WIRELESS OPEN-ACCESS
RESEARCH PLATFORM

Lab 2: Building a Simple Transmitter

Siddharth Gupta & Patrick Murphy

Rice University

WARP Project

Document Revision 7

November 23, 2007

1 Introduction

The goal of this lab exercise is to build a simple transmitter and test it in hardware. The transmitter model will be built in System Generator, converted to a peripheral core and integrated with WARP platform support packages in Xilinx Platform Studio. When complete, the design will transmit a sweeping sinusoid at RF using the WARP FPGA and Radio boards.

Note: All files are stored in `C:\workshop\userN\` where `userN` is your user login location. This location will be referred to as `.\` for the rest of the lab.

2 Sinusoid Generator Model

The first step is to explore the sweeping sinusoid generator model in System Generator.

1. Open MATLAB 2006b and change directory to `.\Lab2_SweepingTx\sysgen\`.
2. Open the System Generator model `sweeping_tx.mdl`.
3. Run the model and view the output on the Simulink Scope. You should see a sweeping sinusoid.
4. Double-click the **FreqSweepRate** block and change the **Initial Value** parameter to 100.
5. Run the simulation again and verify that the rate of frequency sweep is slower than before.

3 Generating the Peripheral Core

The next step is to create an OPB-compliant peripheral core from the System Generator model.

1. If it's still open, close the System Generator model `sweeping_tx.mdl`.
2. On the MATLAB command line, execute: `sysgen2opb('sweeping_tx')`.
3. Open the `sweeping_tx.mdl` model again. Notice that two new subsystems have been created which implement the OPB interface.
4. Double-click the **System Generator** block. Make sure its parameters match those shown in Figure 1.
5. Click the **Settings** button, then click the folder icon. Navigate to `.\xps\system.xmp`, click Open, then click OK.
6. Finally, click Generate. This process will take a few minutes. If it's successful, the dialog box will say Generation Complete.

4 Integrating the Transmitter Model

We have provided a pre-built project in Xilinx Platform Studio for this exercise. This step integrates your custom transmitter with this platform.

1. Open Xilinx Platform Studio. When prompted, select **Open a recent project**, click OK, then navigate to `.\Lab2_SweepingTx\xps\system.xmp`.
2. Click the **IP Catalog** tab on the left of the screen, and expand the **Project Local pcores** category.

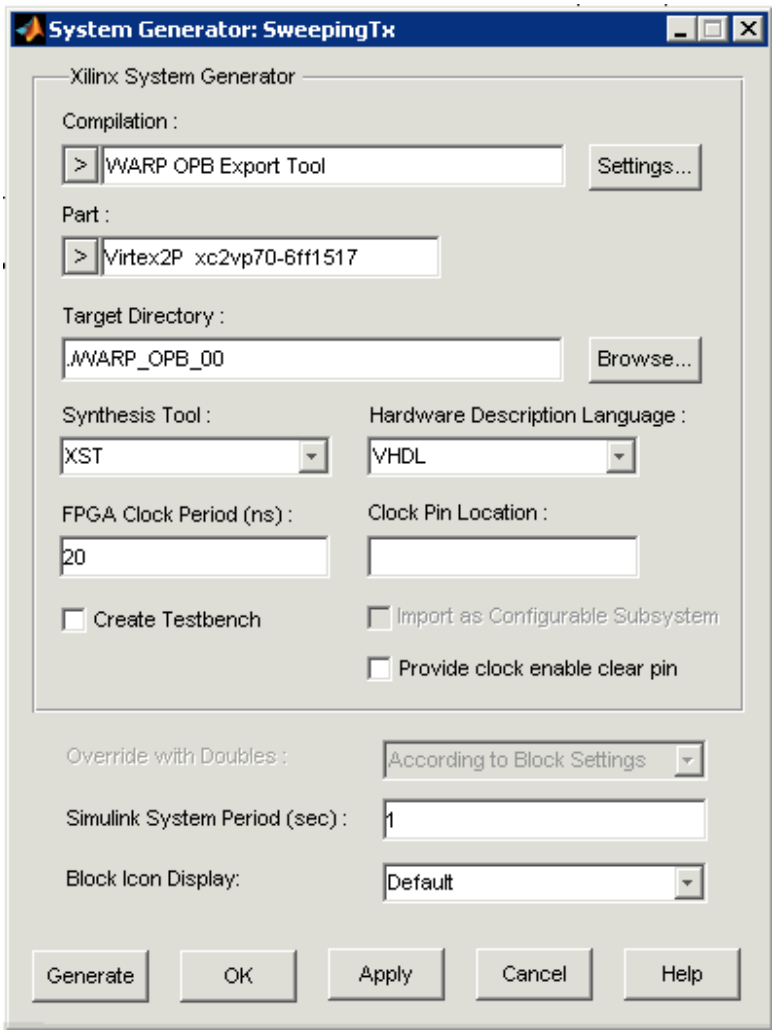

Figure 1: System Generator parameters for OPB core generation

3. Double-click the **sweeping_tx** core and click 'Yes Add IP'.
4. Click the **Bus Interface** button in the middle of the screen and look for the **sweeping_tx** core in the list of included peripherals. Expand the core's entry and click the hollow green circle to attach the core to the OPB.
5. Switch to the **Ports** view and scroll down to the **sweeping_tx** entry. Expand the entry. Enter unique net names for the two DAC ports: **DAC_I** for port **radio2_DAC_I** and **DAC_Q** for port **radio2_DAC_Q**.
6. Scroll to the **radio_bridge_slot_2** core and expand its entry. Enter the same net names (DAC_I and DAC_Q) for ports **user_DAC_I** and **user_DAC_Q**.
7. When complete, your ports list should look like those shown in Figure 2.
8. Switch to the **Addresses** view and click **Generate Addresses**. If you get a warning about your system containing two processors, click 'Ignore'. When this process finishes, the **sweeping_tx** core will have an automatically assigned base address.
9. Choose Hardware → Generate Bitstream to begin the hardware synthesis process.

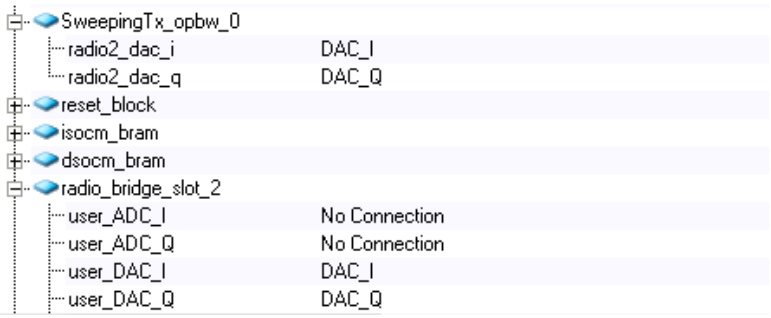

Figure 2: Custom core and radio bridge ports

5 Driving the System from Software

1. Switch to the **Applications** tab on the left of the screen. Expand the Sources list and double-click the one file entry.
2. Look through the C code to understand how the radio is setup and controlled through the **radio_controller** core. Scroll down in the code and look at the line which writes a frequency sweep value to your custom peripheral (it's the line starting with `sweeping_tx_WriteReg...`). This value sets the frequency sweeping rate; a smaller value here results in a slower sweep.

6 Testing the Design in Hardware

1. Make sure your WARP FPGA board is connected to power and USB.
2. In XPS, Choose Device Configuration → Update Bitstream. This will compile any code changes and update the FPGA programming file.
3. Using iMPACT on your local PC, download the bitstream `.\Lab2_SweepingTx\xps\implementation\download.bit` to your FPGA board.
4. If everything works, the radio's green LED will illuminate to indicate the radio is transmitting, and your node will be generating a sweeping sinusoid at RF. You can observe the waveform on the spectrum analyzer in the lab. It should look something like Figure 3.

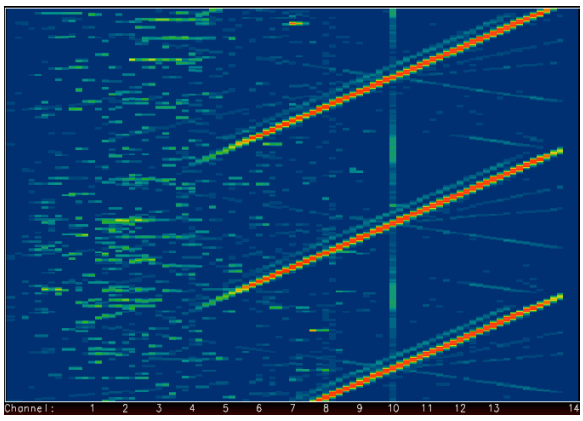

Figure 3: Spectrum Analyzer Output

7 Optional Exercises

If you finish the lab with extra time, here are a few extensions to try.

- Change the frequency sweep rate to switch the slope of your transmitted signal on the spectrogram. You'll have to figure out what value to write to the FreqSweepRate register that effectively negates the frequency accumulator input.
- Try altering some of your radio's settings and observe the changes on the spectrum analyzer. You will need extra functions from the radio controller driver (see http://warp.rice.edu/WARP_API)
  - Change the radio transceiver's center frequency to a different channel.
  - Increase/decrease your radio's transmit gain.
  - Change the transmit low-pass filter bandwidth to its highest setting to see a wider sweep of your transmitted sinusoids.