CSE369 Flappy Bird

Noah Staveley

How to Play:

Switch 9, is the game restart button. Flicking switch 9 on and off again restarts the game.

The goal of the game is to align the bird within the gaps in between the green pipes simulated on the LED Expansion board. You will see 1-3 pipes displayed, depending on the input from the difficulty level switch.

There is a game play clock that can be adjusted using switch 5. This controls the speed at which the green pipes move across the display board towards the flappy bird.

Every time the bird moves through the gap and clears a pipe, the score will increment. If the bird crashes into a pipe or moves out of bounds, the score will reset.

Extra Features:

1 – Two Game Modes

Switch 8 is the mode switch. This game has 2 modes. When switch 8 is OFF, hold = sink. This means when you press and hold key0, the bird will only fly once, and then "gravity" will pull the bird down one LED at every positive edge of the simulated gravity clock.

When switch 8 is ON, hold = fly. This means when you press and hold key0 the bird will fly up one LED at every positive edge of the simulated clock (same clock as gravity clock).

2 – Difficulty levels

You will see 1-3 pipes displayed, depending on the input from the difficulty level switch.

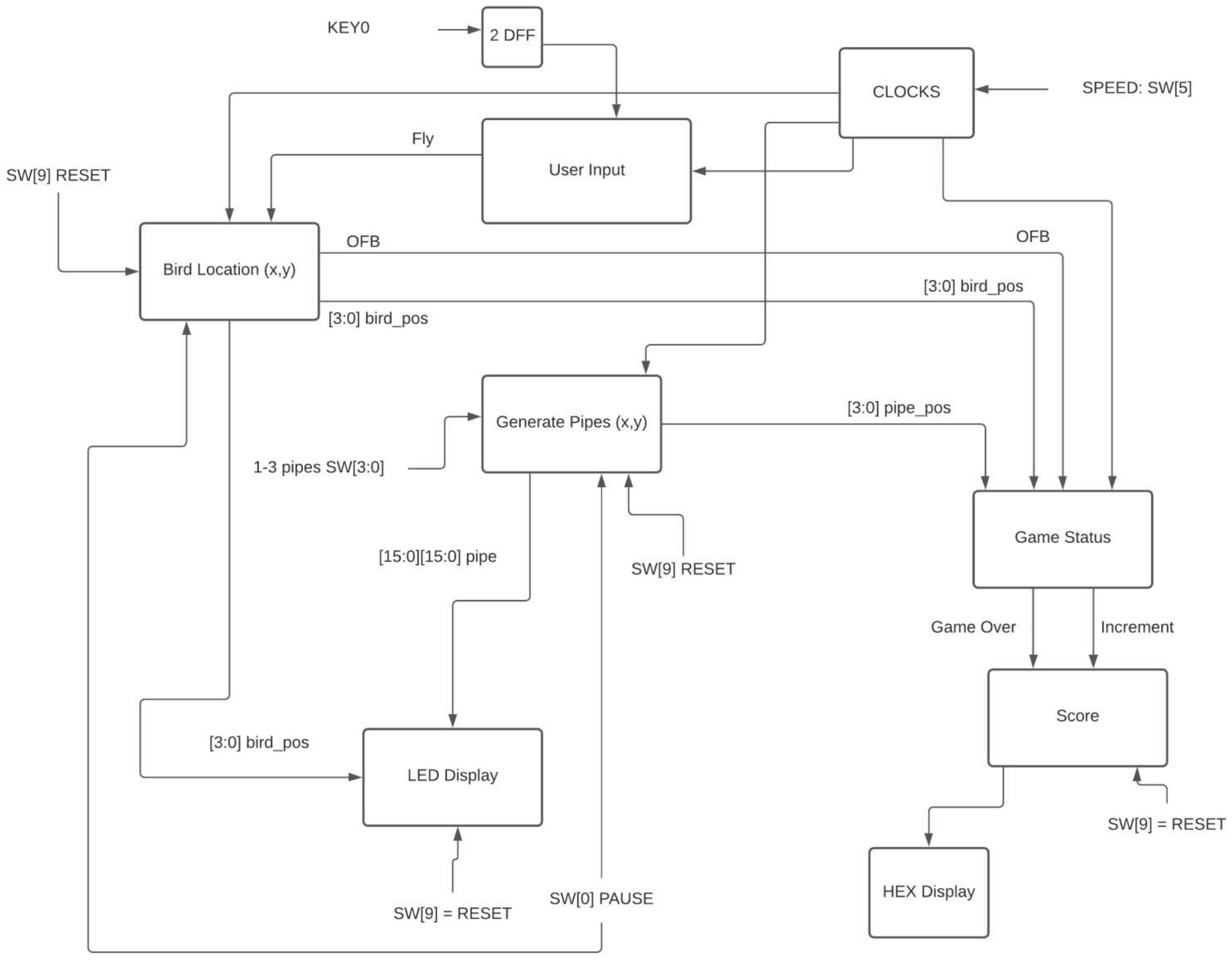
SW[1] ON = level 1, SW[2] ON = level 2, SW[3] ON = level 3.

3- Speed Levels

There are 2 speed levels. When SW[5] is OFF, speed = standard. When SW[5] is ON, speed = fast.

4 – Pause Game

You can pause the game by switching on SW[0]. When SW[0] is ON, the bird and the pipes will stop moving. When the switch is flicked back OFF, the game will continue.



The **user input module** sends the key pressed signal through two flip flops and sets a fly signal that is sent to the bird location module. This module also takes in the Mode switch which determines if fly is high when the button is held down.

The **bird location module** takes in the fly signal from the user input module. When the fly signal is HIGH the bird's y coordinate will increase, otherwise it will decrease on every positive edge of the "gravity" clock.

This module keeps track of the bird height and detects if the bird goes out of bounds. The bird's height is sent to the game status module and the LED display module. If the bird goes out of bounds the OFB signal is set to HIGH. OFB signal is sent to the game status module.

The **generate pipes module** will generate a column with a randomly generated gap size and position.

For example, if the randomly generated gap size = 4 and the randomly generated gap top = 7:

LED[4]-LED[7] will be off and the remaining LEDS in the column will be ON (and set as green).

I will generate 1-3 different pipes depending on the difficulty mode switches. Each pipe will have their own random gap size and position. On every positive edge of the clock, each pipe will move one position to the left until it is out of screen.

The **LED display module** will take in the bird location and pipe locations. It will turn the specified bird LED on as RED. It will also display the green pipes with their randomly generated gaps.

The **game status module** takes in the OFB signal from bird location. If OFB signal is HIGH, the game status module will set the game over output signal to true. This module will detect if a pipe has been cleared or if there was a collision. If a pipe is cleared the increment signal is set to HIGH. If there is a collision, the game over signal is set to HIGH.

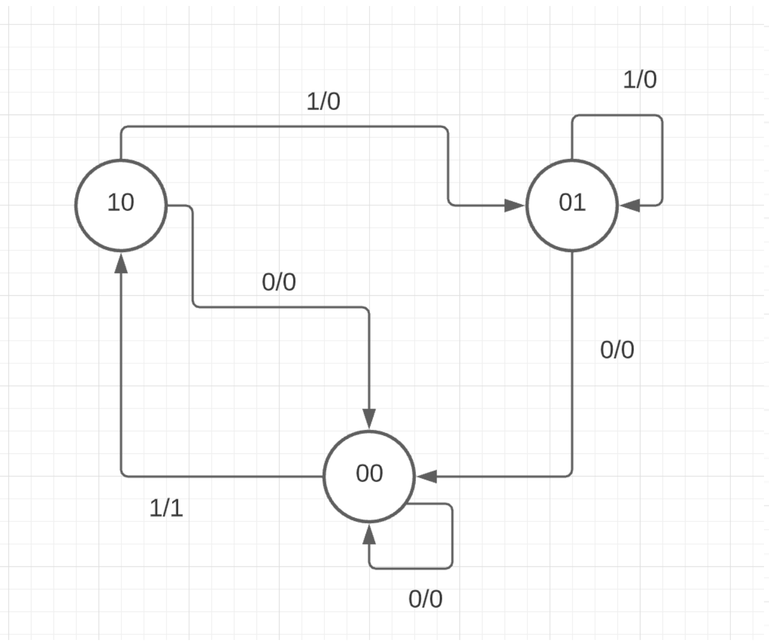
The **score module** takes 2 input signals from the game status module, the game over signal and the increment signal. It also takes the SW[9] reset signal. When game over or reset is HIGH, the score will reset. When increment is HIGH, the score will increment. Score outputs 3 4bit signals to HEX display.

The **HEX display module** takes three 4bit score encodings representing 9 different 7 bit patterns from the score module. Hex Display lights up the appropriate segments on HEX0, HEX1 and HEX2.

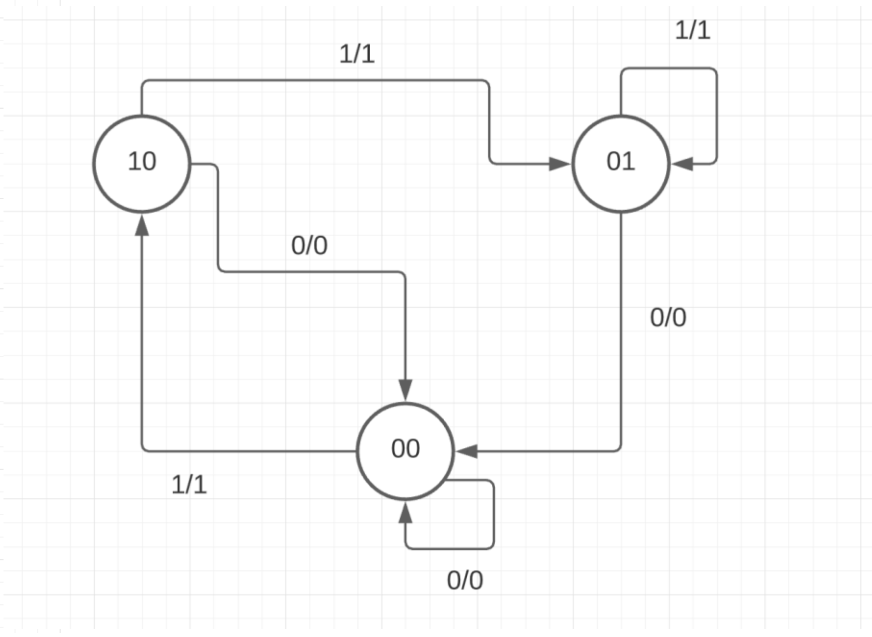
User Input Module:

This module sets a fly signal based on key0 input. 2 modes: hold = fly and hold = sink.

Mode input = 0: hold = sink



Mode input = 1: hold = fly



HOLD = SINK:

There are 3 states needed to recognize when the key has been pressed, the off state encoded as 00, the on state (10) and the hold state (01).

In state 10 if the FSM receives a 0 input, it will move to state 00 and output 0. If the FSM receives a 1 input, it will move to state 01 and output 0.

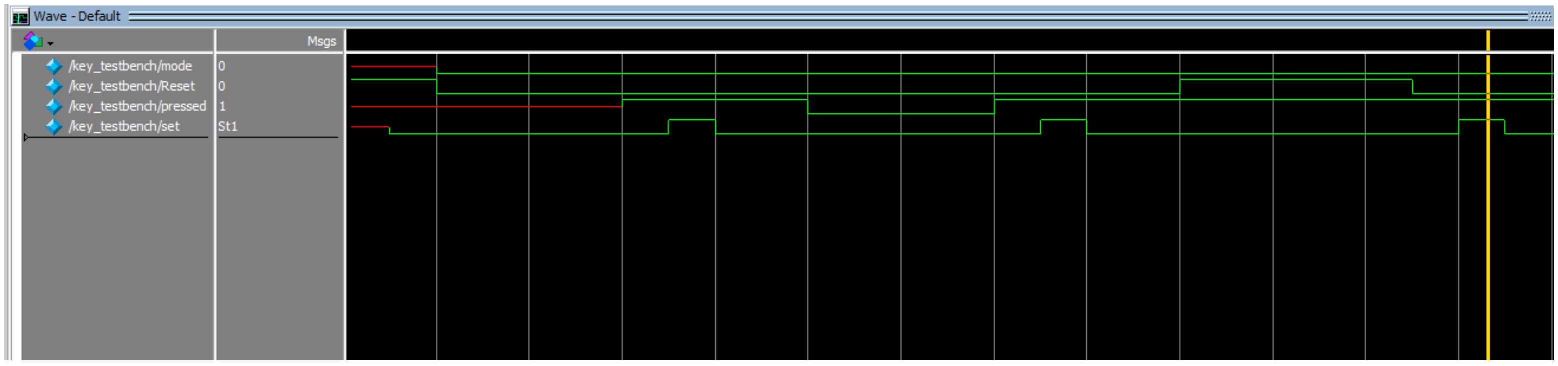
In state 01, if the FSM receives 1, it will stay in state 01 and output 0. If it receives 0, it will move to state 00 and output 0.

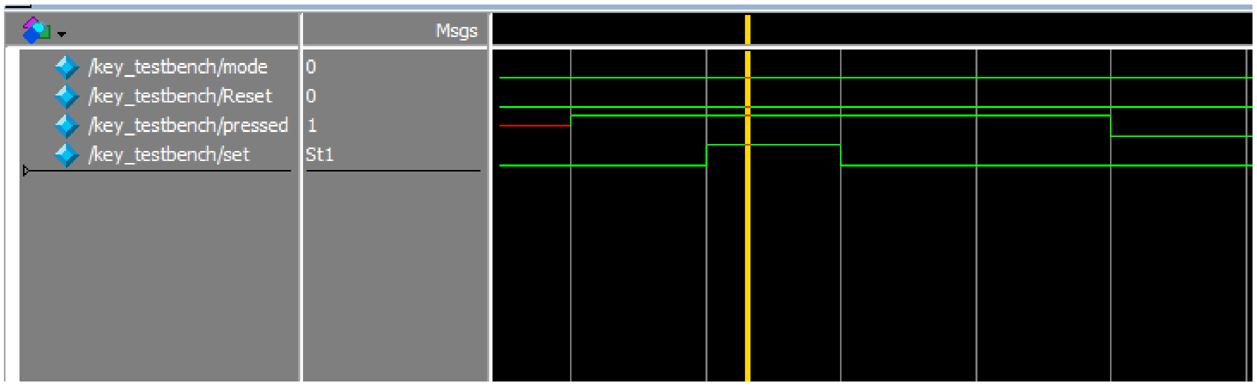
In state 00, if the FSM receives a 0 input, the FSM stays in state 00 and outputs 0. If the FSM receives a 1, it moves to state 10 and output 1.

HOLD = FLY:

The only difference between this FSM and the hold = sink FSM, is that when entering state 01 (the hold state), the FSM will output 1. This will allow the bird to fly while the key is being held down.

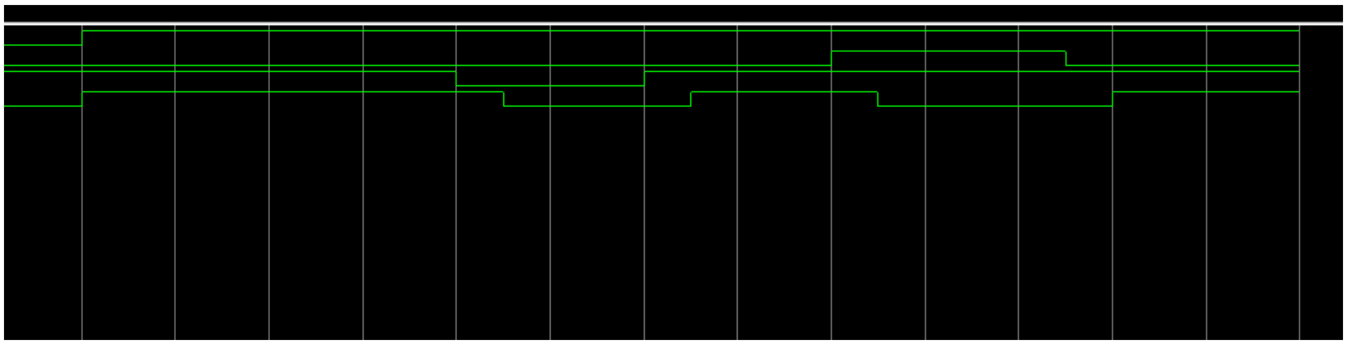
First I am testing the FSM when mode = 0. This is the default mode of the game (when SW[8] is OFF)

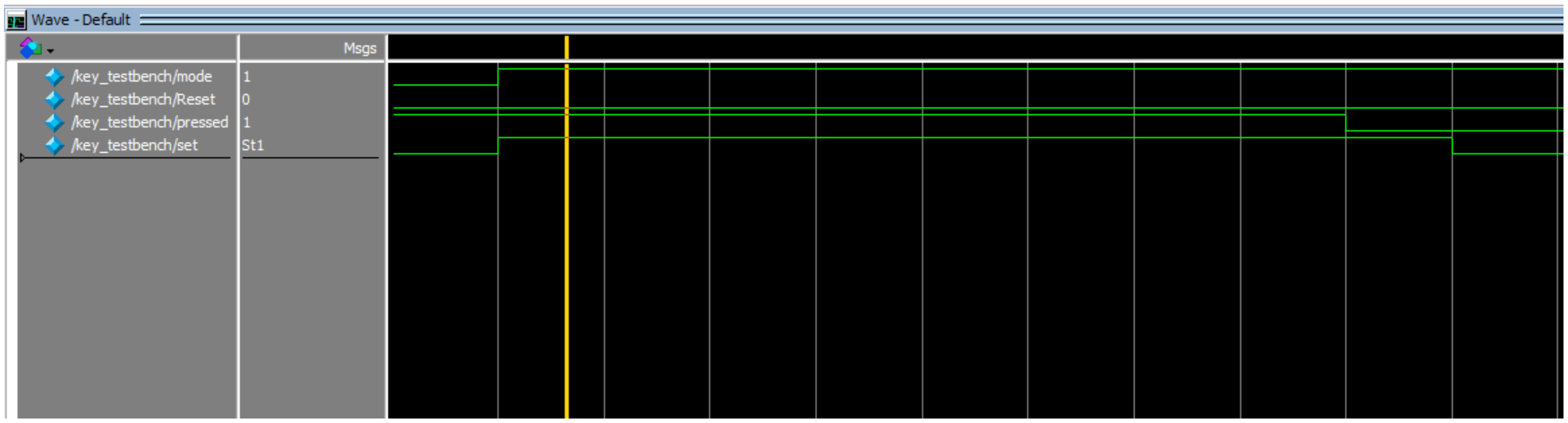




Here you can see when pressed is HIGH, set will be HIGH after 1 clock cycle. Notice that set only stays HIGH for 1 clock cycle. This is because on the next clock cycle we are in the hold state. In the hold state, the output is set to whatever mode is. Since mode = 0, we see set = 0 in the hold state.

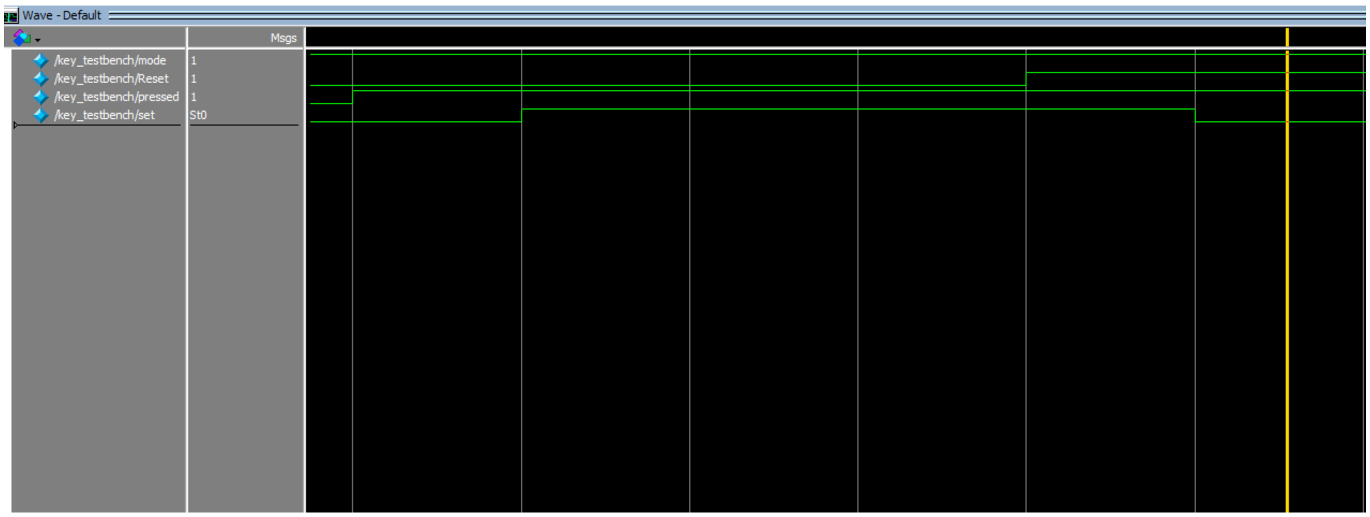
Here I am testing when mode = 1.



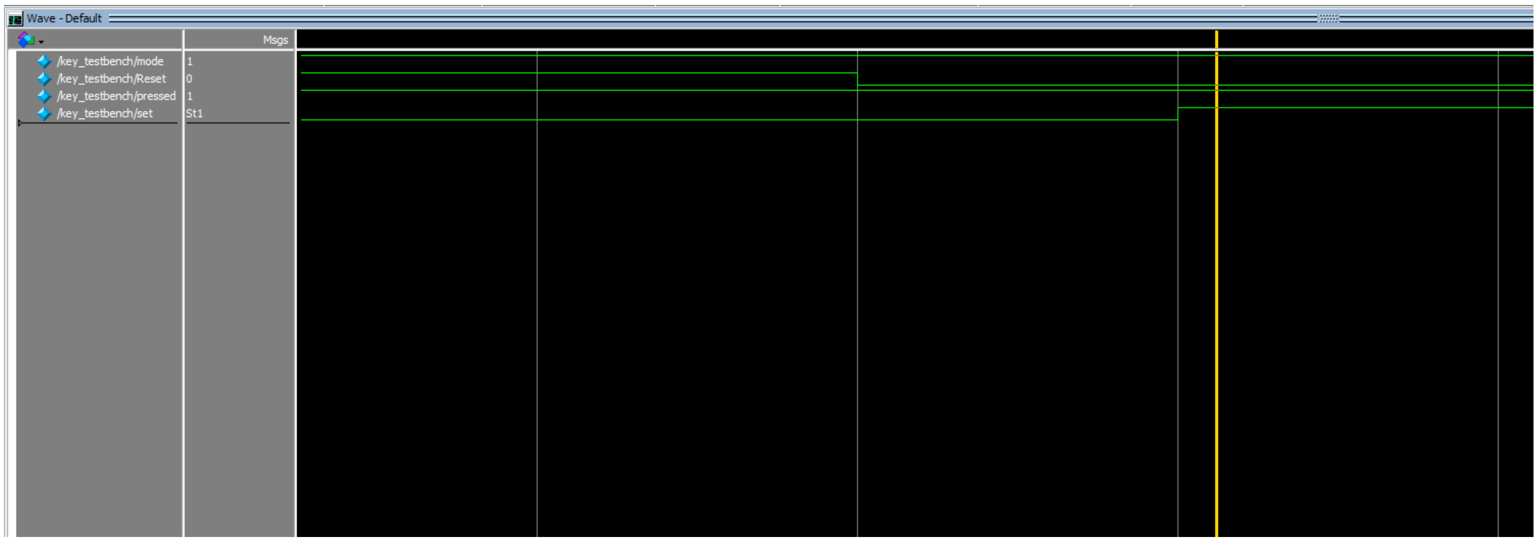


When mode = 1, set stays HIGH until pressed = 0, then after one clock cycle set = 0.

Here, I test the reset when mode = 1.

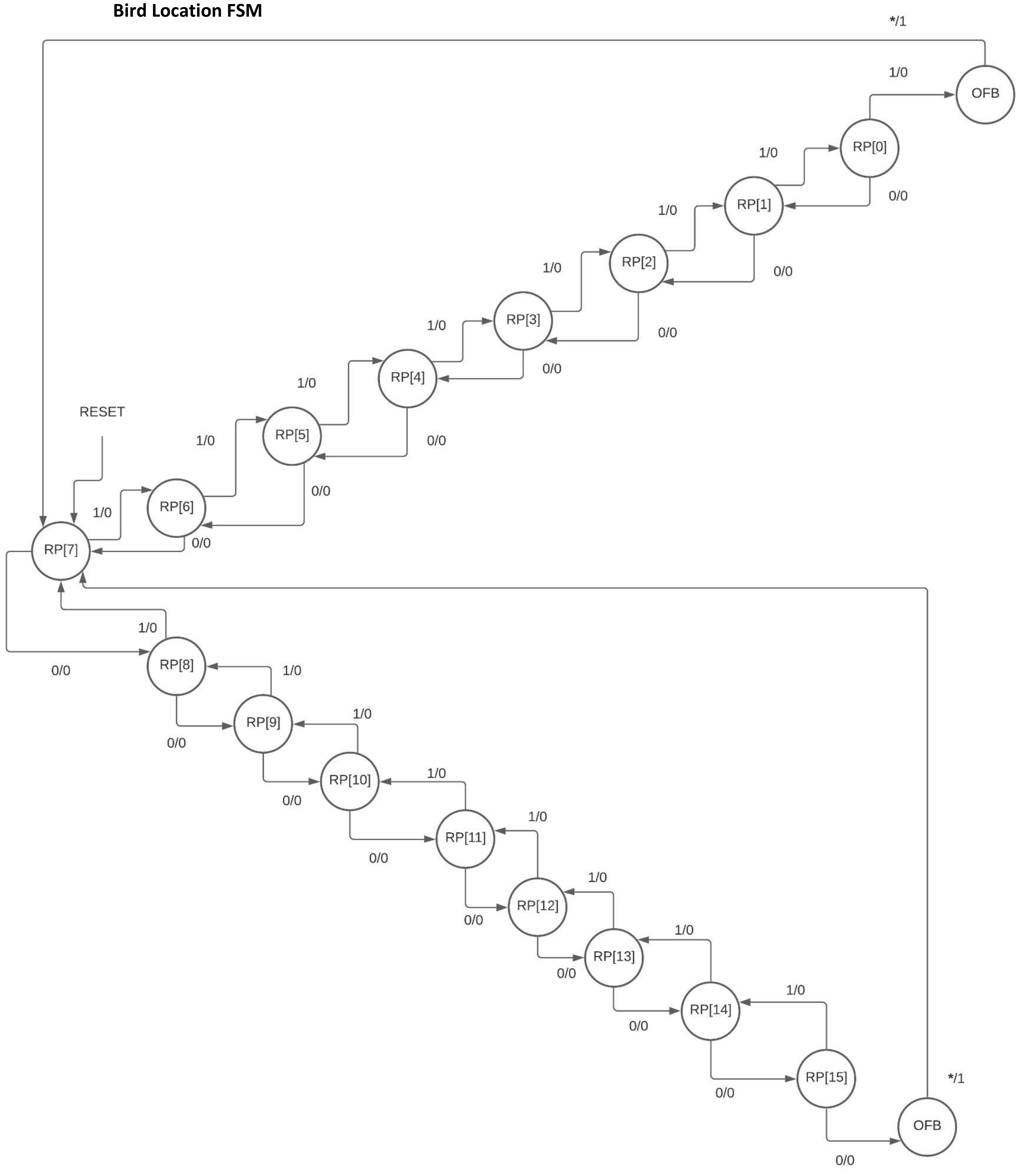


When pressed is set to HIGH, set turns HIGH until reset is HIGH, then after 1 clock cycle set = 0.



Here reset is turned off, and after 1 clock cycle, set is HIGH again.

Bird Location FSM



This FSM will output 1 when the bird has gone out of bounds. Notice this can occur when the FSM is in state RP[0] and receives a HIGH fly signal, or when the FSM is in state RP[15] and receives a LOW fly signal.

The FSM will output 1 in the OFB state. It will then reset to the initial position.

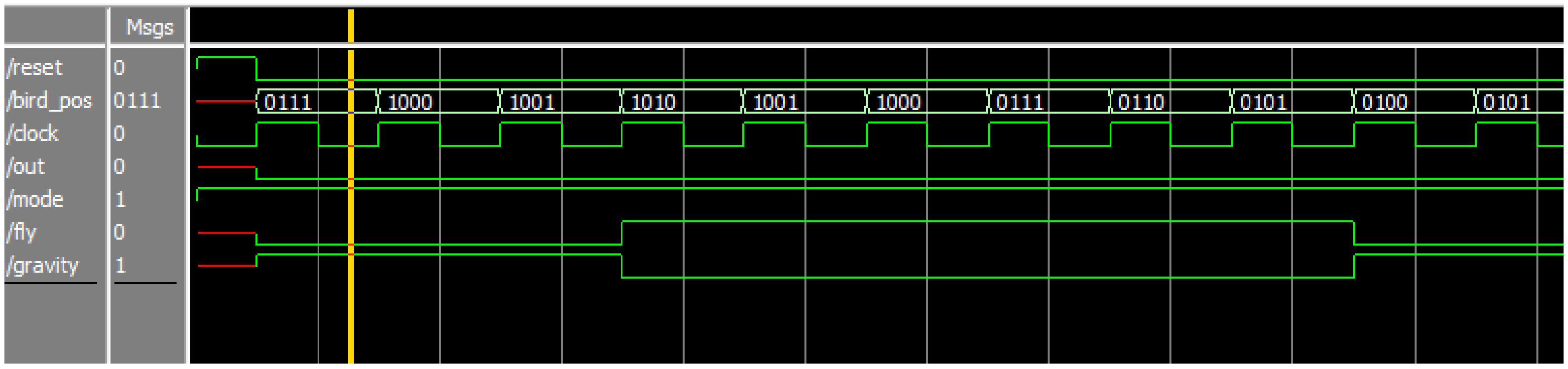
The module will also output the 4-bit bird position (0-15), which represents the vertical index of the bird.

Turning Reset, ON and OFF, sets the bird in the initial, position (index 7).

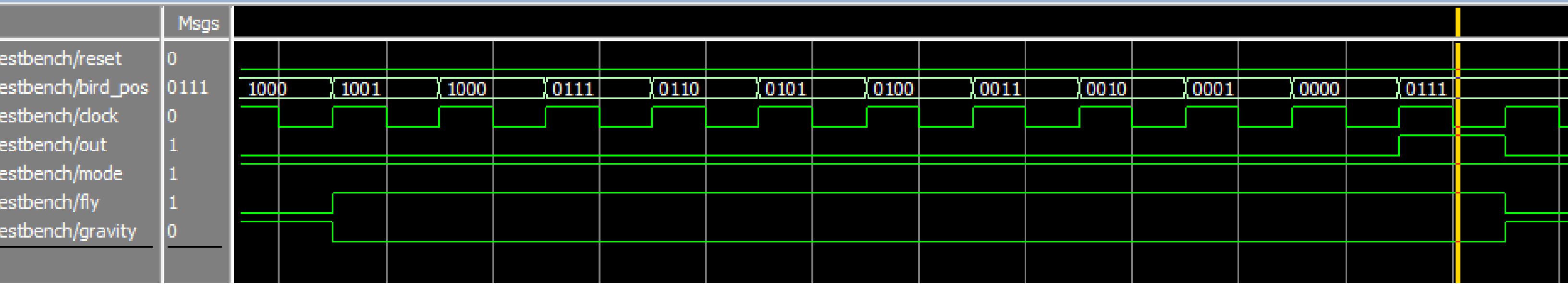
You can see when fly = ON and gravity = OFF, the bird index will decrease, this indicates the bird is flying since the highest LED has an index of 0.

When fly = OFF and gravity = ON, the bird index will increase as the bird sinks at every positive edge of the clock.

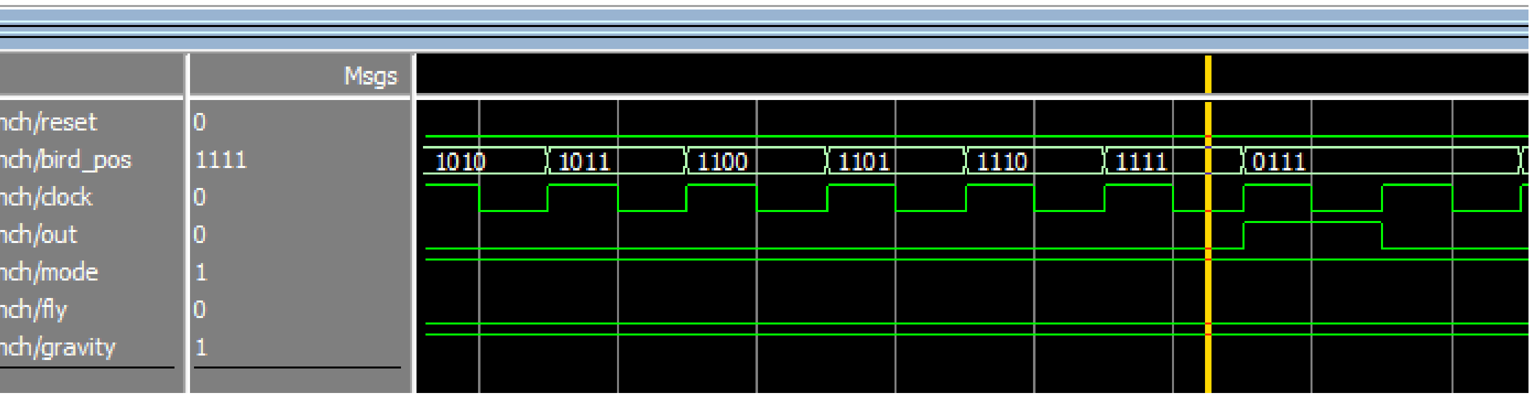
When both fly and gravity are ON and when both fly and gravity are OFF, the bird index will remain unchanged.



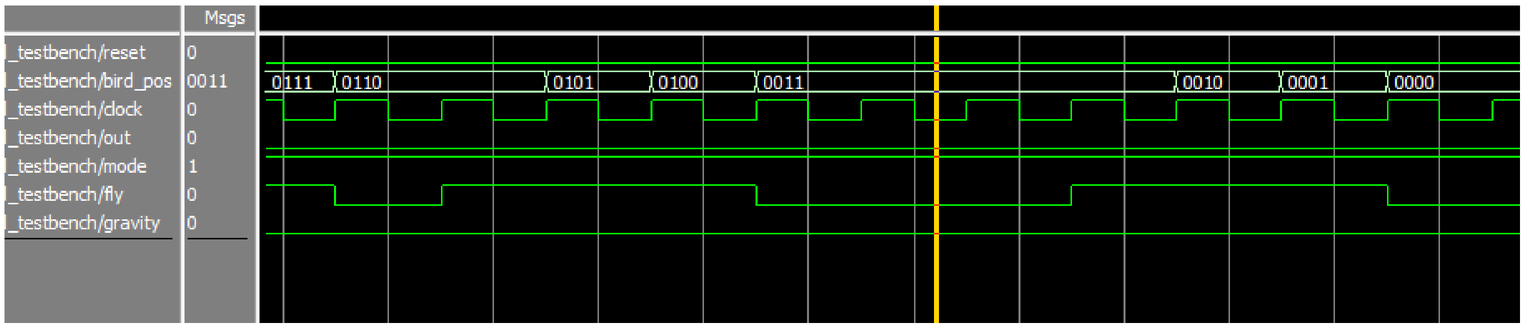
Above, the bird index starts at 7 after reset is turned OFF. Then the bird index increases while gravity = 1 and fly = 0. Then the bird index decreases when fly = 1 and gravity = 0.

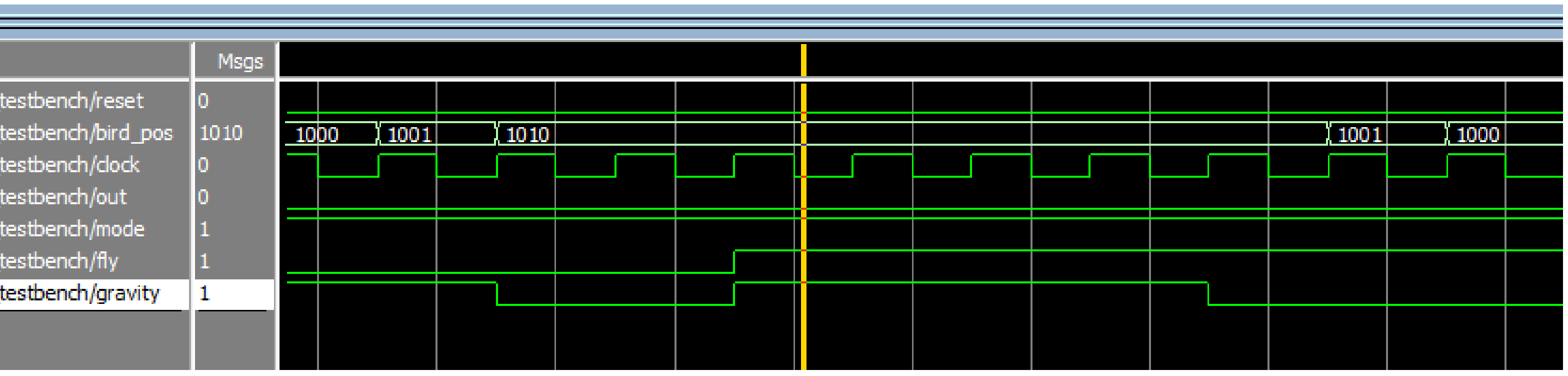


Above, you can see when the bird position is 0 and fly = 1 and gravity = 0, the bird resets to the initial position and the out signal turns ON.



Above you can see when the bird position is 15 and fly = 0 and gravity = 1, the bird resets to the initial position and the out signal turns OFF.





In the screenshots above you can see when both fly and gravity = 0, the bird position does not change.

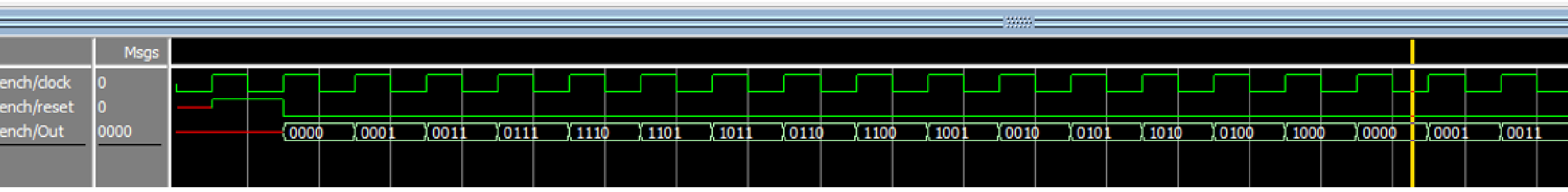
And when both fly and gravity = 1, the bird position does not change.

Generate Pipes Module

This module outputs a [15:0][15:0] array, with the index values of the randomly generated columns set to HIGH and except for the index values of the randomly generated gaps which are set to LOW.

This module generates 14 column placements. The module will cycle through the column placements using a 4bit LFSR. The initial column position is the rightmost column on the LED display. At every positive edge of the clock the column will move one position to the left until it reaches the second to leftmost column. On the next positive edge of the clock, the column will start over in the initial position with a new randomly generated gap size and height. Gap sizes and heights are generated using a 3bit LFSR. Gap size and height can both be any number 0-7.

Each number corresponds to a specific column placement. The program sets each index in the 2d array accordingly and outputs the array to the LED Display module where the appropriate green LEDs are turned on.



Above is a screenshot of the 4 bit LFSR that is used to move the pipes to the left on every positive edge of the clock.

The LFSR state cycle generates the following numbers is the following order:

0, 1, 3, 7, 14, 13, 11, 6, 12, 9, 2, 5, 10, 4, 3.

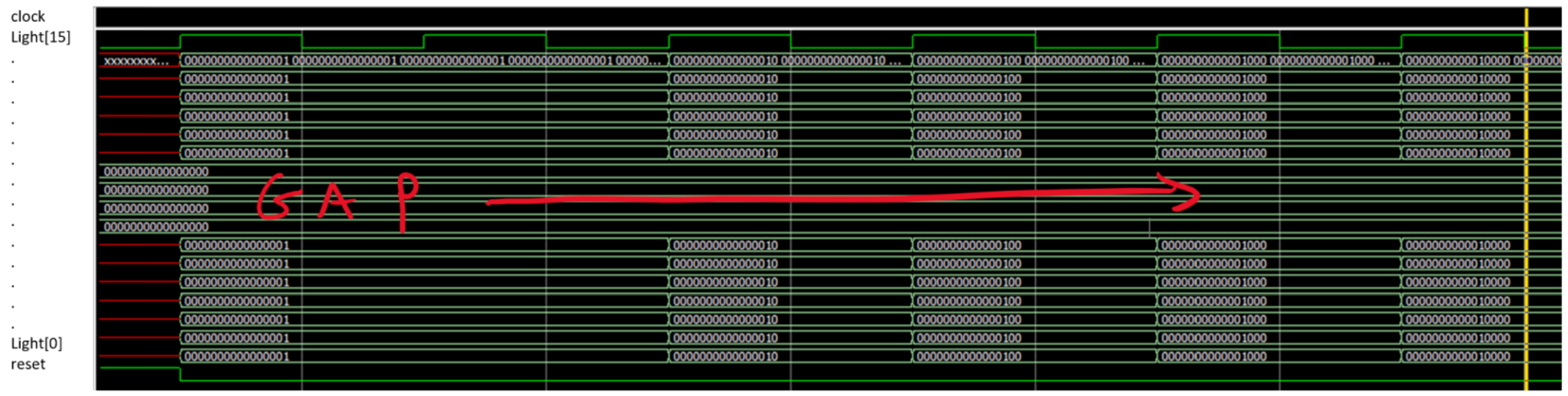
When 0 is generated, every index of the 2D light array is set to 16'b0000000000000001. This is the initial pipe position. This will generate a green pipe on the right most column of the LED Display.

When 1 is generated, every index is set to 16'b0000000000000010.

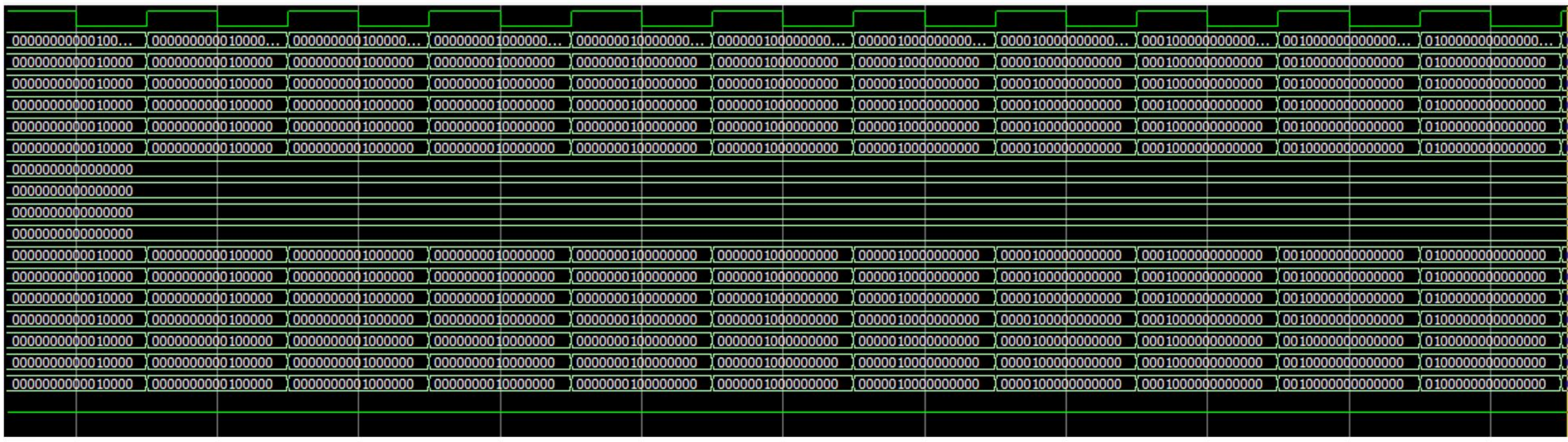
When 3 is generated, every index is set to 16'b0000000000000100.

When 7 is generated, every index is set to 16'b0000000000001000.

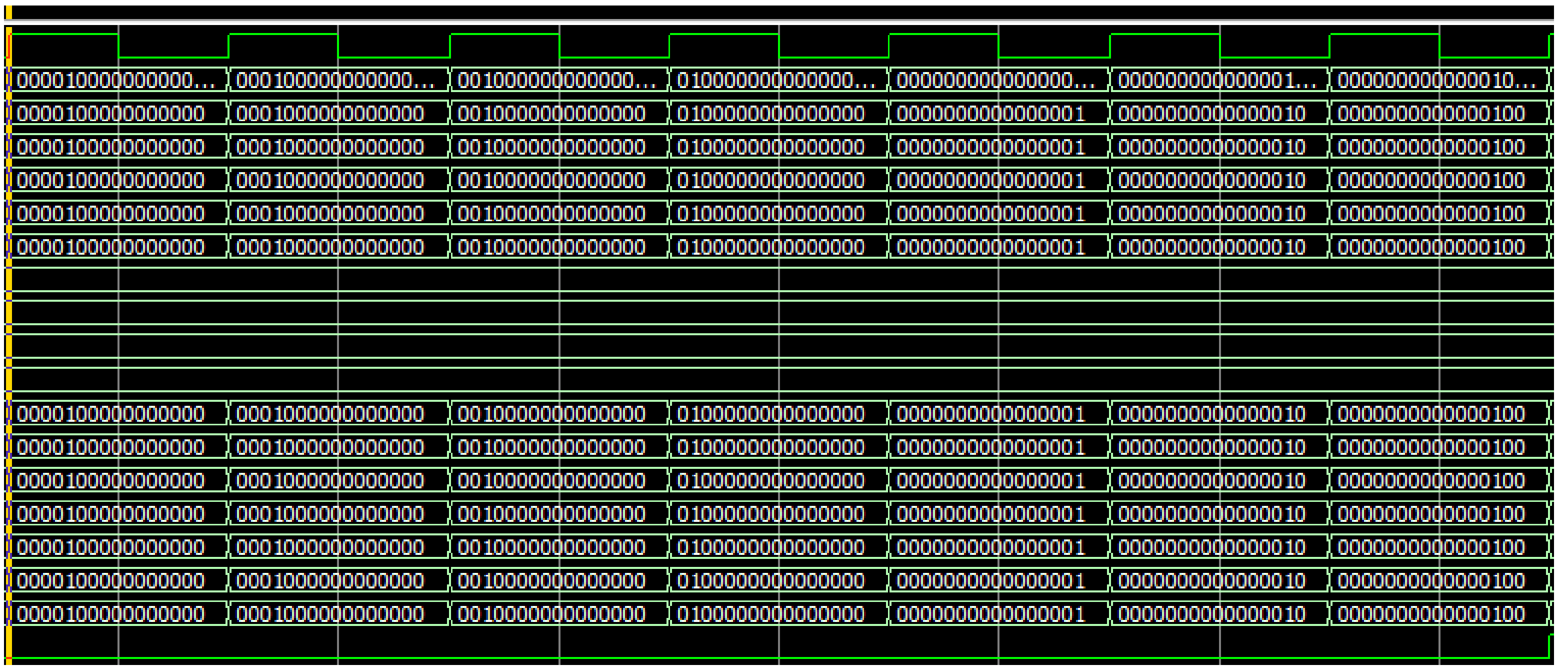
This pattern continues for all 15 numbers.



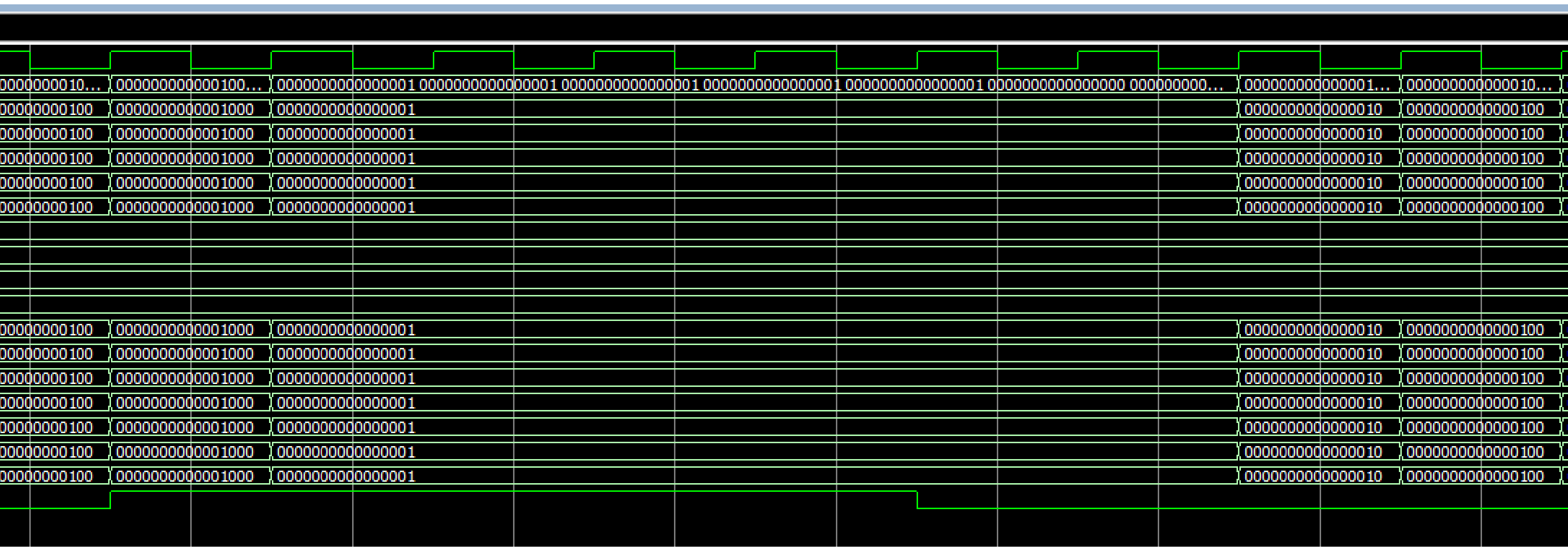
When reset is set to LOW we can see the pipe is at the right most position. At every positive edge of the clock, the pipe moves one position to the left. The gap size and gap position stays the same as the pipe moves across the LED display from right to left. Once we reach the initial position again, a new gap size and position will be generated.



Here you can see, the pipe continues to move one position to the left at every positive edge of the clock. Once the clock reaches the right most position, it will move back to the right most position.



When reset is set to HIGH, the pipe will move back to the rightmost position. The pipe will stay in this initial position until reset is set to LOW again, the pipe will then continue moving one position to the left on every positive edge of the clock.



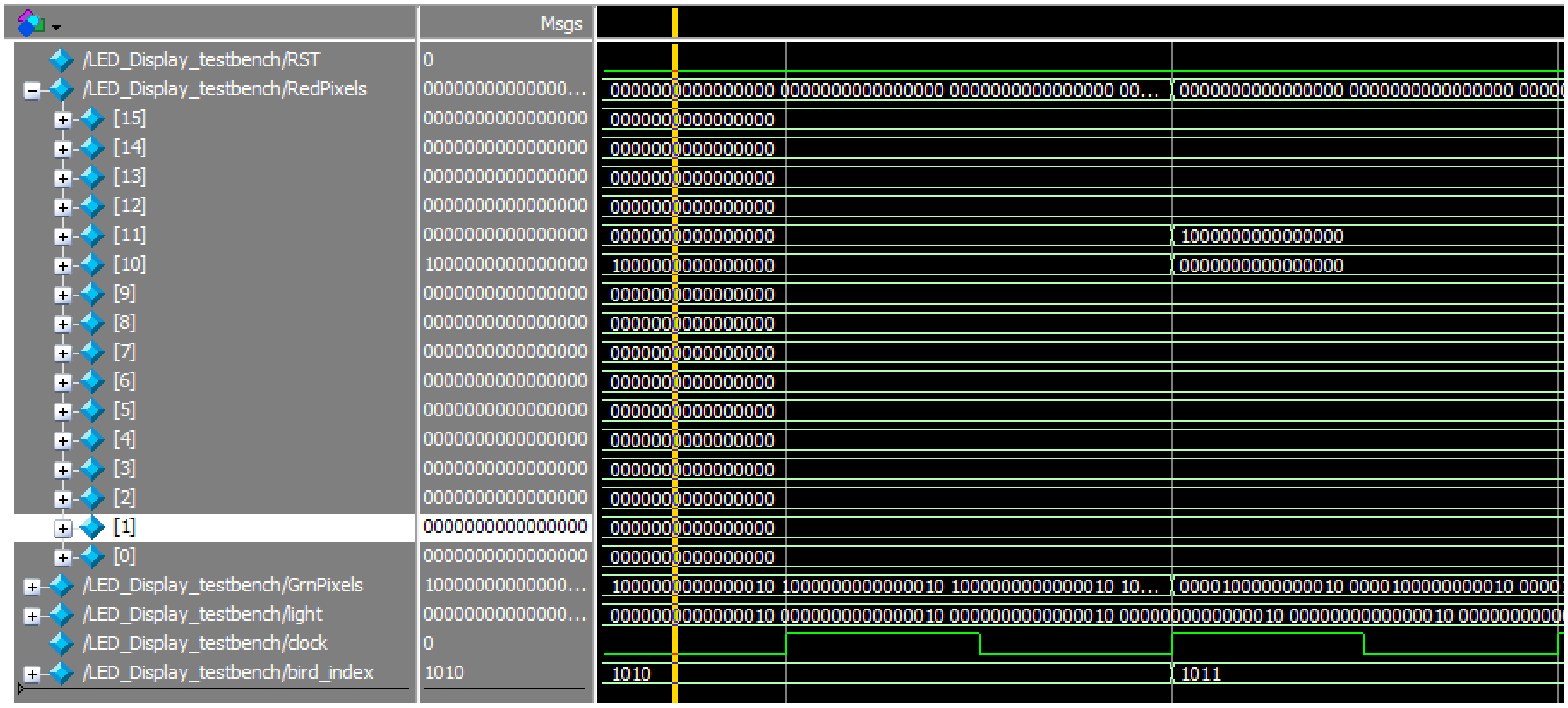
```
0000000010…  000000000000100…  0000000000000001 0000000000000001 0000000000000001 0000000000000001 0000000000000001 0000000000000000 000000000…  000000000000001…  000000000000010…
00000000100  0000000000001000  0000000000000001                                                                                                0000000000000010  0000000000000100
00000000100  0000000000001000  0000000000000001                                                                                                0000000000000010  0000000000000100
00000000100  0000000000001000  0000000000000001                                                                                                0000000000000010  0000000000000100
00000000100  0000000000001000  0000000000000001                                                                                                0000000000000010  0000000000000100
00000000100  0000000000001000  0000000000000001                                                                                                0000000000000010  0000000000000100
00000000100  0000000000001000  0000000000000001                                                                                                0000000000000010  0000000000000100
00000000100  0000000000001000  0000000000000001                                                                                                0000000000000010  0000000000000100
00000000100  0000000000001000  0000000000000001                                                                                                0000000000000010  0000000000000100
00000000100  0000000000001000  0000000000000001                                                                                                0000000000000010  0000000000000100
00000000100  0000000000001000  0000000000000001                                                                                                0000000000000010  0000000000000100
00000000100  0000000000001000  0000000000000001                                                                                                0000000000000010  0000000000000100
00000000100  0000000000001000  0000000000000001                                                                                                0000000000000010  0000000000000100
```

LED Display Module

This module takes in the 4bit bird index and 1-3 [15:0][15:0] light arrays from each instance of the generate pipes module.

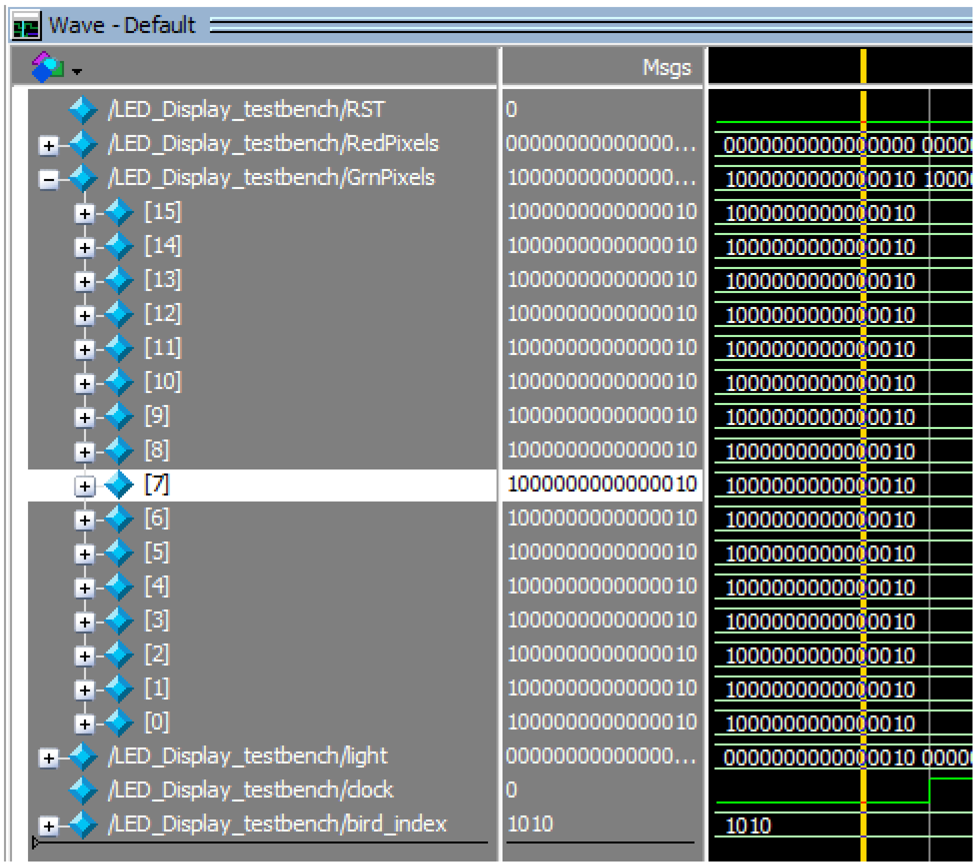
This module turns the Red LED at the bird index ON and sets all other red LEDS OFF.

It also turns the Green LED's in the generated columns ON except for their gaps which will be set OFF. All other green LED's are set OFF.



```
                                     Msgs
/LED_Display_testbench/RST           0
/LED_Display_testbench/RedPixels     00000000000000...   0000000000000000 0000000000000000 0000000000000000 00...  0000000000000000 0000000000000000 0000
    [15]                             0000000000000000    0000000000000000
    [14]                             0000000000000000    0000000000000000
    [13]                             0000000000000000    0000000000000000
    [12]                             0000000000000000    0000000000000000
    [11]                             0000000000000000    0000000000000000                                         1000000000000000
    [10]                             1000000000000000    1000000000000000                                         0000000000000000
    [9]                              0000000000000000    0000000000000000
    [8]                              0000000000000000    0000000000000000
    [7]                              0000000000000000    0000000000000000
    [6]                              0000000000000000    0000000000000000
    [5]                              0000000000000000    0000000000000000
    [4]                              0000000000000000    0000000000000000
    [3]                              0000000000000000    0000000000000000
    [2]                              0000000000000000    0000000000000000
    [1]                              0000000000000000    0000000000000000
    [0]                              0000000000000000    0000000000000000
/LED_Display_testbench/GrnPixels     10000000000000...   1000000000000010 1000000000000010 1000000000000010 10...  0000100000000010 0000100000000010 0000
/LED_Display_testbench/light         00000000000000...   0000000000000010 0000000000000010 0000000000000010 0000000000000010 0000000000000010 0000000000
/LED_Display_testbench/clock         0
/LED_Display_testbench/bird_index    1010                1010                                                     1011
```

When the bird index is 1010 (10), the red LED at index 10 is ON. When bird is 1011 (11), the red LED at index 11 is ON.



```
Wave - Default
                                              Msgs
/LED_Display_testbench/RST                    0
/LED_Display_testbench/RedPixels              00000000000000...   0000000000000000 0000
/LED_Display_testbench/GrnPixels              10000000000000...   1000000000000010 1000
    [15]                                      1000000000000010    1000000000000010
    [14]                                      1000000000000010    1000000000000010
    [13]                                      1000000000000010    1000000000000010
    [12]                                      1000000000000010    1000000000000010
    [11]                                      1000000000000010    1000000000000010
    [10]                                      1000000000000010    1000000000000010
    [9]                                       1000000000000010    1000000000000010
    [8]                                       1000000000000010    1000000000000010
    [7]                                       1000000000000010    1000000000000010
    [6]                                       1000000000000010    1000000000000010
    [5]                                       1000000000000010    1000000000000010
    [4]                                       1000000000000010    1000000000000010
    [3]                                       1000000000000010    1000000000000010
    [2]                                       1000000000000010    1000000000000010
    [1]                                       1000000000000010    1000000000000010
    [0]                                       1000000000000010    1000000000000010
/LED_Display_testbench/light                  00000000000000...   0000000000000010 0000
/LED_Display_testbench/clock                  0
/LED_Display_testbench/bird_index             1010                1010
```

Here you can see the green LEDS are ON, to represent two pipes. The generate pipe module controls when the pipes move and reset.

Game Status Module

This module takes the bird position from the bird location module as well as a 4 bit number representing the position of each pipe.

This module checks if a pipe has reached the leftmost position (where the bird hovers). When a pipe is in this position, the module will check if the bird index is within the pipe's gap. If it is, the pipe has been cleared and the increment signal is set to HIGH. If the bird index does not fall within the gap, the bird has collided with the pipe and the game over signal is set to HIGH.

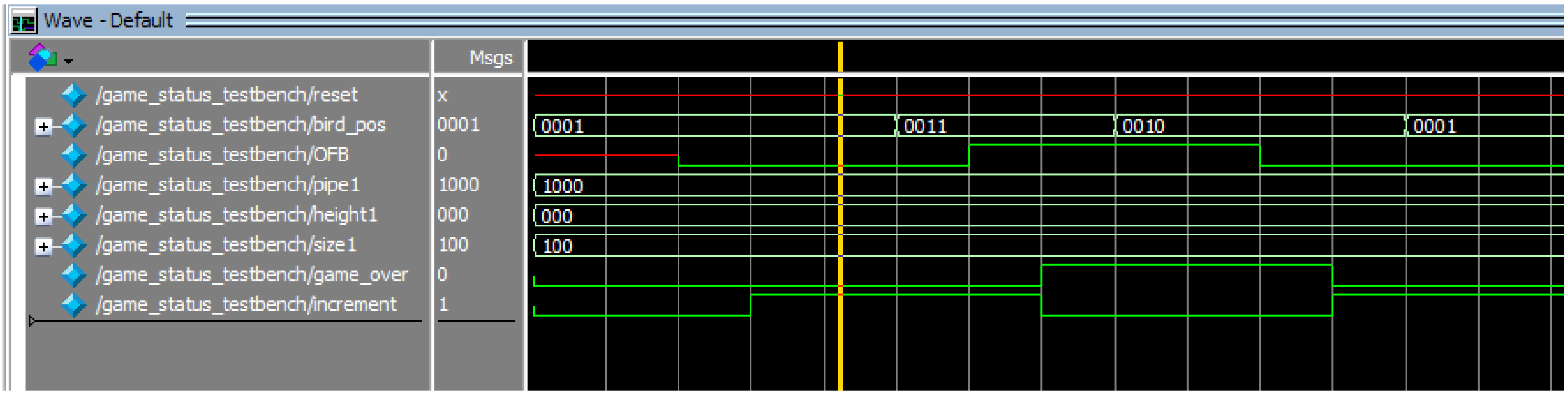
This module also takes the bird out of bounds signal from the bird location module. When this signal is HIGH, game over is set to HIGH.

In the screenshot below, pipe1 is in position 8, for pipe1, this represents the leftmost position.

Here we have height of 0 and a size 4.

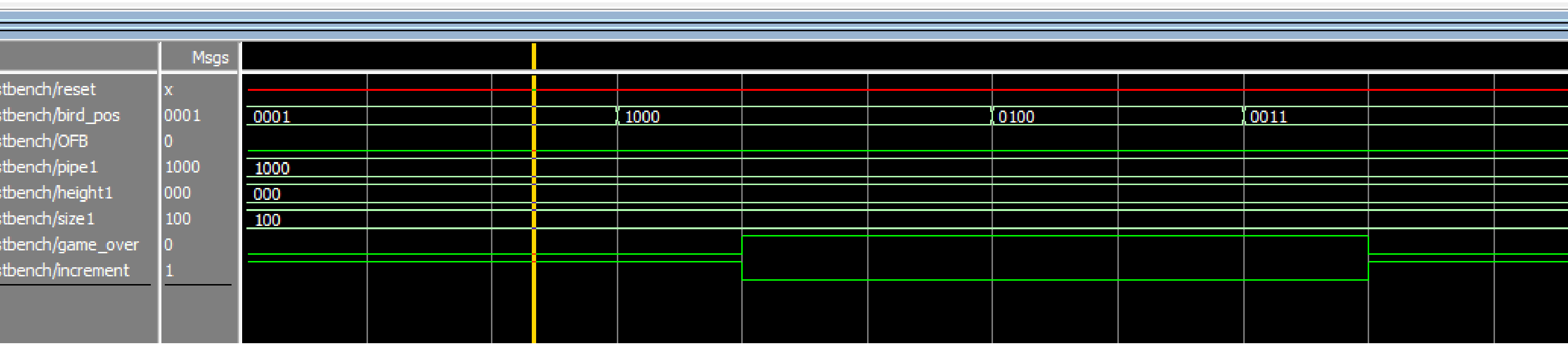
You can see when the bird_pos is index 1 and OFB = 0, game over = FALSE and increment = TRUE, since the bird is within the gap.

You can also see when OFB is set to HIGH, game over goes HIGH until OFB is LOW.



```
/game_status_testbench/reset        x       
/game_status_testbench/bird_pos     0001    0001   0011   0010   0001
/game_status_testbench/OFB          0
/game_status_testbench/pipe1        1000    1000
/game_status_testbench/height1      000     000
/game_status_testbench/size1        100     100
/game_status_testbench/game_over    0
/game_status_testbench/increment    1
```

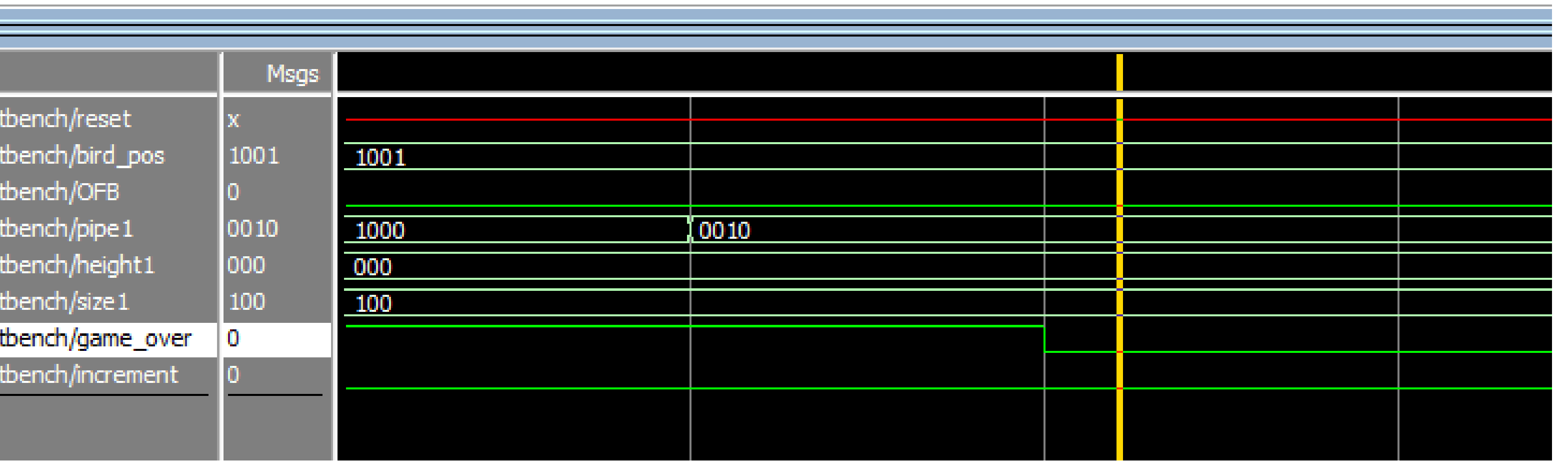
When the bird index = 1, game over = 0 and increment = 1. Then bird index is set to 8, since this is not within the gap and pipe1 is in the leftmost position, game over turns ON and increment turns OFF. Once the bird position is within the gap again (bird_pos = 4'b0011) game over turns OFF and increment turns ON.



```
tbench/reset        x
tbench/bird_pos     0001    0001   1000   0100   0011
tbench/OFB          0
tbench/pipe1        1000    1000
tbench/height1      000     000
tbench/size1        100     100
tbench/game_over    0
tbench/increment    1
```

When the pipe is not in the leftmost position, game over and increment will remain OFF, no matter the bird position. Game over will turn ON, if and only if OFB is ON.

In this screenshot, you can see game over turns off when the pipe position changes from 8 (left most position) to 2 (a different positon). Note increment stays OFF.

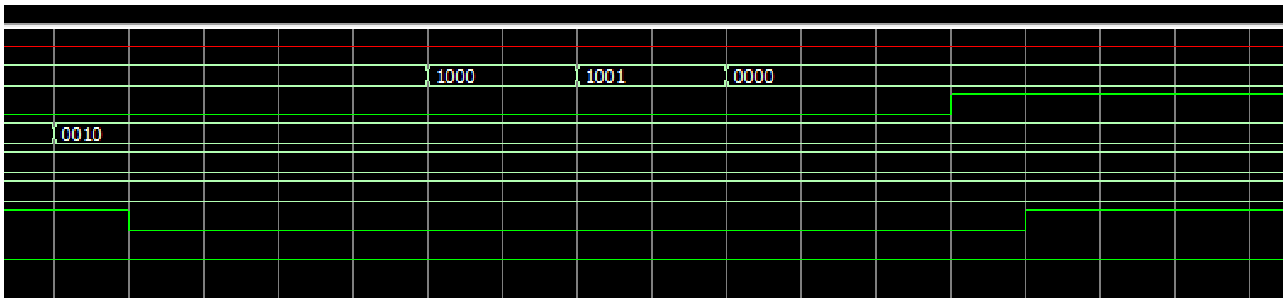


In the screenshot below you can see game over = 1 when OFB = 1.

OFB

Game over

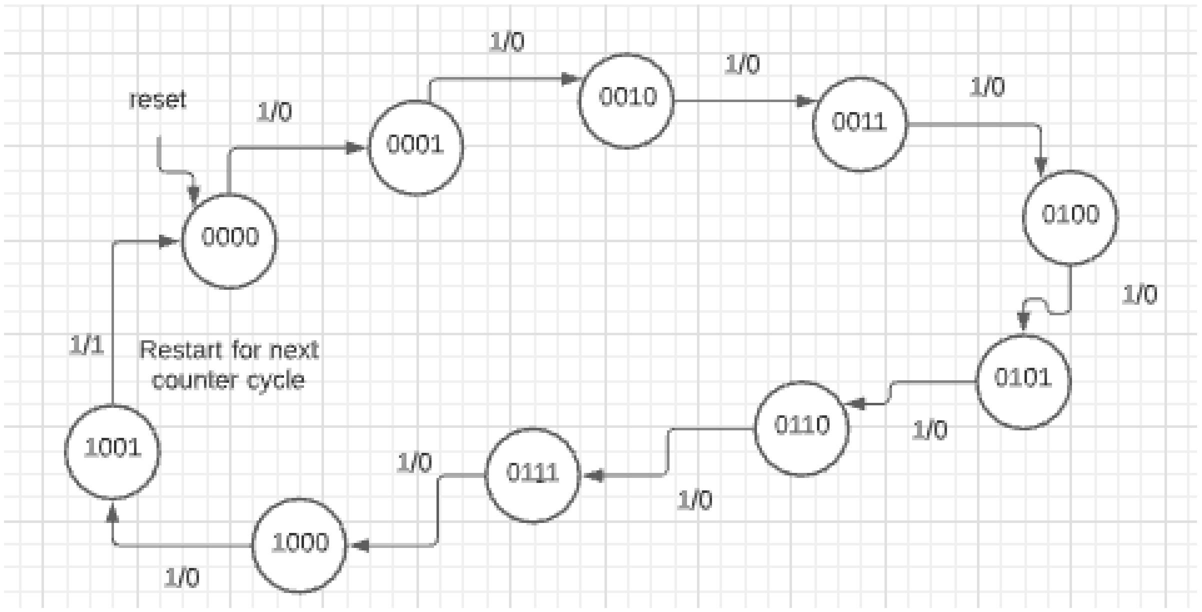
increment



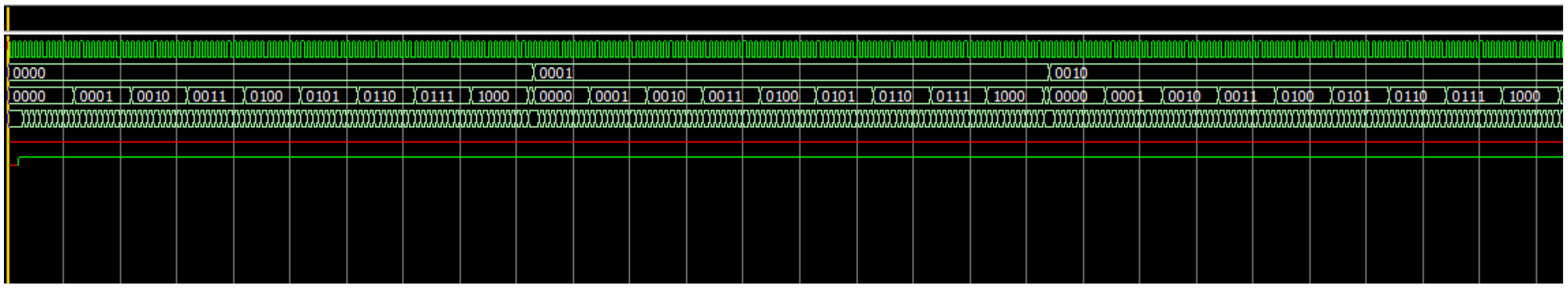
Score

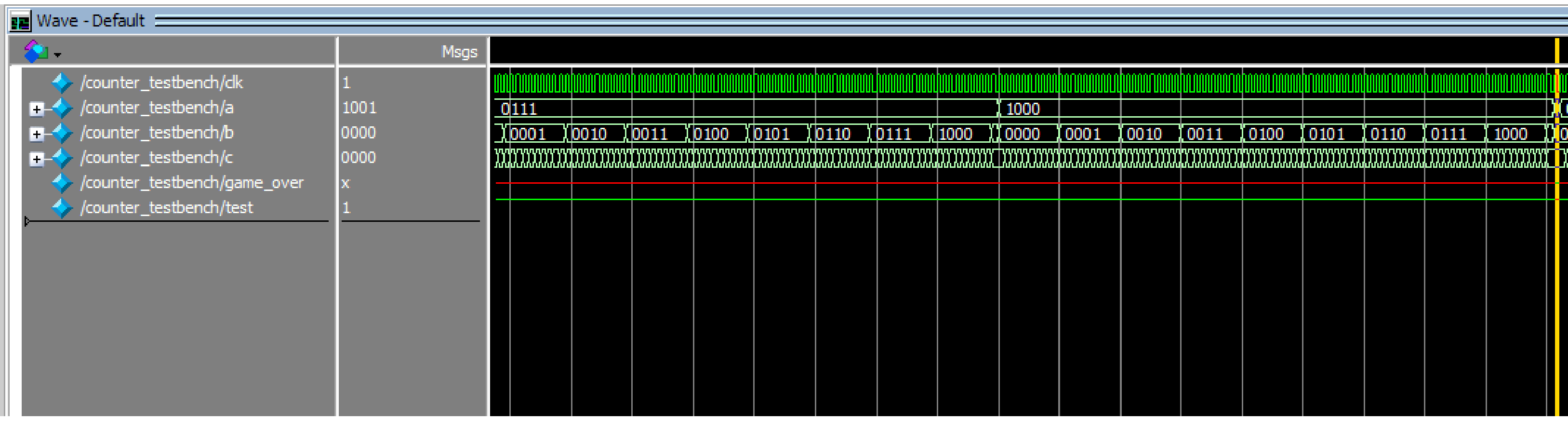
The Score module is a counter that takes the output signals from game status. If increment is HIGH, the score increments. If game status or the Reset SW[9] is HIGH, the score resets.

Score outputs three 4 bit signals which are sent to the HEX display module. Each 4-bit signal is assigned to one of the three HEX displays. The 4bit numbers represent 9 different encoding mapped to patterns to display on the 7seg HEX.



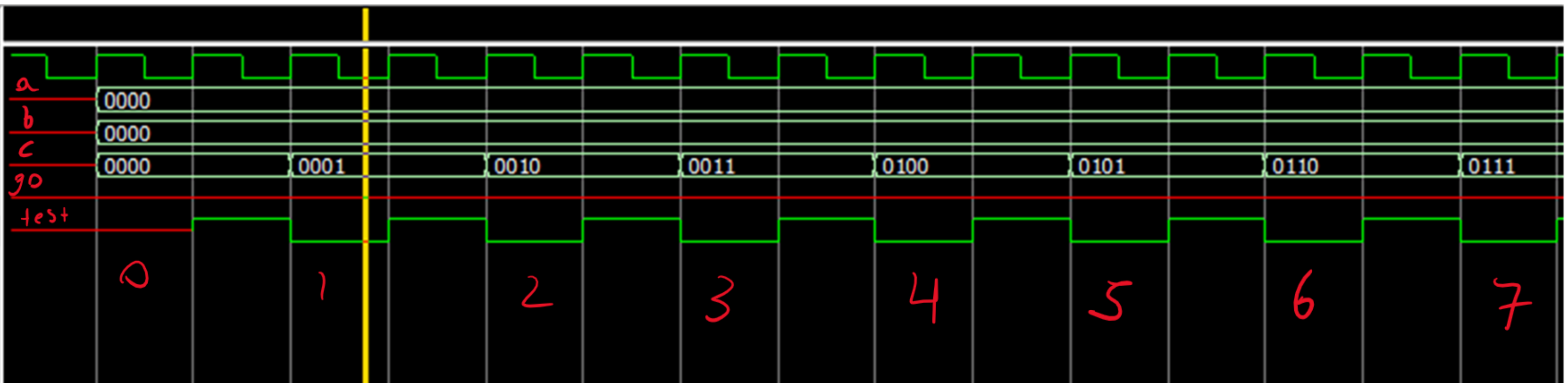
When the increment signal from the game status module is HIGH, the FSM moves to the next state. The FSM counts to 9 and then restarts. The 1 output tells the module to start the count on the next digit.

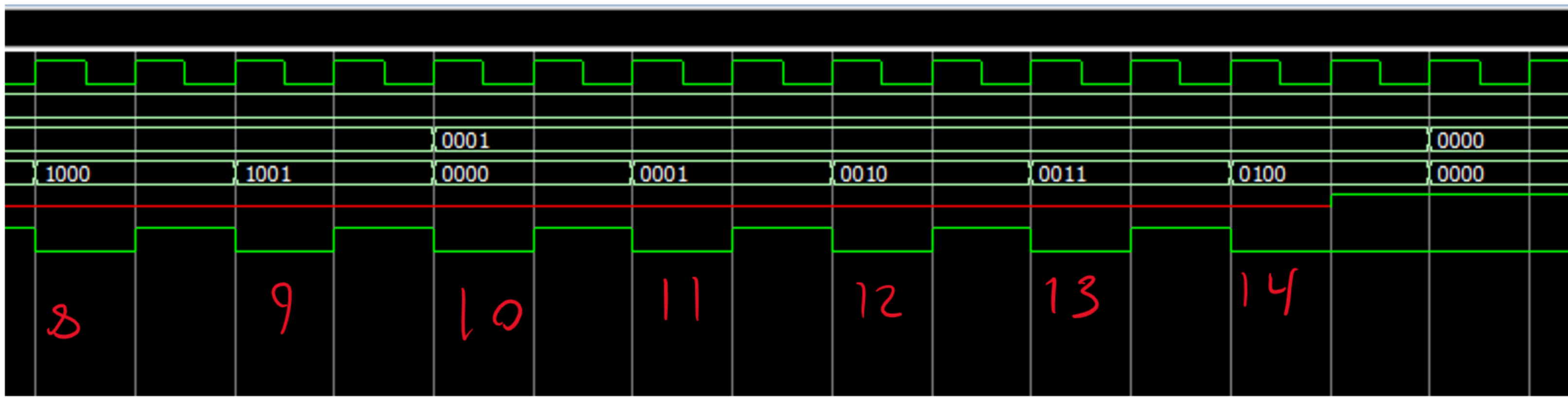




Wave - Default

	Msgs
/counter_testbench/clk	1
/counter_testbench/a	1001
/counter_testbench/b	0000
/counter_testbench/c	0000
/counter_testbench/game_over	x
/counter_testbench/test	1

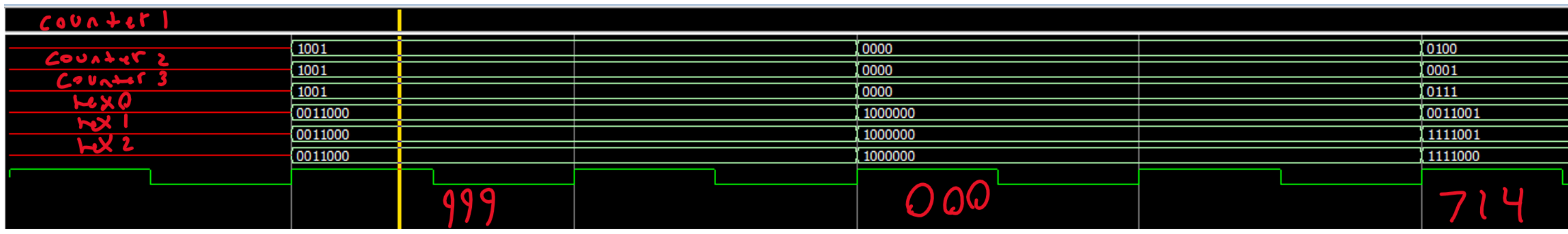




Above you can see the game over signal restarts the count.

Hex Display

This module displays the score as a 3-digit number on three 7 segment HEX Displays. The Hex Display module takes the three 4bit signals sent from the counter and displays the appropriate number on each of the three 7seg hex displays (HEX0, HEX1, HEX2).



To test this system, I started by developing and testing each module separately using ModelSim. Once I had a few modules working, I tested how they worked together by uploading the design onto the FPGA. I continued to add on more modules, each time testing the new module thoroughly on its own in ModelSim, before implementing it in the main DE1-soc file. I continued this way until I had completed the design, and added all the extra features.

Hour count: 40-45