**Overview**

In the Fall 2023 semester, I took CPRE 558, Real Time Operating Systems, at Iowa State University. As a final project, me and a partner decided to create a hardware implementation for a periodic RMS scheduler using Verilog RTL and an FPGA implementation. The goal of this project was to design a hardware architecture that could achieve faster and more efficient context switching for tasks in comparison to a software scheduler that was intended for the same task set.

Our design was centered around the 5 stage MIPS processor that we designed in a previous computer architecture course, where we could use existing code such as a register file or program counter as a baseline. The central idea was that we would have a register file and program counter for each task, and have a control module which would switch between the five tasks based on if they were complete, missed their deadline, or were ready to be used. In the RMS protocol, the task with the lowest period has the highest priority, and is able to preempt currently running tasks when the period refreshes, meaning we had to enable this context switching in our control module. We also included a standard up counter to monitor the current time, as a reference for when each task should become active again. The main benefit of this design was that context switching could occur in a single cycle, which can be much more efficient than a software implementation, and could still be reconfigurable at compile time with the use of an FPGA instead of an ASIC.

Each designed module was tested functionally individually, and we followed integration testing with Verilog testbenches also. When that was complete, we integrated our design onto my Basys 3 FPGA Development board, which has a Xilinx Artix-7 FPGA board. Using the OLED display interface I designed in Summer 2023, we were able to create a display monitor to show the state of each task, including its stored register value, computation time, and if it was complete. Alongisde this, we displayed the current time, which task was currently active, and if the time was at the least common multiple of all task periods, meaning the entire schedule would reset. This design proved the feasibility of a hardware based task scheduler on a reconfigurable FPGA, and further work could be done in comparison to software schedulers in terms of context switching, power consumption, area requirements, timing, and design time.