** Information Technology University**

**Advanced Digital System Design Fall 2021**

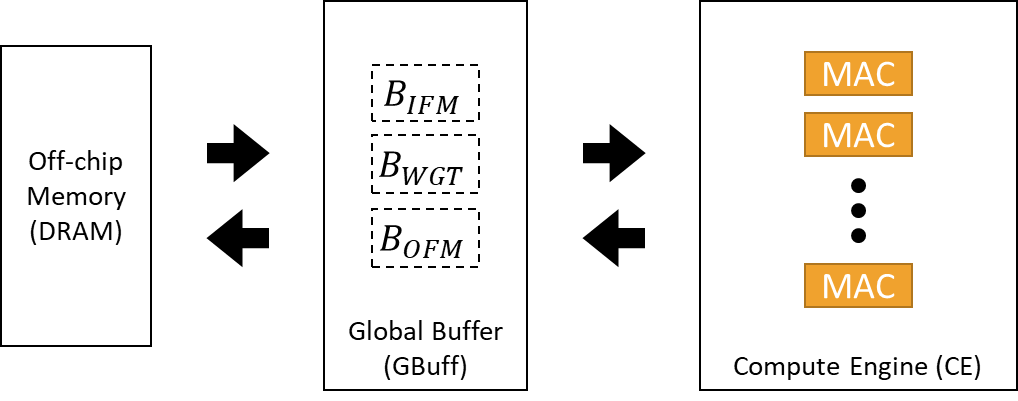
**Final Project**

**Handwritten Digit Recognition on FPGA**

|  |  |
| --- | --- |
| **Date of Launch:** Tuesday, 11-Jan-2022 | **Max Marks:** 500 |
| **Project Demo & Submission:** Monday, 31-Jan-2022 | **Time:** 5:30-9:00PM |
| **Instructor:** Prof. Dr. Rehan Hafiz | **TA:** Hazoor Ahmad |

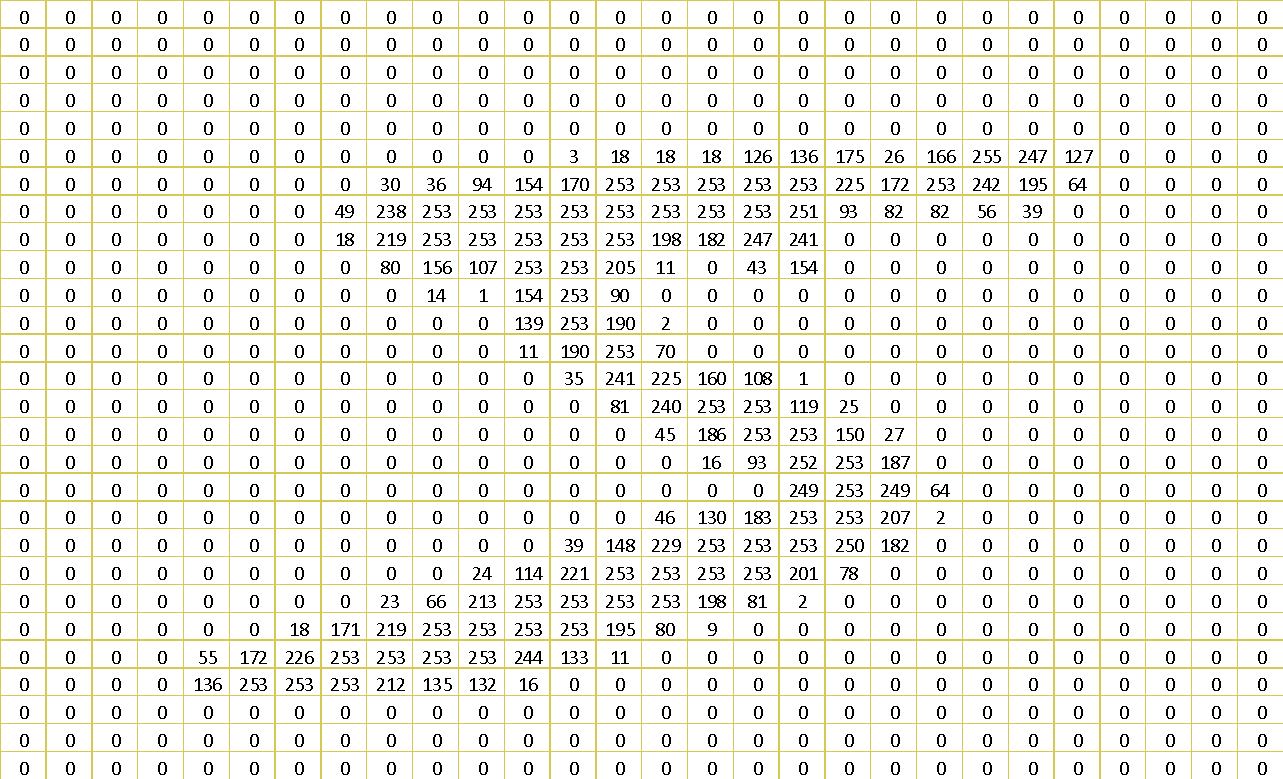
You are hired at Deep Network Design (DND) Lab. Your task is to design a CNN processor shown in **Fig. 1**. It has the following main components:

* **Off-chip memory**: Off-chip memory is enough to keep all input feature maps (IFM), Weights (WGT) and output feature maps (OFM) which are to be used for the whole processing operation.
* **Global Buffer = 1KB**: Global Buffer is an on-chip memory. Based on the IFM, OFM and WGT tiles to be processed you can partition GBuff into , , and .
* **Compute Engine:** CE having an array of MAC Units.

****

**Fig. 1** CNN Processor consisting of DRAM, Global Buffer, and Compute Engine.

# **Phase A**



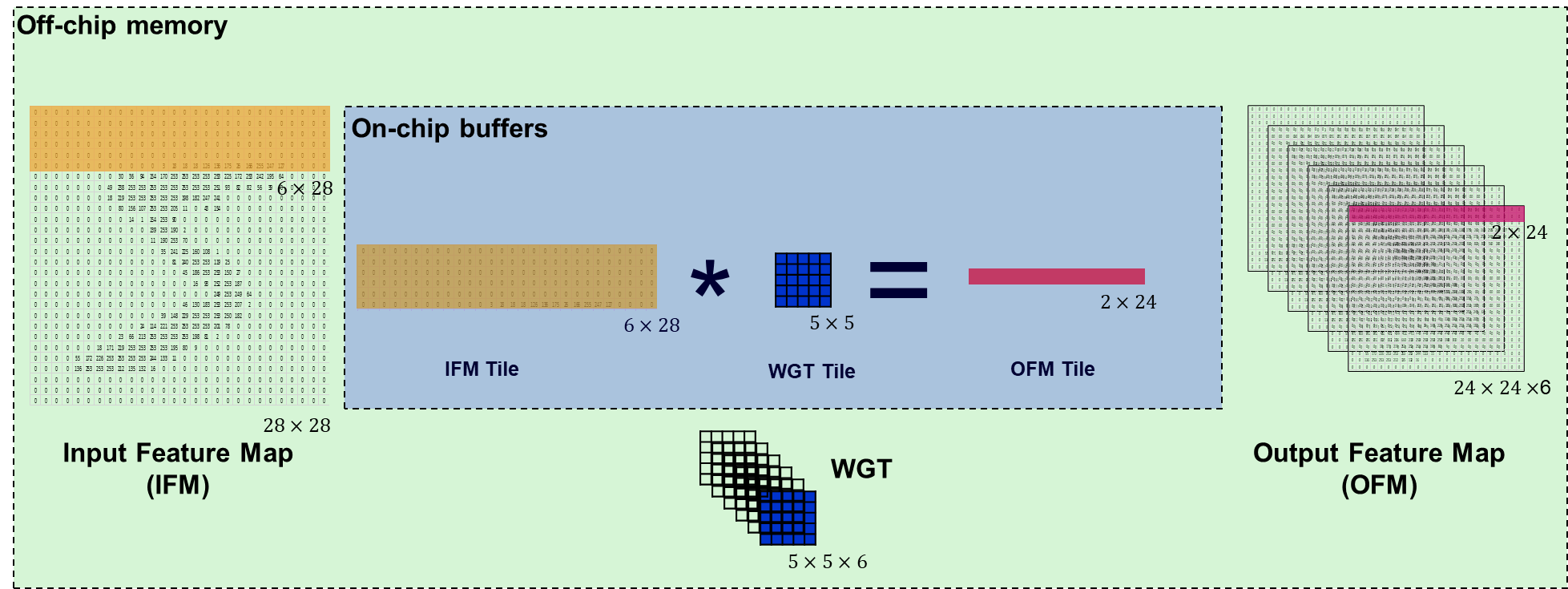
**Fig 2.** Input Image of size

[**ADSD Project Folder Link**](https://drive.google.com/drive/folders/1Czlyi30MXY7DgK71fbuCJ5trLP2sSK2w?usp=sharing) **contains all the supporting files including:**

1. **Input image file (bin\_img\_0\_FP\_3\_13.txt):** This file contains an array of 16-bit binary numbers. These binary numbers are in fixed point format with 3-bits for the decimal part and 13-bits for the fractional part. The total numbers are 784 which are the processed values of the input image of size . The file is arranged in such a way that first twenty-eight numbers correspond to first row of the image shown in **Fig. 2** and 29th to 56th numbers make second row of the image shown in **Fig. 2** and so on.
2. **Conv1 weight file (Conv1\_FP\_1\_15.txt):** This file also contains an array of 16-bit binary numbers. These binary numbers are in fixed point format with 1-bits for decimal part and 15-bits for fractional part. The total numbers are 150 which belong to six filters of size . The file is arranged in such a way that first twenty-five numbers correspond to five rows of the 1st filter and 26th to 50th numbers also belong to the five rows of the 2nd filter and so on.

Your task is to perform 2D convolutions of type ‘valid’ with ‘ReLU’ activation on the input image from file (bin\_img\_0\_FP\_3\_13.txt) using the six filters from the file (Conv1\_FP\_1\_15.txt). The output of the valid convolution on input with six filters of is. Then you have to perform ‘average’ pooling which results in .

First of all, make off-chip memory in your design which should have the space to linearly store the values of image, weights and the output. It shall have the values of image and weight at loading. You are then required to perform the following tasks:



**Fig 3.** Tiled processing of convolution with filters

# **Question 1 (A) Tiled Convolution with Activation**

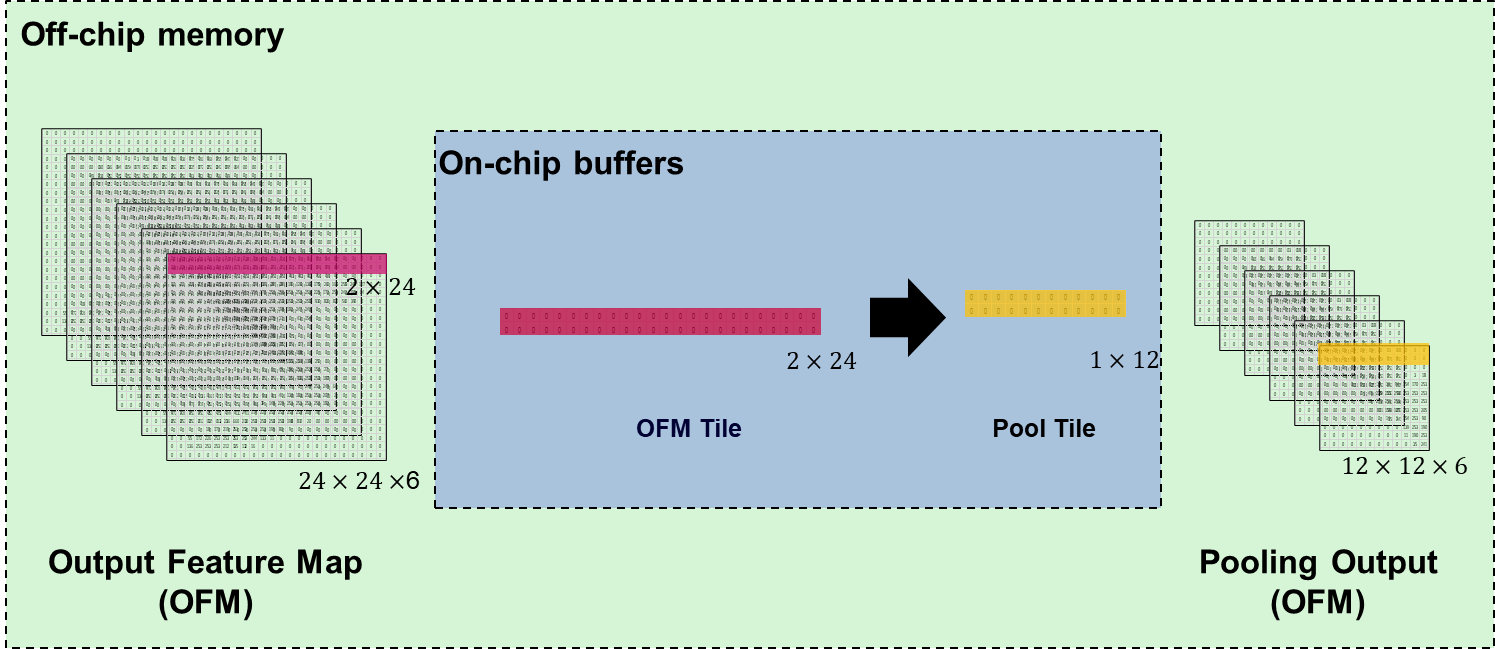
Due to limited on-chip memory of the edge devices we have to process convolution in the form of tiles. Therefore, for the time being you should make on-chip buffers of size , , and for IFM tile (), WGT tile () and OFM tile () respectively as shown in **Fig. 3**. Tiled convolution consists of the following sub-designs:

* Design an address generation mechanism or scheduling engine which fills the on-chip buffers with the next tiles of data upon requirement.
* Design a Compute Engine (CE) using as many MAC units as you like.
* Design a controller for CE which processes on-chip IFM and WG`T tiles to produce OFM tile.
* Activation in this case is ‘ReLU = Rectified Linear Unit’ which can simply be implemented by just checking the negativity of the convolution output. If the output value is negative ‘ReLU’ changes it to zero else no change in output is made.

# **Question 1 (B) Average Pooling**

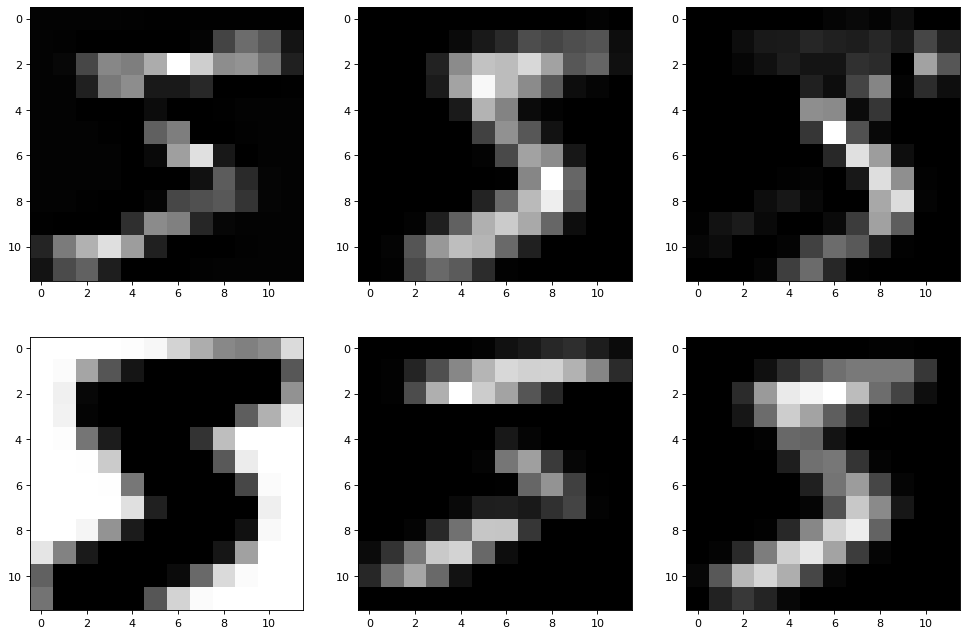
In average pooling four neighboring pixels of an OFM are averaged and the result replaces the four values.

Hint: You can reduce the huge intermediate memory of OFM by merging pooling with convolution. So, think about it !!



**Fig 4.** Tiled processing of pooling

Once you successfully design and simulate the convolution + activation + pooling engine, you have successfully implemented the first layer of the CNN Processor. Expected outputs are shown in **Fig. 5**. And their expected binary files are included the [**ADSD Project Folder Link**](https://drive.google.com/drive/folders/1Czlyi30MXY7DgK71fbuCJ5trLP2sSK2w?usp=sharing).

****

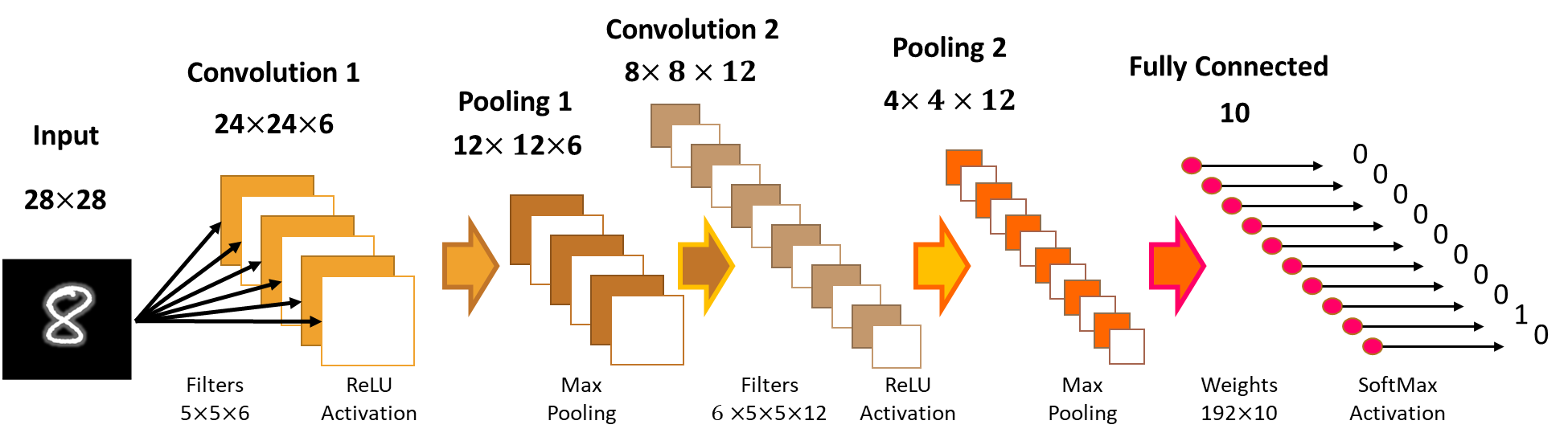
**Fig 5.** Expected outputs of the first convolution + activation + pooling layers

**Groups Evaluation:**

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **G#** | **Reg #** | **Name** | **24-Jan** | | **27-Jan** | | **31-Jan** | | **03-Feb** | | **Overall** |
| **ATT** | **EVA** | **ATT** | **EVA** | **ATT** | **EVA** | **ATT** | **EVA** |  |
| **Group 1** | msee21012 | Agha Ali Nawaz | P | 50 |  |  |  |  |  |  |  |
| msee21014 | Muhammad Rafay Mukhsneen | A | 0 |  |  |  |  |  |  |  |
| msee21027 | Saad Iftikhar | A | 0 |  |  |  |  |  |  |  |
| **Group 2** | msee21001 | Shahab Zada | P | 30 |  |  |  |  |  |  |  |
| msee21010 | Faizan Ali Khan | P | 30 |  |  |  |  |  |  |  |
| msee21026 | Hunain Iftikhar | P | 30 |  |  |  |  |  |  |  |
| **Group 3** | msee21005 | Arbab Haider | P | 40 |  |  |  |  |  |  |  |
| msee20009 | Muhammad Ahmad Waseem | P | 40 |  |  |  |  |  |  |  |
| msee21015 | Muhammad Bilal Chaudhary | P | 40 |  |  |  |  |  |  |  |

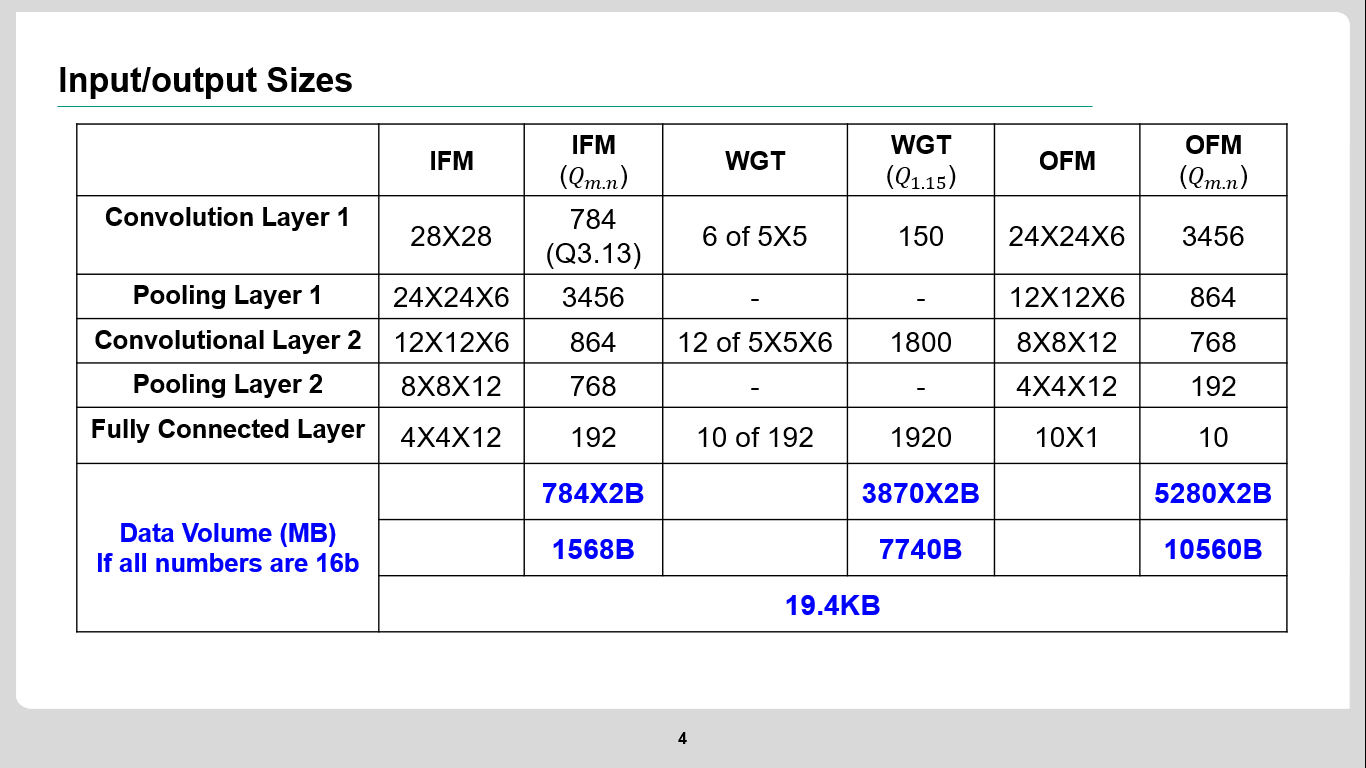
# **Phase B**

Once you have successfully implemented Question 1 you have implemented the First Layer of LeNet Model for Handwritten Digit Recognition shown in Model Hierarchy in **Fig. 6** and Model Summary in **Table 1** .

****

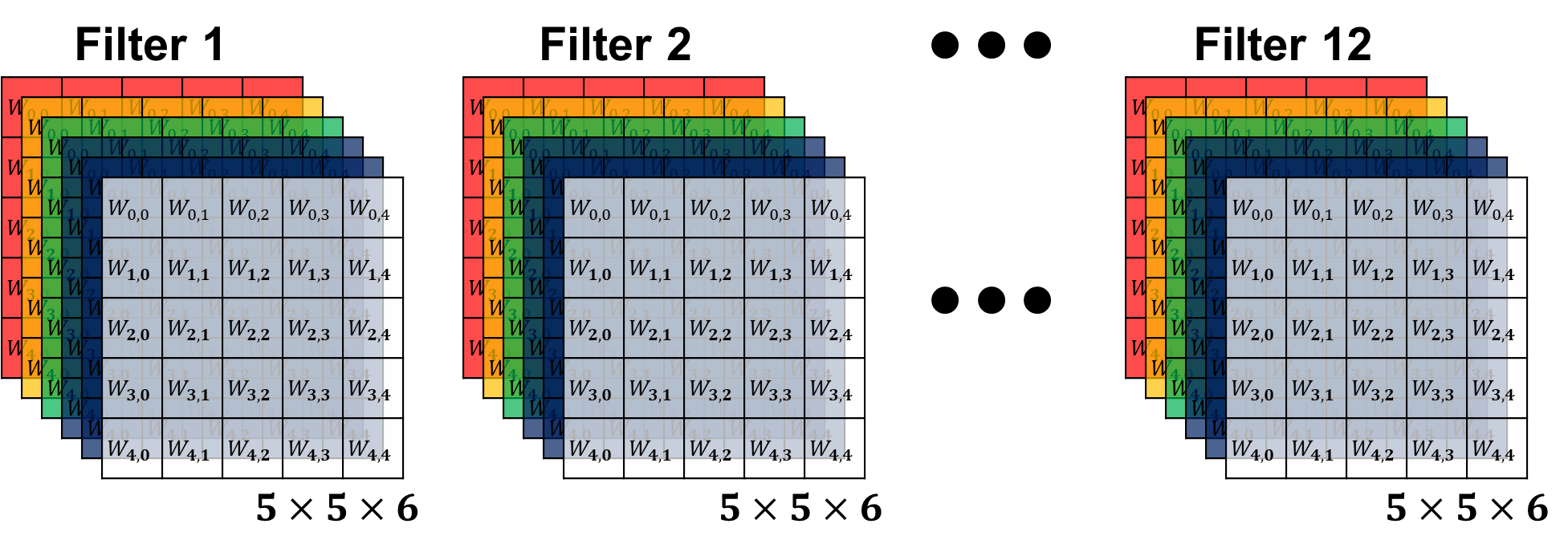
**Fig 6.** LeNet Model for Handwritten Digit Recognition

**Table 1.** LeNet Model Summary: Dimensions of IFMs, WGTs and OFMs



We assume that you have a 3D output of size . Check your project folder [**ADSD Project Folder Link**](https://drive.google.com/drive/folders/1Czlyi30MXY7DgK71fbuCJ5trLP2sSK2w?usp=sharing) for the following new files:

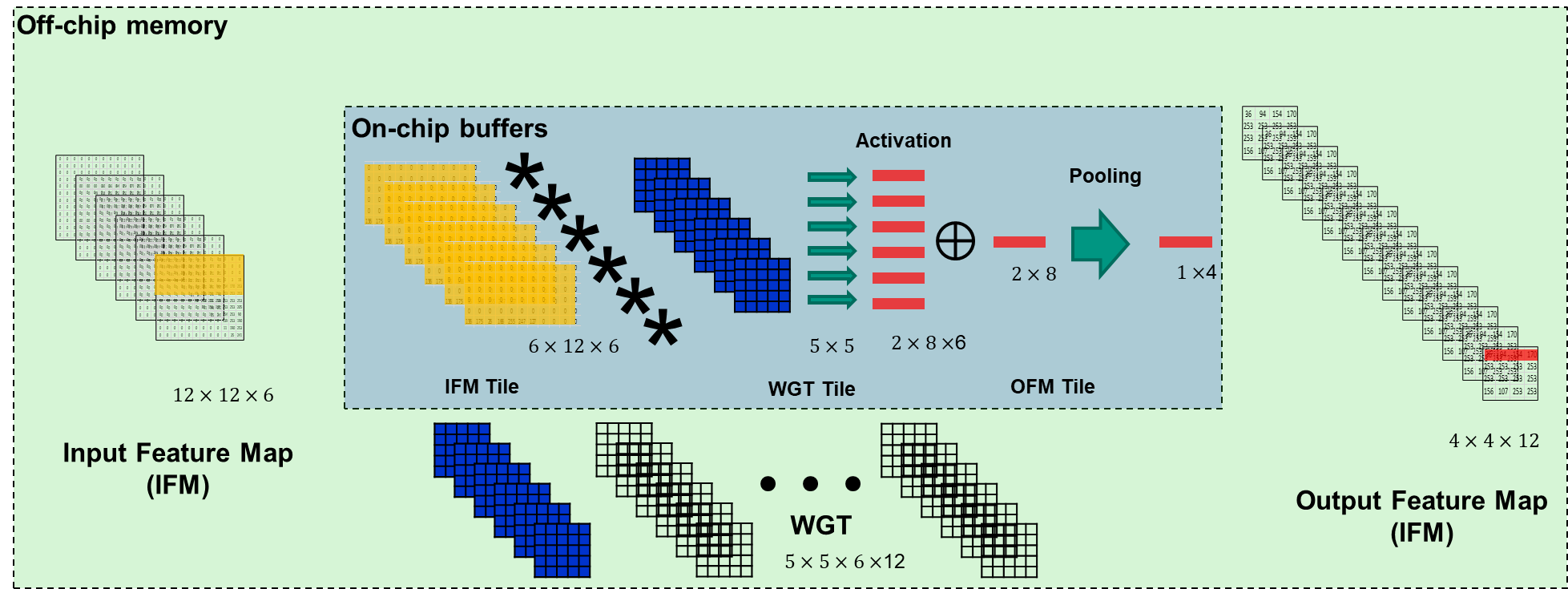
1. **Expected Outputs of Layer 1 (Layer1\_Outputs.xlsx):** This file contains two sheets. First sheet named “Pool1” have six matrices and the next sheet named “Conc1” contains six matrices both sheets in floating point format. You can confirm the result of your Convolution 1 or Pooling 1 output by comparing in MATLAB or Python or by any means.
2. **Conv2 weight file (Conv2\_FP\_1\_15.txt):** This file contains an array of 16-bit binary numbers. These binary numbers are in fixed point format with 1-bits for decimal part and 15-bits for fractional part. The total numbers are 1800 which belong to 12 filters of size . The file is arranged in such a way that first 150 numbers correspond to the six channels of the 1st filter and 151st to 300th numbers belong to the six channels of the 2nd filter and so on. For better understanding weights organization of 2nd layer please refer to **Fig. 7**.

****

**Fig. 7 Demonstration of Weights of 2nd Convolutional Layer of LeNet**

# **Question 2 (A) Implementation of 2nd Layer**

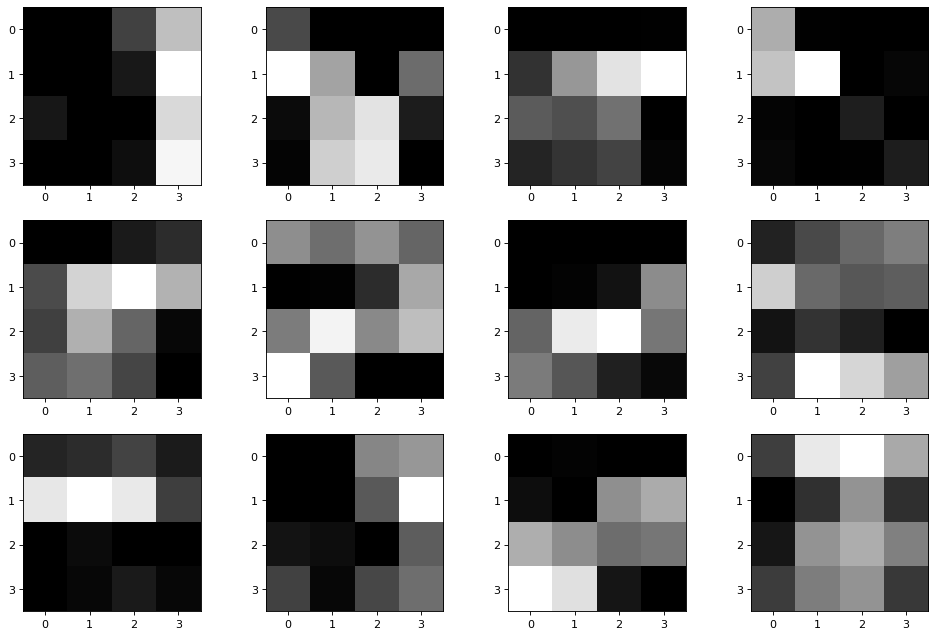
Similar to 1st layer of LeNet Model, second layer contains Convolution, Activation and Pooling operations. The only difference is that the input of convolution is 3 dimensional (output from pooling 1). To process 3D input, frames of the 3D IFM (having dimension ) can be processed individually (similar to 1st layer) and outputs of all channels are then added together to form the OFM frame. This is demonstrated in **Fig. 8**.



**Fig. 8 One possible flow of performing 2D convolution on 3D input data**

Therefore, you can accomplish the implementation of second layer of LeNet by performing the following tasks:

* **Extension to off-chip memory**: Extend your off-chip memory in order to accommodate the WGTs () and OFM (192) of the 2nd layer of LeNet.
* **Design/Extend/Modify the Compute Engine (CE)** to process convolution on 3D IFM data serially or in parallel (your own design choice). Having the following units
  + Should process 2D convolution on the on-chip tile
  + Include an Activation unit capable to applying “ReLU” activation (Explained in Question 1) on every output.
  + Process every frame from every on-chip OFM tile to produce pooling output.
* **Make a comprehensive schedule engine** (address generation unit) which is able to
  + Fill on-chip tiles
  + Read and store output to off-chip memory



**Fig 9.** Expected outputs of the 2nd convolution + activation + pooling layers