San Jose State University
Department of Electrical Engineering
EE178, Fall 2007, Crabill

Laboratory Assignment #3

Objectives

The objective of this lab is to gain familiarity with the PicoBlaze embedded processor from Xilinx. PicoBlaze is a piece of intellectual property owned by Xilinx. Xilinx makes PicoBlaze available to its customers for free, as what is called an IP core – or "core" for short. The idea is that PicoBlaze implements a function applicable in many systems, and Xilinx customers may be able to re-use it in their own designs. Xilinx provides a large variety of IP cores, many of which are free.

In this lab, you will implement an embedded processor system with several peripherals. From a hardware perspective, the system is provided and you only need to assemble the project and create a constraint file. You are encouraged to read the hardware description of the system, however, so that you understand it. This is particularly important because you will encounter PicoBlaze in future lab assignments. Figure 1 shows a block diagram of the system. The omnipresent clock and reset signals have been omitted.

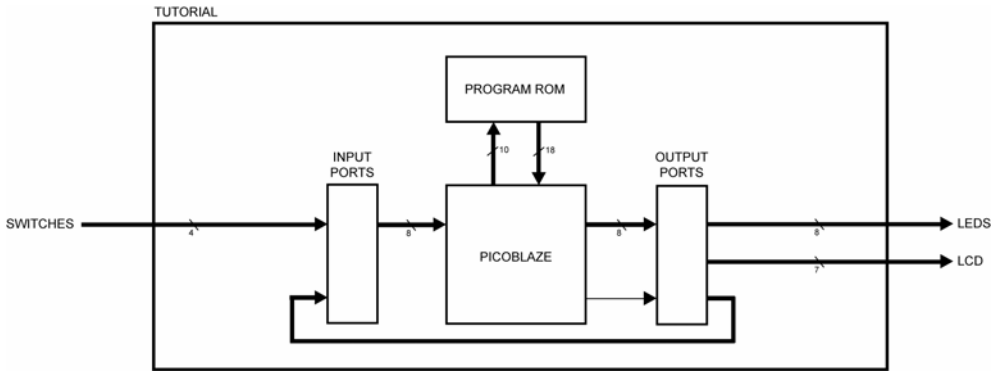

Figure 1: System Block Diagram

The main task for this lab is to write software in PicoBlaze assembly to implement two tasks. The first task is to modify the provided code to display your name on the LCD instead of the instructor's name. The second task is to write code to read the switch settings and mirror them on the LEDs.

When you successfully complete this lab, you will have gained an understanding of how to use PicoBlaze to implement a small embedded processor system.

Bibliography

This lab uses the Verilog-HDL version of PicoBlaze; you can read about PicoBlaze on the Xilinx website at http://www.xilinx.com/picoblaze. Additionally, the hardware system and some software routines used in this assignment are derived from PicoBlaze examples for the Spartan-3E Starter Kit board, also obtained from the Xilinx website. The files you need to complete the lab assignment are posted on the class website; you do not need to download anything from Xilinx.

PicoBlaze Processor

Obtain and unzip the file archive from the class website and use the resulting directory as your project folder. You can move the directory anywhere you please as long as there are no spaces in the path to the

directory. You should skim the included *PicoBlaze 8-bit Embedded Microcontroller User Guide* before proceeding.

System Description and Requirements

In this system, you are not allowed to use latches. You are allowed to use only one clock and only one asynchronous reset signal. The clock must be the 50 MHz clock signal available from the oscillator on the Spartan-3E Starter Kit board. **You will receive zero points if you do not follow these requirements.**

As shown in Figure 1, the system has a number of inputs. There are clock and reset inputs, plus a 4-bit switch input.

clk	clock signal, 50 MHz from oscillator
rst	reset signal
switches[3:0]	4-bit switch input

Also shown in Figure 1 are the outputs. There is an 8-bit LED output and several signals used to control the LCD display. One of these signals is permanently tied to logic one.

leds[7:0]	8-bit LED output
vcc	logic one, disables other devices on LCD data bus
lcd_db[7:4]	4-bit LCD data bus
lcd_e	LCD enable strobe
lcd_rs	LCD register/ram select
lcd_rw	LCD read/write select

You must successfully implement the system from the provided files, and then modify and complete a small program. The software development is split into two parts. Your final software implementation is required to display your name on the LCD and then continually echo the switch settings on the LEDs.

System Hardware Project

The first step is to create a hardware system using Project Navigator. To do this, you need to create a project and then "add existing source" files. While you were skimming the PicoBlaze documentation, you likely noticed the hardware design source files, as well as the subdirectory containing the software assembler and program source code.

In the subdirectory for the assembler, there is a template for an assembly program, software.psm, and a batch file to assemble it. This template is syntactically correct but incomplete – you will return to modify and then complete it later. Assemble the program template by double clicking on the batch file. The assembler generates an additional source file, software.v. This source file contains a BlockROM initialized with executable code generated by the assembler from software.psm.

Now, start a new project with Project Navigator. Then, "add existing source" for the system from the project directory. There are four necessary Verilog-HDL files:

1. tutorial.v (top level system module)
2. testbench.v (testbench for behavioral simulation of top level system module)
3. kcpsm3.v (the PicoBlaze sub-module)
4. software.v (the PicoBlaze software sub-module)

Using Project Navigator, add all the source files to the project in the order listed above. As you add the files, you will notice that Project Navigator actually reads the source and attempts to show you what modules it thinks are missing by marking them with a question mark in the Sources window. When you

are done with this step, no question marks should remain. At this point, also add a new implementation constraint file associated with the top level module.

System Functional Simulation

You should perform some minimal functional simulation of the system. This is important for two reasons. First, it will give you confidence your system is working properly before you implement it. Second, if the system does not behave as expected when you download it, you will have a mechanism to quickly create additional test cases to help debug the problem. The instructor will not help you debug software problems (incorrect system behavior) unless you are able to run a simulation. A simple test bench is included with the downloadable support package for this lab. Feel free to enhance the basic test bench as you see fit.

When you run a simulation, you should be able to watch the PicoBlaze processor fetch instructions from the program ROM. The software-driven LCD activity, however, is dreadfully slow. You would have to run the hardware simulator for a substantial amount of time to see waveform results that are useful. For this reason, consider running a short simulation as a sanity check to make sure you have some reason to expect the system might work, but verify the system in hardware.

System Synthesis and Implementation

Synthesize the system as you have done before. Do not forget to check the synthesis report. You should expect to see the PicoBlaze module generate a fair number of warnings about "simulation mismatch" which may be safely ignored.

Before you implement the system, you will need to edit the constraints file and assign I/O locations and properties. You must find most of the I/O locations for yourself. Use the properties shown in Figure 2.

Design Object List - I/O Pins

I/O Name	I/O Direction	Loc	Bank	I/O Std.	Vref	Vcco	Drive Str.	Termination	Slew	Delay	Diff. Type	Pair Name
clk	Input		BANK0	LVCMOS33	N/A	3.30				NONE	Unknown	
lcd_db[4]	Output		BANK1	LVCMOS33	N/A	3.30	8		SLOW		Unknown	
lcd_db[5]	Output		BANK1	LVCMOS33	N/A	3.30	8		SLOW		Unknown	
lcd_db[6]	Output		BANK1	LVCMOS33	N/A	3.30	8		SLOW		Unknown	
lcd_db[7]	Output		BANK1	LVCMOS33	N/A	3.30	8		SLOW		Unknown	
lcd_e	Output		BANK1	LVCMOS33	N/A	3.30	8		SLOW		Unknown	
lcd_rs	Output		BANK1	LVCMOS33	N/A	3.30	8		SLOW		Unknown	
lcd_rw	Output		BANK1	LVCMOS33	N/A	3.30	8		SLOW		Unknown	
leds[0]	Output		BANK0	LVCMOS33	N/A	3.30	8		SLOW		Unknown	
leds[1]	Output		BANK0	LVCMOS33	N/A	3.30	8		SLOW		Unknown	
leds[2]	Output		BANK0	LVCMOS33	N/A	3.30	8		SLOW		Unknown	
leds[3]	Output		BANK0	LVCMOS33	N/A	3.30	8		SLOW		Unknown	
leds[4]	Output		BANK0	LVCMOS33	N/A	3.30	8		SLOW		Unknown	
leds[5]	Output		BANK0	LVCMOS33	N/A	3.30	8		SLOW		Unknown	
leds[6]	Output		BANK0	LVCMOS33	N/A	3.30	8		SLOW		Unknown	
leds[7]	Output		BANK0	LVCMOS33	N/A	3.30	8		SLOW		Unknown	
rst	Input	V16	BANK2	LVCMOS33	N/A	3.30		PULLDOWN		NONE	Unknown	
switches[0]	Input		BANK1	LVCMOS33	N/A	3.30				NONE	Unknown	
switches[1]	Input		BANK1	LVCMOS33	N/A	3.30				NONE	Unknown	
switches[2]	Input		BANK1	LVCMOS33	N/A	3.30				NONE	Unknown	
switches[3]	Input		BANK1	LVCMOS33	N/A	3.30				NONE	Unknown	
vcc	Output	D16	BANK1	LVCMOS33	N/A	3.30	8		SLOW		Unknown	

Figure 2: PACE Summary Window

Once you have saved the constraint file, implement the system to make sure no errors occur. Generate a programming file and try the design in hardware – you should see the instructor's name on the LCD.

System Software Development

Now you must modify and complete the PicoBlaze assembly program to satisfy the software requirements. The program template contains a number of constant definitions for your convenience and is structured so that you can implement each of the requirements independently. You should first modify the code to display your name – this is fairly easy. Once you have that working, satisfy the second requirement which is to mirror the switch settings on the LEDs.

After you have modified/written the code to implement a given task, you must re-run the batch file to assemble the program. This will generate a new software.v file and you need to re-implement your design

in Project Navigator (simulating again is purely optional and at your discretion). Repeating these steps will be necessary every time you re-assemble the program because the resulting software.v source file will change. This iterative method should be adequate for now. In a later lab, we will learn how to try out code modifications without having to re-implement the design.

System Hardware Verification

Generate a programming file and proceed to verify the system in hardware. To verify the LCD behavior, simply use your eyes! To verify the switch to LED mirroring behavior, exercise the switches and observe the results on the LEDs. You should test that all four switches and all eight LEDs behave as expected.

When you have completed the software tasks, generate a programming file for the PROM and then load your system into the PROM.

Laboratory Hand-In Requirements

Once you have completed a working design, prepare for the submission process. You are required to demonstrate a working design which has been programmed into the PROM. Within nine hours of your demonstration, you are required to submit your entire project directory in the form of a compressed ZIP archive. Use WinZIP to archive the entire project directory, and name the archive l3_yourlastname.zip. That is "l3" as in Lab 3, **not** "13" as in thirteen. For example, if I were to make a submission, it would be l3_crabill.zip. Then email the archive to the instructor. Only WinZIP archives will be accepted. If your archive is too large, you may remove the subdirectories in the project folder.

Demonstrations must be made on the due date. If your circuit is not completely functional by the due date, you should demonstrate and submit what you have to receive partial credit.