# COL215 LAB Assignment 2 : Seven-Segment Display Logic

Mohit Sharma(2019CS10372), Arka Mandal(2019CS50617)

## 1 Aim

**Design a combination circuit that takes a decimal/hexadecimal digit encoded using 4 bits and produces 7-bit output for seven segment displays of BASYS3 FPGA board. Do extensive simulation of the design using Xilinx simulator, and then implement the circuit on BASYS-3 FPGA board.**

## 2 Combinational Logic Design

We have for input bit representing a hexadecimal number between 0 to f, and we need to map them to corresponding display symbol through 7 output bits of seven-segment display.

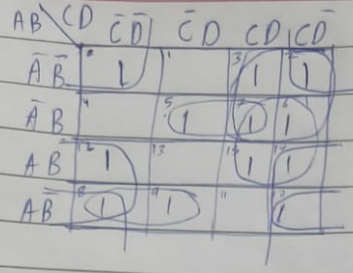
| A(sw[3]) | B(sw[2]) | C(sw[1]) | D(sw[0]) | a | b | c | d | e | f | g | Display |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 2 |
| 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 3 |
| 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 4 |
| 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 5 |
| 0 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 6 |
| 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 7 |
| 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 8 |
| 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 9 |
| 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | A |
| 1 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | b |
| 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | C |
| 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | d |
| 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | E |
| 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | F |

From above mapping we created k-map for each segment to get the combinational logic equations.

As for basys board we need to set output bit 0 to glow the corresponding segment and 1 to not glow it, hence we have set :
seg[0] = not a;
seg[1] = not b;
seg[2] = not c;
seg[3] = not d;
seg[4] = not e;
seg[5] = not f;
seg[6] = not g;

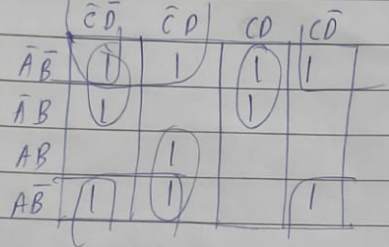
Also anode bit corresponding to rightmost display is set to 0 and other to 1 so that only rightmost display glows.

(a)



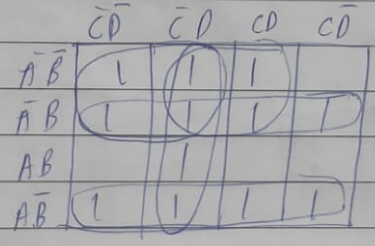

$a = A\bar{B}\bar{C} + \bar{A}BD + A\bar{D} + \bar{A}\bar{C} + BC + \bar{B}\bar{D}$

(b)



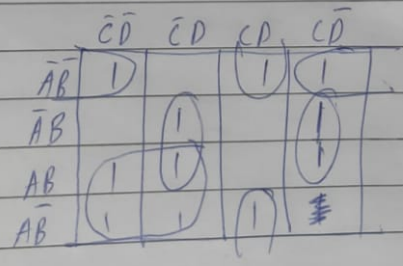

$b = \bar{A}\bar{C}\bar{D} + \bar{A}CD + A\bar{C}D + \bar{B}\bar{C} + \bar{B}\bar{D}$

(c)

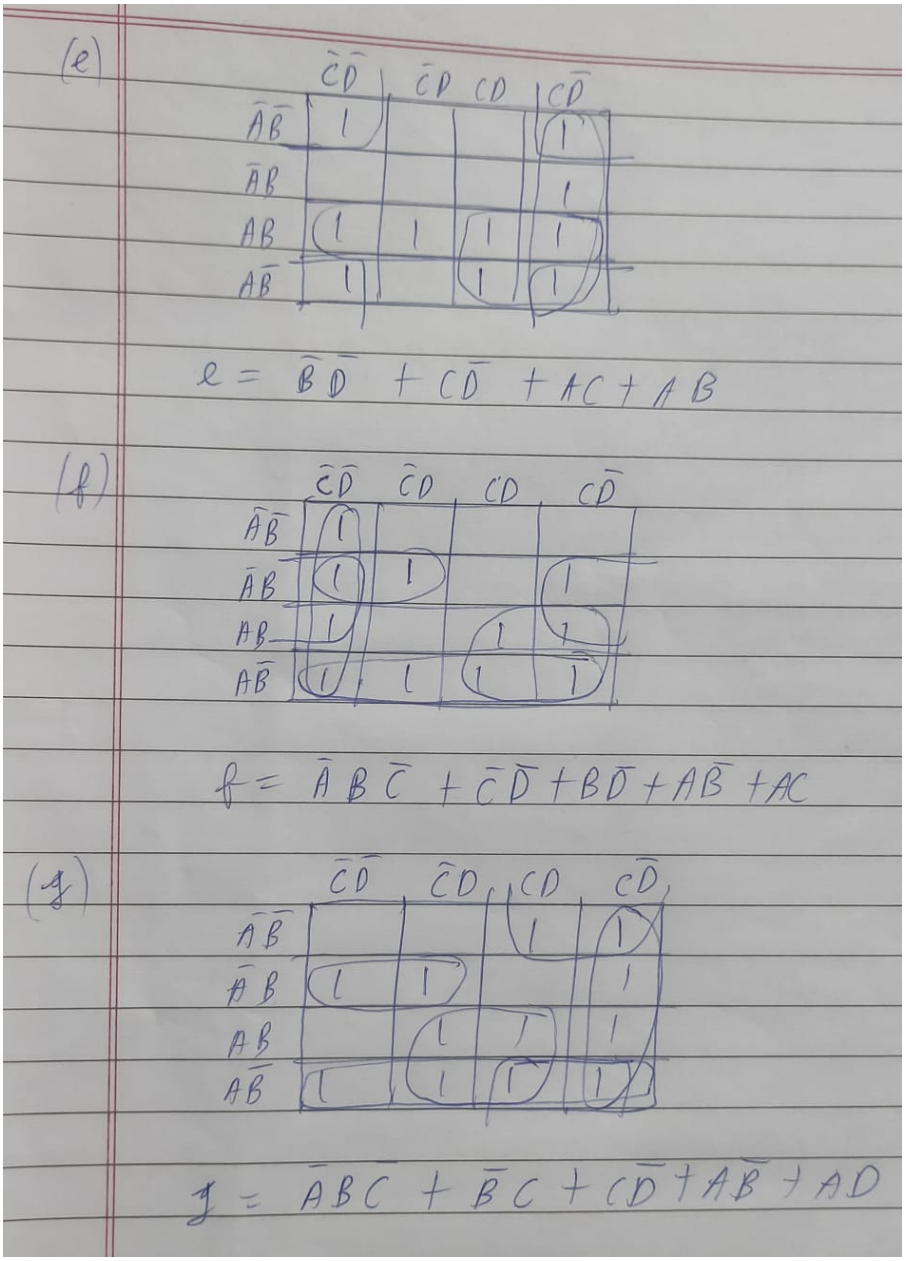



$c = \bar{A}\bar{C} + \bar{A}D + \bar{C}D + \bar{A}B + A\bar{B}$

(d)

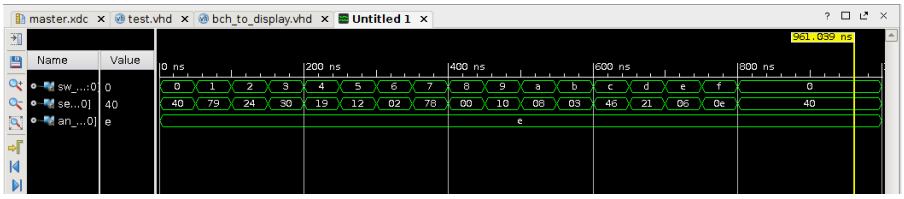



$d = \bar{A}\bar{B}\bar{D} + \bar{B}CD + BC\bar{D} + B\bar{C}\bar{D} + A\bar{C}$




## 3 Simulation

The code is simulated on test-bench created by the name test.vhd.




# 4 Utilisation Report

Utilisation report is also given in the submission.

4 LUT's are used

15 Input Output Blocks are used (4 switch inputs(sw) and 7 segment outputs(seg) and 4 display outputs(an))

# 5 Extra

We earlier tried to write the vhdl code using switch case as:

```
process(sw) is
begin
    case sw is
        when "0000" => seg <="1000000"; --0
        when "0001" => seg <="1111001"; --1
        when "0010" => seg <="0100100"; --2
        when "0011" => seg <="0110000"; --3
        when "0100" => seg <="0011001"; --4
        when "0101" => seg <="0010010"; --5
        when "0110" => seg <="0000010"; --6
        when "0111" => seg <="1111000"; --7
        when "1000" => seg <="0000000"; --8
        when "1001" => seg <="0010000"; --9
        when "1010" => seg <="0001000"; --a
        when "1011" => seg <="0000011"; --b
        when "1100" => seg <="1000110"; --c
        when "1101" => seg <="0100001"; --d
        when "1110" => seg <="0000110"; --e
        when "1111" => seg <="0001110"; --f
        when others => seg <="0111111";
     end case;
     an<="1110";
 end process;
```

Though this is run sequentially but the resource utilisation is same as combinational circuit.