Computer Systems Design - Lab 6

Handed Out: Thursday, 12 Feb. 2015
Due: Thursday, 26 Feb. 2015 by 11:00 AM
Submit via USF Canvas
This is a team assignment. Teams can have up to 3 members
Late submissions will not be accepted

Objective:

To design and synthesize a loopback system using a PicoBlaze microcontroller, and to implement functionality using Assembly code.

Problem:

You are to design and build a simple loopback system capable of echoing data via a serial RS232 connection, as well as echoing data from the switches to the LEDs.

Description:

Consider a simple"loopback" system as in Fig. 1. In such a setup, a PC can send a message which is returned (looped) back to PC by the processor. Such a loopback test is helpful to test the processor is alive or not.

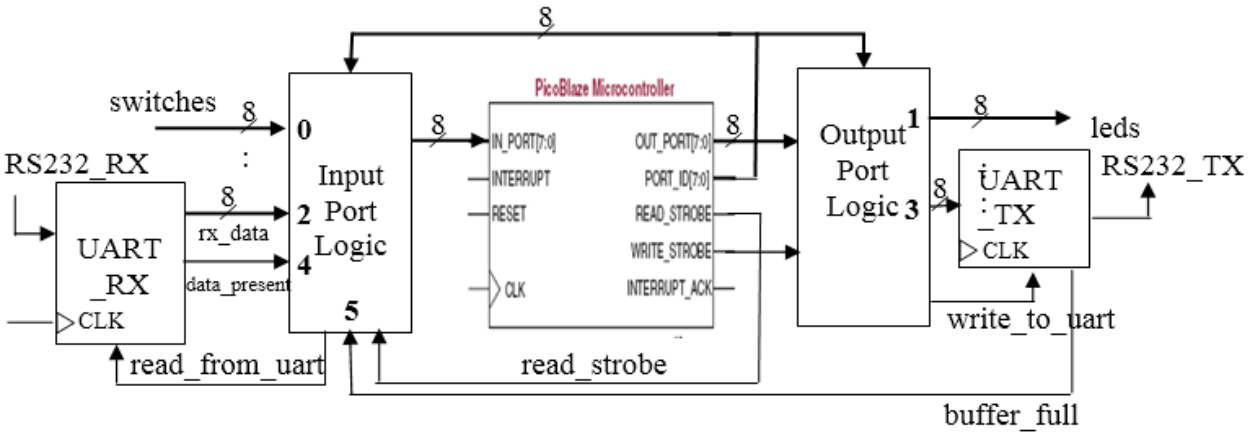

Figure 1: A simple loopback system

A PC (not shown) interfaces with PicoBlaze processor using an RS232 (serial) connection[1]. Let us examine the case when the PC acts as a transmitter. In this case the data arrives serially on RS232_RX line. But we know that PicoBlaze accepts only byte-size data. So we need to convert the serial data into parallel (8-bit) data. In such a case, a UART (Universal Asynchronous Receiver Transmitter) block can help (Figure 2). UART_RX does the serial-to-parallel conversion i.e., the data arriving on RS232_RX line is buffered in a 16-byte buffer. Whenever new data is presented on rx_data output, UART_RX will assert data_present i.e., data_present = 1 in other words data_present validates rx_data. The Input Port Logic block consists of glue logic that connects UART_RX to the IN_PORT. After PicoBlaze reads the data, it can request the next data item by asserting read_by_uart signal.

Now, let us consider the case when PicoBlaze acts as a transmitter i.e., PC is the receiver. In this

[1]The Atlys uses a Serial-over-USB protocol, utilizing the second micro USB port on the board.

Figure 2: UART Transmit and Receive Blocks - High Level View

case, PicoBlaze puts out 8-bit data while PC is expecting serial data. Again, UART (UART_TX in Fig. 1) comes to our rescue! The output bus with id=3 is connected to UART_TX. Output Port Logic block interfaces PicoBlaze with leds and UART_TX.

In this system, we can also manually provide 8-bit data via 8 switches (switches bus in Fig. 1). PicoBlaze can write 8-bit data to drive 8 leds (leds bus in Fig. 1).

Recall that PicoBlaze can accept up to 256 input ports each of which can addressed by 8-bit PORT_ID bus. Similarly, it can drive up to 256 ports and the address of the port being driven appears on PORT_ID bus. In Fig. 1, for example, the id of the input port, switches, is 0.

Note that for the sake of clarity, we have not shown the clock and reset signal connections.

- **cold_start:** Write code to output a message Welcome to Loopback! to the serial port. You need to encode the message in ASCII format. *Caution: UART_TX buffer size is only 16 bytes!!*
- **led_echo:** Write code to read switches and write it, inverted, to the LEDs.
- **rs232_echo:** Write code to check if a byte has been received by UART_RX. If so, send it back to PC via UART_TX.
- **Behavior:** The design must be able to perform all of the above actions **simultaneously**.

Design Constraints: The designs must properly function given the following constraints:

- The design must both function correctly in simulation *and* on the FPGA board.
- The RESET button must provide a global reset functionality. This includes resetting the microprocessor.
- The 100MHz Clock is used as the system clock.
- Every time the design is initialized (reset), it runs cold_start.
- Until it is turned off or reset, it performs the following two actions continuously:
  - Echo Switch values to LEDs.
  - Echo RS232 data back to PC.
- It must do this indefinitely.

Lab Report: You must submit a written lab report. The lab report will consist of (at least) the following sections:

- **Cover:** Include a list of team members (with U#'s) and work distribution for each team member.
- **Introduction:** In your own words, briefly describe the lab assignment.
- **Design:** Briefly describe the structure and function of your design. Discuss your design decisions and any problems you encountered during the design process.
- **Testing:** Discuss how you verified the functionality of the design. This should include a description of your testbench and test vectors, and a brief justification of the completeness of the tests you chose. Also discuss any problems you had with the testing process.
- **Synthesis:** Discuss the synthesis process and results, including any warnings or errors encountered. Describe the function of your design on the FPGA board and discuss how you verified the functionality. Be descriptive.
- **Results:** Discuss the results of the design process, testing and the assignment overall. This should include a discussion of any simulation results and an overall analysis of your design.
- **Conclusion:** Briefly summarize your accomplishments and discuss any failures. Briefly discuss your thoughts, concerns, and feedback for this assignment. If you choose, you may rant about the software, programming language, the assignment itself, or the TAs responsible for the lab.

All reports MUST be submitted in PDF format. No exceptions.

Deliverables: Submit your report PDF and the code ZIP file separately on Canvas. The PDF should be named <TeamName>_lab06_report.pdf. To generate the code ZIP file:

- Select **Project** → **Cleanup Project Files...** from the menu.
- Select **OK** on the pop-up box.
- Select **Project** → **Archive...** from the menu.
- On the new window, make sure **Exclude Generated Files From Archive** is *selected*.
- Choose a location for the archive, name it <TeamName>_lab06_src.zip.
- Select **OK**

Make sure to add your PicoBlaze assembly code files to the src zip file.

Insert any images and waveforms into the report PDF. Submit the report PDF and source code ZIP separately on Canvas. If you wish, you may include copies of all images in a separate ZIP file, submitted through Canvas. Name it <TeamName>_lab06_images.zip.

No re-grading will be done for failure to follow the guidelines. Points will be lost, never to return.

This lab assignment requires a live demonstration. **Demonstrations will occur during lab time on the day the assignment is due.**

Lab 5 - Step by Step Instructions

It's strongly recommended that you perform the steps in the given order below.

1. **Download and Unzip**
   (a) In your home directory, create a new folder 'lab5'.
   (b) Download the loopback.zip file from Blackboard into 'lab5'.
   (c) Unzip the file.
2. **Create ISE Project**
   (a) Create a new ISE project.
   (b) Ensure that you are using the correct settings (Tutorial 1).
   (c) Add copies of **all** Verilog source files and the UCF file from the loopback directory.
3. **Edit 'loopback.v'**
   (a) Complete Worksheet 1 on loopback.v
   (b) Open 'loopback.v' and fill in the blanks.
4. **Edit 'loopback.UCF'**
   (a) Complete Worksheet 2 on FPGA Pin Assignments.
   (b) Open 'loopback.UCF' and fill in the blanks.
5. **Write Assembly Code**
   (a) Open the "Assembler" folder in "loopback."
   (b) Open the 'program.psm' file with a text editor.
   (c) Enter your assembly code for task 1. Note that the ASCII codes are already entered as constants. For example, if you want to load the ASCII code for 'h' then you can write: LOAD sX, ascii_h.
   (d) To assemble the code, the syntax is: ./picoasm -i program.psm -t ROM_form.v
   (e) It will generate a file, 'program.v'. Add a copy of this file to your project. It contains the assembled code for PicoBlaze.
6. **FPGA Synthesis and Implementation**
   (a) Synthesize and generate the lookup.bit file. (Tutorial 4)
   (b) Load the design onto the board. (Tutorial 4)
7. **RS232 Cable Connection**
   (a) Connect the RS232 serial cable to the back of the C4 lab computer, as well as to the XUP board.
   (b) Don't break the computer.
8. **Connect to Serial Port**
   (a) To open the serial terminal, run 'minicom' from a terminal.
   (b) Test the RS232 capabilities of your design (cold start, loopback).
   (c) To exit, Press: 'ctrl+A' then 'ctrl+z' then 'x'.
9. **Finish**
   (a) Write assembly code for led_echo program. (Task 2)
   (b) Write assembly code for rs232_echo program. (Task 3)
10. **(Optional) Create and load PROM file.**
    (a) Do this step only after you have finalized your design.
    (b) The tutorial will be available on Blackboard.
    (c) This allows the design to automatically be loaded when the board is powered on.

Worksheet 1

1. What is the unstruction bus width for the ROM?

2. For the UART 16x_bit_rate counter, the maximum baud count using the following formula:
   $max\_baud\_count = \frac{f_{clk}}{16 * req\_baud\_rate}$
   If f_{clk} = 100MHz and the required baud rate is 9600, what is the maximum baud count?

3. Write the boolean expression for the **write_to_uart** signal.

4. Write the boolean expression for the **write_to_leds** signal.

5. What are the port_id values for the following input ports:
   (a) switches:
   (b) data_rx:
   (c) data_present:
   (d) buffer_full:

6. Write the boolean expression for the **read_from_uart** signal.

7. Continue with step 3 (b).

Worksheet 2

Fill in the pin locations for the following components/signals. Refer to the FPGA User Guide and master_xupv5-lx110t.ucf available on Blackboard.

1. Clock (100MHz):
2. Push Button UP (for reset signal):
3. LED 0:
4. LED 1:
5. LED 2:
6. LED 3:
7. LED 4:
8. LED 5:
9. LED 6:
10. LED 7:
11. SW 0:
12. SW 1:
13. SW 2:
14. SW 3:
15. SW 4:
16. SW 5:
17. SW 6:
18. SW 7:
19. RS232 TX (FPGA_SERIAL1):
20. RS232 RX (FPGA_SERIAL1):

Now continue with Step 4 (b).