Maciej Byczko Bartosz Matysiak	Prowadzący: dr inż. Jacek Mazurkiewicz	Numer ćwiczenia 4
PN 10:50 TP	Temat ćwiczenia: Układy wielobitowych wejść i wyjść	Ocena:
Grupa: B	Data wykonania: 8 Listopada 2021r.	

Spis treści

1 Zadanie 1

1.1 Polecenie

Detektor 2-znakowej sekwencji słów 8-bitowych: wejścia 2 znaków 8-bitowych, 1 wyjście 1-bitowe – sekwencja rozpoznana / sekwencja błędna. Źródło danych: początkowo ”guziki” przystawki, potem klawiatura PC via terminal.

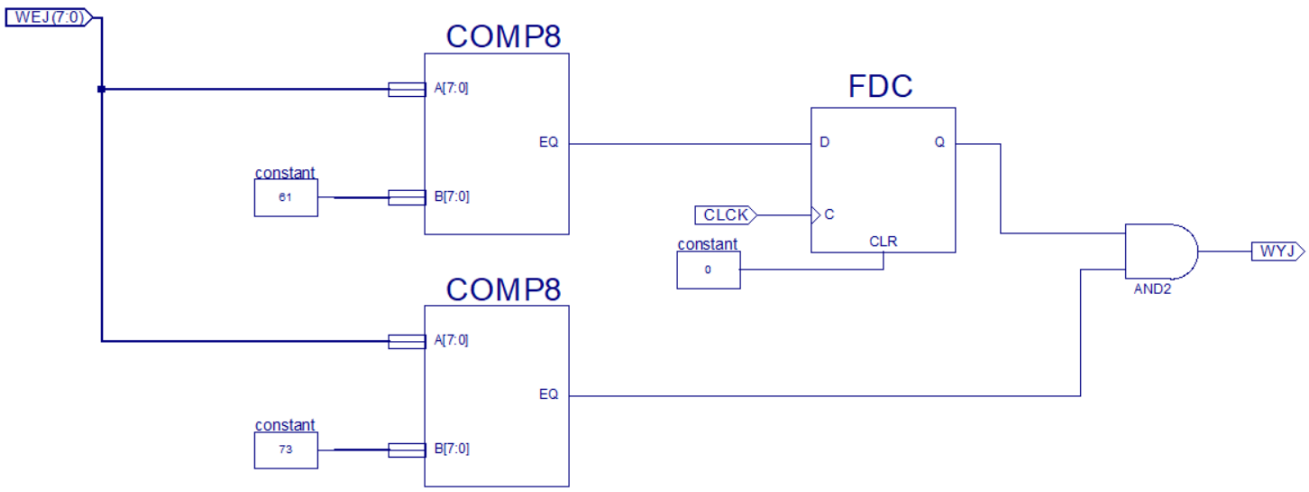
1.2 Rozwiązanie

Do rozwiązania problemu wymagane jest od nas podłączenie dwóch komparatorów 8-bitowych (COMP8), które po pobraniu wartości od użytkownika kolejno po sobie sprawdzają wprowadzone słowa. W zależności od wymagania można wprowadzić także obowiązkowe wprowadzanie wartości w odpowiedniej kolejności.

1.2.1 Schemat stanów

1.2.2 Schematy układu

Schemat dla wersji z przyciskami jako inputem:



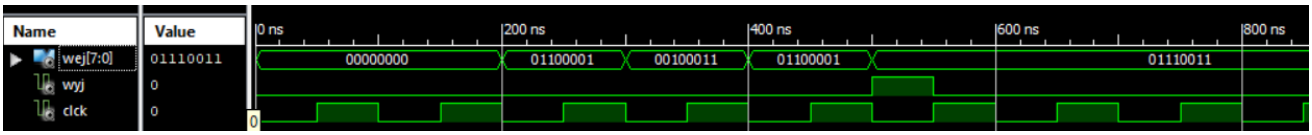
Schemat wersji wykorzystującej klawiaturę jako wejście:

1.2.3 Kod VHDL

```
1  LIBRARY ieee;
2  USE ieee.std_logic_1164.ALL;
3  USE ieee.numeric_std.ALL;
4  LIBRARY UNISIM;
5  USE UNISIM.Vcomponents.ALL;
6  ENTITY scheme_scheme_sch_tb IS
7  END scheme_scheme_sch_tb;
8  ARCHITECTURE behavioral OF scheme_scheme_sch_tb IS
9
10    COMPONENT scheme
11    PORT( WEJ   : IN  STD_LOGIC_VECTOR (7 DOWNTO 0);
12          WYJ   : OUT STD_LOGIC;
13          CLCK  : IN  STD_LOGIC);
14    END COMPONENT;
```

```
15
16      SIGNAL WEJ : STD_LOGIC_VECTOR (7 DOWNTO 0);
17      SIGNAL WYJ : STD_LOGIC;
18      SIGNAL CLCK  : STD_LOGIC := '0';
19
20  BEGIN
21
22      UUT: scheme PORT MAP(
23       WEJ => WEJ,
24       WYJ => WYJ,
25       CLCK => CLCK
26      );
27
28    CLCK <= not CLCK after 50 ns;
29    WEJ <= B"0000_0000", B"0110_0001" after 200 ns, B"0010_0011" after
          300 ns, B"0110_0001" after 400 ns, B"0111_0011" after 500 ns;
30
31  END;
```

1.2.4 Symulacja



1.3 Fizyczna implementacja

1.3.1 Kod UCF

Kod dla wersji z przyciskami jako inputem:

```
# Clocks
NET "CLCK" LOC = "P7" | BUFG = CLK | PERIOD = 5ms HIGH 50%;

# Keys
NET "WEJ(0)" LOC = "P42";
NET "WEJ(1)" LOC = "P40";
NET "WEJ(2)" LOC = "P43";
NET "WEJ(3)" LOC = "P38";
NET "WEJ(4)" LOC = "P37";
NET "WEJ(5)" LOC = "P36";  # shared with ROT_A
NET "WEJ(6)" LOC = "P24";  # shared with ROT_B
NET "WEJ(7)" LOC = "P39";  # GSR

# LEDS
NET "WYJ"  LOC = "P35";
```

Schemat wersji wykorzystującej klawiaturę jako wejście:

2 Zadanie 2

2.1 Polecenie

Układ arytmetyczny pracujący na dwóch argumentach 4-bitowych wyrażonych w kodzie Aikena i generujący stosowny wynik w tymże kodzie.

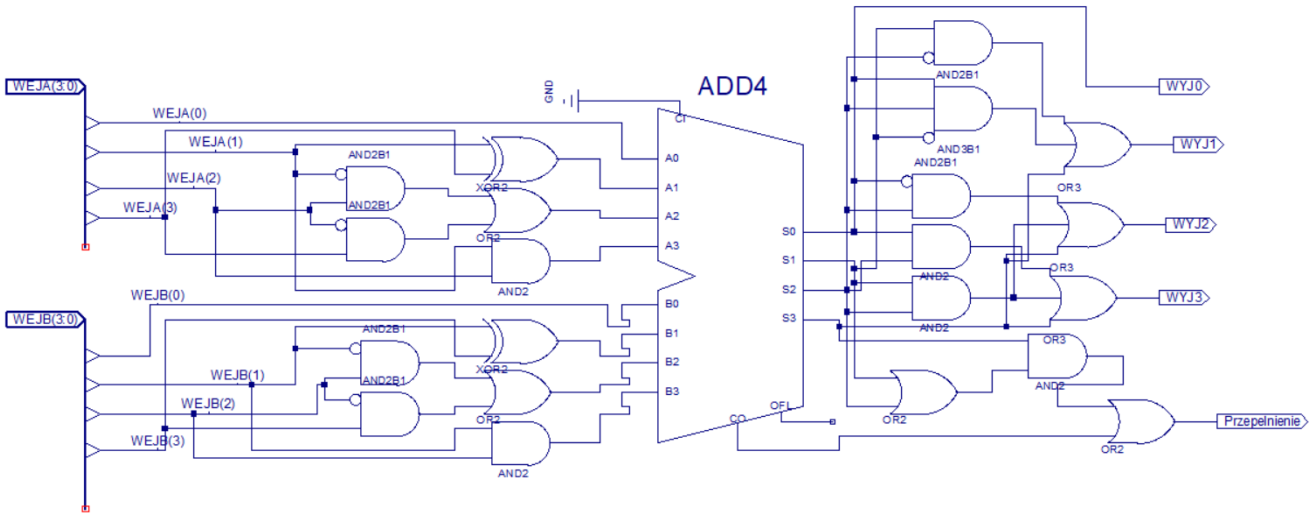
2.2 Rozwiązanie

2.3 Uwagi wstępne

2.3.1 Tabele prawdy

2.3.2 Siatki Karnaugh

2.3.3 Schemat układu

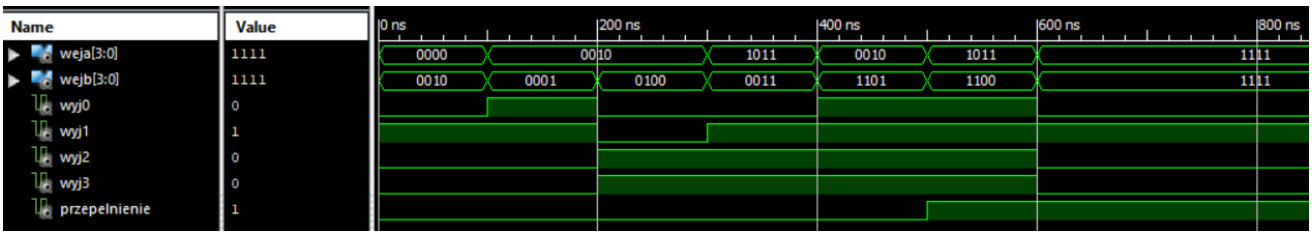


2.3.4 Kod VHDL

```
1  LIBRARY ieee;
2  USE ieee.std_logic_1164.ALL;
3  USE ieee.numeric_std.ALL;
4  LIBRARY UNISIM;
5  USE UNISIM.Vcomponents.ALL;
6  ENTITY aikenAdderScheme_aikenAdderScheme_sch_tb IS
7  END aikenAdderScheme_aikenAdderScheme_sch_tb;
8  ARCHITECTURE behavioral OF aikenAdderScheme_aikenAdderScheme_sch_tb
       IS
9
10     COMPONENT aikenAdderScheme
11     PORT( WYJ0 : OUT STD_LOGIC;
12           WEJA   : IN  STD_LOGIC_VECTOR (3 DOWNTO 0);
13           WEJB   : IN  STD_LOGIC_VECTOR (3 DOWNTO 0);
14           Przepelnienie : OUT STD_LOGIC;
15           WYJ1   : OUT STD_LOGIC;
16           WYJ2   : OUT STD_LOGIC;
17           WYJ3   : OUT STD_LOGIC);
18     END COMPONENT;
```

```
19
20      SIGNAL WYJ0   : STD_LOGIC;
21      SIGNAL WEJA   : STD_LOGIC_VECTOR (3 DOWNTO 0);
22      SIGNAL WEJB   : STD_LOGIC_VECTOR (3 DOWNTO 0);
23      SIGNAL Przepelnienie : STD_LOGIC;
24      SIGNAL WYJ1   : STD_LOGIC;
25      SIGNAL WYJ2   : STD_LOGIC;
26      SIGNAL WYJ3   : STD_LOGIC;
27
28  BEGIN
29
30      UUT: aikenAdderScheme PORT MAP(
31        WYJ0 => WYJ0,
32        WEJA => WEJA,
33        WEJB => WEJB,
34         Przepelnienie => Przepelnienie,
35        WYJ1 => WYJ1,
36        WYJ2 => WYJ2,
37        WYJ3 => WYJ3
38       );
39
40    WEJA <= "0000", "0010" after 100 ns, "0010" after 200 ns, "1011"
          after 300 ns, "0010" after 400 ns, "1011" after 500 ns, "1111"
          after 600 ns;
41    WEJB <= "0010", "0001" after 100 ns, "0100" after 200 ns, "0011"
          after 300 ns, "1101" after 400 ns, "1100" after 500 ns, "1111"
          after 600 ns;
42
43  END;
```

2.3.5 Symulacja



2.4 Fizyczna implementacja

2.4.1 Kod UCF

```
1  # Keys
2  NET "WEJA(0)" LOC = "P42";
3  NET "WEJA(1)" LOC = "P40";
4  NET "WEJA(2)" LOC = "P43";
5  NET "WEJA(3)" LOC = "P38";
6  NET "WEJB(0)" LOC = "P37";
7  NET "WEJB(1)" LOC = "P36";  # shared with ROT_A
8  NET "WEJB(2)" LOC = "P24";  # shared with ROT_B
9  NET "WEJB(3)" LOC = "P39";  # GSR
```

```
10
11 # LEDS
12 NET "WYJ0"   LOC = "P35";
13 NET "WYJ1"   LOC = "P29";
14 NET "WYJ2"   LOC = "P33";
15 NET "WYJ3"   LOC = "P34";
16 NET "Przepelnienie"   LOC = "P28";
```

3 Zadanie 3

3.1 Polecenie

Konwerter cyfry szesnastkowej zapisanej na czterech bitach od 0 do 9, A do F na kod ASCII tej cyfry – wyjście 8-bitowe. Prezentacja wyniku na diodach przystawki, potem na wyświetlaczu 7-segmentowym.

3.2 Rozwiązanie

3.2.1 Schemat stanów

3.2.2 Tabela prawdy

3.2.3 Siatki Karnaugh

3.2.4 Schemat układu

3.2.5 Kod VHDL

3.2.6 Symulacja

3.3 Fizyczna implementacja

3.3.1 Kod UCF

4 Zadanie 4

4.1 Polecenie

Komparator dwóch 4-bitowych cyfr: 2 wejścia po 4 bity, 3 wyjścia 1-bitowe: mniejszy, większy, równy pracujący w kodzie Aikena.