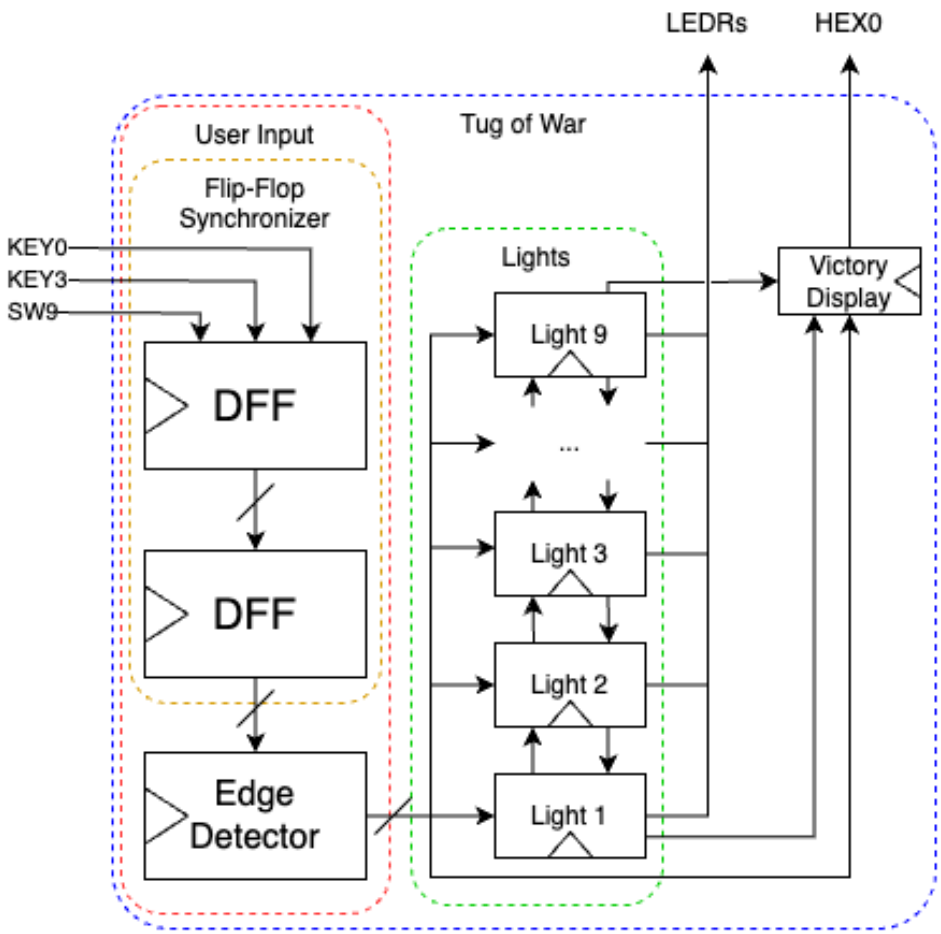
CSE 369 Lab 6

Communicating Sequential Logic

Isaac Wu
2360957
November 13, 2024

1. Top-Level Block Diagram

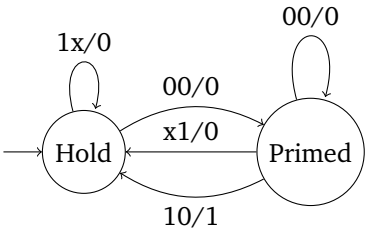


This is my Top-Level Block Diagram for the Tug-of-War module. The Tug-of-War module takes in 3 inputs, KEY0 (for Player 1), KEY3 (for Player 2), and SW9 as the Reset input. These 3 inputs go through two D flip-flops to synchronize the inputs for metastability. I then pass in the user inputs into an Edge Detector to reduce each button press to one cycle in length since our clock is very fast. Then, I pass the configured user inputs into each of the 9 light modules. These light modules have their own state and will determine whether or not their linked LED will be lit, which is output to the LEDRs outside of the ToW module. I also pass the user inputs into a Victory Display module, which controls the HEX0 display depending on if there has been a winner and if so, who won.

2. Edge Detector

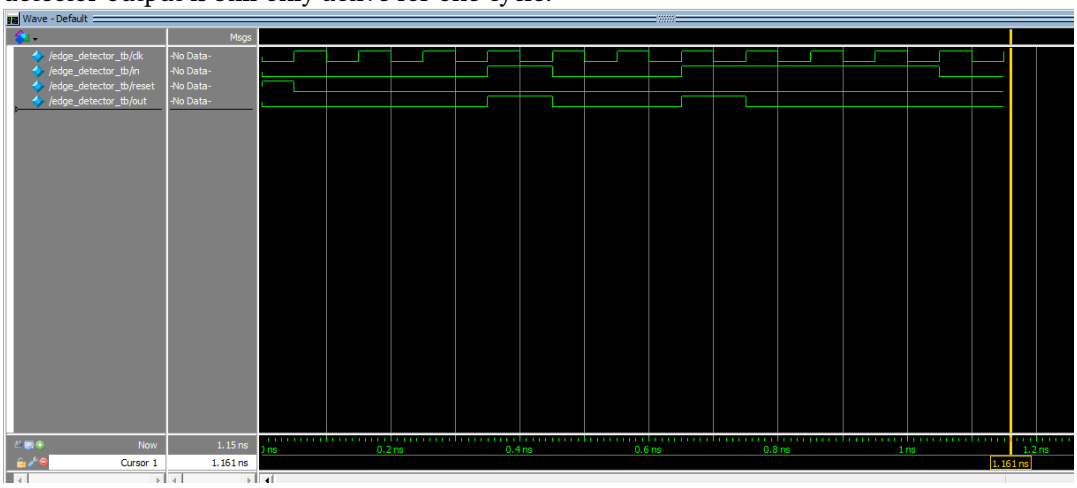
2.1. Finite State Machine

Input: {User Input, Reset}
Output: {Edge output}



2.2. ModelSim Simulation

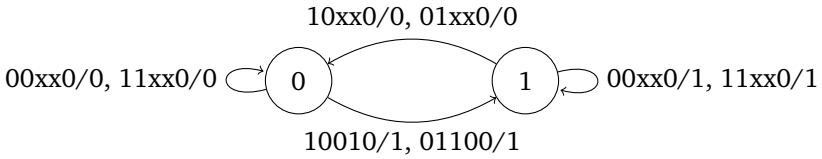
These are the resulting waves of a simple test bench created to verify the functionality of the Edge Detector module. As seen below, when the input is active for one clock cycle, the output is also active for only one clock cycle, as expected. When the input is held for longer than one clock cycle, the edge detector output is still only active for one cycle.



3. Lights

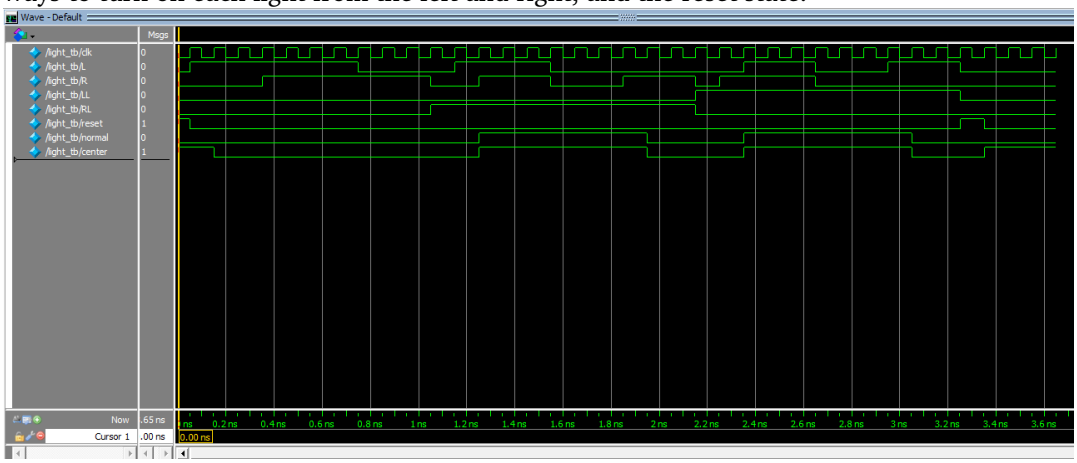
3.1. Finite State Machine

Input: {Left Input, Right Input, Left Light, Right Light, Reset}
Output: {Light Output}
Initial Status and Reset Position is passed in as a parameter



3.2. ModelSim Simulation

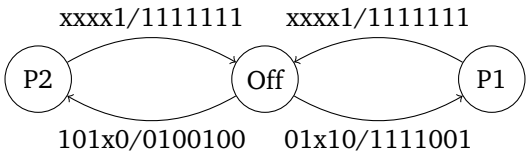
These are the resulting waves for a test bench for the light modules. I defined two light modules, a 'normal' light and a 'center' light. The only difference between these two is the parameter I passed in for its initial state and reset state (0 for normal, 1 for center). I do use the same set of inputs for both lights. There are inputs that are defined as not near the lights–the normal light does not turn on and the center light turns off since the input moves the lit state away from the center. I then test various ways to turn on each light from the left and right, and the reset state.



4. Victory Display

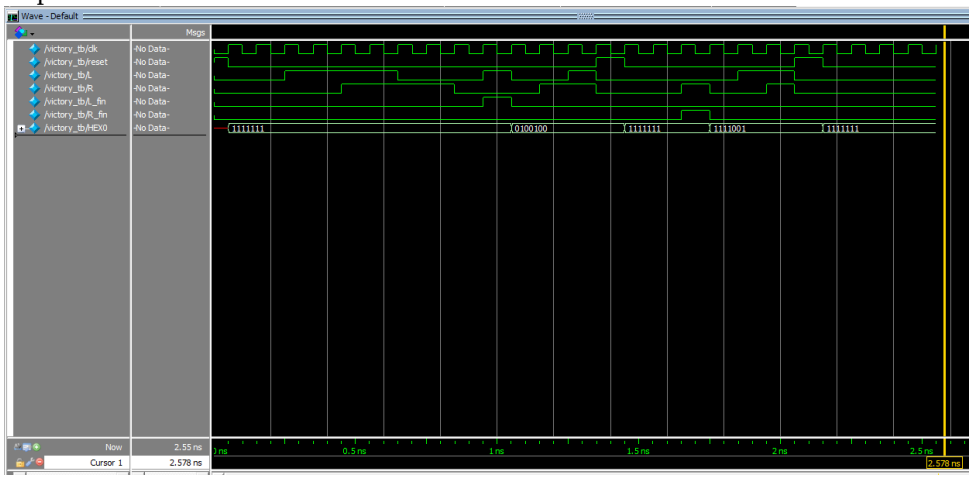
4.1. Finite State Machine

Input: {Left Input, Right Input, Left-most Light, Right-most Light, Reset}
Output: {HEX Display Output}



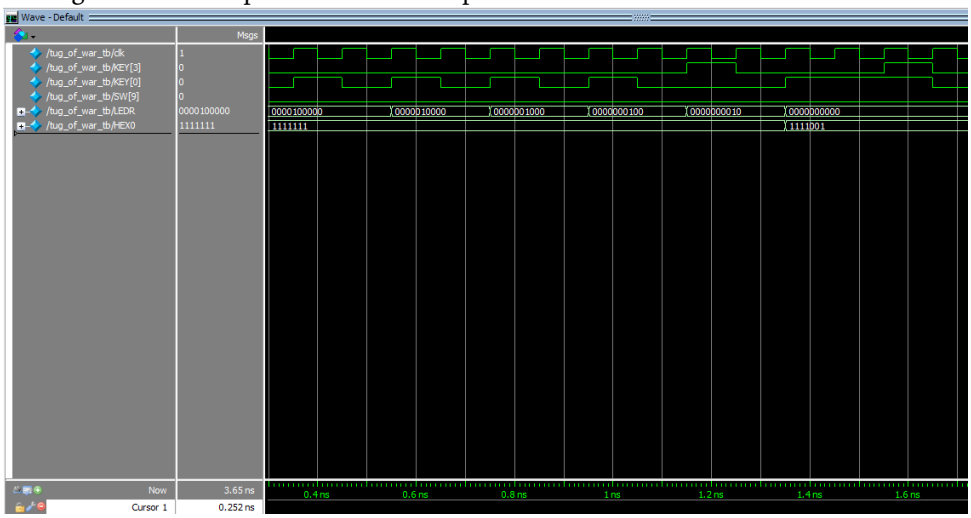
4.2. ModelSim Simulation

This is the wave diagram for a test bench used to test the HEX output, controlled by the Victory module. After a reset input, the HEX output is defined as empty. With inputs not near the ends of the LEDs, the output doesn't change. However, if the module receives a left input, no right input, and the left-most LED is active, then the HEX display will change to show a 2 to indicate Player 2 winning. The same happens for Player 1 on the right side. Note that inputs after a win don't influence the display until a reset input is received.



5. Top-Level ModelSim Simulation

This is what a wave diagram of Player 1 winning looks like. The LED status light starts in the center and moves to the right 2 cycles after each input (due to how our input synchronization works). After the status light moves off of the final LED position, the HEX display will show a 1. Inputs after a win don't change the LED output or the HEX output.



This is a wave diagram of Player 2 winning. The same thing happens, just in reverse. The LED status light starts in the center and moves to the left. After it falls off the left, the HEX display will change to display a 2. Again, any inputs after a victory will not change any displays.

6. Resource Utilization

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	23 (0)	20 (0)	0	0	32	0	|tug_of_war	tug_of_war	work
1	|edge_detector:left|	1 (1)	1 (1)	0	0	0	0	|tug_of_war..tector:left	edge_detector	work
2	|edge_detector:right|	2 (2)	1 (1)	0	0	0	0	|tug_of_war..ector:right	edge_detector	work
3	|light:L1|	3 (3)	1 (1)	0	0	0	0	|tug_of_war|light:L1	light	work
4	|light:L2|	2 (2)	1 (1)	0	0	0	0	|tug_of_war|light:L2	light	work
5	|light:L3|	1 (1)	1 (1)	0	0	0	0	|tug_of_war|light:L3	light	work
6	|light:L4|	1 (1)	1 (1)	0	0	0	0	|tug_of_war|light:L4	light	work
7	|light:L5|	2 (2)	1 (1)	0	0	0	0	|tug_of_war|light:L5	light	work
8	|light:L6|	1 (1)	1 (1)	0	0	0	0	|tug_of_war|light:L6	light	work
9	|light:L7|	1 (1)	1 (1)	0	0	0	0	|tug_of_war|light:L7	light	work
10	|light:L8|	1 (1)	1 (1)	0	0	0	0	|tug_of_war|light:L8	light	work
11	|light:L9|	1 (1)	1 (1)	0	0	0	0	|tug_of_war|light:L9	light	work
12	|sync:sync|	2 (0)	6 (0)	0	0	0	0	|tug_of_war|sync:sync	sync	work
1	|d_ff:d0|	2 (2)	3 (3)	0	0	0	0	|tug_of_wa...nc|d_ff:d0	d_ff	work
2	|d_ff:d1|	0 (0)	3 (3)	0	0	0	0	|tug_of_wa...nc|d_ff:d1	d_ff	work
13	|victory:vic|	5 (5)	3 (3)	0	0	0	0	|tug_of_war|victory:vic	victory	work

7. Misc.

How many hours (estimated) it took to complete this lab in total, including reading, planning, designing, coding, debugging, and testing.

It took around 6 hours to complete this lab.