COL202: Digital Logic and System Design

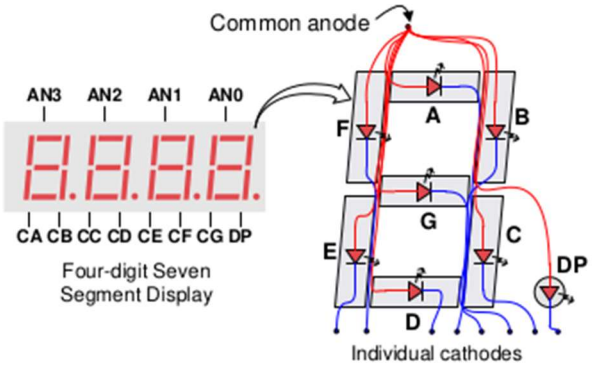
Hardware Assignment - 1
Seven Segment Display

Yash Bansal : (2022CS51133)
Soumyaprabha Dey : (2022CS11107)
Anand Sasikumar : (2022CS11087)

PROBLEM DESCRIPTION

Design and implement a circuit that takes a 4-bit binary equivalent of decimal /hexadecimal number as input from the switches of the Basys3 FPGA board and produces a 7-bit output to display it on one of the 4-digit 7-segment displays on the board by using elementary operations like AND, OR and NOT.

THEORY- (Seven Segment Decoder)



F Pin details for 7-segment display on Basys3 board

- As seen from Fig, every 7-segment display has 7 LEDs named A, B, C, D, E, F and G. They share a common anode but different cathodes. For the four 7-segment displays, there are four anodes by the name AN0, AN1, AN2 and AN3. We must drive the corresponding common anode HIGH and the cathode LOW to turn on an LED. The decimal point has a corresponding LED, which is irrelevant to this problem. For example, if we want to display 4 (corresponding to 0100 given as input)on the left-most 7-segment display, then AN0 should be driven HIGH with all other anodes LOW. Also, B, C, F, and G should be driven LOW and the rest HIGH.
- However, since the BASYS3 uses transistors to drive enough current into the common anode point, the anode enables are inverted. That means if we want to drive a particular anode at HIGH, then that anode should have a LOW(or 0) value.
- So, we will design a combinational logic circuit for which the inputs are the four bits binary equivalent of the given hexadecimal number (the more general case) and outputs correspond to the 7 cathode pins to display the

number on the left most 7-segment display. This is precisely a 7-segment decoder.

DESIGN DECISIONS:

INPUTS				OUTPUTS(SEGMENTS)						
A	**B**	**C**	**D**	**p**	**q**	**r**	**s**	**t**	**u**	**v**
0	0	0	0	1	1	1	1	1	1	0
0	0	0	1	0	1	1	0	0	0	0
0	0	1	0	1	1	0	1	1	0	1
0	0	1	1	1	1	1	1	0	0	1
0	1	0	0	0	1	1	0	0	1	1
0	1	0	1	1	0	1	1	0	1	1
0	1	1	0	1	0	1	1	1	1	1
0	1	1	1	1	1	1	0	0	0	0
1	0	0	0	1	1	1	1	1	1	1
1	0	0	1	1	1	1	1	0	1	1
1	0	1	0	1	1	1	0	1	1	1
1	0	1	1	0	0	1	1	1	1	1
1	1	0	0	1	0	0	1	1	1	0
1	1	0	1	0	1	1	1	1	0	1
1	1	1	0	1	0	0	1	1	1	1
1	1	1	1	1	0	0	0	1	1	1

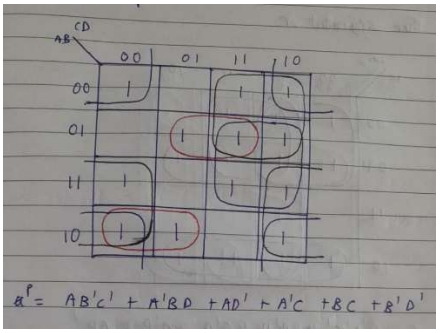
Truth Table

Now, for writing the logic statements for all the seven segments in terms of inputs A, B, C and D, we have used the method of Karnaugh maps.

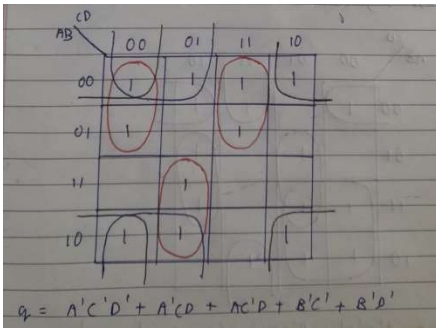
- Since 0 was considered as on and one as off so, we have to invert the values we got from the k-map.

- an0 was given a value of 0 as it was operational, and others were given a value of 1 because they were not.

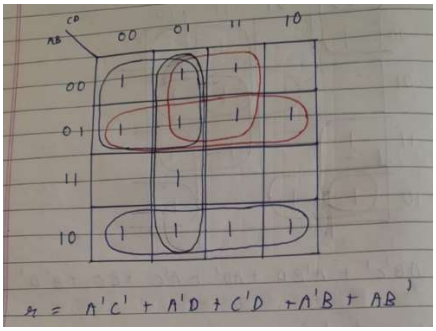
The Karnaugh maps and their corresponding logics are as follows:-


CD
AB
00 01 11 10
p = AB'C' + A'BD + AD' + A'C + BC + B'D'

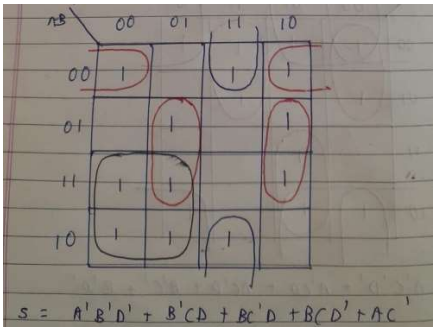
p


CD
AB
00 01 11 10
q = A'C'D' + A'CD + AC'D + B'C' + B'D'

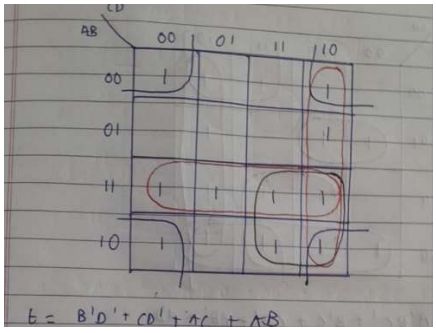
q


CD
AB
00 01 11 10
r = A'C' + A'D + C'D + A'B + AB'

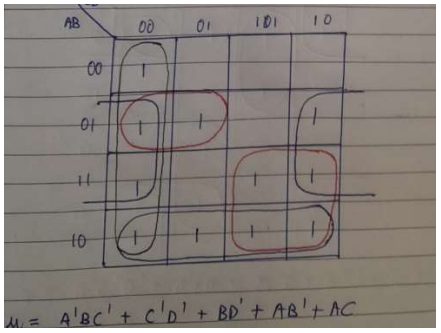
r


AB
00 01 11 10
s = A'B'D' + B'CD + BC'D + BCD' + AC'

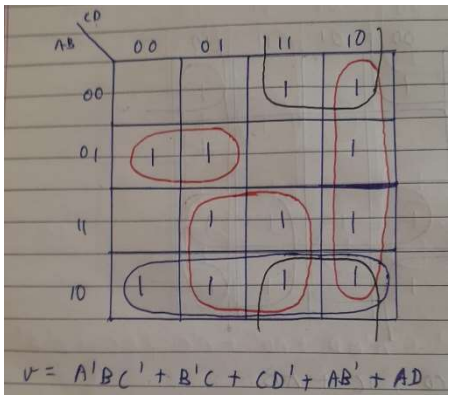
s


CD
AB
00 01 11 10
t = B'D' + CD' + AC + AB

t


AB
00 01 11 10
u = A'BC' + C'D' + BD' + AB' + AC

u

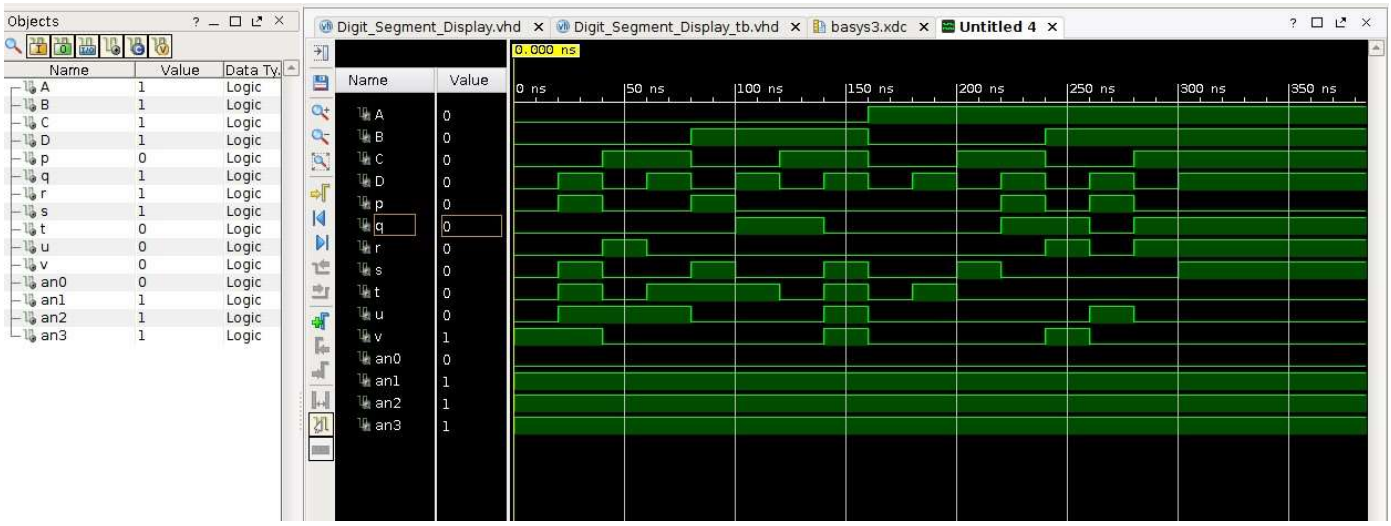

CD
AB
00 01 11 10
v = A'BC' + B'C + CD' + AB' + AD

v

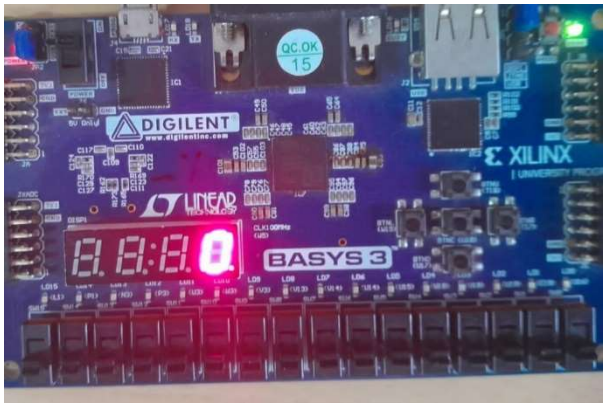
LAB WORK

1. We opened Vivado, created a new project to develop a source file(Digit_Segment_Display.vhd), added a constraint file basys3.xdc, and selected the part number xc7a35tcpg236-1.
2. Source code:-
   We updated the source file as uploaded with this report.
3. Simulation of the Design:-
   For the design simulation, we created a testbench file named Digit_Segment_display_tb.vhd as uploaded.
   After the simulation is completed, this simulation file is disabled.
4. Constraint file
   We updated the constraint file as uploaded so that the inputs are taken, and output is displayed as planned in the design.
5. Synthesis of design
   We click on Run Synthesis, then on Run Implementation, Generate the Bitstream, Auto connect the BASYS3 board and program the device.
   Now, we can give inputs to the BASYS3 board with switches and get the output on the seven segment display

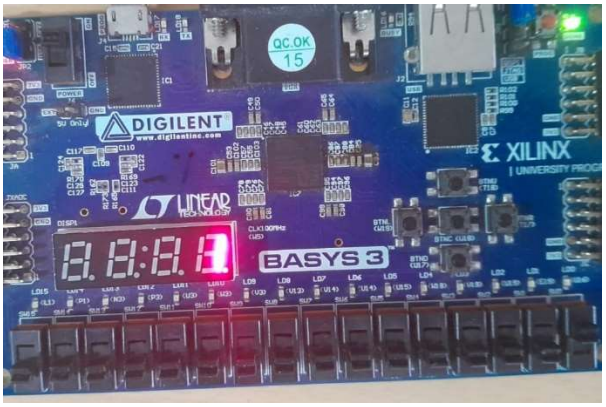
Simulation Snapshots



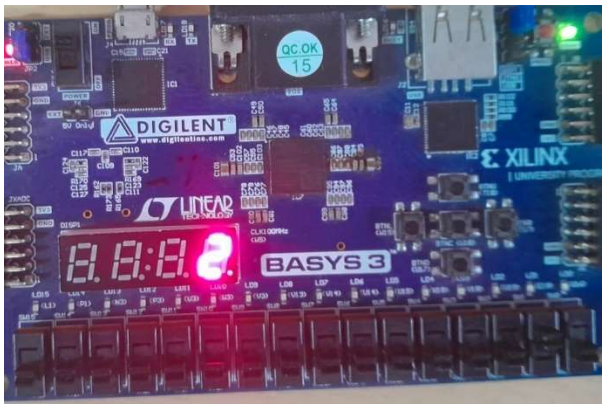
On running the simulation, we got this output form, which matches the expected output.



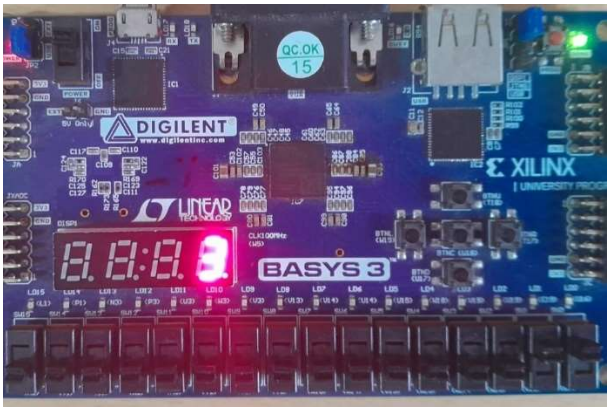
0000



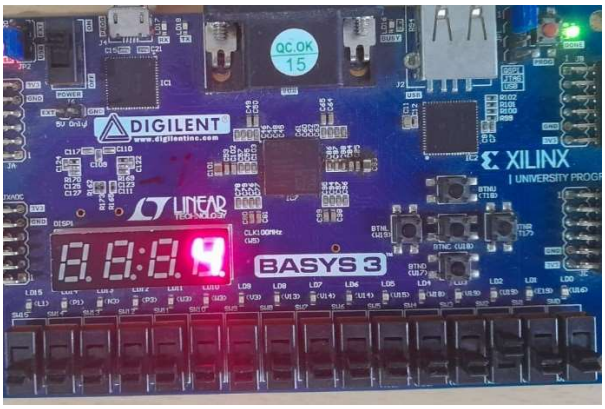
0001



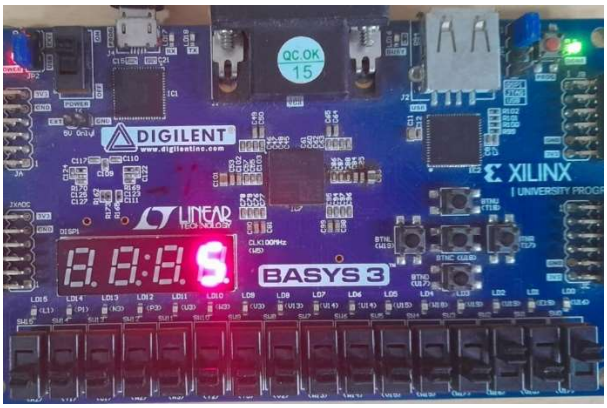
0010



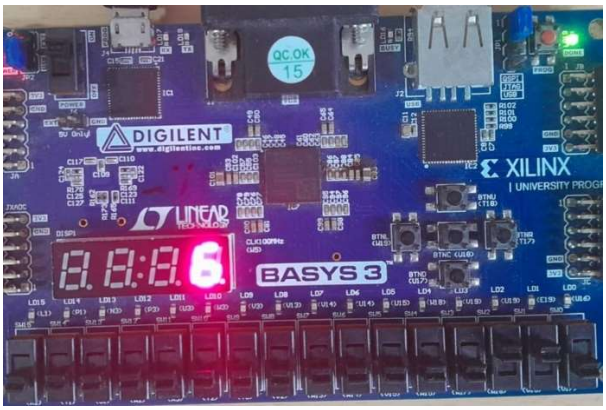
0011



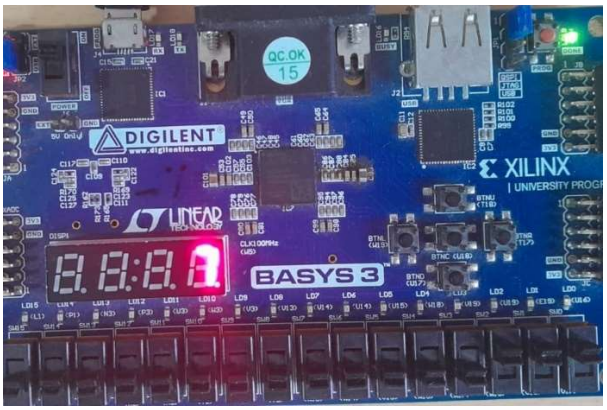
0100



0101



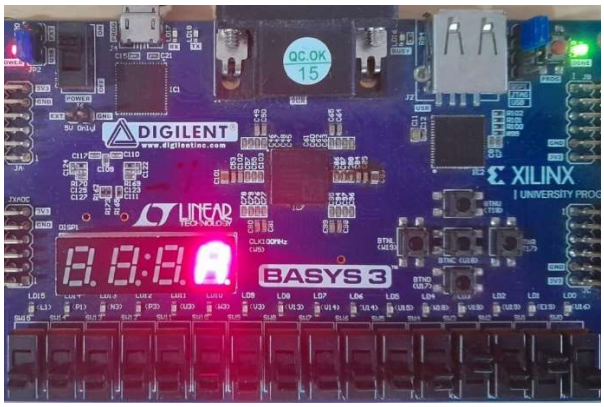
0110



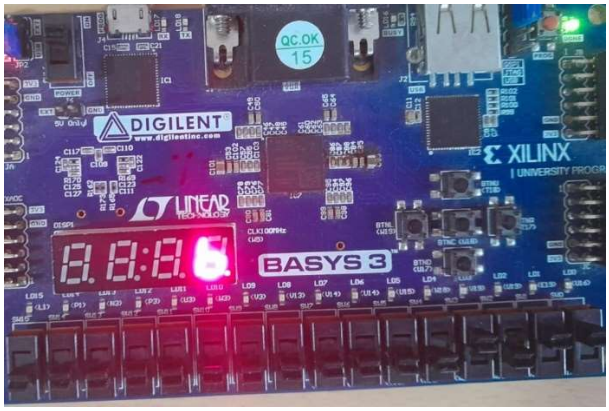
0111

1000

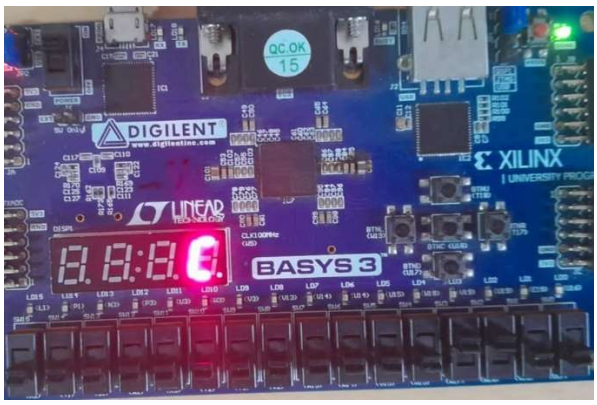
1001



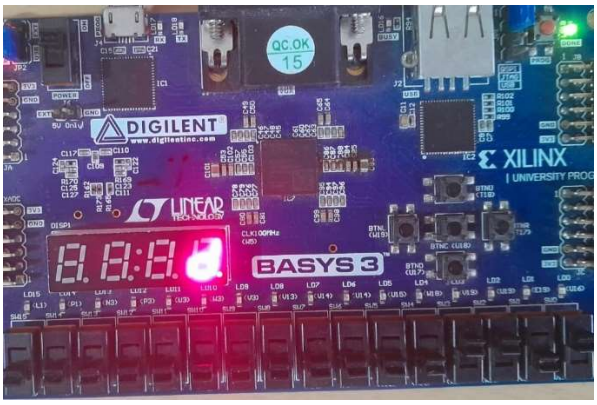
1010



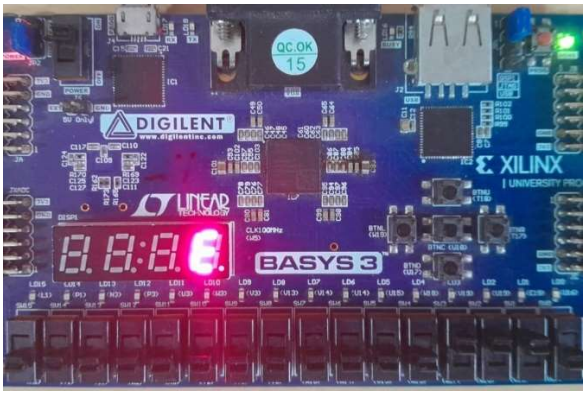
1011



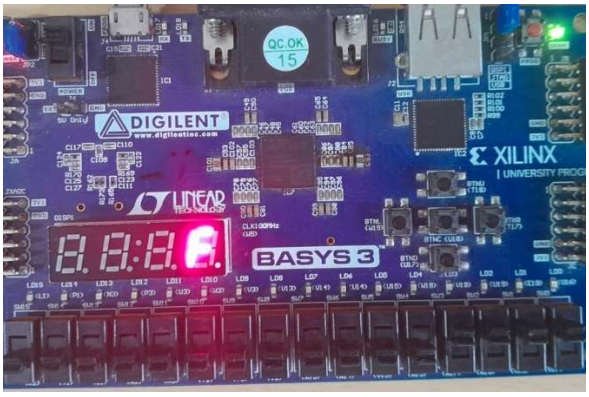
1100



1101



1110



1111