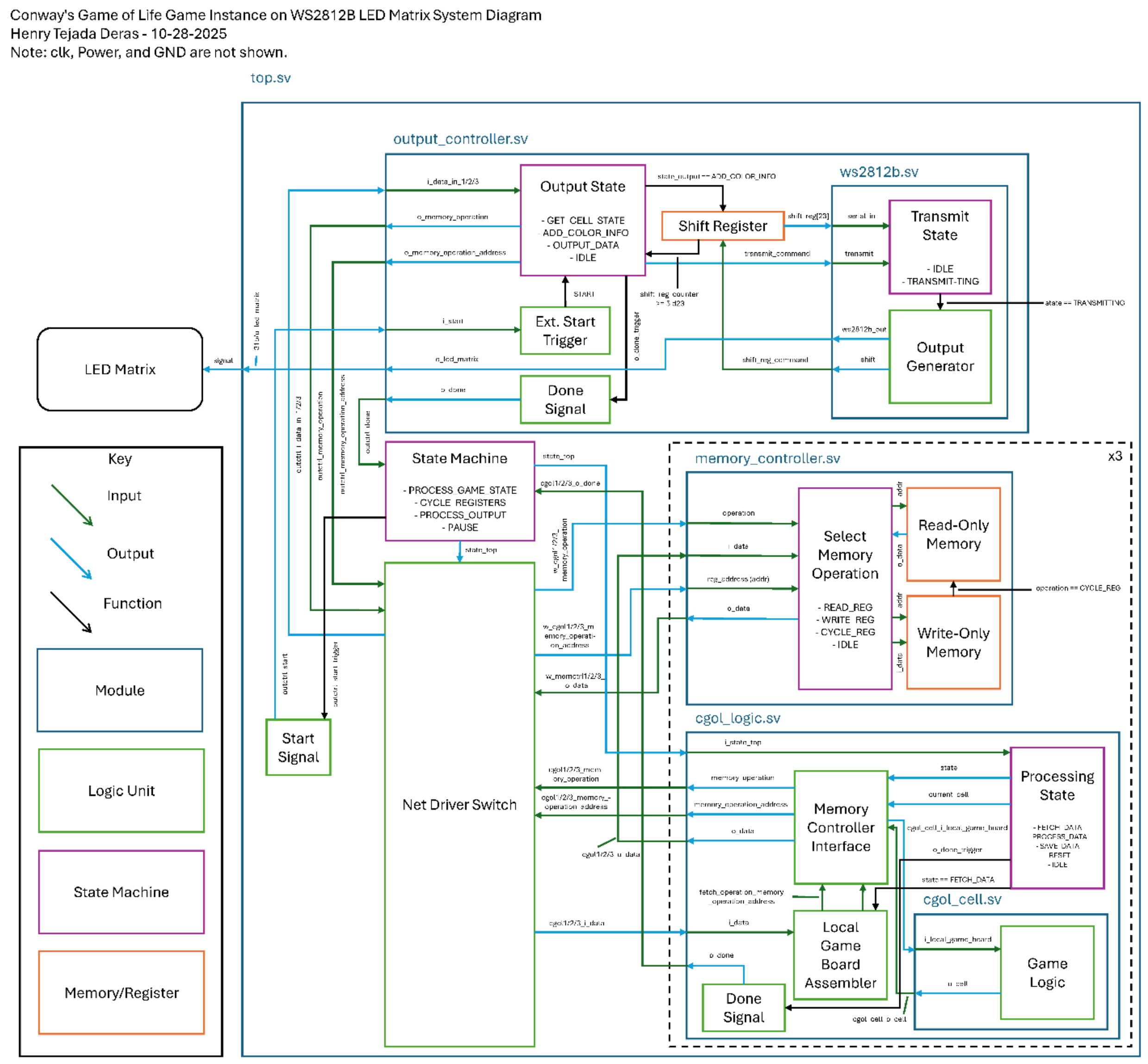
Henry Tejada Deras 10-29-2025 ENGR3410-01.25FA

Mini Project 3 Submission

System Diagram:



To view this system diagram in more detail, please go here: https://olincollege-my.sharepoint.com/:p:/g/personal/htejada_olin_edu/EaaUBnRbaMhNvk4j1_2gIHUBYVxtz3J0_aYpkC93r-WPDw?e=8OP5iW

System Description:

The system can be split up into a few sections, the section to calculate the next game state, the section to save the game states, and the section to output the game state on the LED matrix. The section that calculates the next game state is comprised of two modules, cgol_logic.sv and cgol_cell.sv. The cgol_logic.sv module is essentially a wrapper for the cgol_cell.sv module that actually does the calculations for the next state on a cell-by-cell basis. The role of the cgol_logic.sv module is to interface with the state machine in top.sv and the memory controller in memory_controller.sv. The section that stores the game states is comprised of only one module, memory_controller.sv. That module implements two memory banks, one accessible by other modules as a read-only memory and another accessible by other modules as a write-only memory. The reason for this is to maintain the current and future game states separate to avoid unintended behaviors. The section that outputs the game state on the LED matrix is comprised of two modules, output_controller.sv and ws2812b.sv. The output_controller.sv module is a wrapper module for the ws2812b.sv module to interface with the state machine in top.sv and the memory controller. This is used to keep the ws2812b.sv module running in sync with the main state machine which is the one in top.sv. The state machine in top.sv is used to produce the next game state and to display it on the LED matrix. The state machine in cgol_logic.sv is to get the necessary data for the cgol_cell.sv module and other timing things to make sure the data gets saved to the memory and so the entire 8x8 grid of cells is calculated for the next generation. The state machine in output_controller.sv is to make sure the output for the LED matrix is sent in the correct timings and that the LED matrix driver (ws2812b.sv) gets the next bit to send out at the right time. Most of this implementation comes from working around memory issues that varied from not being able to put register entries into a module's port, not being able to use a 2D array due to not being able to pass it through a module's port, not being able to use packed arrays due to Yosys limitations, among other things I needed to learn when I was trying to implement writing and reading functionality to the memory controller.

<u>Game of Life Progression:</u>

Toad Tester (toad_tester.bin)	Default Seed (default_seed.bin)
00000000	01000010
00000000	00000101
00000000	00001001
00011100	00010100
00111000	01011100
00000000	10001000
00000000	00000001
00000000	00000000
--------	--------
00000000	00000010
00000000	10000111
00001000	00000110
00100100	00100010
00100100	00110000

```
00010000                                10010100
00000000                                00000000
00000000                                00000000
--------                                --------
00000000                                00000100
00000000                                00000000
00000000                                00000000
00011100                                00110110
00111000                                01101000
00000000                                00111000
00000000                                00000000
00000000                                00000000
--------                                --------
00000000                                00000000
00000000                                00000000
00001000                                00000000
00100100                                01101100
00100100                                01000000
00010000                                01101000
00000000                                00010000
00000000                                00000000
--------                                --------
00000000                                00000000
00000000                                00000000
00000000                                00000000
00011100                                01100000
00111000                                10001100
00000000                                01010000
00000000                                00110000
00000000                                00000000
--------                                --------
00000000                                00000000
00000000                                00000000
00001000                                00000000
00100100                                01000000
00100100                                10011000
00010000                                01100000
00000000                                00110000
00000000                                00000000
--------                                --------
00000000                                00000000
00000000                                00000000
00000000                                00000000
00011100                                00000000
00111000                                10110000
00000000                                01001000
```

```
00000000
00000000
--------
00000000
00000000
00001000
00100100
00100100
00010000
00000000
00000000
--------
00000000
00000000
00000000
00011100
00111000
00000000
00000000
00000000
--------
00000000
00000000
00001000
00100100
00100100
00010000
00000000
00000000
--------
00000000
00000000
00000000
00011100
00111000
00000000
00000000
00000000
--------
00000000
00000000
00001000
00100100
00100100
00010000
00000000
```

```
01110000
00000000
--------
00000000
00000000
00000000
00000000
01110000
10001000
01110000
00100000
--------
00000000
00000000
00000000
00100000
01110000
10001000
01100000
01010000
--------
00000000
00000000
00000000
01010000
01100000
10010000
11010000
01100000
--------
00000000
00000000
00000000
01100000
11010000
00010000
10010000
11000000
--------
00000000
00000000
00000000
11000000
10010000
11111000
11100000
```

```
00000000
--------
00000000
00000000
00000000
00011100
00111000
00000000
00000000
00000000
--------
```

```
11000000
--------
00000000
00000000
00000000
11000000
00001001
00001001
00000001
10100000
--------
```

Demo Video: https://olincollege-my.sharepoint.com/:v:/g/personal/htejada_olin_edu/EZZmrwWwQSVMhHfW4NdLSLUBOA-tyqEGQkRZG3AUfQX33Q?e=sKcfQ8

System Verilog Files: https://github.com/H-TejadaDeras/ENGR3410-01.25FA/tree/main/assignments/mini%20project%203