Seven-Segment Display

Seven-segment displays are commonly used as alphanumeric displays by logic and computer systems. A seven segment display is an arrangement of 7 LEDs (see below) that can be used to show any hex number between 0000 and 1111 by illuminating combinations of these LEDs. For example, the red digits on a digital clock use 2-segment LED displays. 7-segment displays come in two flavors: common anode and common cathode. A common anode 7-segment display has all of the anodes tied together while a common cathode 7-segment display has all the cathodes tied together.

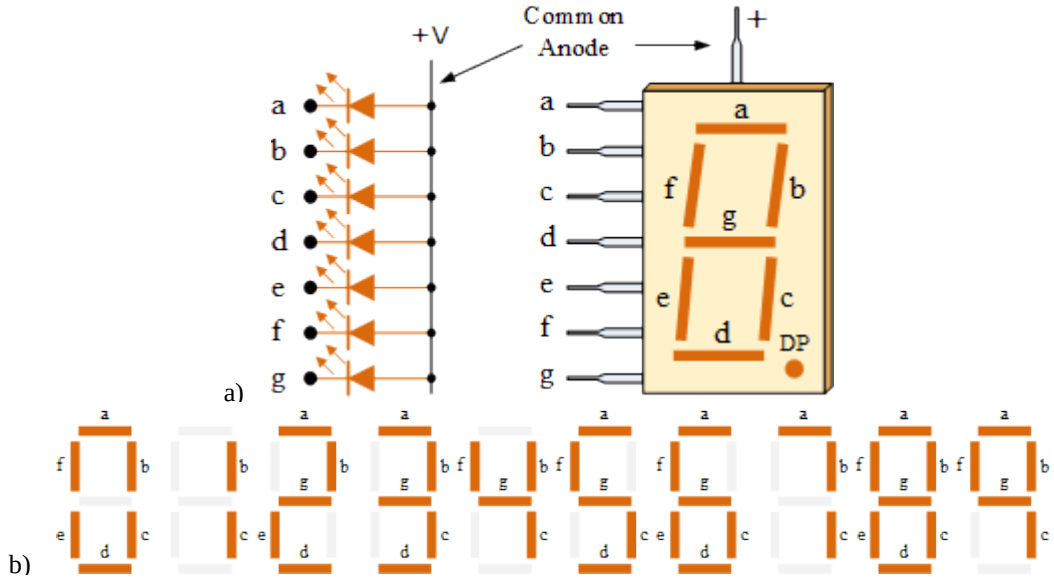

Figure 1 a) Common Anode 7-Segment Display b) 7-Segments for all Numbers

1. Lab Description

Part A

Students are asked to design a binary to 7-segment decoder. Use the HDL template given with this lab to finish your lab assignment.

Fill the truth table

SA	SB	SC	SD	A	B	C	D	E	F	G
0	0	0	0	0	0	0	0	0	0	1
0	0	0	1	1	0	0	1	1	1	1
0	0	1	0	0	0	1	0	0	1	0
0	0	1	1	0	0	0	0	1	1	0
0	1	0	0	1	0	0	1	1	0	0
0	1	0	1	0	1	0	0	1	0	0
0	1	1	0	0	1	0	0	0	0	0
0	1	1	1	0	0	0	1	1	1	1
1	0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	1	1	0	0
1	0	1	0	X	X	X	X	X	X	X
1	0	1	1	X	X	X	X	X	X	X
1	1	0	0	X	X	X	X	X	X	X
1	1	0	1	X	X	X	X	X	X	X
1	1	1	0	X	X	X	X	X	X	X
1	1	1	1	X	X	X	X	X	X	X

Note: Logic '0' turns on the LED in the 7-segment display and Logic '1' turns off the LED in the 7-segment display.

In the above the truth tables 'X' means the output is "don't care".

xDrive the appropriate Boolean expression using the K-maps for A, B, C, D, E, F, and G.

Implement the binary to seven-segment display function decoder using HDL for each output LED A, B, C, D, E, F, and G.

Write your answers from here.

K-MAPS:

A = BC'D + A'B'C'D

B = BC'D + BCD'

C = B'CD'

D = B'C'D + CB'D' + BCD

E = D + BC'

F = B'C + CD + A'B'D

G = A'B'C' + BCD

Decoder Code:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity test is
port (
    clk : in std_logic;
     bcd : in std_logic_vector(3 downto 0);  --BCD input
     segment7 : out std_logic_vector(6 downto 0)  -- 7 bit decoded output.
  );
end test;
--'a' corresponds to MSB of segment7 and g corresponds to LSB of segment7.
architecture Behavioral of test is

begin
```

```
process (clk,bcd)
BEGIN
if (clk'event and clk='1') then
case  bcd is
when "0000"=> segment7 <="0000001";  -- '0'
when "0001"=> segment7 <="1001111";  -- '1'
when "0010"=> segment7 <="0010010";  -- '2'
when "0011"=> segment7 <="0000110";  -- '3'
when "0100"=> segment7 <="1001100";  -- '4'
when "0101"=> segment7 <="0100100";  -- '5'
when "0110"=> segment7 <="0100000";  -- '6'
when "0111"=> segment7 <="0001111";  -- '7'
when "1000"=> segment7 <="0000000";  -- '8'
when "1001"=> segment7 <="0000100";  -- '9'
 --nothing is displayed when a number more than 9 is given as input.
when others=> segment7 <="1111111";
end case;
end if;

end process;

end Behavioral;
```

Test Bench:

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;

ENTITY test_tb IS
END test_tb;

ARCHITECTURE behavior OF test_tb IS
  signal clk : std_logic := '0';
  signal bcd : std_logic_vector(3 downto 0) := (others => '0');
  signal segment7 : std_logic_vector(6 downto 0);
  constant clk_period : time := 1 ns;
BEGIN
  uut: entity work.test PORT MAP (clk,bcd,segment7);
  clk_process :process
```

```
  begin
            clk <= '0';
           wait for clk_period/2;
            clk <= '1';
            wait for clk_period/2;
  end process;
  stim_proc: process
  begin
   for i in 0 to 9 loop
        bcd <= conv_std_logic_vector(i,4);
       wait for 2 ns;
   end loop;
  end process;

END;
```