Solution: TP2

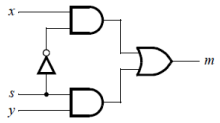
Part I: Switches and Lights

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY SwLeD IS
PORT(SW : IN STD_LOGIC_VECTOR(17 DOWNTO 0);
LEDR: OUT STD_LOGIC_VECTOR(17 DOWNTO 0));
END SwLeD;

ARCHITECTURE LALA OF SwleD IS
BEGIN
LEDR <=SW;
END LALA;
```

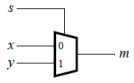
Part II: Multiplexers



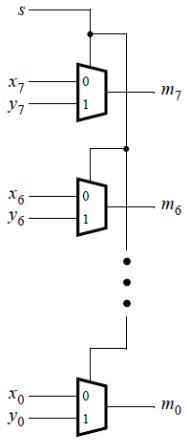
a) Circuit

s	m
0	x
1	y

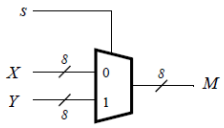
b) Truth table



c) Symbol



a) Circuit



b) Symbol

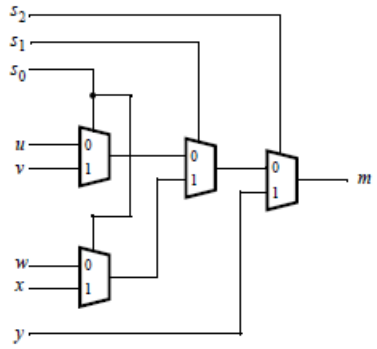
```
library ieee;
use ieee.std_logic_1164.all;
entity Mux2to1 is
port( x: in std_logic_vector(7 downto 0);
                    y: in std_logic_vector(7 downto 0);
                    s: in std_logic;
                    LEDR: OUT std_logic_vector(15 downto 0);
                    m : out std_logic_vector(7 downto 0));
end Mux2to1;
architecture behaviore of Mux2to1 is

begin

m(0) <= ((not(s) and x(0)) or (y(0) and s));
m(1) <= ((not(s) and x(1)) or (y(1) and s));
m(2) <= ((not(s) and x(2)) or (y(2) and s));
m(3) <= ((not(s) and x(3)) or (y(3) and s));
m(4) <= ((not(s) and x(4)) or (y(4) and s));
m(5) <= ((not(s) and x(5)) or (y(5) and s));
m(6) <= ((not(s) and x(6)) or (y(6) and s));
m(7) <= ((not(s) and x(7)) or (y(7) and s));
LEDR<=y&x;

--LEDR(0)<=x(0);
--LEDR(1)<=x(1);
--LEDR(2)<=x(2);
--LEDR(3)<=x(3);
--LEDR(4)<=x(4);
--LEDR(5)<=x(5);
--LEDR(6)<=x(6);
--LEDR(7)<=x(7);
--LEDR(8)<=y(0);
--LEDR(9)<=y(1);
--LEDR(10)<=y(2);
--LEDR(11)<=y(3);
--LEDR(12)<=y(4);
--LEDR(13)<=y(5);
--LEDR(14)<=y(6);
--LEDR(15)<=y(7);
end behaviore;
```

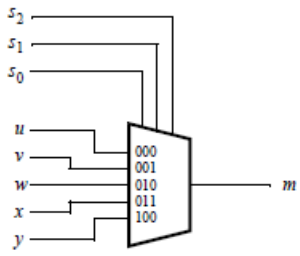
Part III: Multiplexers



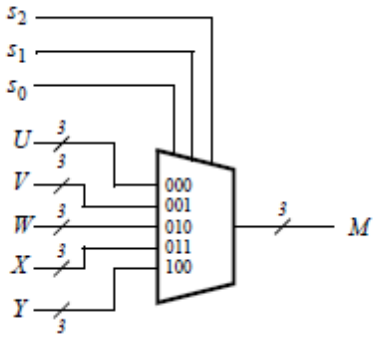
a) Circuit

s_2	s_1	s_0	m
0	0	0	u
0	0	1	v
0	1	0	w
0	1	1	x
1	0	0	y
1	0	1	y
1	1	0	y
1	1	1	y

b) Truth table



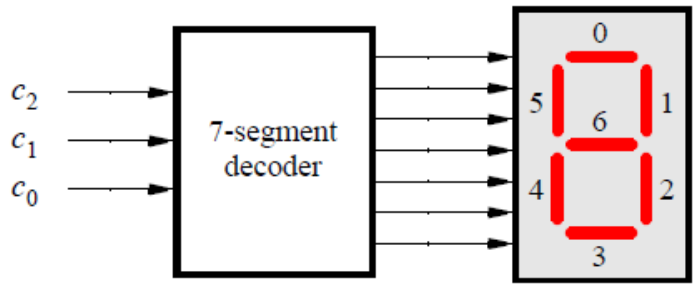
c) Symbol




```
library ieee;
use ieee.std_logic_1164.all;
entity Mux5to1 is
port( u: in std_logic_vector(2 downto 0);
                v: in std_logic_vector(2 downto 0);
                w: in std_logic_vector(2 downto 0);
                x: in std_logic_vector(2 downto 0);
                y: in std_logic_vector(2 downto 0);
                LEDR: OUT std_logic_vector(14 downto 0);
                s0: in std_logic;
                s1: in std_logic;
                s2: in std_logic;
                m : out std_logic_vector(2 downto 0));
```

```
end Mux5to1;
architecture behaviore of Mux5to1 is
begin
m<=u when (S2='0' and S1='0' and S0='0') else
        v when (S2='0' and S1='0' and S0='1') else
        w when (S2='0' and S1='1' and S0='0') else
        x when (S2='0' and S1='1' and S0='1') else
        y when (S2='1' and S1='0' and S0='0') else
        y;
        LEDR<=y&x&w&v&u;
end behaviore;
```

Part IV: 7-segment decoder



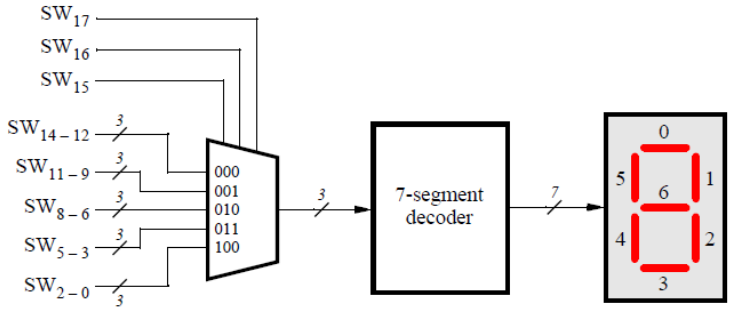
$c_2c_1c_0$	Character
000	H
001	E
010	L
011	O
100	
101	
110	
111	

Table 1. Character codes.

```
--7segemnts est a l'etat bas
LIBRARY ieee;
Use ieee.std_logic_1164.all;
entity segdisplay is
port ( VALUE : in bit_vector(2 downto 0);
DISPLAY : out bit_vector(6 downto 0));
end segdisplay;
architecture BEHAVIOUR of segdisplay is
begin
DISPLAY <= "0001001" when VALUE<="000" else
                        "0000110" when VALUE<= "001" else
```

```
                    "1000111" when VALUE<="010" else
                    "1000000" when VALUE<= "011" else
                    "1111111" when VALUE<="100" else
        "1111111" when VALUE<="101" else
        "1111111" when VALUE<= "110" else
        "1111111" when VALUE<= "111" else
        "1111111";
end BEHAVIOUR;
```

Part V: 7-segment decoder




```
LIBRARY ieee;
Use ieee.std_logic_1164.all;
entity decodseg is
port(sw0,sw1,sw2,sw3,sw4: in std_logic_vector(2 downto 0);
sw5,sw6,sw7: in std_logic;
sortie: out std_logic_vector(6 downto 0));
end decodseg;

architecture BEHAVIOUR of decodseg is
SIGNAL M : STD_LOGIC_VECTOR(2 DOWNTO 0);

COMPONENT Mux5to1 PORT (
    u: in std_logic_vector(2 downto 0);
                v: in std_logic_vector(2 downto 0);
                w: in std_logic_vector(2 downto 0);
                x: in std_logic_vector(2 downto 0);
                y: in std_logic_vector(2 downto 0);
                s0: in std_logic;
                s1: in std_logic;
                s2: in std_logic;
                m : out std_logic_vector(2 downto 0));
END COMPONENT;

COMPONENT segdisplay PORT ( VALUE1 : in std_logic_vector(2 downto 0);
```

```
DISPLAY : out std_logic_vector(6 downto 0));
END COMPONENT;

Begin
M0: Mux5to1 PORT MAP (sw0,sw1,sw2,sw3,sw4,sw5,sw6,sw7, M);
H0: segdisplay PORT MAP (M, sortie);
END BEHAVIOUR;
```

SW_{17} SW_{16} SW_{15}	Character pattern				
000	H	E	L	L	O
001	E	L	L	O	H
010	L	L	O	H	E
011	L	O	H	E	L
100	O	H	E	L	L

Table 2. Rotating the word HELLO on five displays.