CMPE 691/491 - Homework #1

Fall 2014

Start assignment as soon as possible. Work individually, but you can ask your classmates for help when you get stuck, with consideration to the **course collaboration policy (please read it in the course website)**. Please send me email if something isn't clear and I will update the assignment. Changes are logged at the bottom of this page.

Before getting started, you should go through the verilog notes located under Course Readings on the course home page.

A paper copy of everything and electronic copies of all your code and testing files (all in one zipped file) **are due at the beginning of class on the due date**.

Notes:

- [15% of points] Clearly state whether your design is fully functional, and state the failing sections if any exist.
- Make sure your design and code are easily readable and understandable (clear and well commented).
    - Up to 5% extra credit will be given for especially thorough, well-documented, or insightful solutions.
- *** Where three '*'s appear in the homework, perform the required test(s) and turn in a printout of either:
    1. a table printed by your verilog testbench module listing all inputs and corresponding outputs,
    2. an Isim waveform plot which clearly shows (labeled and highlighted) corresponding inputs and outputs, or
    3. a section of testing code which **clearly** compares the designed circuit and a simple reference circuit, and two short cut & paste sections of text from your simulation (one for pass, and one for fail where you purposely make a slight change to your reference code to make it fail) that look something like this:
    `Error: input=0101, out_module=11110000, out_ref=11110001`

    In all three options, each test case must be marked whether the output is correct or not.

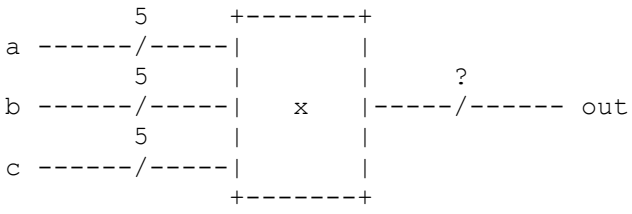
Keep "hardware" modules separate from testing code. Instantiate a copy of your processing module(s) in your testing module (the highest level module) and drive the inputs and check the outputs from there.

Problem 1. [35 pts]

Design and write the verilog for a block that multiplies three 5-bit numbers into a 2's complement output that is sufficiently large to represent all inputs but with no extra bits. The three inputs are as follows:

- `a` is in 2's complement 1.4 format
- `b` is in 2's complement 2.3 format
- `c` is in unsigned 5.0 format

a) [2 pts] How many bits does the output have and where is its decimal point?
b) [3 pts] What is the output's minimum attainable negative value (most negative)?
c) [3 pts] What is the output's minimum attainable positive value?
d) [3 pts] What is the output's maximum attainable positive value?
e) [20 pts] Test the circuit over at least 15 input values (including all extreme cases).
Turn in ***, option 1



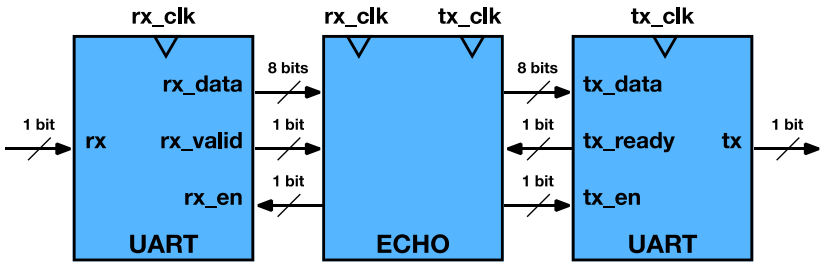
Problem 2. [65 pts]

This project will introduce students to UART communication between a PC and FPGA. This will prepare students for further assignments that require UART. This project will consist of a simple echo messaging, in which data sent from the PC will be echoed back from the FPGA.

Detail Description

1. Use MATLAB to communicate between the PC and FPGA. Below is an example of MATLAB commands to configure a serial port, open it up, echo a message, and close the serial port.
    - sp = serial('Com port number,'BaudRate',115200);
      (for mac or linux machine you can you '/dev/tty.usbserial-A4013DFD' for comport)
    - fopen(sp)
    - fprintf(sp,'Sending a message to FPGA via UART\n');
    - fread(sp)
    - fclose(sp);

2. For FPGA implementation you need to design the echo module and UART interface. A block diagram of the design is shown below:



- For UART (universal asynchronous receiver-transmitter):
    - Download a Verilog implementation of a UART from the internet: http://www.asic-world.com/examples/verilog/uart.html
    - Specs: 8-bit data, 1 stop bit, no parity bit, no hardware or software flow control.

- For FPGA implementation and test, you need to define the UART and clk pins in the UCF file.
  NET "rx_in" LOC = AG15;
  NET "tx_out" LOC = AG20;
  NET "clk" LOC = AH15;
  NET "clk" TNM_NET = "clk";
  TIMESPEC TS_clk = PERIOD "clk" 10 ns HIGH 50 %;
  NET "reset" LOC = U8;
- You need to generate rx-clk and tx_clk from a reference clk that is generated on board according to your UCF file. Tx_clk is set as baudrate e.g. 115200 and rx-clk is 16 times faster. You need to find the appropriate counter numbers to divide the reference clk to generate both tx_clk and rx_clk.

What to return

Fully tested design with testbench and hardware demonstration.

- Verified design in simulation
  - First get the UART interface working correctly with a simple testbench. Then integrate the echo module and test sending a message in a separate testbench. Note: Each design module must be accompanied by a testbench and be verified. A testbench for the UART is provided but you need to modify it in order to make it compatible with your design module.

- Verified design in Hardware and demonstration
  - Use MATLAB and serial interface or any other serial protocol software to send and receive data between PC and FPGA board.