CSE530 Project

CSE 530, Fall 2023

Note:

- Please include your name and PSUID on the first page.
- Submit all files on Canvas.
- The assignment must be submitted on Canvas before the due date (midnight).
- No single line answers are accepted in the submission.
- Refer to the syllabus for late submission policies.
- No kind of collaboration is allowed unless specifically mentioned in the assignment.
- All source materials must be cited. The University Academic Code of Conduct will be strictly enforced.
- All queries related to Assignment should have a subject line **CSE530: Project**
- Please use CANVAS MESSAGES for any queries. Make sure to send your queries to both the TAs for a faster response.

Goal:

Experiment and investigate systolic arrays for accelerating neural networks.

Tools/Software packages required:

- scale-sim-v2[1]
- Conda[2]

Installing and Running Scale-Simv2

You can use your own systems for installing Scale-Simv2 and running the experiments. Note that the Installation requires the conda environment, and the instructions provided below assume that conda is installed

1) conda create -n CSE530_FA21

2) conda activate CSE530_FA21

3) git clone https :// github . com / scalesim - project / scale - sim - v2 . git

4) pip3 install -r < scale_sim_repo_root >/ requirements . txt

You can also refer to the demo Google Colab notebook from ScaleSim github repository.

Running on W135 machines:

We have a pre-built conda environment with scale-simv2 installed.

- Please use the "base" conda environment in "/home/other/CSE530-FA2022/anaconda3". You can activate it by "source /home/other/CSE530-FA2022/condarc".
- You are provided limited space under the "home/grads/<user_id>" directory. We recommend you do not generate the traces for any of your experiments. You can disable traces by passing "save_disk_space" flag as "True" when instantiating the scalesim object.
- NOTE: the base conda environment has the bare minimum packages to run scale sim. If you want to change the scale sim source code or post-process your results, use your own machines. If you feel adding a package might benefit others, please send a canvas message to the TAs requesting package installation, clearly stating your rationale.
- If you are using W135 machines, start your experiments early since you might run out of available RAM memory nearing the deadline because of many users.

Detailed steps:

```
source /home/other/CSE530-FA2022/condarc

mkdir ~/cse530_project

cp -R  /home/other/CSE530-FA2022/scale-sim-v2/configs/  ~/cse530_project/

cp -R /home/other/CSE530-FA2022/scale-sim-v2/topologies/  ~/cse530_project/

cp /home/other/CSE530-FA2022/scale-sim-v2/code-examples/run_scale.py  ~/cse530_project/

cd ~/cse530_project

python run_scale.py
```

Background – Systolic Array

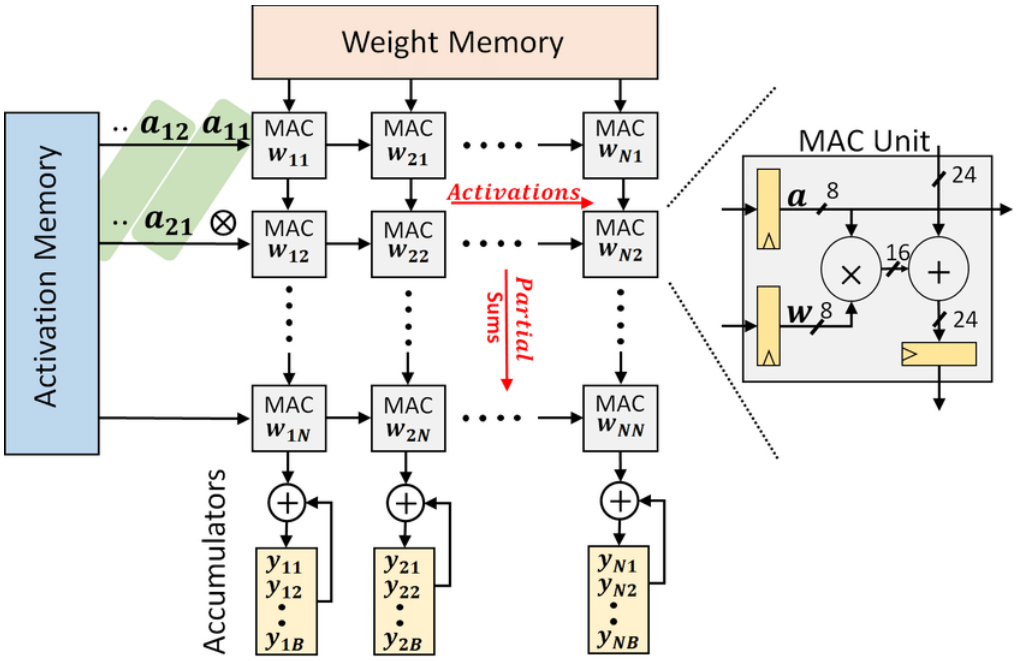

Figure1. Systolic Array [6]

Systolic array consists of a network of multiply and accumulate(MAC) processing units(as shown in Figure). In Figure1, Activation and Weight Memory serve as Inputs which are fed to the MAC units at the edges of the systolic array. Each of the MAC units will accept data from its upstream neighbors. For example, in the case of Figure 1, MAC unit w_{22} accepts inputs from MAC units w_{21} and w_{12} (both the inputs in this case being the outputs of MAC units w_{21} and w_{12}). Each MAC unit will use this data for computing the partial result and will subsequently store the result and will subsequently pass it on to its downstream neighbors. In Figure 1, MAC unit w_{22} passes on its results to MAC units w_{23} and w_{32}. Once the input has been completely fed, the results will be available in the accumulators at the the bottom of the array. They are useful for performing convolution, correlation, matrix multiplication etc. See [8],[9] and [1] for further details regarding the working of systolic arrays. You are encouraged to go through these papers to develop an understanding of the hardware so that you can better analyze the results of your experiments.

Background – Neural Networks, Transformers

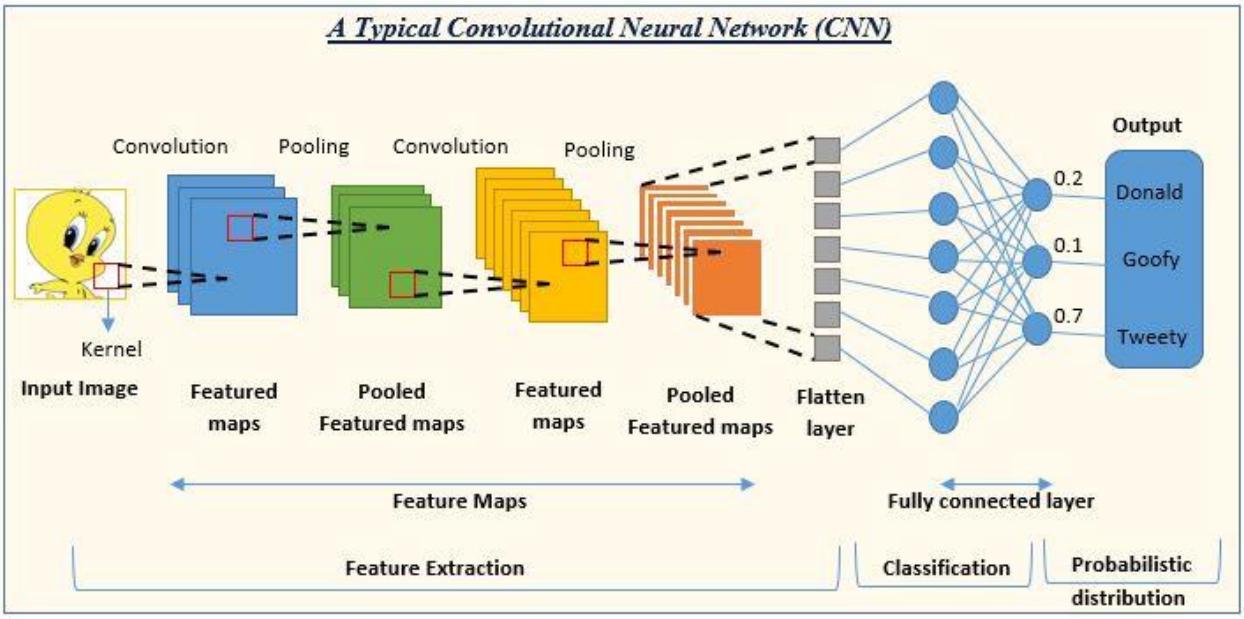

Figure 2 Convolutional Neural network

A convolutional neural network [5] is a type of neural network used for image processing which consists of multiple layers which perform convolution and pooling as well as a fully connected layer as shown in Figure 2. The network accepts an input as an image and provides an outputs with a list of probabilities. Each convolution layer will perform convolution on input provided by the previous layer or the user and the filter/weight associated with the layers to generate an output which serves as input to the next stage. These layers are an ideal workload for exploiting the MAC capabilities of systolic arrays.

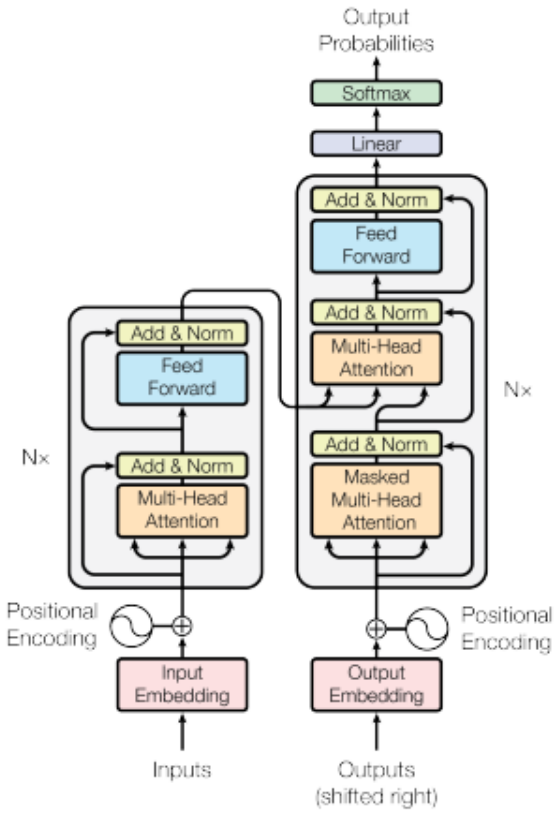

Figure 3 Transformers

Figure 3 shows the architecture of Transformers. It consists of multiple stacks of multi-head attention as well as feedforward layers which processes the input embedding to generate the output. Though this architecture was developed initially for natural language processing, it is finding application in a wide range of fields and forms the basis for many large language models (LLMs). The individual layers can be expressed in terms of matrix multiplication operations and therefore can benefit from systolic arrays. Please read [7] for further details.

Scale-Sim

Acknowledging the utility of using systolic arrays for acceleration of AI workloads such as convolutional neural networks as well as transformers the broader research community has identified the need for cycle accurate, configurable systolic array simulator. Scale-Sim is the result of this effort.

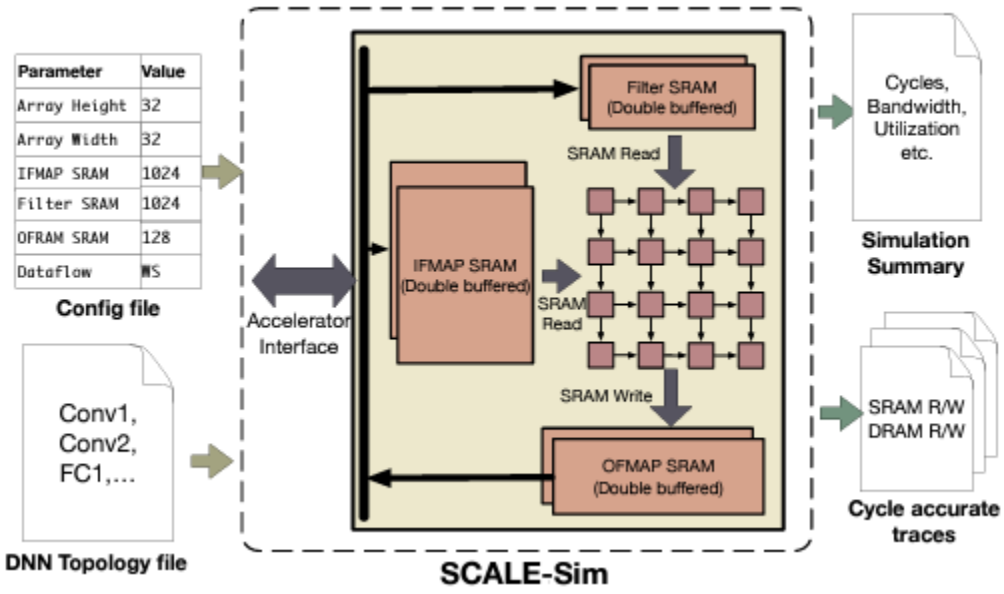

Figure 4 Scale-Sim architecture

Figure 4 depicts the Scale-Sim architecture. The systolic array accelerator, as shown in Figure 5, is assumed to be sitting on the system interconnect separate from the Main processor, and in effect functions as a kind of coprocessor. The processor will offload work to the systolic array which fetches the data from the memory, performs the required operations, stores the results in the main memory, and informs the main processor that it has finished the job via an interrupt. The systolic array is also equipped with buffers for storing the two inputs, as well as a buffer for storing the output to reduce the number of accesses to the main memory.

Scale-Sim accepts two inputs

1) Topology file associated with the convolution neural network or the transformer which is a layer-by-layer description of the convolution neural network or the transformer
2) The config file which describes the hardware that is used for acceleration – the height, width etc.

Scale-Sim generates the output which provides information regarding the number of cycles it took perform the execution, how well it utilized the hardware, bandwidth information etc. Version 1 of Scale-Sim provided some SRAM, as well as DRAM information which is no longer provided in version 2. You are encouraged to read [1] for more details regarding Scale-Sim.

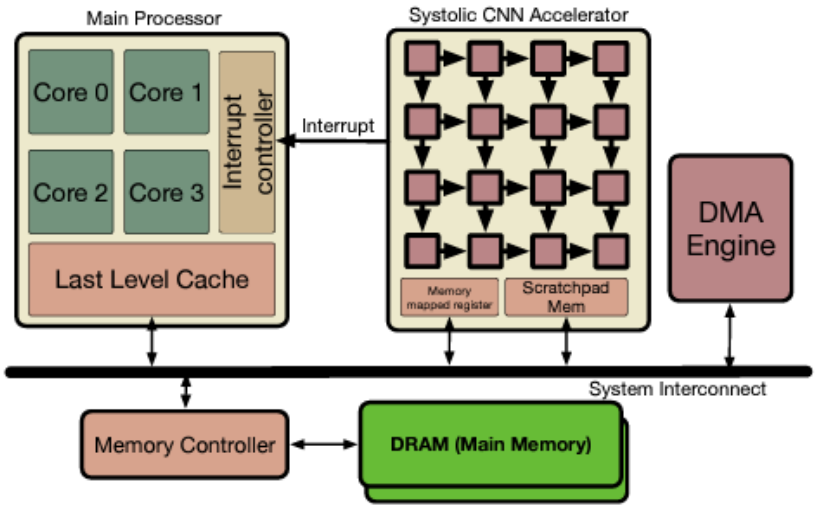

Figure 5 Integration model of accelerator in a systems context

Understanding the systolic array parameters

```
[general]
run_name = scale_example_run_32x32_os

[architecture_presets]
ArrayHeight:    64
ArrayWidth:     64
IfmapSramSzkB:   2048
FilterSramSzkB:  2048
OfmapSramSzkB:  2048
IfmapOffset:    0
FilterOffset:   10000000
OfmapOffset:    20000000
Bandwidth : 10
Dataflow : os
MemoryBanks:   1

[run_presets]
InterfaceBandwidth: CALC
```

Figure 6 Config file parameters

Array height and width indicates the height and width of the array in terms of the number of MACs. IfmapSramSzKb an d FilterSramSzKb indicates the size of the buffer for the two inputs, and OfmapSramSzKb is the size of the buffer for the output. Dataflow indicates the type of data flow you are employing – input stationary (is), output stationary (os), or weight stationary (ws). Please the read [1] to better understand the dataflow parameters.

Understanding the topology parameters

```
Layer name, IFMAP Height, IFMAP Width, Filter Height, Filter Width, Channels, Num Filter, Strides,

Conv1, 224, 224, 7, 7, 3, 64, 2,
CB2a_1, 56, 56, 1, 1, 64, 64, 1,
CB2a_2, 56, 56, 3, 3, 64, 64, 1,
CB2a_3, 56, 56, 1, 1, 64, 256, 1,
CB2s, 56, 56, 1, 1, 64, 256, 1,
IB2b_1, 56, 56, 1, 1, 256, 64, 1,
IB2b_2, 56, 56, 3, 3, 64, 64, 1,
IB2b_3, 56, 56, 1, 1, 64, 256, 1,
IB2c_1, 56, 56, 1, 1, 256, 64, 1,
IB2c_2, 56, 56, 3, 3, 64, 64, 1,
IB2c_3, 56, 56, 1, 1, 64, 256, 1,
CB3a_1, 56, 56, 1, 1, 256, 128, 2,
CB3a_2, 28, 28, 3, 3, 128, 128, 1,
CB3a_3, 28, 28, 1, 1, 128, 512, 1,
CB3s, 56, 56, 1, 1, 256, 512, 2,
IB3b_1, 28, 28, 1, 1, 512, 128, 1,
IB3b_2, 28, 28, 3, 3, 128, 128, 1,
IB3b_3, 28, 28, 1, 1, 128, 512, 1,
IB3c_1, 28, 28, 1, 1, 512, 128, 1,
IB3c_2, 28, 28, 3, 3, 128, 128, 1,
IB3c_3, 28, 28, 1, 1, 128, 512, 1,
IB3d_1, 28, 28, 1, 1, 512, 128, 1,
IB3d_2, 28, 28, 3, 3, 128, 128, 1,
IB3d_3, 28, 28, 1, 1, 128, 512, 1,
CB4a_1, 28, 28, 1, 1, 512, 256, 2,
CB4a_2, 14, 14, 3, 3, 256, 256, 1,
CB4a_3, 14, 14, 1, 1, 256, 1024, 1,
CB4s, 28, 28, 1, 1, 512, 1024, 2,
IB4b_1, 14, 14, 1, 1, 1024, 256, 1,
IB4b_2, 14, 14, 3, 3, 256, 256, 1,
IB4b_3, 14, 14, 1, 1, 256, 1024, 1,
IB4c_1, 14, 14, 1, 1, 1024, 256, 1,
IB4c_2, 14, 14, 3, 3, 256, 256, 1,
IB4c_3, 14, 14, 1, 1, 256, 1024, 1,
IB4d_1, 14, 14, 1, 1, 1024, 256, 1,
IB4d_2, 14, 14, 3, 3, 256, 256, 1,
IB4d_3, 14, 14, 1, 1, 256, 1024, 1,
```

Figure 6 Topology file parameters

Each line describes the characteristics of a layer in the network. Figure 6 shows the topology file for Resnet50. It describes the height and width of the input, output, as well the number of filters as well as the channels and the stride pattern for that layer.

Problem Statement

The goal of this project is to come up with a systolic array architecture that will give the best performance (in cycles) while ensuring the best utilization of the hardware (i.e systolic array)– essentially, we are optimizing for the area delay product. Therefore, you cannot arbitrarily increase the size of the array to get the best performance. We want the most compact hardware as well. This project has two parts

1. Choose any two networks that interest you that are available in the topology directory. You can refer to the relevant papers to understand the network architecture and data flow. For these two networks develop a hardware that gives best performance with the best utilization. Also determine what is the best data flow for that network (input stationary, weight stationary, output stationary). Please refer to submission guidelines to know what sweeps to run
2. In the second part, you are supposed to handcraft a neural network(CNN) on your own from the experience you gained in the first part of the project modelled on Mobilenet, and design the best possible hardware in terms of performance while ensuring maximum utilization. You are also required to determine the best dataflow. Note that this network should be atleast as big as MobileNet (It can be bigger). Once you have designed the optimum hardware vary the filter sizes to provide sensitivity study of hardware utilization (Note:1x1 filters need not be varied).

Submission

1)Provide Config files as well as the output files, report should be limited to 8-10 pages.

2)The report should include a discussion of the rationale for the hardware that performs the best in the case of Part1 as well as Part 2.

3)You can use a square systolic array but be sure to vary the sizes from eyeriss to google config (12x12 to 256x256 in suitable steps).

4)Please provide the rationale for the choice of data flow as well.

5)Include the graphs for points 2-4. We expect at the very least utilization and performance sweeps across the hardware for all three dataflows. You may include any extra graphs that you feel are necessary to illustrate your analysis.

6)There should also be a discussion of which hardware performs the best for all three networks under discussion (Note that there is no right answer for this question, this is to evaluate how well you have understood the hardware and network topology).

References

[1]Samajdar, Ananda; Joseph, Jan Moritz; Zhu, Yuhao; Whatmough, Paul; Mattina, Matthew; Krishna, Tushar. "A systematic methodology for characterizing scalability of DNN accelerators

using SCALE-sim". In IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), 2020.

[2]Conda, https://docs.conda.io/en/latest/

[3] Simonyan, Karen; Zisserman, Andrew. " Very Deep Convolutional Networks for Large-Scale Image Recognition". International Conference on Learning Representations 2015.

[4] He, Kaiming; Zhang, Xiangyu; Ren, Shaoqing; Sun, Jian. "Deep Residual Learning for Image Recognition". IEEE Conference on Computer Vision and Pattern Recognition, 2016.

[5] Chen, Yu-Hsin; Krishna, Tushar; Emer, Joel S.; Sze, Vivienne. "Eyeriss: An EnergyEfficient Reconfigurable Accelerator for Deep Convolutional Neural Networks". CoRR, vol. abs/1704.04760, 2017.

[6] Zhang, Jeff & Rangineni, Kartheek & Ghodsi, Zahra & Garg, Siddharth. (2018). Thundervolt: enabling aggressive voltage underscaling and timing error resilience for energy efficient deep learning accelerators.

[7] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin, “Attention Is All You Need”

[8] Guo-Jie Li and Wah, "The Design of Optimal Systolic Arrays," in IEEE Transactions on Computers, vol. C-34, no. 1, pp. 66-77, Jan. 1985

[9] Kung, "Why systolic architectures?," in Computer, vol. 15, no. 1, pp. 37-46, Jan. 1982