Laboratory V – Fall 2005

Servomotor Controller

Introduction

This lab deals with issues of generating signals to drive the FPGA's I/O pins directly. The RC-200E brings several FPGA pins out to the 50-pin expansion header, along with some power and ground connections. In this lab you will connect a servomotor to one of the pins in the expansion header and use the pushbuttons to make the motor rotate one way, the other way, or not at all. To operate correctly, your code must generate pulses of exactly the correct frequency and pulse widths, and you will verify the correct behavior both by simulation and using an oscilloscope before actually connecting a servomotor to the RC200E.

Project Specifications

Your program is to generate a pulse every 20 msec. If one pushbutton is pressed, the pulse is to be 1.0 msec long; if the other button is pressed, the pulse is to be 2.0 msec long. If neither button is pressed, the pulse is to be 1.5 msec long. The servomotors we are using, Futaba model number 3003, rotates clockwise when it receives 1.0 msec pulses, counterclockwise when it receives 2.0 msec pulses, and not at all when it receives 1.5 msec pulses. (Clockwise and counterclockwise depend on your point of view.)

There are several ways you could design your code to meet these specifications. But for this lab you are required to use one endless loop to time the millisecond intervals and a second endless loop to generate the pulses. The two loops must synchronize with each other using a Handel-C channel.

Lab Activities

1. **Work Through the First Waveform Analyzer Example**
2. **Write a Program that Outputs to a Pin**
    a. **Verify your program using the Waveform Analyzer**
    b. **Verify your program using an Oscilloscope**
3. **Write a Program that Controls a Servomotor**
    a. **Verify your program using the Waveform Analyzer**
    b. **Verify your program using an oscilloscope**
    c. **Verify your program using a servomotor**
4. **Submit a Report of Your Lab Activities**

Work Through the First Waveform Analyzer Example

The Waveform Analyzer (WA) is a tool provided by Celoxica that can be used in two ways: to observe an output signal generated by a Handel-C program, and to generate a pattern of input signals to a Handel-C program. Both are valuable aids in the development of certain kinds of applications. In this lab you will develop a Handel-C application that outputs pulses to a motor, and you will use the WA to make sure the program is generating the pulses correctly during simulation. You can't use the PAL and its simulation support for this project because the PAL simulator doesn't handle the direct-to-pin outputs you need for this project. You won't be doing anything with the pattern generation side of the WA.

To start, create a DK Workspace named "Laboratory V" for this lab in your "My Projects" directory. Add a new project named "View Waveforms" to this workspace, and add a Handel-C source file named *view_waveforms.hcc* to the project. You do not need to configure the project in any way; the default settings for the Debug configuration will suffice. And this project is for simulation only, so you can delete all the other build configurations, including EDIF.

Find the Waveform Analyzer Manual on your system (…\Celoxica\DK\Documentation). Follow the directions through Section 1.4 (pages 5-7) for tracing the execution of a simple Handel-C program. Typing in the code and following the directions carefully will be instructive in itself. Write answers to the following questions within the comments of your code.

- What are DKSync.dll and DKConnect.dll?
- Measure the period of the output waveform using cursors (see page 10). Relate the period to the set clock statement and the values that the variable *x* takes on in the Handel-C program.

Write a Program that Outputs to a Pin

Create a second project named "Generate_Pulses" in the Laboratory III workspace, and configure it for Simulation and for the RC200E in the usual ways. Delete the unused build configurations (Release, Verilog, etc.). Write a macro procedure named *microsec_delay()* that delays the program for the number of microseconds passed as a parameter. Code your program with two threads. One writes to a channel every 20 msec, and the other generates a 1 msec pulse every time it is able to read from the channel[1]. Have the output of the program connect to Pin #3 of the RC200E expansion header.

To connect a variable in your program to an output pin, you need to use a *bus_out()* interface. Chapter 11 of your Handel-C Reference Manual covers interfacing to pins; Section 11.1.4 covers *bus_out()* in particular. You also saw some examples of interface declarations in the Waveform Analyzer project.

You need to know which FPGA pin is connected to pin #3 of the RC200E expansion header. Look it up in the *RC200 Hardware and PSL Reference Manual* in the …\Celoxica\PDK\Documentation\PSL\RC200\Manuals directory.

[1] If you've forgotten how to do this, it's explained in more detail on page 4.

Verify your program using the Waveform Analyzer

My Waveform Analyzer Help web page gives important additional information on using the Waveform Analyzer for this project. (But don't look at it until you have looked up the pin number in the RC200 Reference Manual; the web page gives away the answer!)

Verify your program using an oscilloscope

When you are ready to test your code using an oscilloscope, follow these instructions:

Set all the rotating knobs to their upright positions except the three labeled "VAR" which should be set to the "CAL" (calibrated) positions. Apply power to the scope, and clip the A probe to the "CAL" loop on the lower left front corner of the scope. Press the green "Auto Set" button and adjust the intensity control (under the power switch) so the display is neither too dim nor too bright. There should be two traces on the screen, one showing a square wave (the A channel) and the other showing noise (the B channel). Press the A/B button under the Auto Set button to make the B channel go away. Look at the information displayed in the dim window to the right of the screen and verify that the amplitude of the square wave is 1.2 volts and that the period is 0.5 msec. You may need to adjust the X and Y positions of the trace to be able to measure these values.

With the scope set up, you can connect the probe to pin #3 of the RC200E expansion header. Use a ribbon cable to plug into the expansion header; connecting the oscilloscope probe directly to the pins of the header will bend them. Press Auto Set to get the scope to adjust to the signal you are generating. Press the left or right side of the MTB rocker switch to make the "on time" of your pulses take up two horizontal boxes on the display, and verify that you have 0.5 ms per box (in the dim window) and that the "VAR" knob for X is in the "CAL" position.

Write a Program that Controls a Servomotor

Create a third project named "Motor_Control" in the Laboratory V workspace, and configure it for EDIF (RC200E configuration) and Simulation in the usual way.

Organize your code with two *main()* methods. One main method is to do nothing but write to a channel every 20 msec. The second *main()* method is to execute an endless loop that reads from the channel, then reads the two pushbuttons (use the PAL for this) and turns on the output pin. Depending on the settings of the switches, delay for 1000, 1500, or 2000 microseconds, and then turn off the output pin. Be sure you understand the logic of this code and how it prevents the 20 msec intervals from "drifting." Include comments in your code to explain the design.

Pseudocode for the two *main()* functions.

```
main()
   while (true)
         write to channel
         delay for 20 msec
```

```
main()
   while (true)
         read from channel
         turn on output
         depending on button states, delay for 1, 1.5, or 2 msec.
         turn off output
```

Verify your program using the Waveform Analyzer

Use the Waveform Analyzer to verify that all pulses start on 20 msec intervals and that the pulse widths are correct, depending on which buttons are pressed.

Verify your program using an oscilloscope

Connect the oscilloscope to an RC200E and verify that the pulses are generated every 20 msec (with no drift) and that the width of the pulses can be varied between 1.0, 1.5, and 2.0 msec using the two pushbuttons.

Verify your program using a servomotor

Connect a servomotor to an RC200E and verify that it rotates in the expected directions when the buttons are pressed. Demonstrate your program to Dr. Vickery when you have it working.

Submit a Report of Your Lab Activities

Use a word processor to write a report of your lab activities and send me an email when your project directory is ready.