Eugene Ngo

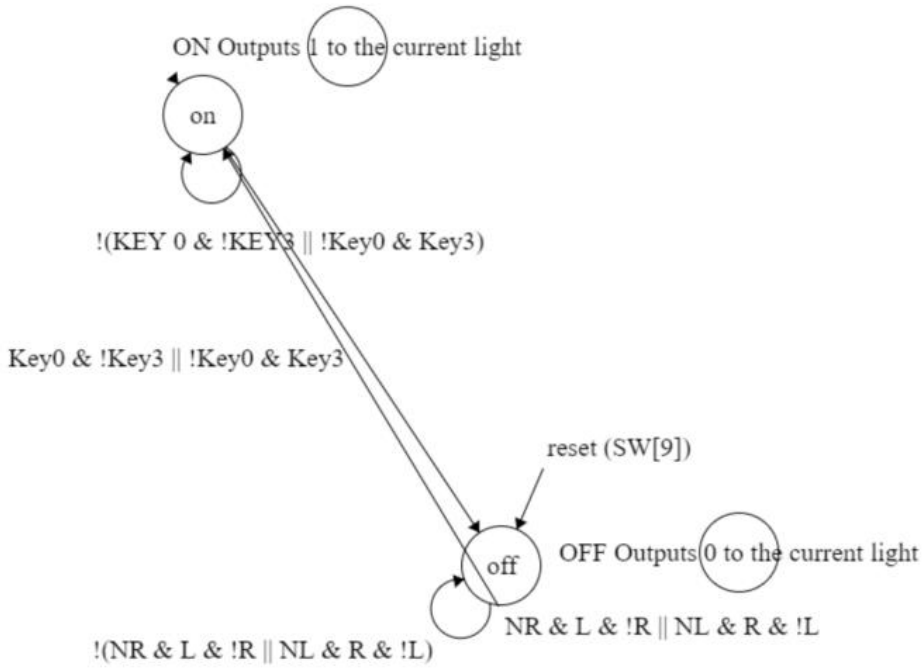
11/18/2022

EE 271

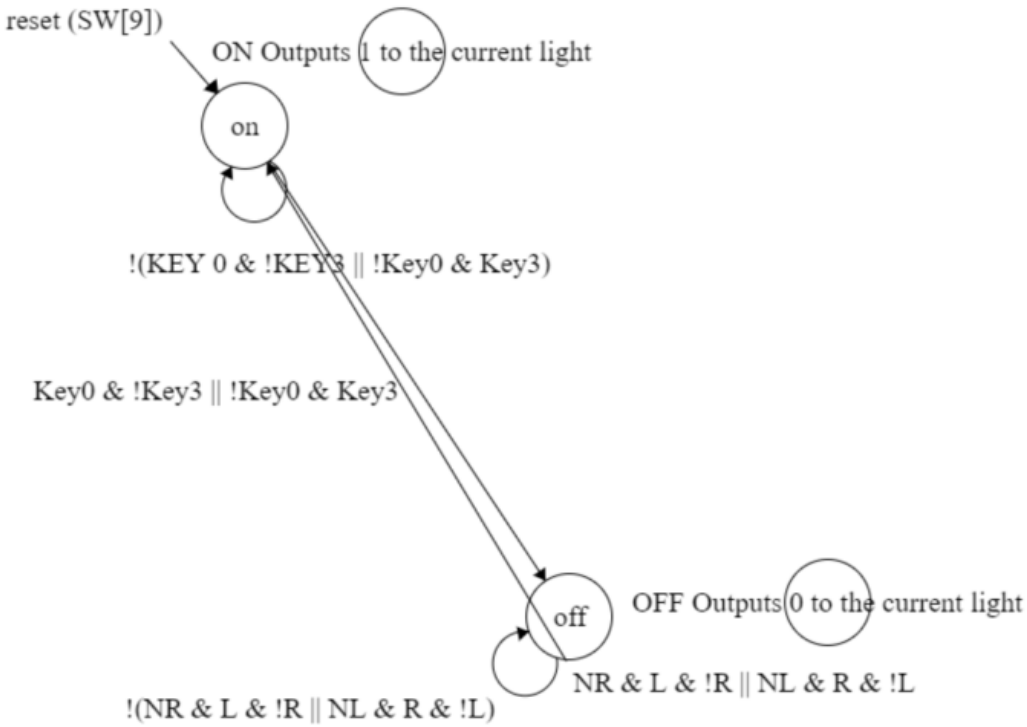
Lab 4 Report

Procedure

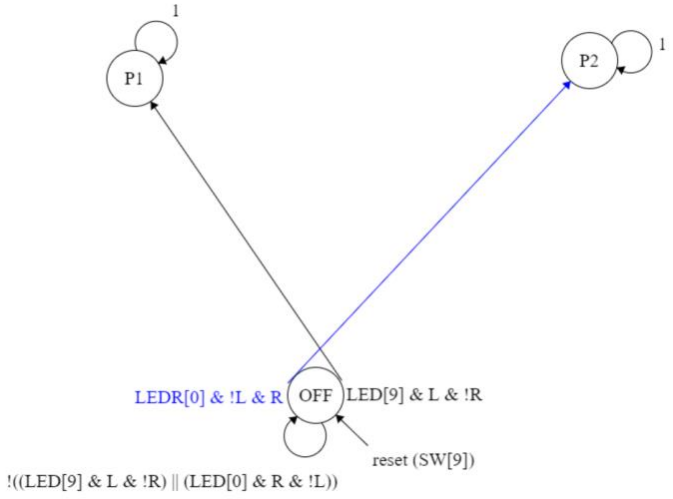
My approach to solving this problem: I started dealing with the metastability of having sequential inputs from the Key0 and Key3, by running them through the doubleflip and then using those doubleflipped values to change states of the circuit. When approaching the tug of war problem, the doubleflipped inputs from Key0 and Key3 are then processed as inputs to the finite state machines, and an output. Since each light LEDR is an output, there needs to be 9 separate finite state machines, all depending on each other. The output to LEDR is ON when the light next to it is ON, and the direction input is towards it. Or more specifically, if the LED on the left of the current LED is ON and the input direction is just to the right (only KEY 0 is pressed for that clock cycle), then the current LED is set to ON, and in the next clock cycle it will turn on. Taking this approach for all the LEDs enables the tug of war behavior. Another FSM for victory conditions is set, to run independent of the tug of war, where if either LEDR[9] or LEDR0 is ON and the inputs are left or right respectively, then HEX0 is set to 2 or 1 respectively, ending the game and displaying the winner on HEX0. To reset the game and play again, SW[9] is the reset switch and will be passed to all the different modules as a reset.



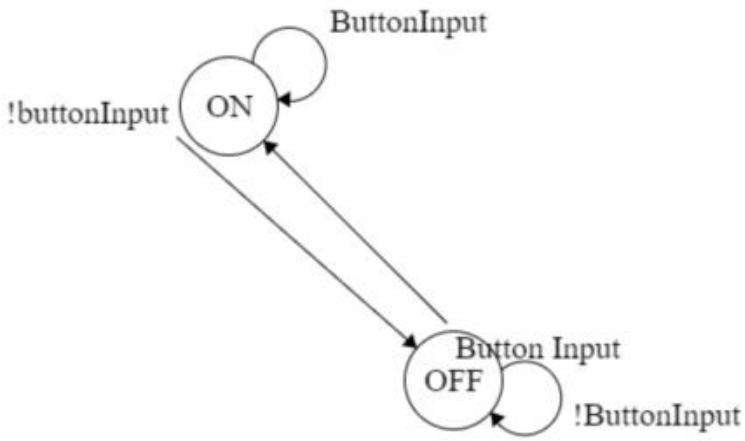
For the FSM above, it represents 9 LEDRs for LEDR 0, 1, 2, 3, 4, 6, 7, 8, 9 where "current light is either one of those lights. NR and NL are the lights next to those LEDs, so in the case of LED 2 it would be NR is LED1 and NL is LED3. With the edge cases, the NR for LED0 is a 1 bit value of Zero and for LED9 the NL is a 1 bit value of Zero. LED 5 is not used for this FSM because it requires a special FSM which will be explained below



The FSM above is for LEDR 5, which the only difference between this FSM and the previous FSM for all the other LEDRs is that it resets to ‘ON’ which essentially begins the tug of war game in the middle.

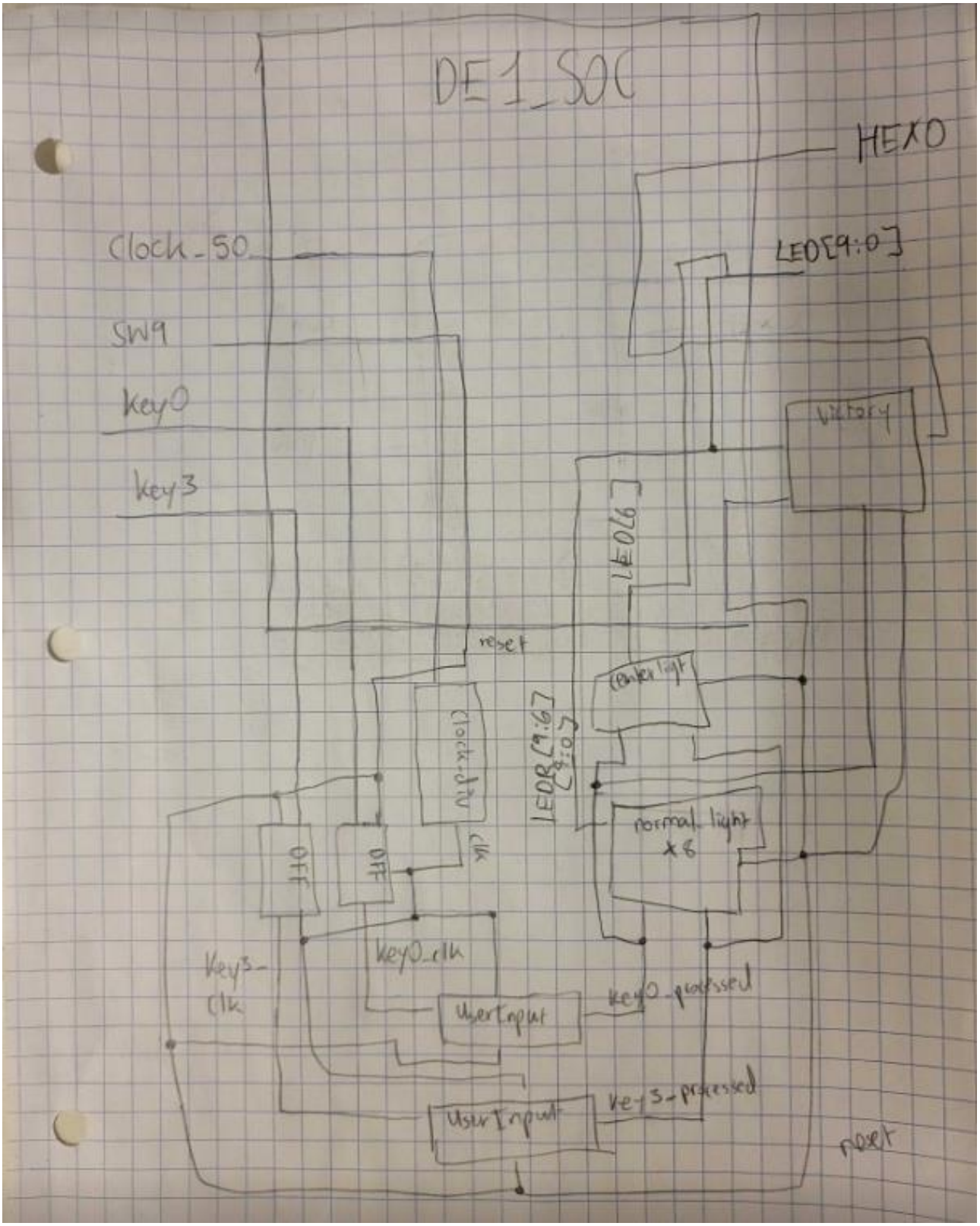


This FSM is for the winning condition. If the current light is on LEDR9 and the input is processed as Left, then the winner is P1. Which is then outputted to the HEX0. If the current light is on LEDR0 and the input is processed as Right, the winner is P2 which is then outputted to the HEX0

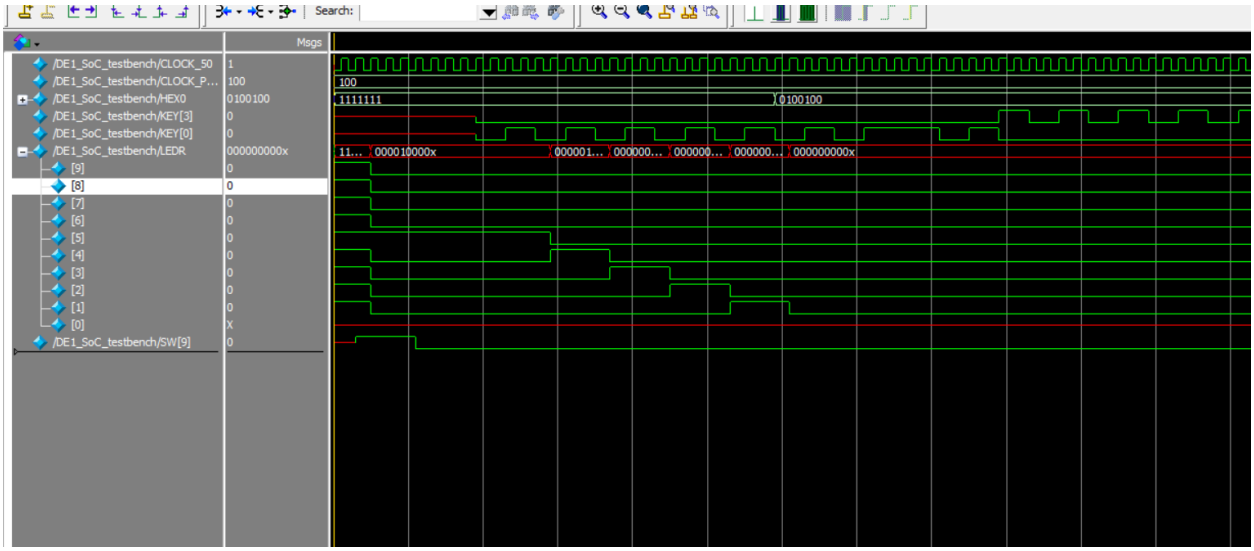


This last FSM exists to simplify the inputs from Key0 and Key3. The button inputs are processed via a doubleflip before this and then processed via this finite state machine, making inputs take 2 clock cycles to be processed, dealing with metastability and simplifying inputs to be easier to process in other FSMs.

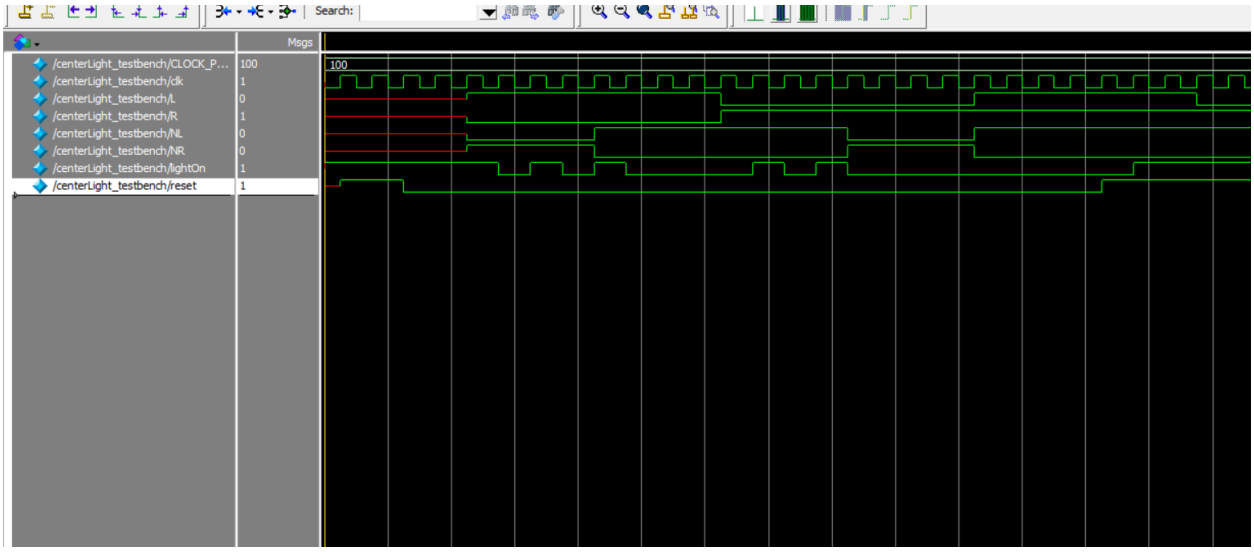
Top Level Block Diagram:



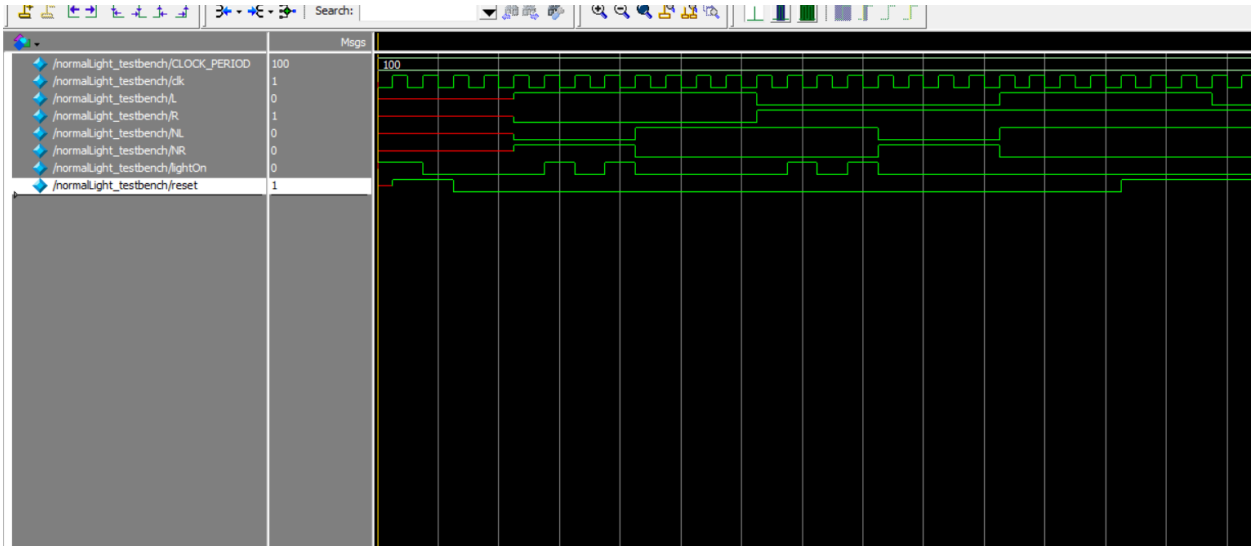
Results



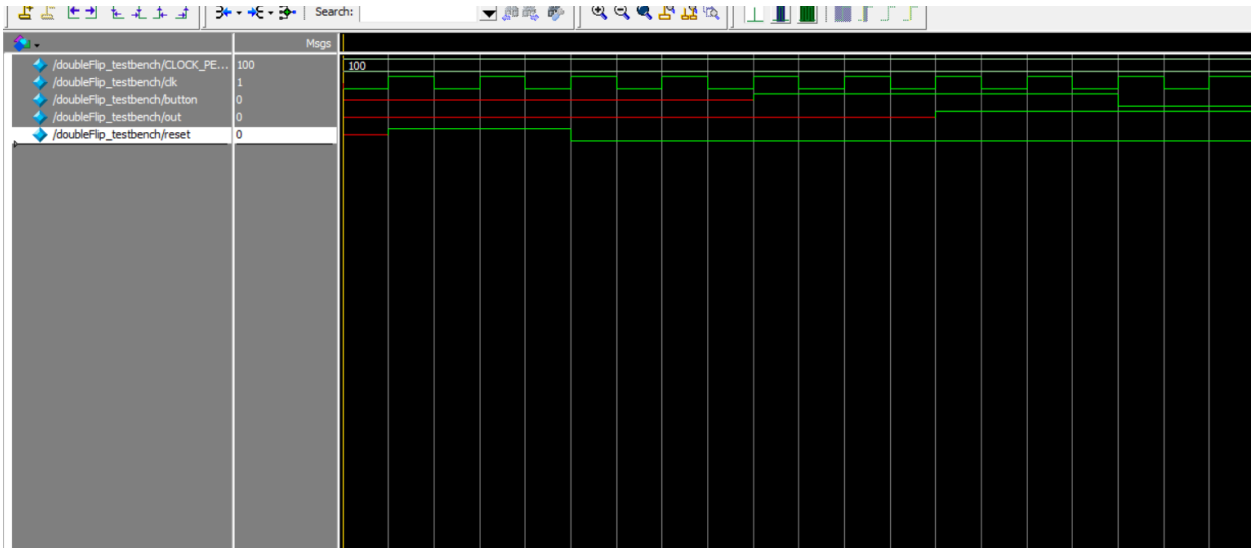
This DE1_SoC simulation tests when users start the game and player 2 presses the button multiple times and wins the game and then HEX0 changes to display the player number that won. Then the game is reset and player 1 presses the button multiple times and wins the game, then HEX0 changes to reflect player 1's victory.



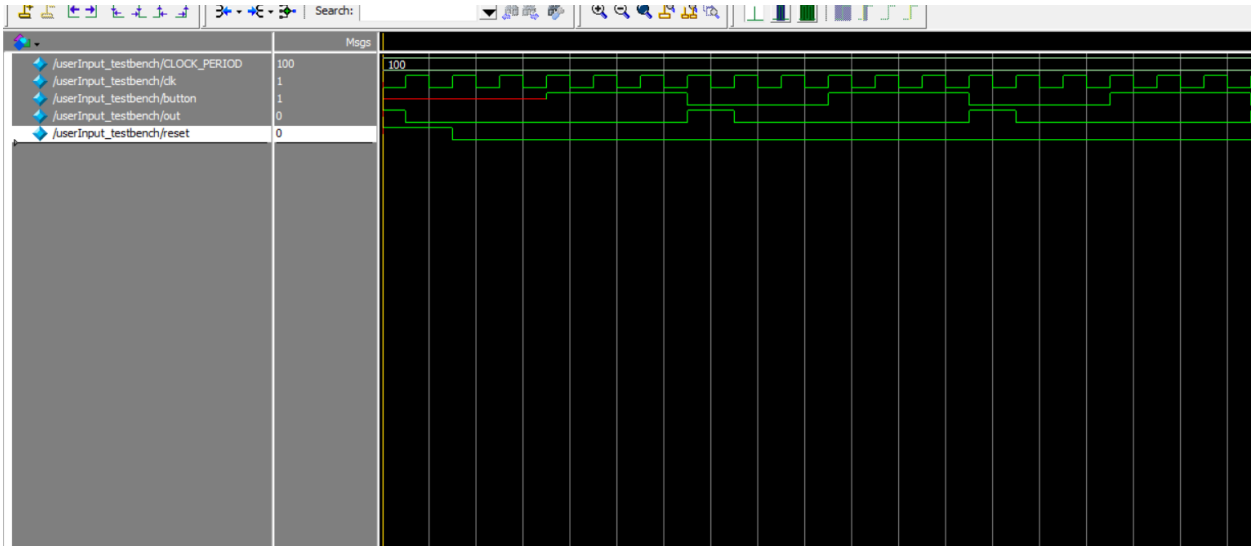
The centerLight simulation tests next state logic for the lights. This module is different from normalLight because on reset, the center light will turn on which can be seen in the simulation.



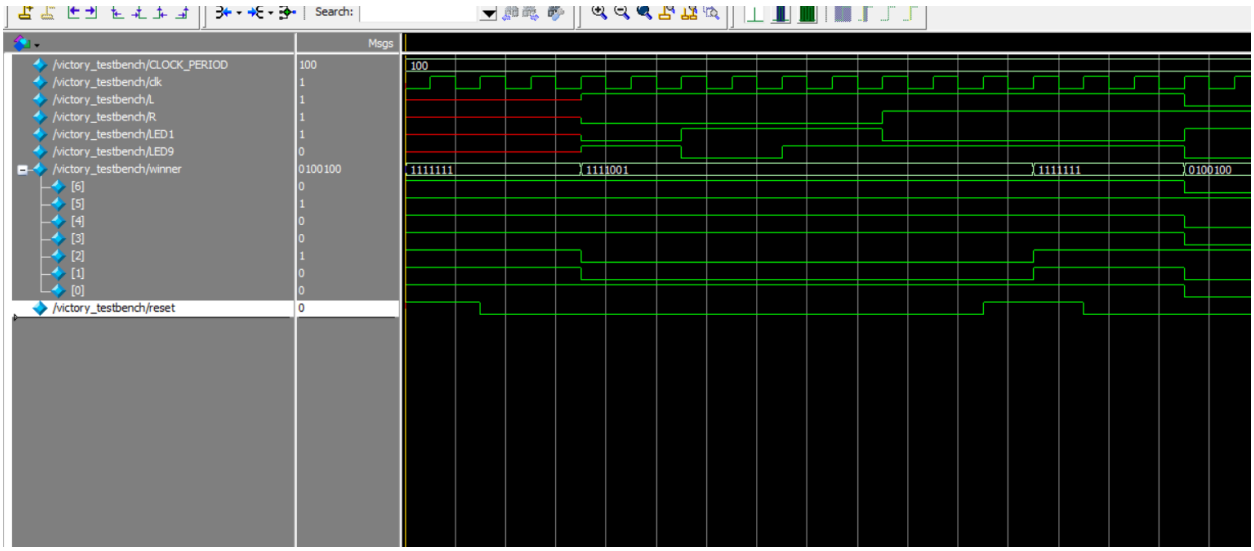
The normalLight simulation tests next state logic for the lights. This module is different from centerLight because on reset, the rest of the lights will be turned off as seen in the simulation.



The doubleFlip simulation tests that when there is a button input, the output is updated to reflect the button input but two clock cycles later which is shown in the simulation. When button is positive, two clock cycles pass on clk and then the output, out, becomes true.

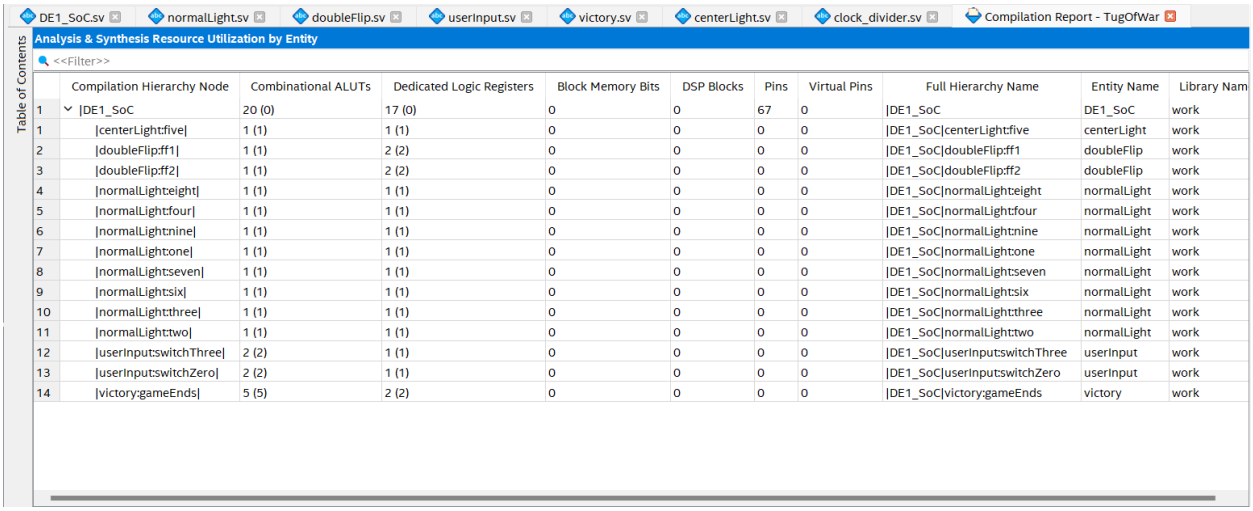


The userInput simulation shows that the button press for tug of war must be a press and release. We can see in the simulation that the button is pressed and becomes one and then 3 clock cycles later once the button is released, then the output, out, returns true.



The victory simulation shows that when the winner is determined, the 7-bit HEX will show the winning player number. When the winner hasn't been determined yet, the HEX display is off. When the game is reset, the HEX display is also reset to off to show that there is a game in progress and no winner has been determined yet.

Resource Utilization by Entity:



DE1_SoC.sv | normalLight.sv | doubleFlip.sv | userInput.sv | victory.sv | centerLight.sv | clock_divider.sv | Compilation Report - TugOfWar

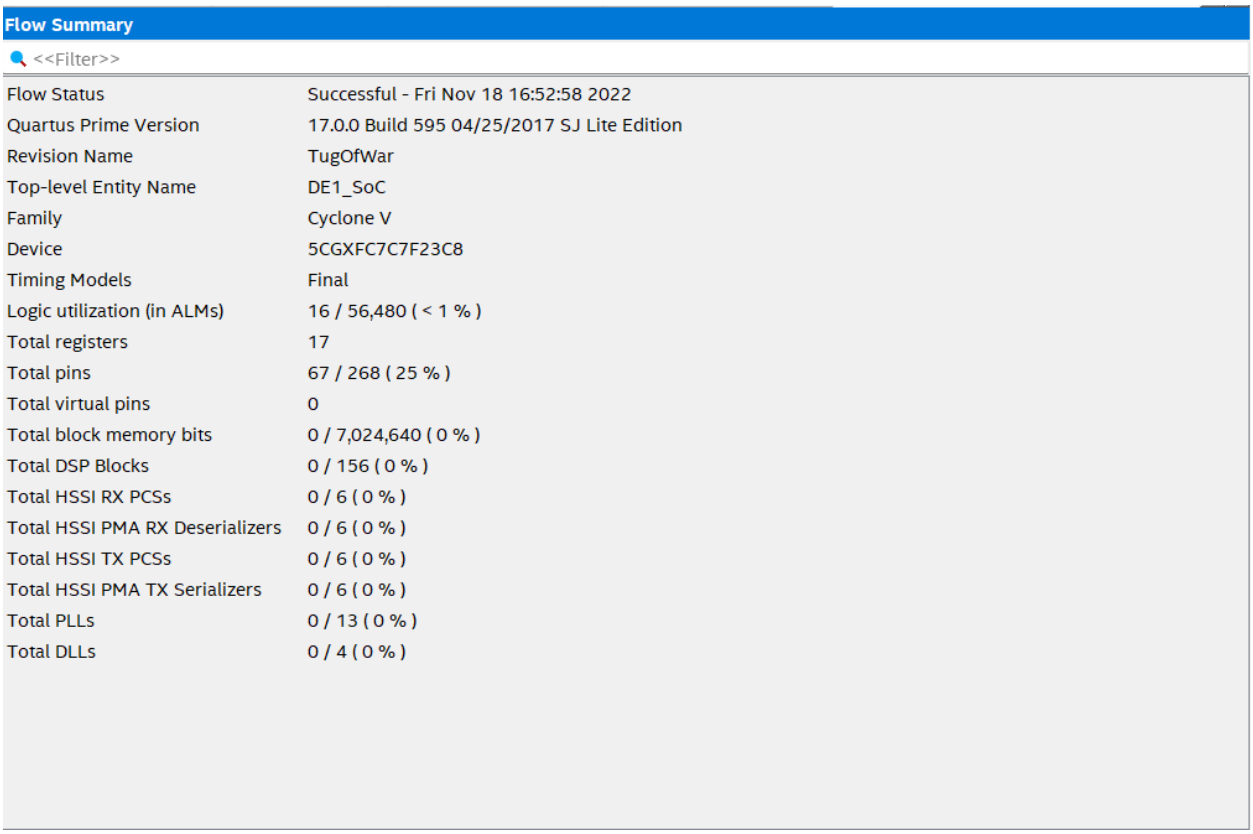
Table of Contents

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 Nam
1	▾ |DE1_SoC	20 (0)	17 (0)	0	0	67	0	|DE1_SoC	DE1_SoC	work
1	|centerLight:five|	1 (1)	1 (1)	0	0	0	0	|DE1_SoC|centerLight:five	centerLight	work
2	|doubleFlip:ff1|	1 (1)	2 (2)	0	0	0	0	|DE1_SoC|doubleFlip:ff1	doubleFlip	work
3	|doubleFlip:ff2|	1 (1)	2 (2)	0	0	0	0	|DE1_SoC|doubleFlip:ff2	doubleFlip	work
4	|normalLight:eight|	1 (1)	1 (1)	0	0	0	0	|DE1_SoC|normalLight:eight	normalLight	work
5	|normalLight:four|	1 (1)	1 (1)	0	0	0	0	|DE1_SoC|normalLight:four	normalLight	work
6	|normalLight:nine|	1 (1)	1 (1)	0	0	0	0	|DE1_SoC|normalLight:nine	normalLight	work
7	|normalLight:one|	1 (1)	1 (1)	0	0	0	0	|DE1_SoC|normalLight:one	normalLight	work
8	|normalLight:seven|	1 (1)	1 (1)	0	0	0	0	|DE1_SoC|normalLight:seven	normalLight	work
9	|normalLight:six|	1 (1)	1 (1)	0	0	0	0	|DE1_SoC|normalLight:six	normalLight	work
10	|normalLight:three|	1 (1)	1 (1)	0	0	0	0	|DE1_SoC|normalLight:three	normalLight	work
11	|normalLight:two|	1 (1)	1 (1)	0	0	0	0	|DE1_SoC|normalLight:two	normalLight	work
12	|userInput:switchThree|	2 (2)	1 (1)	0	0	0	0	|DE1_SoC|userInput:switchThree	userInput	work
13	|userInput:switchZero|	2 (2)	1 (1)	0	0	0	0	|DE1_SoC|userInput:switchZero	userInput	work
14	|victory:gameEnds|	5 (5)	2 (2)	0	0	0	0	|DE1_SoC|victory:gameEnds	victory	work

Flow Summary:



Flow Summary

<<Filter>>

Flow Status	Successful - Fri Nov 18 16:52:58 2022
Quartus Prime Version	17.0.0 Build 595 04/25/2017 SJ Lite Edition
Revision Name	TugOfWar
Top-level Entity Name	DE1_SoC
Family	Cyclone V
Device	5CGXFC7C7F23C8
Timing Models	Final
Logic utilization (in ALMs)	16 / 56,480 (< 1 %)
Total registers	17
Total pins	67 / 268 (25 %)
Total virtual pins	0
Total block memory bits	0 / 7,024,640 (0 %)
Total DSP Blocks	0 / 156 (0 %)
Total HSSI RX PCSs	0 / 6 (0 %)
Total HSSI PMA RX Deserializers	0 / 6 (0 %)
Total HSSI TX PCSs	0 / 6 (0 %)
Total HSSI PMA TX Serializers	0 / 6 (0 %)
Total PLLs	0 / 13 (0 %)
Total DLLs	0 / 4 (0 %)

Brief Overview:

Overall the resulting project fits the specs. When uploaded to the FPGA, a game of tug of war is started and player 1 is assigned to KEY3 while player 2 is assigned to KEY0. Then the two players press their respective buttons as fast as they can until the LED which represents the rope, reaches one of either side. Once a winner is determined then the winning player's number is displayed on HEX0 for viewers and users to see. If players wish to continue playing then switch 9 can be used to reset the game and continue to play.