UNIVERSITY OF CALIFORNIA IRVINE

EECS 298: System-on-Chip Design (Fall 2023)

Project: SoC-based MatMul Accelerator and DNN Accelerator

Due: December 15, 2023, 11:59 PM PST

In this lab, you will work in teams to build two SoC-based accelerators, one for matrix multiplication (PART I) and one for DNN inference. In both parts, you need to build an accelerator IP (or IPs) in Vitis HLS, connect it with other IPs into an accelerator platform and generate bitstream in Vivado, program the host code for KR 260 board with PYNQ library/runtime, and evaluate the performance of your hardware accelerator. You need to submit your code and a project report (in PDF, see "Submission Requirements" section below) to Canvas.

Different from the last lab, in this project, after you create your designs, you will need to actually program your design (bitstream) to FPGA (PL) in the KR 260 board, and run the accelerator (with PYNQ package). The end-to-end performance of your accelerators will be evaluated. Please refer to this page to set up your KR 260 board.

PART I: Matrix Multiplication *(20 pts)*

In this part, you will take the two versions of matrix multiplication accelerators you created from previous lab, and test their performance on the KR 260 board. Compare their performance against the NumPy's matrix multiplication (link) performance.

Requirements

- Create the PYNQ host code to call your accelerators, verify their correctness, and measure their execution time.
- Generate random matrices as test cases, and use NumPy `matmul` results as the golden reference.
- Report the execution time for three versions of matrix multiplication (NumPy version, plain accelerator, optimized accelerator). You may call the accelerator multiple times and measure the total execution time, and then calculate the average execution time.
- Report the hardware resource utilization (FF, LUT, BRAM, DSP) and frequency for your accelerators.
- Grading: correctness (10 pts), report quality (10 pts).

PART II: DNN-based Hand-Written Digit Recognition *(100 pts)*

In this part, you will create an SoC design (FPGA hardware + CPU software) for accelerating the inference of a deep neural network (DNN). The DNN we are accelerating is a modified version of LeNet (Figure 1), which is designed for recognizing hand-written digits (MNIST dataset). We have provided you with a working C/C++ implementation (link), whose performance will be the baseline of your design. You can decide which layers/parts of the DNN to be accelerated with FPGA (the rest will be running on CPU, implemented in Python or C/C++). However, it is **required** that you accelerate **at least two convolution or fully connected layers** in the DNN with the FPGA.

Requirements and Tips

- You need to accelerate **at least two convolution or fully connected layers** in the DNN with the FPGA (PL), and the rest of the digit recognition system will be running on CPU, implemented in Python or C/C++. You can decide which parts of the DNN to be accelerated with FPGA.

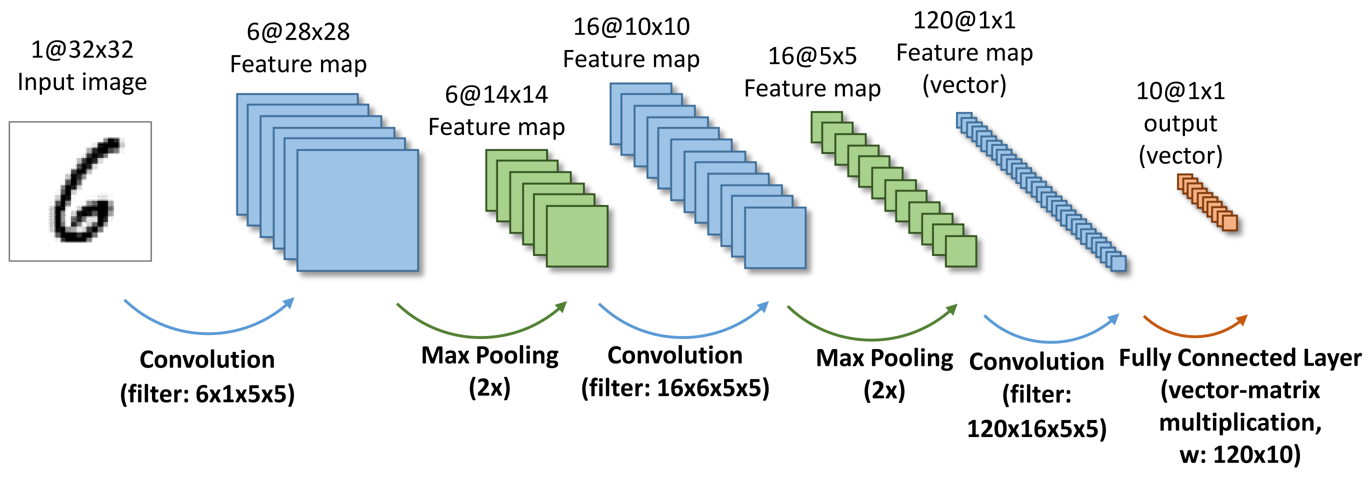

Figure 1: DNN Architecture

- Create a PYNQ host code to call your accelerator, evaluate the test accuracy, and measure the execution time.
- The final accuracy tested with all the 10,000 test samples should be at least **95**%.
- Feel free to modify or completely re-write the provided C/C++ reference code. In fact, you probably need a very different code for HLS so that you can get good performance.
- Apply multiple optimizations to achieve good performance. For example, layer fusion, pipelining, quantization (use low-bitwidth fixed-point formats), etc.

Grading

- PART I: complete the evaluation and report all the required results.
- PART II: Your accelerator should run faster than the CPU baseline (provided C/C++ code running on the ARM CPU), and the testing accuracy should be higher than 95%, otherwise you will lose points.
- PART II: Top 3 teams with fastest implementations will get bonus points to final grades of the course (1st place: 1.5 pts; 2nd place: 1.0 pt; 3rd place: 0.5 pt).
- Grading: completeness (40 pts), accuracy (10 pts), optimizations (30 pts), report quality (20 pts).

Submission Requirements

You need to submit (1) all the code you used for both PART I and PART II, and (2) a project report in ACM format (use "sigconf" template), in PDF. In your report, beside all the performance and hardware resource utilization results, please also provide a detailed introduction and explanation to your DNN accelerator design, including each optimization techniques you used, and discuss the evaluation results.

References

- KR 260 Setup guide
- Lecture recordings on usage of Vitis HLS and Vivado on remote lab and EECS instructional servers. (recording 1, recording 2, EECS221 recording)
- AMD UG1399: Vitis HLS User Guide version 2023.1 (PDF)
- Parallel Programming for FPGAs. [PDF]