# ACTIVITY 5: SEQUENTIAL (T-BIRD)

Jason Harvey Lorenzo, April 1, 2024
*ENGG 125.03, Department of Electronics, Computer, and Communications Engineering*
*School of Science and Engineering, Ateneo de Manila University*
Quezon City, 1108, Philippines
jason.lorenzo@obf.ateneo.edu

***Abstract*—The project involves the emulation of the T-bird tail light pattern using Verilog HDL. The HDL code implementation, depicted in Listing 1, utilizes two 4-bit registers (LEFT and RIGHT) to control the left and right tail lights, respectively. The module operates on each positive edge of the clock signal (CLK), with the LEFT and RIGHT registers being updated based on the states of SW1 and SW2, respectively. The TLIGHT output signal concatenates the LEFT and RIGHT registers to represent the overall state of the tail lights. Simulations conducted using vector waveform analysis in Quartus validate the functionality of the designed module. Additionally, a testbench (tbird_tb) is developed to verify the module's functionality under various test cases, further confirming its adherence to the specified behavior.**

***Index Terms*—Verilog HDL, T-bird tail lights pattern**

## I. Introduction

This activity involves emulating the T-bird tail light pattern in Verilog HDL. The pattern is summarized in Table I. Note that 1 is on and 0 is off. Switches SW0, SW1, and SW2 are inputs.

## II. HDL Code

Listing 1 shows the HDL code implementation of the function defined above.

Listing 1. 74XX138 Decoder HDL Code

```
module tbird(CLK, SW0, SW1, SW2, TLIGHT);
input CLK, SW0, SW1, SW2;
output wire [7:0] TLIGHT;

reg [3:0] LEFT, RIGHT;

assign TLIGHT = {LEFT[3:0], RIGHT[3:0]};

always @(posedge CLK)
begin
    if (SW1) LEFT <= (LEFT[3]) ?
      4'b0000 : {LEFT[2:0], 1'b1};
    else LEFT <= {SW0, SW0, SW0, SW0};

    if (SW2) RIGHT <= (RIGHT[0]) ?
      4'b0000 : {1'b1, RIGHT[3:1]};
    else RIGHT <= {SW0, SW0, SW0, SW0};
end

endmodule
```

TABLE I
T-bird Tail Light Pattern Table

| Pattern | | | Output |
|---|---|---|---|
| SW0 | SW1 | SW2 | LEDs |
| 0 | 0 | 0 | all 0 |
| 0 | 0 | 1 | 00000000<br>00001000<br>00001100<br>00001110<br>00001111 then back to all 0 |
| 0 | 1 | 0 | 00000000<br>00010000<br>00110000<br>01110000<br>11110000 then back to all 0 |
| 0 | 1 | 1 | 00000000<br>00011000<br>00111100<br>01111110<br>11111111 then back to all 0 |
| 1 | 0 | 0 | all 1 |
| 1 | 0 | 1 | 11110000<br>11111000<br>11111100<br>11111110<br>11111111 then back to 11110000 |
| 1 | 1 | 0 | 00001111<br>00011111<br>00111111<br>01111111<br>11111111 then back to 00001111 |
| 1 | 1 | 1 | 00000000<br>00011000<br>00111100<br>01111110<br>11111111 then back to all 0 |

This code defines the module tbird which controls the tail lights based on the state of three switches (SW0, SW1, SW2). The tail lights are represented by the output signal TLIGHT, which is an 8-bit vector with the left 4 bits (LEFT[3:0]) representing the left tail lights and the right 4 bits (RIGHT[3:0]) representing the right tail lights. The module operates on each positive edge of the clock signal (CLK).

Inside the always block, the code first checks the state of SW1 to determine if the left tail lights need to be modified. If SW1 is high, the LEFT lights are shifted left with 1, it checks if the leftmost bit of LEFT is 1. If it is 1, indicating that the leftmost tail light is already illuminated, the entire LEFT vector is cleared to 0000. Otherwise, it shifts the contents of LEFT one position to the left, with the leftmost bit being set to

1. If SW1 is low, indicating no shift operation, LEFT is set to a 4-bit vector consisting of the value of SW0. This is similar for RIGHT, but using shift right operations and checking the rightmost bit instead.

Finally, the TLIGHT output is assigned by concatenating the LEFT and RIGHT vectors, ensuring that the tail lights are appropriately represented in the output signal.

## III. Output Circuit

### A. *Vector Waveform*

Fig. 1 shows the output circuit of the code above.

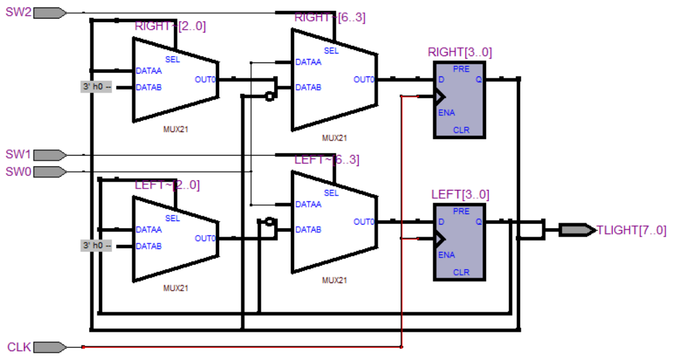

Fig. 1. T-bird Tail Lights Implementation RTL Netlist

The diagram illustrates the functional architecture of the created tbird module. Notice how LEFT and RIGHT tail lights contained in their respective 4-bit registers (bottom and up, respectively) are independently derived from the three input switches using their respective two multiplexers. This solution not only makes the netlist and logic simpler, it also allows for a responsive design when changing input states.

## IV. Simulation

Fig. 2 and 3 shows respective output waveforms of the device given test inputs.

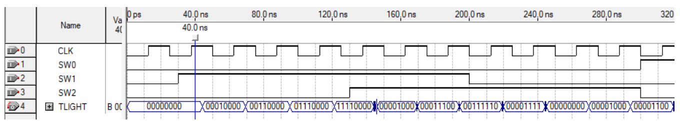

Fig. 2. T-bird Tail Light Implementation Simulation Waveforms

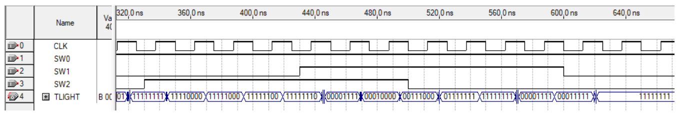

Fig. 3. T-bird Tail Light Implementation Simulation Waveforms Continuation

As depicted in Fig. 2, when all switches (SW0, SW1, and SW2) are in the 000 state, the TLIGHT output remains consistently at 00000000. The cascade effect initiates with the activation of SW1 at approximately 40 ns, prompting the LEFT tail light to begin its sequence. Subsequently, around 138 ns, SW2 activation triggers the cascade for the RIGHT tail light, coinciding with the completion of LEFT's pattern as it resets to 0000. Both SW1 and SW2 remain active until approximately 190 ns, causing their cascades to overlap until the moment SW1 is abruptly deactivated, interrupting its pattern and resulting in an output of 0000. This responsiveness stems from the independent processing of the LEFT and RIGHT tail lights. At approximately 300 ns, as depicted in Fig. **??**, SW0 is activated while the other switches remain off, resulting in an 8-bit output of 11111111. Subsequent observations show similar functionalities upon switching SW1 and SW2 thereafter. These are all consistent with the functionalities identified in Table I.

### A. *Testbench*

Listing 2 shows the created testbench code for the implementation.

Listing 2. 74XX138 Decoder HDL Code

```
‘timescale 1ns / 1ps

module tbird_tb;
  reg CLK, SW0, SW1, SW2;
  wire [7:0] TLIGHT;

  tbird test (
        .CLK(CLK),
        .SW0(SW0),
        .SW1(SW1),
        .SW2(SW2),
        .TLIGHT(TLIGHT)
    );

  always #(5) CLK = ~CLK;

  initial
    begin
      CLK = 0;
      SW0 = 0;
      SW1 = 0;
      SW2 = 0;
      #100;

      // Test Case 1
      SW1 = 1;
      #50;
      SW2 = 1;
      #50;

      // Test Case 2
      SW1 = 0;
      #50;
      SW2 = 1;
      #50;

      // Test Case 3
      SW0 = 1;
```

```
        #50;
        SW1 = 1;
        #50;

        // Test Case 4
        SW2 = 0;
        #50;
        SW1 = 0;
        #50;

        $finish;
    end
endmodule
```

This testbench, tbird_tb, is designed to verify the functionality of the tbird module. It instantiates the tbird module and connects its ports to the testbench inputs and outputs. The testbench generates clock pulses, toggling the CLK signal every 5 time units. It then initializes the input signals (SW0, SW1, SW2) and observes the output signal TLIGHT at different test cases.

The testbench begins by setting all input signals to 0 and simulating for 100 time units to stabilize the system. It then proceeds to execute four test cases, each lasting 50 time units. In each test case, it manipulates the input signals (SW0, SW1, SW2) to simulate different scenarios, such as activating or deactivating switches. After executing all test cases, the simulation is terminated using the $finish system task.

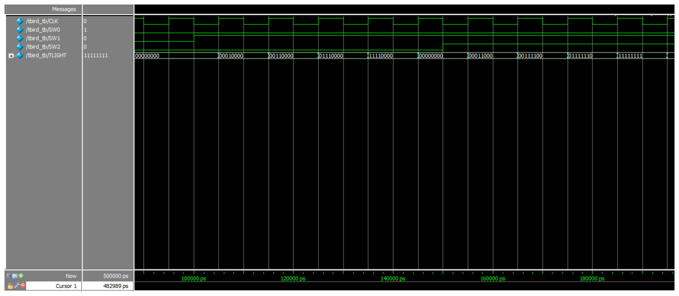

Fig. 4. T-bird Tail Light Implementation Testbench Waveforms

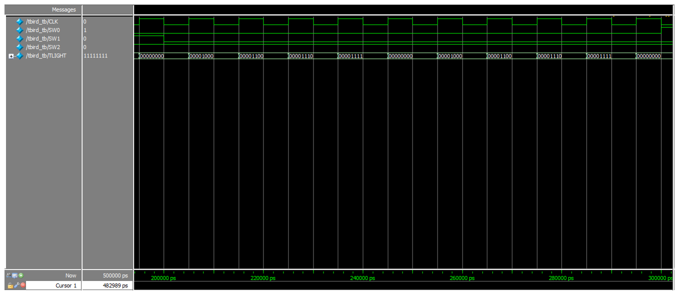

Fig. 5. T-bird Tail Light Implementation Testbench Waveforms Cont. 1

The observed outputs corresponding to the test inputs align with the expected functionalities outlined in Table I and corroborated by the simulations conducted using vector waveform

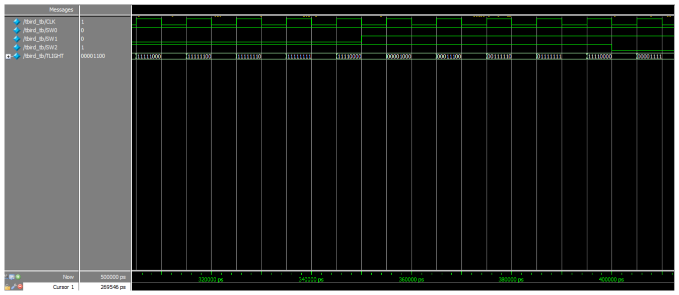

Fig. 6. T-bird Tail Light Implementation Testbench Waveforms Cont. 2

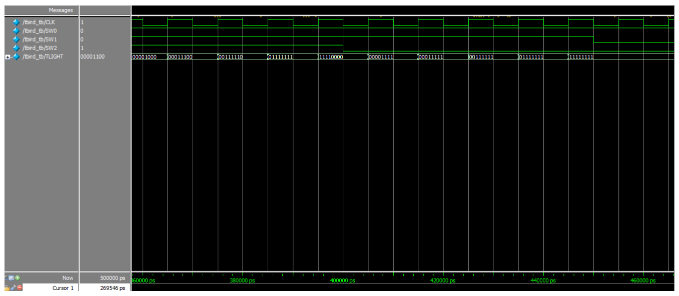

Fig. 7. T-bird Tail Light Implementation Testbench Waveforms Cont. 3

analysis in Quartus. This consistency further validates that the designed T-bird module operates according to its intended specifications.