Elettronica dei Sistemi Digitali
Lab 02

Switches, Decoders, Numbers and Displays

Gruppo: A11

Pronesti, Massimiliano	S245831
Oropallo, Maria Vittoria	S245999
Nicolicchia, Riccardo	S244728
Miceli, Michelangelo	S245478

2 aprile 2020

Indice

Capitolo 1

Controlling a 7-segments display

1.1 Spiegazione teorica

In questo primo esercizio, ci proponiamo di sintetizzare e simulare un decoder capace di pilotare un display a 7 segmenti come quello mostrato in **figura 1.1** al fine di visualizzare delle lettere come in

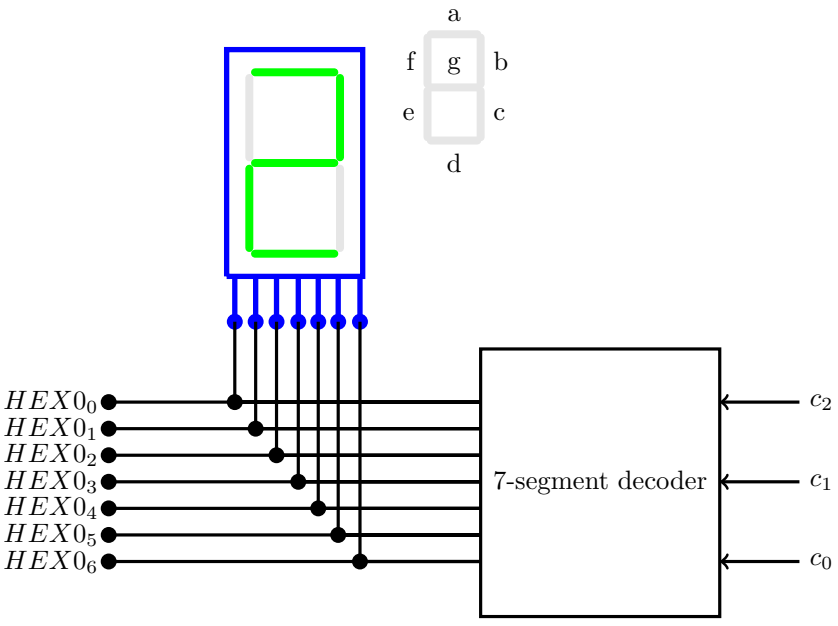


Figura 1.1: Seven segment decoder

Figura 1.2: Output for the four relevant entries

figura 1.2 secondo la tavola di verità in **figura 1.3**.

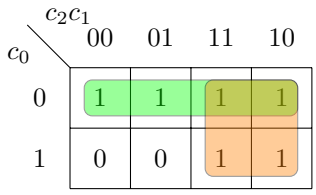
1.2 Procedimento

Poiché non è possibile utilizzare, direttamente, la sintassi del linguaggio VHDL per eseguire una casistica, si è sintetizzata la tavola di verità mostrata in **figura 1.3** manualmente, mediante mappe di Karnaugh.

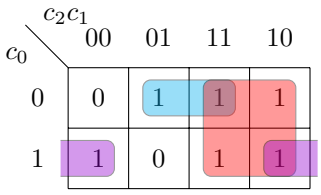
c_2	c_1	c_0	a	b	c	d	e	f	g
0	0	0	1	0	0	1	0	0	0
0	0	1	0	1	1	0	0	0	0
0	1	0	1	1	1	0	0	0	1
0	1	1	0	0	0	0	0	0	1
1	0	0	1	1	1	1	1	1	1
1	0	1	1	1	1	1	1	1	1
1	1	0	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1

Figura 1.3: 7-segment truth table

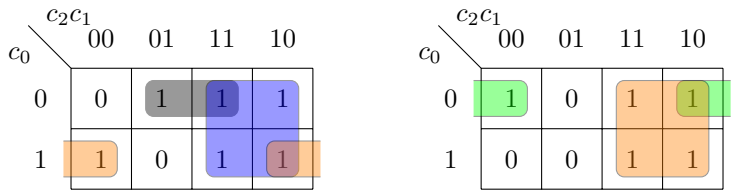
Di seguito sono riportate le K-maps relative alle 7 uscite, laddove si è tenuto conto del fatto che i segmenti si accendono in corrispondenza di uno '0' .



(a)

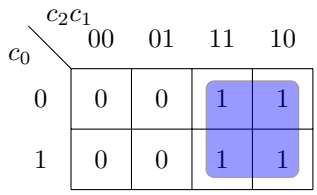


(b)

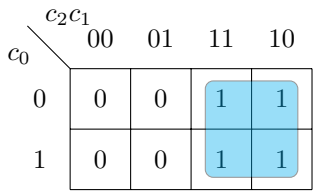


(c)

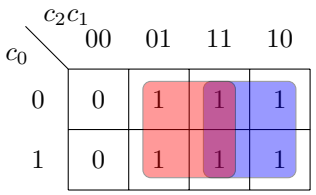
(d)



(e)



(f)



(g)

Figura 1.4: Karnaugh maps

Le funzioni booleane risultanti, dopo aver opportunamente usato i teoremi di DeMorgan ove vantaggiosi, sono mostrate in **Code 1.1**, in appendice.
Si noti che si è assunto spento il display per tutte le altre combinazioni.

1.3 Risultati

Di seguito sono riportate gli output prodotti da Wave ed RTL Viewer di Modelsim.

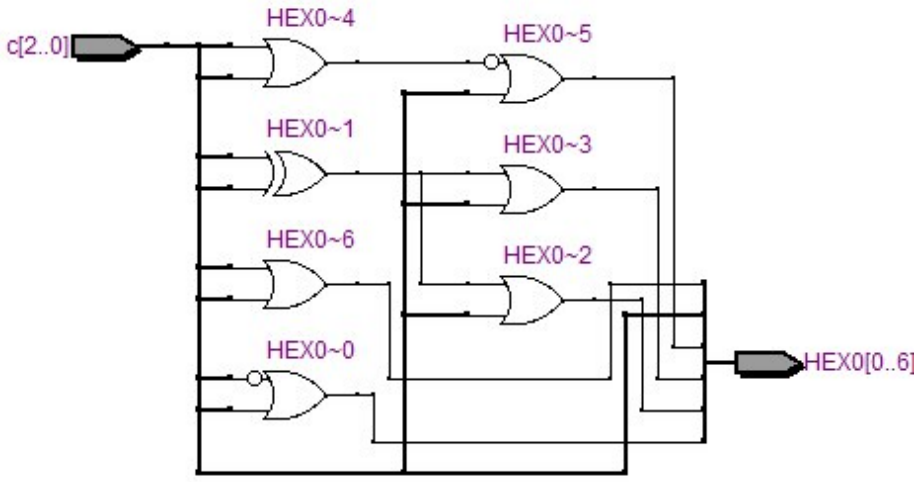


Figura 1.5: RTL viewer exercise no. 1

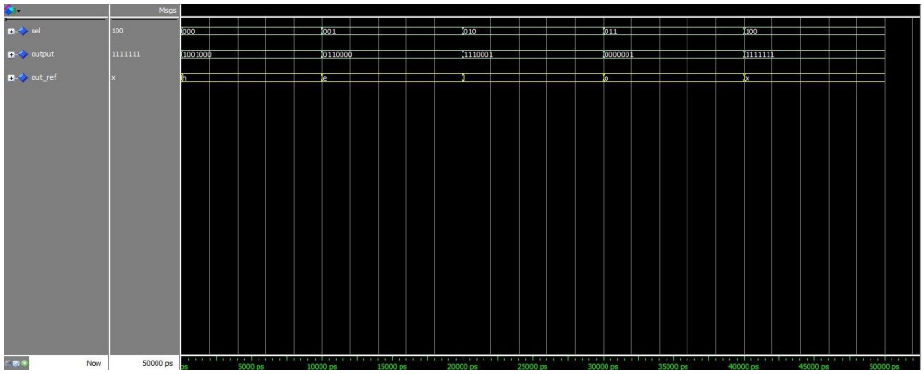


Figura 1.6: wave exercise no. 1

1.4 Appendice

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3
4  entity decoder is
5     port (
6        SW : in std_logic_vector(2 downto 0);
7        HEX0 : out std_logic_vector(0 to 6)
```

```
8     );
9  end decoder;
10
11 architecture Behavior of decoder is
12 begin
13    -- boolean functions from K-map
14    --light switches on at driven value '0'
15    HEX0(0) <= SW(2) or not SW(0);
16    HEX0(1) <= SW(2) or (SW(0) xor SW(1));
17    HEX0(2) <= SW(2) or (SW(0) xor SW(1));
18    HEX0(3) <= SW(2) or not(SW(0) or SW(1));
19    HEX0(4) <= SW(2);
20    HEX0(5) <= SW(2);
21    HEX0(6) <= SW(1) or SW(2);
22
23 end Behavior;
```

Code 1.1: VHDL and synth code exercise no. 1

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3
4  entity decoder_tb is
5  end decoder_tb;
6
7  architecture test of decoder_tb is
8
9     signal sel : std_logic_vector(2 downto 0);
10    signal output : std_logic_vector(0 to 6);
11
12    component decoder
13       port (
14          SW : in std_logic_vector(2 downto 0);
15          HEX0 : out std_logic_vector(0 to 6)
16       );
17    end component;
18
19 begin
20    --assigned values to c2,c1,c0
21    sel <= "000", "001" after 10 ns, "010" after 20 ns, "011" after 30
          ns, "100" after 40 ns;
22    DUT : decoder port map(SW => sel, HEX0 => output);
23
24 end test;
```

Code 1.2: testbench code exercise no. 1

Capitolo 2

Multiplexing the 7-segments display output

2.1 Spiegazione teorica

Il secondo esercizio ha previsto l'estensione dei risultati conseguiti nel precedente, introducendo un multiplexer a 5 vie di parallelismo 3 per pilotare il decoder, secondo la tavola mostrata in **figura 2.1**, dove la specifica sequenza di caratteri è solo a titolo d'esempio.

SW_{17}	SW_{16}	SW_{15}	pattern
0	0	0	H E L L O
0	0	1	E L L O H
0	1	0	L L O H E
0	1	1	L O H E L
1	0	0	O H E L L

Figura 2.1: 7-segment truth table

2.2 Procedimento

Si sono opportunamente instanziati 5 sotto-circuiti analoghi a quello fornito nel testo dell'esercitazione dimodoché alle combinazioni dei tre selettori corrispondessero, "circolarmente" quelle desiderate per il display a 7 segmenti, come mostrato in **Code 2.1**. Si noti che il codice sviluppato per gli esercizi 2 e 3 della prima esperienza laboratoriale è stato riutilizzato.

2.3 Risultati

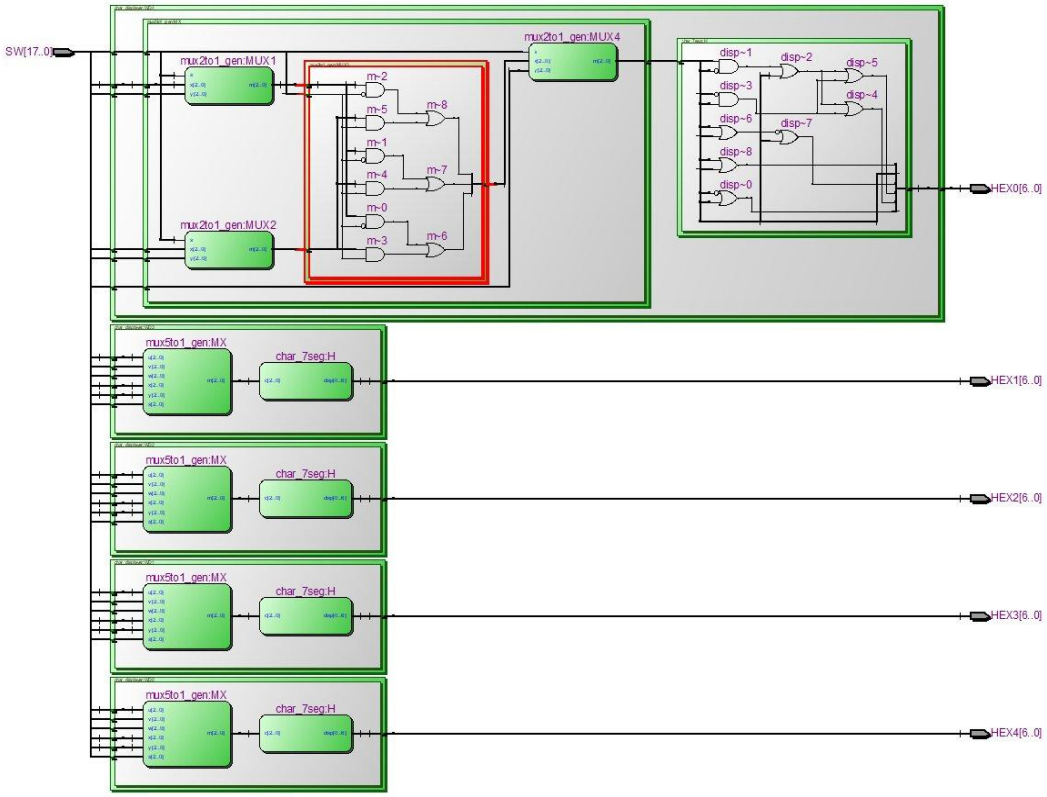


Figura 2.2: RTL view of exercise no. 2

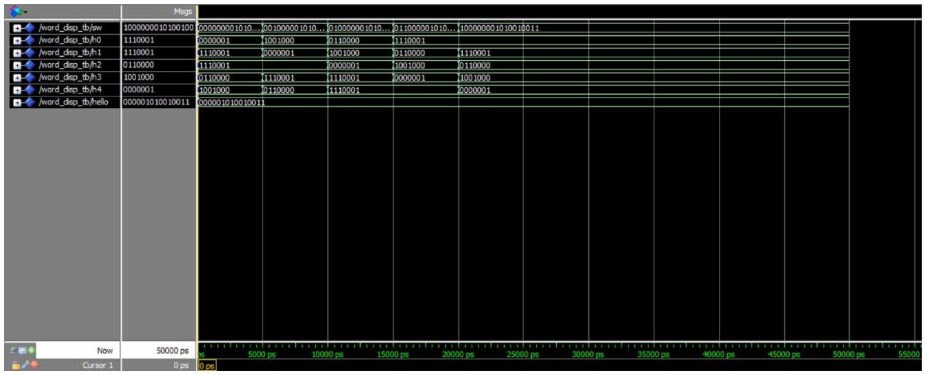


Figura 2.3: wave exercise no. 2

2.4 Appendice

```vhdl
library ieee;
use ieee.std_logic_1164.all;
entity word_disp is
   port (
      SW : in std_logic_vector(17 downto 0);
      HEX0 : out std_logic_vector(6 downto 0);
      HEX1 : out std_logic_vector(6 downto 0);
      HEX2 : out std_logic_vector(6 downto 0);
      HEX3 : out std_logic_vector(6 downto 0);
      HEX4 : out std_logic_vector(6 downto 0)
   );
end word_disp;
architecture Behavior of word_disp is

   component char_displayer
      port (
         sel, CH0, CH1, CH2, CH3, CH4 : in std_logic_vector (2 downto 0);
         HEX : out std_logic_vector (6 downto 0)
      );
   end component;

   signal sel, C0, C1, C2, C3, C4 : std_logic_vector(2 downto 0);

begin
   sel <= SW(17 downto 15);
   C0 <= SW(14 downto 12); -- first char
   C1 <= SW(11 downto 9); -- second char
   C2 <= SW (8 downto 6); -- thir chart
   C3 <= SW(5 downto 3); -- fourth char
   C4 <= SW(2 downto 0); -- fifth char

   WD0 : char_displayer port map(sel, C0, C1, C2, C3, C4, HEX4);
   WD1 : char_displayer port map(sel, C1, C2, C3, C4, C0, HEX3);
   WD2 : char_displayer port map(sel, C2, C3, C4, C0, C1, HEX2);
   WD3 : char_displayer port map(sel, C3, C4, C0, C1, C2, HEX1);
   WD4 : char_displayer port map(sel, C4, C0, C1, C2, C3, HEX0);

end Behavior;
```

Code 2.1: word_disp.vhd

```vhdl
library ieee;
use ieee.std_logic_1164.all;
entity word_disp_tb is
end word_disp_tb;

```

```
6  architecture test of word_disp_tb is
7
8     component word_disp
9        port (
10          SW : in std_logic_vector(17 downto 0);
11          HEX0 : out std_logic_vector(6 downto 0);
12          HEX1 : out std_logic_vector(6 downto 0);
13          HEX2 : out std_logic_vector(6 downto 0);
14          HEX3 : out std_logic_vector(6 downto 0);
15          HEX4 : out std_logic_vector(6 downto 0)
16       );
17    end component;
18
19    -- port interface
20    signal SW : std_logic_vector(17 downto 0);
21    signal H0, H1, H2, H3, H4 : std_logic_vector(6 downto 0);
22
23    -- word to "rotate"
24    signal HELLO : std_logic_vector (14 downto 0);
25
26 begin
27    -- simulating HELLO rotation example
28    --        -H--E--L--L--O-
29    HELLO <= "000001010010011";
30
31
32    SW <= "000" & HELLO,
33       "001" & HELLO after 5 ns,
34       "010" & HELLO after 10 ns,
35       "011" & HELLO after 15 ns,
36       "100" & HELLO after 20 ns;
37
38       DUT : word_disp
39       port map(SW, H0, H1, H2, H3, H4);
40
41 end architecture;
```

Code 2.2: word_disp_tb.vhd

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3
4  -- displays a letter (H,E,L,O) using a 5:1MUX and
5  -- a 7-seg display according to char_7seg logic
6
7  entity char_displayer is
8     port (
9        sel, CH0, CH1, CH2, CH3, CH4 : in std_logic_vector (2 downto 0);
10       HEX : out std_logic_vector (6 downto 0)
```

```
11    );
12 end char_displayer;
13
14 architecture Behavior of char_displayer is
15
16    component mux5to1_gen
17       generic (dw : positive := 1);
18       port (
19          s : in std_logic_vector (2 downto 0);
20          u, v, w, x, y : in std_logic_vector (dw - 1 downto 0);
21          m : out std_logic_vector (dw - 1 downto 0)
22       );
23    end component;
24
25    component char_7seg
26       port (
27          c : in std_logic_vector (2 downto 0);
28          disp : out std_logic_vector(0 to 6)
29       );
30    end component;
31
32    signal m : std_logic_vector (2 downto 0);
33
34 begin
35    MX : mux5to1_gen generic map(3)
36    port map(sel, CH0, CH1, CH2, CH3, CH4, m);
37
38    H : char_7seg port map(m, HEX);
39 end architecture;
```

Code 2.3: char_disp.vhd

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3  entity char_displayer_tb is
4  end char_displayer_tb;
5  architecture test of char_displayer_tb is
6
7     component char_displayer is
8        port (
9           sel, CH0, CH1, CH2, CH3, CH4 : in std_logic_vector (2 downto 0);
10          HEX : out std_logic_vector (6 downto 0)
11       );
12    end component;
13
14    signal sel, CH0, CH1, CH2, CH3, CH4 : std_logic_vector(2 downto 0);
15    signal l_out : std_logic_vector(0 to 6);
16
17 begin
```

```
18    --example input
19    CH0 <= "000"; -- H
20    CH1 <= "011"; -- O
21    CH2 <= "010"; -- L
22    CH3 <= "001"; -- E
23    CH4 <= "111"; -- void
24
25    -- test cases
26    sel <= "000", "001" after 10 ns, "010" after 20 ns, "011" after 30
          ns, "100" after 40 ns;
27
28    -- design under test
29    DUT : char_displayer port map(sel, CH0, CH1, CH2, CH3, CH4, l_out);
30 end architecture;
```

Code 2.4: char_disp_tb.vhd

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3
4  entity char_7seg is
5     port (
6        c : in std_logic_vector (2 downto 0);
7        disp : out std_logic_vector(0 to 6)
8     );
9  end char_7seg;
10 architecture Behavior of char_7seg is
11 begin
12    -- boolean functions resulting from
13    -- Kmaps synth
14    disp(0) <= c(2) or not c(0);
15    disp(1) <= c(2) or (c(0) xor c(1));
16    disp(2) <= c(2) or (c(0) xor c(1));
17    disp(3) <= c(2) or not(c(0) or c(1));
18    disp(4) <= c(2);
19    disp(5) <= c(2);
20    disp(6) <= c(1) or c(2);
21 end Behavior;
```

Code 2.5: char_7seg.vhd

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3
4  entity mux5to1_gen is
5     generic (dw : positive := 1);
6     port (
7        s : in std_logic_vector (2 downto 0);
8        u, v, w, x, y : in std_logic_vector (dw - 1 downto 0);
```

```
9       m : out std_logic_vector (dw - 1 downto 0)
10    );
11  end mux5to1_gen;
12
13  architecture mux_structure of mux5to1_gen is
14    signal m_uv,m_wx,m_uv_wx : std_logic_vector(dw - 1 downto 0);
15
16  begin
17  -- using four 2to1 mux
18  MUX1 : entity work.mux2to1_gen
19        generic map(dw)
20        port map (u, v, s(0), m_uv);
21
22  MUX2 : entity work.mux2to1_gen
23        generic map(dw)
24        port map (w, x, s(0), m_wx);
25
26  MUX3 : entity work.mux2to1_gen
27        generic map(dw)
28        port map (m_uv, m_wx, s(1), m_uv_wx);
29
30  MUX4 : entity work.mux2to1_gen
31        generic map(dw)
32        port map (m_uv_wx, y, s(2), m);
33  end mux_structure;
```

Code 2.6: mux5to1_gen.vhd

```
1   library ieee;
2   use ieee.std_logic_1164.all;
3
4   entity mux2to1_gen is
5   generic ( dw : positive := 1 ); -- data width
6   port (
7         x,y : in std_logic_vector ( dw - 1 downto 0 );
8         s : in std_logic;
9         m : out std_logic_vector ( dw -1 downto 0 )
10  );
11  end mux2to1_gen;
12
13
14  architecture logic of mux2to1_gen is
15    signal s_vector : std_logic_vector ( dw -1 downto 0 );
16
17  begin
18      s_vector <= ( others => s );
19      m <= (NOT (s_vector) AND x) OR (s_vector AND y);
20  end architecture;
```

Code 2.7: mux2to1_gen.vhd

Capitolo 3

Binary to Decimal converter

3.1 Spiegazione teorica

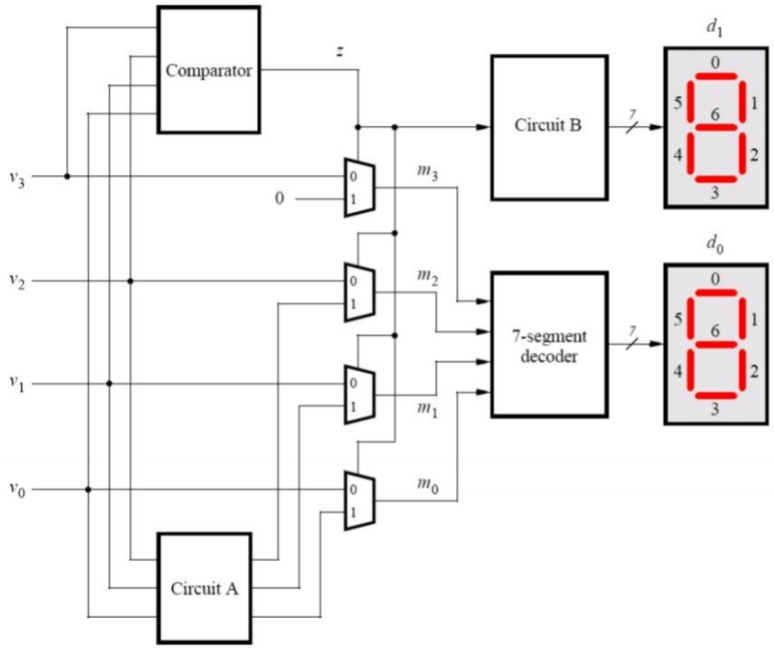


Figura 3.1: Partial block scheme of the system that implements the binary-to-decimal conversion circuit

Si vuole realizzare un circuito in grado di convertire un numero unsigned binario espresso su 4 bit nella sua rappresentazione decimale, servendosi di un circuito analogo a quello mostrato in **figura 3.1**.

3.2 Procedimento

Si descrivono separatamente i blocchetti:

- il comparatore si occupa di verificare che il numero sia maggiore di 9;
- il circuito A permette di scalare i bit di ingresso di 2 unità eseguendo la logica descritta dalla tavola di verità mostrata in **figura 3.2**;
- il circuito B visualizza sul primo display a 7 segmenti uno '0' od un '1' seconda la tavola di verità mostrata in **figura 3.3**;
- il decoder si occupa di tradurre un ingresso binario su 4 bit in un numero decimale rappresentabile su una cifra, secondo la tavola di verità mostrata in **figura 3.4**.

v_2	v_1	v_0	u_2	u_1	u_0
0	0	0	-	-	-
0	0	1	-	-	-
0	1	0	0	0	0
0	1	1	0	0	1
1	0	0	0	1	0
1	0	1	0	1	1
1	1	0	1	0	0
1	1	1	1	0	1

Figura 3.2: truth table circuit A

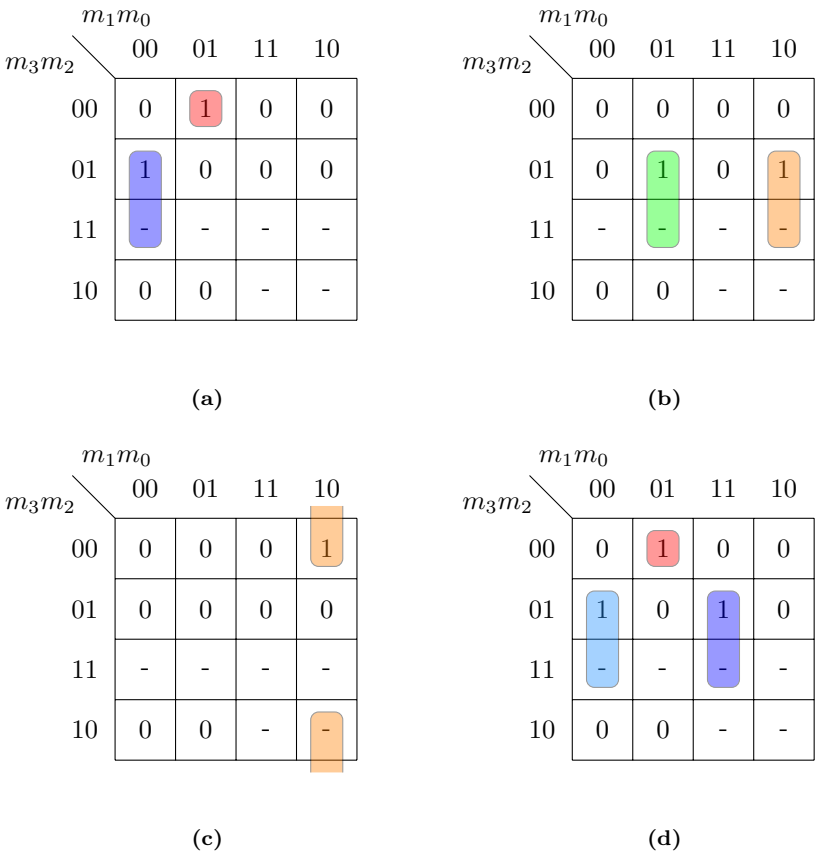
z	a	b	c	d	e	f	g
0	0	0	0	0	0	0	1
1	1	1	1	1	0	0	1

Figura 3.3: truth table circuit B

m_3	m_2	m_1	m_0	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	0	1	0	0

Figura 3.4: truth table decoder

Di quest'ultima, analogamente a quanto fatto nell'esercizio 1, procediamo ad una sintesi manuale tramite K-maps (**figura 3.5**).



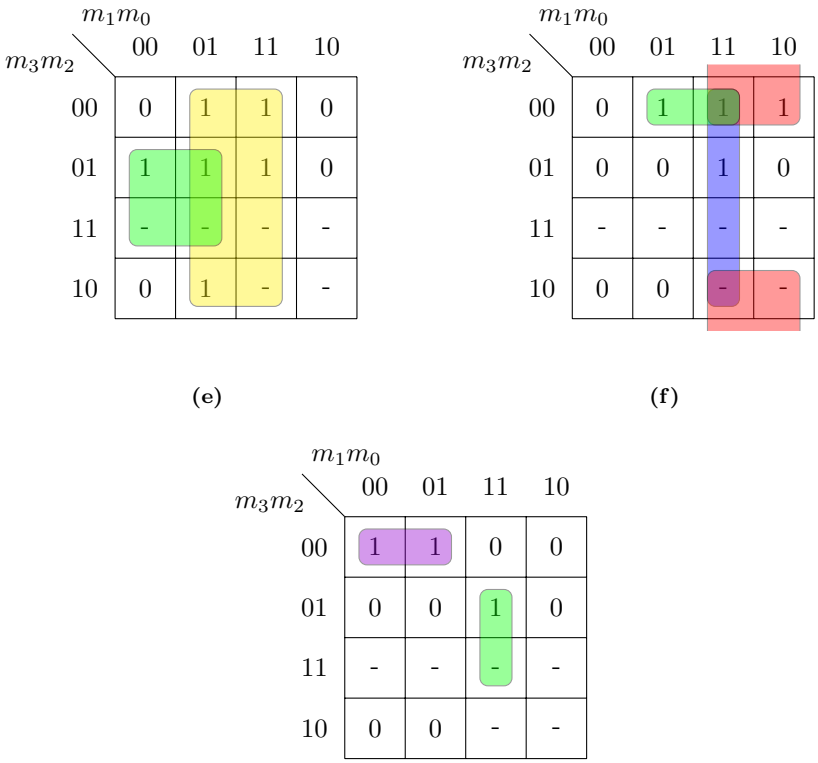
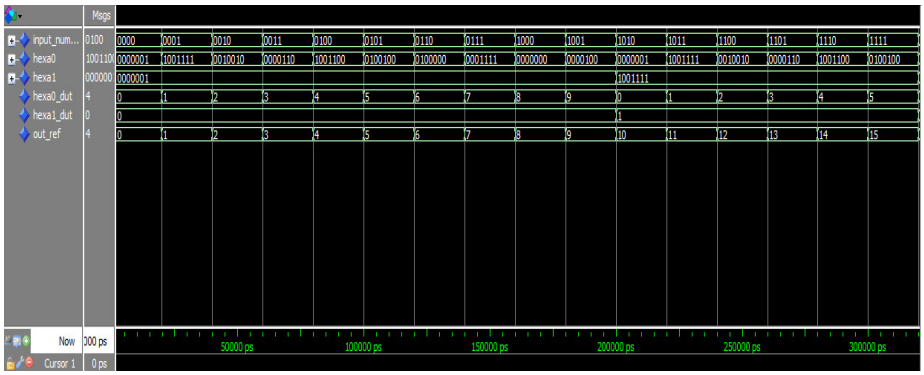


Figura 3.5: Karnaugh maps

3.3 Risultati



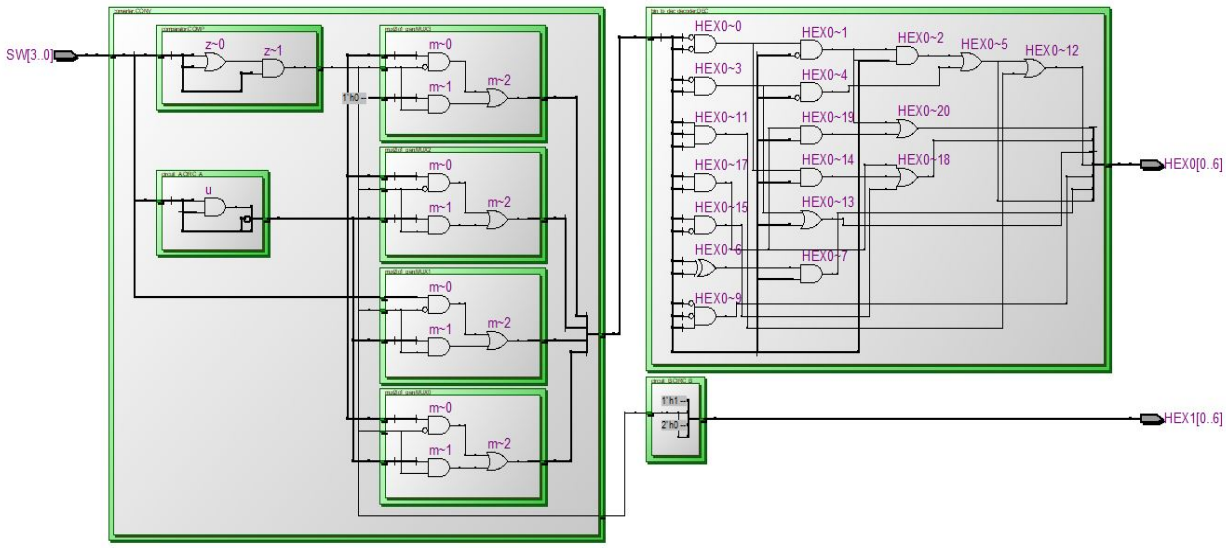


Figura 3.6: RTL exercise no. 3

3.4 Appendice

Si noti che le keywords "when" e "select" sono state utilizzate unicamente nei testbenches ed unicamente per fissare dei valori di riferimento (attesi) per rendere agevole il confronto con l'effettivo comportamento del circuito, ma non ai fini dell'implementazione degli stessi.

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.numeric_std.all;
4
5  entity bin2dec_converter is
6
7    port (
8        SW : in UNSIGNED (3 downto 0);
9        HEX0 : out STD_LOGIC_VECTOR (0 to 6);
10       HEX1 : out STD_LOGIC_VECTOR (0 to 6)
11       );
12
13 end bin2dec_converter;
```

```
14
15 architecture behaviour of bin2dec_converter is
16
17   -- signals between blocks
18   signal z_temp : std_logic;
19   signal m_temp : std_logic_vector (3 downto 0);
20
21   component converter
22       port (
23             v : in unsigned(3 downto 0);
24             m : out std_logic_vector (3 downto 0);
25             z : out std_logic
26       );
27   end component;
28
29   component circuit_B
30     port (
31           z : in std_logic;
32           HEX1 : out std_logic_vector(0 to 6)
33      );
34   end component;
35
36   component bin2dec_decoder
37     port (
38          m : in std_logic_vector (3 downto 0);
39          HEX0 : out std_logic_vector (0 to 6)
40    );
41   end component;
42
43 begin
44
45
46   CONV: converter port map(SW, m_temp, z_temp);
47
48   CIRC_B: circuit_B port map(z_temp, HEX1);
49
50   DEC: bin2dec_decoder port map(m_temp, HEX0);
51
52 end architecture;
```

Code 3.1: bin2dec_converter.vhd

```
1 library ieee;
2 use ieee.std_logic_1164.all;
3 use ieee.numeric_std.all;
4
5 entity bin2dec_converter_tb is
6 end bin2dec_converter_tb;
7
```

```
8
9  architecture behaviour of bin2dec_converter_tb is
10
11   signal input_number : unsigned (3 downto 0);
12   signal HEXA0 : std_logic_vector (0 to 6);
13   signal HEXA1 : std_logic_vector(0 to 6);
14   signal HEXA0_dut : integer;
15   signal HEXA1_dut : integer;
16   signal out_ref : integer;
17
18   component bin2_converter
19     port (
20          SW : in unsigned (3 downto 0);
21          HEX0 : out std_logic_vector (0 to 6);
22          HEX1 : out std_logic_vector (0 to 6));
23   end component;
24
25 begin
26
27   process
28   begin
29
30        input_number <= "0000";
31        wait for 20 ns;
32        input_number <= "0001";
33        wait for 20 ns;
34        input_number <= "0010";
35        wait for 20 ns;
36        input_number <= "0011";
37        wait for 20 ns;
38        input_number <= "0100";
39        wait for 20 ns;
40        input_number <= "0101";
41        wait for 20 ns;
42        input_number <= "0110";
43        wait for 20 ns;
44        input_number <= "0111";
45        wait for 20 ns;
46        input_number <= "1000";
47        wait for 20 ns;
48        input_number <= "1001";
49        wait for 20 ns;
50        input_number <= "1010";
51        wait for 20 ns;
52        input_number <= "1011";
53        wait for 20 ns;
54        input_number <= "1100";
55        wait for 20 ns;
56        input_number <= "1101";
57        wait for 20 ns;
```

```
58        input_number <= "1110";
59        wait for 20 ns;
60        input_number <= "1111";
61        wait for 20 ns;
62    end process;
63
64      CONV: bin2dec_converter port map (input_number, HEXA0, HEXA1);
65
66      with HEXA0 select
67       HEXA0_dut <=
68        0 when "0000001",
69        1 when "1001111",
70        2 when "0010010",
71        3 when "0000110",
72        4 when "1001100",
73        5 when "0100100",
74        6 when "0100000",
75        7 when "0001111",
76        8 when "0000000",
77        9 when "0000100",
78        10 when others;
79
80      with HEXA1 select
81       HEXA1_dut <=
82        0 when "0000001",
83        1 when "1001111",
84        2 when "0010010",
85        3 when "0000110",
86        4 when "1001100",
87        5 when "0100100",
88        6 when "0100000",
89        7 when "0001111",
90        8 when "0000000",
91        9 when "0000100",
92        10 when others;
93
94      out_ref <= to_integer(input_number);
95
96  end architecture;
```

Code 3.2: bin2dec_converter_tb.vhd

```
1   library ieee;
2   use ieee.std_logic_1164.all;
3   use ieee.numeric_std.all;
4
5   entity bin2dec_decoder is
6   port (
7       m : in std_logic_vector (3 downto 0);
```

```
8       HEX0 : out std_logic_vector (0 to 6 )
9   );
10  end bin2dec_decoder;
11
12  architecture k_map of bin2dec_decoder is
13  begin
14
15    -- boolean functions resulting from
16    -- Kmpas synth
17    HEX0(0) <= (not m(3) and not m(2) and not m(1) and m(0)) or (m(2) and
          not m(1) and not m(0));
18    HEX0(1) <= m(2) and (m(1) xor m(0));
19    HEX0(2) <= not m(0) and m(1) and not m(2);
20    HEX0(3) <= (not m(3) and not m(2) and not m(1) and m(0)) or (m(2) and
          not m(1) and not m(0)) or (m(2) and m(1) and m(0));
21
22    HEX0(4) <= m(0) or (m(2) and not m(1));
23    HEX0(5) <= (not m(3) and not m(2) and m(0)) or (not m(2) and m(1)) or
          (m(1) and m(0));
24    HEX0(6) <= (not m(2) and not m(3) and not m(1)) or (m(0) and m(1) and
          m(2));
25
26  end architecture;
```

Code 3.3: bin2dec_decoder.vhd

```
1   library ieee;
2   use ieee.std_logic_1164.all;
3   use ieee.numeric_std.all;
4
5   entity bin2dec_decoder_tb is
6   end bin2dec_decoder_tb;
7
8   architecture struct of bin2dec_decoder_tb is
9
10    signal input_number : std_logic_vector (3 downto 0);
11    signal HEXA0_dut : std_logic_vector (0 to 6 );
12
13    component bin2dec_decoder
14      port (
15           m : in std_logic_vector (3 downto 0);
16           HEX0 : out std_logic_vector (0 to 6));
17    end component;
18
19  begin
20    input_number <= "0000", "0001" after 5 ns, "0010" after 10 ns, "0011"
          after 15 ns,"0100" after 20 ns, "0101" after 25 ns, "0110" after
          30 ns,"0111" after 35 ns, "1000" after 40 ns, "1001" after 45 ns;
21
```

```
22    DEC_DECODER: bin2dec_decoder port map (input_number,HEXA0_dut);
23
24  end architecture;
```

Code 3.4: bin2dec_decoder_tb.vhd

```
1   library ieee;
2   use ieee.std_logic_1164.all;
3   use ieee.numeric_std.all;
4   entity converter is
5      port (
6         v : in unsigned(3 downto 0);
7         m : out std_logic_vector(3 downto 0);
8         z : buffer std_logic
9      );
10  end converter;
11
12  architecture struct of converter is
13
14     component comparator
15        port (
16           v : in unsigned(3 downto 0);
17           z : out std_logic
18        );
19     end component;
20
21     component circuit_A
22        port (
23           v : in unsigned(2 downto 0);
24           u : out unsigned(2 downto 0)
25        );
26     end component;
27
28     component mux2to1_gen
29        generic (dw : positive := 1);
30        port (
31           x, y : in std_logic_vector (dw - 1 downto 0);
32           s : in std_logic;
33           m : out std_logic_vector (dw - 1 downto 0)
34        );
35     end component;
36
37     signal u : unsigned (2 downto 0);
38
39  begin
40     COMP : comparator port map(v, z);
41
42     CIRC_A : circuit_Aport map(v(2 downto 0), u);
43
```

```
44    MUX0 : mux2to1_gen
45    port map(std_logic_vector(v(0 downto 0)), std_logic_vector(u(0 downto
          0)), z, m(0 downto 0));
46
47  MUX1 : mux2to1_gen
48  port map(std_logic_vector(v(1 downto 1)), std_logic_vector(u(1 downto
      1)), z, m(1 downto 1));
49
50  MUX2 : mux2to1_gen
51  port map(std_logic_vector(v(2 downto 2)), std_logic_vector(u(2 downto
      2)), z, m(2 downto 2));
52
53  MUX3 : mux2to1_gen
54  port map(std_logic_vector(v(3 downto 3)), "0", z, m(3 downto 3));
55
56  end architecture;
```

Code 3.5: converter.vhd

```
1   library ieee;
2   use ieee.std_logic_1164.all;
3   use ieee.numeric_std.all;
4
5
6   entity converter_tb is
7   end converter_tb;
8
9   architecture struct of converter_tb is
10
11    signal input_number : unsigned (3 downto 0);
12    signal z_dut : std_logic;
13    signal m_dut : std_logic_vector (3 downto 0);
14
15    component converter
16      port (
17           v : in unsigned(3 downto 0);
18           m : out std_logic_vector(3 downto 0);
19           z : out std_logic
20       );
21    end component;
22
23  begin
24
25        input_number <= "0000",
26        "0001"after 20 ns,
27        "0010" after 40 ns,
28        "0011" after 60 ns,
29        "0100" after 80 ns,
30        "0101" after 100 ns,
```

```
31        "0110" after 120 ns,
32        "0111" after 140 ns,
33        "1000" after 160 ns,
34        "1001" after 180 ns,
35        "1010" after 200 ns,
36        "1011" after 220 ns,
37        "1100" after 240 ns,
38        "1101" after 260 ns,
39        "1110" after 280 ns,
40        "1111" after 300 ns;
41
42    CONV: converter port map( input_number, m_dut, z_dut);
43
44  end architecture;
```

Code 3.6: converter_tb.vhd

```
1   library ieee;
2   use ieee.std_logic_1164.all;
3   use ieee.numeric_std.all;
4
5
6   entity comparator is
7      port (
8           v : in unsigned(3 downto 0);
9           z : out std_logic
10          );
11  end comparator;
12
13  architecture struct of comparator is
14
15  begin
16    --z lit when v > 9
17    z <= v(3) and (v(2) or v(1));
18
19  end struct;
```

Code 3.7: comparator.vhd

```
1   library ieee;
2   use ieee.std_logic_1164.all;
3   use ieee.numeric_std.all;
4
5
6   entity comparator_tb is
7   end comparator_tb;
8
9   architecture struct of comparator_tb is
10
```

```
11    signal input_number : unsigned (3 downto 0);
12    signal z_dut : std_logic;
13
14    component comparator
15      port( v : in unsigned(3 downto 0);
16            z : out std_logic
17      );
18    end component;
19
20  begin
21
22    input_number <= "0101", "0111" after 5 ns, "1111" after 10 ns, "1001"
          after 15 ns, "1010" after 20 ns;
23    COMP: comparator port map (input_number, z_dut);
24
25  end struct;
```

Code 3.8: comparator_tb.vhd

```
1   library ieee;
2   use ieee.std_logic_1164.all;
3   use ieee.numeric_std.all;
4
5
6   entity circuit_A is
7     PORT (
8         v : in unsigned(2 downto 0);
9         u : out unsigned(2 downto 0)
10         );
11  end circuit_A;
12
13  architecture k_map of circuit_A is
14
15  begin
16
17    -- decreasing input bits by 2
18    -- working on first 3 bits
19    u(0) <= v(0);
20    u(1) <= not v(1);
21    u(2) <= v(2) and v(1);
22
23  end architecture;
```

Code 3.9: circuitA.vhd

```
1   library ieee;
2   use ieee.std_logic_1164.all;
3   use ieee.numeric_std.all;
4
```

```
5
6  entity circuit_A_tb is
7  end circuit_A_tb;
8
9  architecture struct of circuit_A_tb is
10
11   signal input_number : unsigned (2 downto 0);
12   signal u_dut : unsigned (2 downto 0);
13
14   component circuit_A
15      PORT (
16           v : IN unsigned(2 downto 0);
17           u : OUT unsigned(2 downto 0)
18      );
19    end component;
20
21 begin
22
23     input_number <= "010",
24     "011"after 20 ns,
25     "100"after 40 ns,
26     "101"after 60 ns,
27     "110"after 80 ns,
28     "111"after 100 ns;
29
30   CIRC_A: circuit_A port map( input_number, u_dut );
31
32 end architecture;
```

Code 3.10: circuitA_tb.vhd

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.numeric_std.all;
4
5
6  entity circuit_B is
7    port (
8        z : in std_logic;
9        HEX1 : out std_logic_vector(0 to 6)
10        );
11 end circuit_B;
12
13
14 architecture k_map of circuit_B is
15
16 begin
17   -- displays 1 when input > 9,
18   -- 0 otherwise
```

```
19    HEX1(0) <= z;
20    HEX1(1) <= '0';
21    HEX1(2) <= '0';
22    HEX1(3) <= z;
23    HEX1(4) <= z;
24    HEX1(5) <= z;
25    HEX1(6) <= '1';
26
27  end architecture;
```

Code 3.11: circuitB.vhd

```
1   ibrary ieee;
2   use ieee.std_logic_1164.all;
3   use ieee.numeric_std.all;
4
5   entity circuit_B_tb is
6   end circuit_B_tb;
7
8
9   architecture behaviuor of circuit_B_tb is
10
11    signal sel : std_logic;
12    signal out_ref : std_logic_vector(0 TO 6);
13
14    component circuit_B
15      port (
16          z : in std_logic;
17          HEX1 : out std_logic_vector(0 TO 6));
18    end component;
19
20  begin
21    sel <= '0', '1' after 5 ns;
22    CIRC_B: circuit_B port map(sel, out_ref);
23
24  end architecture;
```

Code 3.12: circuitB_tb.vhd

Capitolo 4

Binary-to-BCD Converter

4.1 Spiegazione teorica

La quarta parte dell'esercitazione ha previsto l'implementazione di un circuito combinatorio che converte un numero binario espresso su 6 bit nel corrispondente numero decimale su 2 cifre in BCD. Si è fatto uso dell'algoritmo di Double Dabble. In particolare, sono state utilizzate delle LUT che implementano lo shift dei bit e l'aggiunta del numero 0011 (qualora il numero in ingresso sia maggiore di 4).

4.2 Procedimento

Si rappresentano i 6 bit di ingresso su 7 bit tramite zero adding, sicchè i primi 4 bit più significativi vengono mandati in ingresso alla prima LUT ed il bit più significativo nel vettore di uscita è settato di default a '0' (perché il numero più grande che possiamo ottenere è 63, quindi il primo numero sul display è al massimo 6, i.e. 0110 su 4 bit). I restanti 3 bit in uscita dalla LUT fungono da ingressi della LUT successiva insieme al bit successivo del numero binario. Similmente per la terza LUT l'LSB in ingresso va direttamente in uscita, secondo la **4.2**. Uno schema a blocchi è mostrato in **figura 4.1**.

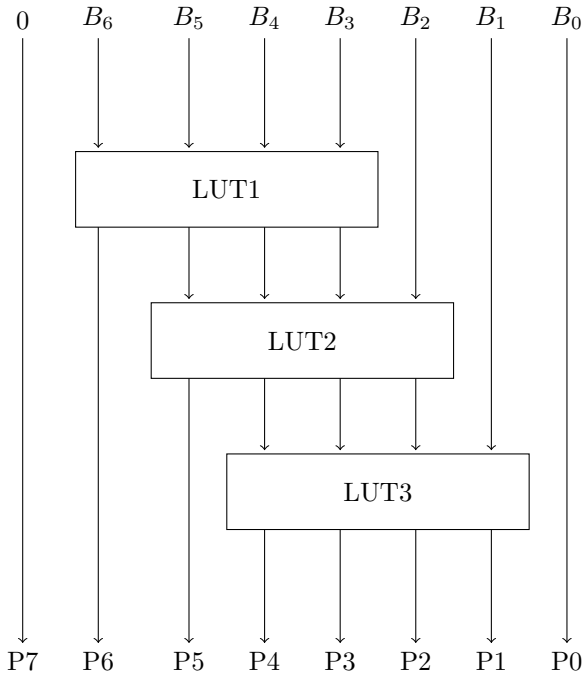
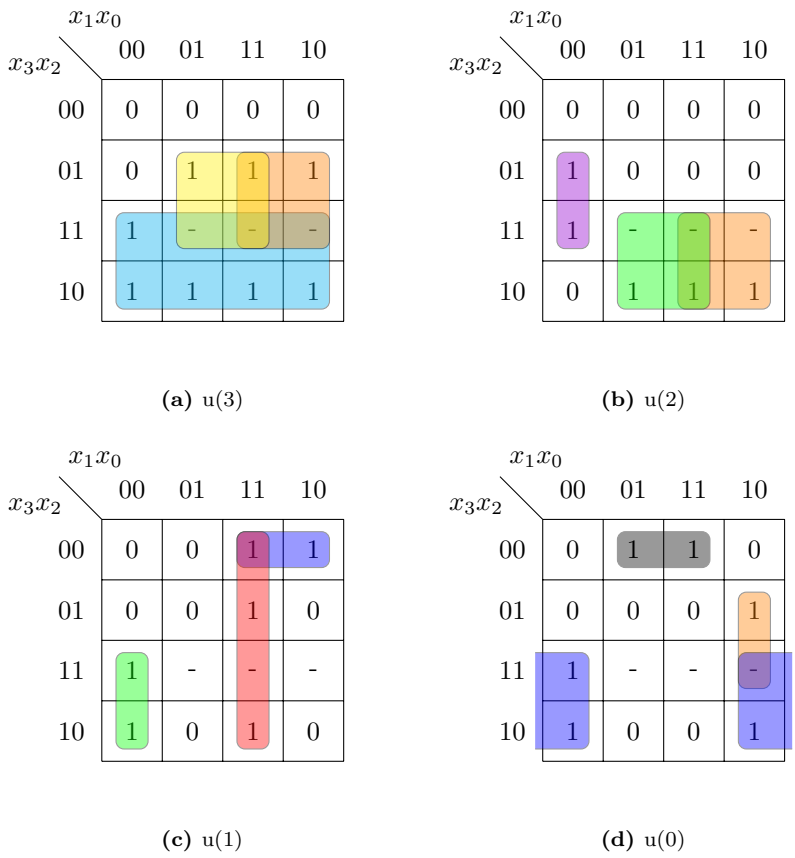


Figura 4.1: Block scheme of circuit

x_3	x_2	x_1	x_0	3	2	1	0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	0
0	0	1	1	0	0	1	1
0	1	0	0	0	1	0	0
0	1	0	1	1	0	0	0
0	1	1	0	1	0	0	1
0	1	1	1	1	0	1	0
1	0	0	0	1	0	1	1
1	0	0	1	1	1	0	0
1	0	1	0	1	1	0	1
1	0	1	1	1	1	1	0
1	1	0	0	1	1	1	1
1	1	0	1	-	-	-	-
1	1	1	0	-	-	-	-
1	1	1	1	-	-	-	-

Figura 4.2: truth table

In **figura 4.3** è mostrata la sintesi manuale della **4.2** tramite mappe di Karnaugh. Le funzioni booleane risultati sono mostrate in appendice.



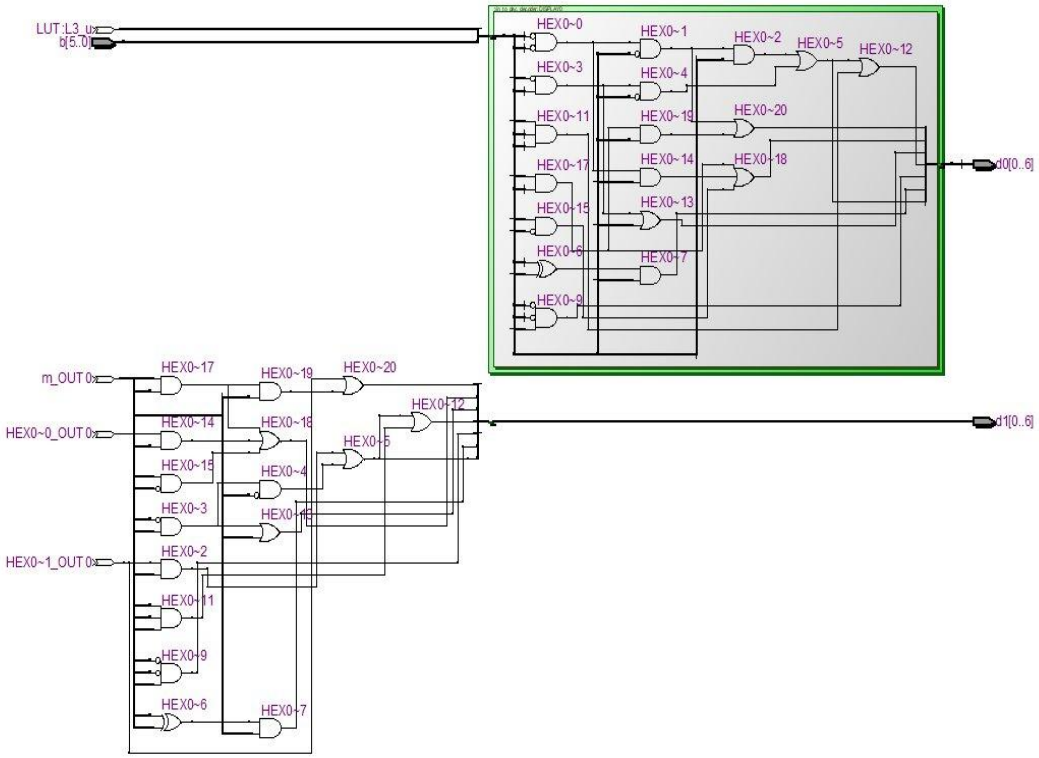
(a) u(3) **(b)** u(2)

(c) u(1) **(d)** u(0)

Figura 4.3: Karnaugh maps

4.3 Risultati

Figura 4.4: Waves exercise no. 4



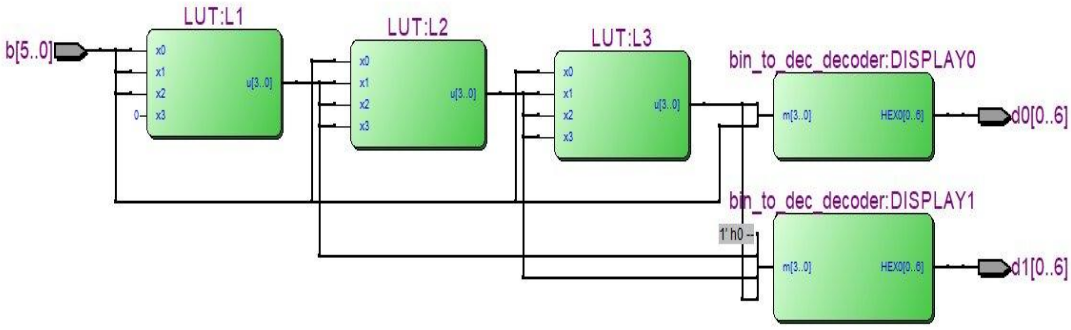


Figura 4.5: RTL views exercise no. 4

4.4 Appendice

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3
4  entity bin6bcd is
5     port (
6        b : in std_logic_vector (5 downto 0);
7        d1 : out std_logic_vector(0 to 6);
8        d0 : out std_logic_vector(0 to 6)
9     );
10 end bin6bcd;
11
12 architecture behavior of bin6bcd is
13
14    component LUT
15       port (
16          x3, x2, x1, x0 : in std_logic;
17          u : out std_logic_vector (3 downto 0)
18       );
19    end component;
20
21    component bin2dec_decoder is
22       port (
23          m : in std_logic_vector (3 downto 0);
24          HEX0 : out std_logic_vector (0 to 6)
25       );
26    end component;
27
28    signal m1 : std_logic_vector(3 downto 0);
29    signal m2 : std_logic_vector(3 downto 0);
30    signal m3 : std_logic_vector(3 downto 0);
```

```
31    signal p : std_logic_vector (7 downto 0);
32
33  begin
34    p(7) <= '0';
35    p(0) <= b(0);
36
37    L1 : LUT port map('0', b(5), b(4), b(3), m1);
38    p(6) <= m1(3);
39
40    L2 : LUT port map(m1(2), m1(1), m1(0), b(2), m2);
41    p(5) <= m2(3);
42
43    L3 : LUT port map(m2(2), m2(1), m2(0), b(1), m3);
44    p(4 downto 1) <= m3(3 downto 0);
45
46  DISPLAY1 : bin2dec_decoder port map(p(7 downto 4), d1);
47  DISPLAY0 : bin2dec_decoder port map(p(3 downto 0), d0);
48
49  end behavior;
```

Code 4.1: bin6bcd.vhd

```
1   library ieee;
2   use ieee.std_logic_1164.all;
3   use ieee.numeric_std.all;
4
5   entity bin6bcd_tb is
6   end bin6bcd_tb;
7
8   architecture test of bin6bcd_tb is
9
10    signal input_bin : std_logic_vector (5 downto 0);
11
12    --bits driving segments (HEX1/HEX0)
13    signal d1_dut, d0_dut : std_logic_vector(0 to 6);
14
15    --expexted display values (decimals/units)
16    signal out1, out0 : integer;
17
18    component bin6bcd
19       port (
20          b : in std_logic_vector (5 downto 0);
21          d1 : out std_logic_vector(0 to 6);
22          d0 : out std_logic_vector(0 to 6)
23       );
24    end component;
25  begin
26    -- test cases
27    input_bin <= "000000",
```

```
28        "000110" after 5 ns,
29        "000111" after 10 ns,
30        "001010" after 15 ns,
31        "001011" after 20 ns,
32        "001111" after 25 ns,
33        "010010" after 30 ns,
34        "010100" after 35 ns,
35        "011000" after 40 ns,
36        "011011" after 45 ns,
37        "100001" after 50 ns,
38        "100101" after 55 ns,
39        "101100" after 60 ns,
40        "110000" after 65 ns,
41        "110010" after 70 ns,
42        "110011" after 75 ns,
43        "111000" after 80 ns,
44        "111111" after 85 ns;
45
46        DUT : bin6bcd
47        port map(input_bin, d1_dut, d0_dut);
48
49    --expected decimal value on diplay d1
50    with d1_dut select
51    out1 <= 0 when "0000001",
52            1 when "1001111",
53            2 when "0010010",
54            3 when "0000110",
55            4 when "1001100",
56            5 when "0100100",
57            6 when "0100000",
58            10 when others;
59
60    --expected decimal value on diplay d0
61    with d0_dut select
62    out0 <= 0 when "0000001",
63            1 when "1001111",
64            2 when "0010010",
65            3 when "0000110",
66            4 when "1001100",
67            5 when "0100100",
68            6 when "0100000",
69            7 when "0001111",
70            8 when "0000000",
71            9 when "0000100",
72            10 when others;
73 end test;
```

Code 4.2: bin6bcd_tb.vhd

```vhdl
library ieee;
use ieee.std_logic_1164.all;

entity LUT is
   port (
       x3, x2, x1, x0 : in std_logic;
       u: out std_logic_vector (3 downto 0)
    );
end LUT;

architecture behavior of LUT is

begin
-- boolean functions from manual synth
u(3) <= x3 or (x2 and x0) or (x2 and x1);
u(2) <= (x3 and x0) or (x3 and x1) or (x2 and not x1 and not x0);
u(1) <= (x1 and x0) or ( not x3 and not x2 and x1) or (x3 and not x1 and not x0);
u(0) <= ( x3 and not x0) or (not x3 and not x2 and x0) or (x2 and x1 and not x0);

end behavior;
```

Code 4.3: LUT.vhd

```vhdl
library ieee;
use ieee.std_logic_1164.all;

entity LUT_tb is
end LUT_tb;

architecture test of LUT_tb is

   signal b_dut : std_logic_vector(3 downto 0);
   signal out_dut : std_logic_vector(3 downto 0);
   component LUT is
      port (
         x : in std_logic_vector(3 downto 0);
         u : out std_logic_vector (3 downto 0)
      );
   end component;

begin
   b_dut <= "0000", "0010" after 10 ns, "0100" after 20 ns, "0101" after 30 ns, "1000" after 40 ns, "1100" after 50 ns;
   DUT : LUT port map(x => b_dut, u => out_dut);

end test;
```

Code 4.4: LUT_tb.vhd