ELEC 522, Advanced VLSI
Rice University, Fall 2022
4 October 2022
Due: Thursday 20 October 2022 at 11:59pm
Independent work is expected.

Project 3: Using Vitis HLS to Implement Matrix Multiplication

Goals:
In project 2, we have looked at how to build a VLSI signal processing system by using the graphic-based Xilinx Model Composer tool. However, text-based high-level synthesis (HLS) tools are becoming more and more popular due to their capability of rapid system prototyping. In this project, we will use the Xilinx Vitis HLS tool to design a matrix multiplication system which could be easily targeted to ASIC/FPGA. Based on Project 2 we have become familiar with the design flow and will use methods to optimize the architecture and boost the throughput by exploring the potential parallelism of the system.

Project procedures:

1. Review the concepts from Project 2 and the VitisHLS tutorials and Model Composer black-box integration. (Vitis HLS labs for ZedBoard are posted on Canvas. Please see: ELEC 522 Files / CAD_Tool_Examples / Xilinx_Vitis_HLS_Labs)
2. Use the Vitis HLS tool to design a matrix multiplication system. Write C code for matrix multiplication. Generate HDL code with the appropriate architecture constraints. Verify your C code and HDL code to make sure that they function correctly. Check the scheduling result. Adjust architecture constraint parameters to optimize your architecture (you might need to slightly modify your C code to adapt it to your architecture optimization). You can build this from the sample Vitis HLS matrix multiplication code in the tutorials.
3. Repeat step 3 until you get a good balance between system throughput and hardware resource costs, by changing different combinations of optimization methods. Make your choice and generate HDL code for your final system. You should try at least 2 optimization step iterations, that is, the **original plus 2**. If there is no improvement over the original, please explain in the documentation.
4. Your system should preserve the data streaming aspects of the systolic array structure as much as possible. That is, you should maintain multiple I/O ports that allow data streaming from the Model Composer “host” model. We are trying to avoid an initial block copy of the matrix from the “host” to one single memory in the FPGA. The reason we would avoid a block copy is that it does not allow for the overlap/pipelining of communication and computation as in a systolic array.

5. Import your Vitis HLS code into a Model Composer model by using the Vitis HLS Model Composer block. Use a testbench to test your system. (You could modify your Model Composer model and testbench from the Model Composer Project 2 on Matrix Multiplication.) There is the Vitis HLS block in Model Composer in the "HDL/User-Defined Functions" palette. There are some extra tutorials at: https://www.youtube.com/watch?v=25oO4LSgA2U (Note that we have the built-in version of HLS not the stand-alone version.) Also see: https://www.youtube.com/watch?v=HumzeOylfyE In addition, please look at the ug1399-vitis-hls.pdf on Canvas in ELEC 522/ Files / CAD_Tool_Documentation / Vitis_HLS

Test and verification:

1. First write C testbench code to verify the function of your code. Use Vitis HLS simulation tools to verify at this level (Refer to Vitis HLS tutorials).
2. In Model Composer, wrap your HDL code from Vitis HLS with a "Vitis HLS block" and then put it into a Model Composer model to test the complete matrix multiplication as in Project 2.
3. With Model Composer, generate for ZedBoard to do Hardware in the loop Co-Simulation to verify performance of your Vitis HLS code on the ZedBoard in the lab.

Requirements:

1. The system should handle 4x4 matrix multiplication. You could assume simple data types, such as 16-bit two's complement fixed-point numbers as inputs and you can constrain the output also to 16-bit.
2. You could use **up to 16 multipliers**. You can also use less than 16 multipliers if you think that there is a "sweet point" for the latency-area tradeoff. You need to describe your design considerations when deciding how many multipliers to use and explain why you chose that number. Remember that there is no standard answer for this project.
3. The data should stream into and out of your Vitis HLS module and not be block copied if possible. This may involve some additional work on the I/O interface. Please look at the Vitis HLS tutorials for info on PIPELINE and INTERFACE on github at: https://github.com/Xilinx/Vitis-HLS-Introductory-Examples . There is also a FIFO capability.
4. Try to maximize the throughput as much as possible as you can. The methods include but are not limited to maximizing the clock frequency, optimizing loop performance, parallel processing…
5. Generate from Model Composer first for HDL Netlist to get timing and resource estimate and report the results (before you generate for ZedBoard Co-Sim.) Compare the resources used (DSP48, LUT, FF registers) between project 2 and 3, the maximum clock frequency between projects 2 and 3, and the number of clock cycle delay (latency) and matrix multiplication throughput between projects 2 and 3.

6. Support continuous matrix multiplication, that is, enable your design to stream multiple problems through the array. Please indicate how much delay is needed between each problem.
7. This should be a multi-mode system that supports both matrix-matrix multiplication and matrix-vector multiplication as in Project 2. Your design should have good reconfigurability and scalability.

Documentation:
In your documentation, please describe how you changed your C code and the architecture constraints to optimize your design. Describe your final choice of the constraint configurations. Explain the advantages of your final design over the other trial designs that you made. Explain the timing scheduling results from the Vivado reports. If you think you have done something clever that makes verifying your C code and HDL code faster and easier, please also describe how you do it in your report. Give the timing and resource information from the synthesis and place and route implementation report. Calculate maximum throughput of your system. Compare the results of this project with those from the Model Composer matrix multiplication project 2 and discuss the differences between the two design flows that possibly cause the different results.

Please include your “C” code files, your Model Composer .slx model file and any other testbench files needed to run and verify your design. Please also include a file which includes your simulation results, either as a text file or a waveform. Please include step by step instructions on building and simulating you design so that the graders can test your design. Also include a screen capture of the results of your Hardware in the loop Co-Simulation testing results.