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
- Continue using the Nexys4 DDR board to implement the designed circuit

Starts: Week 4 (starting September 10, 2018)

Demo Due: at the end of your lab session

Demo: Demo both part A) and B) of this lab handout to TAs/ULA.

Pre-Lab Assignment (25 pts) individual work:

1) Create the truth table and derive the optimized/minimized Boolean expression in sum-of-products form for the seven-segment decoder. See section A.1 on page 2. Use K-map to minimize the equation for each output.
2) Write the Verilog code for your circuit.

The TA will check your work during the first 30 mins of your lab session. If you do not have it, you will get 0 point for this part.

Code submission: Submit 2 Verilog (. v) files and one .xdc files by 11.59 PM on Friday Sep 14, 2018

1) Verilog (.v) file – using ***Procedural assignment statement*** of your seven-segment decoder.
2) Verilog (.v) file for Testbench of your SevenSegment circuit (Part (A))
3) .xdc file for your SevenSegment circuit (Part (A))

Files available on D2L under Lab 2 module

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

Reference:

- On D2L, under "Resources_Xilinx_Nexys4_DDRBoard module", ***Nexys4ddr_ReferenceManual.pdf*** (***section 10: Basic I/O page 17-20)***

Lab Overview

In this lab assignment,

A) You will first ***design, functionally simulate,*** and ***implement a seven-segment decoder on the Nexys4 DDR board.***
The seven-segment decoder has 4 inputs and 7 outputs. It receives a 4-bit input number and activates the appropriate segments of the display such that the decimal value of the input is shown.

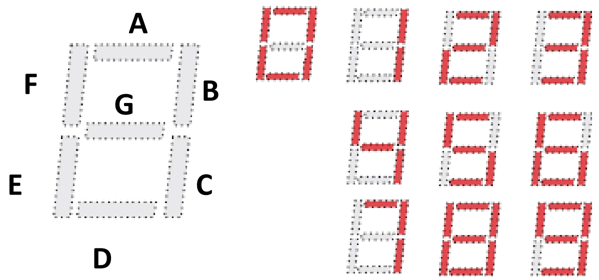
B) Your designed seven-segment decoder module will then be used in the circuit that can display 2-digit numbers (TwoDigitDisplay.v). This circuit will also be implemented on the Nexys4 DDR board.

Part A)

A.1 Follow the combinational logic design process to design the **seven-segment decoder**

- Create the truth table
- Write the Boolean expression in sum of products format (optimize/minimize it using K-map or Boolean Algebra properties)

The Nexys4 DDR board contains two four-digit common anode seven-segment LED displays. Each display digit has 7 segments called A, B, C, …, G. 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**, we want to activate segment B and C. Therefore the outputs are A = 1, B = 0, C = 0, D = 1, E = 1, F = 1, G = 1.

if the **4-bit input** is **6**, we want to activate every segment EXCEPT segment B. Therefore the outputs are A = 0, B = 1, C = 0, D = 0, E = 0, F = 0, G = 0.

Below is the **truth table (complete it).** Then you will **use K-map** (or Boolean Algebra properties) **to find the optimized Boolean expression for each output (segment).**

4-bit input

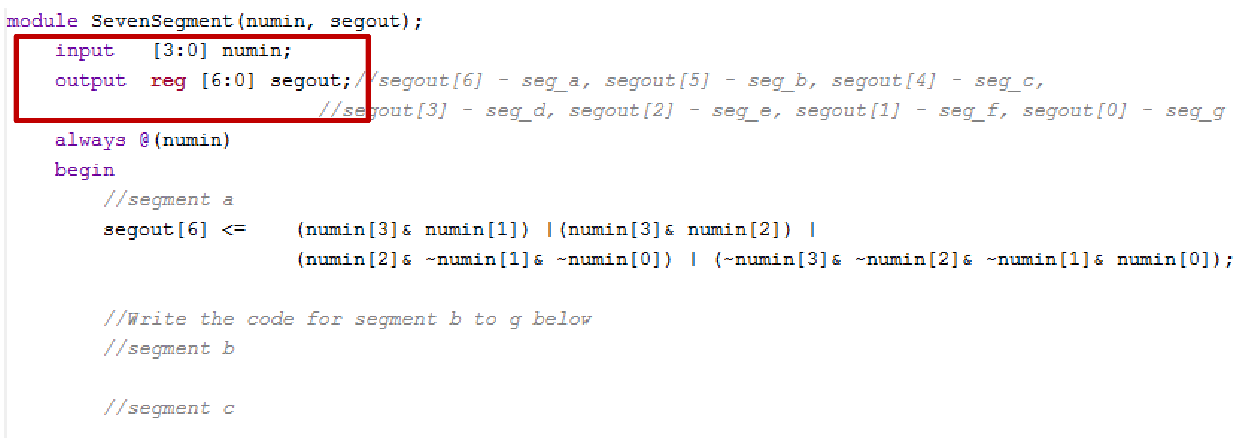
n3	**n2**	**n1**	**n0**	**segout6 A**	**segout5 B**	**segout4 C**	**segout3 D**	**segout2 E**	**segout1 F**	**segout0 G**
0	0	0	0							
0	0	0	1							
0	0	1	0							
0	0	1	1							
0	1	0	0							
0	1	0	1							
0	1	1	0							
0	1	1	1							
1	0	0	0							
1	0	0	1							
1	0	1	0	1	1	1	1	1	1	1
1	0	1	1	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

Note: when the decimal values of the 4-bit number are 10 -15, all segments/outputs are set to 1 (to make them NOT light up).

A.2 Write the Verilog code **using a given template** on D2L (under lab 2) called **SevenSegment.v**

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, you will see the following:



```
module SevenSegment(numin, segout);
    input   [3:0] numin;
    output  reg [6:0] segout;//segout[6] - seg_a, segout[5] - seg_b, segout[4] - seg_c,
                           //segout[3] - seg_d, segout[2] - seg_e, segout[1] - seg_f, segout[0] - seg_g
    always @(numin)
    begin
        //segment a
        segout[6] <=    (numin[3]& numin[1]) |(numin[3]& numin[2]) |
                        (numin[2]& ~numin[1]& ~numin[0]) | (~numin[3]& ~numin[2]& ~numin[1]& numin[0]);

        //Write the code for segment b to g below
        //segment b

        //segment c
```

A multi-bit (vector) signal in Verilog:

Consider [3:0] numin,

the signal *numin* has 4 bits, for example, if numin = 0110, it means that

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

The output is defined as [6:0] segout where segout[6] corresponds to segment A of the seven-segment display.
Additionally, the Verilog code for segment A (segout[6]) is already given to show how you can use each individual bit of the multi-bit signal to write your logic equation.

How do I get that equation? For segment A (segout[6]),

	n1n0			
n3n2	**00**	**01**	**11**	**10**
00	0	1	0	0
01	1	0	0	0
11	1	1	1	1
10	0	0	1	1

segout[6] = n3n2 + n3n1 + n2n1'n0' + n3'n2'n1'n0

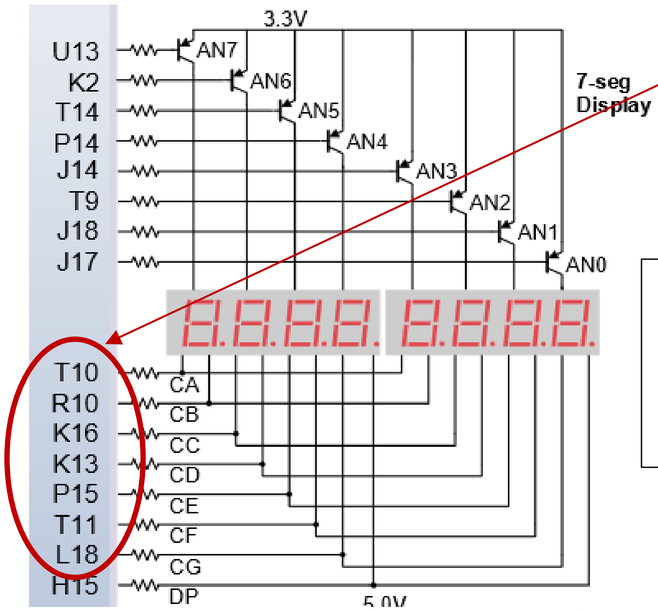
A.3 Functionally simulate your Verilog code of your designed seven-segment decoder (Write the testbench). When you write your testbench, use the followings to get started.

```
module SevenSegment_tb( );
   reg [3:0] numin_tb;                //vector signal is also used in the testbench
   wire [6:0] segout_tb;

   SevenSegment m1(numin_tb, segout_tb);

   initial
   begin
      //case 0
      numin_tb = 4'b0000;             //4'b – b is for binary and 4 is for 4-bit
      #10
      //case 1
      numin_tb = 4'b0001;
      // use this idea for other combinations (all combinations of inputs)
```

A.4 If the waveform is correct, assign pin numbers to the input and output signals of your designed circuit (create .xdc file). Pick 4 swtiches (prefer SW3, SW2, SW1 and SW0) for the 4 input signals and use the picture below with pin numbers for the 7 output signals.



Use these pins numbers for

segment A (segout[6]) (CA in the picture)

to

segment G (segout[0]) (CG in the pic)

One example of assigning pin T10 to segment A (segout[6])

```
set_property IOSTANDARD LVCMOS33 [get_ports {segout[6]}]

set_property PACKAGE_PIN T10 [get_ports {segout[6]}]
```

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

A.5 Implement it on the Nexys4 DDR 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. If you see all numbers 0 – 9 working correctly, demo this part (part A) to your TAs/ULA.

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

To implement the display for two digits (00 – 99). Your designed 7-segment display will be used as a module in the circuit.

B.1 Get the two following files from lab 2 folder on D2L:

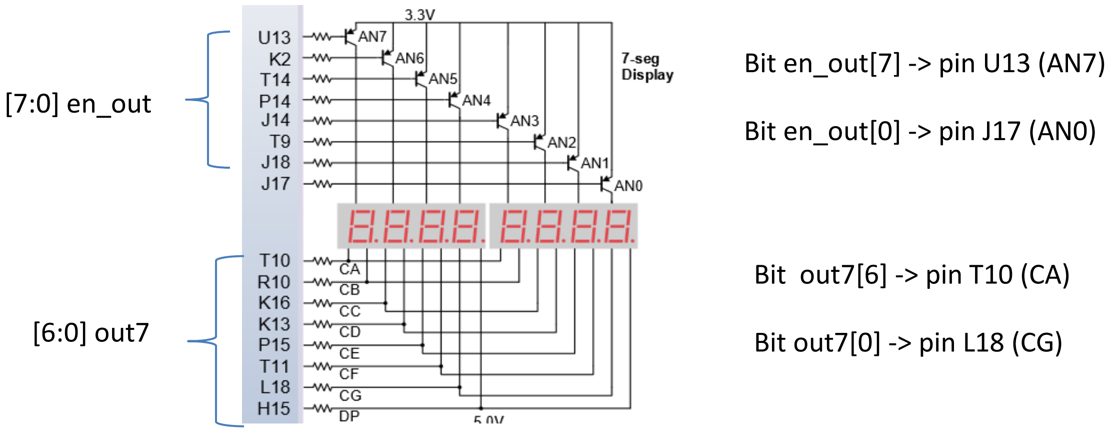
1) TwoDigitDisplay.v and 2) TwoDigitDisplay.xdc files.

In **TwoDigitDisplay.v**, the inputs that you will use is **[6:0] Number**. The decimal 00-99 are represented by 7-bit binary number.

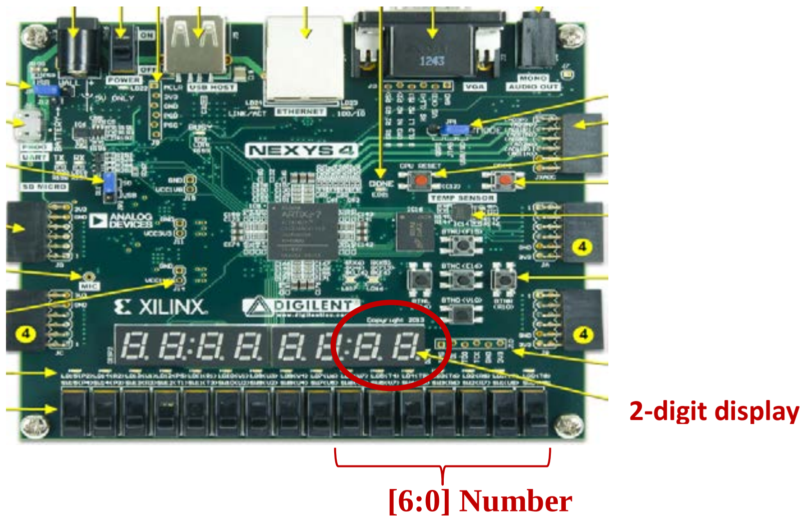
```
module TwoDigitDisplay(Clk, Number, out7, en_out);
   input  Clk;
   input  [6:0] Number;
   output [6:0] out7; //seg a, b, ... g
   output reg [7:0] en_out;
```

In **TwoDigitDisplay.xdc**, **the pins are already mapped for you** – you can start using this file in your project

- Clk is mapped to pin **E3 (100 MHz osciallator)**
- SW6 – SW0 are mapped to Number[6] – Number[0]
  Number[6] is Most Significant Bit (MSB) of a given number -> SW6 (pin U18)
  Number[0] is Least Significant Bit (LSB) of a given number -> SW0 (pin J15)



Below are what you will use



B.2 Synthesize and implement this circuit to the Nexys4 DDR board. Check to see that the 2 digits displayed correctly correspond to the 7-bit binary number (from using 7 switches above).

When you understand how this works, demo it to TA. TA will ask you to show several numbers by moving the switches (KNOW YOUR BINARY).

Grading Criteria:

Pre-lab (Individual work) (25 points)

1. Create the truth table and derive the optimized/minimized Boolean expression in sum of products form for the seven-segment decoder
2. Write the Verilog code of the seven-segment decoder.

In- lab:

1) Part A: Simulation and Download to the board

   1) (12 pts) Write a testbench to functionally simulate (Behavioral simulation) your seven-segment decoder to exhaustively test all possible input combinations.
   2) (13 pts) Run Synthesis and Do Post-synthesis simulation.

   Demo the above to TA before going to the next step.

   3) (25 pts) Write .xdc file to assign the pins) and program your design of 7-segment decoder to the Nexys4 board.

   When working, Demo this part to TA for every input combination (0000 – 1111).

3. (25 points) Part B (Implement 2-digit display): synthesize and download the TwoDigitDisplay on the board.
   When working, Demo this part to TA. TA will ask you to display several decimal numbers.