ECEN 4303/5080 Project: Neural Network Hardware Implementation for Handwritten Digit Classification

Fall 2021

Overall description:
In this course project, we would like to make students understand and learn the fundamental knowledge of integrated circuit design. Through this project, student will learn how to apply the knowledge from this course to practical applications.

There are two project options available this course.
For the first option (pre-defined course project): the student will learn the basic operation and creation of machine learning models and how to implement them using hardware design. The primary goal of this project is to implement a hardware based neural network that will classify input images (e.g., handwritten digits from 0 – 9). Any neural network architecture can be used for this project, though it is recommended that a multilayer perceptron (MLP) is used due to the ease of implementation.

For the other option (self-defined course project): another project option available to students is a research-oriented project, which is purposely for graduate and some undergraduate students who are involved or interested in research topics in hardware design. The deadline exactly follows the same schedule as the default project. **If you want to choose this option, you need to discuss the project topic with me first and then need to obtain my approval.**

Submission requirements for each phase:
- **Initial report (1 - 2 pages):** demonstrate a clear plan for your project including a schedule of personal due dates for each stage of the project.
- **Midterm report (2+ pages):** report whatever you have at the current stage and you will need to demonstrate a clear plan for the remaining works.
- **Final report (6+ pages):** overall neural network classification/inference/forward-path hardware implementation.

1. A PDF report: clearly indicating what you have done (what your plan is) and explain how you implemented this work. Finally, you should demonstrate your experimental results and provide the corresponding explanations of what they are (and why they are).
2. Source code (not applied to Phase I).
3. A PDF report and a separate code zip file (if available) need to be uploaded to Canvas.

Timelines and grades:

	Deadline	Grades	Note
Initial report	9/18/2021	10%	1-2 pages
Midterm report	10/30/2021	20%	2+ pages
Final report	12/7/2021	70%	6+ pages
Overall		100%	

Detailed descriptions:
There are four phases of this project:

- Phase 1: Demonstrate the plan of your project including what you do step by step and what are your target dates for each step.
- Phase 2: Implement a software based neural network to obtain the corresponding weights for future classification or inference.
- Phase 3: Verilog (System Verilog) sub-module implementations and verification (i.e., adders and multipliers).
- Phase 4: Overall neural network classification hardware implementation.

Here is the detailed description of each phase for the first project option (for the second self-defined project, it should follow similar phases, but you will need to define them by yourself):

Phase 1:
This phase asks that each student first collects and reads the related background information and fully understands what this project is about. Based on your understandings, you conclude what you learn from those articles in your initial report and provide a detailed plan (e.g., timelines for different tasks) for this project.

Some useful materials:

1. Simple explanation on Neural Networks: https://www.youtube.com/watch?v=aircAruvnKk
2. Introduction to Deep Learning with Python: https://pythonprogramming.net/introduction-deep-learning-neural-network-pytorch/
3. Deeper explanation on Neural Networks: https://www.coursera.org/lecture/intro-to-deep-learning/multilayer-perceptron-mlp-yy1NV
4. Introduction to artificial Neural Networks: https://www.oreilly.com/library/view/neural-networks-and/9781492037354/ch01.html

Phase 2:
This phase asks each student to study a basic neural network implementation (i.e., multiple-layer perceptron (MLP)). In this phase, the student should understand the basic process of the neural

network training (e.g., forward path and backward path) and their corresponding Python codes. Then, students are required to follow the tutorial to build their own MLP model and train it to obtain the weights and biases of their neural networks. These values need to be stored and will be used for the final hardware implementation. In other words, these weights are expected to be stored in memory and will be used for the classification computations in the hardware neural networks. (Please note that you should carefully store the format of the weights and think about how to convert them into hardware representations).
The default option of the neural network model is a MLP model with three layers (i.e., input, hidden and output layers). The size of the model is 784-50-10. The training and classification dataset is the MNIST database of handwritten digit (http://yann.lecun.com/exdb/mnist/). Some other neural network models are highly encouraged, and the extra bonus will be considered when grading.

Here is the more detailed tutorial for the neural network:
https://colab.research.google.com/drive/1AURbpTaO1qLsH87SYzyO3gael-6FpJ0Y?authuser=0#scrollTo=UBj_IMd8ZJq7

Alternative link:
https://github.com/lab4isc/ECEN4303_Project-Description-Fall-21-/blob/main/mnist_pytorch.ipynb

Phase 3:

This phase starts to focus on the Verilog implementations for sub-modules of neural networks. A neural network-based classifier consists of adders, multipliers, and activation functions. In this phase, each student is required to implement the Verilog modules of adders, multipliers, and the activation function (e.g., Sigmoid function and ReLU function). You should implement all of these modules, simulate them, and validate their functionalities, separately. The default value representation is 16-bit fixed point value (8-bit integer and 8-bit fraction). According to your weights from neural network training (such as maximum values or precision), you can change the value representation to a wider-range value or floating-point value. The associated justification might be added into your report.

In this phase, adders and multipliers are required to be implemented based on the 16-bit ripple-carry and the 16-bit bit-array multiplier as shown in Figure 1 and 2 (also as shown in Chapter 11 of Textbook). Other more advanced adders and multipliers (e.g., carry-lookahead adder or Kogge-Stone adder) are welcome and will be graded with the consideration of extra credit. The simulation results should be included to validate the implemented adder/multiplier working well.

Hardware synthesis report has up to 10% extra credit. To synthesize the designed hardware, you need to log in to the servers. The server connection and the synthesis tutorial can be found in Appendix I&II.

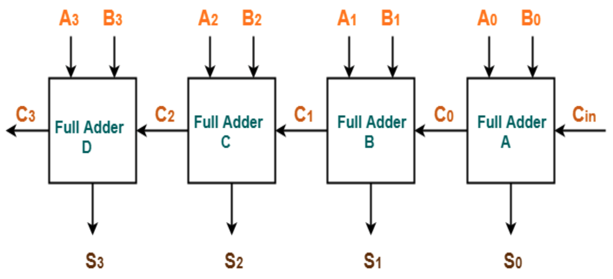


Figure 1: An example of 4-bit ripple-carry adder

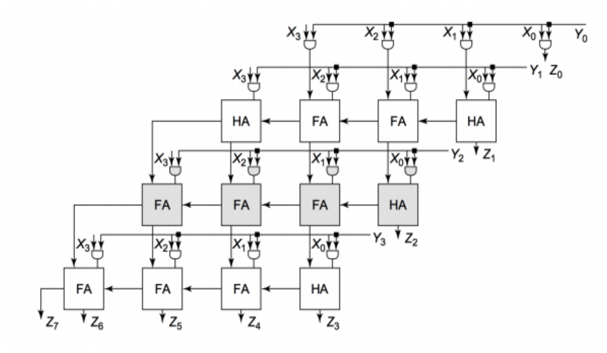


Figure 2: An example of 4-bit bit-array

Phase 4:
Based on Phase 2 and Phase 3, you should implement the Neural Network based classifier from Phase 2 into a Verilog based hardware. The hardware module should be based on the sub-modules of Phase 3 and the final classifier should be a hierarchical module structure. You should validate the functionality of the hardware Neural Network classifier in ModelSim or other hardware simulation tools (several testing images are good enough like 5 or 10 images). To do that, testing images should be the inputs of the model (probably directly read from one txt file). The trained weights from Phase 2 should be stored in the memory.

Report requirement:

- **Initial report:** the report should include two parts. One is that you need to demonstrate what you have read/collected related to this project and how much you have understood about the big picture of this project. The second part is to provide a detailed plan including timelines for different tasks.
- **Midterm report:** the report should clearly demonstrate what you have done and provide the related results. Moreover, the explanation of how and why you obtain those results should be provided. As well as a discussion of what is left to be done in your project and how you are going to proceed.
- **Final report:** your final report should describe the project and its results using the guidelines below. You should provide the background of the existing works which are related to your works. Also, you should provide the formula or the architecture you used in the paper and provides a detailed description of those implementation. You will submit an electronic version of your report to the **class site (i.e., Canvas)**.

Guidelines for the Project Report:
The final report for your project should be written in the style of a **single column with 12 pt font size and a single line space**. It should be **6+ pages** long including references. Your report must include at least the following:

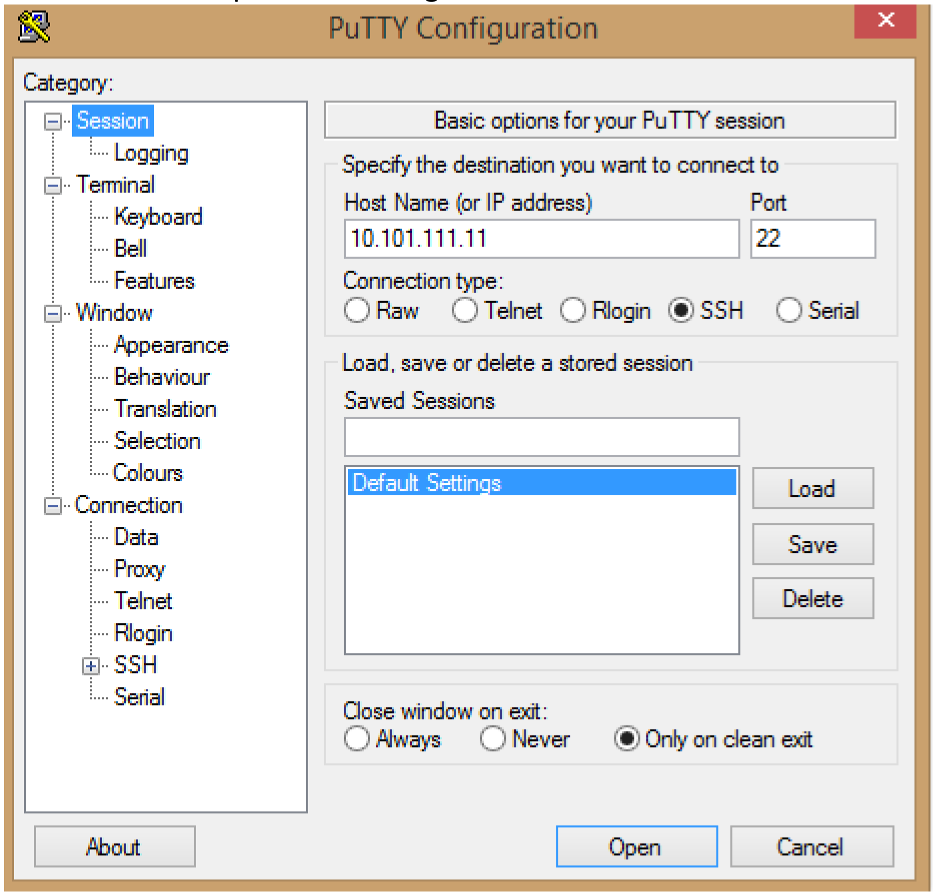
a. The project title with your name(s), student ID number(s), course number, and date.
b. About 150-word abstract clearly summarizing the project like what you did and how you implemented, and a summary of your main results and conclusions.

c. An introduction describes the background of this project.
d. A thorough description of your methodology.
e. An explanation, interpretation, and validation of your results using appropriate statistical and modeling techniques.
f. A conclusion summarizing what you've done and what it means in a broader context.

Grading criteria:
The general grading criteria for the report include:
1. Completeness (40%): based on how much you have finished in your report. (Not applied to the initial report)
2. Clarity (including report format and grammar check) (20%): clearly describe the tasks what you have done and how you implemented.
3. Result demonstration (30%): clearly show the enough results based on your results. The report should clearly demonstrate you understand what you have done in this project and why you obtain the results (maybe desired or non-desired results).
4. Methodology (10%): It is an overall evaluation based on what methods you use in the MLP implementation. It includes the methods that are used for the implementations of adders, multipliers, activation functions, value representations, memory implementations, etc.

Appendix I. Server Connection (e.g., server IP is 10.101.111.11 with a username "xxx"):

For windows users, you can download WinSCP to transfer files and use Putty (like a terminal) to connect the server.
In WinSCP and Putty, put the IP address with port number 22 like the following figure. Then you can input your username and password to log in the server.



Specify the destination you want to connect to
Host Name (or IP address): 10.101.111.11
Port: 22
Connection type: Raw Telnet Rlogin [✓ SSH] Serial

Load, save or delete a stored session
Saved Sessions
Default Settings

Close window on exit: Always Never [✓ Only on clean exit]

For Linux users, you can use the following command to connect the server:

```
ssh xxx@10.101.111.11
```

SCP command can help you transfer data from one system to another system. For more details, you can google search for how to use SCP to transfer data.

Appendix II. Synthesis tutorial on ECEN server:

1. Setup the environment on the OSU ECEN server

```
$ Eda–tools
$ source .synopsys_dc.setup
```

2. Download synth.zip, put in /YOUR_PATH /

3. Finish you Verilog codes and put them in /YOUR_PATH /synth/hdl/

4. Set the my_verilog_files in /YOUR_PATH /synth/scripts/synth.tcl

```
set my_verilog_files [list xxx1.v xxx2.v …]
```

where xxx1.v, xxx2.v … are your Verilog files.

5. Set toplevel in /YOUR_PATH /synth/scripts/synth.tcl

```
set my_toplevel xxx
```

where xxx is your top Verilog file name.

6. Run synthesis

```
$ dc_shell-xg-t -64bit -f scripts/synth.tcl | tee synth.out
```