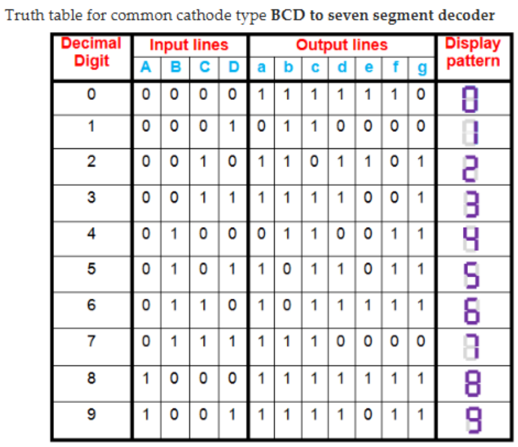
Laboratory Exercise 2

Numbers and Displays

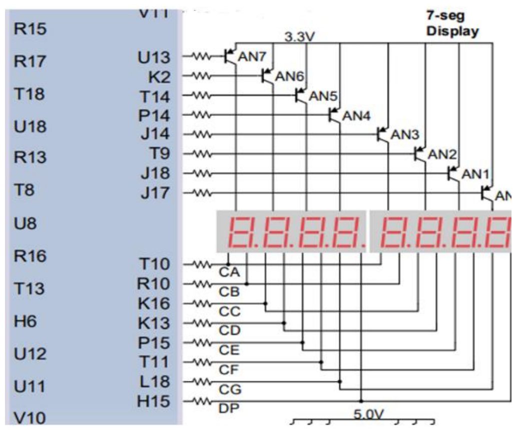
This is an exercise in designing combinational circuits that can perform binary-to-decimal number conversion and binary-coded-decimal (BCD) addition.

Part I

We wish to display on the 7-segment displays *HEX7* to *HEX0* the values set by the switches SW_{15-0}. The objective of this task is to implement and verify the BCD (Binary-Coded Decimal) to Decimal conversion using the Nexys 4 DDR FPGA development board. You will design a digital circuit that takes a 4-bit BCD input and converts it to its corresponding decimal value, displaying the result on **one seven-segment display (HEX0)** of the Nexys 4 DDR kit. Your circuit should be able to display the digits from **0 to 9** and should treat the valuations 1010 to 1111 as don't-cares.


Truth table for common cathode type **BCD to seven segment decoder**

Decimal Digit	Input lines				Output lines							Display pattern
	A	B	C	D	a	b	c	d	e	f	g	
0	0	0	0	0	1	1	1	1	1	1	0	0
1	0	0	0	1	0	1	1	0	0	0	0	1
2	0	0	1	0	1	1	0	1	1	0	1	2
3	0	0	1	1	1	1	1	1	0	0	1	3
4	0	1	0	0	0	1	1	0	0	1	1	4
5	0	1	0	1	1	0	1	1	0	1	1	5
6	0	1	1	0	1	0	1	1	1	1	1	6
7	0	1	1	1	1	1	1	0	0	0	0	7
8	1	0	0	0	1	1	1	1	1	1	1	8
9	1	0	0	1	1	1	1	1	0	1	1	9



1. Create a new project which will be used to implement the above truth table on the Nexys 4 DDR FPGA board. The intent of this exercise is to manually derive the logic functions needed for the 7-segment displays. You should use only simple **Verilog case statements** in your code and specify each logic function as a Boolean expression.

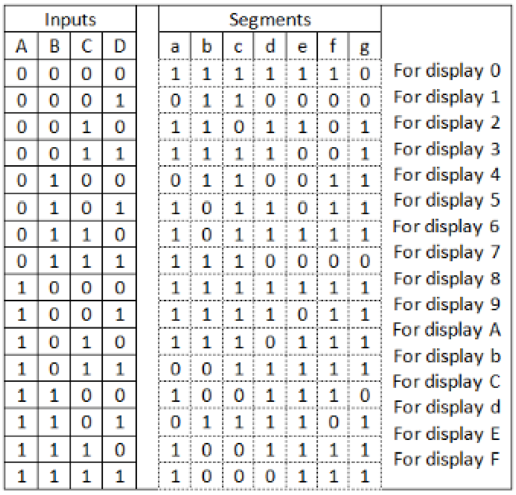
2. Write a Verilog code that provides the necessary functionality. Include this code in your project and assign the pins on the FPGA to connect to the switches and 7-segment displays, as indicated in the above snap of 7-seg display of Nexys 4 DDR FPGA.

3. Compile the project and download the compiled circuit into the FPGA chip.

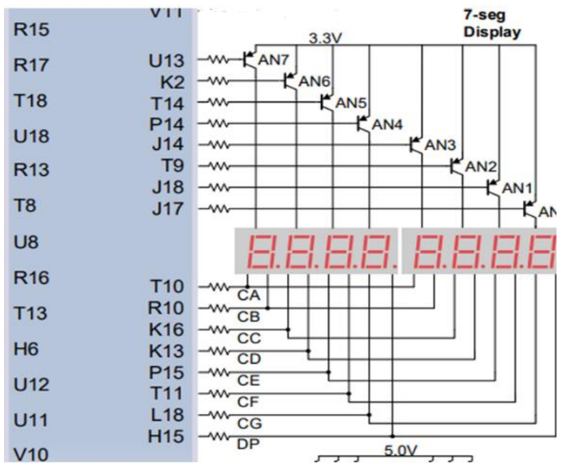
4. Test the functionality of your design by toggling the switches and observing the displays.

Part II

We wish to display on the 7-segment displays *HEX7* to *HEX0* the values set by the switches SW_{15-0}. The objective of this lab task is to design, implement, and verify a Binary Coded Decimal (BCD) to Hexadecimal conversion circuit on the Nexys 4 DDR FPGA development board. You will create a digital circuit that takes a 4-bit BCD input, converts it to its corresponding Hexadecimal value, and displays the result on two seven-segment display **(HEX1, HEX2)** of the Nexys 4 DDR kit. Your circuit should be able to display the digits from **0 to F** and should treat the valuations above 1111 as don't-cares.



Inputs					Segments							
A	B	C	D		a	b	c	d	e	f	g	
0	0	0	0		1	1	1	1	1	1	0	For display 0
0	0	0	1		0	1	1	0	0	0	0	For display 1
0	0	1	0		1	1	0	1	1	0	1	For display 2
0	0	1	1		1	1	1	1	0	0	1	For display 3
0	1	0	0		0	1	1	0	0	1	1	For display 4
0	1	0	1		1	0	1	1	0	1	1	For display 5
0	1	1	0		1	0	1	1	1	1	1	For display 6
0	1	1	1		1	1	1	0	0	0	0	For display 7
1	0	0	0		1	1	1	1	1	1	1	For display 8
1	0	0	1		1	1	1	1	0	1	1	For display 9
1	0	1	0		1	1	1	0	1	1	1	For display A
1	0	1	1		0	0	1	1	1	1	1	For display b
1	1	0	0		1	0	0	1	1	1	0	For display C
1	1	0	1		0	1	1	1	1	0	1	For display d
1	1	1	0		1	0	0	1	1	1	1	For display E
1	1	1	1		1	0	0	0	1	1	1	For display F



1. Create a new project which will be used to implement the above truth table on the Nexys 4 DDR FPGA board. The intent of this exercise is to manually derive the logic functions needed for the 7-segment displays. You should use only simple **Verilog case statements** in your code and specify each logic function as a Boolean expression.

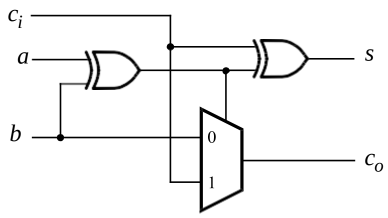
2. Write a Verilog code that provides the necessary functionality. Include this code in your project and assign the pins on the FPGA to connect to the switches and 7-segment displays, as indicated in the above snap of 7-seg display of Nexys 4 DDR FPGA.

3. Compile the project and download the compiled circuit into the FPGA chip.

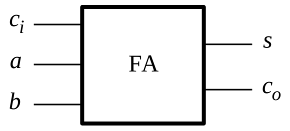
4. Test the functionality of your design by toggling the switches and observing the displays.

Part III

Figure 2*a* shows a circuit for a *full adder*, which has inputs a, b, and c_i, and produces the outputs s and c_o. Parts *b* and *c* of the figure show a circuit symbol and truth table for the full adder, which produces the two-bit binary sum $c_os = a + b + c_i$. Figure 2*d* shows how four instances of this full adder module can be used to design a circuit that adds two four-bit numbers. This type of circuit is usually called a *ripple-carry* adder, because of the way that the carry signals are passed from one full adder to the next. Write a Verilog code that implements this circuit, as described below.



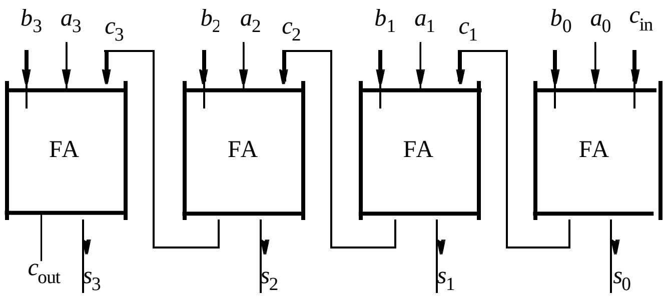
a) Full adder circuit



b) Full adder symbol

b	a	c_i	c_o	s
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

c) Full adder truth table



d) Four-bit ripple-carry adder circuit

Figure 2. A ripple-carry adder circuit.

1. Create a new Vivado project for the adder circuit. Write a Verilog module for the full adder subcircuit and write a top-level Verilog module that instantiates four instances of this full adder.

2. Use switches SW_{7-4} and SW_{3-0} to represent the inputs A and B, respectively. Use SW_8 for the carry-in c_{in} of the adder. Connect the *SW* switches to their corresponding red lights LEDR, and connect the outputs of the adder, c_{out} and S, to the green lights LEDG.

3. Include the necessary pin assignments for the NEXYS 4 DDR board, compile the circuit, and download it into the FPGA chip.

4. Test your circuit by trying different values for numbers A, B, and c_{in}.