ECE5997 Hardware Accelerator Design and Automation, Fall 2021
School of Electrical Computer Engineering, Cornell University

---

Lab 3: Digit Recognition System (Part 2)

Due Wednesday, December 1, 2021, 11:59pm
Late submission: 4% penalty per day; cannot be late by more than 6 days

---

1 Introduction

In the tutorial on *Software-Hardware Codesign with CORDIC*, you saw how we modified the CORDIC design from Lab 1 to implement the synthesized RTL on the Xilinx Zynq field-programmable system-on-chip (SoC). In this lab, you will be responsible for implementing your `digitrec` design from Lab 2 on the same device. Furthermore, you will apply **loop pipelining** to optimize the design and measure the execution time of the software-only, baseline-FPGA, and optimized-FPGA implementations of `digitrec`. Finally, you will use the information found in Vivado HLS's generated reports to estimate the runtime of the FPGA designs and compare your numbers with measurements made on the ZedBoard.

2 Materials

You are given a zip file named *lab*3*.zip* on *ecelinux* under */classes/ece*5997*/labs*, which contains the following directories:

- **ecelinux**: contains an **incomplete** C++ project for you to build the `digitrec` HLS design and synthesize it to a hardware module. This code should be completed on `ecelinux`.
- **zedboard**: contains *symbolic links* to the files in the **ecelinux** directory required for software execution of `digitrec` on CPU. Also contains an **incomplete** C++ project to build the host program for invoking the `digitrec` module on FPGA.

Note that a *symbolic link* is a special file which points to another file — accessing the link basically accesses the original file. The files `digitrec.h`, `digitrec.cpp`, and `digitrec_test.cpp` in the **zedboard** directory point to their counterparts in the **ecelinux** directory. This means you only have to complete the `digitrec` design in the **ecelinux** directory and the changes will be automatically propagated to the **zedboard** directory. Make sure to copy both directories to the ZedBoard or else the link will be broken.

3 Design Overview

You will again use the k-nearest-neighbors (k-NN) algorithm for digit recognition (consult the Lab 2 document for details on the k-NN algorithm). **Because the focus in this lab is the hardware implementation, the value of `k` will be fixed to 3.** You will implement and evaluate the performance for three designs:

- A **baseline** `digitrec` design that does not use any HLS optimization directives (`vivado_hls run_base.tcl`).
- An **unrolled** `digitrec` design which is similar to what you did in Lab 2 where unrolling and array partitioning are applied (`vivado_hls run_unroll.tcl`).
- A **pipelined** `digitrec` design which applies loop pipelining in addition to the previous optimizations (`vivado_hls run_pipeline.tcl`).

You will use the information from the Vivado HLS synthesis report to **estimate** the performance of your hardware design, and verify that the physical hardware achieves a performance close to the estimate. You will also measure the performance of the software execution on both the ZedBoard and **ecelinux** for comparison. Timers identical to those from the CORDIC tutorial will be used.

4 Guidelines and Hints

4.1 Coding and Debugging

Your first task is to complete the software-only implementation on **ecelinux**. Similar to Lab 2, the main body of the `digitrec` function is finished, and you only need to complete `update_knn` and `vote_knn`. But this time you are required to **only use constant-bound loops in the `update_knn` function**.

To complete the hardware design, you will also need to fill in `dut` with code to communicate with the FIFO channels. The `digitrec_test.cpp` testbench is responsible for running the software-only implementation of `digitrec`. Timers have already been added to the csim. If you add print statements to debug your code, make sure to remove them before doing runtime measurement.

Your next task is to complete the FPGA implementation on a ZedBoard. The process of generating the bitstream, logging onto a ZedBoard, and programming the ZedBoard using the bitstream is identical to what we showed in the CORDIC tutorial. Be aware that **it can take 20-30 minutes to generate the bitstream**.

You are also responsible for completing the `host` program. We have taken care of reading the input and reference data sets from file and activating the timers. Make sure to use **batch** data transfer in your code to minimize communication overhead.

4.2 Hardware Design Optimization

Use the provided scripts `run_base.tcl`, `run_unroll.tcl` `run_pipeline.tcl`. Your source code will be identical between these designs. Note that we have already added the necessary HLS optimization directives in each of these Tcl scripts.

After synthesizing the unrolled design, you should check that the latency is reduced by around `10x` in the synthesis report. In pipelined design, we are pipelining the outer loop (labeled as `L1800`, which iterates 1800 times). **So please verify after pipelining that the `L1800` loop is indeed pipelined to an II of 1 in the report.**

If you are interested to learning more about the pipelining directive, check out the following reference:

- Vivado Design Suite User Guide, High-Level Synthesis, UG902 (v2019.1) [1]
  - *set_directive_pipeline* p.453

  You can find examples of code snippets using the pipeline pragma throughput the user guide.

4.3 Report

- Please write your report in a **single-column and single-space format with a 10pt font size. Page limit is 2. Please include your names and NetIDs on the report.**
- There should be a section describing how you implemented the `digitrec` system. Explain how you communicated data between the processor and FPGA, as well as what preprocessing was necessary to put the data in the right format.
- There should be a section discussing the effects of each design (baseline, unrolled, and pipelined) on the synthesized hardware. This section should contain a table which reports the latency and resource usage of each design. Compare these numbers discuss them using your understanding of how the unrolling and pipelining optimizations work.
- There should be a section reporting the measured performance of each `digitrec` system implementation. Specifically, **this section should contain a table which lists the observed runtime for each implementation, including ecelinux-software, zedboard-software (i.e., ARM), zedboard-fpga-baseline, zedboard-fpga-unrolled, and zedboard-fpga-pipeline**. This table should also have a column that reports the runtime speedup normalized against zedboard-software.

  In addition, for the FPGA implementations, please use the synthesis report to estimate execution time in hardware (note that the FPGA clock is operating at 100MHz in this particular setup). **Please discuss how you obtained your estimations, and compare your theoretical and observed results and explain any discrepancies.**
- All of the figures and tables should have captions. These captions should do their best to explain the figure (explain axis, units, etc.). Ideally you can understand the report just by looking at the figures and captions. But please avoid just putting some results and never saying anything about them.
- The report should only show screenshots from the tool when they demonstrate some significant idea. If you do use screenshots, make sure they are readable (e.g., not blurry). In general, you are expected to create your own figures. While more time consuming, it allows you to show the exact results, figures, and ideas you wish to present.

5 Deliverables

Please submit your lab on CMS. You are expected to submit your report and your code and scripts (and only these files, not the project files generated by the tool) in a zipped file named **digitrec2.zip** that contains the following contents:

- **report.pdf**: the project report in pdf format.
- The folders **ecelinux**, **zedboard**, and **bonus**. These should contain the completed source files for the software-only, FPGA, and optimized software-only implementations of the `digitrec` design. Make sure the design can be built using the Makefile and scripts in the folders. Please run `make clean` to remove all the generated output files.

6 Acknowledgement

The baseline FPGA+Linux setup used in tutorial is based on the Xillinux distribution provided by Xillybus Ltd. (http://xillybus.com/xillinux)

References

[1] Xilinx Inc., *Vivado Design Suite User Guide: High-Level Synthesis UG902 (v2019.1)*, Available at `https://www.xilinx.com/support/documentation/sw_manuals/xilinx2019_1/ug902-vivado-high-level-synthesis.pdf`