Advanced Computer Arithmetic Circuits and Systems

Project: Neural Network Accelerator for Digit Recognition

T.A. Xinkuang Geng

December 13, 2023

1 Introduction

In this project, you will design a neural network accelerator for digit recognition based on the multi-precision MAC unit in Lab 1.

2 Description

Figure 1 shows the neural network architecture used in this project. Each neuron of the first layer is activated by a rectified linear unit (ReLU). The symbols im, x1, x2, x3, and x4 represent activation values in different stages, and W1 and W2 denote the weights of the two linear layers.

The initial 10×10 input matrix, representing the corresponding image with a handwritten digit, undergoes flattening, matrix multiplication, scaling, ReLU, matrix multiplication, scaling, and generate the output x4. The index of the maximum element in x4 indicates the predicted value for the handwritten digit.

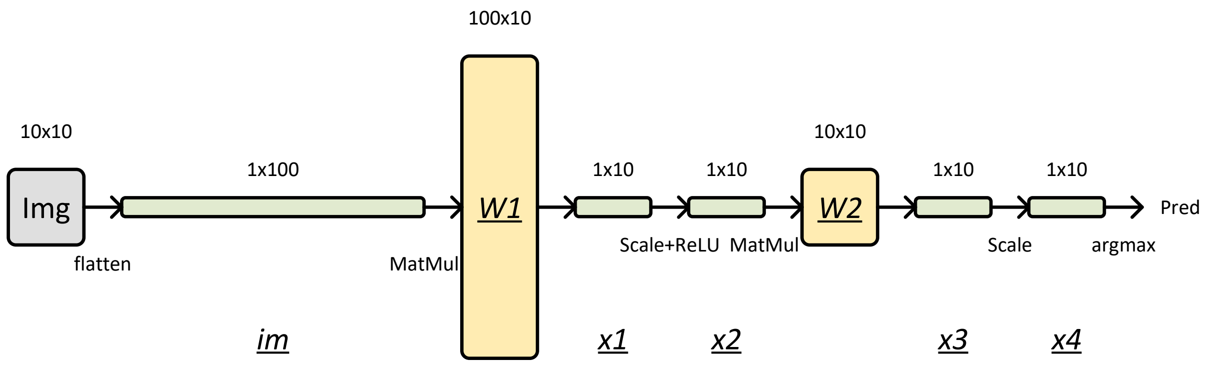

Figure 1: A two-layer neural network architecture with 1.1k parameters for digit recognition.

The inference process can be expressed as

$$\text{pred} = \operatorname{argmax}\left(\left(\operatorname{ReLU}\left((im \times w1) \gg 2\right) \times w2\right) \gg 4\right), \tag{1}$$

where $\times$ represents MatMul. w1 / w2 and x2 / x4 (after shifting) are already restricted to the range of 8-bit signed integer. ReLU truncate negative values to zero as

$$\mathrm{ReLU}\,(x) = \begin{cases} x, & x \geq 0 \\ 0, & x < 0 \end{cases} \tag{2}$$

This network is already pre-trained and quantized. In the “weights.py” of the reference project, parameters for two weight matrices are provided as the following form.

```
w2 = [
[-32, -18, -11,   3,   7, -13,  16, -12,  -6, -18],
[ 13,  16,  -4, -15, -28,  30, -27, -29,  33,  11],
[-11,   2,  18,   8, -12,   2,   0, -11, -26, -13],
[ -3, -16,  10,  -6,  10,  -4, -19,  -8, -18,  15],
[  9,  -5, -20,   4,   0,  22, -24,   1,  -4,  -4],
[ -4,  -4, -18,   0,   5, -17,   1, -14,  21,  12],
[ -9,  12, -17,  21, -16,  -6,   1, -30,  12,  -8],
[ 20, -24,  14, -12, -21,  -1,  -5,  25, -16, -10],
[  9,  -2, -12, -14,  23,  -9, -16, -20, -10, -13],
[ -1,   3, -11, -16,  -4,  -4, -18,  13, -10,   1]
]
```

The input images for the example are also provided in “images.py”. Three images containing different handwritten digits are shown as follows. It is noteworthy that the input images only contain the digits 0 and 1.

```
im4 = [                      im7 = [                      im9 = [
0,0,0,0,0,0,0,0,0,0,         0,0,0,0,0,0,0,0,0,0,         0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,1,0,0,0,         0,0,0,0,0,0,0,0,0,0,         0,0,0,0,1,1,1,0,0,0,
0,0,0,0,0,1,1,0,0,0,         0,0,0,1,1,1,1,1,0,0,         0,0,0,1,0,0,0,1,0,0,
0,0,0,0,1,0,1,0,0,0,         0,0,0,0,0,0,1,0,0,0,         0,0,0,1,0,0,0,1,0,0,
0,0,0,1,0,0,1,0,0,0,         0,0,0,0,0,0,1,0,0,0,         0,0,0,0,1,1,1,0,0,0,
0,0,1,1,1,1,1,1,1,0,         0,0,0,0,0,1,0,0,0,0,         0,0,0,0,0,0,1,0,0,0,
0,0,0,0,0,0,1,0,0,0,         0,0,0,0,0,1,0,0,0,0,         0,0,0,0,0,1,0,0,0,0,
0,0,0,0,0,0,1,0,0,0,         0,0,0,0,1,0,0,0,0,0,         0,0,0,0,1,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,         0,0,0,0,1,0,0,0,0,0,         0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,         0,0,0,0,0,0,0,0,0,0,         0,0,0,0,0,0,0,0,0,0,
]                            ]                            ]
```

Despite containing only 1.1k 8-bit parameters, this neural network has achieved close to 90% accuracy on the MNIST dataset. You can construct different images to test the network by filling the 10×10 matrix with 1s.

The “sim.py” in the reference project provides a simulation code. In fact, your task is to design an accelerator that implements the computation process in “sim.py” based on the your multi-precision MAC.

Figure 2 shows a possible accelerator architecture. You can implements the matrix multiplication through an processing element (PE) array.

Figures 3 and 4 respectively show one-dimensional PE arrays with different data flows, and you can choose one strategy to implement the circuit.

The data is sequentially read from SRAM into the PE array. For example, Pa in Figures 3 and 4 completes the following MAC operations:

$$ya = a0 \times x0 + a1 \times x1 + a2 \times x2 + a3 \times x3 + a4 \times x4. \tag{3}$$

Notice that, compared to <u>x2</u> and <u>w2</u> within $[-128, 127]$, the MAC operations between <u>im</u> and <u>w1</u> can be implemented based on an 8×4 multiplier, given that <u>im</u> contains only 0s and 1s. This

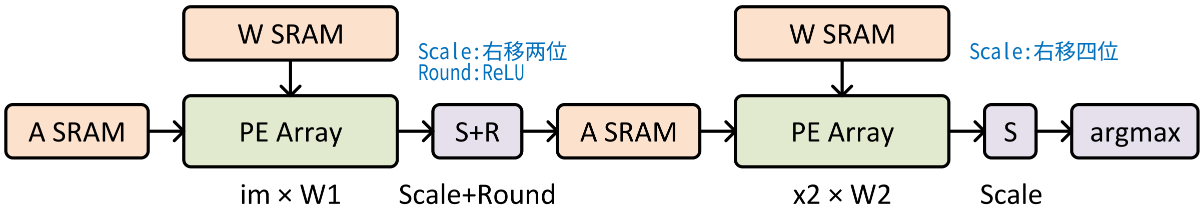

Figure 2: Computation Flow.

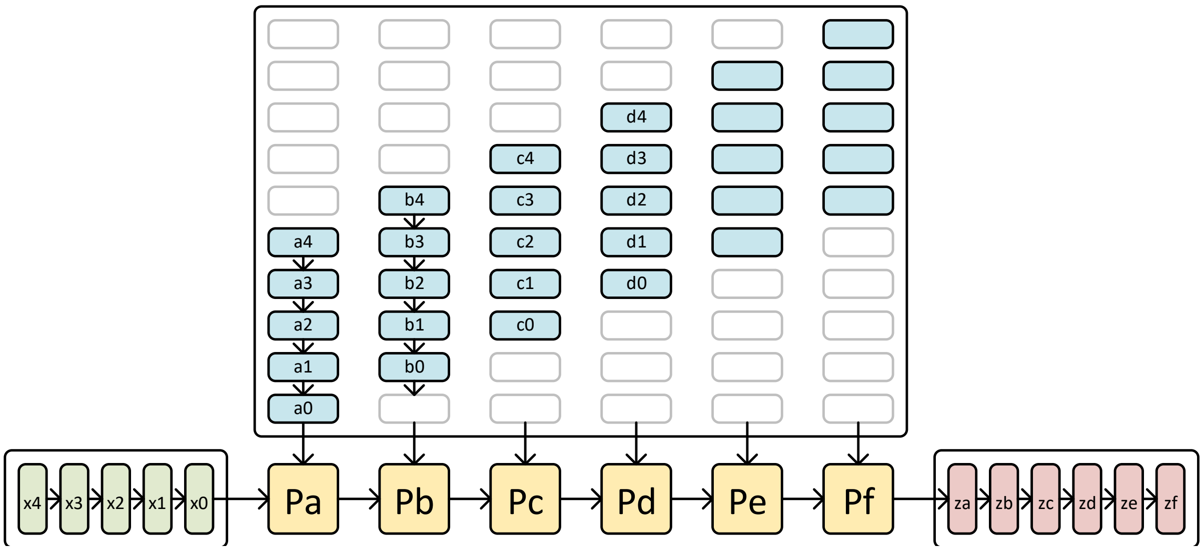

Figure 3: Systolic PE Array.

implies that when computing $im \times w1$, the PE array can simultaneously process two input images, as illustrated in Figure 5.

It can be seen that PE arrays of different architectures and precisions exhibit different requirements for the arrangement of data in SRAM. You should determine the shape, size, and initial data layout of the SRAM instance based on your architecture.

As an example, we provide a script “bin.py” for generating the bin files, which can be used to initialize your SRAM using the readmemb function in Verilog.

> It is recommended to place the SRAM in your testbench, and the accelerator is only connected to SRAM through its ports. Note that the provided SRAM file is only a simulation model and should not be included in logic synthesis.

> It is recommended to first run “python sim.py” and have a clear understanding of the computation steps before designing the accelerator.

3 Requirements

1 Please ensure that the files you submit are organized into the following structure and packed as a zip file.

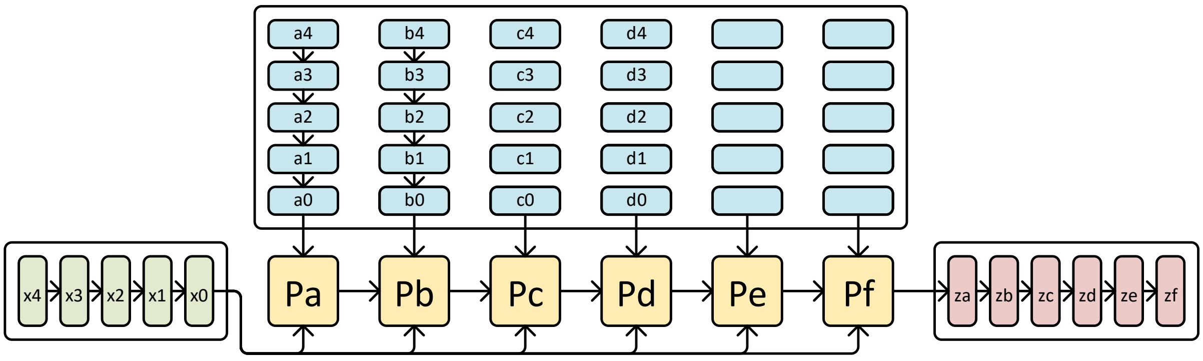

Figure 4: Parallel PE Array.

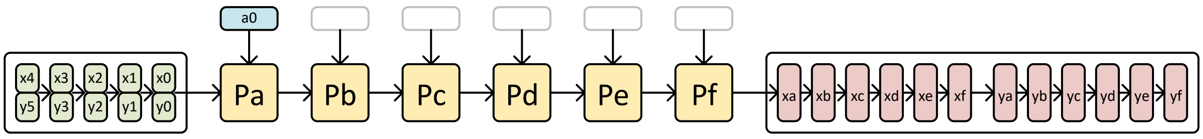

Figure 5: Reduce precision and increase throughput.

```
     YourID_YourName_Project.zip  # such as 122012340001_XXX_Project.zip
     |-- report.pdf
     |-- py
     |    |-- bin.py  # Generate bin files for your testbench.
     |    |-- images.py  # You can construct new images in this file.
     |    |-- sim.py
     |    |-- weights.py
     |    |-- other_auxiliary_scripts.py
     |-- rtl
     |    |-- your_modules.v
     |    |-- fused_signed_mac_32p8t8_2x24p8t4.v
     |    |-- sram.v
     |-- tb
     |    |-- your_testbench.v
     |    |-- xxx.bin  # Bin files used in your testbench.
     |    |-- # Indicate the digit in naming, such as im7.bin and im6im9.bin.
```

2 Arithmetic operators ($+, -, *, /$) are prohibited in the code of the PE array. But feel free to use any Verilog operators in your control logic.

3 Please ensure that the Python scripts you submit can generate proper bin files for your testbench.

4 Some test points (several images different from the provided examples) will be applied to your design, and the pass rate will determine your final score.

5 In the experimental report, it is suggested to provide a detailed description of your design methodology, incorporating codes and figures. Screenshots and analysis of simulation waveforms and synthesis results are necessary. It is encouraged to construct some new images yourself and show the recognition results in the report.

4 Grading

- Test Points – 30%

- Multi-Precision – 10%
- Synthesis Results – 20%
- Experimental Report – 40%