CS223 Laboratory Assignment 5

Designing a Master/Slave Mode Controller

Section 1:	09.12.2019	Monday	08:40-12:25
Section 2:	10.12.2019	Tuesday	08:40-12:25
Section 3:	09.12.2019	Monday	13:40-17:25
Section 4:	10.12.2019	Tuesday	13:40-17:25
Section 5:	05.12.2019	Thursday	08:40-12:25
Section 6:	06.12.2019	Friday	08:40-12:25

Location: EA Z04 (in the EA building, straight ahead past the elevators)

Groups: Each student will do the lab individually. Group size = 1

Preliminary Report (40 points)

In today's lab you are required to design and implement a control unit for the management of two processes running in redundant mode. These advance designs and SystemVerilog models should be prepared in advance, and assembled neatly into a Preliminary Report with a printed cover page and printed pages for the schematics and SystemVerilog codes. Each page should have a proper heading. The contents of the report should be as follows:

a) A cover page which includes the following: course name and code number, the number of the lab, your name and student ID, date, number of your trainer pack (remember lab policies. You must always use same pack number).

b) Design of two process modules has following steps:

Master Process:

1. Start a timer with 20 millisecond (msec) (T_{20msec}) resolution. (Implement this timer by a module)
2. Implement a counter module, as an instance of this module initiate an up counter (C_{master}), this counter will use the overflow flag of T_{20msec} timer as clock input. You are required to use T_{20msec} timer in order to have a 1000 msec time interval by C_{master}.
3. Implement a master process module which will check “Master Mode” user switch (Figure 1), if Master mode is active according to Table 1 than the process will increment C_{master} counter, and also the process will drive the “Master LED” with a period of 2 sec, %50 duty cycle.
4. A register with load module will be implemented and an instance of this module will be declared with the name Reg_{master}. Reg_{master} register content will be incremented by 1 on each overflow of C_{master}. The updated Reg_{master} register value will be shown on the 7-segment display if system is working in master mode.
5. Master/slave mode selection will be done according to master and slave user switch positions (Table 1). If both switches are disabled (M/S: 0 0) than “Enable C_{master}” and “Enable C_{slave}” user switches will be checked. If both counters are enabled with these user switches than master process will have priority and system will work in master mode, if both counters are disabled than system will stop (Mode: None) and both “LED Master” and “LED Slave”

will be **ON**. "Enable C_{master}" and "Enable C_{slave}" user switches (Figure 1) will be used in order to enable or disable the corresponding counter operation.

<table>
<tr><th>Master (M)</th><th>Slave (S)</th><th>Executing Process</th><th>LED Master</th><th>LED Slave</th></tr>
<tr><td>0</td><td>0</td><td colspan="3">Check "Enable C_{master}" and "Enable C_{slave}" user switches
<table>
<tr><th>Enable C_{master}</th><th>Enable C_{slave}</th><th>Mode</th></tr>
<tr><td>0</td><td>0</td><td>None</td></tr>
<tr><td>0</td><td>1</td><td>Slave Mode*</td></tr>
<tr><td>1</td><td>0</td><td>Master Mode*</td></tr>
<tr><td>1</td><td>1</td><td>Master Mode*</td></tr>
</table></td></tr>
<tr><td>0</td><td>1</td><td>Slave Mode*</td><td>OFF</td><td>Blink 500 msec ON – 500 msec OFF</td></tr>
<tr><td>1</td><td>0</td><td>Master Mode*</td><td>Blink 1000 msec ON – 1000 msec OFF</td><td>OFF</td></tr>
<tr><td>1</td><td>1</td><td>Master Mode*</td><td>Blink 1000 msec ON – 1000 msec OFF</td><td>OFF</td></tr>
</table>

Table 1: Master-Slave Mode Selection

Slave Process:

6. Initiate an up counter (C_{slave}) this counter will use the overflow flag of T_{20msec} timer as clock input. You are required to use T_{20msec} timer in order to have a 500 msec time interval by C_{slave}.
7. Implement a slave process module which will check both "Master Mode" and "Slave Mode" user switches (Figure 1) if "Slave Mode" is selected according to the conditions given in Table 1, than the process will increment C_{slave} counter as described above, and also the process will drive the "Slave LED" with a period of 1 sec, %50 duty cycle.
8. A register with load module will be implemented and an instance of this module will be declared with the name Reg_{slave}. Reg_{slave} register content will be incremented by 1 on each overflow of C_{slave} counter. The updated Reg_{slave} register value will be shown on the 7-segment display if system is working in slave mode.

*On each mode change from slave to master or master to slave the register corresponding to the active state (Reg_{master} or Reg_{slave}) will be cleared to 0.

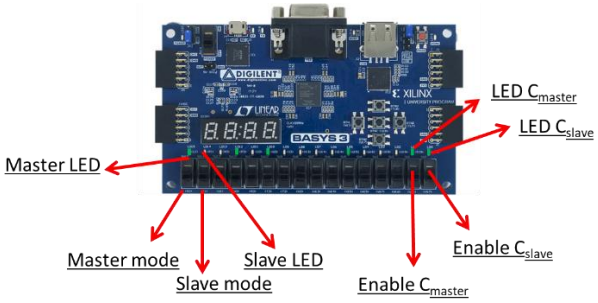

Figure 1: Master-Slave Mode Selection

The whole system with controller and datapath is given in Figure 2.

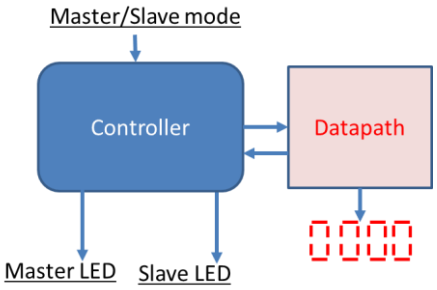

Figure 2: Master/Slave Module System Level

Main data path components of the design is given to you in Figure 3. You need to design the control unit by yourself with a FSM. Draw the controller's state machine in your report.

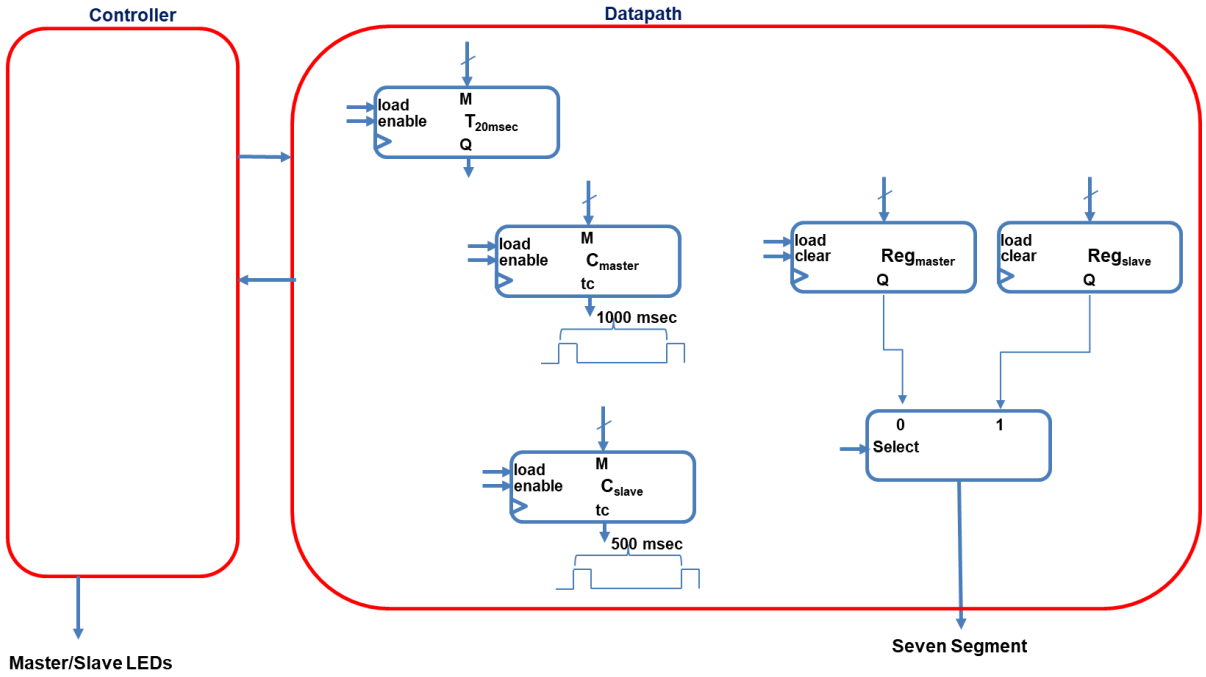

Figure 3: Main Datapath Components

c) Write the SystemVerilog code for the full design of master_slave Module. Prepare a testbench to your circuit.

The Preliminary Report will be turned in at the start of lab. You may need a copy of your designs and SystemVerilog programs with you in the lab to refer to or possibly correct and change it.

Implementation on FPGA (60 points)

a) Create a Vivado project and enter the SystemVerilog module into Vivado you prepared in preliminary work.

b) Simulate the master_slave module for all user switch combinations using the testbench code you prepared in preliminary. Test your circuit, if correct show it to your TA. Your simulation should clearly show the intermediate steps, so your TA can confirm them.
c) Implement your code on FPGA (you do not need Beti board in this lab). The master or slave register content should be connected to two digits of 7-segment module as a 4-digit hexadecimal number (use SystemVerilog code for 7-segment module from lab4). Assign a pushbutton (not switch) on BASYS3 to system reset.
Note: pushbuttons on BASYS3 give you a clean pulse while you keep them pressed without any vibration. Then, you do not need to implement any additional module like debouncer for them.

d) When you are ready, show your implementation on the BASYS3 board to TA. After resetting, he/she will give arbitrary inputs by using user switches then see the result on 7-segment display.

Submit your code for MOSS similarity testing

Finally, when you are done and before leaving the lab, you need to upload the file *StudentID_SVerilog.txt* created in the Implementation with FPGA part. Be sure that the file contains exactly and only the codes which are specifically detailed above. If you have multiple files, just copy and paste them in order, one after another inside text file. Check the specifications! Even if you didn't finish, or didn't get the SystemVerilog part working, you must submit your code to the Unilica Assignment for similarity checking. Your codes will be compared against all the other codes in all sections of the class, by the MOSS program, to determine how similar it is (as an indication of plagiarism). So be sure that the code you submit is code that you actually wrote yourself! All students must upload their code to the 'Unilica>Assignment' specific for your section. Check submission time and don't miss it before leaving the lab. After taking a backup of your work, don't forget to delete it from computer. Because students of other sections will work with your system too.

Clean Up

1) Clean up your lab station, and return all the parts, wires, the Beti trainer board, etc. Leave your lab workstation for others the way you would like to find it.

2) CONGRATULATIONS! You are finished with this Lab and are one step closer to becoming a computer engineer.

NOTES
--Advance work on this lab, and all labs, is strongly suggested.
--Be sure to read and follow the Policies for CS223 labs, posted in Unilica.

LAB POLICIES
1. There are three computers in each row in the lab. Don't use middle computers, unless you are allowed by lab coordinator.

2. You borrow a lab-board containing the development board, connectors, etc. in the beginning. The lab coordinator takes your signature. When you are done, return it to his/her, otherwise you will be responsible and lose points.
3. Each lab-board has a number. You must always use the same board throughout the semester.
4. You must be in the lab, working on the lab, from the time lab starts until you finish and leave. (bathroom and snack breaks are the exception to this rule). Absence from the lab, at any time, is counted as absence from the whole lab that day.
5. No cell phone usage during lab. Tell friends not to call during the lab hours--you are busy learning how digital circuits work!
6. Internet usage is permitted only to lab-related technical sites. No Facebook, Twitter, email, news, video games, etc.--you are busy learning how digital circuits work!
7. If you come to lab later than 20 minutes, you will lose that session completely.
8. When you are done, DO NOT return IC parts into the IC boxes where you've taken them first. Just put them inside your Lab-board box. Lab coordinator will check and return them later.