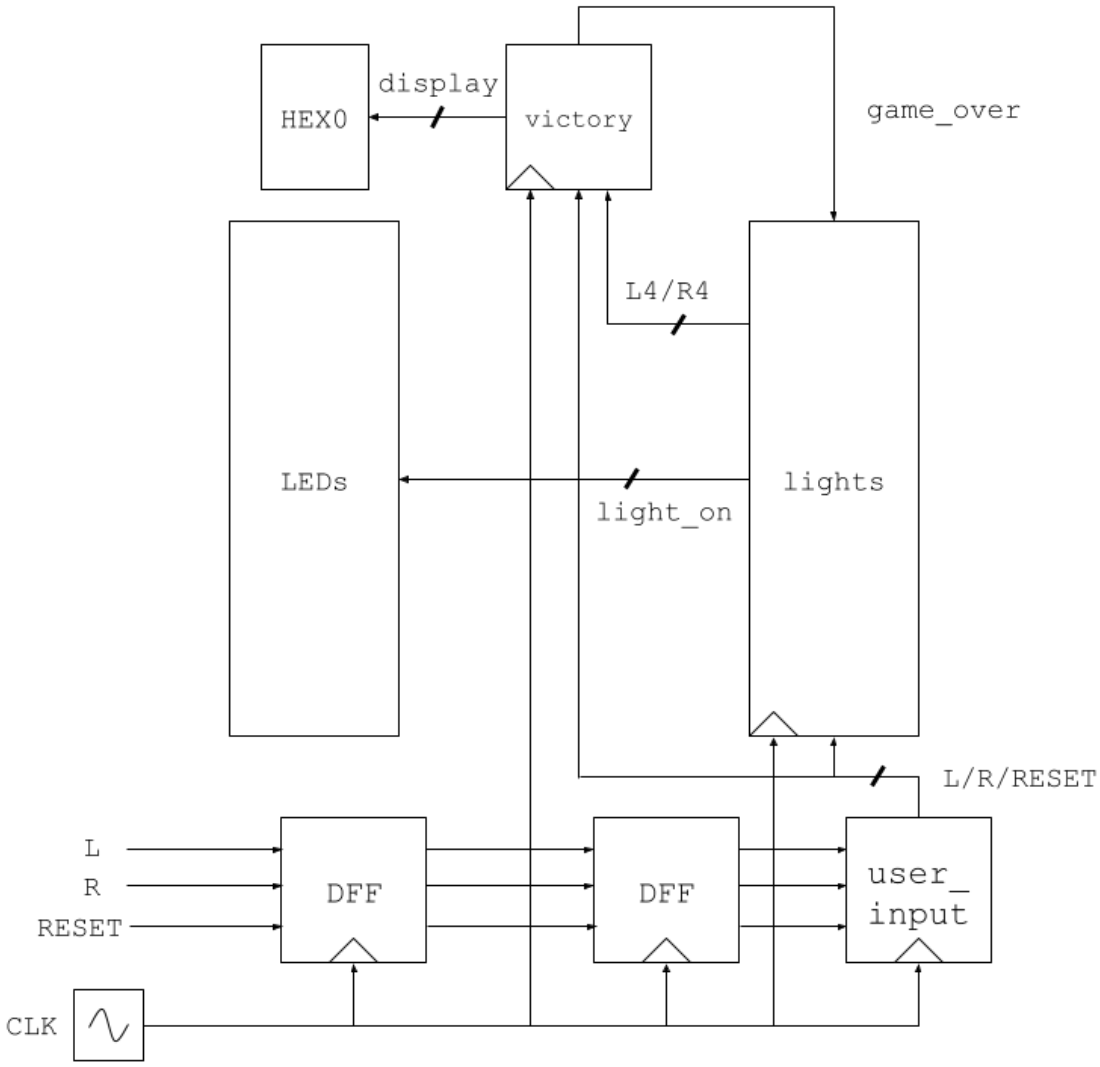
Lab 6

Communicating Sequential Logic

Connor Aksama – 1778028

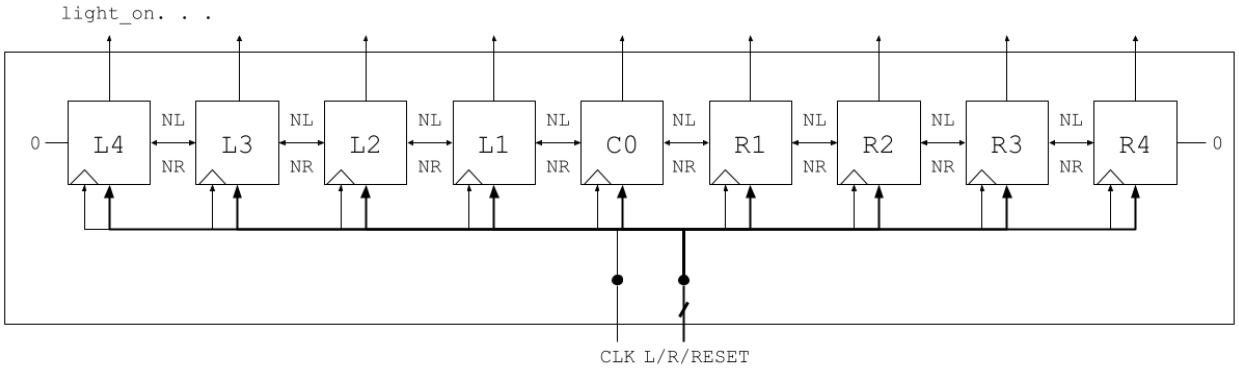
The top-level block diagram, showing the major modules and how they are interconnected.

Tug-of-War Top Level Block Diagram



The top-level block diagram of the major modules in this lab. The `lights` block abstracts 9 individual light-controlling modules (See below). `DFF` blocks are used to stabilize direct user input, the `user_input` block detects the moment an input toggles high, and the `victory` block controls the HEX0 display and feeds back a `game_over` state to each light module.

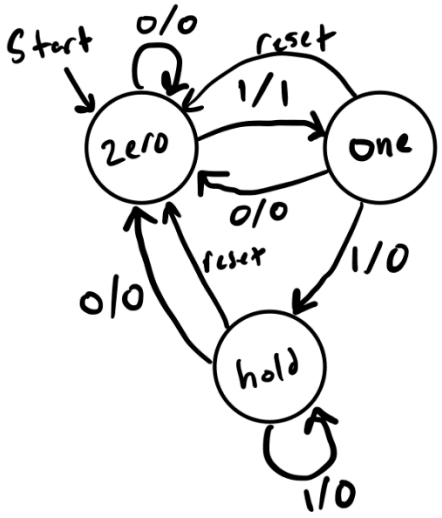
`lights` Block Top Level Block Diagram



The block diagram for the individual light-controlling modules. The output signal from an individual block is wired into the block to its left (as an input signal, `NR`), and to the block to its right (as an input signal, `NL`).

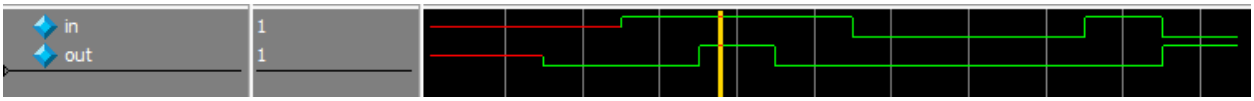
For each of the major modules, include a state diagram (if applicable) and screenshot of the ModelSim simulation. Also include a ModelSim simulation for the top-level module).

User Input State Diagram



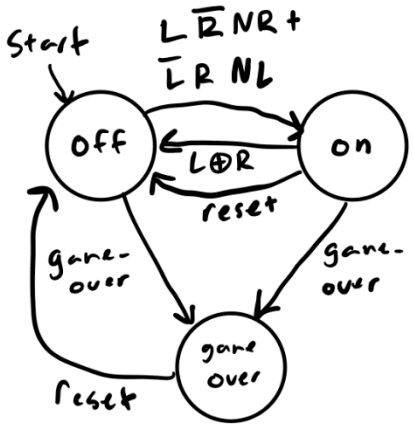
The state diagram for the `user_input` module. This state machine will output 1 for the first 1 in a consecutive run of 1s, and 0 for every other input.

User Input Simulation



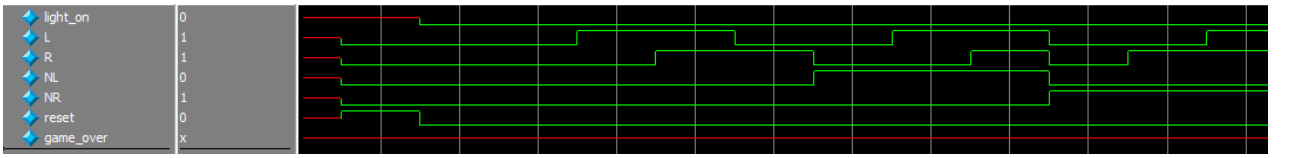
This screenshot shows output for all of the possible state transitions for the user_input state machine. Notice that the output toggles high for only one clock cycle after the input toggles high.

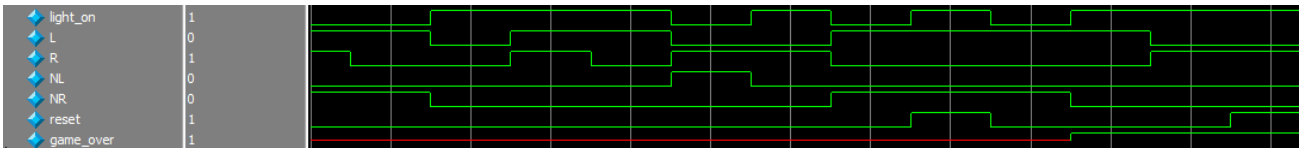
Normal Light State Diagram



The state diagram for `L4-L1` and `R4-R1` modules. This state machine takes 6 inputs: `L, R, NL, NR, game_over,` and `reset.` If the expression associated with a transition is true, then the state machine follows that transition to the next state. All other transitions not pictured are self-transitions.

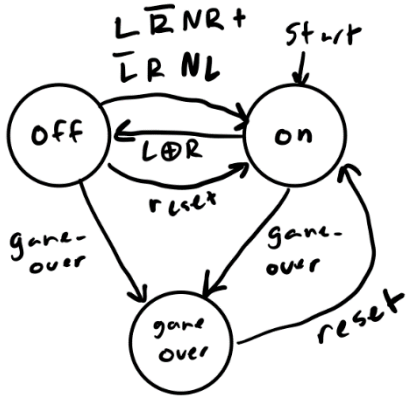
Normal Light Simulation





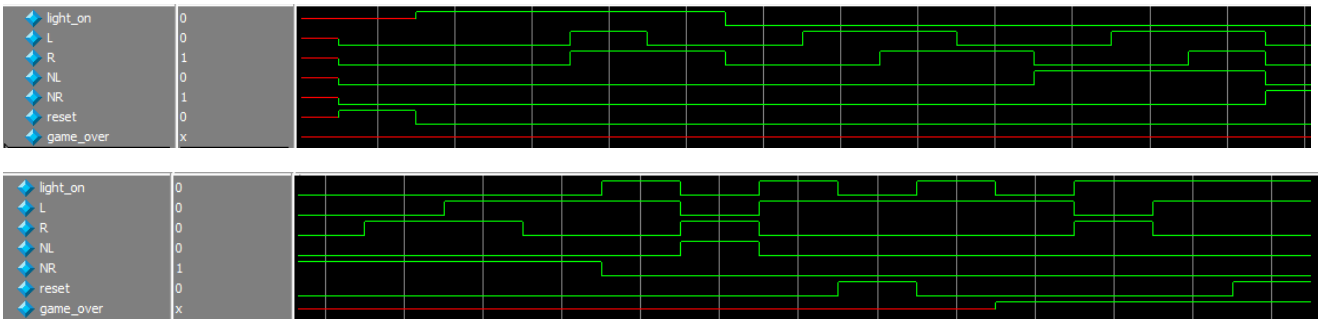
The simulation for the `normal_light` module. This simulation runs through all of the possible state transitions and shows the resulting `light_on` output. Notice that the output starts off, and is toggled on when an appropriate input is received and stays on unless one of the appropriate inputs is received.

Center Light State Diagram



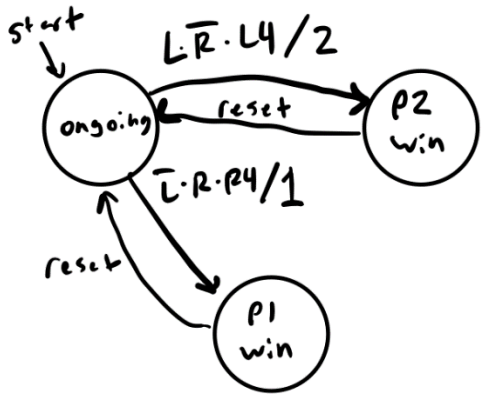
The state diagram for the `C0` module. This state machine takes 6 inputs: `L`, `R`, `NL`, `NR`, `game_over`, and `reset`. If the expression associated with a transition is true, then the state machine follows that transition to the next state. All other transitions not pictured are self-transitions.

Center Light Simulation



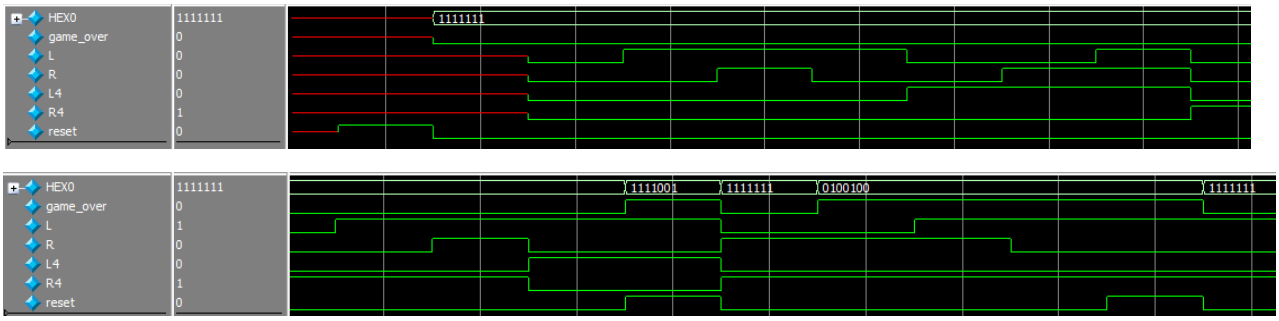
The simulation for the `center_light` module. This simulation runs through all of the possible state transitions and shows the resulting `light_on` output. Notice that the output starts on, and is toggled off when an appropriate input is received and stays off unless one of the appropriate inputs is received.

Victory State Diagram



The state diagram for the `victory` module. This state machine takes 5 inputs: `L, R, NL, NR,` and `reset.` When the state machine is in one of the “win” states, it outputs the corresponding number (1/2) to the HEX0 display and a true `game_over` signal. All other transitions not pictured are self-transitions.

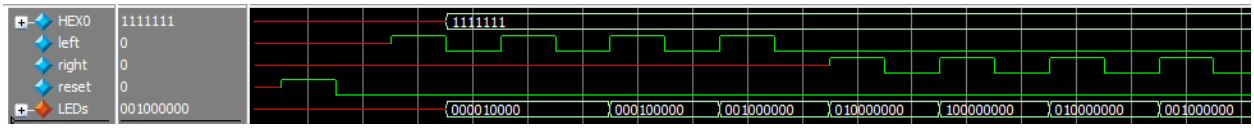
Victory Simulation



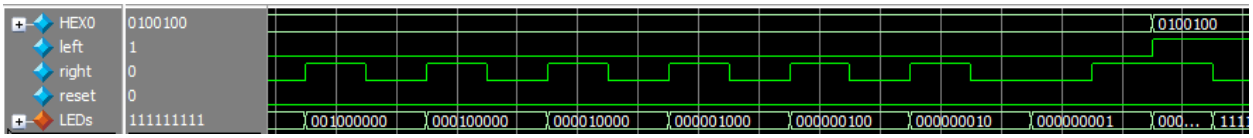
The simulation for the victory module. Notice that the HEX0 display shows the correct pattern and the `game_over` signal toggles on when transitioning into a winning state. Any other input while the module is in a winning state does not change the output. Only the reset signal puts the module back in its original state.

Tug-of-War Simulation

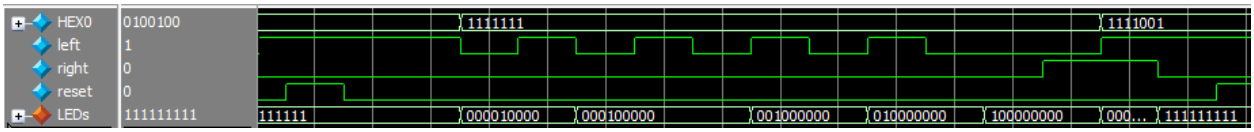
1:



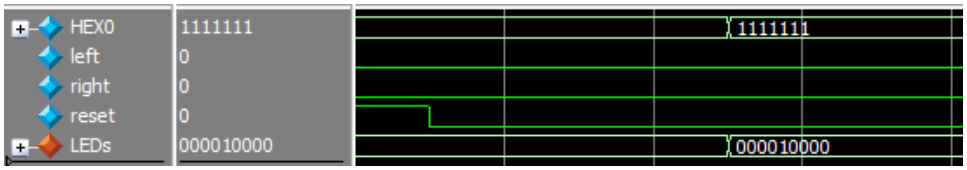
2:



3:



4:



The simulation for the top-level `tug_of_war` module. This is simulating moving the playfield to the left end (1), moving the playfield to the right end (1-2), a player 1 victory (2-3), and a player 2 victory (3-4). Notice that whenever the circuit enters a winning state, the only input that will change the output is a reset signal (which reverts the circuit to its original state).

A screenshot of the "Resource Utilization by Entity" page, showing your design's computed size.

Analysis & Synthesis Resource Utilization by Entity

<<Filter>>

	Compilation Hierarchy Node	Combinational ALUTs	Dedicated Logic Registers	Block Memory Bits	DSP Blocks	Pins	Virtual Pins	Full Hierarchy Name	Entity Name	Library Name
1	|tug_of_war	18 (0)	21 (0)	0	0	32	0	|tug_of_war	tug_of_war	work
1	|center_light:c0|	2 (2)	1 (1)	0	0	0	0	|tug_of_war...er_light:c0	center_light	work
2	|dff_3:d0|	0 (0)	3 (3)	0	0	0	0	|tug_of_war|dff_3:d0	dff_3	work
3	|dff_3:d1|	0 (0)	3 (3)	0	0	0	0	|tug_of_war|dff_3:d1	dff_3	work
4	|normal_light:l1|	1 (1)	1 (1)	0	0	0	0	|tug_of_war...al_light:l1	normal_light	work
5	|normal_light:l2|	1 (1)	1 (1)	0	0	0	0	|tug_of_war...al_light:l2	normal_light	work
6	|normal_light:l3|	1 (1)	1 (1)	0	0	0	0	|tug_of_war...al_light:l3	normal_light	work
7	|normal_light:l4|	1 (1)	1 (1)	0	0	0	0	|tug_of_war...al_light:l4	normal_light	work
8	|normal_light:r1|	1 (1)	1 (1)	0	0	0	0	|tug_of_war...al_light:r1	normal_light	work
9	|normal_light:r2|	1 (1)	1 (1)	0	0	0	0	|tug_of_war...al_light:r2	normal_light	work
10	|normal_light:r3|	1 (1)	1 (1)	0	0	0	0	|tug_of_war...al_light:r3	normal_light	work
11	|normal_light:r4|	1 (1)	1 (1)	0	0	0	0	|tug_of_war...al_light:r4	normal_light	work
12	|user_input:left_press|	2 (2)	2 (2)	0	0	0	0	|tug_of_war...:left_press	user_input	work
13	|user_input:right_press|	2 (2)	2 (2)	0	0	0	0	|tug_of_wa...ight_press	user_input	work
14	|victory:v0|	4 (4)	2 (2)	0	0	0	0	|tug_of_war|victory:v0	victory	work

Resource utilization for the `tug_of_war` top-level module.

Time Estimation

This lab took approximately 7 hours, in total, to complete.