Sprawozdanie - laboratorium 4

Maksim Birszel, Piotr Karoń

15 grudnia 2019

1 Zadanie 1

Zaprojektować moduł, zasymulować i zaimplementować na płycie ZL-9572 6-bitowy detektor sekwencji Moore o sekwencji : 001100

1.1 Założenia

- Projekt ma zawierać tylko 1 moduł VHDL z szablonem FSM z 3 procesami
- Symulacja dla ciągu: $x = **abcdef**abcde\overline{f}**$
- wektor 18-bitowy + proces z pętlą
- Sprzęt: CLK_LF, CE->K0, x->K1, RST->K7, y->LED0

1.2 Moduł VHDL

```
1  entity detektor6b_moora is
2      Port ( x : in  STD_LOGIC;
3             y : out  STD_LOGIC;
4             CE : in  STD_LOGIC;
5             RST : in  STD_LOGIC;
6             CLK_LF : in  STD_LOGIC);
7  end detektor6b_moora;
8
9  architecture Behavioral of detektor6b_moora is
10
11         type state_type is (q0, q1, q2, q3, q4, q5, q6);
12         signal state, next_state : state_type;
13
14 begin
15
16         SYNC_PROC : process (clk_LF)
17         begin
```

```
18              if rising_edge(clk_LF) then
19                      if (RST = '1') then
20                              state <= q0;
21                      else
22                              state <= next_state;
23                      end if;
24              end if;
25      end process;
26
27      OUTPUT_DECODE : process (state)
28      begin
29              case (state) is
30              when q6 =>
31                      y <= '1';
32              when others =>
33                      y <= '0';
34              end case;
35      end process;
36
37      NEXT_STATE_DECODE : process (state, x)
38      begin
39              next_state <= q0;
40
41              case (state) is
42                      when q0 =>
43                              if (x = '1') then
44                                      next_state <= q0;
45                              else
46                                      next_state <= q1;
47                              end if;
48
49                      when q1 =>
50                              if (x = '1') then
51                                      next_state <= q0;
52                              else
53                                      next_state <= q2;
54                              end if;
55
56                      when q2 =>
57                              if (x = '1') then
58                                      next_state <= q3;
59                              else
60                                      next_state <= q2;
61                              end if;
62
```

```
63                          when q3 =>
64                                  if (x = '1') then
65                                          next_state <= q4;
66                                  else
67                                          next_state <= q1;
68                                  end if;
69
70                          when q4 =>
71                                  if (x = '1') then
72                                          next_state <= q0;
73                                  else
74                                          next_state <= q5;
75                                  end if;
76
77                          when q5 =>
78                                  if (x = '1') then
79                                          next_state <= q0;
80                                  else
81                                          next_state <= q6;
82                                  end if;
83
84                          when q6 =>
85                                  if (x = '1') then
86                                          next_state <= q0;
87                                  else
88                                          next_state <= q1;
89                                  end if;
90
91                          end case;
92          end process;
93
94  end Behavioral;
```

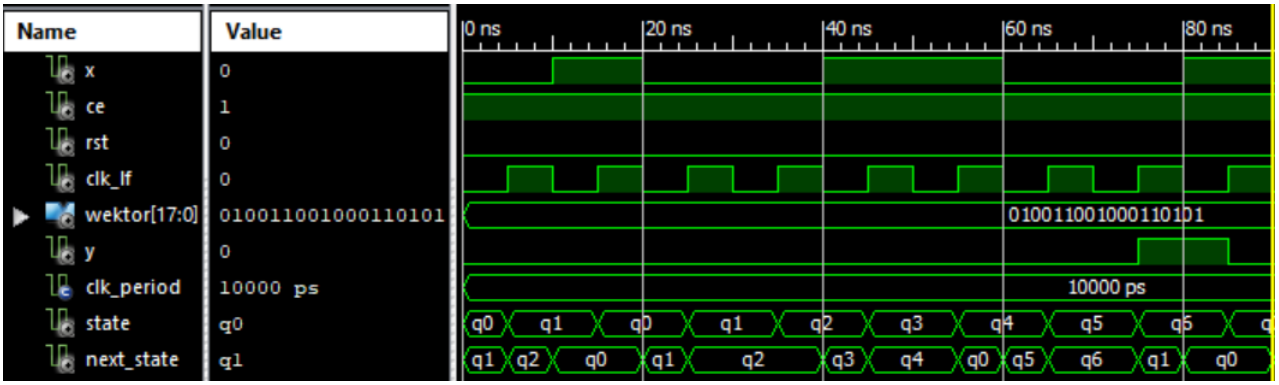
1.3 Test Bench

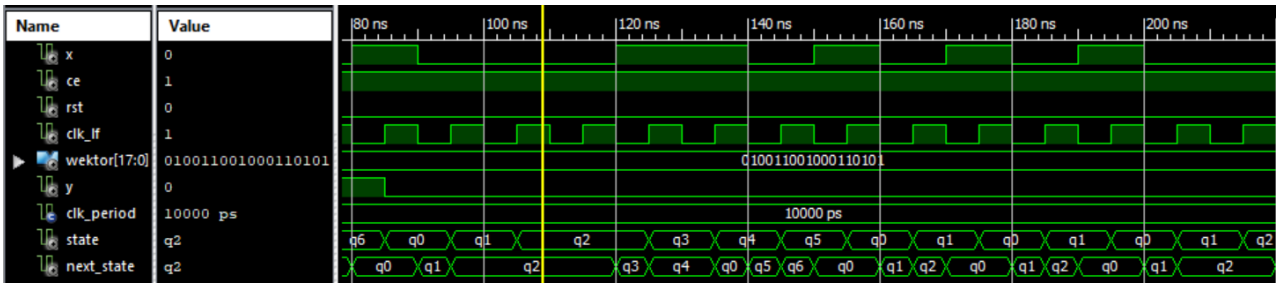
```
1  ENTITY detektor_testbench IS
2  END detektor_testbench;
3
4  ARCHITECTURE behavior OF detektor_testbench IS
5
6      COMPONENT detektor6b_moora
7      PORT(
8           x : IN  std_logic;
9           y : OUT  std_logic;
10          CE : IN  std_logic;
11          RST : IN  std_logic;
```

```
12          CLK_lf : IN  std_logic
13        );
14    END COMPONENT;
15
16    signal x : std_logic := '0';
17    signal CE : std_logic := '1';
18    signal RST : std_logic := '0';
19    signal CLK_LF : std_logic := '0';
20    signal wektor : std_logic_vector (17 downto 0) := "010011001000110101";
21
22    signal y : std_logic;
23
24    constant CLK_period : time := 10 ns;
25
26 BEGIN
27
28    uut: detektor6b_moora PORT MAP (
29          x => x,
30          y => y,
31          CE => CE,
32          RST => RST,
33          CLK_LF => CLK_LF
34        );
35
36    CLK_process :process
37    begin
38                CLK_LF <= '0';
39                wait for CLK_period/2;
40                CLK_LF <= '1';
41                wait for CLK_period/2;
42    end process;
43
44    stim_proc : process
45    begin
46                for i in 17 downto 0 loop
47                        x <= wektor(i);
48                        wait until falling_edge(CLK_LF);
49                end loop;
50
51    end process;
52 END;
```

1.4 Symulacja behawioralna

Symulacja została podzielona na dwie części, aby łatwiej było odczytać poszczególne wartości.





2 Zadanie 2

Zaprojektować moduł, zasymulować i zaimplementować na płycie ZL-9572 6-bitowy detektor sekwencji Moore (jak w zadaniu 1), ale za pomocą RTL z instrukcjami przerzutników FDCE oraz "with select".

2.1 Moduł VHDL

```
1  entity detektor_zadanie2 is
2      Port ( x : in  STD_LOGIC;
3             y : out  STD_LOGIC;
4             CE : in STD_LOGIC;
5             CLK : in STD_LOGIC;
6             CLR : in STD_LOGIC
7                          );
8  end detektor_zadanie2;
9
10 architecture Behavioral of detektor_zadanie2 is
11         signal intQ : STD_LOGIC_VECTOR (2 downto 0);
12         signal state : STD_LOGIC_VECTOR (2 downto 0);
```

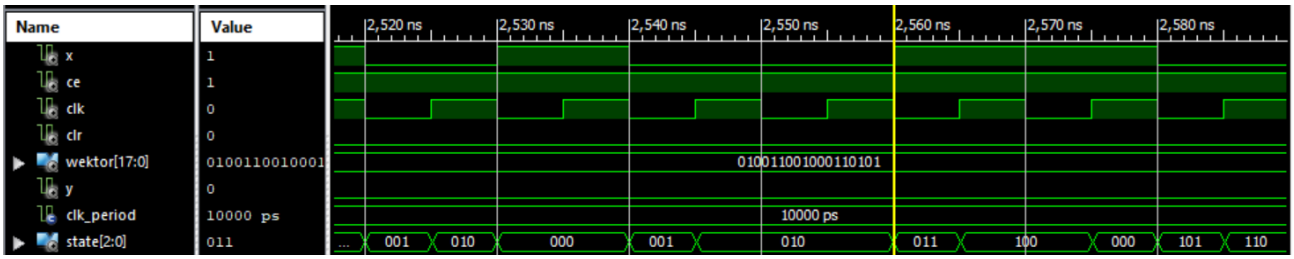
```
13 begin
14
15         FDCE0 : FDCE port map ( D => state(0), CE => CE, Q => intQ(0),
16                                                  CLR => CLR, C => CLK );
17
18         FDCE1 : FDCE port map ( D => state(1), CE => CE, Q => intQ(1),
19                                                  CLR => CLR, C => CLK );
20
21         FDCE2 : FDCE port map ( D => state(2), CE => CE, Q => intQ(2),
22                                                  CLR => CLR, C => CLK );
23
24         with x & intQ select
25                 state <= "001" when "0000",
26                          "000" when "1000",
27                          "010" when "0001",
28                          "000" when "1001",
29                          "010" when "0010",
30                          "011" when "1010",
31                          "001" when "0011",
32                          "100" when "1011",
33                          "101" when "0100",
34                          "000" when "1100",
35                          "110" when "0101",
36                          "000" when "1101",
37                          "001" when "0110",
38                          "000" when "1110",
39                          "XXX" when others;
40
41         with intq select
42                 y <= '1' when "110",
43                 '0' when others;
44
45 end Behavioral;
```

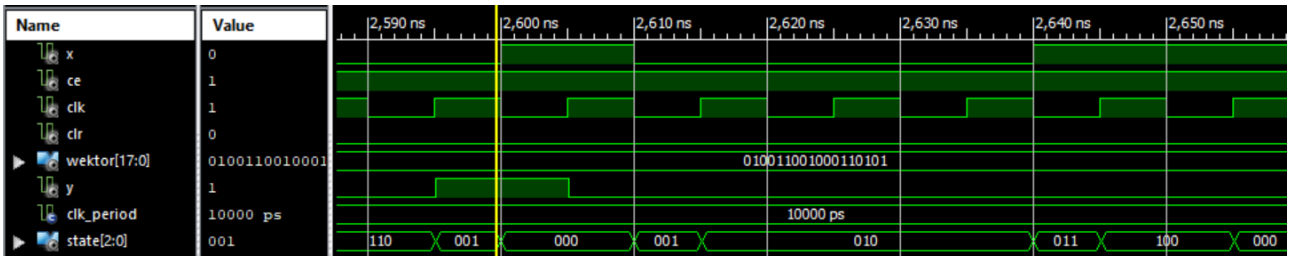
2.2 Test Bench 2

Test bench w tym zadaniu wyglądał identycznie jak w zadaniu pierwszym.

2.3 Symulacja behawioralna

Symulacja jest przesunięta w czasie ze względu na problemy ze środowiskiem ISE, który nie pozwalał na zrestartowanie symulacji po dodaniu do wykresu obiektu state.





3 Wnioski

Oba moduły udało się poprawnie zaimplementować, a symulacja przebiegła według wstępnych założeń.
Najwięcej problemów sprawiło odpowiednie przypisanie wartości poszczególnym stanom, w skrajnych przypadkach, kiedy np. ostatni stan danego przejścia jest równocześnie pierwszym stanem nowego.
W drugim zadaniu wykorzystaliśmy moduł stworzony na poprzednich zajęciach, który polegał na stworzeniu 3-bitowego licznika rewersyjnego z użyciem FDCE.