Prelab 6

Part I

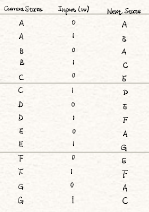
2. Answer the following questions in your prelab: given the starter code, is the `resetn` signal is an synchronous or asynchronous reset? Is it active high, or active low? Given this, what do you have to do in simulation to reset the FSM to the starting state? **(PRELAB)**

1) It's a synchronous reset

2) It's active low

3) Make reset_n (SW[0]) = 0, then creat a posedge of clock (KEY[0])

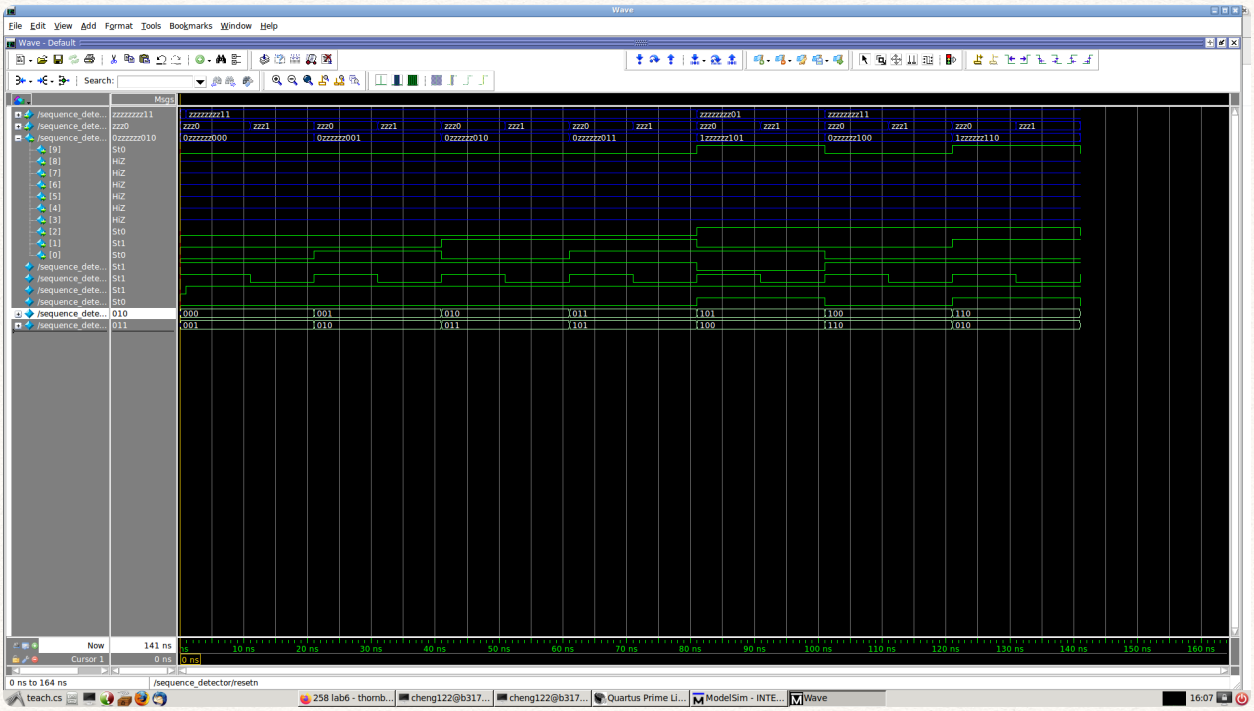
3. Complete the state table showing how the present state and input value determine the next state and the output value. Fill in all the missing parts of the template code to implement the FSM based on the state table you derived. Include completed code in your prelab. **(PRELAB)**



see sequence_detector.v

for completed Verilog code.

4. Simulate your circuit using ModelSim for a variety of input settings, ensuring the output waveforms are correct. Include a few screenshots that shows the simulation output. **(PRELAB)**



Part II

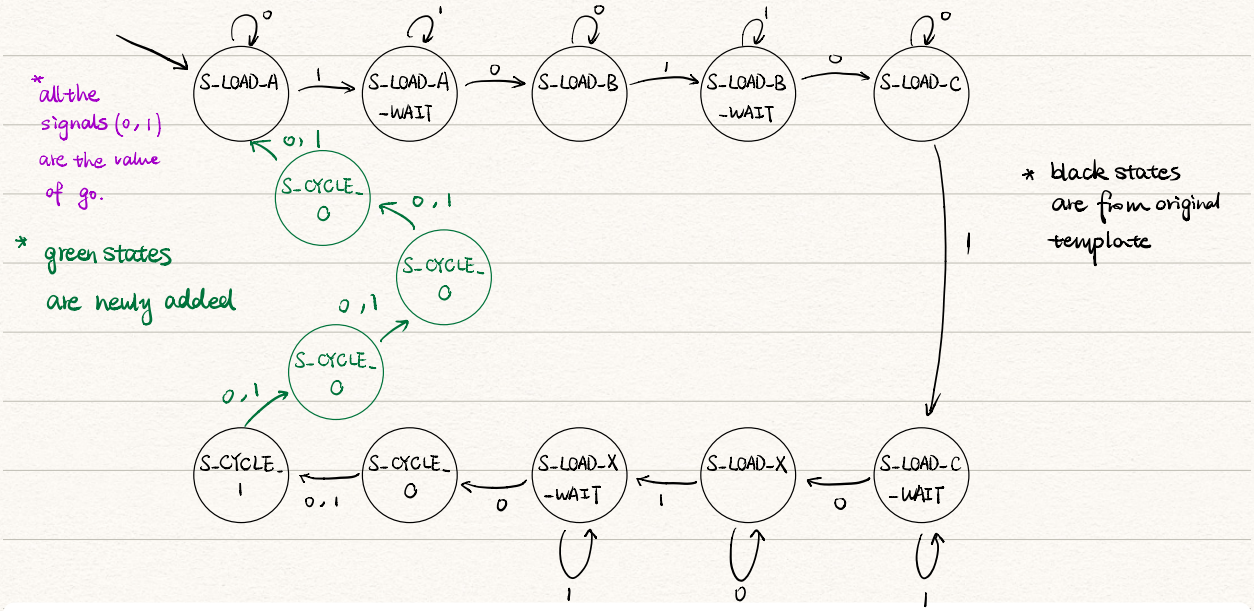
1. Examine the provided starter Verilog code for this part (`poly_function.v`, which is available on Quercus, as well as in the appendices of this handout). This is a major step in this part of the lab. You will not need to write much Verilog yourself, but you will need to fully understand the circuitry described by the provided Verilog to be able to make your modifications. **(PRELAB)**

I've read and understand the circuitry.

2. Determine a sequence of steps similar to the datapath example shown in lecture that controls your datapath to perform the required computation. You should draw a table that shows the state of the Registers and control signals for each cycle of your computation. Include this table in your prelab. **(PRELAB)**

Cycle	Computation	alu_op	ld_a	ld_b	ld_r	ld_alu_out	alu_select_a	alu_select_b	A	B	R
0	Bx	1	0	1	0	1	01	11	B	x	/
1	$Bx+A$	0	0	1	0	1	01	00	Bx	A	/
2	Cx	1	1	0	0	1	10	11	C	x	/
3	Cx^2	1	1	0	0	1	00	11	Cx	x	/
4	Cx^2+Bx+A	0	0	0	1	0	01	00	Cx^2	Bx+A	Cx^2+Bx+A

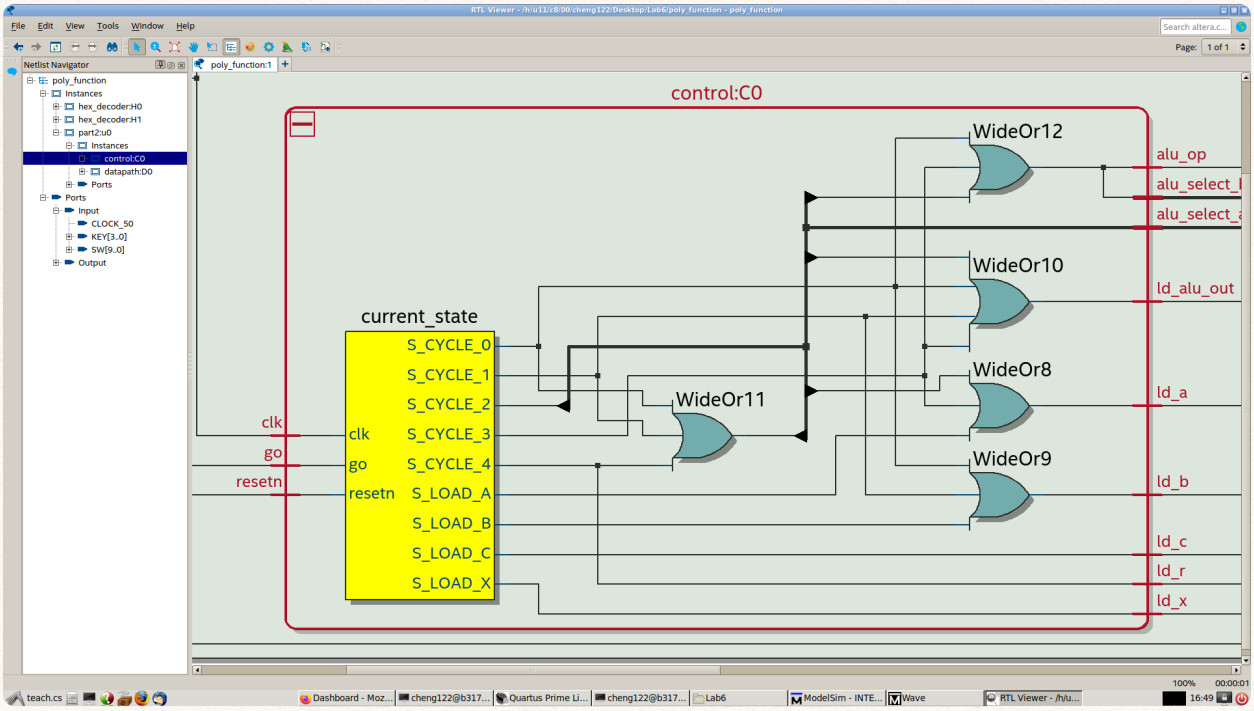
3. Draw a state diagram for your controller starting with the register load states provided in the example FSM. Include the state diagram in your prelab. **(PRELAB)**

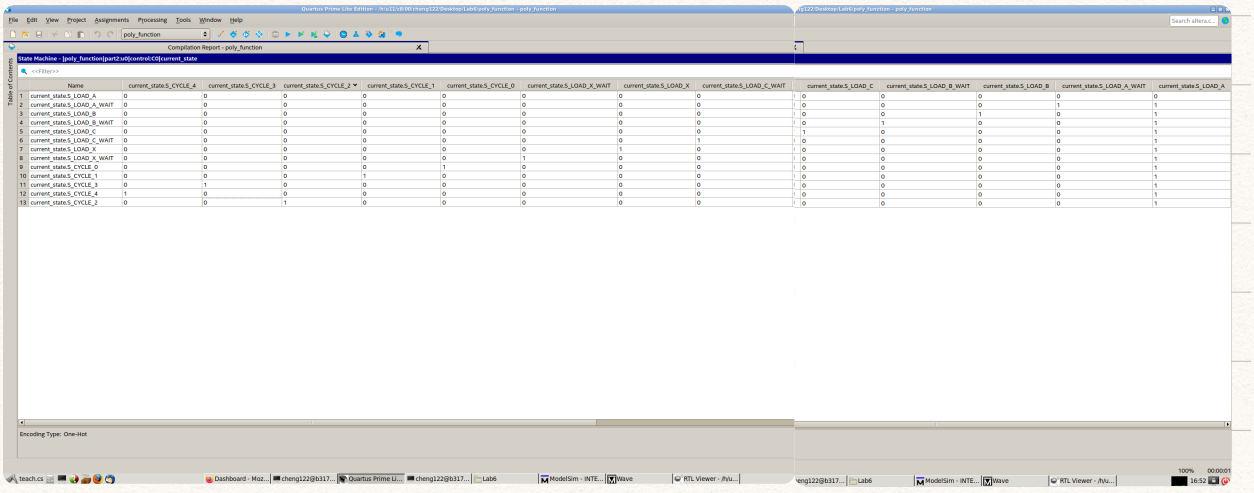


4. Modify the provided FSM code to implement your controller and synthesize it. You should only modify the control module. Include your modified code in the prelab. **(PRELAB)**

See poly_function.v

5. To examine the circuit produced by Quartus Prime open the RTL Viewer tool (Tools > Netlist Viewers > RTL Viewer). Find (on the left panel) and double-click on the box shown in the circuit that represents the finite state machine, and determine whether the state diagram that it shows properly corresponds to the one you have drawn. To see the state codes used for your FSM, open the Compilation Report, select the Analysis and Synthesis section of the report, and click on State Machines. Include a screenshot of the generated FSM in your prelab. **(PRELAB)**





6. Simulate your circuit with ModelSim for a variety of input settings, ensuring the output waveforms are correct. It is recommended that you start by simulating the datapath and controller modules separately. Only when you are satisfied that they are working individually should you combine them into the full design. Why is this approach better? (Hint: Consider the case when your design has 20 different modules.) Include few screenshots of simulation output in your prelab. **(PRELAB)**