Lab Assignment #3 – DNN Accelerator Modeling and Design Space Exploration

Goal

This lab focuses on modelling the performance of DNN accelerators and exploring the design space of systolic array-based accelerators for DNN workloads. For the first part, you will use SCALE-Sim, an open-source accelerator simulator. You will be responsible for sizing the memory and compute of the said accelerator subjected to design constraints such that we can get maximum processing with optimum/minimum memory and compute resources. In the last part, you will design your own simple DNN accelerator simulator in Python. We provide a fix set of accelerator architecture and workload specification as inputs, and your job is to write a simulator for that DNN accelerator that can generate the number of execution cycles for the given workload.

Part 1: Design Space Exploration with Scale-Sim

Setup starter code and environment

In this lab, we will be using SCALE-Sim, which is an open-source simulator for systolic array-based DNN accelerators. It takes in two inputs from the user – first is the workload description and second is the description of the accelerator's architecture and mapping. It generates multiple metrics including the cycles required to run the specified workload on the specified accelerator.

First, clone the SCALE-Sim simulator from this repo: scalesim-project/scale-sim-v2: Repository to host and maintain scale-sim-v2 code

Second, download "Lab3.zip" from Canvas. "Lab3.zip" contains the files listed below.

1. dnn_layers.csv (Workload file)

You can run this lab on Sol or on your personal machine. We've verified the execution on Sol and can offer support if you encounter any issues while running there.

Understanding and Running SCALE-Sim

Make sure you've read the SCALE-Sim paper: arxiv.org/pdf/1811.02883

Go through the README.md file of the SCALE-Sim-v2 repository. The documentation explains the inputs, output files and logs, and the internal of the tool. The following paragraphs give brief descriptions to get you started with running the simulator.

Inputs: SCALE-Sim expects two input files from the user. First is the config file, which captures the architectural configuration of the systolic array-based accelerator to be simulated. Here is what each parameter means:

ArrayHeight:	Number of rows of the systolic array
ArrayWidth:	Number of columns of the systolic array
IfmapSramSzkB:	The size of the SRAM buffer which holds input matrix in kB
FilterSramSzkB:	The size of the SRAM buffer which holds weight matrix in kB
OfmapSramSzkB:	The size of the SRAM buffer which holds generated output matrix in kB
IfmapOffset:	The base address offset for elements in the input matrix
FilterOffset:	The base address offset for elements in the weight matrices
OfmapOffset:	The base address offset for elements in the generated output matrix
Dataflow:	Indicating one of the Output, Weight, or Input stationary dataflows
Bandwidth:	User specified limit for BW (Elements/cycle)
InterfaceBandwith:	Flag to either use the BW constrain specified above or ignore the constrain and calculate the max bandwidth

You can look at sample config files in the "`configs/`" folder inside SCALE-Sim-v2 repo (`google.cfg` captures Google TPU's configuration).

The second input is a CSV file capturing the layer-wise parameters of a neural network. SCALE-Sim-v2 processes this file line-by-line, thus simulating each layer sequentially. The parameters captured in this file are input dimensions, filter dimensions, number of filters, number of channels, strides, etc. You can look at example topology files in "`topologies/`" folder inside the SCALE-Sim-v2 repo.

Running SCALE-Sim: You will need python3 to run SCALE-Sim-v2. Running SCALE-Sim is fairly easy, as it's just one command:

```
python3 <scale_sim_repo_root>/scalesim/scale.py -c
<path_to_config_file> -t <path_to_layer_file>
```

```
(my_env) (base) [kmhatre@sc024:~/lab_3/scale-sim-v2]$ python
scalesim/scale.py -c configs/my_config.cfg -t ../layers_lab3.csv
====================================================
****************** SCALE SIM *********************
====================================================
Array Size:     256x256
SRAM IFMAP (kB):        2048
SRAM Filter (kB):       2048
SRAM OFMAP (kB):        1024
Dataflow:       Weight Stationary
CSV file path:  ../layers_lab3.csv
Bandwidth:      20
```

```
Working in USE USER BANDWIDTH mode.
====================================================

Running Layer 0
100%|████████████████████| 7582/7582 [00:03<00:00, 2314.63it/s]
Compute cycles: 7581
Stall cycles: 0
Overall utilization: 21.22%
Mapping efficiency: 26.59%
Average IFMAP DRAM BW: 20.000 words/cycle
Average Filter DRAM BW: 20.000 words/cycle
Average OFMAP DRAM BW: 160.353 words/cycle

Running Layer 1
100%|██████████████████| 12418/12418 [00:03<00:00, 3145.87it/s]
Compute cycles: 12417
Stall cycles: 0
Overall utilization: 13.16%
Mapping efficiency: 96.43%
Average IFMAP DRAM BW: 20.000 words/cycle
Average Filter DRAM BW: 20.000 words/cycle
Average OFMAP DRAM BW: 19.999 words/cycle
************ SCALE SIM Run Complete ****************
```

Outputs: Other than the messages on `stdout`, SCALE-Sim generates various detailed outputs in the different files stored under the directory "`test_runs/<run_name>`". The files in this directory are described below:

> **COMPUTE_REPORT.csv:** Provides layer wise cycle count for computing and average utilization in %.
> **BANDWIDTH_REPORT.csv:** Provides layer wise average bandwidth for SRAM reads/writes, and DRAM reads/writes.
> **DETAILED_ACCESS_REPORT.csv:** Captures various detailed metrics like number of bytes read/written for SRAM or DRAM transactions, start time and stop time in cycles for these transactions etc.

Design Space Exploration using SCALE-Sim

In this lab, you are tasked with designing **two** efficient systolic array-based accelerators, both running **weight stationary dataflow** at **1GHz**. First is a data-center class accelerator capable of providing a peak throughput of 131 TeraOps/sec (131.072 TeraOps/sec to be precise), and the second is a mobile class accelerator with a peak throughput rated at 2 TeraOps/sec (2.048 TeraOps/sec to be precise). Please treat each Multiply-and-Accumulate (MAC) as 2 operations. Workloads are CNN layers described in `dnn_layers.csv` file.

Your job is to propose accelerator architecture configurations suited for the specified workloads subject to the following constraints:

1) Each of the maximum DRAM bandwidth should not be more than 20

bytes/cycle (i.e., IFMAP DRAM bandwidth < 20 bytes/cycle, Filter DRAM bandwidth < 20 bytes/cycle, OFMAP DRAM bandwidth < 20 bytes/cycle.)

2) Total on-chip memory (i.e., sum of all the three buffer sizes – IFMAP, Filter, OFMAP) should not exceed 1MB

You should evaluate the effectiveness of the accelerator architecture configurations using the following cost:

$$Cost = \sum_{layers} \{cycles * (avgDRAMIfmapBW + avgDRAMFilterBW + avgDRAMOfmapBW\}$$

You need to perform design space exploration of the accelerator with the goal of minimizing this cost, which in turn minimizes the cycle count and the total average DRAM bandwidth. **You will need to explore the design space by varying:**

- The size and shape of the systolic array
- The size of on-chip memories (IFMAP, Filter, OFMAP)

Essentially, you will invoke Scale-Sim in a loop with different architectural parameters or configurations related to systolic array and on-chip memory sizes and then analyze/evaluate the cost for each architecture configuration to find the one with the minimum cost.

What to submit? [50 points]

- **A PDF report**

Your report should include the following details:

1. Methodology for selecting the on-chip memory size to explore [5 points]
2. Methodology for selecting the systolic array size and shape to explore [5 points]
3. How you explored the design space (useful experiments you performed) [5 points]
4. Insights you gathered from the experiments [5 points]
5. Charts from design space exploration (one for datacenter class accelerator and one for edge class accelerator) [10 points]
6. Final results of the design space exploration (architecture parameters that satisfy the requirements and the minimum value of cost) (one for datacenter class accelerator and one for edge class accelerator) [10 points]

- **CSV file for cost (10 points)**

  Your submission should also contain a CSV file called cost.csv that includes all the explored configurations in the following format:

```
DataCenter_Conv1, <size of sys array>, <shape of sys array>, <ifmap
sram size kb>, <filter sram size kb>, <ofmap sram size kb>,
<cycles>, <avg dram ifmap bw>, <avg dram filter bw>, <avg dram ofmap
bw>, cost
……
```

```
DataCenter_Conv2, <size of sys array>, <shape of sys array>, <ifmap
sram size kb>, <filter sram size kb>, <ofmap sram size kb>,
<cycles>, <avg dram ifmap bw>, <avg dram filter bw>, <avg dram ofmap
bw>, cost
……
Mobile_Conv1, <size of sys array>, <shape of sys array>, <ifmap sram
size kb>, <filter sram size kb>, <ofmap sram size kb>, <cycles>,
<avg dram ifmap bw>, <avg dram filter bw>, <avg dram ofmap bw>, cost
……
Mobile_Conv2, <size of sys array>, <shape of sys array>, <ifmap sram
size kb>, <filter sram size kb>, <ofmap sram size kb>, <cycles>,
<avg dram ifmap bw>, <avg dram filter bw>, <avg dram ofmap bw>, cost
……
```

Extra Credit [10 pts]

1. Find the maximum total on-chip memory size that still satisfies the DRAM bandwidth constraint (max 20 bytes per cycle for each type of data). What is the value of cost for this architecture? (3 points)
2. Find the minimum total on-chip memory size that still satisfies the DRAM bandwidth constraint (max 20 bytes per cycle for each type of data). What is the value of cost for this architecture? (3 points)

In the report, explain how you found the maximum and minimum total on-chip memory sizes. Provide the costs for each scenario (datacenter and edge) with minimum and maximum on-chip memory sizes. (4 points)

Part 2: Create your own DNN performance simulator

Note: This part is totally independent from Scale-Sim.

In this part, you will design a DNN performance simulator that generates the number of cycles required to execute a given DL workload on a given accelerator. We've provided the architecture of the accelerator and the specifications of the DL workload. Your task is to design a simulator in Python that, given the architecture specification, tiles the workload optimally to maximize hardware (compute units and on-chip memory) utilization. Our primary objective is to help you grasp the concept of tiling and perform calculations to estimate execution cycles.

Architecture specifications

This is a very simple DNN accelerator. There are 8 matrix units on this accelerator. Each matrix unit can perform a matrix multiplication of 8x8x8 (m=8, k=8, n=8) in 16 cycles. Do not worry about what style this unit is – systolic based, dot product based, vector/SIMD based, etc. Each matrix unit has local memory (or buffers) to store an 8x8 input matrix, an 8x8 weight matrix, and an 8x8 output matrix. Assume that all 8 matrix units can execute in parallel.

There is a shared on-chip memory (SRAM) of 2 MB on the accelerator. All the matrix units have full access to this on-chip memory. Do not worry about whether this memory has enough ports or not. Assume that it can feed all 8 matrix units if needed at the same time. The on-chip memory bandwidth is 192 bytes per cycle (across all matrix units). Read bandwidth is 128 bytes per cycle and write bandwidth is 64 bytes per cycle.

The accelerator is connected to an off-chip memory (DRAM). Assume that its bandwidth is 6 bytes per cycle. Read bandwidth is 4 bytes per cycle and write bandwidth is 2 bytes per cycle.

This accelerator is only designed for int8 operations. All elements take 1 byte to store. Processing elements inside the matrix units also operate in int8 precision. The accumulators are also int8 in size.

Summary of accelerator specification:

- Matrix unit (m=8, k=8, n=8): 16 cycles
- Number of matrix units: 8
- SRAM size: 2 MB
- SRAM bandwidth:
    - Read: 128 Bytes/cycle
    - Write: 64 Bytes/cycle
- DRAM bandwidth:
    - Read: 4 Bytes/cycle

- Write: 2 Bytes/cycle

Workload specifications

There are two workloads that we want to execute on this accelerator. Their specifications are:

Workload 1: M = 4096, K = 4096, N = 4096

Workload 2: M = 512, K = 512, N = 512

Input matrix is MxK, Weight matrix is KxN, Output matrix is MxN. The precision of the inputs, weights, and outputs of the workload is int8.

Other specifications

Assume that reducing partial results across tiles is free (no cycles are consumed in moving partial results of a tile from one matrix unit to another, and no cycles are consumed in reducing the partial results).

Assume that the accelerator has complete overlap of data loads from DRAM to SRAM, from SRAM to matrix units, and the computation in matrix units. That is, to calculate the total number of cycles to execute a workload, you can use max(compute cycles, SRAM read cycles, SRAM write cycles, DRAM read cycles, DRAM write cycles). This also means that the SRAM can be double buffered.

SRAM read refers to the data read by the matrix units from SRAM. SRAM write refers to the data written by the matrix units to the SRAM. DRAM read refers to the data read by the SRAM from DRAM. DRAM write refers to the data written by the SRAM to the DRAM.

In your tiling scheme, you should not assume any reuse of data across tiles at the SRAM-matrix unit level. Data read from SRAM into the matrix unit will be used for one 8x8x8 computation only. If the same tile of data is required for another 8x8x8 computation, it will be read again by the matrix units from SRAM. *This is not optimal, but this will help us ensure that you can get the same results as us and hence makes grading easier.*

The tile size at the DRAM to SRAM level will be such that it maximizes the use of SRAM. There are multiple tiles sizes possible that will maximize the use of SRAM. You just need to choose one.

Tile size at the SRAM to matrix unit level will be 8x8x8.

Your simulator

Your simulator will be a Python program that takes the architecture and workload specifications as inputs, and outputs the **total cycles** required to execute the workload, **the cycles required for data transfer**, and the **cycles required for computation**. It will also output the **total bytes transferred at each level** (DRAM to

SRAM, and SRAM to Matrix Unit). You are allowed to hardcode the architecture specifications of the accelerator inside your program (e.g., you can declare variables on top of the file for these architecture specifications), but you should take workload specifications (M, N, K) from the command line. These should not be hardcoded inside the simulator.

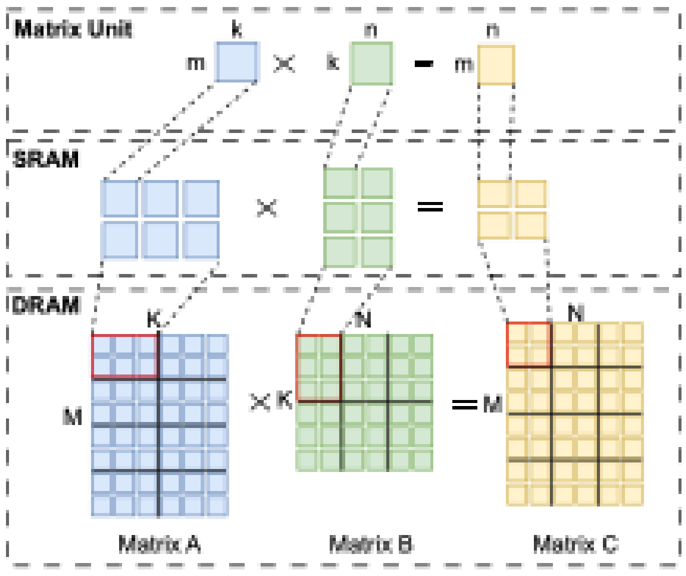
Summary of required outputs (these should be printed on the screen/stdout):

1. Total execution cycles
2. Total cycles consumed by compute (matrix units)
3. Total cycles consumed for data transfer from DRAM to SRAM
4. Total cycles consumed for data transfer from SRAM to matrix units
5. Total bytes transferred from DRAM to SRAM (read) and SRAM to DRAM (write)
6. Total bytes transferred from SRAM to matrix unit (read) and matrix unit to SRAM (write)

Suggestions and guidance

Here are some general suggestions or guidance that will help you design this simulator. Feel free to approach the problem in any way you see fit.

Carefully understand the size of workload and the size of the accelerator. Think about how you can divide the workload in small chunks (tiles) such that it can fit on the matrix unit and utilize it completely. Then think about the size of the available SRAM. What chunk (tile) of the input, weight and output can you hold in the SRAM at any given time. Based on this you can get a certain amount of data from the DRAM and store it in SRAM. From the SRAM, you can chunk (tile) it and bring it to the matrix unit buffers. This 2-level tiling looks like the following figure:



What to submit? [50 points]

- **A PDF report (Combine this report with the Part1 report and submit only 1 report)**

Your report should include the following details:

1. Methodology for designing the accelerator. This should contain a brief explanation of tiling and should include equations/expressions used to calculate the compute cycles and the data transfer cycles. [12.5 points]
2. Provide the following for this accelerator: [5 points]
    a. Peak compute throughput (in MACs per cycle)
3. Provide the following for each workload: [2 * 2 * 2.5 points]
    a. Tile size used for SRAM to matrix unit transfers. Number of tiles created at this level.
    b. Tile size used for DRAM to SRAM transfers. Number of tiles created at this level.
4. Snapshot of your simulator's output when run for each workload [10 points]

- **Python Code [12.5 points]**
  Submit your python code with proper comments.