Brodric Young

ECEN 340

Lab #4

Verilog Simulation, 7-Segment Display

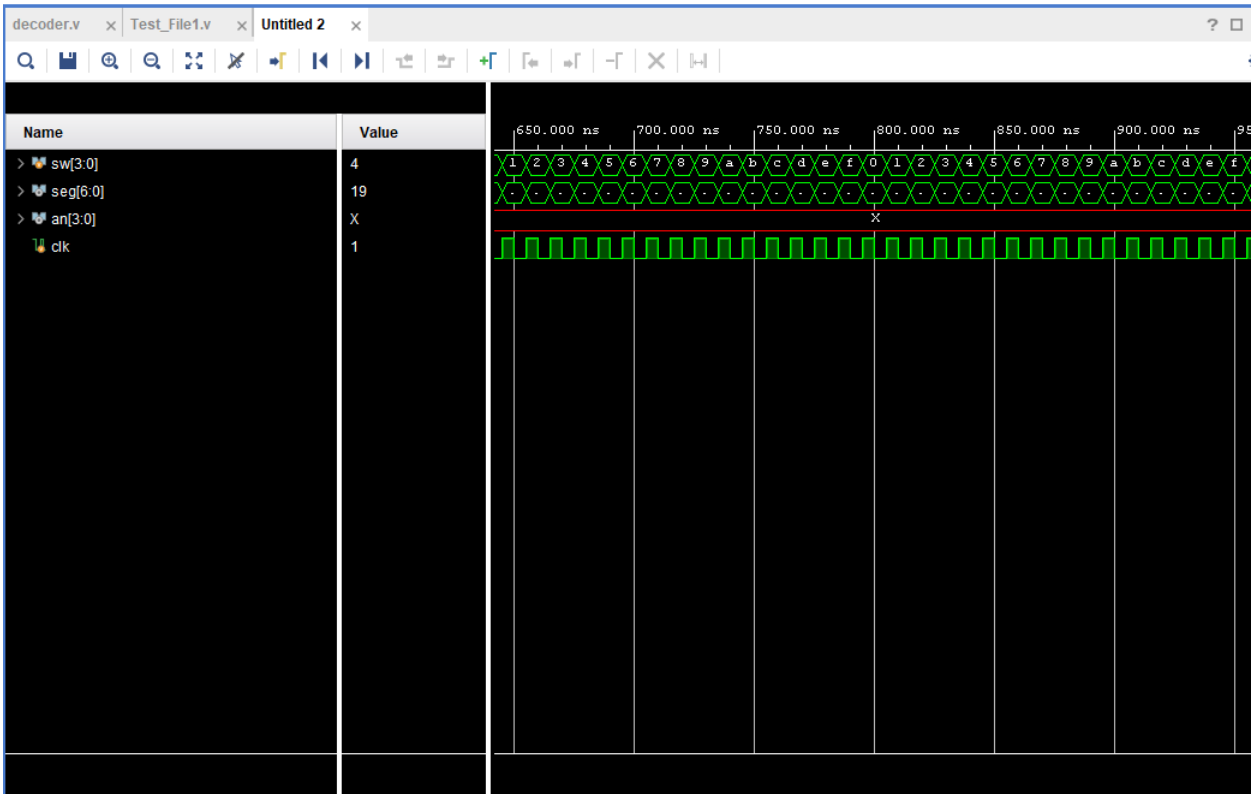
Purpose of the lab:

1. To learn how to interface with a 7-segment display
2. To learn how to simulate by "forcing" signals
3. To learn how to simulate with a test bench

In the first part of this lab, we designed a circuit in Verilog to implement a decoder circuit on the FPGA to display the first 16 digits in hexadecimal on one of the 7-segment displays given the value in binary from the switches input. The code we came up with is here as well as the test bench code:

```
`timescale 1ns / 1ps
// decode switch inputs to display on 7 segment display
module decoder(
   input [3:0] sw,         // 4-bit input for digit
   output reg [6:0] seg,    // 7-segment display output
   output dp, [3:0] an             // Decimal point (set to always off) and Anode control
   );

   assign dp = 1'b1;        // Decimal point off
   assign an = 4'b1110;      // Display the first digit (assuming 4 digits, and enabling the first one)

   always @ (sw)
      case (sw)
         4'b0000: seg = 7'b1000000;  // 0
         4'b0001: seg = 7'b1111001;  // 1
         4'b0010: seg = 7'b0100100;  // 2
         4'b0011: seg = 7'b0110000;  // 3
         4'b0100: seg = 7'b0011001;  // 4
         4'b0101: seg = 7'b0010010;  // 5
         4'b0110: seg = 7'b0000010;  // 6
         4'b0111: seg = 7'b1111000;  // 7
         4'b1000: seg = 7'b0000000;  // 8
         4'b1001: seg = 7'b0011000;  // 9
         4'b1010: seg = 7'b0001000;  // A
         4'b1011: seg = 7'b0000011;  // b
```

```
        4'b1100: seg = 7'b1000110;  // C
        4'b1101: seg = 7'b0100001;  // d
        4'b1110: seg = 7'b0000110;  // E
        4'b1111: seg = 7'b0001110;  // F
        default: seg = 7'b1111110;  // lol
    endcase
endmodule



`timescale 1ns / 1ps
// decoder test bench module
module test_bench_decoder(
  );

  reg  [3:0] sw;
  wire [6:0] seg;
  wire [3:0] an = 4'b1110;
  reg clk = 1'b0;

  //clock and decoder called
  always #5 clk = ~clk;
  decoder U1 (sw, seg, an);

  initial
  begin
  sw = 4'b0000;
  #1000 $finish;
  end

  always @ (posedge clk)  //Addes 1 to sw every positive rise of the clock
    begin
    sw = sw + 1;
    end
endmodule
```



Conclusion:

After writing the code, we tested it by programming the board and adjusting the switches and we also tested it through simulations. The first simulation we did was by forcing constants and verifying the result was correct. Then we used the test bench code to simulate every combination of switches and verify each of their outputs, which can be seen in the image above. In the end our design was 100% functional, worked as we wanted it too, and passed all our tests through simulation and through physical testing on the FPGA.