Introduction: What was your inspiration or initial idea for this project? The introduction should also give a brief description of the goals of the project.

In the high frequency trading space, FPGAs are often used to accelerate time-sensitive computations to beat the other players, this is a case of high-performance acceleration of mathematical tasks, which is one of the main applications of FPGAs. In class, we looked at how we can create boolean logics and finite state machines which can be used to accelerate these common tasks on FPGAs. We wanted to expose ourselves the process of optimizing code on FPGAs and get exposure in writing low-level code to program these devices. Furthermore, an exercise in visualizing these computations interested us, so we decided to visualize a live-data feed and do a computation (find moving-average) on the data feed input.

Motivation & Justification: Why are you motivated to work on this project, and what problem is it trying to solve? For those of you utilizing the FPGA for HW acceleration, this section should include justification for your design by quantifying the performance of a non-FPGA-accelerated solution and identifying/discussing the potential improvement of your design. For RPi-based projects, this section should motivate why addressing this problem on a constrained HW platform is interesting / cool / important, and quantify the challenges in doing so (e.g., memory requirements, real-time performance needs).

Moving average calculation involves iterations of addition, division, and manipulation of variables on the memory. By using the bit representation of integers and computations & a custom bitstream to compute moving averages on the programmable logic, we hoped to speed up the process of calculating moving averages.

We used the Raspberry Pi to send the numbers to the FPGA, then the moving average was computed in the programmable logic using the new number & the values that were stored previously. Then, the output was sent from the FPGA to Raspberry Pi via the IO pins of the board.

For testing hardware acceleration achieved with an FPGA, we compared the time taken to run one iteration of moving averages using the programmable logic vs. using the CPU on the FPGA. When using the custom bitstream, computation took around 1×10^{-5} seconds, whereas it took about 4×10^{-5} seconds to run the same algorithm using the CPU.

Experimental Set-Up: What hardware platform are you using to accomplish your goals? Here is where you should include any circuit diagrams, design diagrams, or other visual representations (e.g., pictures of your lab set-up) of your demo configuration.

Raspberry Pi and FPGAs, connected through the GPIO pins. We use the Pi to stream data to the FPGA, and to process the output to create a visualization in real time.

Methods & Resources: What software tools, related projects, and other resources are you utilizing to complete your project? Here, you should include links and brief descriptions of any existing software libraries or open-source projects that either informed your development or that you integrated with your project.

Vivado to implement our hardware design written in verilog

Python to create Pi-side visualization and data streaming server

Preliminary Testing & Results: Provide a brief description of your experience of the project so far. What roadblocks and adjustments have you made? This section should also provide any and all preliminary results available (e.g., performance of software across platforms, the memory requirements and performance of your current system, more detail on the overall project design).

We have experimented with Pynq overlays which we explored in class to implement the moving average, and created a successful averaging mechanism using boolean operation, but we found that it was not possible to store values in a register using these overlays, which was our motivation to writing the code in Verilog instead. We have struggled to identify the GPIO pin numbers to establish our constraints due to fragmented documentation, but we were able to establish a stable connection for both input and output between the Pi and FPGA.

Some problems we encountered were in finding ways to sync the clock between the two devices. First, we were using the clock signal created by the FPGA, but we have then transitioned to using a clock signal generated by the Pi.

Demo Day To-Do List: What remaining goals and tasks are on your list that you aim to complete in time for our practice demos (Dec. 13) and final demo day (Dec. 15).

By the demo day, we want to have the moving average logic working on verilog. Right now, we are not able to reliably save and retrieve values from the register, so we are trying to figure out how to do this. After we figure this out, we should be able to get our code working.

I am hoping we can do some sort of time comparison vs. the Pi, but I am not sure how much speed improvement we are currently getting, as our clock signal is generated by the Pi. We might do a theoretical calculation, with the assumption of having a fast data feed.