LAB Assignment #2 for CMPE 415

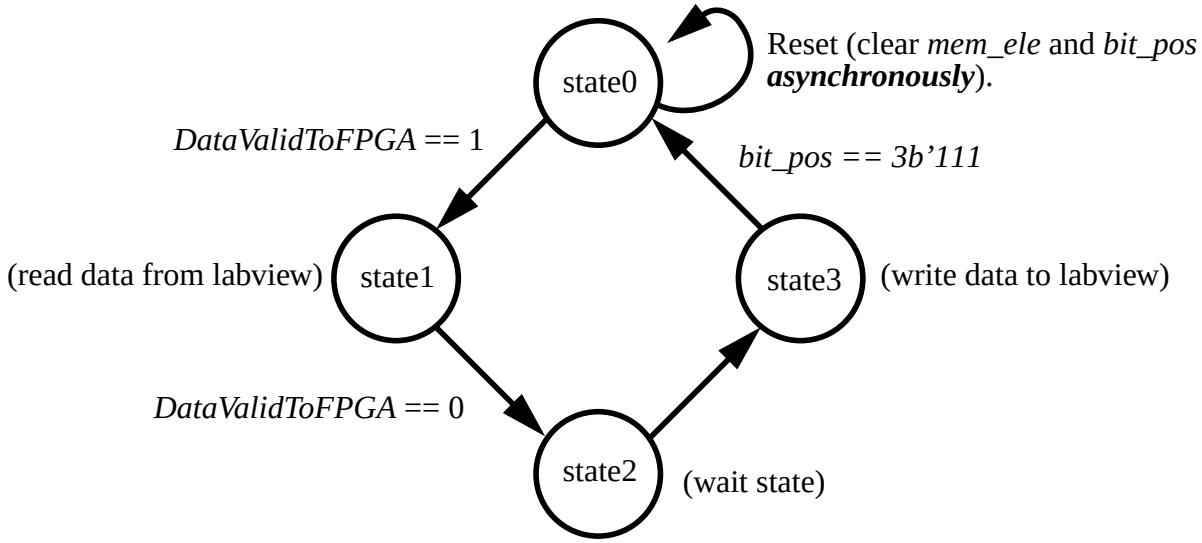
Assigned: Fri., Oct 5, 2007
Due: Thur., Oct 16,, 2007

Description: Write a state machine that reads 8 data bits from a serial data stream and then writes them back.

This state machine will eventually serve as the data transfer engine for the project. The state machine is to read the sequence of values placed on the *DataToFPGA* input port (1-bit value), one at a time, save them internally and then write them back in the same order (not reverse order). I've given the timing diagram generated from simulations of a state machine that I've written on my web page: "**Parallel port timing diagram**". Your state machine should generate a timing diagram very similar to this -- I've indicated where there can be discrepencies. The timing diagram includes the port signals as well as some of the internal signals that I use within my Verilog code. I've also included a template and a test bench on my web page that you should use "**Parallel port template**" and "**Parallel port test bench**". The signals and their meaning are given as follows:

- **ClkToFPGA**: The clock is generated by labview and sent to the FPGA through the parallel port. The FPGA should sample data on *DataToFPGA* on each rising edge of *ClkToFPGA* when *DataValidToFPGA* is high. When returning data to labview, the FPGA should write a new value on the *DataFromFPGA* wire on the falling edge of the *ClkToFPGA* so that it is stable for sampling by labview on the rising edge.
- **Reset**: The reset signal for the FPGA (sent by labview through the parallel port). Clear memory and return to state0. This is an asynchronous signal and should be implemented as an asynchronous reset on the registers in the schematic that is generated by ISE.
- **DataValidToFPGA**: Signal from labview that indicates that data on the *DataToFPGA* wire is valid and can be sampled on the next rising edge of *ClkToFPGA*. This signal is asynchronous with the *ClkToFPGA* signal. Once it is asserted, the FPGA should sample the *DataToFPGA* wire on the next rising edge of *ClkToFPGA*.
- **DataToFPGA**: 1 bit port for data sent from labview to the FPGA.
- **DataValidFromFPGA**: Signal from the FPGA to labview that indicates that data on the *DataFromFPGA* wire is valid and can be sampled and saved by labview on the next rising edge of *ClkToFPGA*. This wire can only be asserted **synchronously** since it is generated by the FPGA -- I show it going high with a rising edge of *ClkToFPGA* in the timing diagram. I then insert a wait cycle to give labview time to sample the first value, which it does on the NEXT rising edge of *ClkToFPGA*. This line remains high until the last of the 8 bits is sent back to labview. See the timing diagram.
- **DataFromFPGA**: 1 bit port for data sent from the FPGA to labview.

Your state machine should include 4 states: state0 through state3. state0 is the initial state and should become the current state asynchronously when Reset is asserted or synchronously following the process of writing the array back to labview. See the state transition diagram. When *Reset* is asserted, you should clear memory -> at this point, the 8-bit register for the data and the address pointer (I call them *mem_ele* and *bit_pos* in the timing diagram).



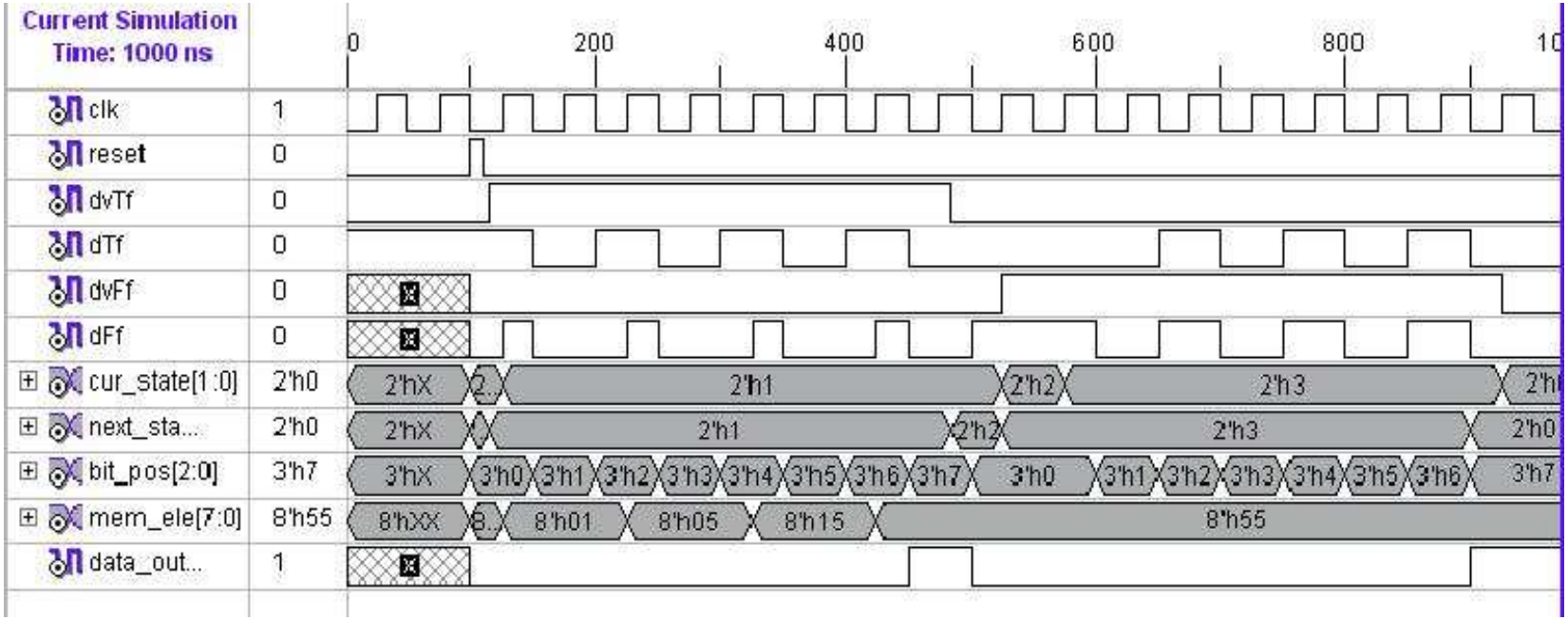
Laboratory Report Requirements:

1) Turn in a commented copy of your verilog code.
2) Turn in the schematic diagram that represents the synthesized structural representation of the behavioral code.
3) Run a simulation and turn in a timing diagram of your state machine similar to what I have posted.

Since this lab involves only simulation experiments, each of you must work on it **independently** and turn in a separate report, i.e., this is NOT a group project.

Grading:
60% Verilog code correctly specifies desired functionality.
10% Meaningful comments in verilog code.
30% Meaningful simulation results.

TIMING DIAGRAM ON NEXT PAGE


Current Simulation Time: 1000 ns
0
200
400
600
800
clk
1
reset
0
dvTf
0
dTf
0
dvFf
0
dFf
0
cur_state[1:0]
2'h0
2'hX
2'h1
2'h2
2'h3
next_sta...
2'h0
bit_pos[2:0]
3'h7
3'hX
3'h0
3'h1
3'h2
3'h3
3'h4
3'h5
3'h6
3'h7
mem_ele[7:0]
8'h55
8'hXX
8'h01
8'h05
8'h15
data_out...
1