Reklama Tekstowa

Student [redacted]
Nr Albumu 160128
PTCID 15
Kierunek Inforamtyka
Wydział Wydział Informatyki i Teleinforamtyki
Ćwiczenie 4

Multiplekser

Wybór wejść wektorów danych jest realizowany według numeracji binarnej realizowanej w kodzie Greya, pozwala to na uzyskanie kolejnej postaci wyświetlanego słowa po zmianie tylko jednego bitu ciągu określającego aktualny etap cyklu wyświetlania (wejścia SW15-SW17)
Wartości kodu Greya: 000,001,011,010,110,111,101,100
Dlatego realizacja funckji multipleksera będzie realizowana według podanego kodu VHDL:

```
PROCESS(S, U0, U1, U2, U3, U4, U5,U6,U7)
    BEGIN
        CASE S IS
            WHEN "000" => M <= U0;
            WHEN "001" => M <= U1;
            WHEN "011" => M <= U2;
            WHEN "010" => M <= U3;
            WHEN "110" => M <= U4;
            WHEN "111" => M <= U5;
            WHEN "101" => M <= U6;
            WHEN "100" => M <= U7;
        END CASE;
END PROCESS;
```

Na podstawie wejścia S transkodera, które jest wektorem 3 bitowym przypisywane jest wyjściu M jedno z wejść U0, U1, U2, U3, U4, U5, U6, U7. Na wejście S będzie podawany wekor SW[15-17], a na wejścia U0, U1, U2, U3, U4, U5, U6, U7 będą podawane sygnały SW[0-2], SW[3-5], SW[6-8],SW[9-11], SW[12-14] odpowiedzialne na przetrzymywanie kodów znaków wyświetlanych na wyświetlaczu.

Transkoder

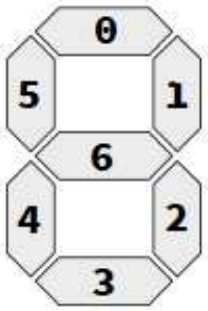
Zadaniem transkodera w reklamie teksotwej jest zamiana kodu liter na kod wyświetlacza 7 segmentowego. Transkoder posiada 3 wejścia bitowe (wejścia na kod znaku) i 7 wyjść bitowych (wyjścia prowadzące do wejść wyświetlacza).

znak	kod NKB
C	001
E	010
F	011
G	100
H	101
-	110
-	111
spacja	000

Tabela 1: Tabela kodów NKB podanych znaków zapisanych na 3 bitach. Znak "w tabeli oznacza brak symbolu

znak	bity wyświetlacza						
	0	1	2	3	4	5	6
C	1	0	0	1	1	1	0
E	1	0	0	1	1	1	1
F	1	0	0	0	1	1	1
G	1	0	1	1	1	1	1
H	0	1	1	0	1	1	1
spacja	0	0	0	0	0	0	0

Tabela 2: Tabela z reprezentacją bitową jaką trzeba podać na wejścia wyświetlacza by uzyskać dany znak na wyświetlaczu.



Rysunek 1: Wyświetlacz 7 segmentowy z zaznaczonymi numerami bitów jakie odpowiadają za dany segment.

Tablice Carnough do ustalenia funckji Bool'a na kolejne wyjścia transkoder.

we2 we1 we0	0 0	0 1	1 1	1 0
0	0_0	1_1	1_3	1_2
1	1_4	0_5	$\oslash_7$	$\oslash_6$

Tabela 3: Tablica Carnough dla wyjścia *wy0* transkodera. Grupy: (2, 3, 6, 7), (1, 3), (4, 6)

we2 we1 we0	0 0	0 1	1 1	1 0
0	0_0	0_1	0_3	0_2
1	0_4	1_5	$\oslash_7$	$\oslash_6$

Tabela 4: Tablica Carnough dla wyjścia *wy1* transkodera. Grupy: (5, 7)

we2 we1 we0	0 0	0 1	1 1	1 0
0	0_0	0_1	0_3	0_2
1	1_4	1_5	$\oslash_7$	$\oslash_6$

Tabela 5: Tablica Carnough dla wyjścia *wy2* transkodera. Grupy: (4, 5, 6, 7)

we2 we1 we0	0 0	0 1	1 1	1 0
0	0_0	1_1	0_3	1_2
1	1_4	0_5	$\oslash_7$	$\oslash_6$

Tabela 6: Tablica Carnough dla wyjścia *wy3* transkodera. Grupy: (4, 6), (1), (2, 6)

we2 we1 we0	0 0	0 1	1 1	1 0
0	0_0	1_1	1_3	1_2
1	1_4	1_5	$\oslash_7$	$\oslash_6$

Tabela 7: Tablica Carnough dla wyjścia *wy4* transkodera. Grupy: (4, 5, 6, 7), (1, 3, 5, 7), (2, 3, 6, 7)

we2 we1 we0	0 0	0 1	1 1	1 0
0	0_0	1_1	1_3	1_2
1	1_4	1_5	$\oslash_7$	$\oslash_6$

Tabela 8: Tablica Carnough dla wyjścia *wy5* transkodera. Grupy: (4, 5, 6, 7), (1, 3, 5, 7), (2, 3, 6, 7)

we2 we1 we0	0 0	0 1	1 1	1 0
0	0_0	0_1	1_3	1_2
1	1_4	1_5	$\oslash_7$	$\oslash_6$

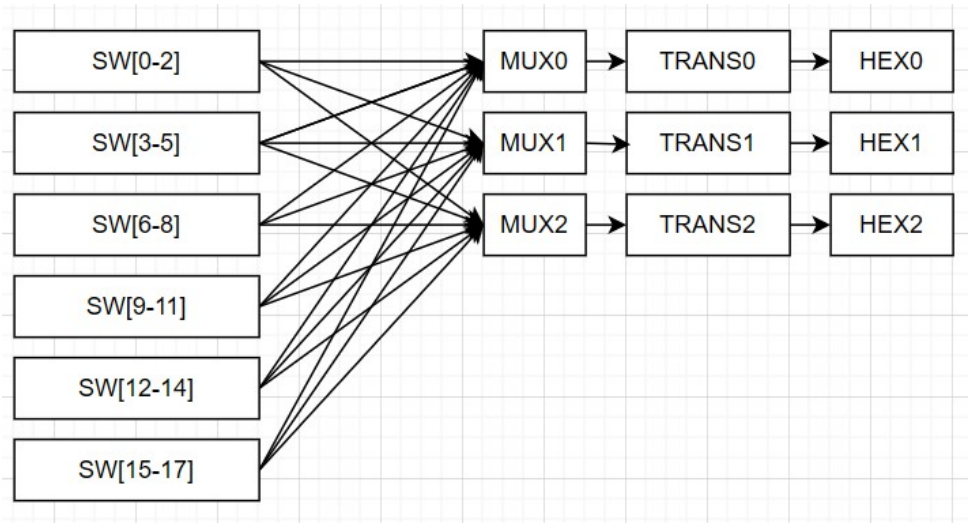
Tabela 9: Tablica Carnough dla wyjścia *wy6* transkodera. Grupy: (4, 5, 6, 7), (2, 3, 6, 7)

wy0 = we1 + $\overline{we2}$we0 + we2$\overline{we0}$
wy1 = we2we0
wy2 = we2
wy3 = we1$\overline{we0}$ + we2$\overline{we0}$ + $\overline{we2we1}$we0
wy4 = we0 + we1 + we2
wy5 = we0 + we1 + we2
wy6 = we1 + we2

Funckje te trzeba jednak zanegować gdyż w wyświetlaczu siedmiosegmentowym dany segment jest podświetlany, gdy podawany jest stan niski. Dlatego ostateczne funkcję kolejnych wyjść transkodera(i odpowiednio wejść wyświetlaczy) realizowane za pomocą kodu VHDL

```
1 Display(0) <= NOT(C(1) OR (NOT C(2) AND C(0)) OR ( C(2) AND NOT C(1) AND NOT C(0)));
2 Display(1) <= NOT(C(2) AND C(0));
3 Display(2) <= NOT C(2);
4 Display(3) <= NOT((C(1) AND NOT C(0))OR(C(2) AND NOT C(0)) OR (NOT C(2) AND NOT C(1) AND
   ↪  C(0)));
5 Display(4) <= NOT(C(0) OR C(1) OR C(2));
6 Display(5) <= NOT(C(0) OR C(1) OR C(2));
7 Display(6) <= NOT(C(1) OR C(2));
```

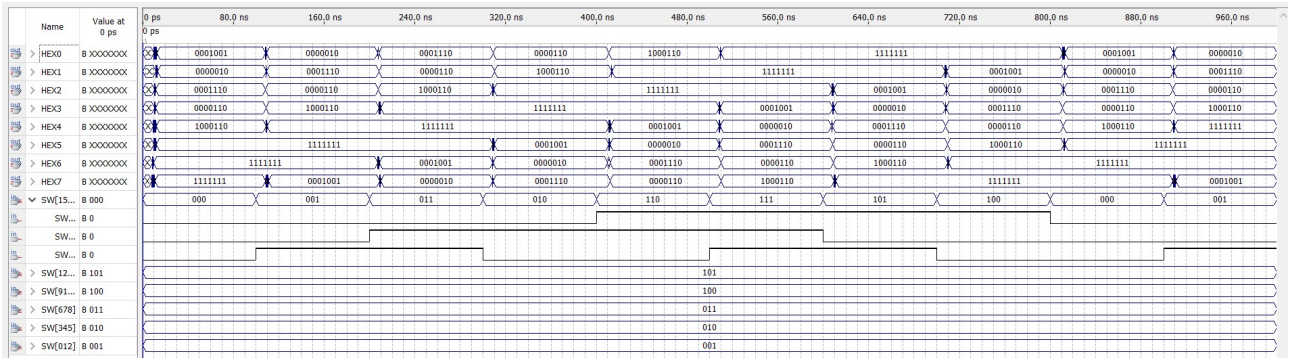
Układ ideowy



Rysunek 2: Schemat ideowy trzech modułów sterujących wyświetlaczami dla rekalmy tekstowej.

Wejście danych 3 bitowego wektora kodu kroku pracy urządzenia dla wszystkich multiplekserów pozostaje takie samo. Natomiast dla 8 wektorów 3 bitowych dla danych kodów znaków/liter, wejścia są przesunięte między sobą o 1 wektor. np. jeżeli pierwszy multiplekser ma na kolejnych wejściach podany ciąg znaków: C, E, F, G, H, SPACJA, SPACJA, SPACJA to następny w kolejności multiplekser będzie miał ten ciąg przesunięty o 1 pozycję czyli np. : E, F, G, H, SPACJA, SPACJA, SPACJA, C. Co spowoduje, że pierwszy multiplekser w pierwszym stanie pracy będzie wybierał znak 'C', natomiast drufi multiplekser będzie wybierał znak 'E'

Wynik Symulacji za pomocą symulacji Waveform



Rysunek 3: Przebieg Rzeczywisty Reklamy Tekstowej

4.1

Kod VHDL

Ostateczny kod VHDL realizujący zadanie reklamy tekstowej.

CW4.vhd :

```
1 LIBRARY ieee;
2 USE ieee.std_logic_1164.all;
3 ENTITY CW4 IS
```

```
4           PORT ( SW : IN STD_LOGIC_VECTOR(17 DOWNTO 0);
5           --wyświetlacze
6           HEX0 : OUT STD_LOGIC_VECTOR(6 DOWNTO 0);
7           HEX1 : OUT STD_LOGIC_VECTOR(6 DOWNTO 0);
8           HEX2 : OUT STD_LOGIC_VECTOR(6 DOWNTO 0);
9           HEX3 : OUT STD_LOGIC_VECTOR(6 DOWNTO 0);
10          HEX4 : OUT STD_LOGIC_VECTOR(6 DOWNTO 0);
11          HEX5 : OUT STD_LOGIC_VECTOR(6 DOWNTO 0);
12          HEX6 : OUT STD_LOGIC_VECTOR(6 DOWNTO 0);
13          HEX7 : OUT STD_LOGIC_VECTOR(6 DOWNTO 0));
14  END CW4;
15
16  ARCHITECTURE strukturalna OF CW4 IS
17  CONSTANT SPACJA: STD_LOGIC_VECTOR(2 DOWNTO 0):="000"; -- KOD SPACJI - uwaga na rodzaj ”” 
     ↪  przy kompilacji
18          --DEKLARACJA KOMPONENTÓW
19          COMPONENT mux3bit8to1 -- muliptekser
20                  PORT (
21                          S, U0, U1, U2, U3, U4, U5,U6,U7: IN STD_LOGIC_VECTOR(2 DOWNTO 0);
                             ↪  --WEKTOR STERUJĄCY I 8 wektorów INFORMACYJNYCH
22                          M : OUT STD_LOGIC_VECTOR(2 DOWNTO 0)
23                  );
24          END COMPONENT;
25          COMPONENT char7seg -- transkoder
26                  PORT(
27                          C : IN STD_LOGIC_VECTOR(2 DOWNTO 0);
28                          Display : OUT STD_LOGIC_VECTOR(6 DOWNTO 0)
29                          );
30          END COMPONENT;
31
32          SIGNAL M0 : STD_LOGIC_VECTOR(2 DOWNTO 0);
33          SIGNAL M1 : STD_LOGIC_VECTOR(2 DOWNTO 0);
34          SIGNAL M2 : STD_LOGIC_VECTOR(2 DOWNTO 0);
35          SIGNAL M3 : STD_LOGIC_VECTOR(2 DOWNTO 0);
36          SIGNAL M4 : STD_LOGIC_VECTOR(2 DOWNTO 0);
37          SIGNAL M5 : STD_LOGIC_VECTOR(2 DOWNTO 0);
38          SIGNAL M6 : STD_LOGIC_VECTOR(2 DOWNTO 0);
39          SIGNAL M7 : STD_LOGIC_VECTOR(2 DOWNTO 0);
40
41          BEGIN
42          -- KONKRETYZACJA UŻYCIA KOMPONENTÓW
43          -- SW(17 DOWNTO 15) sterujący sygnal multipleksera
44                  -- DO WYKONANIA : KONKRETYZACJE KOLEJNYCH MULTIPLEKSERÓW UKŁADU
45                  MUX0: mux3bit8to1 PORT MAP (SW(17 DOWNTO 15), SW(14 DOWNTO 12), SW(11 DOWNTO
                     ↪  9),        SW(8 DOWNTO 6), SW(5 DOWNTO 3), SW(2 DOWNTO
                     ↪  0),SPACJA,SPACJA,SPACJA, M0);
46                  MUX1: mux3bit8to1 PORT MAP (SW(17 DOWNTO 15), SW(11 DOWNTO 9), SW(8 DOWNTO
                     ↪  6), SW(5 DOWNTO 3), SW(2 DOWNTO 0),SPACJA,SPACJA,SPACJA, SW(14 DOWNTO
                     ↪  12), M1);
47                  MUX2: mux3bit8to1 PORT MAP (SW(17 DOWNTO 15), SW(8 DOWNTO 6), SW(5 DOWNTO
                     ↪  3), SW(2 DOWNTO 0),SPACJA,SPACJA,SPACJA, SW(14 DOWNTO 12), SW(11 DOWNTO
                     ↪  9), M2);
48                  MUX3: mux3bit8to1 PORT MAP (SW(17 DOWNTO 15), SW(5 DOWNTO 3), SW(2 DOWNTO
                     ↪  0),SPACJA,SPACJA,SPACJA, SW(14 DOWNTO 12), SW(11 DOWNTO 9), SW(8 DOWNTO
                     ↪  6), M3);
```

```
49                MUX4: mux3bit8to1 PORT MAP (SW(17 DOWNTO 15), SW(2 DOWNTO
                   ↪  0),SPACJA,SPACJA,SPACJA, SW(14 DOWNTO 12), SW(11 DOWNTO 9), SW(8 DOWNTO
                   ↪  6), SW(5 DOWNTO 3), M4);
50                MUX5: mux3bit8to1 PORT MAP (SW(17 DOWNTO 15), SPACJA,SPACJA,SPACJA, SW(14
                   ↪  DOWNTO 12), SW(11 DOWNTO 9), SW(8 DOWNTO 6), SW(5 DOWNTO 3), SW(2 DOWNTO
                   ↪  0),M5);
51                MUX6: mux3bit8to1 PORT MAP (SW(17 DOWNTO 15), SPACJA,SPACJA, SW(14 DOWNTO
                   ↪  12), SW(11 DOWNTO 9), SW(8 DOWNTO 6), SW(5 DOWNTO 3), SW(2 DOWNTO
                   ↪  0),SPACJA,M6);
52                MUX7: mux3bit8to1 PORT MAP (SW(17 DOWNTO 15), SPACJA, SW(14 DOWNTO 12),
                   ↪  SW(11 DOWNTO 9), SW(8 DOWNTO 6), SW(5 DOWNTO 3), SW(2 DOWNTO
                   ↪  0),SPACJA,SPACJA,M7);
53                -- DO WYKONANIA : KONKRETYZACJE KOLEJNYCH TRANSKODERÓW
54                H0: char7seg PORT MAP (M0, HEX0);
55                H1: char7seg PORT MAP (M1, HEX1);
56                H2: char7seg PORT MAP (M2, HEX2);
57                H3: char7seg PORT MAP (M3, HEX3);
58                H4: char7seg PORT MAP (M4, HEX4);
59                H5: char7seg PORT MAP (M5, HEX5);
60                H6: char7seg PORT MAP (M6, HEX6);
61                H7: char7seg PORT MAP (M7, HEX7);
62        END strukturalna;
```

transkoder.vhd :

```
1  -- IMPLEMENTACJA TRANSKODERA
2  LIBRARY ieee;
3  USE ieee.std_logic_1164.all;
4  ENTITY char7seg IS
5          PORT (
6                  C : IN STD_LOGIC_VECTOR(2 DOWNTO 0);
7                  Display : OUT STD_LOGIC_VECTOR(6 DOWNTO 0)
8                  );
9          END char7seg;
10
11 ARCHITECTURE strukturalna OF char7seg IS
12         BEGIN
13         -- . . . do uzupełnienia opis struktury za pomocą równań boolowskich specyfikowanych
            ↪  zgodnie z zasadami języka VHDL
14         Display(0) <= NOT(C(1) OR (NOT C(2) AND C(0)) OR ( C(2) AND NOT C(1) AND NOT C(0)));
15          Display(1) <= NOT(C(2) AND C(0));
16         Display(2) <= NOT C(2);
17         Display(3) <= NOT((C(1) AND NOT C(0)) OR (C(2) AND NOT C(0)) OR ( NOT C(2) AND NOT
            ↪  C(1) AND C(0)));
18         Display(4) <= NOT(C(0) OR C(1) OR C(2));
19         Display(5) <= NOT(C(0) OR C(1) OR C(2));
20         Display(6) <= NOT(C(1) OR C(2));
21
22         END strukturalna;
```

multiplekser.vhd :

```
1  -- implementacja multipleksera 8 do 1 (wektor 3 bitowy)
2  LIBRARY ieee;
3  USE ieee.std_logic_1164.all;
4
5  ENTITY mux3bit8to1 IS
```

```
6         PORT (
7                 S, U0, U1, U2, U3, U4, U5,U6,U7: IN STD_LOGIC_VECTOR(2 DOWNTO 0);
8                 M : OUT STD_LOGIC_VECTOR(2 DOWNTO 0)
9                 );
10 END mux3bit8to1;
11
12 ARCHITECTURE strukturalna OF mux3bit8to1 IS
13         BEGIN
14          -- . . . do uzupełnienia opis struktury za pomocą równań boolowskich
             ↪ specyfikowanych zgodnie z zasadami języka VHDL
15         PROCESS(S, U0, U1, U2, U3, U4, U5,U6,U7)
16          BEGIN
17               CASE S IS
18                     WHEN "000" => M <= U0;
19                     WHEN "001" => M <= U1;
20                     WHEN "011" => M <= U2;
21                     WHEN "010" => M <= U3;
22                     WHEN "110" => M <= U4;
23                     WHEN "111" => M <= U5;
24                     WHEN "101" => M <= U6;
25                     WHEN "100" => M <= U7;
26           END CASE;
27     END PROCESS;
28 END strukturalna;
```

Indeks komentarzy