Final Report Group01

110061638 呂政和 110061639 林致佑 10061641 鄧向凱

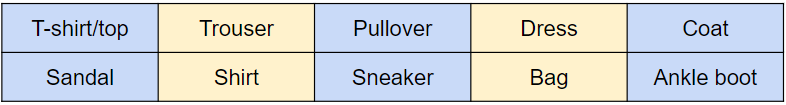
Github: https://github.com/hank871116/HLS\_final\_project

1. **Background Introduction**

Based on lab FINN, we try to add customized IP after NN core, which is generated by FINN, for our target of the final project.

**Fashion MNIST**

First of all, we decided Fashion MNIST to be the dataset of our NN core. It has a training set of 60,000 examples and a test set of 10,000 examples. Each example is a 28x28 grayscale image, associated with a label from 10 classes. The classes are shown below:



The main reason we choose Fashion MNIST as the dataset of NN core is that the pattern size is much smaller than other datasets do. The graph below shows some patterns in this dataset.



**Vision Library**

Since we have to stream the data into the NN IP core, we should resize the origin pattern to fit the required data size of the input port of NN core. The Vision Library provided by Xilinx supports the resize function properly. It can help us resize the input image into a proper format to fit the model of VGG9. Also, we learn how to use it through the tutorial of xilinx and some guidance from github.

1. **Function Description**

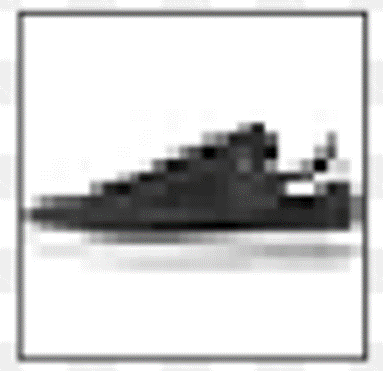
**Our Target**

Our target is to design a function which can identify the test images from Fashion MNIST and add label text on the test images.

**Function Description**

Our final project can be divided into two main parts. The first part is to identify the test data and to output the integers from 0 to 9, which represent ten different categories. The second part is that we will convert the numbers into label text and paste them onto the image as output. In this way, we can directly confirm the results of the function. The below left graphs are the origin data, and the below right graphs would be the output data. Take “Sandal” as an example, if our NN core identifies the pattern successfully, it is expected that we will get the picture with a “Sandal ” label from the output port.

****

****

**How to Do It**

To generate NN IP core, basically we use the .ipynb file from lab FINN; to test NN core, we write python to detect the accuracy; to generate the other IPs, we use Vitis HLS; to connect wire and generate bitstream file and so on, we use Vivado. Also,we choose pynq-Z2 as the FPGA platform.

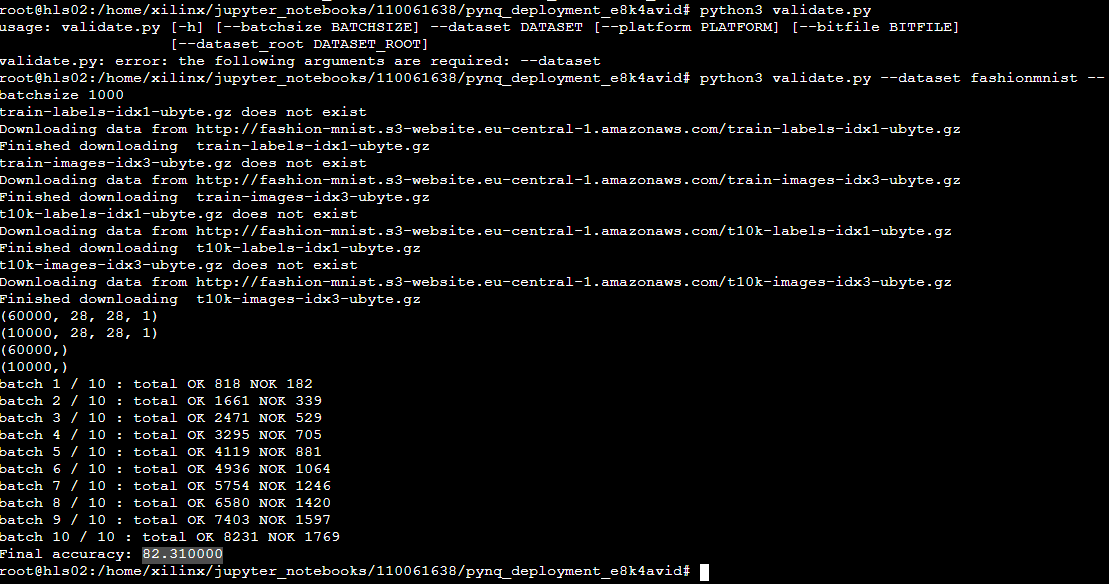
1. **Results**

First, we train our own FashionMnist NN core, as we had set the accuracy to be high enough to use, you can check the training result below that the accuracy is 86.78%, it achieves the goal we set in the proposal.

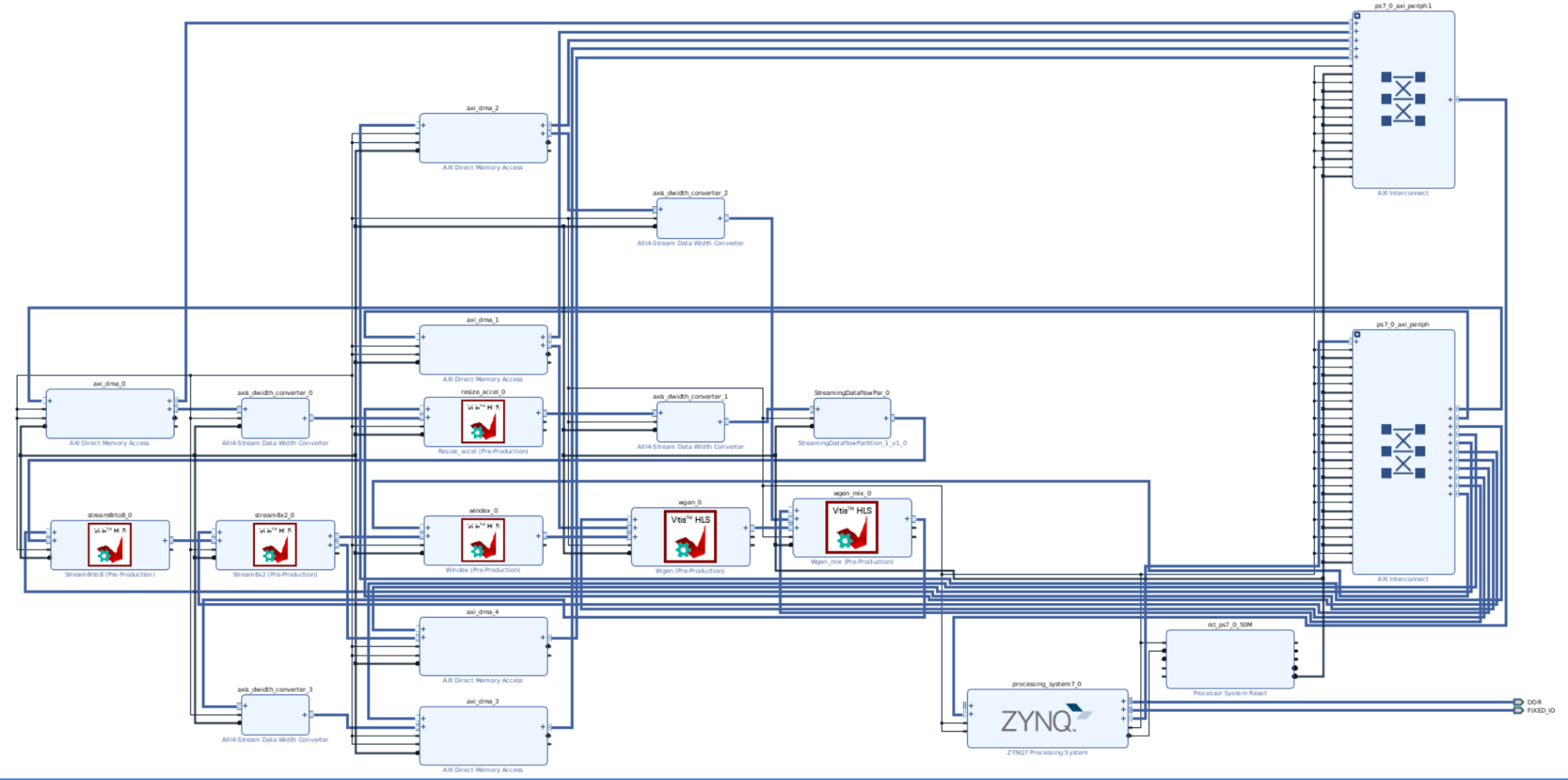


Fig. NN core accuracy

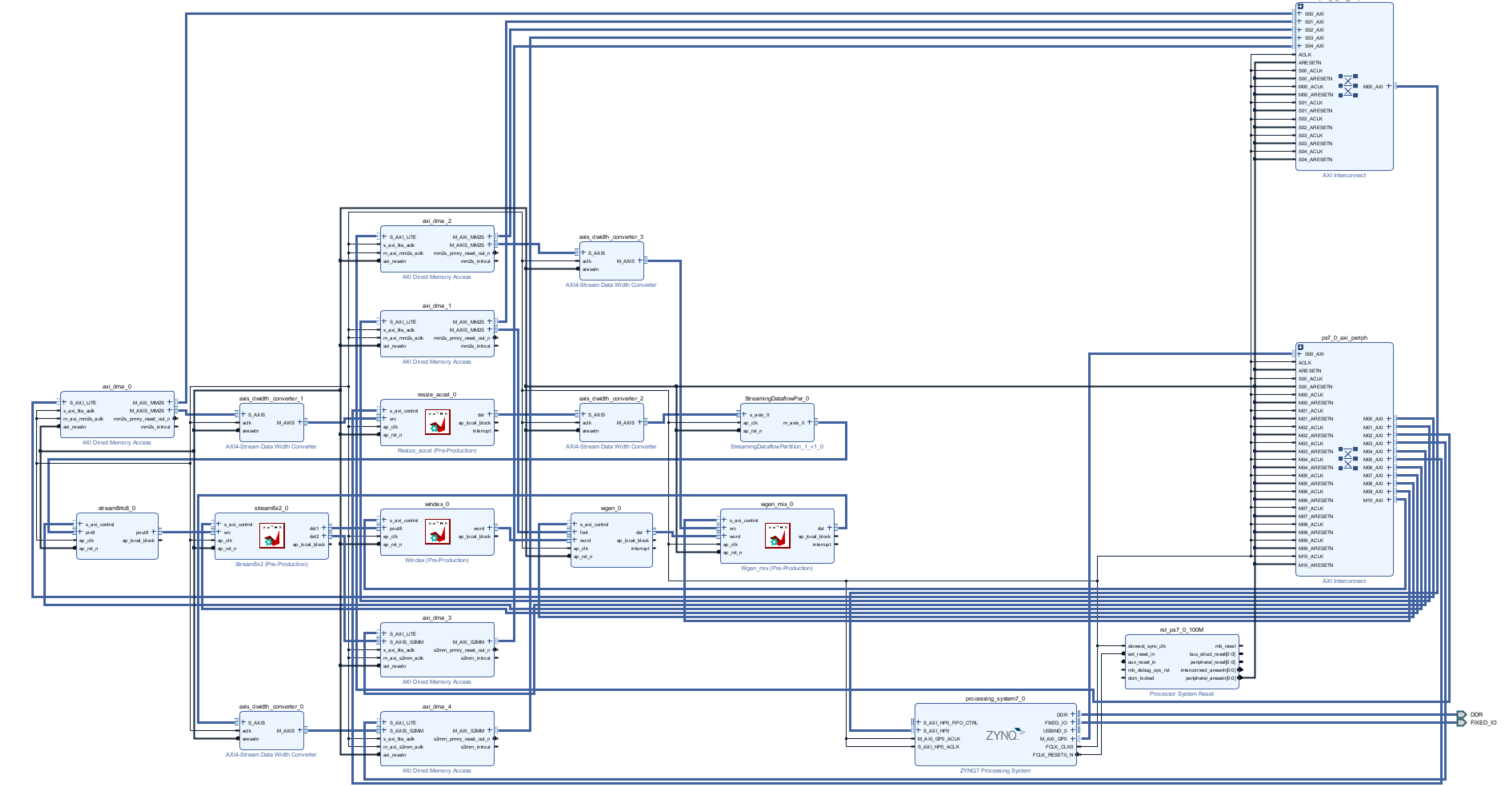
And the accuracy apply onto pynq is 82.31%



Second, we tried to reproduce the whole project from the NTU group to ensure the method is correct. The figure below is the Vivado block design that is exactly the same as the NTU group.



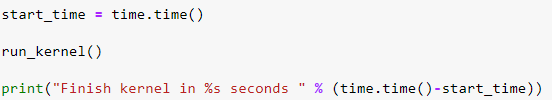
We also use our NN model IP to generate a block design, figure below is the vivado screen shot.



After we finish generating bitstream, we put the bit file into the jupyter notebook to see the result running on the pynq board. First of all, we load the design using Overlay() and initialize required accelerators. (ex: design.axi\_dma\_0) Afterwards, allocate input and output buffer with appropriate size for DMA channel transfer. Then, we can set accelerator attributes such as font height/width and image resizer source/destination size.

| Original image | Caption added by HW |
| --- | --- |
|  |  |

On the jupyter notebook, we do the kernel runtime evaluation with time.time() function to estimate the performance. The kernel ran for about 37.47 ms for one image classification and caption adding.





1. **Issues we encountered**

**A.** Due to the input image of Cifar10 and FashionMnist, we had to rearrange the padding in each layer in the vgg9 model, Cifar10 data size is 32x32x3, FashionMnist data size is 28x28x1.

**B.** When reproducing the project of the NTU group, there are several IPs been used in this project, such as stream8rto8,windex… and so on.Most of them we can simply put the Cpp code into VITIS HLS and process the synthesis and also export RTL .But one IP, resize, includes the other header file related to the Vision library, so we download the whole file in Xilinx and get the file we need, totally 15 files.

**C.** After finishing generating the IP, the next step is to generate the NN core into an IP, but refer to the lab before, we didn’t find any code to produce the IP, we then go to check the finn github from Xilinx. We do find some useful tool to generate the core IP into the file, after adding it to the vivado repository, we get the NN core IP we needed.

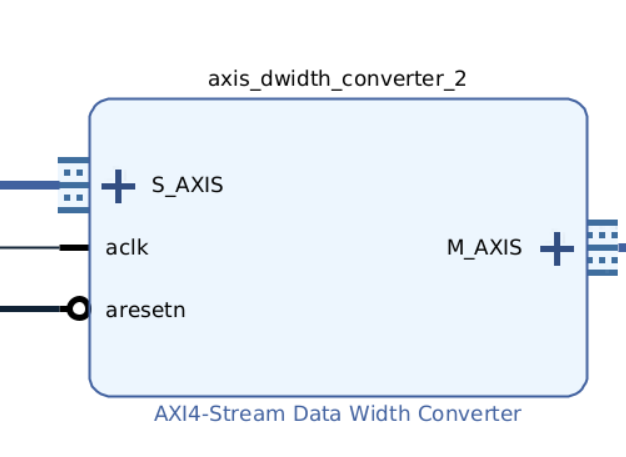


Fig. NN core IP

**D.** After auto connect, there are still some ports unconnected, we need to connect those ports manually.

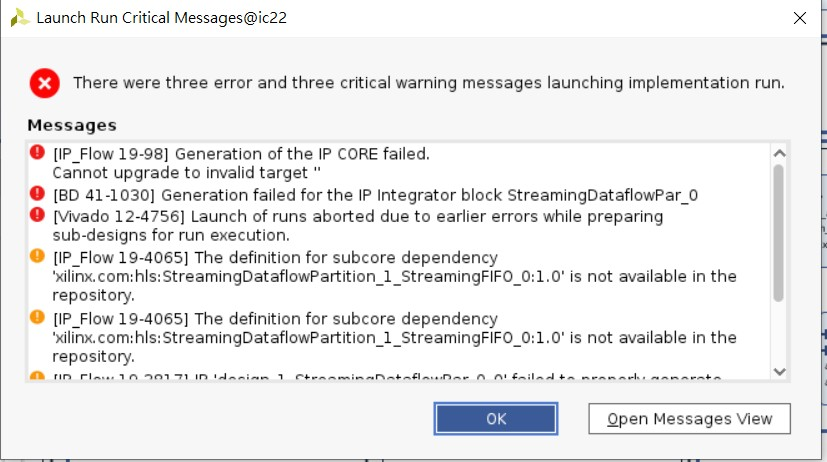
**E.** When generating bitstream files, there’s an error that the file path is too long to be compiled, we then use linux to generate the bitstream file. But there’s no pynq-z2 board file in it, so we turned to the TA for help, and it was solved after that.

**F.** Though our functionality is similar to the final project from NTU, due to the difference of the NN core and dataset, we still need to put some effort into the code. Take resize kernel for example, since the initial image pixel is 28\*28, we set the downscale parameter to 5, and it is expect that the resulting image pixel will be 24\*24. In addition, different datasets contain different classes, so we need to change the text represented by each label before generating them. Graph below shows different text of the label generated by the kernel.



**Unsolved**

**A.** On linux, a new error is reported when generating a bitstream file(shown below).



**5. References**

<https://github.com/Xilinx/finn>

<https://github.com/Xilinx/SDAccel_Examples/issues/23>

<https://support.xilinx.com/s/question/0D52E00006iHlrHSAS/cannot-upgrade-to-invalid-target-?language=en_US>

<https://xilinx.github.io/Vitis_Libraries/vision/2022.1/index.html>