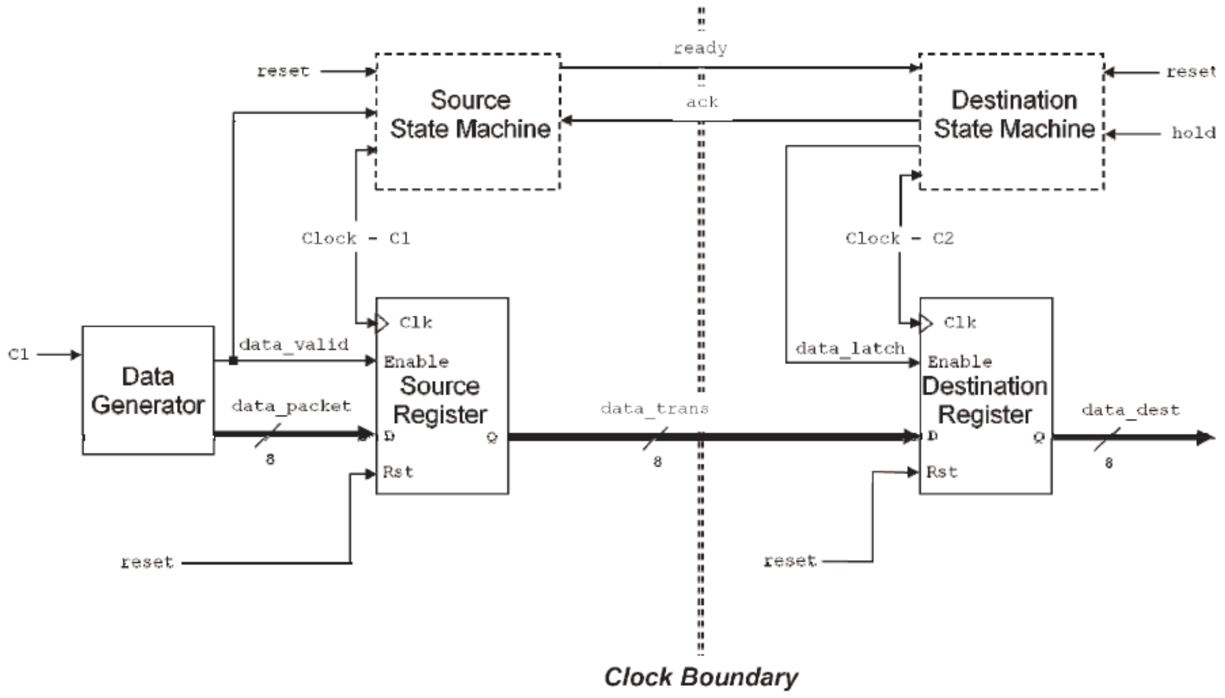
EGRE 427 Advanced Digital Design
Homework #4 – crossing clock boundaries

This homework must be pledged as your own (individual) work – write it out and sign it.

For this lab/homework assignment, you will be designing a scheme for transmitting data across an asynchronous clock boundary using a 4-state handshaking protocol. The schematic of the system is shown below.



1. The data generator component simulates some system that is generating data synchronous to clock **C1**, but at some rate which is unknown.

   a. That is, the data is generated on a **C1** clock edge, but the number of **C1** clock cycles between subsequent data packets is unknown.

   b. Along with the data, the system generates a *data_valid* signal that is asserted high (‘1’) when the data is valid.

   c. The data from the data generator is latched into the source register using the *data_valid* signal.

   d. This *data_valid* signal also indicates the start of the next data transmission across the clock boundary.

2. On the destination side, which is synchronous with clock **C2**, there is a destination register that is controlled by a signal called *data_latch*.

    a. This signal must be asserted, synchronous with **C2**, during the period when the data from the source register, on the *data_packet* signal, is valid.

    b. In addition, there is a control signal on the destination side called *hold*.

    c. When the *hold* signal is asserted, it indicates that part of the system down-stream from the Destination Register is not ready to accept new data.

    d. If *hold* is asserted, the Destination State Machine must complete the current data transfer between itself and the Source State Machine, if one is in progress, <u>but it can not start a new data transfer until the *hold* signal is deasserted.</u>

3. There are two control signals, *ready*, and *ack*, that cross the clock boundary with the data.

    a. These signals are to be used to control the actual passing of the data using the 4-state handshaking protocol.

    b. Because these signals cross a clock boundary, they **MUST** have techniques applied to them to avoid metastability problems.

4. The VHDL source code for this problem is contained in the directory

    /mentor/examples/vhdl/egre427_hw3.

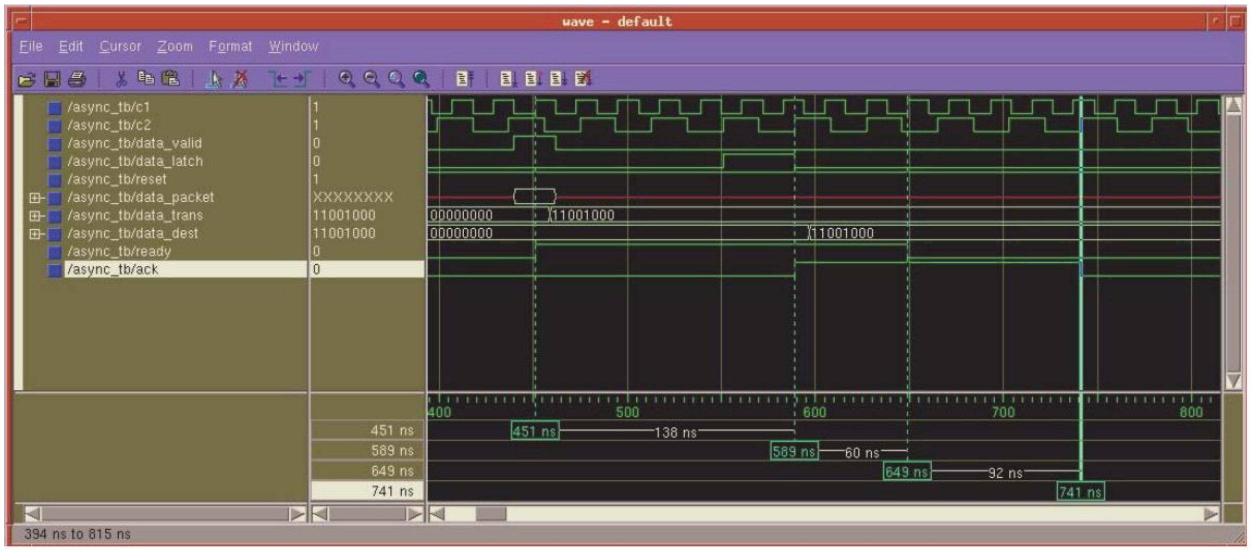
    a. There are three VHDL files there. The file `reg8.vhd` contains the behavioral description of the 8-bit registers used for the source and destination registers shown above.

    b. The file `data_gen.vhd` contains the behavioral description of the data generator component.

    c. Finally, the file `async_tb.vhd` contains the test bench that connects those three components as shown above.

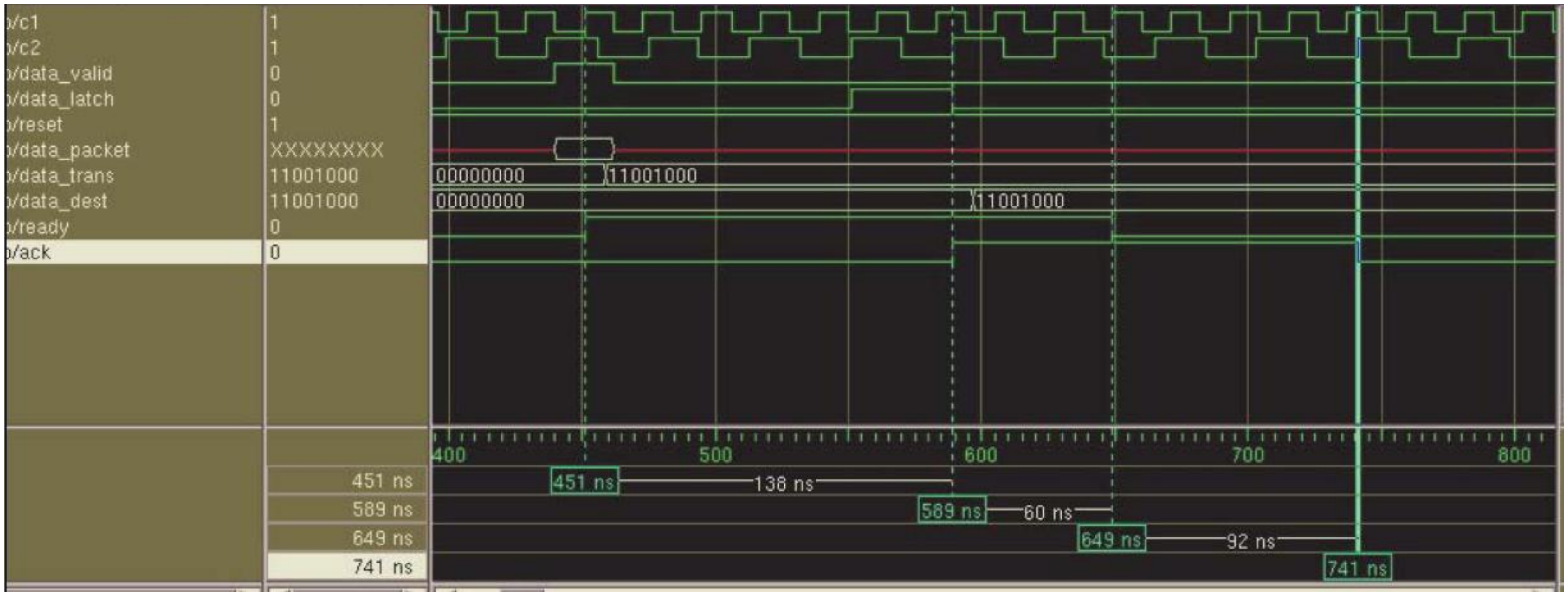
5. Your assignment is to design and implement, in synthesizable VHDL, the source and destination state machines shown in the figure above, instantiate them in the asynch_tb, connected as shown, and run the simulation showing proper 4-state data transmission across the clock boundary.

   a. A correct result waveform will look like the one below. Note the 4-states as described in class can clearly be seen on the *ready* and *ack* signals, and the *data_latch* signal is asserted while the *ready* signal is still valid from the source state machine.



```
/async_tb/c1           1
/async_tb/c2           1
/async_tb/data_valid   0
/async_tb/data_latch   0
/async_tb/reset        1
/async_tb/data_packet  XXXXXXXX
/async_tb/data_trans   11001000
/async_tb/data_dest    11001000
/async_tb/ready        0
/async_tb/ack          0

451 ns
589 ns
649 ns
741 ns
394 ns to 815 ns
```


/c1
/c2
/data_valid
/data_latch
/reset
/data_packet
/data_trans
/data_dest
/ready
/ack
1
1
0
0
1
XXXXXXXX
11001000
11001000
0
0
00000000
11001000
00000000
11001000
400
500
600
700
800
451 ns
589 ns
649 ns
741 ns
451 ns
138 ns
589 ns
60 ns
649 ns
92 ns
741 ns

6. Once your simulation is functioning correctly, you must use the Precision synthesis tool to synthesize your state machines to ensure that they are in fact, synthesizable.

    a. There is a tutorial on the class Blackboard page (VHDL Synthesis Tutorial) that takes you through this process for a simple VHDL adder component.

    b. Find this tutorial on the Blackboard page under Course Documents - Supplemental Materials, and go through it on your own.

    c. Use the same procedure on this design and target the Xilinx FPGA's as you did for the tutorial.

    d. Perform both a postsynthesis and timing simulation as you did in the tutorial.

    e. Note that in both the post-synthesis and timing simulations, you will have to replace the behavioral models of the state machines in the testbench with the gate-level netlists output by the tools and in the timing simulation, you will have two SDF files to back-annotate to the respective state machines.

    f. In the timing simulation, the initial clock periods of 110 ns and 190 ns should be very conservative. Perform a series of simulations (use TYP timing) where you decrease the period 5 ns at a time for each clock until the timing simulation no longer functions correctly.

    g. Note the minimum clock period possible for this Xilinx implementation in your writeup.

7. For this assignment, you must turn in a cover sheet with your name, date, and a brief paragraph describing the design of your state machines and the results of your synthesis and timing simulations.

    a. *Include a diagram of the state transition graph of each state machine similar to the ones shown in class.*

    b. You also need to hand in a complete listing of all of your VHDL code for your design and testbench.

    c. Your submission must also include a simulation output waveform printout showing the complete simulation with all data transmissions and a zoom-in of one data transmission as shown above.

    d. Include the simulation results of the fastest working timing simulation and the first timing simulation where the simulation did not function correctly.

    e. Finally, you must include the portion of the Precision transcript window that shows the gatecount of the synthesized result for each state machine.