3300 Lab 4
Switch-to-7-Segment Display Interface on Nexys A7

Group C
Rohan Walia
Parsa Ghasemi
7/11/25

Code:
```
`timescale 1ns / 1ps

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];
    assign dp = 1'b1;

    reg [19:0] tmp;
    reg [3:0] digit;
    reg [7:0] AN_tmp;

    // Cnode logic based on digit
    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

    // Clock and reset logic
    always @(posedge clk or negedge rst_n)
```

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

    // Digit selection based on s
    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

    // AN (anode) signal selection
    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
```

```
`timescale 1ns / 1ps

module seg7_driver_tb();

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

    // Instantiate seg7_driver
    seg7_driver uut (
        .clk(clk),
        .SW(SW),
        .rst_n(rst_n),
        .Cnode(Cnode),
        .dp(dp),
        .AN(AN),
        .LED(LED)
    );

    always #5 clk = ~clk;

    initial begin
        clk = 0;
        rst_n = 0;
        SW = 16'h0000;

        #20 rst_n = 1;

        SW = 16'hBEEF;
        #2_000_000;

        SW = 16'h1234;
        #2_000_000;

        $finish;
    end

endmodule
```

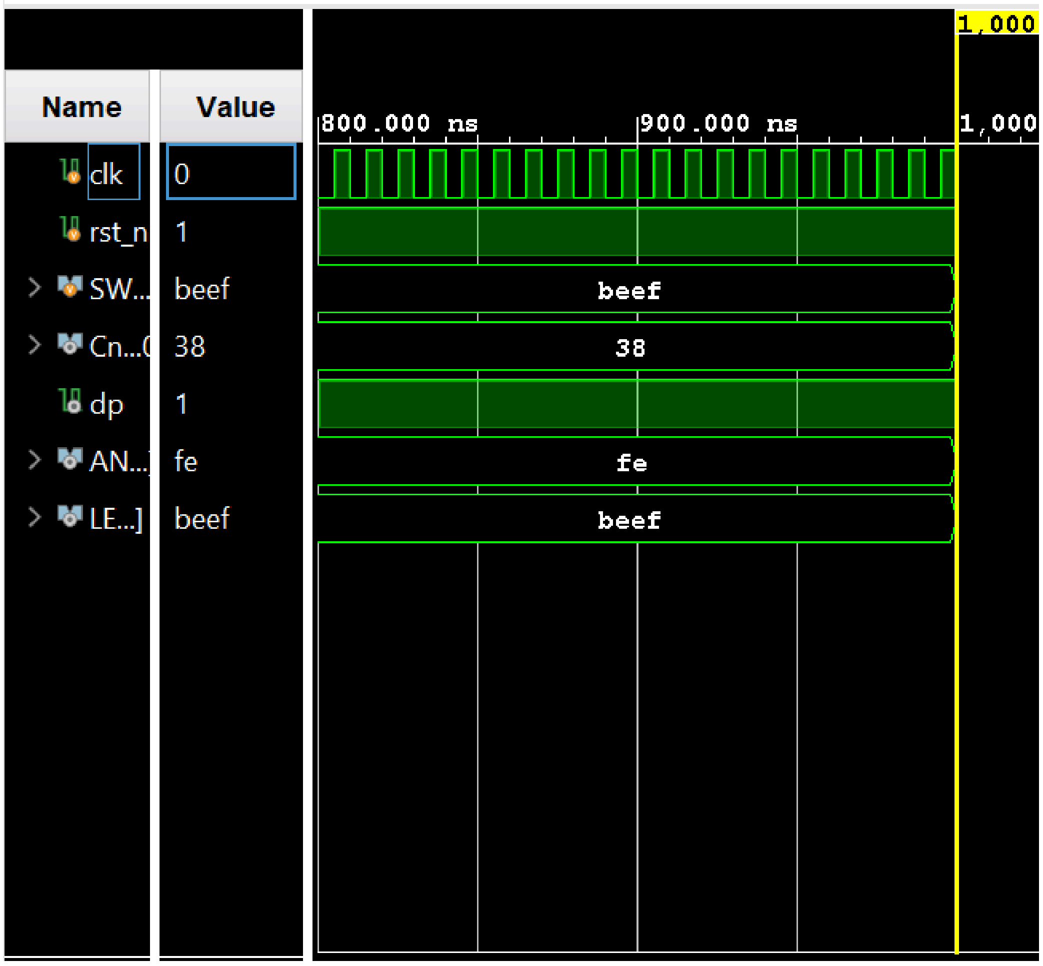
```
## This file is a general .xdc for the Nexys4 DDR Rev. C
## To use it in a project:
## - uncomment the lines corresponding to used pins
## - rename the used ports (in each line, after get_ports) according to the top level signal
names in the project

## 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 {clk}];


##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 LVCMOS33 } [get_ports { SW[8] }];
#IO_L24N_T3_34 Sch=sw[8]
set_property -dict { PACKAGE_PIN U8    IOSTANDARD LVCMOS33 } [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
```

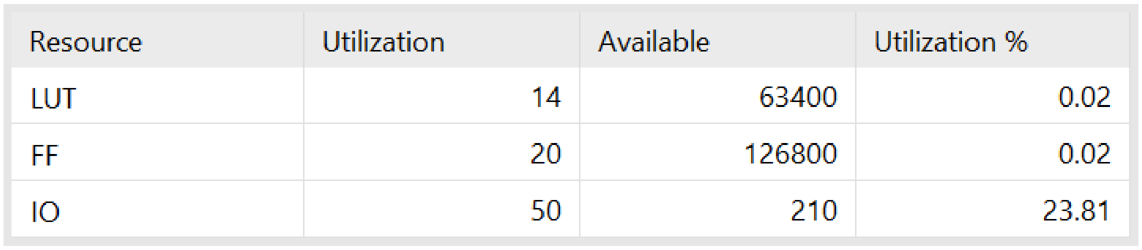
The seg7_driver.v module shows numbers on an 8-digit 7-segment display by quickly switching between digits using a clock signal. It takes a 16-bit input from switches (SW) and splits it into 4 bits each to show hex digits on the right side of the display. It uses a counter to decide which digit is active, and case statement to light up the correct segments for each number. The testbench seg7_driver_tb.v turns on the clock, gives the module the value BEEF, and waits to see the digits. This helps check that everything works.

Waveform screenshot

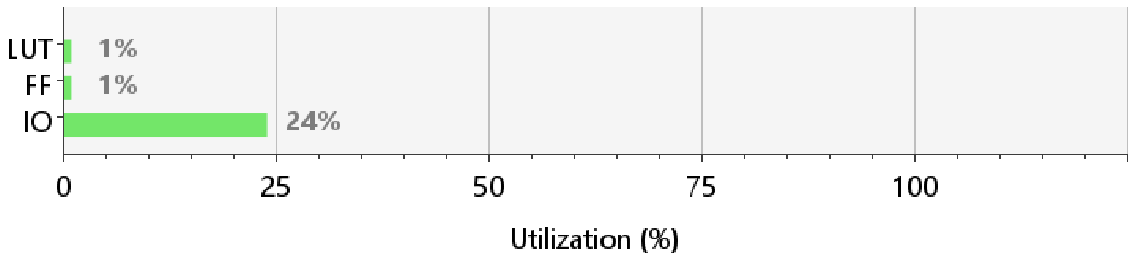


Name	Value
clk	0
rst_n	1
SW...	beef
Cn...(	38
dp	1
AN...]	fe
LE...]	beef

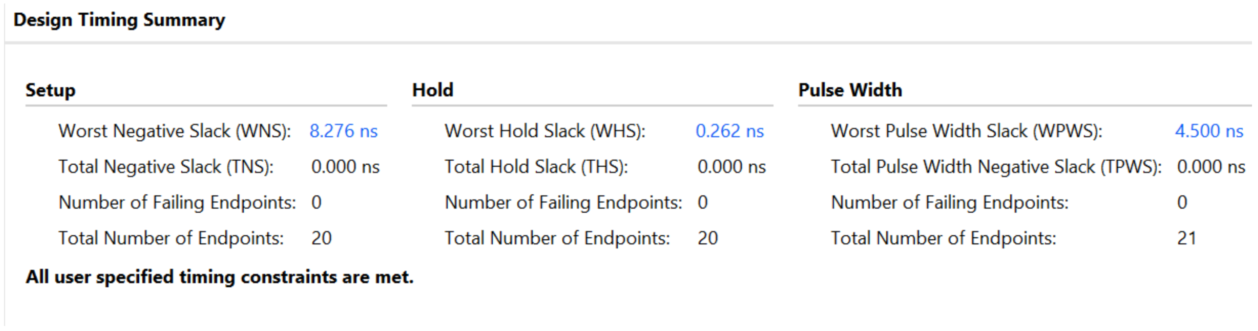
LUTs and FF screenshot:



Resource	Utilization	Available	Utilization %
LUT	14	63400	0.02
FF	20	126800	0.02
IO	50	210	23.81



Timing screenshot:



Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	8.276 ns	Worst Hold Slack (WHS):	0.262 ns	Worst Pulse Width Slack (WPWS):	4.500 ns
Total Negative Slack (TNS):	0.000 ns	Total Hold Slack (THS):	0.000 ns	Total Pulse Width Negative Slack (TPWS):	0.000 ns
Number of Failing Endpoints:	0	Number of Failing Endpoints:	0	Number of Failing Endpoints:	0
Total Number of Endpoints:	20	Total Number of Endpoints:	20	Total Number of Endpoints:	21

All user specified timing constraints are met.

Contribution:
Rohan Walia (50%): Implementation, demo, verilog, report
Parsa Ghasemi (50%): testbench code, testing, report

Demo link: https://youtu.be/-nOBAn13pNE

Reflections: Overall the lab helped us get a better understanding of converting binary inputs to hex and getting the seven segment displays to work on the board. Out understanding of decoders, multiplexers, and decimal conversion has been greatly improved.