Laboratory I

Basic Lab Procedures

Introduction

The purpose of this laboratory is to make sure your account is working correctly and to introduce you to the *design flows* used to develop FPGA projects in this course. You should be able to complete all the laboratory activities during the two lab sessions of the first week of the course. Your report for this lab is due by midnight on the due date listed on the course web page.

Activities

1. Set Up Your Account
2. Configure DK
3. Simulate and Run Celoxica Examples
4. Modify the SevenSeg Example
5. Submit a Report of Your Laboratory Activities

Set Up Your Account

Log into your account using the TREE domain from any computer in the lab. Create a directory (folder) named "My Projects" under your "My Documents" directory. Copy the entire directory "C:\Program Files\Celoxica\PDK\Examples\PAL" to your "My Documents" directory. Log out of your account and log into it from another computer. Verify that the PAL folder appears in your "My Projects" directory on the other computer.

Every time you log in or out, your entire "profile" is copied from/to the TREE domain server, which is named Maple. Your profile consists of your "My Documents directory and some other information from the "C:\Documents and Settings" directory. This is convenient because it lets you access your own files and folders from any computer, but it can make logging in and out time consuming if a lot of information is being copied. There is an H: drive on Maple where you can store files and folders that will *not* be copied and restored every time you log in and out. Create a directory on the H: drive with your name, and *move* the Examples directory there. Log out and back in on another computer. You should still be able to access the material you put on the H: drive, but the logging out/in process should have gone faster.

You can access your account from any of the black IBM computers in A-227 or A-205. You can get a key to A-205 from a secretary in the CS Department office any time the office is open. You can also access the computers using Microsoft's "Remote Desktop" application, which is usually on the Communications submenu of the Start Menu on XP systems. The computers are accessed using the form

<name>.cs.qc.edu. If you do use Remote Desktop, you must be sure you log off the computer when you finish using it. If you "disconnect" rather than logging off, you will make it impossible for anyone else to use the computer and you will be put in jail for the next six months.

Likewise, be sure to always log off when you finish using one of the computers in A-205 or A-227.

Configure DK

The Celoxica IDE for Handel-C development is called DK ("Design Kit"), and is accessible from an icon on the desktop and/or the Start Menu. You can set up a keyboard shortcut to it by right-clicking on the icon, selecting Properties and entering the shortcut key you want to use. (I use Ctrl-Alt-D.) There are some things you need to set up one time to get the IDE to work "right:"

- Select Tools → Options → Tabs. Change the tab size from 4 to 2, and select the "Insert spaces" radio button. This controls the behavior of the DK built-in editor. You can also edit programs using another editor if you wish, but if you do, you must set the equivalent options in that editor in order for your work to meet the Coding Guidelines for this course. The Vim and Emacs editors are available on the laboratory computers. Note that you cannot change the tabwidth or substitute spaces for tabs using Notepad, so that editor is not allowed as a program editor in this course.
- In the Options dialog, select Directories and set the directories for Include files to "C:\Program Files\Celoxica\PDK\Hardware\Include" and the directories for Library modules to "C:\Program Files\Celoxica\PDK\Hardware\Lib".
- You may change other options if you wish to customize your "DK Experience" in some way.

Verify that if you log off and back in that the options you set remain in effect.

Simulate and Run Celoxica Examples

Exit DK if it is running.

Use Windows Explorer to navigate to your copy of the Celoxica Examples, and go down to the PDK directory. There should be a dozen or so directories there and one file, named Examples.hw. (If you can't see the ".hw" extension on the file name, I suggest you open Explorer's Tools -> Options dialog and uncheck the "Hide extensions for known file type" checkbox in the View tab.) Double-clicking on the Extensions.hw file will launch DK and automatically open the Examples "workspace." (".hw" stands for Handel-C workspace.) The left of the two dropdown lists in the toolbar is for selecting the "Active Project." Use it to select the "SevenSeg" project. The right dropdown list is for selecting the "Active Build Configuration" for the current Active Project. Select the Sim build configuration. Use the Project -> Settings menu to bring up the Project Settings dialog. There are some relative pathnames in there that work only when the project is run from the Celoxica installation directory, not from the H: drive where

you made your copy. Go to the Linker tab and change the "..\..\..\" part of the pathname in the "Additional C / C++ Modules" list to "C:\Program Files\Celoxica\PDK."

You should now be able to build a simulation of the SevenSeg program by clicking on the Build icon on the toolbar (it looks like some blue buildings) or by pressing F7. You can run the simulation by clicking on the Run button (next to the Build button) or by pressing F5. One of the seven segment displays on the PAL Console that comes up should start stepping through a sequence of numbers and shapes. The fastest way to stop the simulation is to click the close button on the black Command Prompt window that came up when you started the simulation.

You have just used the DK IDE to compile a Handel-C program and link it to some libraries to produce a Windows executable program that simulates the behavior of the program running on an FPGA connected to a seven segment display. Compare the PAL simulation console with the actual RC-200E and observe the number of switches, buttons, LEDs, and seven segment displays available on both.

Now select the "RC200E" Active Build configuration for the SevenSeg project. You need to make some changes to the project settings for this configuration too. This time, go all the way to the last tab in the Project Settings, which is called "Build Commands." There are two relative pathnames there that will prevent the project from building. Because the Celoxica\PDK\Software\Bin directory is already in your Windows PATH, you can simply eliminate all the "..\" and "Software\Bin" parts of the two command lines, leaving just "Call edifmake_rc200 SevenSeg" and "beep."

Now build the SevenSeg project again. This time, after compiling the program, the edifmake_rc200 command (it's a batch file) will pass the EDIF netlist generated by the Handel-C compiler to the Xilinx configuration programs that will generate a ".bit" file suitable for downloading to the RC200E. (The beep command will alert you when the Xilinx tools finish.) Be sure there are no errors and that the sevenseg.bit file was generated successfully.

Connect an RC200E to the parallel port of your workstation and run the FTU2 program that is in Windows' Start → Celoxica → PDK menu. (You can right-click on the icon in the menu, select Properties, and assign a shortcut key such as Ctrl-Alt-F to this program if you like.) Select the RC200E from the left dropdown list of the FTU2, and test the port settings. If the FTU detects the RC200E all right, select FPGA, browse to the sevenseg.bit file, and use the configure button (or double-click on the filename in the FTU's list of configurations) to download the bit file to the FPGA. Verify that the real seven segment display does the same thing as the simulated one.

Simulate and run other projects from the Examples workspace: LED, keyboard, mouse, and Console are all good ones to try.

Modify the SevenSeg Example

Open up the SevenSeg project and double-click on the Handel-C source file

(seven_seg.hcc) if you haven't done so already.

Modify the program so that it displays the value of an 8-bit unsigned number (using two displays). The leftmost 4 bits are to be displayed (in hexadecimal) in the left display and the rightmost 4 bits in the right display. The 8-bit number is to increment by one at a rate of two times per second. The value of the third argument to *PalSevenSegWriteDigit()* is the setting of the decimal point you might not have noticed next to the seven segments on each display. I think you will be able to make the changes by studying the existing code even though you haven't studied Handel-C yet. But ask how to do it if you get stuck.

Simulate your new program.

Download and run it.

Now look up the PAL code for reading the states of the two pushbuttons in the PAL API Reference Manual available on the course web site. For simulation, you should use "button" numbers 8 and 9 because these correspond to slide switches that you can use to simulate holding a button in. (The simulated buttons only respond to mouse clicks, not to holding a mouse button down.) But for the real device, the two pushbuttons are switches 0 and 1. You can handle this with the standard C language conditional compilation facility, perhaps as follows:

```
#ifdef USE_SIM.
#define Switch_0 8
#define Switch_1 9
#else
#define Switch_0 0
#define Switch_1 1
#endif
```

Change your program so that the 8-bit counter counts forward when the left button is pressed and backward when the right button is pressed. The counter is to stop when neither button is pressed.

Simulate and run your new program.

Change the type of the Count variable from unsigned 8 to signed 8. What difference does it make?

Submit a Report of Your Lab Activities

Use a word processor to write a report of your lab activities that follows the format of the Lab Report Guidelines for this course, and email it to me by Midnight on the due date.