Maciej Byczko Bartosz Matysiak	Prowadzący: dr inż. Jacek Mazurkiewicz	Numer ćwiczenia 5
PN 10:50 TP	Temat ćwiczenia: Układy Kombinacyjne i Sekwencyjne w VHDL-u	Ocena:
Grupa: B	Data wykonania: 6 Grudnia 2021r.	

Spis treści

1 Zadanie 1

1.1 Polecenie

Implementacja funkcji logicznej $G(w, x, y, z) = \prod(0, 2, 3, 4, 6, 7, 9, 11, 12, 13, 15)$ w VHDL-u za pomocą:

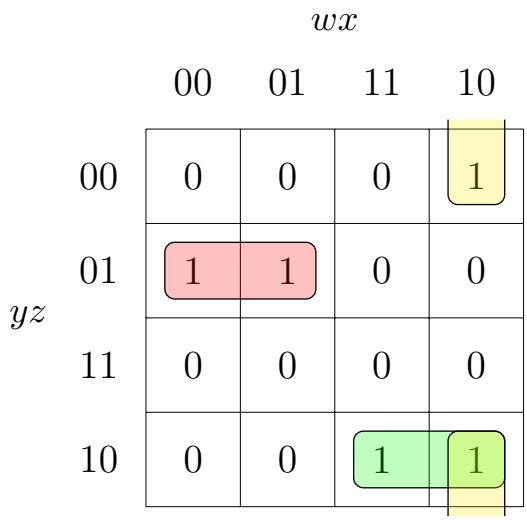
1. Zapis równań boolowskich
2. Metoda zapisu tablicowego

1.2 Rozwiązanie

1.2.1 Tabela prawdy

Kod dziesiętny	w	x	y	z	G
0	0	0	0	0	0
1	0	0	0	1	1
2	0	0	1	0	0
3	0	0	1	1	0
4	0	1	0	0	0
5	0	1	0	1	1
6	0	1	1	0	0
7	0	1	1	1	0
8	1	0	0	0	1
9	1	0	0	1	0
10	1	0	1	0	1
11	1	0	1	1	0
12	1	1	0	0	0
13	1	1	0	1	0
14	1	1	1	0	1
15	1	1	1	1	0

1.2.2 Siatka Karnaugh



Rysunek 1: $Wyj_G = w\overline{x}\overline{z} + \overline{w}\overline{y}z + wy\overline{z}$

1.3 Zapis za pomocą równań boolowskich

1.3.1 Kod VHDL

```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3
4  entity gfunction is
5      Port ( W : in  STD_LOGIC;
6             X : in  STD_LOGIC;
7             Y : in  STD_LOGIC;
8             Z : in  STD_LOGIC;
9             S : out  STD_LOGIC);
10 end gfunction;
11
12 architecture Dataflow of gfunction is
13
14 begin
15   S <= (not Z and W and not X) or (not Y and Z and not W) or (Y and
       not Z and W);
16 end Dataflow;
```

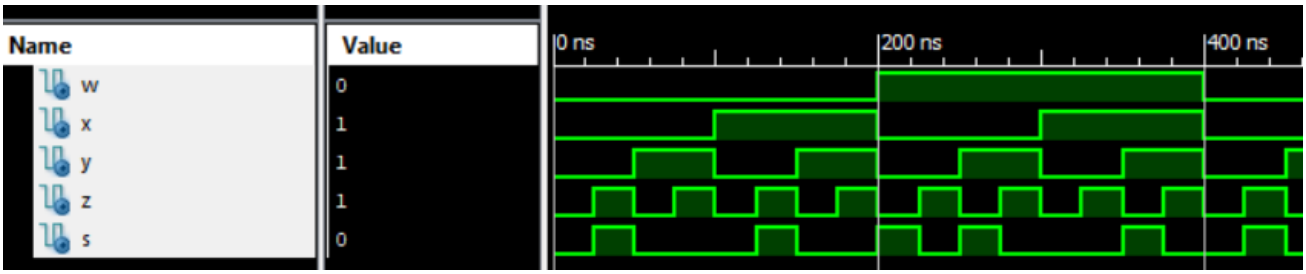
1.4 Kod VHDL TestBench

```
1  LIBRARY ieee;
2  USE ieee.std_logic_1164.ALL;
3
4  ENTITY gfunctionTestbench IS
5  END gfunctionTestbench;
6
7  ARCHITECTURE behavior OF gfunctionTestbench IS
```

```
8
9       -- Component Declaration for the Unit Under Test (UUT)
10      COMPONENT gfunction
11      PORT(
12           W : IN  std_logic;
13           X : IN  std_logic;
14           Y : IN  std_logic;
15           Z : IN  std_logic;
16           S : OUT  std_logic
17          );
18      END COMPONENT;
19
20     --Inputs
21      signal W : std_logic := '0';
22      signal X : std_logic := '0';
23      signal Y : std_logic := '0';
24      signal Z : std_logic := '0';
25
26    --Outputs
27      signal S : std_logic;
28  BEGIN
29
30    -- Instantiate the Unit Under Test (UUT)
31      uut: gfunction PORT MAP (
32             W => W,
33             X => X,
34             Y => Y,
35             Z => Z,
36             S => S
37            );
38     -- Stimulus process
39      stim_procw: process
40      begin
41         wait for 200 ns;
42       w <= not w;
43      end process;
44
45    stim_procx: process
46     begin
47         wait for 100 ns;
48       x <= not x;
49     end process;
50
51    stim_procy: process
52     begin
53         wait for 50 ns;
54       y <= not y;
55     end process;
56
57    stim_procz: process
58     begin
```

```
59        wait for 25 ns;
60      z <= not z;
61    end process;
62
63 END;
```

1.4.1 Symulacja



1.4.2 Fizyczna implementacja (Kod UCF)

```
1 # Keys
2 NET "W" LOC = "P42";
3 NET "X" LOC = "P40";
4 NET "Y" LOC = "P43";
5 NET "Z" LOC = "P38";
6
7 # LEDS
8 NET "S"  LOC = "P35";
```

1.5 Zapis tablicowy

1.5.1 Kod VHDL

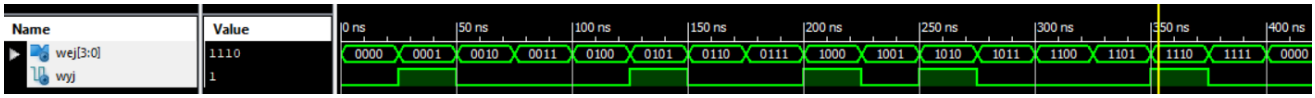
```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3
4  entity gfunctiontruthtable is
5      Port ( WEJ : in  STD_LOGIC_VECTOR (3 downto 0);
6             WYJ : out  STD_LOGIC);
7  end gfunctiontruthtable;
8
9  architecture Dataflow of gfunctiontruthtable is
10
11 begin
12   with WEJ select
13   WYJ <= '1' when "0001" | "0101" | "1000" | "1010" | "1110",
14          '0' when others;
15 end Dataflow;
```

1.6 Kod VHDL TestBench

```
1 LIBRARY ieee;
2 USE ieee.std_logic_1164.ALL;
3
4 ENTITY gFunctionTestBenchViaTruthTable IS
```

```
5  END gFunctionTestBenchViaTruthTable;
6
7  ARCHITECTURE behavior OF gFunctionTestBenchViaTruthTable IS
8
9      -- Component Declaration for the Unit Under Test (UUT)
10     COMPONENT gfunctiontruthtable
11     PORT(
12          WEJ : IN  std_logic_vector(3 downto 0);
13          WYJ : OUT  std_logic
14         );
15     END COMPONENT;
16
17    --Inputs
18    signal WEJ : std_logic_vector(3 downto 0) := (others => '0');
19
20   --Outputs
21    signal WYJ : std_logic;
22 BEGIN
23
24   -- Instantiate the Unit Under Test (UUT)
25    uut: gfunctiontruthtable PORT MAP (
26            WEJ => WEJ,
27            WYJ => WYJ
28          );
29    -- Stimulus process
30    stim_proc0: process
31    begin
32       wait for 25 ns;
33     WEJ(0) <= not WEJ(0);
34    end process;
35
36   stim_proc1: process
37    begin
38       wait for 50 ns;
39     WEJ(1) <= not WEJ(1);
40    end process;
41
42   stim_proc2: process
43    begin
44       wait for 100 ns;
45     WEJ(2) <= not WEJ(2);
46    end process;
47
48   stim_proc3: process
49    begin
50       wait for 200 ns;
51     WEJ(3) <= not WEJ(3);
52    end process;
53
54 END;
```

1.6.1 Symulacja



1.6.2 Fizyczna implementacja (Kod UCF)

```
# Keys
NET "WEJ(0)" LOC = "P42";
NET "WEJ(1)" LOC = "P40";
NET "WEJ(2)" LOC = "P43";
NET "WEJ(3)" LOC = "P38";

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

2 Zadanie 2

2.1 Polecenie

Implementacja układu translatora kodu **4-bit kod NKB na 4-bit kod Aikena** w VHDL-u za pomocą:

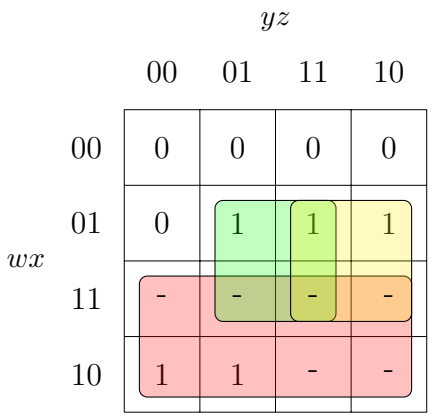
1. Zapis równań boolowskich
2. Metoda zapisu tablicowego

2.2 Rozwiązanie

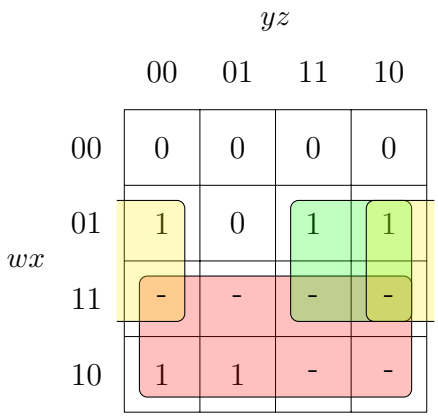
2.2.1 Tabela prawdy

Kod dziesiętny	NKB				Kod Aikena			
	w	x	y	z	w	x	y	z
0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1
2	0	0	1	0	0	0	1	0
3	0	0	1	1	0	0	1	1
4	0	1	0	0	0	1	0	0
5	0	1	0	1	1	0	1	1
6	0	1	1	0	1	1	0	0
7	0	1	1	1	1	1	0	1
8	1	0	0	0	1	1	1	0
9	1	0	0	1	1	1	1	1
10	1	0	1	0	-	-	-	-
11	1	0	1	1	-	-	-	-
12	1	1	0	0	-	-	-	-
13	1	1	0	1	-	-	-	-
14	1	1	1	0	-	-	-	-
15	1	1	1	1	-	-	-	-

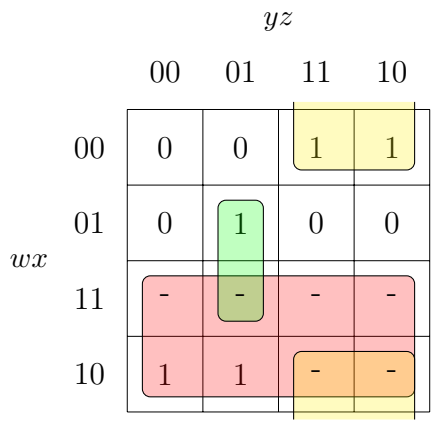
2.2.2 Siatki Karnaugh



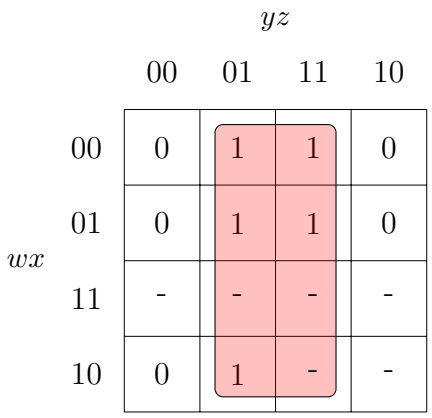
$$w_A = xz + xy + w$$



$$x_A = x\overline{z} + xy + w$$



$$y_A = \overline{x}y + x\overline{y}z + w$$



$$z_A = z$$

2.3 Zapis za pomocą równań boolowskich

2.3.1 Kod VHDL

```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3
4  entity boolean_aiken is
5      Port ( w : in  STD_LOGIC;
6             x : in  STD_LOGIC;
7             y : in  STD_LOGIC;
8             z : in  STD_LOGIC;
9             a : out  STD_LOGIC;
10            b : out  STD_LOGIC;
11            c : out  STD_LOGIC;
12            d : out  STD_LOGIC);
13 end boolean_aiken;
14
15 architecture Dataflow of boolean_aiken is
16
17 begin
18 a <= (x and z) or (x and y) or (w);
19 b <= (x and not z) or (x and y) or (w);
20 c <= (not x and y) or (x and not y and z) or (w);
21 d <= (z);
22
23 end Dataflow;
```

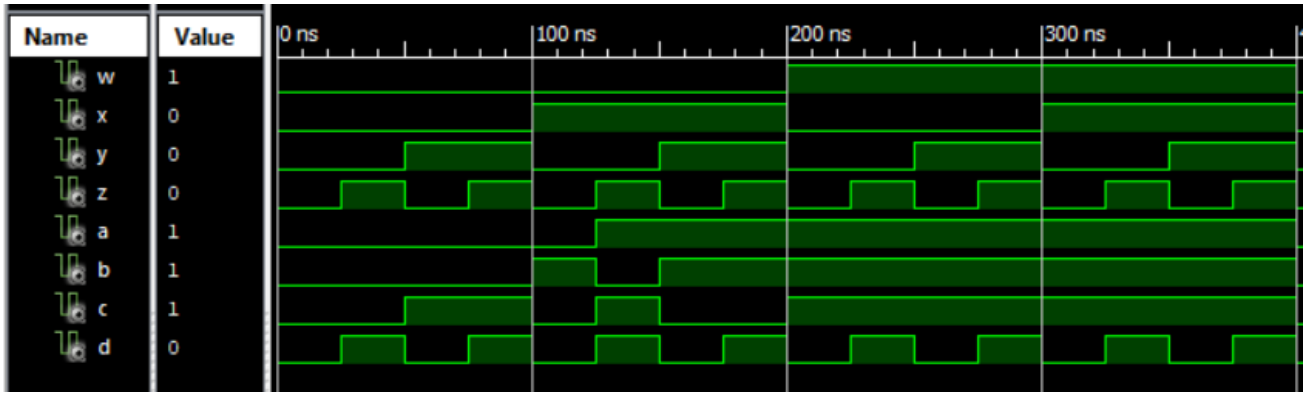
2.4 Kod VHDL TestBench

```
1  LIBRARY ieee;
2  USE ieee.std_logic_1164.ALL;
3
4  ENTITY boolean_aiken_testbench IS
5  END boolean_aiken_testbench;
6
7  ARCHITECTURE behavior OF boolean_aiken_testbench IS
```

```
8
9       -- Component Declaration for the Unit Under Test (UUT)
10      COMPONENT boolean_aiken
11      PORT(
12           w : IN  std_logic;
13           x : IN  std_logic;
14           y : IN  std_logic;
15           z : IN  std_logic;
16           a : OUT  std_logic;
17           b : OUT  std_logic;
18           c : OUT  std_logic;
19           d : OUT  std_logic
20          );
21      END COMPONENT;
22
23     --Inputs
24      signal w : std_logic := '0';
25      signal x : std_logic := '0';
26      signal y : std_logic := '0';
27      signal z : std_logic := '0';
28
29     --Outputs
30      signal a : std_logic;
31      signal b : std_logic;
32      signal c : std_logic;
33      signal d : std_logic;
34  BEGIN
35
36     -- Instantiate the Unit Under Test (UUT)
37      uut: boolean_aiken PORT MAP (
38             w => w,
39             x => x,
40             y => y,
41             z => z,
42             a => a,
43             b => b,
44             c => c,
45             d => d
46            );
47
48     -- Stimulus process
49      stim_procw: process
50      begin
51         wait for 200 ns;
52       w <= not w;
53      end process;
54
55     stim_procx: process
56      begin
57         wait for 100 ns;
58       x <= not x;
```

```
59     end process;
60
61   stim_procy: process
62     begin
63         wait for 50 ns;
64      y <= not y;
65     end process;
66
67   stim_procz: process
68     begin
69         wait for 25 ns;
70      z <= not z;
71     end process;
72 END;
```

2.4.1 Symulacja



2.4.2 Fizyczna implementacja (Kod UCF)

```
1  # Keys
2  NET "Z" LOC = "P42";
3  NET "Y" LOC = "P40";
4  NET "X" LOC = "P43";
5  NET "W" LOC = "P38";
6
7  # LEDS
8  NET "D"  LOC = "P35";
9  NET "C"  LOC = "P29";
10 NET "B"  LOC = "P33";
11 NET "A"  LOC = "P34";
```

2.5 Zapis tablicowy

2.5.1 Kod VHDL

```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3
4  entity table_aiken is
5      Port ( WEJ : in  STD_LOGIC_VECTOR (3 downto 0);
6             A : out  STD_LOGIC;
```

```
7             B : out  STD_LOGIC;
8             C : out  STD_LOGIC;
9             D : out  STD_LOGIC);
10 end table_aiken;
11
12 architecture Dataflow of table_aiken is
13
14 begin
15 with WEJ select
16   A <= '0' when "0000" | "0001" | "0010" | "0011" | "0100",
17        '1' when others;
18 with WEJ select
19   B <= '0' when "0000" | "0001" | "0010" | "0011" | "0101",
20        '1' when others;
21 with WEJ select
22   C <= '0' when "0000" | "0001" | "0100" | "0110" | "0111",
23        '1' when others;
24 with WEJ select
25   D <= '0' when "0000" | "0010" | "0100" | "0110" | "1000",
26        '1' when others;
27
28 end Dataflow;
```

2.6 Kod VHDL TestBench

```
1  LIBRARY ieee;
2  USE ieee.std_logic_1164.ALL;
3
4  ENTITY table_aiken_testbench IS
5  END table_aiken_testbench;
6
7  ARCHITECTURE behavior OF table_aiken_testbench IS
8
9      -- Component Declaration for the Unit Under Test (UUT)
10     COMPONENT table_aiken
11     PORT(
12          WEJ : IN  std_logic_vector(3 downto 0);
13          A : OUT  std_logic;
14          B : OUT  std_logic;
15          C : OUT  std_logic;
16          D : OUT  std_logic
17         );
18     END COMPONENT;
19
20
21    --Inputs
22     signal WEJ : std_logic_vector(3 downto 0) := (others => '0');
23
24   --Outputs
25     signal A : std_logic;
26     signal B : std_logic;
27     signal C : std_logic;
```

```
28    signal D : std_logic;
29
30  BEGIN
31
32    -- Instantiate the Unit Under Test (UUT)
33    uut: table_aiken PORT MAP (
34            WEJ => WEJ,
35            A => A,
36            B => B,
37            C => C,
38            D => D
39          );
40
41    -- Stimulus process
42    stim_proc0: process
43    begin
44        wait for 25 ns;
45     WEJ(0) <= not WEJ(0);
46    end process;
47
48   stim_proc1: process
49    begin
50        wait for 50 ns;
51     WEJ(1) <= not WEJ(1);
52    end process;
53
54   stim_proc2: process
55    begin
56        wait for 100 ns;
57     WEJ(2) <= not WEJ(2);
58    end process;
59
60   stim_proc3: process
61    begin
62        wait for 200 ns;
63     WEJ(3) <= not WEJ(3);
64    end process;
65
66  END;
```

2.6.1 Symulacja

2.6.2 Fizyczna implementacja (Kod UCF)

```
1  # Keys
2  NET "WEJ(0)" LOC = "P42";
```

```
3  NET "WEJ(1)" LOC = "P40";
4  NET "WEJ(2)" LOC = "P43";
5  NET "WEJ(3)" LOC = "P38";
6
7  # LEDS
8  NET "D"   LOC = "P35";
9  NET "C"   LOC = "P29";
10 NET "B"   LOC = "P33";
11 NET "A"   LOC = "P34";
```

3 Zadanie 3

3.1 Polecenie

Detektor sekwencji 11011, automat Mealy-ego, jedno wejście, jedno wyjście, brak resetu, sekwencja prawidłowa 5-bitowa w VHDL-u jako maszyna stanów.

3.2 Rozwiązanie

3.2.1 Opis symboliki

Alfabet wejściowy

- $z_0 = 0$
- $z_1 = 1$

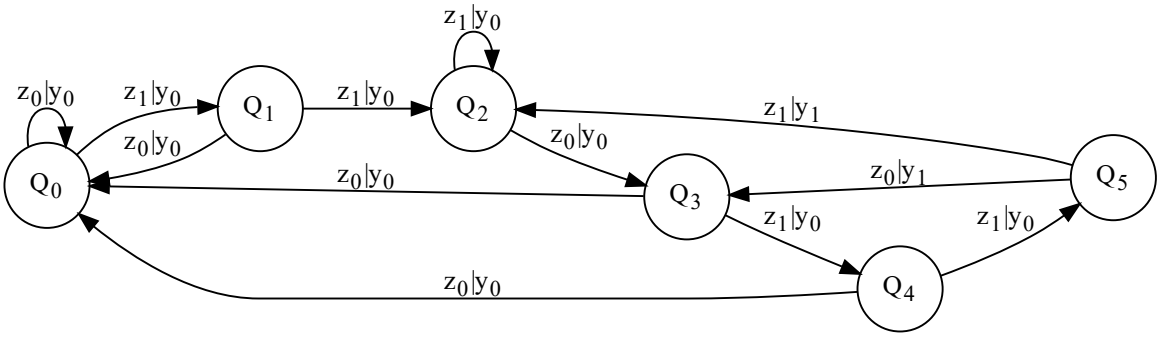
Stany wewnętrzne

- q_0 - stan początkowy | wprowadzono niepoprawny ciąg bitów
- q_1 - wprowadzono pierwszą cyfrę prawidłowego ciągu
- q_2 - wprowadzono drugą cyfrę prawidłowego ciągu
- q_3 - wprowadzono trzecią cyfrę prawidłowego ciągu
- q_4 - wprowadzono czwartą cyfrę prawidłowego ciągu
- q_5 - wprowadzono poprawną sekwencję

Alfabet wyjścia

- y_0 - Wprowadzony ciąg nadal jest niepoprawny
- y_1 - Wprowadzono poprawną sekwencję

3.2.2 Schemat grafowy



3.2.3 Tabela prawdy

S	Q(t)			Z	Q(t+1)			Y
	Q_2	Q_1	Q_0		Q_2	Q_1	Q_0	
Q_0	0	0	0	0	0	0	0	0
Q_0	0	0	0	1	0	0	1	0
Q_1	0	0	1	0	0	0	0	0
Q_1	0	0	1	1	0	1	0	0
Q_2	0	1	0	0	0	1	1	0
Q_2	0	1	0	1	0	1	0	0
Q_3	0	1	1	0	0	0	0	0
Q_3	0	1	1	1	1	0	0	0
Q_4	1	0	0	0	0	0	0	0
Q_4	1	0	0	1	1	0	1	0
Q_5	1	0	1	0	0	1	1	1
Q_5	1	0	1	1	0	1	0	1
-	1	1	0	0	-	-	-	-
-	1	1	0	1	-	-	-	-
-	1	1	1	0	-	-	-	-
-	1	1	1	1	-	-	-	-

3.2.4 Kod VHDL

```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3
4  entity detectormodule is
5      Port ( Z : in  STD_LOGIC;
6             CLK : in  STD_LOGIC;
7             Y : out  STD_LOGIC);
8  end detectormodule;
9
10 architecture Behavioral of detectormodule is
11
12 type state_type is (Q0, Q1, Q2, Q3, Q4, Q5);
13 signal state: state_type := Q0;
14 signal next_state : state_type := Q0;
15
16 begin
17   process1 : process(CLK)
18   begin
19     if rising_edge(CLK) then
20       state <= next_state;
21     end if;
22   end process process1;
23
24   process2: process(state, Z)
25   begin
26     next_state <= state; -- default
27     case state is
28       when Q0 =>
29         if Z = '1' then
30           next_state <= Q1;
31         end if;
32       when Q1 =>
33         if Z = '0' then
34           next_state <= Q0;
35         else
36           next_state <= Q2;
37         end if;
38       when Q2 =>
39         if Z = '0' then
40           next_state <= Q3;
41         end if;
42       when Q3 =>
43         if Z = '0' then
44           next_state <= Q0;
45         else
46           next_state <= Q4;
47         end if;
48       when Q4 =>
49         if Z = '0' then
```

```
50            next_state <= Q0;
51          else
52            next_state <= Q5;
53          end if;
54        when Q5 =>
55          if Z = '0' then
56            next_state <= Q3;
57          else
58            next_state <= Q2;
59          end if;
60      end case;
61    end process process2;
62
63    -- process3
64    Y <= '1' when state = Q5 else '0'; -- brak roznicy miedzy
         automatem moorea i mealyego
65
66  end Behavioral;
```

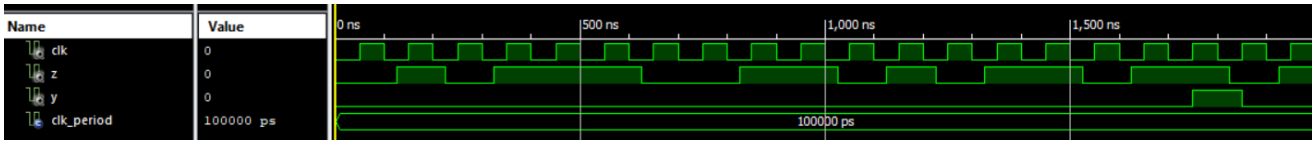
3.2.5 Kod VHDL TestBench

```
1  LIBRARY ieee;
2  USE ieee.std_logic_1164.ALL;
3
4  ENTITY detectorTestBench IS
5  END detectorTestBench;
6
7  ARCHITECTURE behavior OF detectorTestBench IS
8
9      -- Component Declaration for the Unit Under Test (UUT)
10     COMPONENT detectormodule
11     PORT(
12          Z : IN  std_logic;
13          CLK : IN  std_logic;
14          Y : OUT  std_logic
15         );
16     END COMPONENT;
17
18    --Inputs
19     signal Z : std_logic := '0';
20     signal CLK : std_logic := '0';
21
22    --Outputs
23     signal Y : std_logic;
24
25     -- Clock period definitions
26     constant CLK_period : time := 100 ns;
27
28 BEGIN
29
30    -- Instantiate the Unit Under Test (UUT)
31     uut: detectormodule PORT MAP (
```

```
32            Z => Z,
33            CLK => CLK,
34            Y => Y
35          );
36
37    -- Clock process definitions
38    CLK_process :process
39    begin
40     CLK <= '0';
41     wait for CLK_period/2;
42     CLK <= '1';
43     wait for CLK_period/2;
44    end process;
45
46
47    -- Stimulus process
48    stim_proc: process
49    begin
50     -- initial : '0'
51          wait for 125 ns;
52
53     Z <= '1';
54     wait for 100 ns;
55
56     Z <= '0';
57     wait for 100 ns;
58
59     Z <= '1';
60     wait for 300 ns;
61
62     Z <= '0';
63     wait for 200 ns;
64
65     Z <= '1';
66     wait for 200 ns;
67
68     Z <= '0';
69     wait for 100 ns;
70
71     Z <= '1';
72     wait for 100 ns;
73
74     Z <= '0';
75     wait for 100 ns;
76
77     Z <= '1';
78     wait for 200 ns;
79
80     Z <= '0';
81     wait for 100 ns;
82
```

```
83      Z <= '1';
84      wait for 200 ns;
85
86      Z <= '0';
87      wait for 100 ns;
88
89      Z <= '1';
90      wait for 100 ns;
91    end process;
92
93  END;
```

3.2.6 Symulacja



3.2.7 Fizyczna implementacja (Kod UCF)

```
1  # Clocks
2  NET "CLK" LOC = "P7" | BUFG = CLK | PERIOD = 5ms HIGH 50%;
3
4  # Keys
5  NET "Z" LOC = "P42";
6
7  # LEDS
8  NET "Y"  LOC = "P35";
```

3.3 Implementacja wyświetlacza LED

3.3.1 Kod VHDL

```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3
4  entity detector_led is
5      Port ( Z : in  STD_LOGIC;
6             Y : out  STD_LOGIC;
7             LED : out  STD_LOGIC_VECTOR (7 downto 0);
8             CLK : in  STD_LOGIC);
9  end detector_led;
10
11 architecture Behavioral of detector_led is
12
13 type state_type is (Q0, Q1, Q2, Q3, Q4, Q5);
14 signal state: state_type := Q0;
15 signal next_state : state_type := Q0;
16
17 begin
18
19 process_clk : process(CLK)
```

```
20    begin
21      if rising_edge(CLK) then
22        state <= next_state;
23      end if;
24    end process process_clk;
25
26  process2: process(state, Z)
27    begin
28      next_state <= state; -- default
29      case state is
30        when Q0 =>
31          if Z = '1' then
32            next_state <= Q1;
33          end if;
34        when Q1 =>
35          if Z = '0' then
36            next_state <= Q0;
37          else
38            next_state <= Q2;
39          end if;
40        when Q2 =>
41          if Z = '0' then
42            next_state <= Q3;
43          end if;
44        when Q3 =>
45          if Z = '0' then
46            next_state <= Q0;
47          else
48            next_state <= Q4;
49          end if;
50        when Q4 =>
51          if Z = '0' then
52            next_state <= Q0;
53          else
54            next_state <= Q5;
55          end if;
56        when Q5 =>
57          if Z = '0' then
58            next_state <= Q3;
59          else
60            next_state <= Q2;
61          end if;
62      end case;
63    end process process2;
64
65    -- process3
66    Y <= '1' when state = Q5 else '0'; -- brak roznicy miedzy
       automatem moorea i mealyego
67
68      process3: process(state)
69          begin
```

```
70              case state is
71              when Q5 =>
72                  LED <= "10010010";
73              when Q4 =>
74                  LED <= "10011001";
75              when Q3 =>
76                  LED <= "10110000";
77              when Q2 =>
78                  LED <= "10100100";
79              when Q1 =>
80                  LED <= "11111001";
81              when Q0 =>
82                  LED <= "11000000";
83          end case;
84      end process process3;
85
86
87  end Behavioral;
```

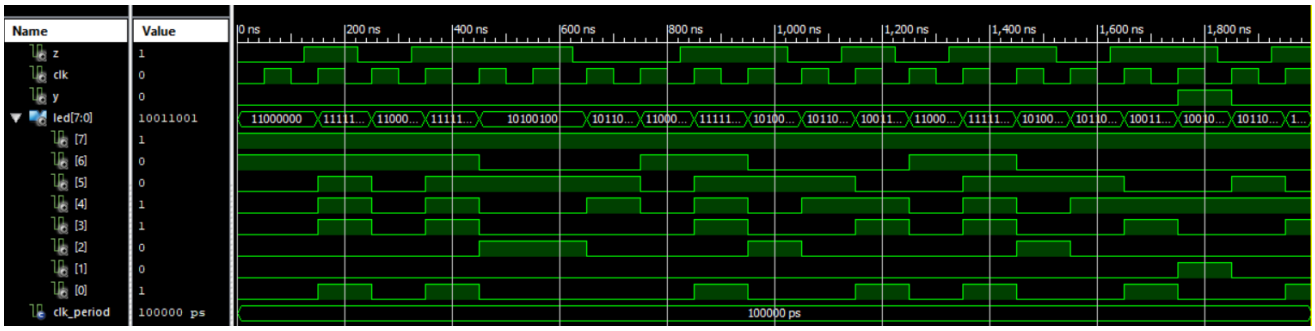
3.3.2 Kod VHDL TestBench

```
1  LIBRARY ieee;
2  USE ieee.std_logic_1164.ALL;
3
4  ENTITY detector_led_testbench IS
5  END detector_led_testbench;
6
7  ARCHITECTURE behavior OF detector_led_testbench IS
8
9      -- Component Declaration for the Unit Under Test (UUT)
10
11     COMPONENT detector_led
12     PORT(
13          Z : IN  std_logic;
14          Y : OUT  std_logic;
15          LED : OUT  std_logic_vector(7 downto 0);
16          CLK : IN  std_logic
17         );
18     END COMPONENT;
19
20
21    --Inputs
22     signal Z : std_logic := '0';
23     signal CLK : std_logic := '0';
24
25   --Outputs
26     signal Y : std_logic;
27     signal LED : std_logic_vector(7 downto 0);
28
29     -- Clock period definitions
30     constant CLK_period : time := 100 ns;
31
```

```
32 BEGIN
33
34   -- Instantiate the Unit Under Test (UUT)
35    uut: detector_led PORT MAP (
36            Z => Z,
37            Y => Y,
38            LED => LED,
39            CLK => CLK
40          );
41
42    -- Clock process definitions
43    CLK_process :process
44    begin
45     CLK <= '0';
46     wait for CLK_period/2;
47     CLK <= '1';
48     wait for CLK_period/2;
49    end process;
50
51
52
53    -- Stimulus process
54    stim_proc: process
55    begin
56     -- initial : '0'
57        wait for 125 ns;
58
59     Z <= '1';
60     wait for 100 ns;
61
62     Z <= '0';
63     wait for 100 ns;
64
65     Z <= '1';
66     wait for 300 ns;
67
68     Z <= '0';
69     wait for 200 ns;
70
71     Z <= '1';
72     wait for 200 ns;
73
74     Z <= '0';
75     wait for 100 ns;
76
77     Z <= '1';
78     wait for 100 ns;
79
80     Z <= '0';
81     wait for 100 ns;
82
```

```
83      Z <= '1';
84      wait for 200 ns;
85
86      Z <= '0';
87      wait for 100 ns;
88
89      Z <= '1';
90      wait for 200 ns;
91
92      Z <= '0';
93      wait for 100 ns;
94
95      Z <= '1';
96      wait for 100 ns;
97    end process;
98
99  END;
```

3.3.3 Symulacja



3.3.4 Fizyczna implementacja (Kod UCF)

```
1  # Clocks
2  NET "CLK" LOC = "P7" | BUFG = CLK | PERIOD = 5ms HIGH 50%;
3
4  # Keys
5  NET "Z" LOC = "P42";
6
7  # LEDS
8  NET "Y"  LOC = "P35";
9
10 # DISPL. 7-SEG
11 NET "LED(0)" LOC = "P12"; # Seg. A; shared with LED<10>
12 NET "LED(1)" LOC = "P13"; # Seg. B; shared with LED<8>
13 NET "LED(2)" LOC = "P22"; # Seg. C; shared with LED<12>
14 NET "LED(3)" LOC = "P19"; # Seg. D; shared with LED<14>
15 NET "LED(4)" LOC = "P14"; # Seg. E; shared with LED<15>
16 NET "LED(5)" LOC = "P11"; # Seg. F; shared with LED<9>
17 NET "LED(6)" LOC = "P20"; # Seg. G; shared with LED<13>
18 NET "LED(7)" LOC = "P18"; # Seg. DP;shared with LED<11>
```

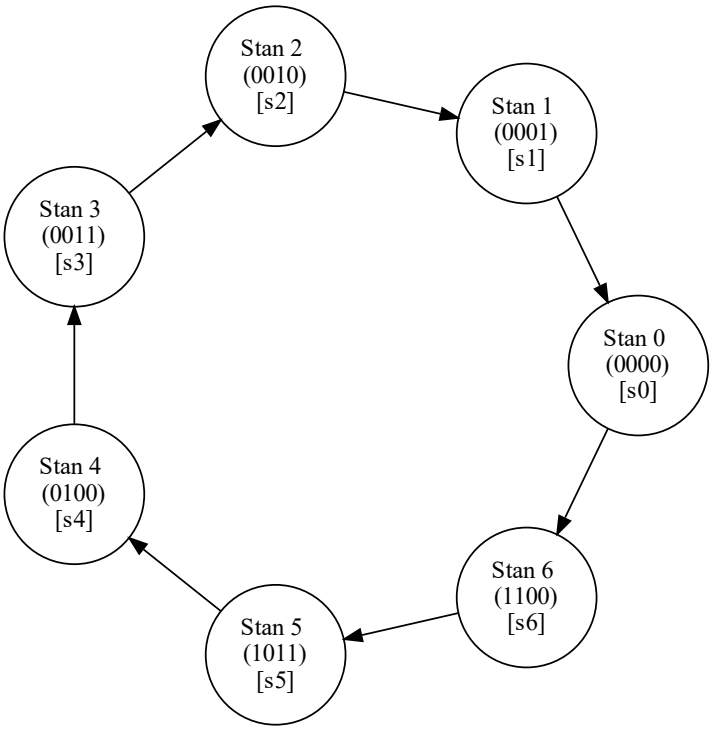
4 Zadanie 4

4.1 Polecenie

Zaprojektować licznik synchroniczny liczący w tył na bazie kodu Aikena w zakresie 0-6 (mod 7) jako maszyna stanów.

4.2 Rozwiązanie

4.2.1 Schemat stanów



4.2.2 Tabela prawdy

n	Q(t)				Q(t+1)			
	Q_3	Q_2	Q_1	Q_0	Q_3	Q_2	Q_1	Q_0
0	0	0	0	0	1	1	0	0
1	0	0	0	1	0	0	0	0
2	0	0	1	0	0	0	0	1
3	0	0	1	1	0	0	1	0
4	0	1	0	0	0	0	1	1
5	1	0	1	1	0	1	0	0
6	1	1	0	0	1	0	1	1

4.2.3 Kod VHDL

```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3
4  entity aiken_counter is
5      Port ( CLK : in  STD_LOGIC;
6             Q : out  STD_LOGIC_VECTOR (3 downto 0));
7  end aiken_counter;
8
9  architecture Behavioral of aiken_counter is
10
11   type state_type is (s6, s5, s4, s3, s2, s1, s0);
12   signal state : state_type := s6;
13
14 begin
15
16   processc: process(CLK)
17   begin
18     if rising_edge(CLK) then
19       case state is
20         when s6 => state <= s5 after 5ns;
21         when s5 => state <= s4 after 5ns;
22         when s4 => state <= s3 after 5ns;
23         when s3 => state <= s2 after 5ns;
24         when s2 => state <= s1 after 5ns;
25         when s1 => state <= s0 after 5ns;
26         when s0 => state <= s6 after 5ns;
```

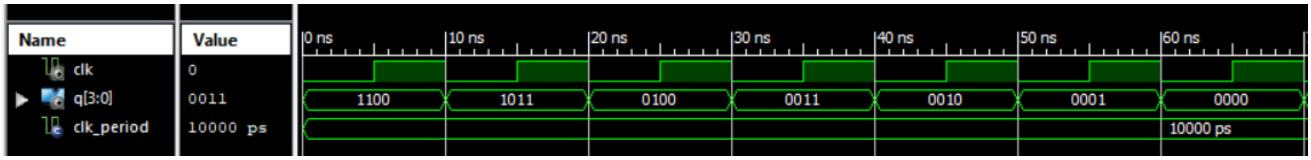
```
      end case;
    end if;
  end process processc;

  Q(3) <= '1' when state = s6 or state = s5 else '0';
  Q(2) <= '1' when state = s6 or state = s4 else '0';
  Q(1) <= '1' when state = s5 or state = s3 or state = s2 else '0';
  Q(0) <= '1' when state = s5 or state = s3 or state = s1 else '0';

end Behavioral;
```

4.2.4 Kod VHDL Testbench

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY aiken_counter_testbench IS
END aiken_counter_testbench;

ARCHITECTURE behavior OF aiken_counter_testbench IS

    -- Component Declaration for the Unit Under Test (UUT)
    COMPONENT aiken_counter
    PORT(
         CLK : IN  std_logic;
         Q : OUT  std_logic_vector(3 downto 0)
        );
    END COMPONENT;

   --Inputs
   signal CLK : std_logic := '0';

  --Outputs
   signal Q : std_logic_vector(3 downto 0);

   -- Clock period definitions
   constant CLK_period : time := 10 ns;

BEGIN

  -- Instantiate the Unit Under Test (UUT)
   uut: aiken_counter PORT MAP (
          CLK => CLK,
          Q => Q
        );

   -- Clock process definitions
   CLK_process :process
   begin
    CLK <= '0';
    wait for CLK_period/2;
    CLK <= '1';
```

```
40      wait for CLK_period/2;
41    end process;
42  END;
```

4.2.5 Symulacja



4.2.6 Fizyczna implementacja (Kod UCF)

```
1  # Clocks
2  NET "CLK" LOC = "P7" | BUFG = CLK | PERIOD = 5ms HIGH 50%;
3
4  # LEDS
5  NET "Q(0)"   LOC = "P35";
6  NET "Q(1)"   LOC = "P29";
7  NET "Q(2)"   LOC = "P33";
8  NET "Q(3)"   LOC = "P34";
```

5 Wnioski

Na początku trudność sprawiło nam testowanie modułów w języku VHDL. Problem rozwiązał się sam, w momencie, gdy natknięcia się na materiał instruktażowy, w którym pokazane zostało, że przy testowaniu modułu, plik testowy vhd należy wygenerować na podstawie napisanego modułu (sam moduł nie stanowi pliku testowego).

Zadanie 3 wymagało od nas podłączenia wyświetlacza LED, mieliśmy dostępny już gotowy moduł lecz wymagał on od nas narysowania schematu co jest bardzo żmudne. Obeszliśmy to poprzez własną implementację wyjść aby wyświetlić odpowiednie cyfry na wyświetlaczu.

Wszystkie zadania zostały wykonane bardzo sprawnie i bez większych problemów, tworzenie układów w kodzie VHDL jest znacznie przyjemniejsze niż rysowanie schematów ponieważ jest szybsze i łatwiejsze w szukaniu błędów gdyż edytor sam nam powie, z którą linijką jest coś nie tak.