525.742 SOC FPGA Design Lab

Laboratory Project 3a

SDR Signal Source Simulation

Late Assignments: Up to 1 week late for 10 point penalty

**Introduction**

In this lab we will continue the development of a software defined radio (SDR) on our evaluation board by creating the most basic signal processing chain – a programmable frequency sinusoidal input signal which will be the test input to our radio. The input signal will be generated by a numerically controlled oscillator (NCO), also called a direct digital synthesizer (DDS).

The main point of emphasis of this lab is to use all available tools at our disposal to thoroughly understand our design. This will focus on modelling our component in Matlab, and then simulating in ISIM. We will be experimenting with these, in preparation for including the core in an FPGA design in Lab 3b.

Note that there is some instructor provided Matlab attached to the lab assignment which will give you a good head start on the tasks needed for this assignment.

***Part I: IP Construction***

*Create a blank brand-new project which is targeted to the Zynq device on the Zybo Z7-020* Click on “IP Catalog” and locate the DDS Compiler. Double click, and use the GUI to make a DDS with the following capabilities:

* < 1 Hz frequency resolution. Our DDS is running at 125 MHz, this means that we need a 27 bit phase accumulator (frequency resolution = 125 MHz / 2^27).
* 16-bit signed output words, only one is required (sine or cos)
* Taylor Series Noise Shaping
* The ability to reset the phase to zero (put a reset pin on the DDS so you can start it from scratch any time you’d like).
* Note : no phase output is required

***Part II: Gate Level Simulation***

After customizing the IP, construct a testbench for it, so that we can investigate its performance. Instantiate the DDS core into the testbench, and include in your testbench all signals that are necessary to produce a signal that starts at angle 0 and is as close to 50kHz as possible. Also include a piece of code that will write all valid output samples from the DDS to a text file. (There is an example of how this can be done in the lecture slides)

Run your simulation for long enough that your file has at least 8192 samples in it. Keep that text file that you created, you will be using it in Part III and you will also need to submit it as part of the final submission package.

**Part III : Matlab Model**

If we were constructing a simple function like an adder or multiplier, it would be quite clear how to make a test bench and it would be easy to know that we were getting the absolute correct answer (bit accurate) out of the core. However, as the cores become more complex, knowing that they are doing **exactly** what they are supposed to be doing is not a matter of simple inspection. Yes, we know that the DDS is producing something that looks kind of right, but is it’s frequency exactly right? Is it’s SFDR as predicted? These are more challenging to verify. This is particularly true with signal processing cores, where performance is usually measured in a complex system with lots of inter-related cores. For this reason, a designer of a large FPGA usually creates or has access to a system-level Matlab model which describes how everything should work together. It is exceptionally convenient if that model can predict the **exact** behavior of the FPGA core itself. For many of the cores, Xilinx provides a bit-accurate model of how the core performs that can be run in C or Matlab, a much better (and faster) place to visualize data than with ISIM. For this part of the lab, we will use the provided model with Matlab : first to validate that our core is producing exactly the results that we expect, and next to measure a couple key performance characteristics of the DDS core.

Note that if configured identically to the DDS core itself, the provided model should produce 100% identical data.

* locate the bit accurate model for the DDS. This is going to be a zip file which is included for you once you’ve generated the ip. For my project, that is here : C:\projects\lab3\_dds\lab3\_dds.gen\sources\_1\ip\dds\_compiler\_0\cmodel
* Unzip that model somewhere, anywhere that is convenient, pick the nt64 for windows and lin64.zip for linux machines.
* Open matlab and navigate into that directory. To get started, run the script “make\_dds\_compiler\_v6\_0\_mex.m”. Unless you’ve made MEX files before, you will probably get an error about “Supported Compiler Not Detected”. To fix this you will need to install “mingw”
  + Add-Ons -> Get Add-Ons
  + Search for mingw, and select it and then click “install”
  + Run the “make\_dds\_compiler\_v6\_0\_mex.m” script again
  + It will likely still not work – go to line 16 of the make\_dds\_compiler\_v6\_0\_mex.m file and remove the string : “-lgmp” from the end of the compile line. Then run the script again.

At this point, you are setup to use the bit accurate model in matlab (but of course don’t really know how to use it). So, let’s start with an example. Copy the two provided files (in the lab assignment – dds\_simulation\_example.m and simulated\_sig\_gen\_107374.txt into the directory where you have built the model. Run dds\_simulation\_example.m and two plots should be created. Plot 1 overlays the results from the bit accurate model with that of what I recorded in my vivado simulation (the text file). Plot 2 subtracts the two for easier differencing when they are close. **Note : they aren’t close now, that is intentional.** Our goal for this lab will be to create a matlab simulation that

1. uses the bit accurate model to model how the Xilinx DDS will create a sinusoid as close to 50kHz as possible. We will call this signal “sig\_model”
2. Using that model, answer a couple of questions about the performance of the Xilinx DDS core and use it as the foundation for a system level model of what we will be constructing in the FPGA.
3. Compares the results from the gate-level simulation of the core (recorded in Part II of the lab, we will call this signal sig\_sim) with the results from the bit-accurate model. (they should be identical, because this is a bit-accurate model after all)

Next, in Lab 3b, we will implement the DDS in our FPGA, record the output using an ILA, and add those results to our matlab comparison!

**What to Turn In :**

2 separate files:

1. a word document that shows two plots that you created from your matlab with the DDS set to produce a 50KhZ signal, and the answers to a few questions:
   1. Time domain plot of 8192 points of the two signals in question (sig\_model, and sig\_sim) overlayed on top of one another
   2. A plot of the difference between sig\_model and sig\_sim (i.e. plot(sig\_model-sig\_sim))
   3. Question : what changes about the result when you operate the DDS in ‘Unit Circle Mode’ vs the alternative?
   4. Question : compare the resource utilization of the core (DSP slices / BRAM) when you make the same DDS (27 bit phase, 16 bit output) using the Taylor Series Noise Shaping vs. No noise shaping?
   5. Question : what phase increment value did you use to attain 50kHz? What is the computed actual frequency created to the nearest **1/1000th** of a Hz
2. A zip file which contains your matlab script (so that I can run it) and the text file you recorded from your gate-level simulation in Vivado. The VHDL and project is not required

When you are done with this part of the lab, you should have a very good idea of exactly what the DDS will be producing for Lab3b, and exactly what parameters you should set in order to make it do that. Start constructing the hardware for Lab3b if you finish early, it will be good to have a head start!