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

3300L

Dia Agrawal and Robert Lainez Torres

Code And Explanation

The driver code given is used to define the logic for the 7 segment display. The code goes through all 8 states sequentially with a 2^17 counter which is 10ms per all 8 switches which looks almost stagnant to the human eye. Of 32 inputs, each 4 bit input corresponds to a 7 segment display which is connected to our AN and Cnode variables.

```
module seg7_driver(
   input clk,
   input rst_n,
   input[31:0] sw,
   output reg [6:0] Cnode,
   output dp,
   output [7:0] AN );
   reg [19:0] tmp;
   reg[3:0] digit;

   assign dp =1'b1;

   always@(digit)
     case(digit)
        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];
        3'd4:digit=sw[19:16]; 3'd5:digit=sw[23:20];3'd6:digit=sw[27:24]; 3'd7:digit=sw[31:28];
        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
```

The next set of code shows the constraint file which declares the switches which control the 8 displays, the leds that light up to the corresponding switches and, the connected 7 segment display power (cnode) and specific instruction (AN). The button rst_n is the reset button.

```
## Clock signal
set_property -dict { PACKAGE_PIN E3    IOSTANDARD LVCMOS33 } [get_ports { clk }]; #IO_L12P_T1_MRCC_35 Sch=clk100mhz
#create_clock -add -name sys_clk_pin -period 10.00 -waveform {0 5} [get_ports {CLK100MHZ}];

##Switches
set_property -dict { PACKAGE_PIN J15   IOSTANDARD LVCMOS33 } [get_ports { sw[0] }]; #IO_L24N_T3_RS0_15 Sch=sw[0]
set_property -dict { PACKAGE_PIN L16   IOSTANDARD LVCMOS33 } [get_ports { sw[1] }]; #IO_L3N_T0_DQS_EMCCLK_14 Sch=sw[1]
set_property -dict { PACKAGE_PIN M13   IOSTANDARD LVCMOS33 } [get_ports { sw[2] }]; #IO_L6N_T0_D08_VREF_14 Sch=sw[2]
set_property -dict { PACKAGE_PIN R15   IOSTANDARD LVCMOS33 } [get_ports { sw[3] }]; #IO_L13N_T2_MRCC_14 Sch=sw[3]
set_property -dict { PACKAGE_PIN R17   IOSTANDARD LVCMOS33 } [get_ports { sw[4] }]; #IO_L12N_T1_MRCC_14 Sch=sw[4]
set_property -dict { PACKAGE_PIN T18   IOSTANDARD LVCMOS33 } [get_ports { sw[5] }]; #IO_L7N_T1_D10_14 Sch=sw[5]
set_property -dict { PACKAGE_PIN U18   IOSTANDARD LVCMOS33 } [get_ports { sw[6] }]; #IO_L17N_T2_A13_D29_14 Sch=sw[6]
set_property -dict { PACKAGE_PIN R13   IOSTANDARD LVCMOS33 } [get_ports { sw[7] }]; #IO_L5N_T0_D07_14 Sch=sw[7]
set_property -dict { PACKAGE_PIN T8    IOSTANDARD LVCMOS18 } [get_ports { sw[8] }]; #IO_L24N_T3_34 Sch=sw[8]
set_property -dict { PACKAGE_PIN U8    IOSTANDARD LVCMOS18 } [get_ports { sw[9] }]; #IO_25_34 Sch=sw[9]
set_property -dict { PACKAGE_PIN R16   IOSTANDARD LVCMOS33 } [get_ports { sw[10] }]; #IO_L15P_T2_DQS_RDWR_B_14 Sch=sw[10]
set_property -dict { PACKAGE_PIN T13   IOSTANDARD LVCMOS33 } [get_ports { sw[11] }]; #IO_L23P_T3_A03_D19_14 Sch=sw[11]
set_property -dict { PACKAGE_PIN H6    IOSTANDARD LVCMOS33 } [get_ports { sw[12] }]; #IO_L24P_T3_35 Sch=sw[12]
set_property -dict { PACKAGE_PIN U12   IOSTANDARD LVCMOS33 } [get_ports { sw[13] }]; #IO_L20P_T3_A08_D24_14 Sch=sw[13]
set_property -dict { PACKAGE_PIN U11   IOSTANDARD LVCMOS33 } [get_ports { sw[14] }]; #IO_L19N_T3_A09_D25_VREF_14 Sch=sw[14]
set_property -dict { PACKAGE_PIN V10   IOSTANDARD LVCMOS33 } [get_ports { sw[15] }]; #IO_L21P_T3_DQS_14 Sch=sw[15]

## LEDs
set_property -dict { PACKAGE_PIN H17   IOSTANDARD LVCMOS33 } [get_ports { led[0] }]; #IO_L18P_T2_A24_15 Sch=led[0]
set_property -dict { PACKAGE_PIN K15   IOSTANDARD LVCMOS33 } [get_ports { led[1] }]; #IO_L24P_T3_RS1_15 Sch=led[1]
set_property -dict { PACKAGE_PIN J13   IOSTANDARD LVCMOS33 } [get_ports { led[2] }]; #IO_L17N_T2_A25_15 Sch=led[2]
set_property -dict { PACKAGE_PIN N14   IOSTANDARD LVCMOS33 } [get_ports { led[3] }]; #IO_L8P_T1_D11_14 Sch=led[3]
set_property -dict { PACKAGE_PIN R18   IOSTANDARD LVCMOS33 } [get_ports { led[4] }]; #IO_L7P_T1_D09_14 Sch=led[4]
set_property -dict { PACKAGE_PIN V17   IOSTANDARD LVCMOS33 } [get_ports { led[5] }]; #IO_L18N_T2_A11_D27_14 Sch=led[5]
set_property -dict { PACKAGE_PIN U17   IOSTANDARD LVCMOS33 } [get_ports { led[6] }]; #IO_L17P_T2_A14_D30_14 Sch=led[6]
set_property -dict { PACKAGE_PIN U16   IOSTANDARD LVCMOS33 } [get_ports { led[7] }]; #IO_L18P_T2_A12_D28_14 Sch=led[7]
set_property -dict { PACKAGE_PIN V16   IOSTANDARD LVCMOS33 } [get_ports { led[8] }]; #IO_L16N_T2_A15_D31_14 Sch=led[8]
set_property -dict { PACKAGE_PIN T15   IOSTANDARD LVCMOS33 } [get_ports { led[9] }]; #IO_L14N_T2_SRCC_14 Sch=led[9]
set_property -dict { PACKAGE_PIN U14   IOSTANDARD LVCMOS33 } [get_ports { led[10] }]; #IO_L22P_T3_A05_D21_14 Sch=led[10]
set_property -dict { PACKAGE_PIN T16   IOSTANDARD LVCMOS33 } [get_ports { led[11] }]; #IO_L15N_T2_DQS_DOUT_CSO_B_14 Sch=led[11]
set_property -dict { PACKAGE_PIN V15   IOSTANDARD LVCMOS33 } [get_ports { led[12] }]; #IO_L16P_T2_CSI_B_14 Sch=led[12]
set_property -dict { PACKAGE_PIN V14   IOSTANDARD LVCMOS33 } [get_ports { led[13] }]; #IO_L22N_T3_A04_D20_14 Sch=led[13]
set_property -dict { PACKAGE_PIN V12   IOSTANDARD LVCMOS33 } [get_ports { led[14] }]; #IO_L20N_T3_A07_D23_14 Sch=led[14]
set_property -dict { PACKAGE_PIN V11   IOSTANDARD LVCMOS33 } [get_ports { led[15] }]; #IO_L21N_T3_DQS_A06_D22_14 Sch=led[15]

##7 segment display
set_property -dict { PACKAGE_PIN T10   IOSTANDARD LVCMOS33 } [get_ports { Cnode[6] }]; #IO_L24N_T3_A00_D16_14 Sch=ca
set_property -dict { PACKAGE_PIN R10   IOSTANDARD LVCMOS33 } [get_ports { Cnode[5] }]; #IO_25_14 Sch=cb
set_property -dict { PACKAGE_PIN K16   IOSTANDARD LVCMOS33 } [get_ports { Cnode[4] }]; #IO_25_15 Sch=cc
set_property -dict { PACKAGE_PIN K13   IOSTANDARD LVCMOS33 } [get_ports { Cnode[3] }]; #IO_L17P_T2_A26_15 Sch=cd
set_property -dict { PACKAGE_PIN P15   IOSTANDARD LVCMOS33 } [get_ports { Cnode[2] }]; #IO_L13P_T2_MRCC_14 Sch=ce
set_property -dict { PACKAGE_PIN T11   IOSTANDARD LVCMOS33 } [get_ports { Cnode[1] }]; #IO_L19P_T3_A10_D26_14 Sch=cf
set_property -dict { PACKAGE_PIN L18   IOSTANDARD LVCMOS33 } [get_ports { Cnode[0] }]; #IO_L4P_T0_D04_14 Sch=cg
set_property -dict { PACKAGE_PIN H15   IOSTANDARD LVCMOS33 } [get_ports { dp }]; #IO_L19N_T3_A21_VREF_15 Sch=dp
set_property -dict { PACKAGE_PIN J17   IOSTANDARD LVCMOS33 } [get_ports { AN[0] }]; #IO_L23P_T3_FOE_B_15 Sch=an[0]
set_property -dict { PACKAGE_PIN J18   IOSTANDARD LVCMOS33 } [get_ports { AN[1] }]; #IO_L23N_T3_FWE_B_15 Sch=an[1]
set_property -dict { PACKAGE_PIN T9    IOSTANDARD LVCMOS33 } [get_ports { AN[2] }]; #IO_L24P_T3_A01_D17_14 Sch=an[2]
set_property -dict { PACKAGE_PIN J14   IOSTANDARD LVCMOS33 } [get_ports { AN[3] }]; #IO_L19P_T3_A22_15 Sch=an[3]
set_property -dict { PACKAGE_PIN P14   IOSTANDARD LVCMOS33 } [get_ports { AN[4] }]; #IO_L8N_T1_D12_14 Sch=an[4]
set_property -dict { PACKAGE_PIN T14   IOSTANDARD LVCMOS33 } [get_ports { AN[5] }]; #IO_L14P_T2_SRCC_14 Sch=an[5]
set_property -dict { PACKAGE_PIN K2    IOSTANDARD LVCMOS33 } [get_ports { AN[6] }]; #IO_L23P_T3_35 Sch=an[6]
set_property -dict { PACKAGE_PIN U13   IOSTANDARD LVCMOS33 } [get_ports { AN[7] }]; #IO_L23N_T3_A02_D18_14 Sch=an[7]

##Buttons
set_property -dict { PACKAGE_PIN C12   IOSTANDARD LVCMOS33 } [get_ports { rst_n }]; #IO_L3P_T0_DQS_AD1P_15 Sch=cpu_resetn
```

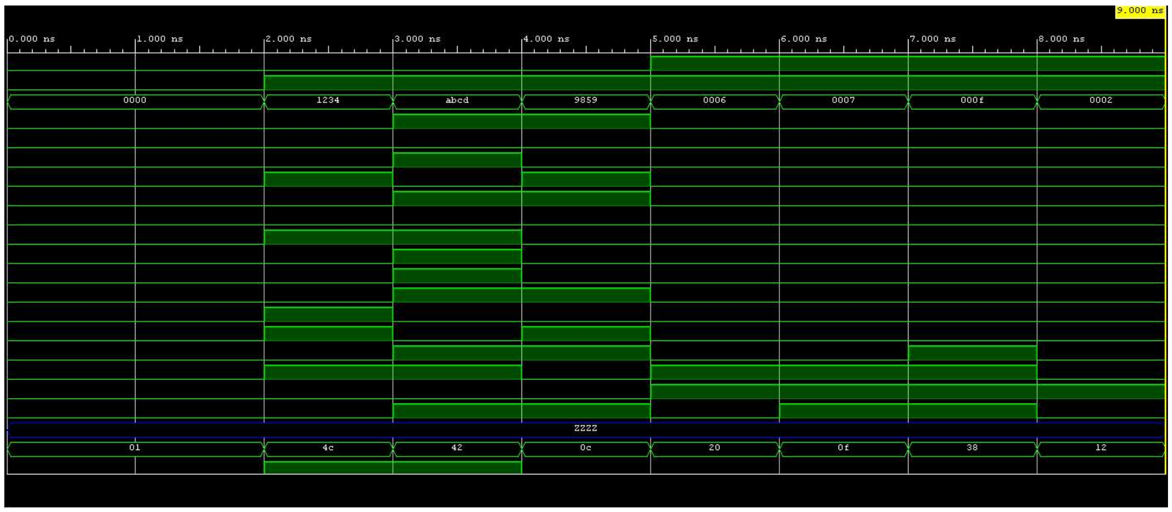
The top file declares the 16 bit input SW concatenated to fit a 32 bit display, therefore mirroring the outputs on the 4 segment display. An LED is also assigned to each corresponding switch to show the state of the switches. The module is then instantiated.

```
module top(
  input clk,
  input rst_n,
  input [15:0] sw,
  output [15:0] led,
  output [6:0] Cnode,
  output dp,
  output [7:0] AN
          );


  assign led = sw;  // Mirror switches to LEDs


  wire [31:0] sw_ext = {sw, sw};  // Zero-extend to match 32-bit input

  seg7_driver display(
    .clk(clk),
    .rst_n(rst_n),
    .sw(sw_ext),
    .Cnode(Cnode),
    .dp(dp),
    .AN(AN)
  );

endmodule
```

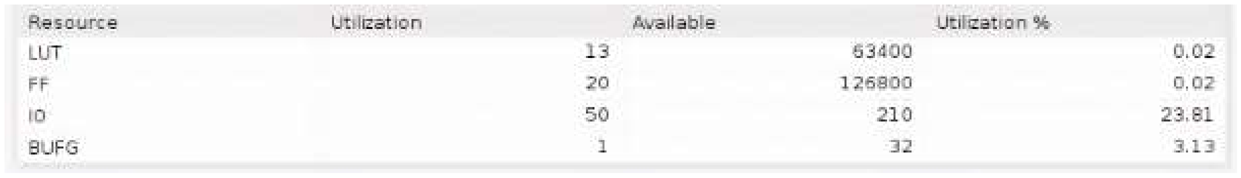
The testbench first instantiates the top module and then tests the reset button, and different values of switch which results are shown after.

```
`timescale 1ns / 1ps

module top_tb;

  reg clk = 0;
  reg rst_n = 0;
  reg [15:0] sw;
  wire [15:0] LED;
  wire [6:0] Cnode;
  wire dp;
  wire [7:0] AN;

  // Instantiate DUT
  top uut (
    .clk(clk),
    .rst_n(rst_n),
    .sw(sw),
    .led(led),
    .Cnode(Cnode),
    .dp(dp),
    .AN(AN)
  );

  // Clock generator: 10ns period → 100 MHz
  always #5 clk = ~clk;

  initial begin
    rst_n = 0; sw = 16'h0000;
    #2;
    rst_n = 1;
    sw = 16'h1234;
    #1;
    sw = 16'hABCD;
    #1;
    sw = 16'h9859;
    #1;
    sw = 4'b0110;
    #1;
    sw = 4'b00111;
    #1;
    sw = 4'b1111;
    #1;
    sw = 4'b0010;
    #1;
    $stop;
  end

endmodule
```

Here is the behavioral testbench simulation that verifies all inputs and outputs.



The utilization report is shown below as well as the percent utilized. Our I/O is the both utilized as we are using switches, LEDs, 7 Segment displays and a button.



Resource	Utilization	Available	Utilization %
LUT	13	63400	0.02
FF	20	126800	0.02
IO	50	210	23.81
BUFG	1	32	3.13

Robert's Section:
I uploaded the final code to github and used my testbench code for the final simulation. I created my own testbench code and top code and worked on the top code together to debug errors. I gathered the behavioral simulation. We both explained different parts of the code on the lab report. I covered the explanation for the testbench code and the constraint file.
Dia's Section:
I created a testbench code and a top code as well. I helped debug errors in the top code especially and combined my code with my lab partner. I grabbed the utilization report and then created the video demo for our lab. In the lab report, I wrote the sections explaining the driver code and the top code.