lab 8 report - pedroa2 - conway game of life

How to use:
The inputs you use are the following
Switch 9 resets the LED BOARD when on
Switch 8 controls the game state, when flipped the game starts and cells starts interacting
Switch 7-5 controls the presets
000 - enables the board to be controlled by the user, more on this later
001 - 110 - loads different presets
111 - clears the board

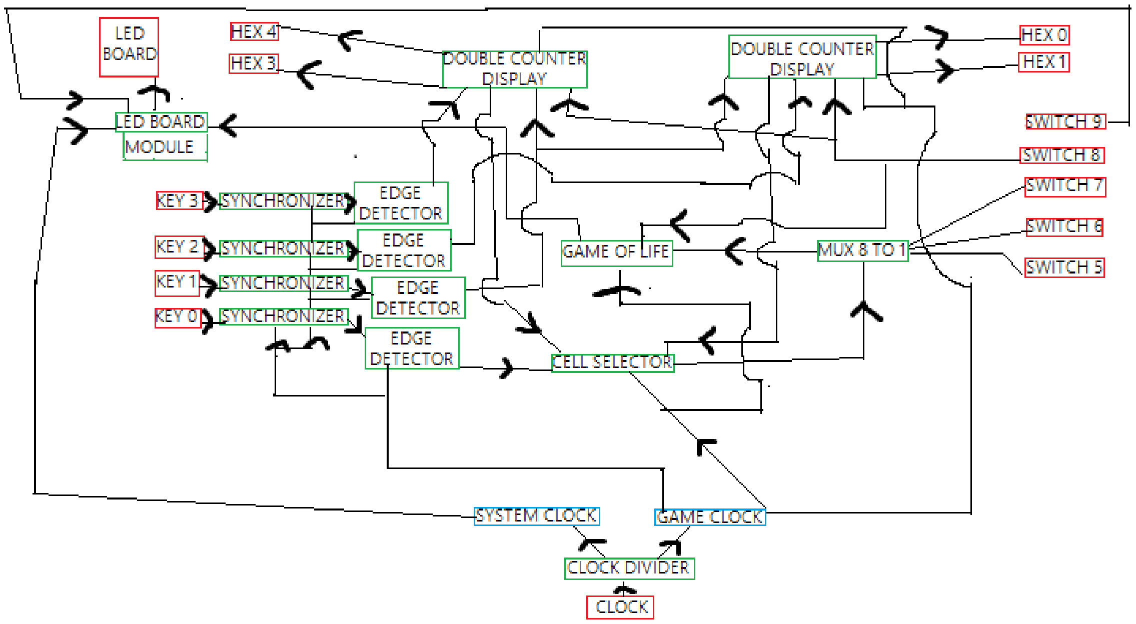
The Keys are used to turn on the lights on the board manually
After resetting the coordinates are initially 0 , 0
Pressing Key 3 increases the y coordinate by one
Pressing Key 2 increases the x coordinate by one
Pressing Key 1 decreases the x and y coordinate by one
Pressing Key 0 marks/demarks the current selected x and y coordinate

You can see the current x and y coordinates on the HEX displays

To reset the board you need to set the switches 7-5 to 111 and then start the game

You can also freeze the board by flipping switch 8 while the game is going on

Block diagram



Interconnections and modules explanations

CLOCK DIVIDER - module given - Utilizes the system clock to output to different clock signals, SYSTEM CLOCK, that is used by the LED BOARD MODULE and GAME CLOCK used by GAME OF LIFE, CELL SELECTOR, DOUBLE DISPLAY COUNTER, EDGE DETECTOR, SYNCHRONIZER

LED BOARD MODULE - given led driver that is used to drive the led board - utilizes the SYSTEM CLOCK, and receives the output from the GAME OF LIFE module to know what leds to turns on, switch 9 is used to reset it. The output is forward to the LED BOARD

SYNCHRONIZER - used to reduce metastability from the in board KEYS, receives a KEY input and outputs to the EDGE DETECTOR. SWITCH 8 resets the output to low. Uses the GAME CLOCK.

EDGE DETECTOR - used to guarantee that holding the KEY does not continually output a 1. Receives input from SYNCHRONIZER and outputs it to the DOUBLE DISPLAY COUNTERS and the CELL SELECTOR. SWITCH 8 resets the output to low. Uses the GAME CLOCK.

DOUBLE DISPLAY COUNTER - used to count the current coordinates of the user selection and also display that on the HEX displays. Receives an edged detected input and outputs to the HEX DISPLAY, also sends out the current count to the CELL SELECTOR. SWITCH 8 resets the counter to 0, both internally and to the HEX display. Uses the GAME CLOCK.

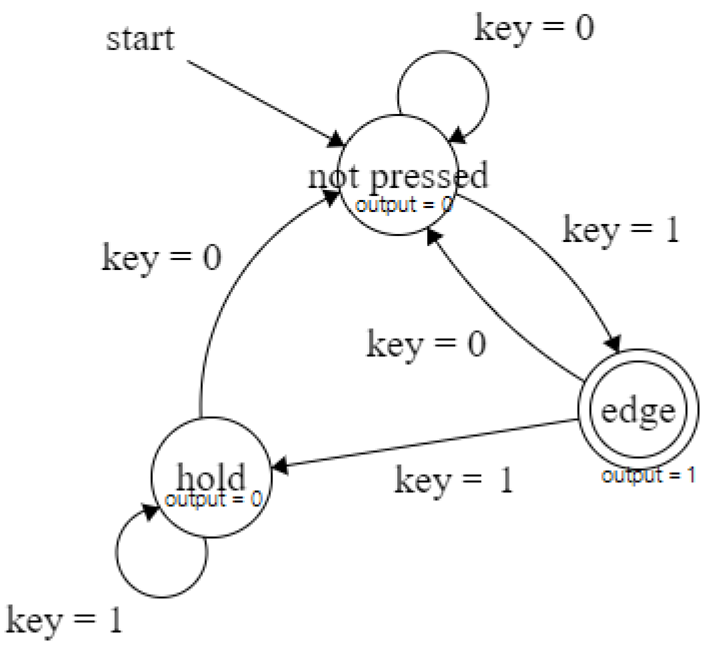
CELL SELECTOR - used to select/deselect a cell, SWITCH 8 resets the output to 0, both internally and to the HEX display. Uses the GAME CLOCK. Takes in the information from the DOUBLE DISPLAY COUNTER to know which cell to select. The input from EDGE DETECTOR KEY 0, marks/unmarks the cell.

MUX8TO1 - This MUX outputs to the GAME OF LIFE what cells are initially on, this varies from the input in SWITCH[7:5], user selection (000), presets (001 - 110) and clear board (111). SWITCH 8 makes it output 0. The output of this mux is sent to the GAME OF LIFE module.

GAME OF LIFE - this is the module that controls the cells and determines how they should behave based on the rules of the game. SWITCH 8 unfreezes/freezes the game. Uses the GAME CLOCK.

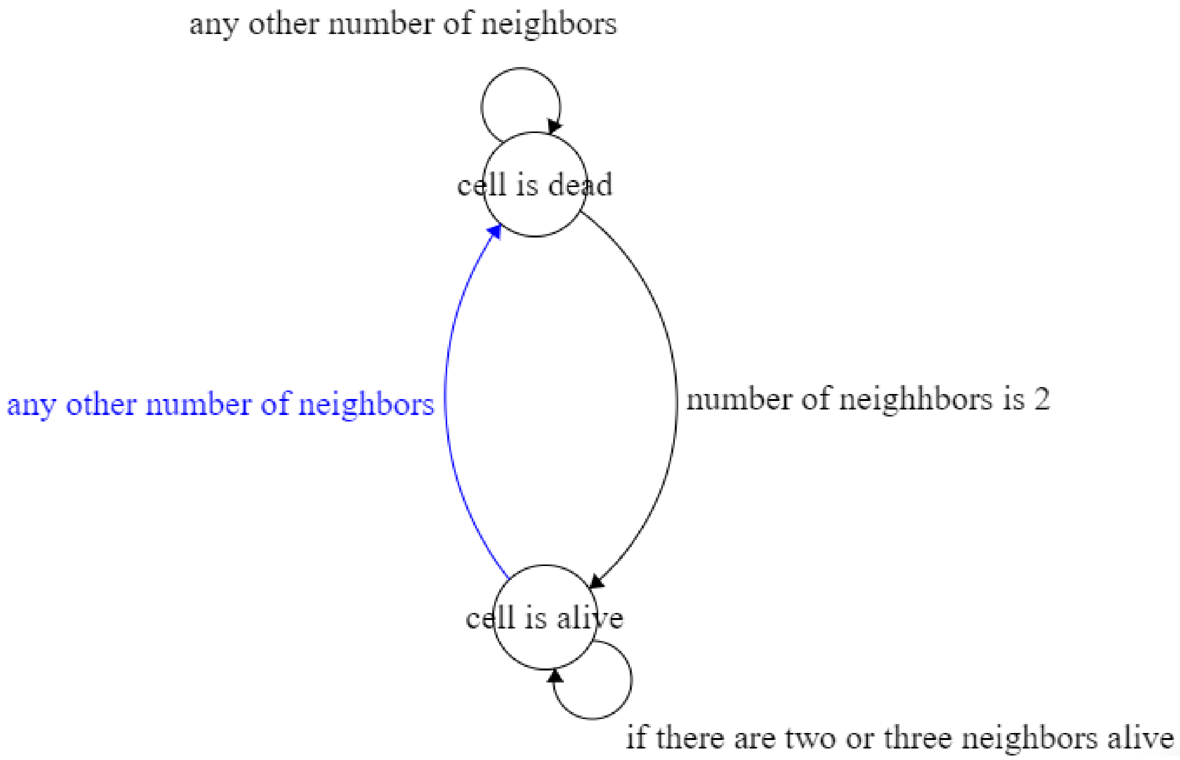
FSMs:

Edge detector



This is my edge detector FSM, we start on not pressed, if key is pressed we go to edge, if key is still pressed we go to hold and stay there until key is released. If key is released ever we go back to not pressed.

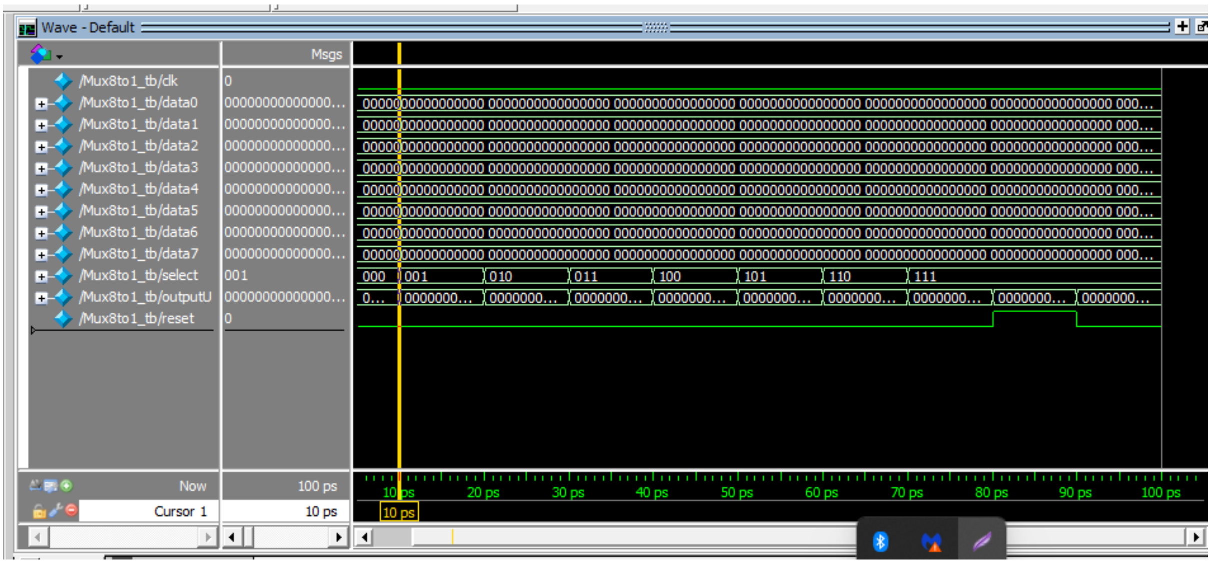
GAME OF LIFE


any other number of neighbors
cell is dead
any other number of neighbors
number of neighhbors is 2
cell is alive
if there are two or three neighbors alive

this fsm is used inside game of life to control the state of the cells.

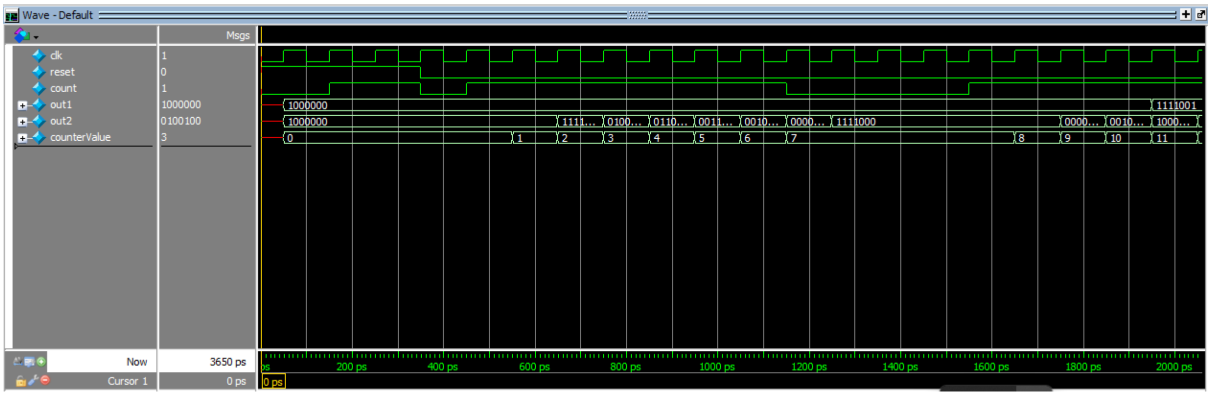
TEST BENCHES:

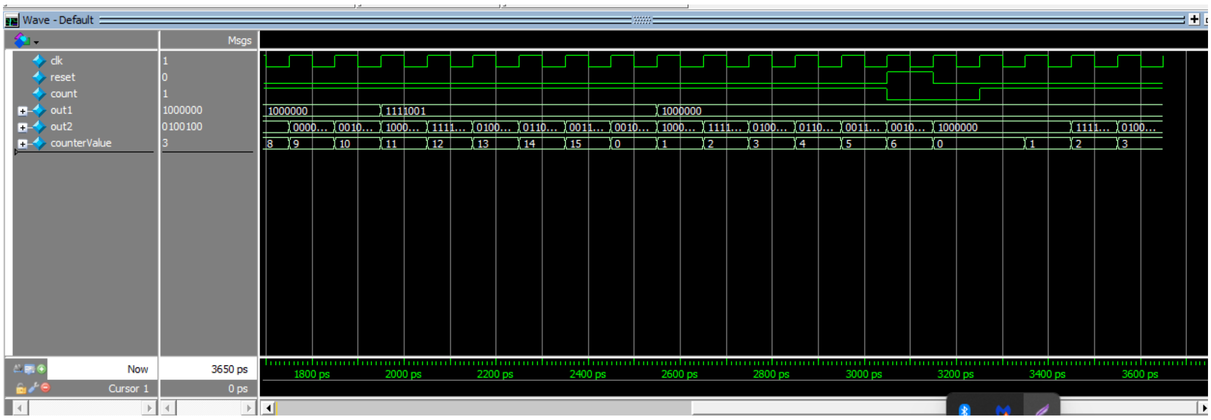
mux_8_to_1


Wave - Default
Msgs
/Mux8to1_tb/clk 0
/Mux8to1_tb/data0 00000000000000...
/Mux8to1_tb/data1 00000000000000...
/Mux8to1_tb/data2 00000000000000...
/Mux8to1_tb/data3 00000000000000...
/Mux8to1_tb/data4 00000000000000...
/Mux8to1_tb/data5 00000000000000...
/Mux8to1_tb/data6 00000000000000...
/Mux8to1_tb/data7 00000000000000...
/Mux8to1_tb/select 001
/Mux8to1_tb/outputU 00000000000000...
/Mux8to1_tb/reset 0
000 001 010 011 100 101 110 111
Now 100 ps
Cursor 1 10 ps
10 ps 20 ps 30 ps 40 ps 50 ps 60 ps 70 ps 80 ps 90 ps 100 ps

For this simulation I tested out every combination of selection and checked if the output is correctly outputted.

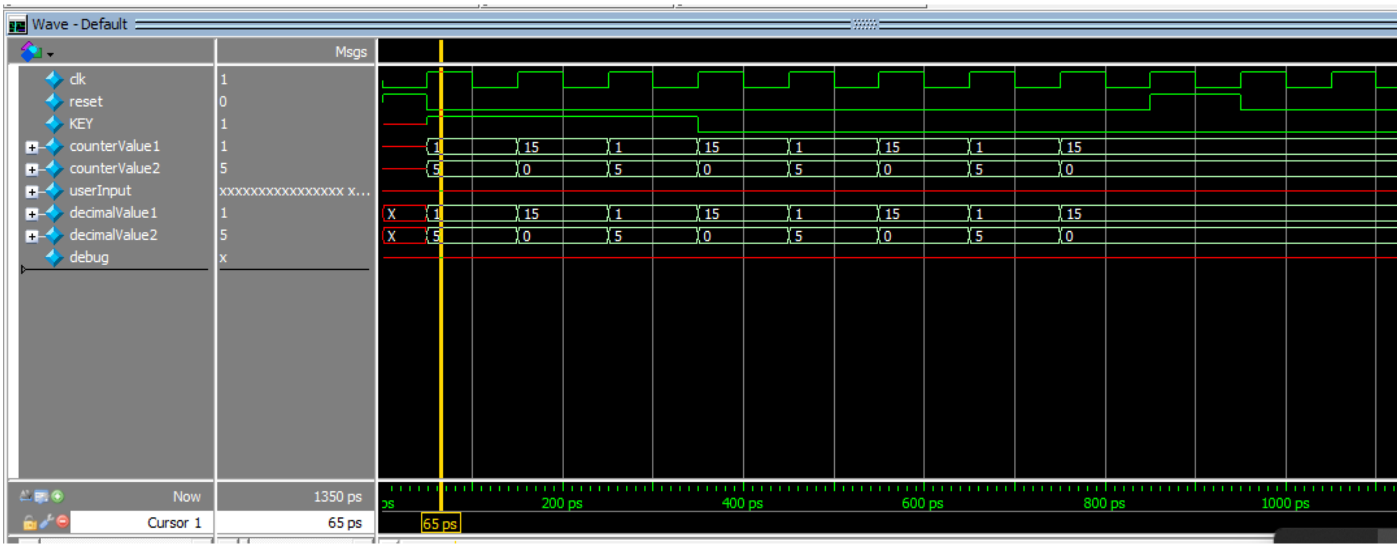
double display counter





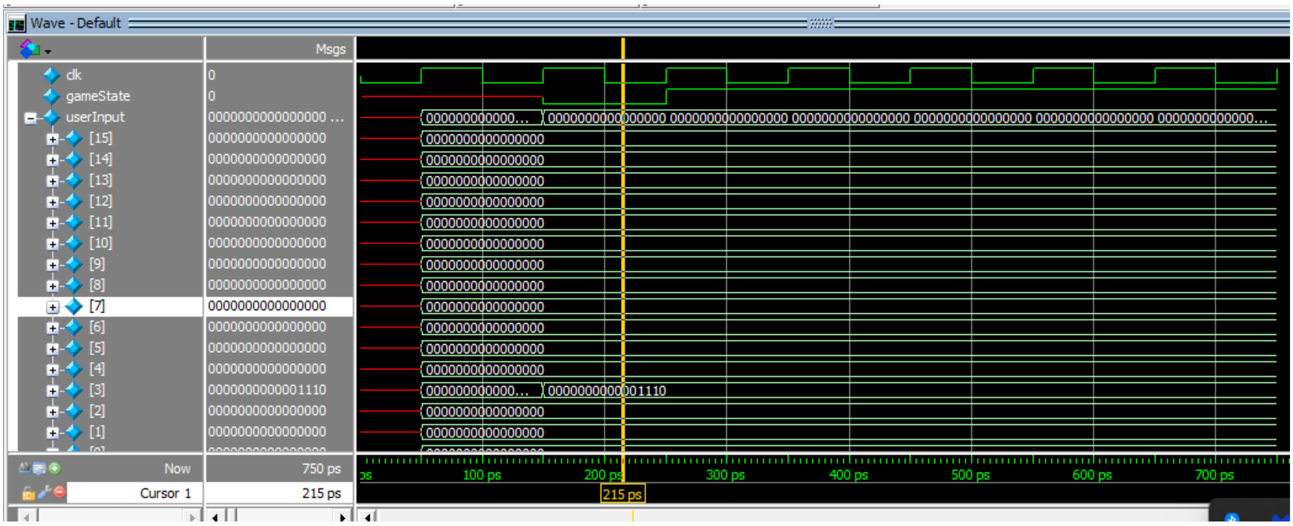
For this module I tested out sending in key inputs and checking if the counter is increasing, and if the output is correct. I also reseted the signal and checked if the counter loops.

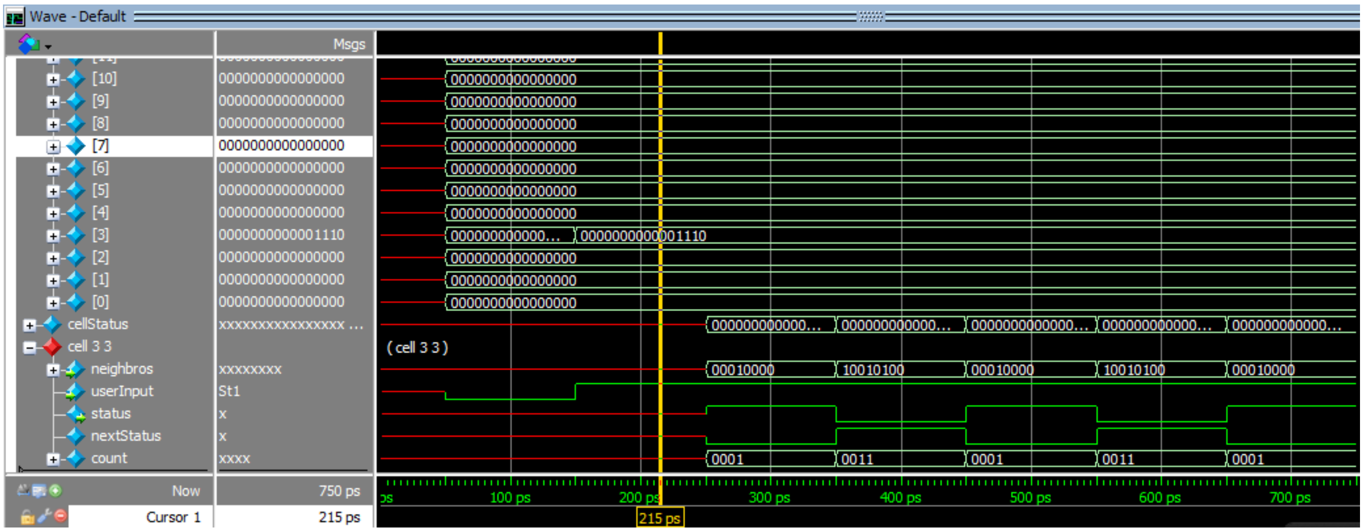
cell selector



For this simulation I inputted different count values and tested out pressing the KEY, and seeing if the output gets correctly outputted.

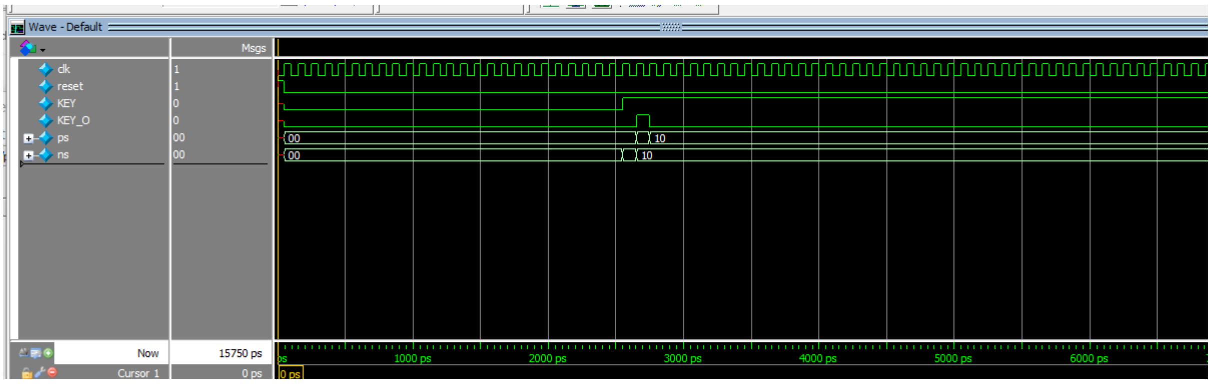
game of life module

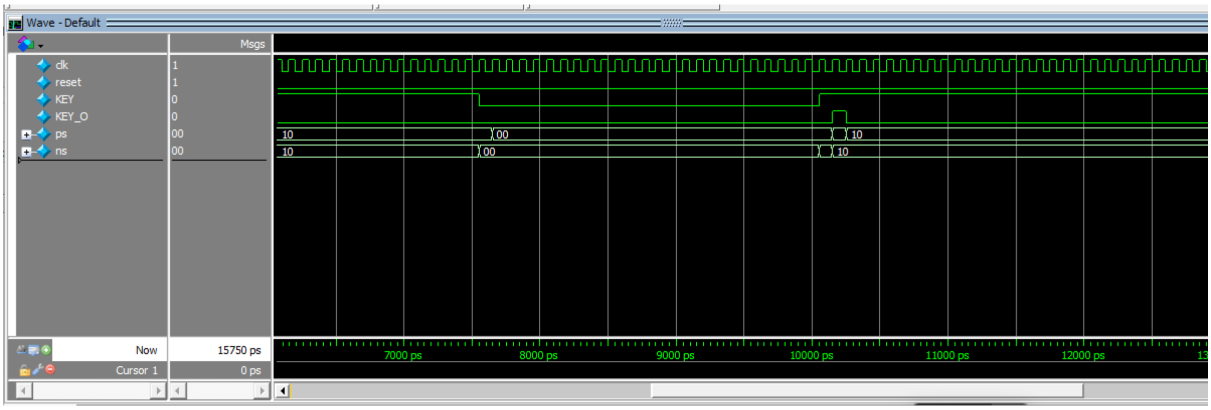


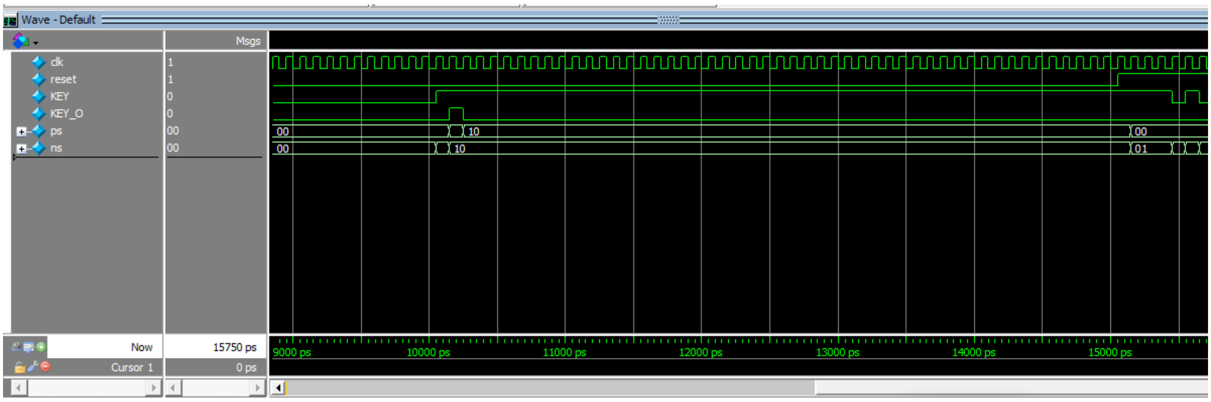


For this module I tested very basic input, it is really hard to use model sim to visualize the game, so I just wanted to see if the basics are working. I Tested how a cell works when receiving different game states. And also tested out user input being able to control the leds.

EDGE DETECTOR model sim

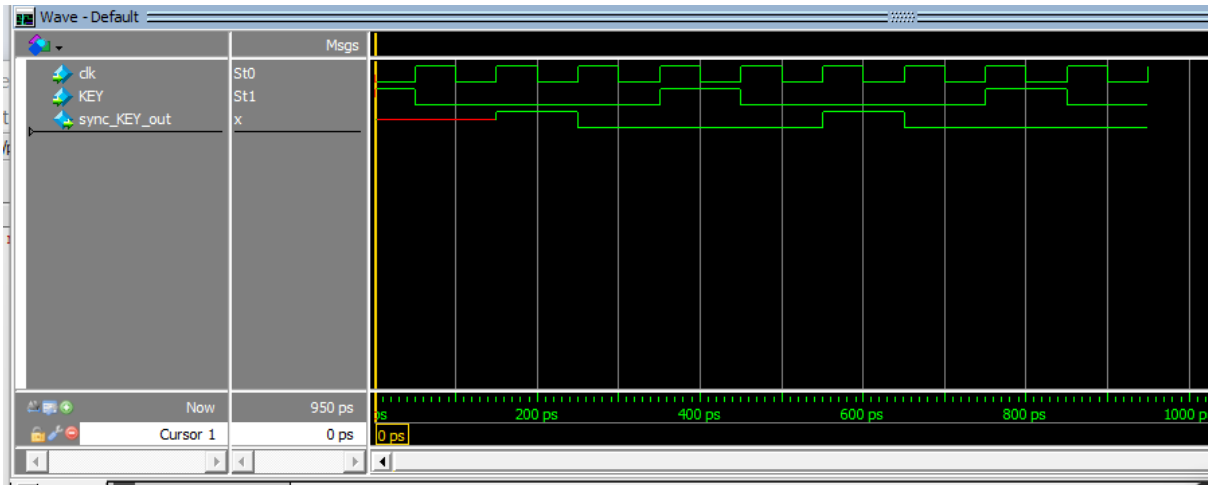






As you can see here, when the key is pressed, we only output 1 (key_o) for only one clock cycle, even though we are holding the key. After testing that I test out every single state transition.

Synchronizer



You can see here that sync_key_out matches Key after going by the two flip flops, which we can verify by looking at the clock cycles.

Testing overview

For testing I made model sims for every single module and even subroutines inside the modules. It is tricky to test out the 16 x 16 double arrays thus initially i worked on 4 x 4 grid. After getting good results on my simulations I compiled to the board, behavior wasn’t exactly right, so it was a process of reverse engineering the wrong behavior , adding it to a testbench, correcting the bugs, checking the sim again and compiling to the board. To finally check if the game was working correctly I tested presets and compared to online versions of conway game of life.

Around 25 hours of work