Name:
Digital System Design I
CST 231
Lab Assignment 6: UART Receiver
Due: Check course website

You should work with a partner for this lab assignment. You are allowed to use course materials, the internet, and ask the instructor or teaching assistant for limited assistance. You must use your own board(s). Please review the entire assignment before starting.

You have one week(s) to complete this lab. It is your responsibility to manage your time in such a way that your lab is complete before the due date without requiring last minute instructor assistance or checkoff. While instructor will attempt to assist and check off the lab at the last minute, there is no guarantee. You must demo this lab and submit the lab report before due date.

1 Introduction

In this lab, you will build a UART receiver. The UART receiver shall adhere to the following specifications:

1. 9600 baud
2. 2 stop bits
3. 1 start bit
4. 8 data bits
5. Even parity bit

Your project should be organized in the following manner:

1. Receiver module
2. Clock divider for receiver

2 Instructions

2.1 Task 1: Develop the clock divider for receiver

1. Your master clock oscillator frequency varies. Make sure you determine your oscillator frequency. use the clock divider class notes to determine the size of register required, as well as the count-up value.
2. Develop a clock divider for the transmitter. The clock should have a 50 percent duty cycle. Recall that the receiver will need to use 8X or 16X oversampling. Choose one and implement the clock divider.
3. After developing the clock divider, verify with instructor. Typically, I would have asked you to look on the oscilloscope. If you do have an oscilloscope, you can probe a pin to verify clock frequency, although it is not necessary this term.

2.2 Task 2: Develop the receiver

The receiver must be demoed with the following conditions:

1. Send data to FPGA using Putty or Hyperterminal and a USB to UART converter
2. 9600 baud, 2 stop bits, even parity (2021)
3. Data must be shown using the 7-segment display.
4. If 'A' is sent to the FPGA, the 7-segment should show the ASCII value '41'. It must handle a-z, A-Z, and 0-9.

   http://www.asciitable.com

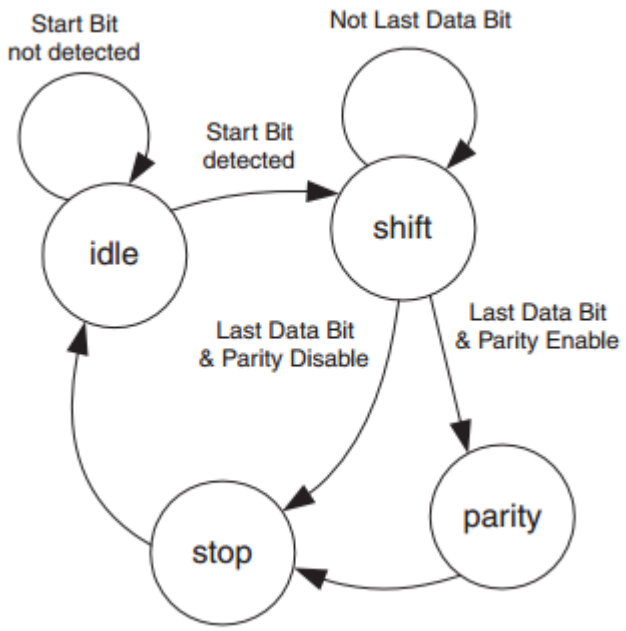
The receiver is a bit trickier than the transmitter. The first item to note is that you must be oversampling at a rate of 8x to 16x, so you will be using the clock divider you developed in part 2. Another thing to realize is that the LSB will be received first by the receiver, so if your register is called dataReg[7:0], you should store the first bit in dataReg[7]. ModelSim is your friend.

1. Please note that you are free to design your own state machine and that the state machine diagram pictured may not be exactly the one you require.
2. Follow the state machine diagram as given by Lattice Semiconductor below as a starting point.
3. Don't forget about parity. Set an LED or some other indicator when parity error is detected.
4. The receiver module should have the following items in the module declaration:

```
module rx(
input dataIn // Serial data in from PUTTY through USB-UART converter
input clk  //This is your 8x or 16x oversample clock
output dataReady // This output indicates that data is available from receiver.
output [7:0] data // This 8-bit parallel bus is where the receiver outputs
//received data to another module.
);
```

5. Assuming that 8x sampling is chosen, start in the idle state until the line drops to '0' indicating a start bit.
6. It would be tempting to immediately say go to shift state, but this would be incorrect. Once the line drop is detected, count 3 more cycles and verify that it is a start bit. Remember that sampling must take place at midbit. Alternatively, count past the midbit of the start bit directly to the midbit of first data bit. (It would be helpful if you drew it out yourself to verify your understanding).
7. Count four more cycles to get to the start of the first data bit.
8. Go to shift state.
9. In shift state, count 4 cycles and take a midbit sample. Store this value in the dataReg or whatever was chosen as the data register, keeping in mind that LSB is received first. Count another 8 cycles to get to the midbit of the next sample and keep repeating until all data bits have been received.
10. After last data bit is received, go to stop bit state.
11. Return to idle once two stop bits are detected in stop state.

12. Don't forget you also need to calculate parity.



(From Lattice UART App Note)

3 Submission

Electronically submit the lab.

1. Submit your cleaned and zipped project files.
2. Submit the following in a Word document.

   Calculations required for the clock divider.

   Include your RTL schematic.

   Include a short one paragraph explanation of how your design functions. How does your oversampling function? How does your RX function.

   Include a system block diagram indicating the layout of your code (modules). This may include RX module, clock divider, clock select module, etc.

   Include your detailed state machine diagram for the UART receiver.