ECE 3300L.01 - Lab 4

Switch-to-7-Segment Display Interface on Nexys A7

Professor Mohamed Aly
Group Q: Kevin Tang(015429622) &
Jared Mocling(015215057)

July 11th, 2025

Objective:

Binary ↔ Hexadecimal ↔ Decimal Conversion:
Translate 4-bit switch inputs (0–15) into proper segment patterns for display.

Verilog-HDL Design:
Implement structural Verilog modules including decoders, multiplexers, and verify via simulation.

Peripheral Integration:
Interface slide switches, LEDs, and the 8-digit 7-segment display on the

Nexys A7 FPGA

Code:

```
module seg7_driver(
   input clk,
   input rst_n,
   input [15:0] SW,
   output reg [6:0] Cnode,
   output dp,
   output [7:0] AN,
   output [15:0] LED
   );

   assign LED = SW[15:0];
   reg [19:0] tmp;
   reg [3:0] digit;

   assign dp = 1'b1;

   always@(digit)
      case(digit)//setting the 7-segment display to display the corresponding number it
receives
         4'd0: Cnode=7'b0000001; 4'd1: Cnode=7'b1001111; 4'd2: Cnode=7'b0010010;
         4'd3: Cnode=7'b0000110; 4'd4: Cnode=7'b1001100; 4'd5: Cnode=7'b0100100;
         4'd6: Cnode=7'b0100000; 4'd7: Cnode=7'b0001111; 4'd8: Cnode=7'b0000000;
         4'd9: Cnode=7'b0001100; 4'd10:Cnode=7'b0001000;4'd11:Cnode=7'b1100000;
         4'd12:Cnode=7'b0110001;4'd13:Cnode=7'b1000010;4'd14:Cnode=7'b0110000;
         4'd15:Cnode=7'b0111000;default: Cnode=7'b1111111;
      endcase

   always@(posedge clk or negedge rst_n)
```

```
    if(!rst_n) tmp<=0;
    else tmp<=tmp+1;

  wire [2:0] s = tmp[19:17];

  always@(s, SW)
    case (s)
      3'd0:digit=SW[3:0];
      3'd1:digit=SW[7:4];
      3'd2:digit=SW[11:8];
      3'd3:digit=SW[15:12];
      default:digit=4'b0000;
    endcase

  reg [7:0] AN_tmp;
  always@(s)
    case(s)
      3'd0:AN_tmp=8'b11111110;3'd1:AN_tmp=8'b11111101;
      3'd2:AN_tmp=8'b11111011;3'd3:AN_tmp=8'b11110111;
      3'd4:AN_tmp=8'b11101111;3'd5:AN_tmp=8'b11011111;
      3'd6:AN_tmp=8'b10111111;3'd7:AN_tmp=8'b01111111;
      default:AN_tmp=8'b11111111;
    endcase

  assign AN=AN_tmp;
endmodule
```

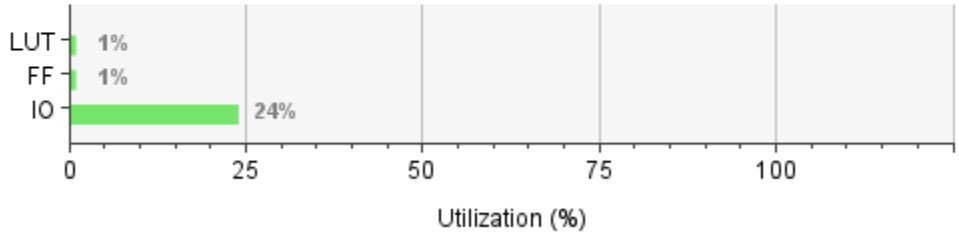
This code is designed to take in the 16 switches as a 16-bit input and divide it into 4 segments. It then decodes the 4 separate 4-bit numbers into hexadecimal, which is then displayed onto the 4 separate 7-segment displays.

```
`timescale 1ns / 1ps

module seg7_driver_tb();

    reg clk_tb, rst_n_tb;
    reg [31:0] SW_tb;
    wire [6:0] Cnode_tb;
    wire dp_tb;
    wire [7:0] AN_tb;

    seg7_driver tb (
        .clk(clk_tb),
        .SW(SW_tb),
        .rst_n(rst_n_tb),
        .Cnode(Cnode_tb),
        .dp(dp_tb),
        .AN(AN)
    );

    always #5 clk_tb = ~clk_tb;
    initial begin
        clk_tb = 0;
        rst_n_tb = 0;
        SW_tb = 32'h00000000;

        #20 rst_n_tb = 1;

        #20 SW_tb = 32'hABCD1234;

        #20
        $stop;
    end

endmodule
```

This testbench allows us to test a 32-bit input and see how the code reads it. It also allows us to test when switches are flipped and see how the outputs are affected.

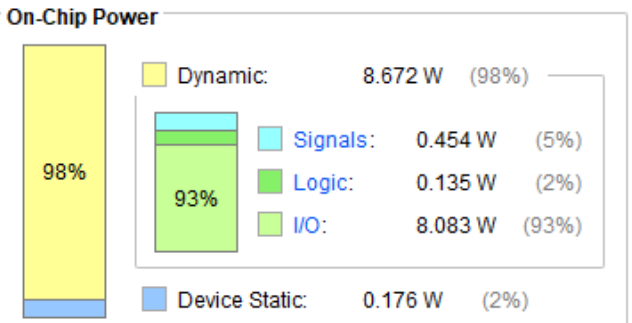
Utilization

Resource	Utilization	Available	Utilization %
LUT	18	63400	0.03
FF	20	126800	0.02
IO	50	210	23.81

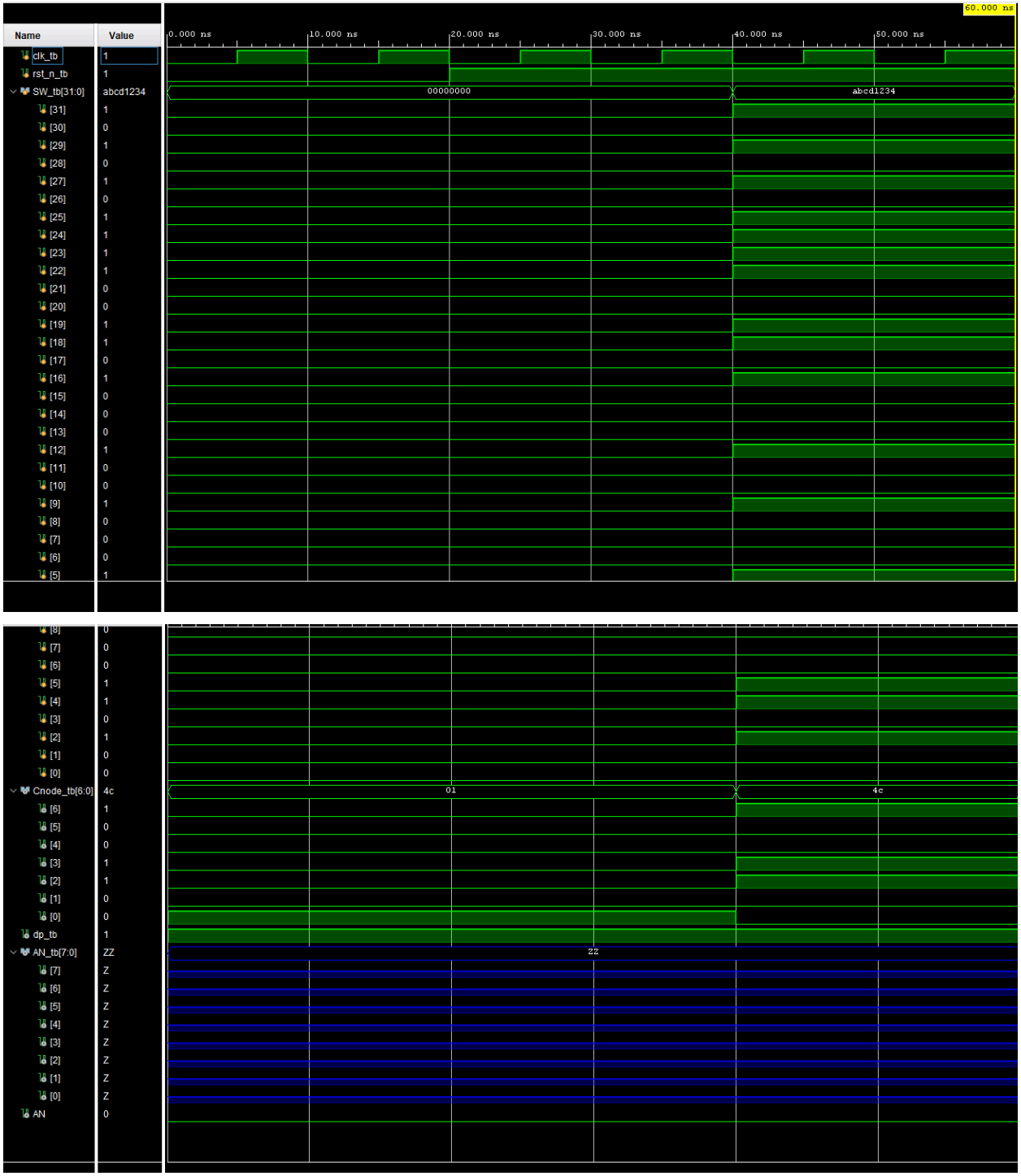


Power analysis from Implemented netlist. Activity derived from constraints files, simulation files or vectorless analysis.

Total On-Chip Power:	**8.848 W**
Design Power Budget:	**Not Specified**
Process:	typical
Power Budget Margin:	**N/A**
Junction Temperature:	**65.4°C**
Thermal Margin:	19.6°C (4.3 W)
Ambient Temperature:	25.0 °C
Effective ϑJA:	4.6°C/W
Power supplied to off-chip devices:	0 W
Confidence level:	Low



Simulation Table



The waveform correctly displays the 32-bit number we chose to put it, as well as the second 32-bit number that was imputed 40ms later.

Contributions:

Kevin Tang (50%) - board demo, compiled code, report.

Jared Mocling (50%)- compiled code, simulation code, Synthesis reports, report