As a 1 credit independent study during my Graduate concurrent semester in the Fall 2023 term, I worked with Dr. Duwe to do some research on the open-source tool OpenFPGA. This tool is unique in that it allows you to define FPGA architectures using XML files with tools such as VPR, VTR, and FPGA-Spice, to generate and simulate Verilog and SPICE netlists for varying FPGA architectures. The main goal of OpenFPGA is to combine multiple different FPGA architecture tools in one package to allow for shorter development time, due to the nature of high development times to design FPGAs and following netlist generation. Due to this, in their papers, they have reported that designs have become more streamlined due to high development times. By utilizing this tool, it should be easier for both commercial and open-source developers to create FPGA fabric netlists and generate place and route outputs with a faster turn around time.

For this research, I began by reading over the background research papers surrounding OpenFPGA and reading through their online documentation. I was able to learn more about what made up an FPGA fabric, and what different design considerations an engineer must make when determining the right FPGA architecture. This was extremely beneficial to me, since I have worked on developing RTL for FPGA’s for multiple years now, and it helped to give me a better understanding of place and route techniques after taking a digital VLSI course and my open-source digital ASIC fabrication senior design project.

I also spent some time working with the OpenFPGA git repository and tools directly. For this, I learned how to install the proper dependencies and build the git repo, alongside utilizing a Docker instance to run precompiled binaries for the toolflow. I ran multiple testbench configurations, and learned that OpenFPGA has two different types of testbench procedures which are fully automated. A user can provide functional Verilog benchmarks, which can be either reconfigured onto the FPGA in an initial configure mode or compared with a direct mapped netlist inside the FPGA. The testbench which requires flashing a bitstream takes much longer, but is useful to verify the FPGA fabric can generate a proper bitstream and generate the same matching output to a target functional Verilog benchmark. To run a testbench, I had to also generate an FPGA Verilog fabric netlist, in which multiple different XML files can be referenced to alter the FPGA architecture description.

Finally, I learned how to alter the cells referenced in the FPGA XML architecture description files. I ran through two documented processes for this, including using a custom functional Verilog module and an open-source cell from the same Skywater 130nm PDK that I utilized for my senior design project. Both of these Verilog designs can be referenced on the path directly in the XML files, and will then be able to be included for the proper module name when the total FPGA Verilog fabric netlist is generated through the OpenFPGA toolflow. This is extremely beneficial, since you can then use custom cells such as the open-source Skywater 130nm PDK, which can be carried through after synthesis for place and route generation and fabrication.