Project 1 Report

George Conwell, 661995055
ECSE 6680, Section 1

MATLAB:

MATLAB was used to both calculate the tap coefficients of the intended lowpass filter, and take the quantized tap coefficients from the FIR filter and create a frequency response. *Figure 1* is the frequency response according to the requirements of a passband at 0.2 normalized frequency, a stopband of 0.23 normalized frequency, and a stopband attenuation of 80 dB. The function used, designfilt(), will automatically calculate the minimum-order design when given the aforementioned requirements. After running the MATLAB code, designfilt() outputted 175 coefficient values (Coefficients.txt), meaning a 175 tap lowpass filter is required to meet the design requirements.

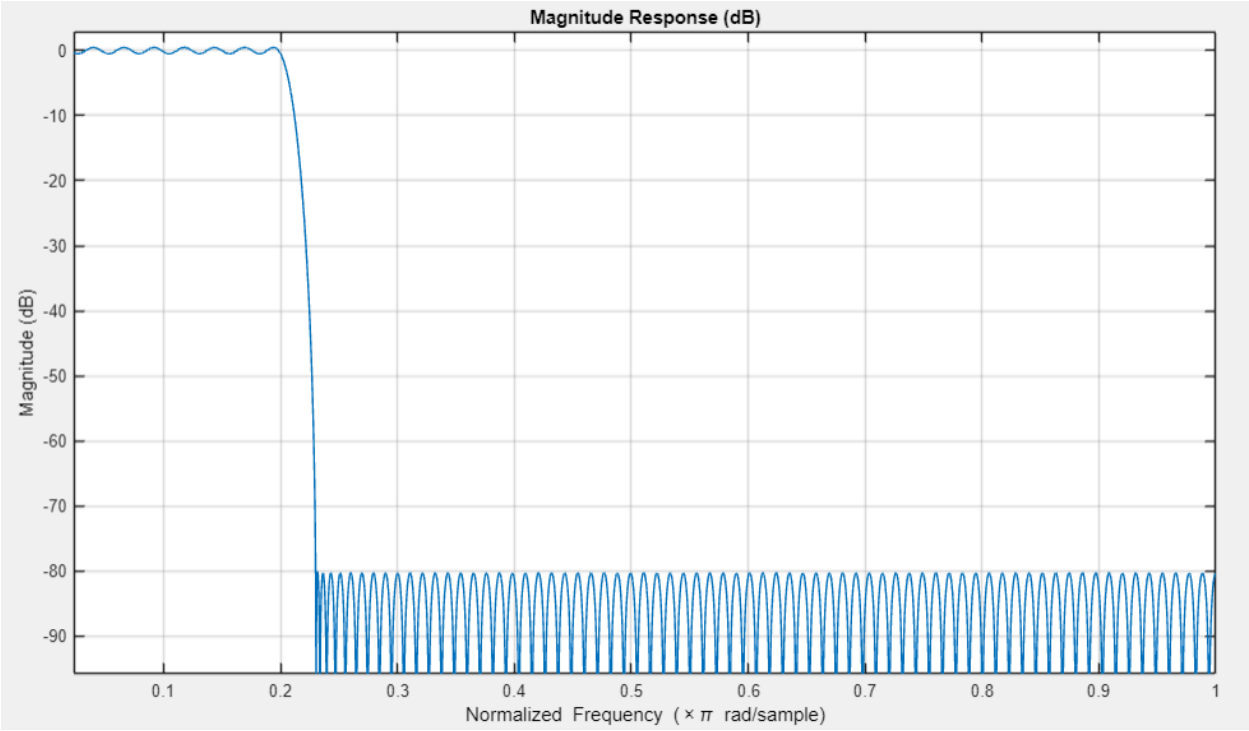

Figure 1: Low pass filter following design requirements

Figure 2 shows the frequency response of a MATLAB function inputting the quantized tap coefficients from the SystemVerilog code. Here a function, dsp.FIRFilter() takes in a matrix (Output.txt) with tap coefficients and plots the frequency response. *Figure 2* has a very similar passband and stopband to *Figure 1*, but lacks the design requirement of a stopband attenuation of 80 dB and stopband ripple is very irregular. Both of these changes are likely due to the decrease in signal resolution brought about from the quantized tap coefficients, an expected result.

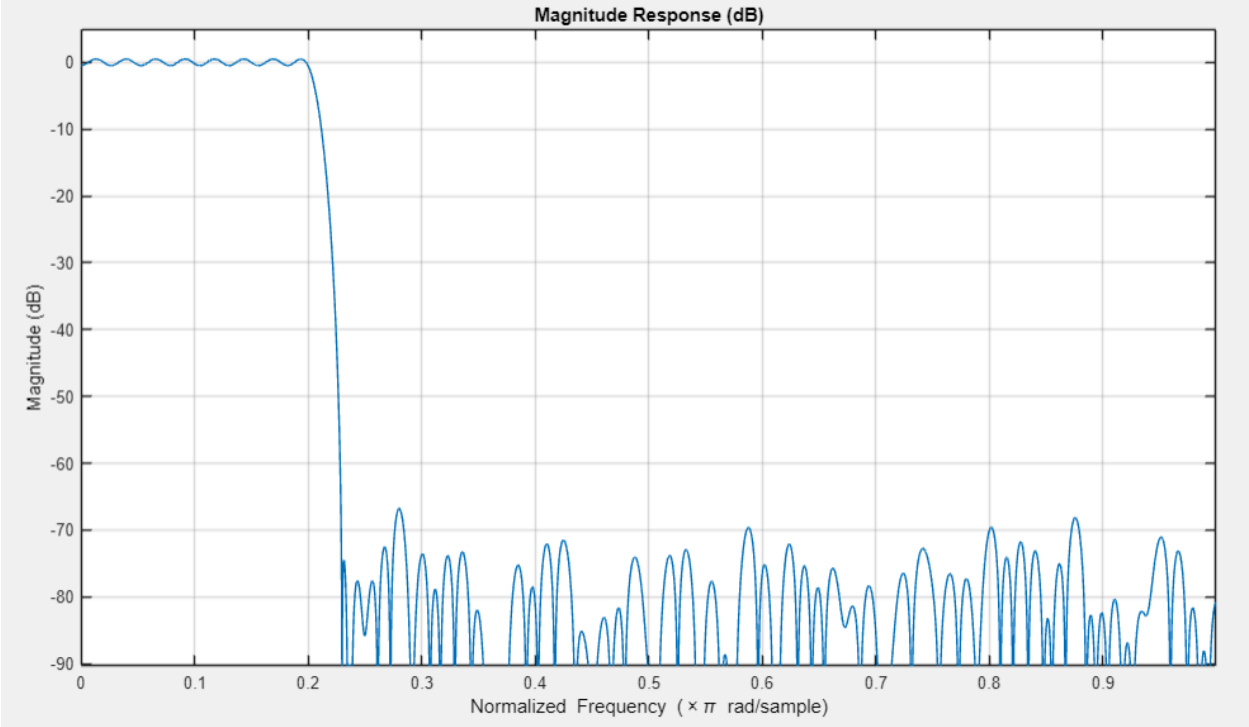

Figure 2: Low pass filter using quantized tap coefficients

SystemVerilog:

The SystemVerilog code implements a pipelined FIR filter architecture. The structure follows the pipelining architecture outlined in the class textbook, and matches *Figure 3* except with 175 taps instead of 4 taps.

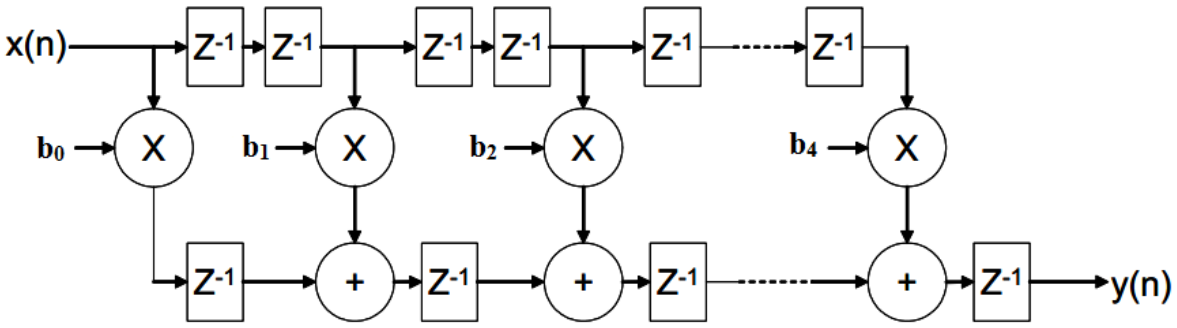

Figure 3: Pipelined FIR filter [Xilinx]

The Testbench.sv file starts by reading the tap coefficient file (Coefficients.txt) and converting the floating point data into a quantized integer of 16 bits. For quantization 16 bits was selected as it allowed fairly close approximation of input floats, these values are within the range -1 to 1 and computing the resolution 2/65,536 = 0.000030517578125. To start the value one is added to the input floating points, to map between 0 and 2, and then are divided by the resolution, and rounded to the nearest integer. There were no issues with overflow. Now the input data is quantized and stored to the QuantizedCoefficient.txt file.

The LowpassFilter.sv file starts by reading the quantized tap coefficients into array, h. Then the following variable arrays are initialized: delay, stageOut, and sumOut. These can be seen on *Figure 4*, an annotated version of *Figure 3.* These arrays will contain all relevant data as it passes through the pipelined FIR filter.

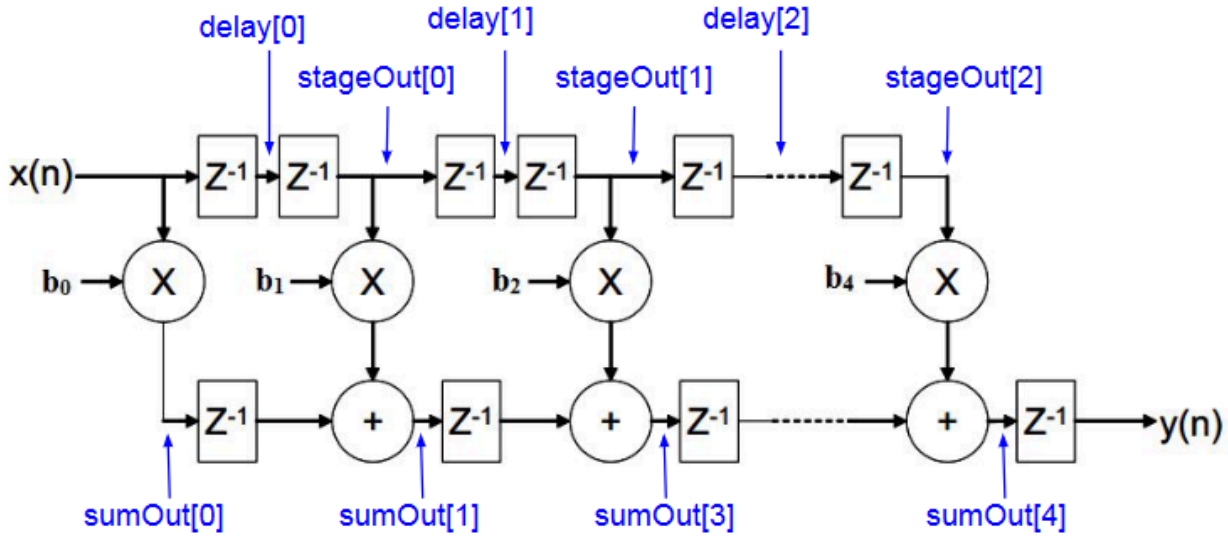

Figure 3: Annotated pipelined FIR filter [Xilinx]

The next section of code is the operational sequence of the pipelined FIR filter. Where every positive clock edge a new value of x(n) is inputted, and the filter moves data across the delays. As an impulse function really only has a data point of interest (x(0) = 1) a variable, n, is used to track which tap has 1 value of the impulse function. This variable, n, is updated based on the value of delay[n] or stageOut[n], as the next tap is used after stageOut[n]. This variable, n, is also used to shift both the top and bottom paths of *Figure 1* towards y(n). The main logic both shifts the data in sumOut, but also controls the addition of data to sumOut. Finally, there is some initialization logic due x(n) not properly updating delay, and sumOut in the first cycle. The output y(n) is equal to sumOut[174], after the 175th tap, is sent back to the testbench.

The testbench then receives the integer and decodes it to the approximate floating point value by doing the reverse of the quantization technique mentioned earlier. This new quantized tap coefficient is written to a file for storage, which can be read into MATLAB for analysis.

Intel Quartus was used to analyze the hardware implementation of the SystemVerilog code. Using the Power Analyzer Tool, the total thermal power estimate for the design is 198.33 mW. As this is an FPGA software, there is no tool to determine the exact size of the hardware implementation, but this design does require 3,330 registers. Using the Timing Analyzer tool, the max frequency of the design is 157.11 MHz.

Conclusion:

The low pass filter requirements are met by the FIR filter, and the quantized filter frequency response did differ from the original filter frequency response, but it was expected. The design of the hardware implementation was successful, a pipelined design reduced the

critical path in the FIR filter, thus allowing a much higher max frequency. One disadvantage of the pipelined FIR filter, that it takes 175 clock cycles to see output other than zero, as the data must be shifted through 175 delays. Overall the hardware met the specifications resulting in a successful design and operation of a pipelined FIR filter.