San Jose State University
Department of Electrical Engineering
EE178, Spring 2018, Crabill

# Laboratory Assignment #7

## 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 a reference design.

In this lab, you will implement an embedded processor system with basic peripherals. From a hardware perspective, the system is provided. 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.

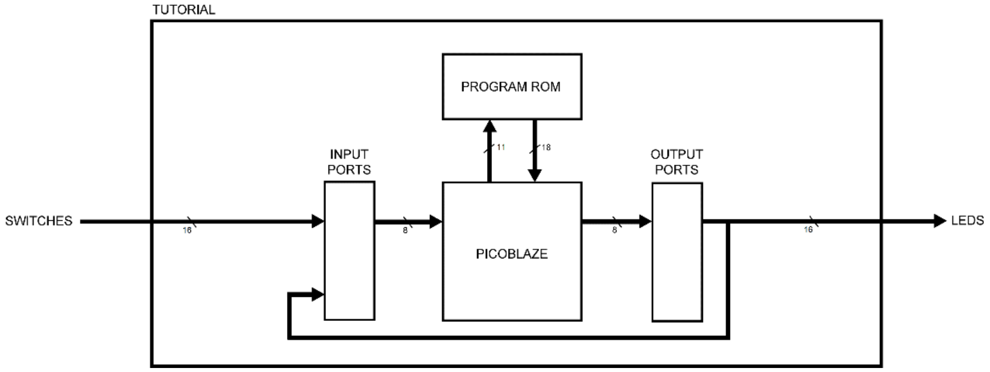


**Figure 1: System Block Diagram**

The main task for this lab is to write software in PicoBlaze assembly to monitor the switch inputs, perform an “algorithm” on the input value, and display the result 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 draws heavily from documents on the Xilinx website http://www.xilinx.com. I would like to thank Xilinx for making this material available. This lab is effectively a customized introduction to Xilinx Vivado using Verilog-HDL and the Digilent Basys3.

This lab uses PicoBlaze; you can read more about it at http://www.xilinx.com/picoblaze. Additionally, the hardware system is derived from PicoBlaze examples, 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.

## System Description and Requirements

In this design, you are not allowed to use latches. You are allowed to use only one clock. The clock must be the 100 MHz clock signal available from the oscillator on the board. You will receive zero points if you do not follow these requirements.

The system has a single clock input. It has an additional input to receive signals from 16 switches and an output to control 16 light emitting diodes.

| | |
|---|---|
| clk | clock signal, 100 MHz from oscillator |
| switches[15:0] | switches |
| leds[15:0] | light emitting diodes |

You must successfully implement the system from the provided files, and then modify the small assembly program that has been provided for you. Your final result is required to implement the algorithm described in the comments of the assembly program source.

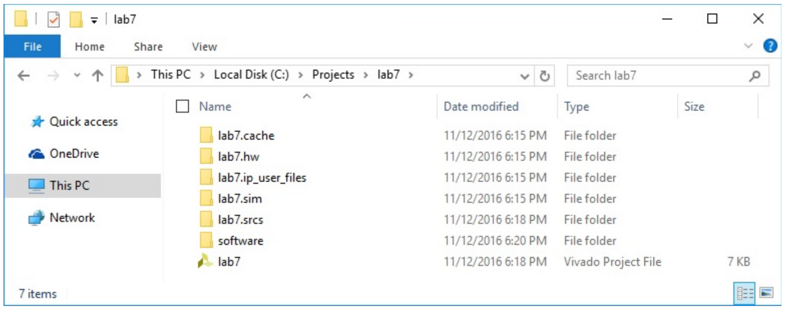
## System Hardware Project

Create a project using the following files posted on the class website. Instead of creating source files and copying text, use the add source capability to directly add the files to the project. When adding files, you have the option to "Copy sources into project" which is advisable, so that the files do not get separated from the project directory. This option is a checkbox at the bottom of the add sources dialog, keep your eye open for it. One file, however, we will handle differently.

| | | |
|---|---|---|
| 1. | testbench.v | (top level testbench) |
| 2. | tutorial.v | (top level design) |
| 3. | tutorial.xdc | (top level design constraint file) |
| 4. | kcpsm6.v | (embedded processor) |

The source file that requires different handling is the software.v file, which is located in a software directory along with other supporting files. The software.v file is an implementation of a ROM containing the assembled PicoBlaze program. You will not edit the software.v file manually – it is generated from the assembly source file software.psm using the kcpsm6.exe assembler program.

You will need to add the software.v file as source to your project, but before you add it, copy the entire software directory into your project as shown in Figure 2. This will ensure all your design sources, including software.psm and software.v, reside within your project.



| Name | Date modified | Type | Size |
|---|---|---|---|
| lab7.cache | 11/12/2016 6:15 PM | File folder | |
| lab7.hw | 11/12/2016 6:15 PM | File folder | |
| lab7.ip_user_files | 11/12/2016 6:15 PM | File folder | |
| lab7.sim | 11/12/2016 6:15 PM | File folder | |
| lab7.srcs | 11/12/2016 6:18 PM | File folder | |
| software | 11/12/2016 6:20 PM | File folder | |
| lab7 | 11/12/2016 6:18 PM | Vivado Project File | 7 KB |

**Figure 2: Copy Software to Your Project Directory**

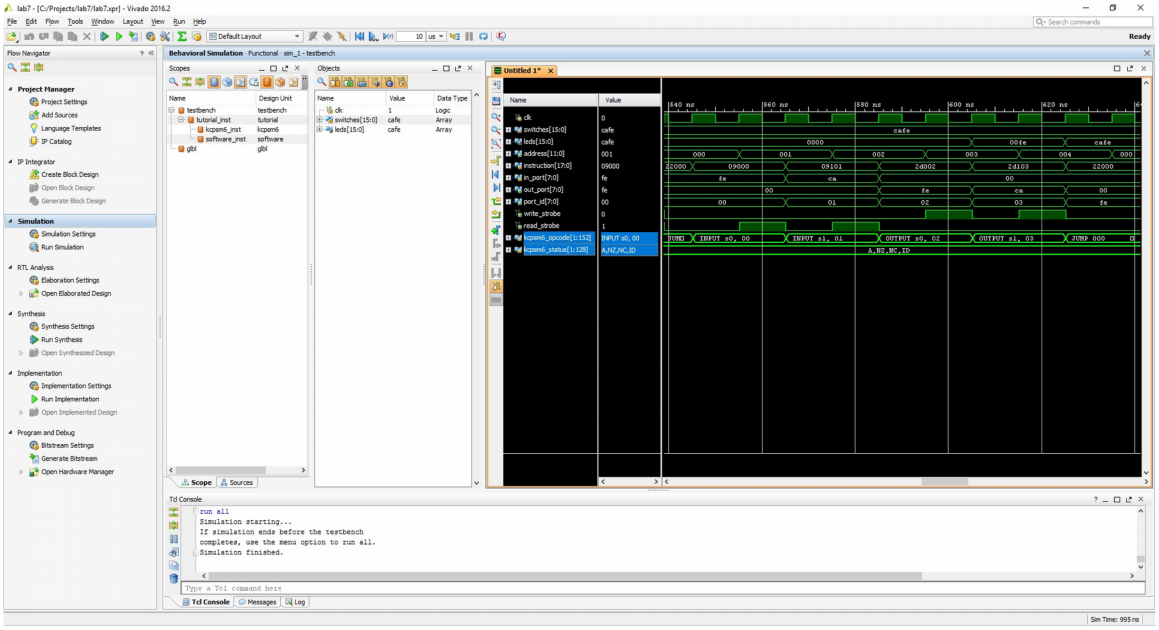
Now, add the software.v file as source to your project. Make absolutely sure you select the software.v file that is inside the software directory, inside your project. For this one file, un-check the option to "Copy Sources into Project". By unchecking this option, Vivado will NOT copy the software.v file into its project source repository, but instead will reference it directly from where it exists.

Directly referencing the software.v file from where it exists will be important, so that you can make edits to the software.psm file and re-assemble it – creating a revised software.v file – without needing to move the revised software.v file or its contents anywhere.

## 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 and execute instructions from the program ROM. In fact, if you probe into the PicoBlaze processor and add the “kcpsm6_opcode” and “kcpsm_status” signals to the waveform, then right click on them to change their radix to ASCII, you can easily observe the operation of your program. See Figure 3 for an example.


lab7 - [C:/Projects/lab7/lab7.xpr] - Vivado 2016.2
File Edit Flow Tools Window Layout View Run Help
Default Layout
Ready
Flow Navigator
Behavioral Simulation - Functional - sim_1 - testbench
Untitled 1*
540 ns
560 ns
580 ns
600 ns
620 ns
Sim Time: 995 ns

| Name | Value |
|---|---|
| clk | 0 |
| switches[15:0] | cafe |
| leds[15:0] | cafe |
| address[11:0] | 001 |
| instruction[17:0] | 09000 |
| in_port[7:0] | fe |
| out_port[7:0] | fe |
| port_id[7:0] | 00 |
| write_strobe | 0 |
| read_strobe | 1 |
| kcpsm6_opcode[1:152] | INPUT s0, 00 |
| kcpsm6_status[1:128] | A,NZ,NC,ID |

```
run all
Simulation starting...
If simulation ends before the testbench
completes, use the menu option to run all.
Simulation finished.
```

**Figure 3: Functional Simulation**

As provided, the program (as expressed in software.psm and assembled into software.v) reads from the switch input ports and writes the values to the LED output ports – essentially, the tutorial design is behaving as a virtual wire, connecting the switches to the LEDs, through the execution of program.

## System Synthesis and Implementation

Synthesize and implement your design exactly as you did in the tutorial. Do not forget to check the reports. As a general practice, you will want to review all errors and warnings. These point to areas of concern that you should either address or justify. If the design fails one or more timing specifications the reports will indicate this is the case.

Test your design in hardware, you should observe one-to-one correspondence between switch settings and LED illumination. Does the circuit behave as you expect? If it does not, seek assistance.

## 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 you can insert your modifications adjacent to the comments that state the functional requirements of your modified program.

Edit the assembly source file software.psm in a text editor. After saving your modifications to the assembly source, simply “drag and drop” the software.psm file onto the kcpsm6.exe icon. This will update the software.v file referenced in your project, unless an error occurs during assembly. Make sure to review the assembler messages to confirm your modified program has assembled without error.

After changing the program and assembling it, you can perform functional simulation or reset the design run and re-generate a bitstream to test in hardware. Every time you make modifications to software.v (by running the assembler) you must reset the design run and re-generate a bitstream to test in hardware. Once you are confident the design works properly, demonstrate your final result to the instructor.

## Laboratory Hand-In Requirements

Once you have completed a working design, prepare for the submission process. You are required to demonstrate a working design. Within four 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 lab7_yourlastname_yourfirstname.zip. For example, if I were to make a submission, it would be lab7_crabill_eric.zip. Then email the archive to the instructor. Only WinZIP archives will be accepted.

Demonstrations must be made on or before the due date. If your circuit is not completely functional by the due date, you should turn in what you have to receive partial credit.