525.742 SOC FPGA Design Lab
Laboratory Project 5
Full Microprocessor-controlled Digital Downconverter

NOTE : Please read the rubric carefully, some of the details of the requirements are captured best there.

**Introduction**

In this lab we will fully integrate the DSP chain of the software defined radio (SDR) we have been developing in the previous labs. At the end of this laboratory, we will have a signal processing block which can tune to any 36kHz wide segment of spectrum between 0 and 62.5 MHz, convert it to baseband, and play it out of the audio DAC.

**Lab Description**

This laboratory will be based heavily on the framework developed in Lab4. The following defines the concept of operation:

- A/D converter digitizes input signals. As in Lab 3 and 4, *An on-chip DDS will still simulate the A/D converter in this lab.* (125MHz sample rate). This DDS, exactly as used in lab 3 and 4 will be referred to as the "signal source", or the "fake ADC"
- Data from the fake ADC is mixed with the output of a tunable DDS so that any section of the digitized bandwidth can be moved to baseband (DC)
- Data from the mixer is then passed to a channel selection filter (developed in Lab 4), which will filter out all but a narrow band around DC, and decimate by 2560 to a sample rate of approximately 48.8kHz
- Data from the output of the filter should be scaled correctly, and sent to the DAC interface.
- **The DAC Interface in this lab must be used from the IP catalog.** In other words, you should take the provided IP (attached to the assignment in Canvas) place it in a directory of your choice named for example "ip_repo" and point your project to look there for any IP blocks that it can put in the catalog. Then you can instantiate the dac interface in your block diagram or in your top VHDL just like any other IP from the catalog – it should show up there!

**Lab Requirements (Check Grading Sheet for detail)**

- As in Lab 3, 4 the processor accepts commands to change the frequency of the fake ADC. ('F', 'u', 'U', 'd', 'D').
- Additionally, the processor should accept a new command 'T', which will tune the radio. This will accept a desired tune frequency from the user, and it will use that input to correctly set that phase increment of the tuner DDS such that the tune frequency is

shifted to 0 (DC).

- The Processor should accept tuning commands over the serial port in the format listed below. It should respond to these commands by tuning the receiver to the specified frequency (in other words, a signal injected into the analog input at that frequency should show up at DC in your output data path). Frequencies from at least the range of 1MHz to 60MHz should be supported.
    - "T11250000" (tunes to 11.25 MHz)
    - "T2000000" (tunes to 2 MHz)
- Data from the signal processing section should always be coming out of the DAC such that "Q" (the imaginary part of the sample) is on the Left Channel, and "I" (the real part) is on the Right channel
- The student should also implement 2 more commands for the processor to handle ('+','-'). These will be used to increase and decrease the volume of the output signal. There should be 10 settings, 0-9. 8 should be the same amplitude that you have used for Lab 3 and Lab 4. Each step is 6dB. This volume change can most easily be done by changing registers 2 and 3 of the CODEC, but can also be done by manually shifting the bits of the data before it is sent to your DAC interface.

**Checkout :**

1. Verify tuning and Signal Processing Path:
    a. Tune the receiver to a frequency specified by the instructor (ex: "T30000000" to tune to 30MHz)
    b. Set your fake ADC (just like Lab 3,4) to Inject a tone near that frequency which will pass your filter band (e.g. F30000100 injects a 30.0001MHz tone).
    c. Sweep injected frequency through the filter passband in both directions (+ and - from the tune center). Verify output of DAC
        i. Note that it will be very hard to hear the difference between + and – frequencies (after all, both are sinusoids of the same freq on L and R channels, differing only in phase relationship).
    d. Sweep the injected signal well outside the passband and make sure that signals from other channels are not present on the the DAC output.
2. Verify volume Control
    a. Return your fake ADC to something which is close enough to the tune frequency to be heard. Adjust the volume up and down using + and - keys

**What to Turn in**

Student should submit the following

1. An unzipped BOOT.BIN which operates your entire design
2. your complete Vivado/Vitis projects. For Vitis, please **export from Vitis itself** to create a zip file. Expected submission is "vitis_export_archive.ide.zip"
3. Please also submit a brief text description of any special things that should be noted when running your project. If there are any parts that don't work…etc.