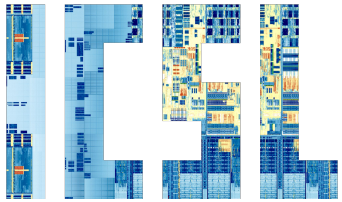


NPUsim: Full-System, Cycle-Accurate, Functional Simulations of Deep Neural Network Accelerators

Workshop on Modeling and Simulation of Systems and Applications (ModSim)

Oct. 7, 2021

Bogil Kim

bogilkim@yonsei.ac.kr

School of Electrical Engineering
Yonsei University

Necessity of NPU Simulator

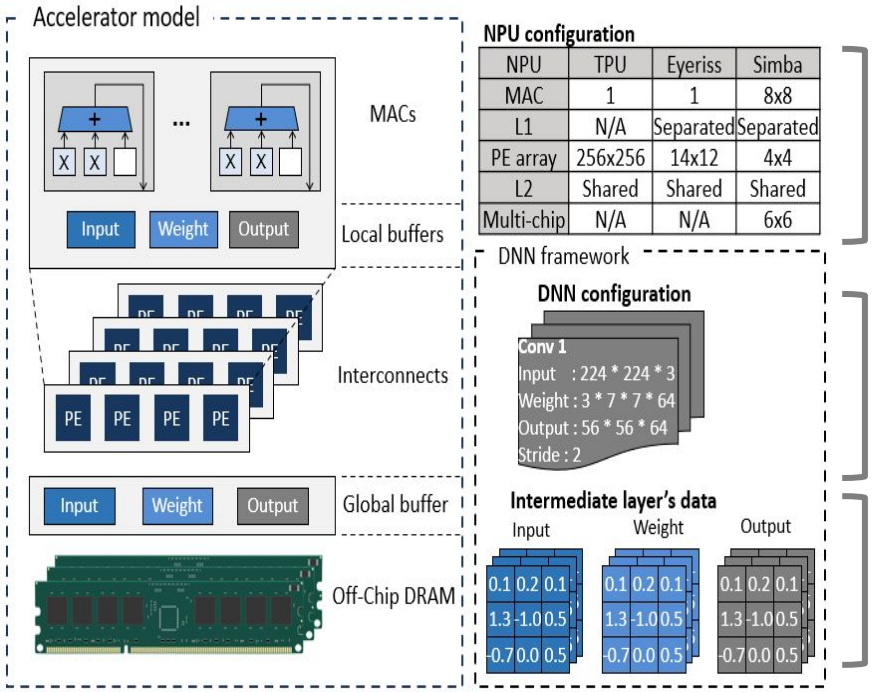
- The DNN accelerators had been focusing on their own specific designs.
- The comparisons were made in most cases against CPUs or GPUs.
- Existing simulators support only a small fraction DNN

- **Related work:** Simulation frameworks for DNN accelerators

Simulators	Features
SCALE-Sim	Simple calculator, a lot of constraints
SMAUG	Implemented on gem5, not flexible
STONNE	Implemented partial operation, fixed dataflow

The goal of NPUsim is to develop a full-system, cycle-accurate, and functional simulation framework of DNN accelerators

NPUsim overview



NPU configuration

NPU	TPU	Eyeriss	Simba
MAC	1	1	8x8
L1	N/A	Separated	Separated
PE array	256x256	14x12	4x4
L2	Shared	Shared	Shared
Multi-chip	N/A	N/A	6x6

NPUsim models a DNN accelerator based on the NPU configuration file.

Given network topology, schedule DNN execution depending on the dataflow.

Simulate DNN execution with the actual data from DNN software framework.

- **Supporting DNN accelerators**

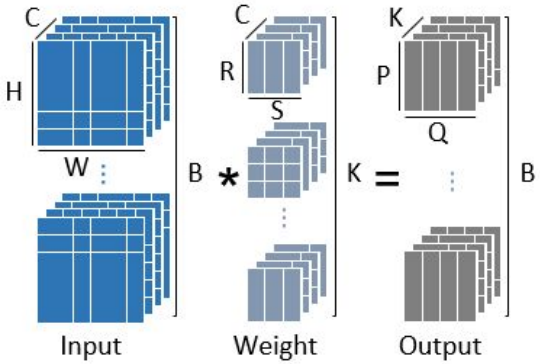
Single chip accelerators:
- DianNao
- Eyeriss
- TPU

Multi chip accelerators:
- DaDianNao
- Eyeriss v2
- Simba

Accelerators for Optimization techniques
- SCNN
- SparTen

Scheduling Methodology

- **The computation of a neural layer**




```
for(B)
 for(K)
  for(C)
   for(P)
    for(Q)
     for(R)
      for(S)
       output += input*Weight;
```

K	Output Channel
B	Input Batch
P	Output Height
Q	Output Width
C	Input Channel
R	Weight Height
S	Weight Width

- DNN computation can be expressed as loops with 7 parameters, and this concept can be applied to the DNN execution on accelerators.

- **Scheduling example: Weight stationary**

```
1  for(group(G))
2    for(output channel(K))
3      for(input channel(C))
4        for(filter height(R))
5          for(filter width(S))
6            load(weight, K, C, R, S);
7            for(batch size(B))
8              for(output height(P))
9                for(output width(Q)) {
10                 load(output,  B, K, P, Q);
11                 load(input,   B, C, H, W);
12                 store(output, B, K, P, Q);
```

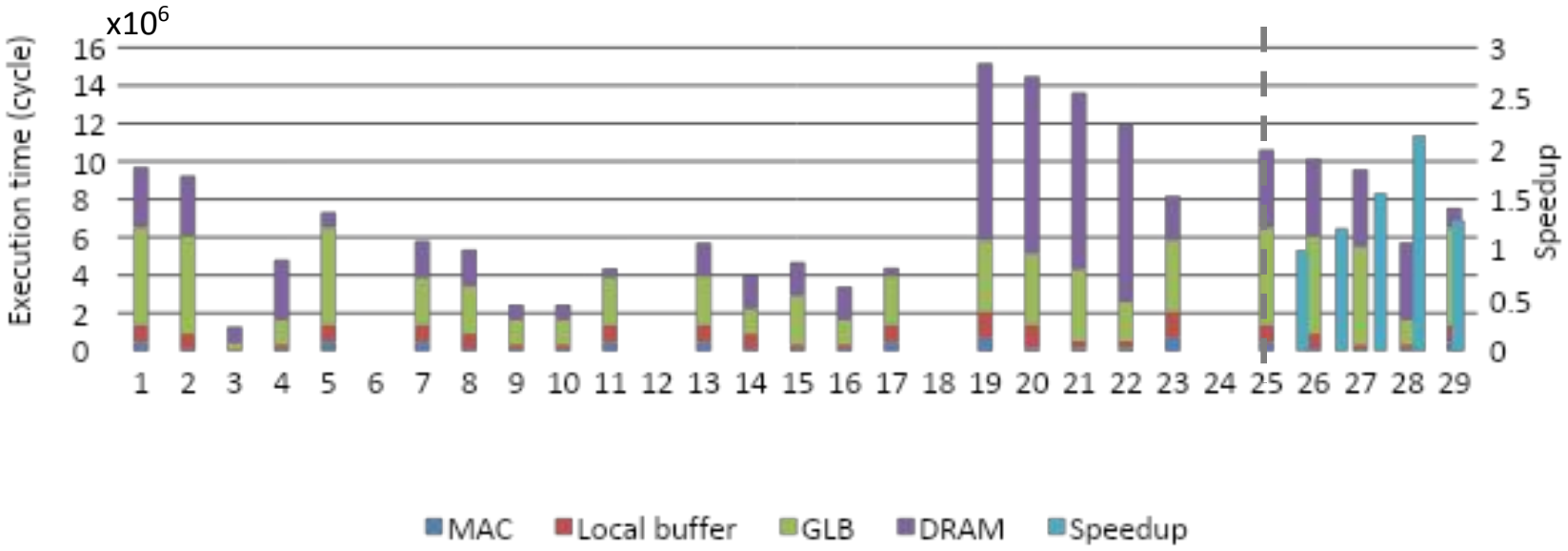
Place weight parameters at outer loops and put an operation for weight inside of outer loops.

Place input and output parameters at inner loops and put operations for other data innermost of loops.

- Weight is reused as many times as the number of iterations of inner loops.
- NPUsim stores a series of operations in order at the scheduler.

Case study

- **The effect of scaling-up/scaling-out components of Eyeriss on execution time**



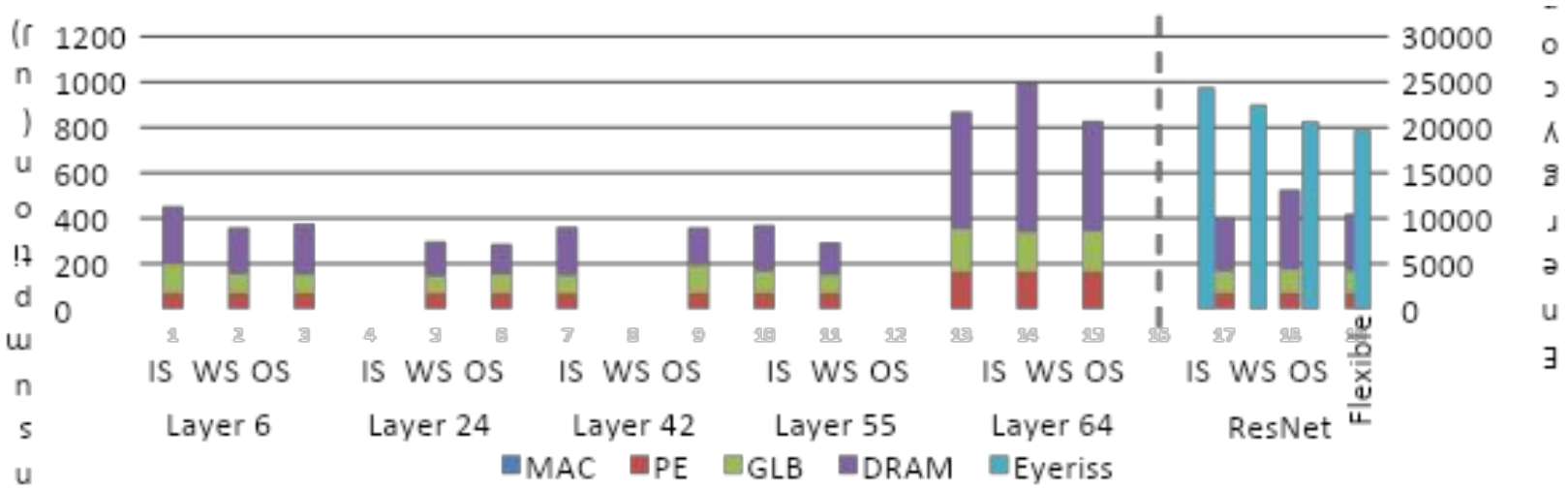
- Each component indicates that the specification is increased by 4 times than baseline accelerator (base).

  ex) BW: Increase DRAM bandwidth by 4 times.

- The relative execution time for each component affects the efficacy of scaling-up/scaling-out the component.

Case study

- **Energy consumption depending on the various dataflow**



- The best dataflow varies depending on the relative data size of each layer.
- Flexible dataflow reduces energy consumption by 20% compared to fixed dataflow when executing ResNet-50.

Conclusion

- NPUsim features for full-system modeling, cycle-accurate simulation, and capability of functional simulation.
- NPUsim enables in-depth analysis of DNN execution on accelerators.
- The source code will be available at https://github.com/yonsei-icsl/npusim.