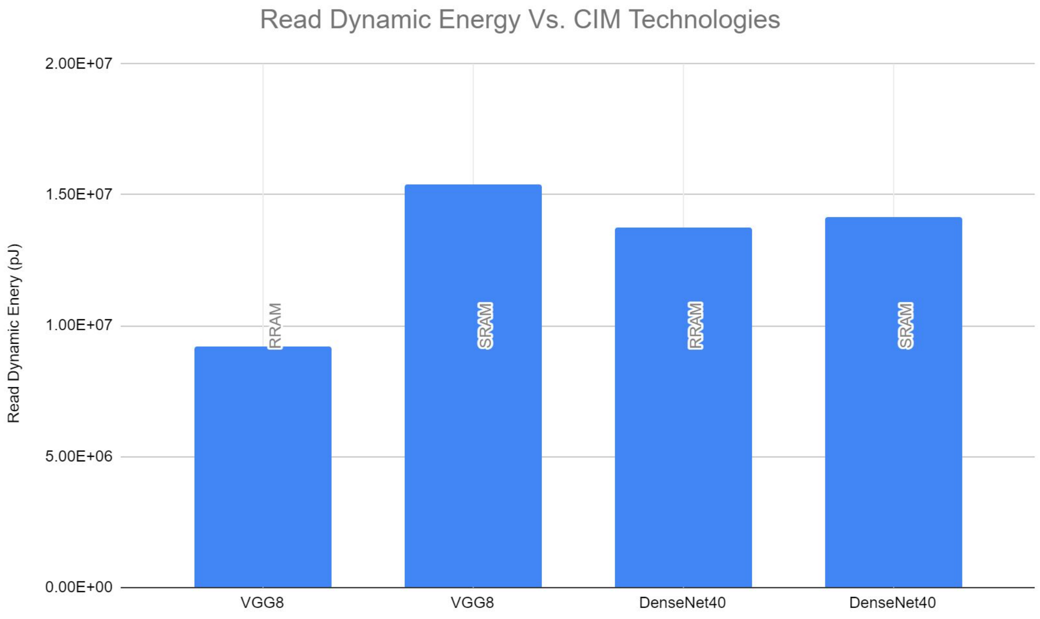
Neurosim Flow

Result 1



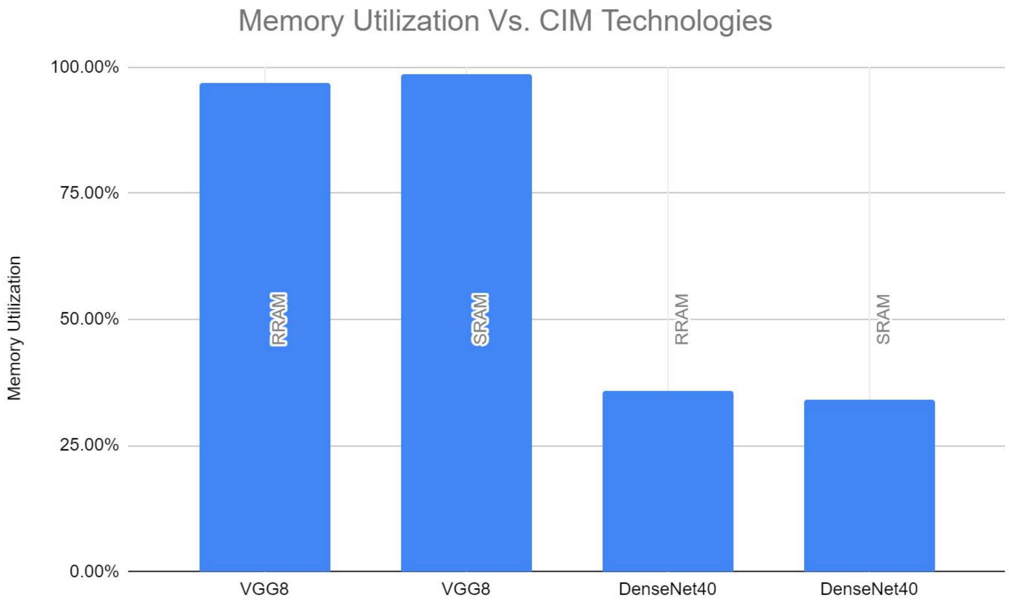
RRAM devices are better at read operations than SRAM.

Result 2

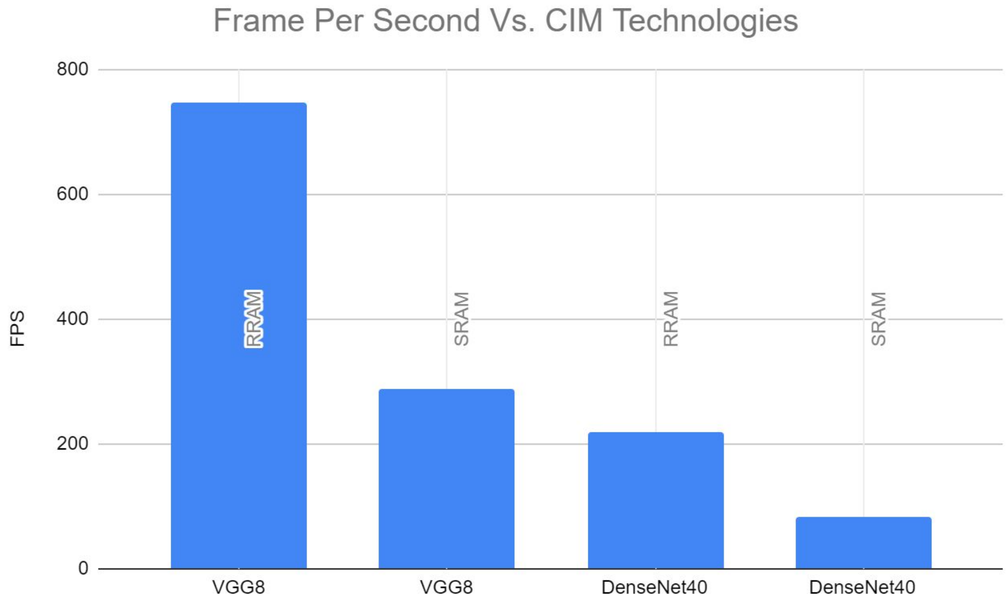


RRAM devices takes less energy for read operations than SRAM.

Result 3

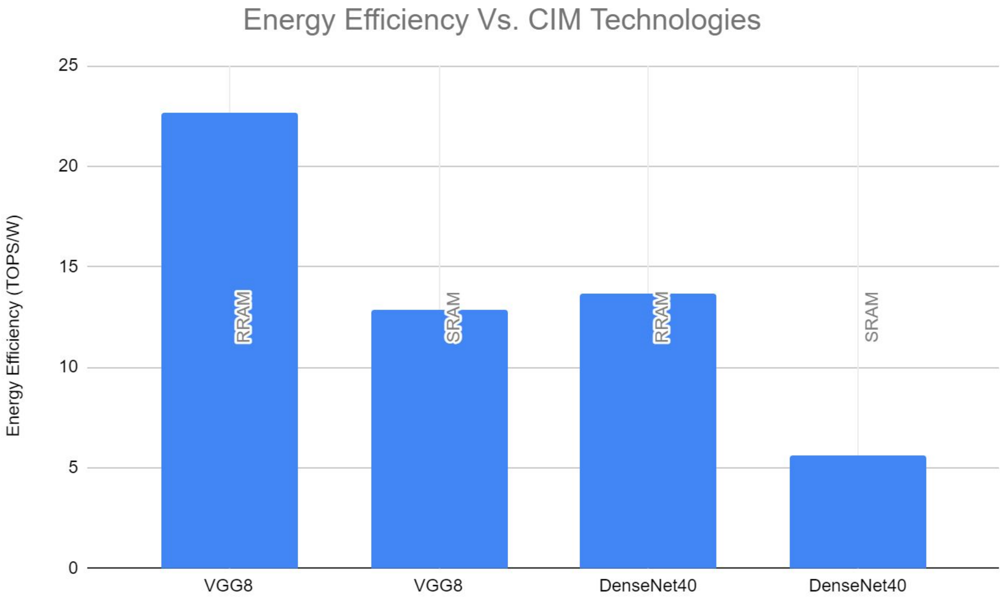

Memory Utilization Vs. CIM Technologies
Memory Utilization
100.00%
75.00%
50.00%
25.00%
0.00%
RRAM
SRAM
RRAM
SRAM
VGG8
VGG8
DenseNet40
DenseNet40

Result 4



RRAM devices are faster than SRAM.

Result 5



RRAM devices are more energy efficient than SRAM.

Conclusions & Future Work

- We analyzed the performance of SRAM and ReRAMS for accelerating computer vision models using NeuroSim with Pytorch.
- **ReRAMs had lower read latency than SRAMS.**
- **ReRAMs also consumed less dynamic energy compared to SRAMS.**
- Ultimately, this means f**aster frame per second** and **higher energy efficiency** for ReRAMS.
- In future, we will compare SRAMs and ReRAMS for write operations.

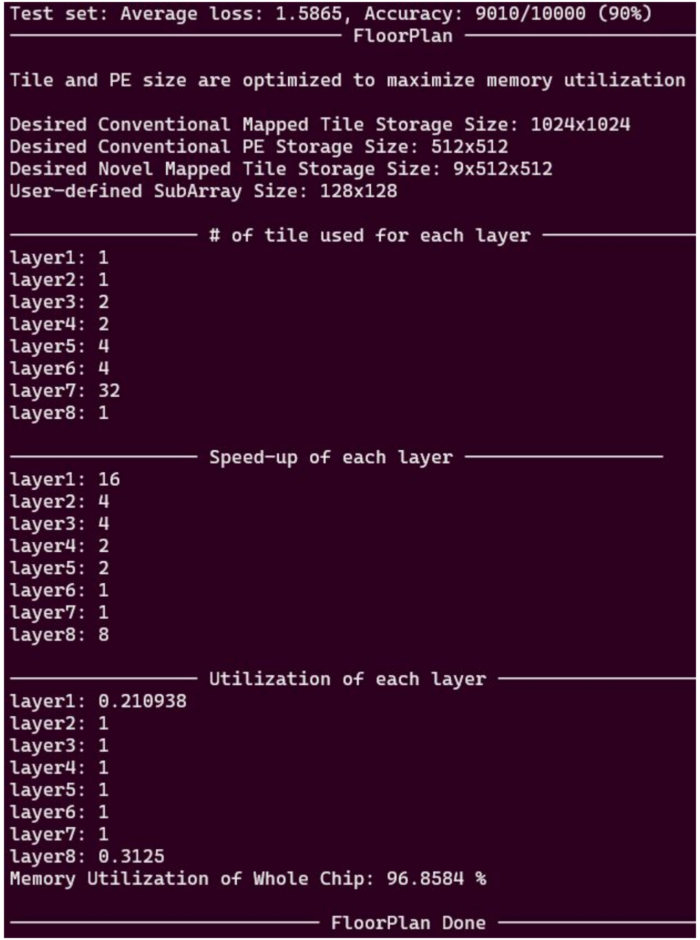
Challenges and Limitations

- Neurosim only works with Linux.
- Current version of **Neurosim only support** quantized and floating point **convolutional and fully connected layers**.
- Many state-of-the-art machine learning algorithms in computer vision, natural language processing, and graph learning utilizes custom layers not supported by Neurosim.
- We have not looked at the write performance measures. Consequently, the comparisons between SRAM and RRAM is not complete.

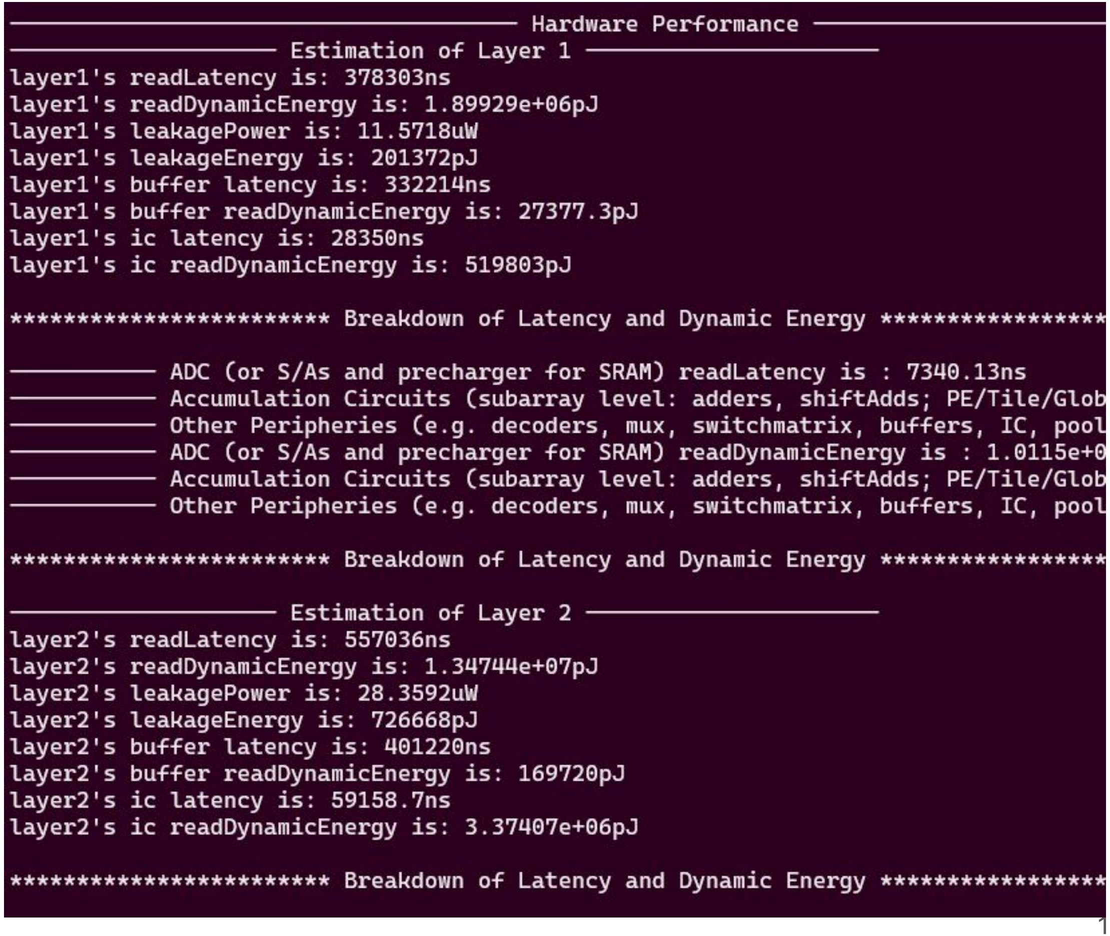
Thank You

Timeline

- Literature review (on-going)
- Setup Neurosim and preliminary testing (done)
- **Run experiments and obtain results (remaining)**
- **Prepare report and presentation (remaining)**



```
Test set: Average loss: 1.5865, Accuracy: 9010/10000 (90%)
────────────────────────────── FloorPlan ──────────────────────

Tile and PE size are optimized to maximize memory utilization

Desired Conventional Mapped Tile Storage Size: 1024x1024
Desired Conventional PE Storage Size: 512x512
Desired Novel Mapped Tile Storage Size: 9x512x512
User-defined SubArray Size: 128x128

───────────────── # of tile used for each layer ─────────────
layer1: 1
layer2: 1
layer3: 2
layer4: 2
layer5: 4
layer6: 4
layer7: 32
layer8: 1

───────────────── Speed-up of each layer ─────────────────
layer1: 16
layer2: 4
layer3: 4
layer4: 2
layer5: 2
layer6: 1
layer7: 1
layer8: 8

───────────────── Utilization of each layer ──────────────────
layer1: 0.210938
layer2: 1
layer3: 1
layer4: 1
layer5: 1
layer6: 1
layer7: 1
layer8: 0.3125
Memory Utilization of Whole Chip: 96.8584 %

──────────────────────────────── FloorPlan Done ──────────────────
```



```
─────────────────────────────────────────────── Hardware Performance ──────────────
────────────────────────── Estimation of Layer 1 ──────────────────────────
layer1's readLatency is: 378303ns
layer1's readDynamicEnergy is: 1.89929e+06pJ
layer1's leakagePower is: 11.5718uW
layer1's leakageEnergy is: 201372pJ
layer1's buffer latency is: 332214ns
layer1's buffer readDynamicEnergy is: 27377.3pJ
layer1's ic latency is: 28350ns
layer1's ic readDynamicEnergy is: 519803pJ

*********************** Breakdown of Latency and Dynamic Energy *****************

──────────── ADC (or S/As and precharger for SRAM) readLatency is : 7340.13ns
──────────── Accumulation Circuits (subarray level: adders, shiftAdds; PE/Tile/Glob
──────────── Other Peripheries (e.g. decoders, mux, switchmatrix, buffers, IC, pool
──────────── ADC (or S/As and precharger for SRAM) readDynamicEnergy is : 1.0115e+0
──────────── Accumulation Circuits (subarray level: adders, shiftAdds; PE/Tile/Glob
──────────── Other Peripheries (e.g. decoders, mux, switchmatrix, buffers, IC, pool

*********************** Breakdown of Latency and Dynamic Energy *****************

────────────────────────── Estimation of Layer 2 ──────────────────────────
layer2's readLatency is: 557036ns
layer2's readDynamicEnergy is: 1.34744e+07pJ
layer2's leakagePower is: 28.3592uW
layer2's leakageEnergy is: 726668pJ
layer2's buffer latency is: 401220ns
layer2's buffer readDynamicEnergy is: 169720pJ
layer2's ic latency is: 59158.7ns
layer2's ic readDynamicEnergy is: 3.37407e+06pJ

*********************** Breakdown of Latency and Dynamic Energy *****************
```