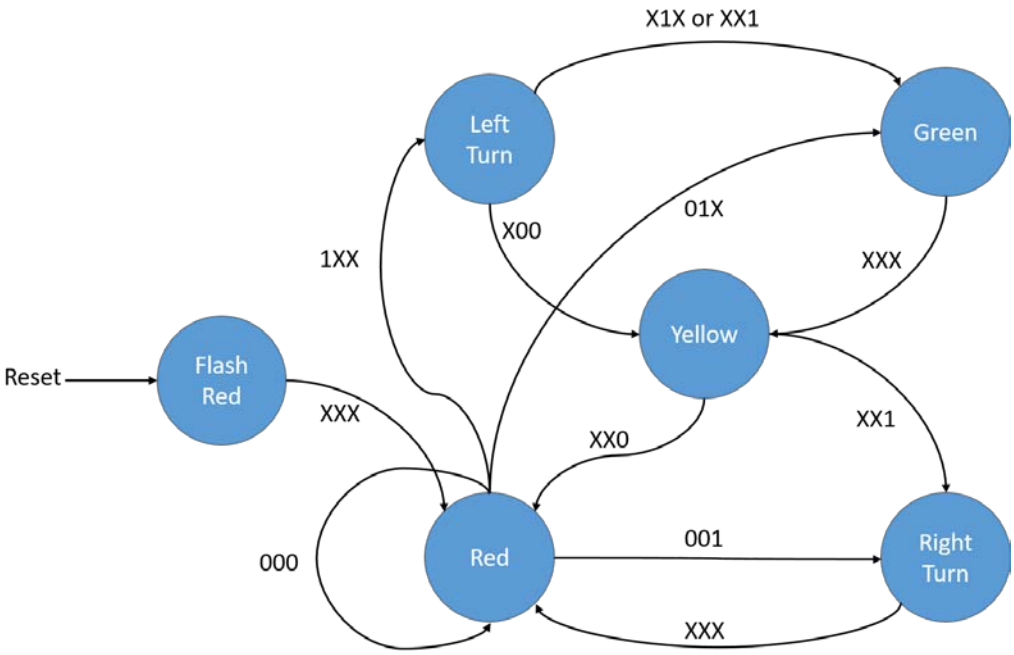
ECE3270 Digital System Design
Lab 3: State Machines

Lab Overview: The purpose of this lab is to implement a state machine that keeps track of the current traffic signal status. The state machine will then need to have the libraries signals added to work with OpenCL. The requirements of this lab consist of completing the VHDL and OpenCL design, submitting all modified source files to the assign server, and completing the findings report using the LaTeX template previously provided. You should include discussion of the type of state machine generated and include a copy of the state machine diagram generated by Quartus.

Part I

You are to design a Moore state machine that operates a traffic signal. The machine requires 6 states (flash_red, red, green, yellow, left_green, right_red) to keep track of the states for a traffic signal. The state machine will accept 3 bits as input that represent 3 lanes (left, middle, right). The state transitions are shown in the below image.



The output of this machine will be the same as the decoder from the previous assignment. Refer to the Lab 2 documentation for these outputs. All states map to their corresponding outputs, but left turn needs to map to Left & Red. If you choose, you may use the Lab 2 file to decode the states (custom enumeration of states would be required). This solution would be more elegant and require less code than manually typing all outputs. An X is a don't care bit, in case there is any confusion.

Clearly enumerate your states. Map the bitwise output of your states to the LEDs on the DE1-SoC board and map the input to three of the switches. Use a key to simulate a clock to signify when your state machine should update. Use another key as a low-active, asynchronous reset that will set the signal to flash_red. **DO NOT** set LEDR or any of the SWx pins as external signals. Give your inputs and outputs meaningful names for this component and map these pins to the appropriate I/O on the DE1-SoC board. This will help when completing Part II.

Part II

To work with OpenCL, you will need to make a minor modification to your traffic component. State Machines require the use of the *ivalid* pin to make sure your state machine will not update too often in the OpenCL environment. When checking if your machine should update, check that *ivalid* is high. This will tell the machine the input data is valid and will allow the machine to go to the next state. Note that *ivalid* should not be on the sensitivity list! Feel free to add this into Part I so you only have one VHDL description of your traffic file.

You will need to download the Traffic.tgz file and extract it on one of the ULLAB machines. You will need to edit the appropriate files in the device folder to create your component and map your signals to the appropriate library signals. Your input from the OpenCL host will come in on *datain(2 downto 0)* and you will write your output to *dataout*. You will map *clock*, *ivalid*, and *resetn* to the respective pins on your state machine component. If you chose to use your decoder for your machine, you will need to edit the *traffic_rtl.xml* file and add the name of that file in the file list (just use the other files included as examples for syntax).

Compile the host and device binaries (the device will take significantly longer to compile than Lab 2) and copy them to your SD card. When executing the program, the default execution will require you enter 3 bits at a time (010, 101, 111, etc.) to determine your state transitions. Follow the instructions on the screen to make sure it all works! If you instead run "./traffic –t", the program will run a default set of state transitions. You will find the expected output of this in the README.txt file.

Submissions

You will need to upload all edited files to the assign server as Assignment 3 (include the decoder file if you choose to use this). You will also need to upload your report to Turnitin on the Canvas assignment. The due date is October 20, 2017.