Pre-lab Questions

Q1. Answer the following questions about the filter

a. What is the order of your filter?

b. How many biquad sections are in your filter?

c. Look at the group delay.

  i. Is it constant?

  ii. In terms of *time* what is the largest the group delay is in the passband?

  iii. In terms of *time* what is the smallest the group delay is in the passband?

  iv. If you had an FIR filter of the same order, what would be the answer to **Error! Reference source not found.** & **Error! Reference source not found.**?

  v. Is this filter linear-phase in the passband? Explain your answer.

Q2. Examine coefficients of the direct-form II SOS filter. (You only need to examine the matrix SOS and ignore G in Matlab)

a. What is the largest absolute value found among these coefficients?

b. Examine the pole/zero plot. Explain how you know that all coefficients (both numerator and denominator) are certainly within the range of -2 to 2. Hint: we proved this for the denominator in class based upon an assumption that happens to be true for the numerator in this case too.

c. Convert the filter out of SOS form and into a single section (in the Edit menu). Discuss the range of (absolute) values now seen for the coefficients.

d. Is it easier to work with the SOS coefficients in fixed point or the single-section coefficients in fixed point? Why?

Q3. Write a MATLAB script to read in SOS matrix, normalize the coefficients for unit DC gain (see below) and prints them in Q15 format so that you can paste them to your verilog and C code.
Each row of the SOS matrix contains the coefficients for a second order section.
The coefficients are ordered: b0 b1 b2 a0 a1 a2
In your script first determine the 0 Hz gain for each biquad section and then normalize the numerator values so that each section has unit gain at 0 Hz. Again you can ignore the variable G exported by Fdatool.
To normalize the DC gain of a section to unity multiply the b values for that section by

$$(a0 + a1 + a2) / (b0 + b1 + b2)$$

Q4. Now let us look at the impulse response for a bit. Export the filter to the Workspace as Objects. You can get the impulse response by using the impz function. The syntax would be **`impz(Hd)`** assuming your filter was exported as "Hd". That command causes a graph to pop up, but it also returns some values. Try "**`help impz`**".

a. What is the sum of the absolute values of the impulse response? What does that tell you of interest? Explain why.

b. If you wanted to input values which maximized the output value, what would be the last 25 values you would input (in order from last in to 25th to last)? Assume inputs must be between 1 and -1 (inclusive).

Q5. Print the pole/zero plot.

a. Label which poles should be paired with which zeros.

b. Label each zero/pole pair with a single letter name (A, B, etc.).
c. According to the ordering heuristic we are using for biquad sections, list the order we'd put the biquad section in from closest to the input to closest to the output.
d. Using the same scheme as above, list the order MATLAB put the biquad sections in.

Q6. Go back to lab 2 and look at how we implemented the FIR filter in lab 2 Part 3.

a. Before sum gets shifted and returned what is the range of representation for sum?
b. What is the range of representation for the return value?
c. Modify the program from lab2 part 3 so that if sum's value is out-of-range for the return value, we saturate the return value. Think about how overflow was handled in lab 4. Do this with simple > and < operators rather than bit selection…

Q7. Think about the issues associated with implementing a filter with an arbitrary number of biquad sections. Describe, in your own words, how to implement an SOS-based IIR filter. How it would likely be different if you are to use only 1 single-section, i.e., the coefficients you obtained in Q2(c)?

Q8. In your own words, explain the role of each of the arguments to **`iircas5()`**.

Q9. How does the implementation handle internal overflow protection?

Q10. Answer the following questions about the following special-purpose registers used in the code

a. TCR0 is the timer control register for timer 0. What do the constants associated with "TIME_STOP" and "TIME_START_AUTOLOAD" do? Explain.
b. TCR0_1 isn't defined in the SPRUFO2 document under that name. Looking over the code and that document, what is the name of that register in the document? What is its role?

In-lab & Post-lab Questions

Q1. What should `biq_32` be initialized to if you were to build the filter in **Error! Reference source not found.** of the pre-lab?

Q2. How long does it take your IIR filter to process a single input? Be sure the compiler is fully optimizing (-O3).

Q3. How long does it take your transposed IIR filter to process a single input?

Q4. What is the worst case delay for your FPGA implementation? How does it compare to your design on C5515?

Q5. At a high-level, what is going on in this code? Specifically explain when `Timer_Handler()` is being called and what that function is doing. How do calls to `Timer_Handler()` result in the LEDs flashing?

Q6. What's the pre-scalar value located in TCR0_0? What does it mean?

Q7. Go back to the filter coefficients you used for the **G2**. Use FDATool to design that filter again in Matlab. Write a Matlab script to plot out the gain vs. frequency for each delay stage. Use the Matlab DSP library function `[H,F] = FREQZ(B,A,N,Fs)` to help you. Plot out the gain vs. frequency for each delay stage. Be sure to set your axis appropriately. Turn in your code and plots.