Lab 2 (100 points): Combinational Logic Design – Two-Digit Display

Objectives

- Use the combinational logic design process to design a combinational circuit – seven-segment decoder
- Learn how to use a vector (multi-bit) signal in Verilog
- Use the testbench to validate the designed code before implementation
- Program the Nexys4 DDR FPGA board to implement the designed circuit

Start: Week 4 (September 11, 2023) – attend your scheduled lab session
Demo Due: by the end of your lab session during Week 4

Code submission: Submit the following files by 11.59 PM on Friday September 15, 2023

1) Verilog (.v) file of your seven-segment decoder.
2) Verilog (.v) file for Testbench of your SevenSegment circuit (Part (A))
3) .xdc file that you modify in part (C)

Files available on D2L under Lab 2 module (zipped folder Lab2_Codetemplate)

- Template Verilog Module for a seven-segment decoder (SevenSegment.v)
- Verilog Module for 2-digit display (TwoDigitDisplay.v)
- Constraint file (Nexys4DDR_Master.xdc)

Reference:

- On D2L, under "Resources_Xilinx_Nexys4_DDRBoard module", ***Nexys4ddr_ReferenceManual.pdf*** ***(section 10: Basic I/O page 17-20)***
- On D2L, under Content -> Vivado_DownLoad_Tutorial_Resources,
  Xilinx Vivado **Simulation Tutorial** and
  Xilinx Vivado **Synthesis Tutorial**, Nexys 4DDR
- On D2L, under Content -> Lecture Notes -> Unit: Verilog, Verilog_Number_Vector_Concatenator.pptx and Verilog_If_case.pptx

Lab Overview

In this lab assignment,

(A) You will first design and synthesize a seven-segment decoder.

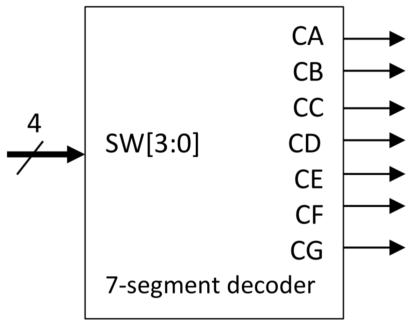
(B) You will then implement it on the Nexys4 DDR FPGA board

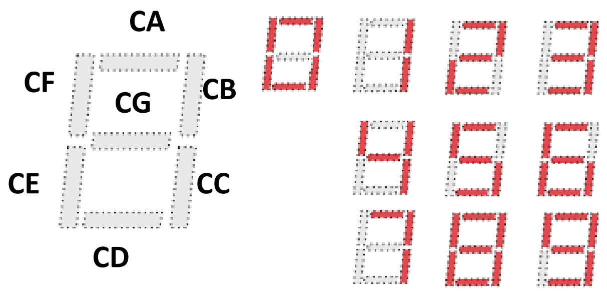
(C) Your designed seven-segment decoder module will then be used in the circuit that can display 2-digit numbers (TwoDigitDisplay.v). This circuit will also be implemented on the Nexys4 DDR board.

Part (A) Seven-segment decoder

A seven-segment decoder is used to convert a binary coded decimal (BCD) into a 7-segment code such that it displays a decimal number based on the input value. It has 4 inputs or a 4-bit input and 7 outputs where each output controls each individual LED.

The Nexys4 DDR board contains two four-digit common anode seven-segment LED displays. Each display digit has 7 segments called CA, CB, CC, …, CG. Each segment can be activated independently such that they can be used to display number 0 – 9 as shown below.





To activate each segment (make it light up), <u>logic 0 is used</u>.

For example,

if the **4-bit input** is **1** (binary 0001), to display 1, we want to activate segment CB and CC. Therefore, the outputs are CA = 1, CB = 0, CC = 0, CD = 1, CE = 1, CF = 1, CG = 1.

if the **4-bit input** is **6** (binary 0110), to display 6, we want to activate every segment EXCEPT segment CB. Therefore, the outputs are CA = 0, CB = 1, CC = 0, CD = 0, CE = 0, CF = 0, CG = 0.

A1) Design the seven-segment decoder by **writing its truth table**. Since we want to display only a decimal digit, when the 4-bit input represents a decimal value from 10-15, we will turn OFF all the segments by making all outputs to 1 as shown in the table below

SW[3]	SW[2]	SW[1]	SW[0]	CA	CB	CC	CD	CE	CF	CG
0	0	0	0							
0	0	0	1							
0	0	1	0							
0	0	1	1							
0	1	0	0							
0	1	0	1							
0	1	1	0							
0	1	1	1							
1	0	0	0							
1	0	0	1							
1	0	1	0	1	1	1	1	1	1	1
1	0	1	1	1	1	1	1	1	1	1
1	1	0	0	1	1	1	1	1	1	1
1	1	0	1	1	1	1	1	1	1	1
1	1	1	0	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1

A2) Write the **Verilog code using case statement** in the code template called **SevenSegment.v** (on D2L, under Lab 2 module, you will see the zipped folder called ***Lab2_Codetemplate***)

Note: you MUST use the given template and make sure that the module is named "SevenSegment". Otherwise, your circuit will NOT work when you move to Part B of this lab.

In the SevenSegment.v file, it contains

```
`timescale 1ns / 1ps
module SevenSegment(SW,CA,CB,CC,CD,CE,CF,CG);
        // 4-bit input using switches 3 - 0
   input [3:0] SW;
        //each segment of seven-segment display
   output reg CA, CB, CC, CD, CE, CF, CG;

        //write your code using case statement here
endmodule
```

Note:

1) Do not change the name of input and output signals. These names are used such that they match with the signal names used in the Nexys4DDR_Master xdc file.
2) We will use a multi-bit or vector signal for the 4-bit input of the 7-segment decoder.
   Multi-bit or vector signal in Verilog
   Consider [3:0] numin,
   the signal *numin* has 4 bits, for example, if numin = 4'b1100, it means that

numin[3]	numin[2]	numin[1]	numin[0]
1	1	0	0

A3) **Write a testbench** for this 7-segment decoder. Use the followings to get started.

```
`timescale 1ns / 1ps
module SevenSegment_tb( );
   reg [3:0] SW_tb;     //a vector signal is also used in the testbench
   wire CA_tb, CB_tb, CC_tb, CD_tb, CE_tb, CF_tb, CG_tb;

   //Call SevenSegment module to be tested here

   initial
   begin
        //case 0
        SW_tb = 4'b0000;     //b is for binary and 4 is for 4-bit
         #100;
        SW_tb = 4'b0001;     //case 1
        #100;
        // use this idea for other combinations (all combinations of inputs)
    end
```

A3.1) Perform the behavioral simulation

A3.2) Run synthesis and perform the post-synthesis functional simulation.
When your group gets the correct waveforms, Demo them to TA or ULA.

Part B) Download to FPGA board
B1) Pin assignments:

In the zipped folder ***Lab2_Codetemplate*** on D2L under lab2 module that you download, there is a file called ***Nexys4DDR_Master.xdc***.

In your Vivado project, use Add Sources, choose Add or create constraints, choose Add Files to add ***Nexys4DDR_Master.xdc*** as the constraint file to your Vivado project.

B2) click on Run Synthesis, Run Implementation, Generate Bitstream.

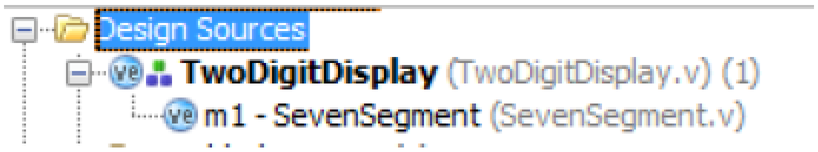
B3) When you successfully generate bitstream, program or download to the FPGA board.

If it works, all seven-segment digits should display the decimal number corresponding to the binary number that you enter using the 4 switches. TA and ULA will test all numbers from 0 – 9 to make sure that your designed circuit works.

Part C) Implement 2-digit display on the Nexys4 DDR board

The Verilog code (called TwoDigitDisplay.v in the zipped lab 2 folder) is already written to implement the display for two digits (00 – 99). Your designed 7-segment display will be called as a module in this code.

C1) In your Vivado project (the same one that you use for "SevenSegment.v"), use Add Sources, choose Add or create design sources, choose Add Files to add ***TwoDigitDisplay.v*** to your Vivado project.
DO NOT MODIFY TwoDigitDisplay.v code. You can use this code without any modification. Right-click on the "TwoDigitDisplay.v" file under Design Sources (Project Manager) and choose "**set as Top**". This is to make the TwoDigitDisplay module the top-level design of the Vivado project. If working, you should see the following two files in Design Sources. Observe the three dots (signify "top level") in front of the TwoDigitDisplay.v file.



Design Sources
TwoDigitDisplay (TwoDigitDisplay.v) (1)
m1 - SevenSegment (SevenSegment.v)

In TwoDigitDisplay.v, the 7-bit input [6:0] SW is used since the decimal 00-99 are represented by 7-bit binary number.

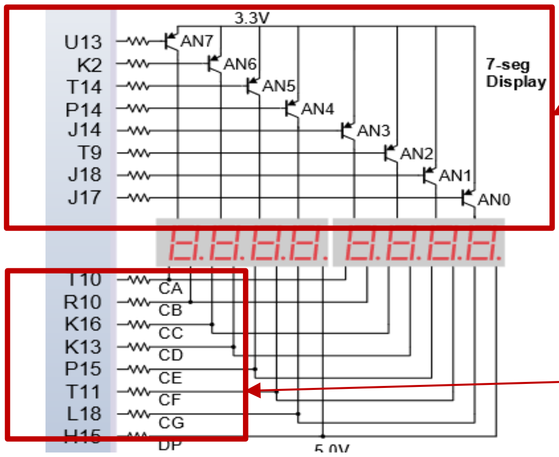
C2) Open *Nexys4DDR_Master.xdc*, **Uncomment (remove #)** in front of the following statements in that file.

DO NOT MODIFY any statement in the .xdc file (only remove # to uncomment). Find the statements below in the file. Then remove # in front of that statement.

This is done so that the pins mapping to the inputs and outputs of the circuit can be used by Vivado to generate proper bitstream.

```
set_property -dict { PACKAGE_PIN E3    IOSTANDARD LVCMOS33 } [get_ports { CLK100MHZ
}]; #IO_L12P_T1_MRCC_35 Sch=clk100mhz
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 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]
```



AN[x] above are used to control which digit to turn on

CA, CB, CC, CD, CE, CF, CG are used to control each segment

Resource: From page 18 of *Nexys4ddr_ReferenceManual.pdf on D2L*

C3) click on Run Synthesis, Run Implementation, Generate Bitstream.

C4) When you successfully generate bitstream, you will now program the FPGA board.

When your group successfully program the FPGA board, demo them to TA or ULA by the end of your lab session. TA will test by picking a few cases of decimal value between 0 – 99 and you will enter the binary representation of each decimal number using the switches (KNOW YOUR BINARY) to make the 7-segment digits on the board display that number.

Grading Rubrics

1) You can design the 7-segment decoder (write its truth table)	maximum of 15 out of 100 points
2) You can write a Verilog code to describe the designed circuit in 1)	maximum of 45 out of 100 points
3) You can write a testbench to perform a Behavioral simulation and the post-synthesis functional simulation	maximum of 70 out of 100 points
4) You can successfully program the FPGA board for the 7-segment decoder	maximum of 90 out of 100 points
5) You can successfully program the FPGA board for the two-digit display circuit	maximum of 100 out of 100 points