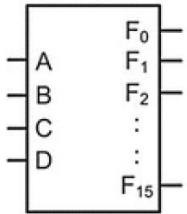


2. Опис MSI комбинационих мрежа у VHDL-у

Задатак 6.1.2

Написати VHDL модел који описује 4-у-16 “one-hot” декодер задат блок дијаграмом и таблицом истинитости на слици 6.1. Користити конкурентну доделу сигнала и логичке операторе. Дефинистаи ентитет као што је приказано на слици 6.2.

4-to-16 One-Hot Decoder



ABCD	F ₁₄	F ₁₃	F ₁₂	F ₁₁	F ₁₀	F ₉	F ₈	F ₇	F ₆	F ₅	F ₄	F ₃	F ₂	F ₁	F ₀
0 0 0 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0 0 0 1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
0 0 1 0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
0 0 1 1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
0 1 0 0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
0 1 0 1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
0 1 1 0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
0 1 1 1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
1 0 0 0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
1 0 0 1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
1 0 1 0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
1 0 1 1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
1 1 0 0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
1 1 0 1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
1 1 1 0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
1 1 1 1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Слика. 6.1 Функционалност 4-у-16 “one-hot” декодера

```
entity decoder_1hot_4to16 is
    port (A : in Bit_vector (3 downto 0);
          F : out bit_vector (15 downto 0));
end entity;
```

Слика. 6.2 Дефиниција ентитета за 4-у-16 “one-hot” декодер

Решење задатка 6.1.2

$F_0 = A'B'C'D'$ $F_1 = A'B'C'D$ $F_2 = A'B'CD'$ $F_3 = A'B'CD$
 $F_4 = A'BC'D'$ $F_5 = A'BC'D$ $F_6 = A'BCD'$ $F_7 = A'BCD$
 $F_8 = AB'C'D'$ $F_9 = AB'C'D$ $F_{10} = AB'CD'$ $F_{11} = AB'CD$
 $F_{12} = ABC'D'$ $F_{13} = ABC'D$ $F_{14} = ABCD'$ $F_{15} = ABCD$

```
entity decoder_1hot_4to16 is
    port (A : in bit_vector (3 downto 0);
          F : out bit_vector (15 downto 0));
end entity;
```

```
architecture decoder_1hot_4to16_arch of decoder_1hot_4to16 is
begin
    F(0) <= not A(3) and not A(2) and not A(1) and not A(0);
    F(1) <= not A(3) and not A(2) and not A(1) and A(0);
    F(2) <= not A(3) and not A(2) and A(1) and not A(0);
```

```

F(3) <= not A(3) and not A(2) and A(1) and A(0);
F(4) <= not A(3) and A(2) and not A(1) and not A(0);
F(5) <= not A(3) and A(2) and not A(1) and A(0);
F(5) <= not A(3) and A(2) and A(1) and not A(0);
F(6) <= not A(3) and A(2) and A(1) and A(0);
F(7) <= A(3) and not A(2) and not A(1) and not A(0);
F(8) <= A(3) and not A(2) and not A(1) and A(0);
F(9) <= A(3) and not A(2) and A(1) and not A(0);
F(10) <= A(3) and not A(2) and A(1) and A(0);
F(11) <= A(3) and A(2) and not A(1) and not A(0);
F(12) <= A(3) and A(2) and not A(1) and A(0);
F(13) <= A(3) and A(2) and A(1) and not A(0);
F(14) <= A(3) and A(2) and A(1) and A(0);
end architecture;

```

Задатак 6.1.3

Написати VHDL модел који описује 4-у-16 “one-hot” декодер задат блок дијаграмом и таблицом истинитости на слици 6.1. Користити условну доделу сигнала. Дефинистаи ентитет као што је приказано на слици 6.2.

Решење задатка 6.1.3

```

entity decoder_1hot_4to16 is
    port (A : in bit_vector (3 downto 0);
          F : out bit_vector (15 downto 0));
end entity;

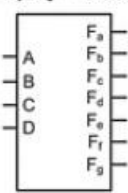
architecture decoder_1hot_4to16_arch of decoder_1hot_4to16 is
begin
    F <= "0000000000000001" when (A="0000") else
        "0000000000000010" when (A="0001") else
        "0000000000000100" when (A="0010") else
        "0000000000001000" when (A="0011") else
        "0000000000010000" when (A="0100") else
        "0000000000010000" when (A="0101") else
        "0000000000100000" when (A="0110") else
        "0000000001000000" when (A="0111") else
        "0000000010000000" when (A="1000") else
        "0000000100000000" when (A="1001") else
        "0000001000000000" when (A="1010") else
        "0000010000000000" when (A="1011") else
        "0000100000000000" when (A="1100") else
        "0001000000000000" when (A="1101") else
        "0010000000000000" when (A="1110") else
        "0100000000000000" when (A="1111");
end architecture;

```

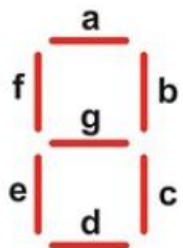
Задатак 6.1.7

Написати VHDL модел који описује 4-улазни 7-сегментни HEX карактер декодер коришћењем селекционе доделе сигнала. Користити дефиницију ентитета као што је приказано на слици 6.4. Систем има 4-битни улазни вектор A и 7-битни излазни вектор F. Индивидуалне скаларне вредности у оквиру излазног вектора “F(6 downto 0)” одговарају сегментима на дисплеју a, b, c, d, e, f и g. Логичка 1 на излазу одговара акцији LED = ON. Дисплеј ће приказати HEX карактере 0–9, A, b, c, d, E, и F, који одговарају 4-битном улазном коду A. Шаблон за креирање таблице истинитости је приказан на слици 6.3. Сигнали у овој таблицу одговарају ентитету на следећи начин: A = A(3), B = A(2), C = A(1), D = A(0), Fa = F(6), Fb = F(5), Fc = F(4), Fd = F(3), Fe = F(2), Ff = F(1), and Fg = F(0).

7-Segment Display Decoder



7-Segment Display Layout



A	B	C	D		F _a	F _b	F _c	F _d	F _e	F _f	F _g
0	0	0	0								
0	0	0	1								
0	0	1	0								
0	0	1	1								
<hr/>											
0	1	0	0								
0	1	0	1								
0	1	1	0								
0	1	1	1								
<hr/>											
1	0	0	0								
1	0	0	1								
1	0	1	0								
1	0	1	1								
<hr/>											
1	1	0	0								
1	1	0	1								
1	1	1	0								
1	1	1	1								

Слика. 6.3 Таблица истинитости декодера за 7-сегментни дисплеј

```
entity decoder_7seg_4in is
  port (A      : in bit_vector(3 downto 0);
        F      : out bit_vector(6 downto 0));
end entity;
```

Слика. 6.4 Дефиниција ентитета декодера за 7-сегментни дисплеј

Решење задатка 6.1.7

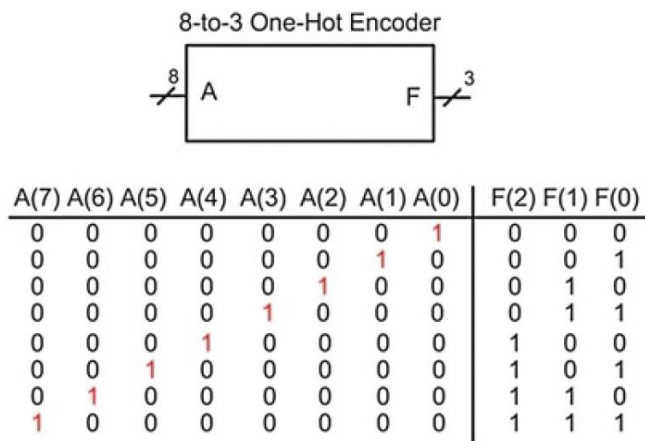
A	B	C	D	Fa	Fb	Fc	Fd	Fe	Ff	Fg
0	0	0	0	1	1	1	1	1	1	0
0	0	0	1	0	1	1	0	0	0	0
0	0	1	0	1	1	0	1	1	0	1
0	0	1	1	1	1	1	1	0	0	1
0	1	0	0	0	1	1	0	0	1	1
0	1	0	1	1	0	1	1	0	1	1
0	1	1	0	1	0	1	1	1	1	1
0	1	1	1	1	1	1	0	0	0	0
1	0	0	0	1	1	1	1	1	1	1
1	0	0	1	1	1	1	1	0	1	1
1	0	1	0	1	1	1	0	1	1	1
1	0	1	1	0	0	1	1	1	1	1
1	1	0	0	0	0	0	1	1	0	1
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

```
entity decoder_7seg_4in is
    port (A : in bit_vector (3 downto 0);
          F : out bit_vector (6 downto 0));
end entity;

architecture decoder_7seg_4in_arch of decoder_7seg_4in is
begin
    with (A) select
        F <= "1111110" when "0000",
              "0110000" when "0001",
              "1101101" when "0010",
              "1111001" when "0011",
              "0110011" when "0100",
              "1011011" when "0101",
              "1011111" when "0110",
              "1110000" when "0111",
              "1111111" when "1000",
              "1111011" when "1001",
              "1110111" when "1010",
              "0011111" when "1011",
              "0001101" when "1100",
              "0111101" when "1101",
              "1001111" when "1110",
              "1000111" when "1111";
end architecture;
```

Задатак 6.2.2

Написати VHDL модел за 8-у-3 "one-hot" бинарни кодер коришћењем конкурентне доделе сигнала и логичких оператора. Блок дијаграм и таблица истинитости за кодер су приказани на слици 6.5. Користити дефиницију ентитета као што је приказано на слици 6.6.



Слика. 6.5 Функционалност 8-у-3 “one-hot” бинарног кодера

```
entity encoder_8to3_binary is
    port (A : in bit_vector (7 downto 0);
          F : out bit_vector (2 downto 0));
end entity;
```

Слика. 6.6 Дефиниција ентитета за 8-у-3 “one-hot” бинарни кодер

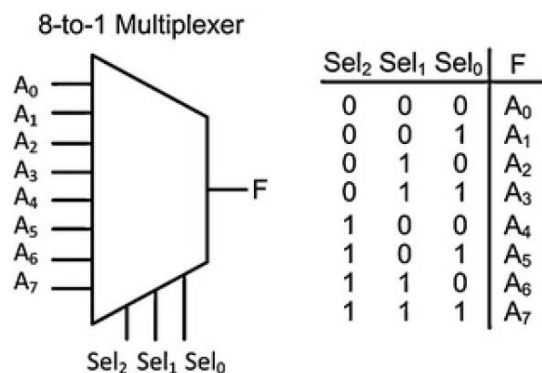
Решење задатка 6.2.2

```
entity encoder_8to3_binary is
    port (A : in bit_vector (7 downto 0);
          F : out bit_vector (2 downto 0));
end entity;

architecture encoder_8to3_binary_arch of encoder_8to3_binary is
begin
    F(2) <= A(7) or A(6) or A(5) or A(4);
    F(1) <= A(7) or A(6) or A(3) or A(2);
    F(0) <= A(7) or A(5) or A(3) or A(1);
end architecture;
```

Задатак 6.3.3

Написати VHDL модел за 8-у-1 мултиплексер коришћењем условне доделе сигнала. Блок дијаграм и таблица истинитости за мултиплексер су приказани на слици 6.7. Користити дефиницију ентитета као што је приказано на слици 6.8.



Слика. 6.7 Функционалност 8-у-1 мултиплексера

```
entity mux_8to1 is
    port (A : in bit_vector (7 downto 0);
          Sel : in bit_vector (2 downto 0);
          F : out bit);
end entity;
```

Слика. 6.8 Дефиниција ентитета за 8-у-1 мултиплексер

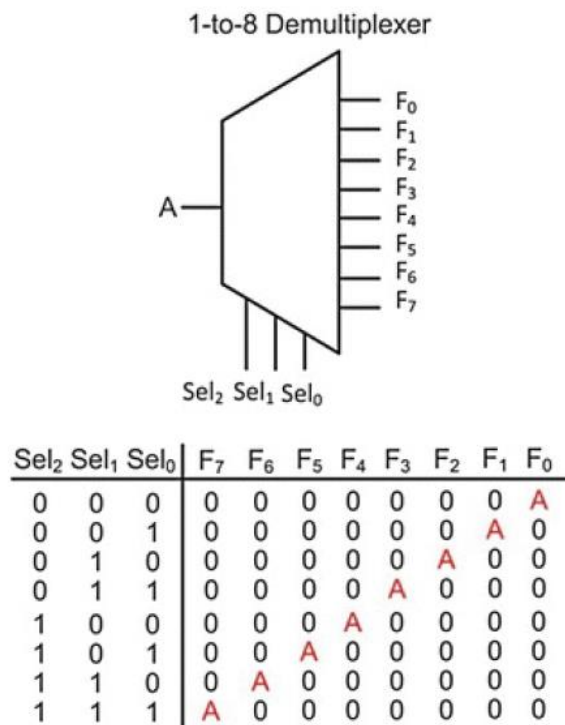
Решење задатка 6.3.3

```
entity mux_8to1 is
    port (A : in bit_vector (7 downto 0);
          Sel : in bit_vector (2 downto 0);
          F : out bit);
end entity;

architecture mux_8to1_arch of mux_8to1 is
begin
    F <= A(7) when (Sel = "000") else
        A(6) when (Sel = "001") else
        A(5) when (Sel = "010") else
        A(4) when (Sel = "011") else
        A(3) when (Sel = "100") else
        A(2) when (Sel = "101") else
        A(1) when (Sel = "110") else
        A(0) when (Sel = "111");
end architecture;
```

Задатак 6.4.4

Написати VHDL модел за 1-у-8 демултиплексер коришћењем селекционе доделе сигнала. Блок дијаграм и таблица истинитости за мултиплексер су приказани на слици 6.9. Користити дефиницију ентитета као што је приказано на слици 6.10.



Слика. 6.9 Функционалност 1-у-8 демултиплексера

```
entity demux_1to8 is
    port (A : in bit;
          Sel : in bit_vector (2 downto 0);
          F : out bit_vector (7 downto 0));
end entity;
```

Слика. 6.10 Дефиниција ентитета за 1-у-8 демултиплексер

Решение задатка 6.4.4

```
entity demux_1to8 is
    port (A      : in  bit;
          Sel    : in  bit_vector (2 downto 0);
          F      : out bit_vector (7 downto 0));
end entity;

architecture demux_1to8_arch of demux_1to8 is
begin
    with (Sel) select
        F(7) <= A when "000"), '0' when others;
    with (Sel) select
        F(6) <= A when "001"), '0' when others;
    with (Sel) select
        F(5) <= A when "010"), '0' when others;
    with (Sel) select
        F(4) <= A when "011"), '0' when others;
    with (Sel) select
        F(3) <= A when "100"), '0' when others;
    with (Sel) select
        F(2) <= A when "101"), '0' when others;
    with (Sel) select
        F(1) <= A when "110"), '0' when others;
    with (Sel) select
        F(0) <= A when "111"), '0' when others;
end architecture;
```