ENSC 453/894 Lab Assignment 3

ENSC 453/894 is a programming-intensive course. You will work on five lab assignments on CPU, FPGA, and GPU programming, which will help you gradually master the highly demanded programming skills in the information and communications technology (ICT) industry sector. You need to form a 2-people team to work on lab 2 to 5 and will be evaluated per team. On our course website https://canvas.sfu.ca/courses/86638/pages/lab-logistics, some general grading logistics have been posted. However, the detailed grading scheme for each lab (which often reveals the answers or clues to the answers) will NOT be released until your lab has been graded: imagine you are in a real interview, nobody will tell you what the detailed grading scheme is.

Lab Assignment 3 (10 marks):

In lab assignment 3, your task is to implement the same *kernel_gemm* function from lab assignment 1 and 2 on the Alveo U50 FPGA (same as Lecture 6 and 7) using Vitis HLS C/C++.

Note your lab 3 implementation will not run on the actual FPGA. We just do the C simulation and co-simulation (RTL simulation) to verify the algorithm and hardware correctness, and measure the performance from Vitis HLS report, including the estimated total number of clock cycles and clock frequency, which gives you the estimated execution time.

Specifically, you will apply various HLS optimizations:

- [2 marks] Explicitly convert the code into the load-compute-store functions and apply the buffering technique. Note the original matrix size of 4096*4096 is too big and is stored in off-chip DRAM (i.e., HBM on Alveo U50). To make sure your computation is done using on-chip data, you have to explicitly use a smaller array to **buffer** a tile of the matrix onchip and perform all the computation on the buffer. Specifically, you will have three functions: 1) **load** a tile of the original matrix from off-chip DRAM into the on-chip buffer (i.e., your smaller array); 2) **compute** the matrix multiplication on the buffer all on-chip; 3) **store** the output buffer (for C matrix) to the off-chip DRAM. You will have a similar outer loop nest to your lab 2 to iteratively do the load-compute-store for all tiles.
- [1 mark] Pipeline the data transfer in the load and store functions. Please write your own loop with appropriate HLS pragmas and don't use a library call such as *memcpy*. Also note loop parallelization doesn't apply here in load/store functions, since each load/store is connected with only one off-chip DRAM port and there is no parallelism in the hardware.
- [3 marks] Optimize the compute function using appropriate pipeline and parallelization techniques. Here you can apply any combination of loop pipelining, loop unrolling, and other loop transformations if needed, with or without HLS pragmas. Corresponding array partitioning techniques would be needed for the A, B, and C **matrix buffers** (note array partitioning should be done for the on-chip buffers instead of the original matrix in the offchip DRAM) to enable the pipelining and parallelization.

- [1 mark] Write your own testbench and tcl script ***"run_hls.tcl"*** to run c-sim and co-sim to make sure your FPGA code get the same results as your CPU version. Run Vitis HLS to synthesize the design and report the performance and resource usage.
    - For c-sim and performance ranking, please use input matrix size of 4096*4096, and your own best configuration of buffer size.
    - For co-sim, please use input matrix size of 32*32 and buffer size of 16*16 to make the execution time manageable.
    - When you submit lab 3, please include two separate folders in lab3.zip for the above two different sizes with slightly different tcl scripts.
- Note you don't need the Ping-Pong buffer and memory coalescing optimizations, which will be left to lab assignment 4 and will NOT be used in lab 3 performance ranking.

Please note if your code doesn't pass c-sim or doesn't pass HLS synthesis, then you can get at most 5 marks. If your code passes c-sim and HLS synthesis, but doesn't pass co-sim, then you can get at most 6 marks. If your code passes all the c-sim, HLS synthesis, and co-sim, then you are eligible to participate in the ranking and get all 10 marks based on the detailed grading scheme.

Your goal of lab 3 is to achieve the best performance (estimated execution time = number of cycles / frequency, which are available in HLS report), while using no more than 60% of the Alveo U50 FPGA resource summarized in the table below.

Resource	RAM_18K	DSP	FF	LUT	URAM
Available	2,688	5,952	1,743,360	871,680	640

Typically, we set the target clock frequency to 300MHz. A frequency higher than 300MHz is not useful since the FPGA platform only runs at a maximum of 300MHz (which you will try out in lab 4 and project final report). You have to double check that the HLS estimated clock period is no greater than your target clock period (i.e., its estimated clock frequency is no lower than your target clock frequency), which is available in HLS report; otherwise, it means you cannot achieve target clock frequency and you may set it to a lower frequency or use less resource, which of course will slow down your execution time. **For designs where HLS estimated clock frequency is lower than your final target clock frequency (your final target clock frequency may be lower than 300MHz, which is ok), they are NOT eligible for the performance ranking.**

For the total amount of resource utilization, please make sure your FPGA kernel uses no more than 60% of the Alveo U50 FPGA resource, for each of the RAM_18K, DSP, FF, LUT, and URAM. Otherwise, you are NOT eligible for the performance ranking. Take DSP usage for example, you can use at most 5,952 * 60% = 3,751 DSPs. The reason is that usually the FPGA platform also uses some resource, and a high resource usage will lead to final bitstream generation failure: typically, it's very hard to achieve more than 70% resource utilization (for your FPGA kernel) on the datacenter FPGAs. You will build the final FPGA bitstream in lab 4 and project final report, which takes hours for a single build. By then, some of you may find even 60% resource utilization could fail the bitstream generation, if your FPGA kernel is not carefully designed.

You are also required to write a simple lab report to report the performance speedup of each optimization that you applied and its corresponding resource utilization. In your final code, please submit: 1) one sub-folder named as "c-sim" for the c-sim and HLS synthesis configuration that achieves the best performance; 2) the other sub-folder named as "co-sim" for the co-sim configuration; 3) your lab report. **TA will synthesize all submitted lab 3 codes on the ENSCRLA lab servers using Vitis HLS 2020.2 and grade based on the performance of your code (with 4096*4096 matrix size): the smaller the estimated execution time your FPGA kernel design takes, the higher marks you will get. 3 marks (out of total 10 marks) are reserved for performance competition:**

For performance ranking, the competition marks are given as below:

- Top 2 groups: get 3 marks
- Top 3-6 groups: get 2.5 marks
- Top 7-10 groups: get 2 marks
- Top 11-15 groups: get 1.5 mark
- Top 16-20 groups: get 1 mark
- Top 21-25 groups: get 0.5 marks
- Top 26-30 groups: get 0 mark

Note: For grading logistics and remote machine access, please refer to the course website. If you have any questions, please post them on Canvas discussion board. In addition to the actual coding for implementation, you need to provide good commenting and coding styles, which count in your lab assignment marking (negative marking).

Assignment Submission:

Your lab assignment 3 will be submitted electronically through Canvas. You will need to submit a single ***YOUR_LAB_GROUP.zip*** file (your lab group name is available on Canvas, e.g., LA03_05). This format makes it easy for TA to do auto-grading. Please double check the format correctness. Failure to comply with this format will result in a **ZERO** point for this assignment. To zip your files in Linux, Go to your lab3 directory

1. *clean your files in each sub-folder "c-sim" and "co-sim"*
2. *cd .. //go one level up*
3. *mv lab3* ***YOUR_LAB_GROUP*** *//rename your lab3 folder using your lab group*
4. *4. zip -r YOUR_LAB_GROUP.zip YOUR_LAB_GROUP //zip all your lab3 files into a single .zip*

Submission Deadline:

Your lab assignment 3 is due at **11:59:59pm on Friday, Feb 13, 2026**. You need to meet the deadline: every 10 minutes late submission, you lose 1 mark; that is, 100 minutes late, you will get 0 mark on this lab.

Lab Demonstration:

You will have to demo your lab assignment 3 to the TA in the lab sessions. **The demo has to be done on the ENSC-RLA lab servers**. Only code from your Canvas submission is allowed in the lab demo. Each student group has around 10 minutes to explain your code to the TA. If you fail to do the demo (without a medical note), or if it is determined that you do not understand the code being evaluated, you will be awarded zero on lab assignment 3 and considered as cheating. Also please show up on the demo day on time (TA will send out your scheduled time), otherwise you will lose 0.5 mark.

Note on ENSC-RLA servers:

ENSC-RLA lab servers are Linux virtual machines deployed on FAS-RLA servers; due to various complications, it’s challenging for IT to set up the FPGA synthesis environment on the native FASRLA servers. All of your FPGA synthesis will be done on ENSC-RLA servers, as we have a limited number of MMC servers that host the actual FPGAs and GPUs. TA will release a tutorial video on how to access ENSC-RLA lab servers and run the FPGA synthesis.