F1B404 - Artificial intelligence implementation

Supervisors: Jérémy Nadal, Nicolas Farrugia

16 January 2018 - 12 February 2018

1 Objective

This supervised work is divided into 7 sessions, each lasting 3 hours. The main objective is to design and implement, on FPGA target, a character recognition algorithm based on neural networks. The training of the network, based on the MNIST set, must be performed on computer, while the classifier is executed on FPGA.

The classifier must be integrated in logic programmable part of the **Zynq Z7020 CLG484** system on chip, with a target performance of **97%** on the **MNIST test set** (grayscale images of size 28×28). At least **10000 classifications per second** are expected. There are no specifications regarding the latency and the power consumption. The input/output interfaces are given in appendix 1. For simplicity, the classifier must be a **Fully Connected Neural Network (FCNN) with a few hidden layers**.

2 Planning

7 milestones has been identified in this project:

1. The network must be designed with Pytorch, in floating point format (computer precision). The hyper parameter choices are left to the students discretion.
2. The weights, bias and input/output layer samples must be quantified in fixed point format. The bit format used in the network is left to the students discretion.
3. Design of the hardware architecture, estimation of the hardware resources, especially the memory requirement.
4. Description of the hardware architecture in VHDL.
5. Debug and validation through a testbench (provided on Moodle). The supervisor validates the architecture behavior on the full test set.
6. Integration on the Zedboard, validation in a real platform.
7. Restitution of the solution: oral presentation!

Table 1 shows a planning of the milestones the students must work on during each session.

Table 1: Planning of the sessions and milestones

Session	22/01	23/01 a.m.	23/01 p.m.	...	11/02 a.m.	11/02 p.m.	12/02 a.m.	12/02 p.m.
Milestones	1,2	3	4	...	4,5	5	6	7

3 Deliverables and evaluation

The expected deliverables are the following:

1. A report of a few pages (around 3 or 4) describing the implemented hardware architecture. A justification of the different design choices must be included. **Deadline: before the 5th session (11/02)**.

2. The VHDL code of the implemented neural network. The behaviour of the produced VHDL code will be tested by the supervisor on the full MNIST test set (10000 images). **Deadline: before the 7th session (12/02)**.

During the last session (12/02 aftermoon), all the student must present their work in the form of an oral presentation. As support, slides must be prepared. The presentation lasts **15 minutes**, followed by **5 minutes** of questions. This project will be **evaluated based on the deliverable results and the oral presentation**. The notation is as follow:

- **7 points** for the report (first deliverable).
- **3 points** for the validation on the full MNIST test set (second deliverable).
- **10 points** for the oral presentation.

4 Availables resources

To faciliate the software and hardware implementation of the classifier, the following resources are provided on Moodle:

- A Python code to be completed.
- A VHDL code of the FCNN top unit, to be completed. More details are presented in Section 5.
- A VHDL code of a generic LUT component for easily storing the weights coefficients of the network.
- A VHDL code of a generic RAM components (single port or dual ports).
- A VHDL testbench the students must use to debug and validate their implementation.
- A package containing basic functions (ex: log2) and array types.

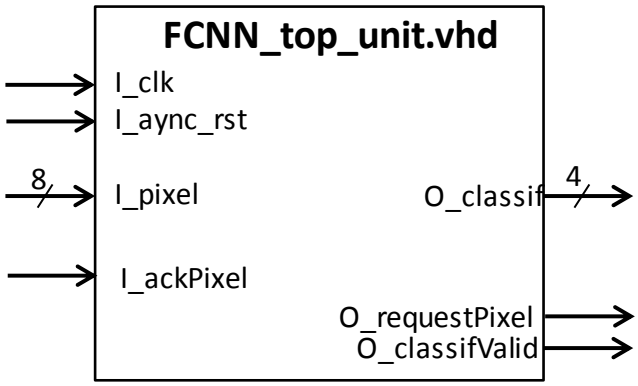

Figure 1: Input and ouput inferfaces of the FCNN top unit

5 Interface of the FCNN unit

A VHDL code of the FCNN unit to implement is given on Moodle. The input and output interfaces of the FCNN top unit are shown in the Figure 1, and a description of those inputs/outputs is provided below:

- *I_clk* and *I_async_rst*: the typical system clock and asynchronous reset (high state for activation).
- *I_pixel*: the signal corresponding to the pixel value, coded on 8 bits. Each pixel are send serially to the FCNN unit.
- *I_ackPixel*: acknowledge the request pixel signal (*O_requestPixel*), and inform that the *I_pixel* signal has a new valid value and must be processed after the following clock's rising edge.
- *O_requestPixel*: active at 1, request for the next pixel value.
- *O_classif*: the output signal corresponding to the obtained class index, from 0 to 9 (therefore, coded on 4 bits).
- *O_classifValid*: inform when the *O_classif* output is updated, i.e. a new classification result is obtained. This signal must be set at 1 **during one clock cycle**.

Figure 2 (next page) provides a data flow example, showing how the input and output signals must behave.

Note about the *I_pixel* 8 bits format. In python, each image pixel is generally represented by real number "pixelValue in the [0,1] interval. For the hardware implementation, they are converted in the following 8 bits format: 0,XXXXXXXX or XXXXXXXX $(\times 2^{-8})$. This representation is obtained through the following equation:

$$I\_pixel = \text{integerToBinary}\Big(\min\big(\text{round}(256 \times \text{pixelValue}), 255\big)\Big)$$

Note the signal *I_pixel* is saturated at the value 255×2^{-8} in the quantization process. Therefore, the values in the interval $]255 \times 2^{-8}, 1]$ are approximated by 255×2^{-8}.

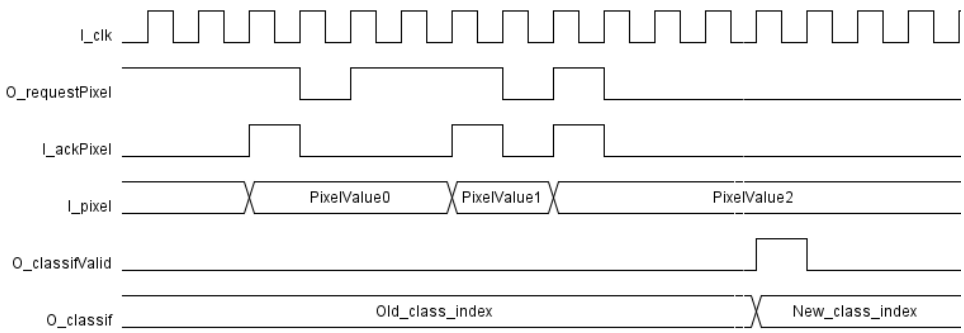

Figure 2: Data flow example for interfacing the FCNN unit.

6 Testbench description

The provided testbench instanciate, as the unit under test, the top FCNN unit. The testbench loads the pixel values from the images in the MNIST test set, and transmit them to the FCNN unit by following the control interfaces described in 5. When a valid result is available from the FCNN, the testbench compares the obtained class indexes to the expected targets. The number of images to test until the simulation stop can be configured through the parameter "C_NUM_CLASSIF_SIMUL", and the number of error can be visualized by monitoring the signal named "S_NUMBER_CLASSIF_-ERRORS".

The files loaded by the testbench are located in the provided "testbench_files" folder. The image files are located in the "PixelData" sub-folder. They are named "PixeldataX.tb", where X is the image indexed in the MNIST test set obtained through Pytorch. For instance, the image indexed 5 corresponds to a testbench file named "PixelData5.tb". In Pytorch, the images in matrix format were "vectorized" row by row, then written in the file. Therefore, each file is composed of 784 rows, each row containing a number ranging from 0 to 255. Figure 3 graphically shows the order in which the pixels, from a 28×28 image, are read and transmitted to the FCNN unit.

Finally, the filepath of each testbench files can be modified through the parameters "C_IMAGE-_FILEPATH" and "C_TARGET_FILEPATH" directly in the testbench.

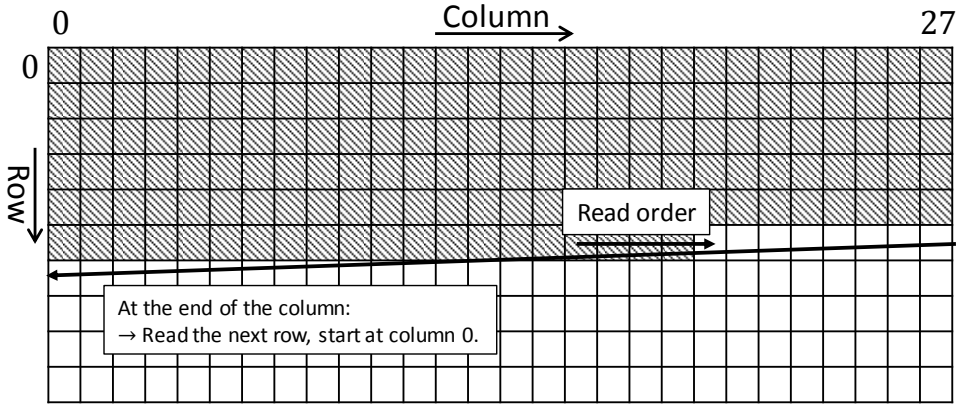

Figure 3: Order in which the pixels are read and transmitted to the FCNN unit.