525.742 SOC FPGA Design Lab

Laboratory Project 4B

FIR Filter Implementation

**Introduction**

In Part A of this lab, you designed a pair of filters which (when combined together) will effectively serve as a channel selection filter for our radio. Since the simulation was done so thoroughly, instantiating the actual filters into our fpga design should be relatively painless. Furthermore, we have an excellent model in matlab which should be able to predict exactly what our design will do under any circumstances. In this part of the lab, we will complete the effort, implementing the FIR filters we designed, audibly listening to the results, and doing a quick validation to make sure that everything behaves as expected. As you are designing, make sure you understand and properly handle any clock domain crossings. This week, your laboratory will be expected to de devoid of any timing errors when the clocks are constrained to operate at the expected frequencies.

**Procedure**

Designing the Filter Hardware

Step 1: customizing the cores

In this step you will want to create two FIR compiler cores, one for each filter you designed. Since you have already modeled the exact parameters that you want to use for each filter core, this process should be much easier, and devoid of any guessing. You can create these cores in the block diagram or use them directly in hdl, either will work, and each has it’s advantages. (My personal favorite is to have the filter instantiated in an HDL file, but the block diagram certainly works). Use the ‘coe’ files that you wrote out from your model in Part A. Make sure to specify decimation rates, input/output sizes, coefficient quantization...etc. (your model is the guide here). IMPORTANT : if they aren’t there already, copy your COE files into a space inside your project directory structure now before specifying them in the FIR compiler GUI. The most common problem I see that causes people to lose points for an incomplete project is that the COE files are somewhere else, and don’t get included when the student submits the project.

Step 2: adding filters to the design

add the two filters, cascaded as in your model, to your complete DDS design from the last lab. Their input data should be driven from the output of the dds. Add or move an ila on the output of the filter chain so that you can quantitatively observe the filter output (we will be looking at this later) For debugging, you may use ILAs to look at any of the stages, but the final output is required for data analysis

Step 3:

When testing the design, we’d like to be able to listen to **either** the filtered output, **or** the unfiltered DDS as in Lab 3. To this end, have SW0 select whether the DAC interface gets its data from the DDS or from the output of your new filter chain.

Step 4:

Test the design, and make sure that your filters filter! In short, this lab should be indistinguishable from your Lab 3, except that there will be no aliasing as you go to higher frequencies.

Step 5:

Using an ILA, capture at least 1024 output points when exposed to a signal in the transition band of the filter (25kHz). Plot the results and compare **qualitatively** with a signal from your bit-accurate model simulation. Do the results line up? (Note, we won’t expect bit level accuracy here, since we aren’t driving the two models with the exact same signal; however we should expect the amplitude of the resultant signal to be roughly the same).

Step 6:

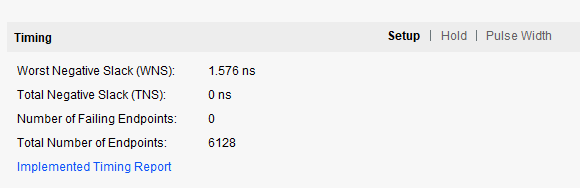
Do the same analysis as step 5 for a signal input at 50kHz (this is in the stop-band of the filter).

**Software Implementation**

The software for this lab can be unchanged from your previous lab (DDS) -- no software is needed this week!

**What to Turn In (in addition to your complete project archive and vitis export)**

1. **A BOOT.BIN file which can be put on an SD card and runs your project.** Please include this file directly (i.e. not inside a zip file) Note – don’t worry about power-up behavior, I will press PS\_RST to demonstrate
2. **A document which:**
   1. **shows your ILA plot and the corresponding plot from the bit-accurate model for a signal in the transition band. The goal here it to show that your filter behaves like the model. Please submit this file directly (i.e. not inside a zip file)**
   2. **Lists the various clocks in your design (you may exclude the “debug\_clk”), what signals cross between them, and how you assured that there were no timing violations or possible errors in operation. Cut and paste the text results from report\_cdc, and make sure to explain the total number of “Endpoints” – it should make sense.**
   3. **Contains a screenshot showing the results of static timing analysis like below (the project summary page is perfect here…) The student should use this information to estimate at what clock rate this design could be safely run before setup violations might occur.**

****