<table>
<tr><td>Termin zajęć
Wtorek NP
7:30 – 11:00</td><td colspan="2">Układy cyfrowe i systemy wbudowane</td></tr>
<tr><td colspan="2">Osoby wykonujące ćwiczenie:
Jakub Suski 264028, Adam Czekalski 264488</td><td>Grupa:
D</td></tr>
<tr><td colspan="2">Tytuł ćwiczenia:
Układy sekwencyjne</td><td>Laboratorium nr:
3</td></tr>
<tr><td>Data wykonania ćwiczenia</td><td>24.10.2023</td><td rowspan="2">Ocena:</td></tr>
<tr><td>Data oddania sprawozdania</td><td>7.11.2023</td></tr>
</table>

Na zajęciach laboratoryjnych na płytce CPLD ZL-9572 zaimplementowano układy sekwencyjne: licznik synchroniczny mod 9 negatywny w kodzie +3 oraz detektor sekwencji „11100” w postaci automatu Mealy-ego.

1. Licznik synchroniczny mod 9 negatywny w kodzie +3

Proces syntezy:

Zdecydowano się na implementację na przerzutnikach typu „D”.

n		Q_3	Q_2	Q_1	Q_0		Q_3'	Q_2'	Q_1'	Q_0'		D_3	D_2	D_1	D_0
0		0	0	1	1		1	0	1	1		1	0	1	1
1		0	1	0	0		0	0	1	1		0	0	1	1
2		0	1	0	1		0	1	0	0		0	1	0	0
3		0	1	1	0		0	1	0	1		0	1	0	1
4		0	1	1	1		0	1	1	0		0	1	1	0
5		1	0	0	0		0	1	1	1		0	1	1	1
6		1	0	0	1		1	0	0	0		1	0	0	0
7		1	0	1	0		1	0	0	1		1	0	0	1
8		1	0	1	1		1	0	1	0		1	0	1	0
9		1	1	0	0		-	-	-	-		-	-	-	-
10		1	1	0	1		-	-	-	-		-	-	-	-
11		1	1	1	0		-	-	-	-		-	-	-	-
12		1	1	1	1		-	-	-	-		-	-	-	-
13		-	-	-	-		-	-	-	-		-	-	-	-
14		-	-	-	-		-	-	-	-		-	-	-	-
15		-	-	-	-		-	-	-	-		-	-	-	-

Minimalizacja metodą Karnaugh:

Q_3Q_2 \ Q_1Q_0	00	01	11	10
00	-	-	1	-
01	0	0	0	0
11	-	-	-	-
10	0	1	1	1

Q_3Q_2 \ Q_1Q_0	00	01	11	10
00	-	-	1	-
01	0	0	0	0
11	-	-	-	-
10	0	1	1	1

Q_3Q_2 \ Q_1Q_0	00	01	11	10
00	-	-	1	-
01	0	0	0	0
11	-	-	-	-
10	0	1	1	1

$$D_3 = \overline{Q_3}\,\overline{Q_2} + Q_3Q_0 + Q_3Q_1$$

Q_3Q_2 \ Q_1Q_0	00	01	11	10
00	-	-	0	-
01	0	1	1	1
11	-	-	-	-
10	1	0	0	0

Q_3Q_2 \ Q_1Q_0	00	01	11	10
00	-	-	0	-
01	0	1	1	1
11	-	-	-	-
10	1	0	0	0

Q_3Q_2 \ Q_1Q_0	00	01	11	10
00	-	-	0	-
01	0	1	1	1
11	-	-	-	-
10	1	0	0	0

$$D_2 = Q_3\overline{Q_1}\,\overline{Q_0} + Q_2Q_0 + Q_2Q_1$$

Q_3Q_2 \ Q_1Q_0	00	01	11	10
00	-	-	1	-
01	1	0	1	0
11	-	-	-	-
10	1	0	1	0

Q_3Q_2 \ Q_1Q_0	00	01	11	10
00	-	-	1	-
01	1	0	1	0
11	-	-	-	-
10	1	0	1	0

$$D_1 = \overline{Q_1}\,\overline{Q_0} + Q_1 Q_0$$

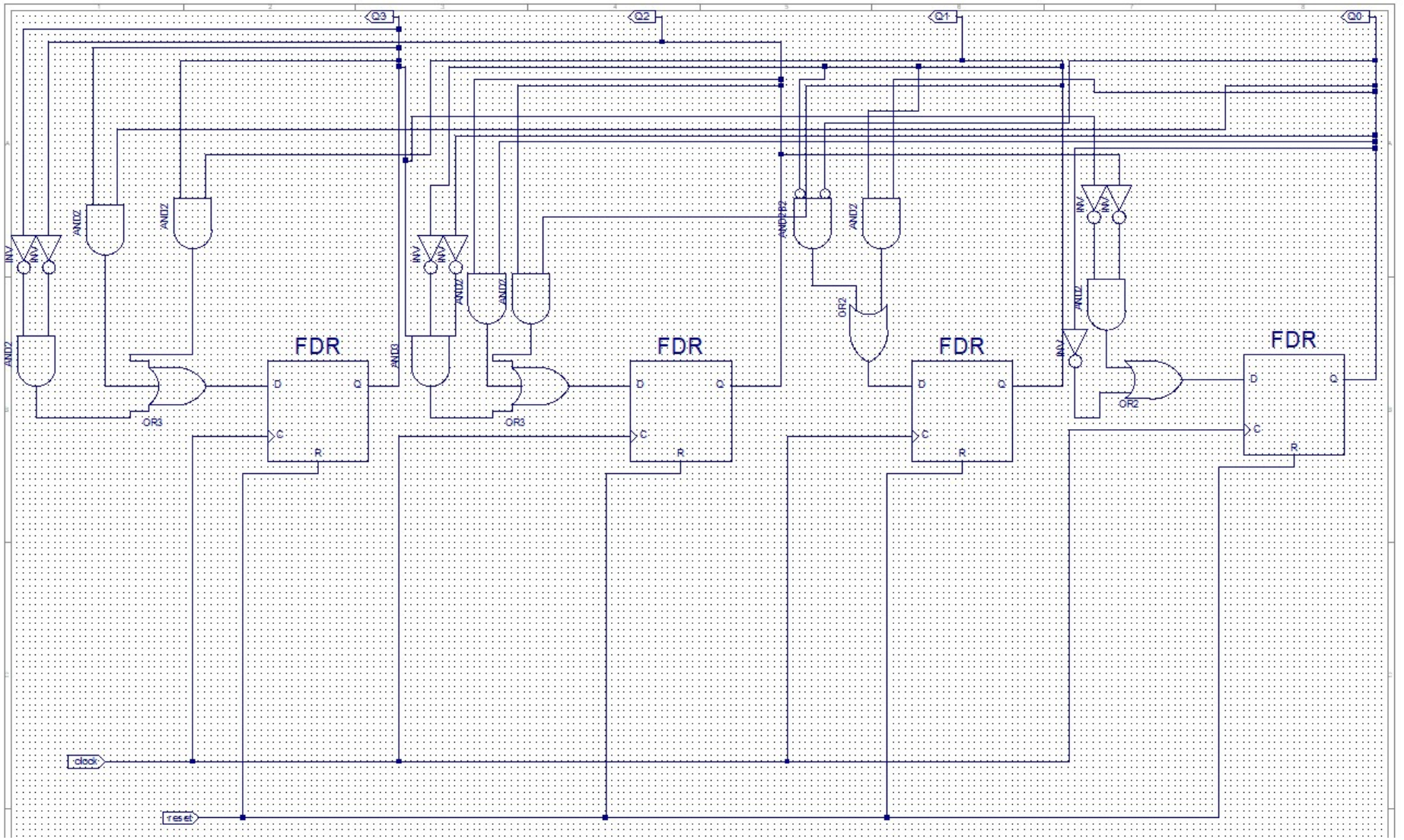
Q_3Q_2 \ Q_1Q_0	00	01	11	10
00	-	-	1	-
01	1	0	0	1
11	-	-	-	-
10	1	0	0	1

Q_3Q_2 \ Q_1Q_0	00	01	11	10
00	-	-	1	-
01	1	0	0	1
11	-	-	-	-
10	1	0	0	1

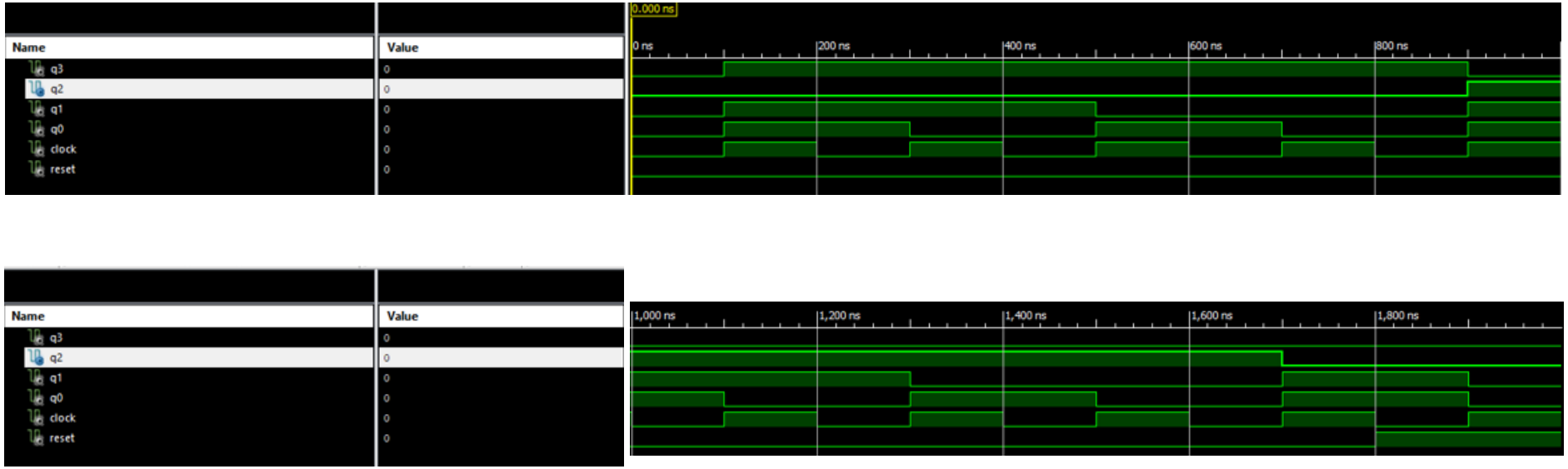
Q_3Q_2 \ Q_1Q_0	00	01	11	10
00	-	-	1	-
01	1	0	0	1
11	-	-	-	-
10	1	0	0	1

$$D_0 = \overline{Q_0} + \overline{Q_3}\,\overline{Q_2}$$

Proces implementacji:



Rysunek 1-1 Układ licznika synchronicznego mod 9 negatywnego w kodzie +3



Rysunek 1-2 Przebiegi czasowe w symulatorze ISim

```
15  LIBRARY ieee;
16  USE ieee.std_logic_1164.ALL;
17  USE ieee.numeric_std.ALL;
18  LIBRARY UNISIM;
19  USE UNISIM.Vcomponents.ALL;
20  ENTITY zad1sch_zad1sch_sch_tb IS
21  END zad1sch_zad1sch_sch_tb;
22  ARCHITECTURE behavioral OF zad1sch_zad1sch_sch_tb IS
23
24     COMPONENT zad1sch
25     PORT( Q2 :   OUT STD_LOGIC;
26           Q3    :   OUT STD_LOGIC;
27           Q1    :   OUT STD_LOGIC;
28           Q0    :   OUT STD_LOGIC;
29           clock :   IN  STD_LOGIC;
30           reset :   IN  STD_LOGIC);
31     END COMPONENT;
32
33     SIGNAL Q2    :   STD_LOGIC;
34     SIGNAL Q3    :   STD_LOGIC;
35     SIGNAL Q1    :   STD_LOGIC;
36     SIGNAL Q0    :   STD_LOGIC;
37     SIGNAL clock :   STD_LOGIC :='0';
38     SIGNAL reset :   STD_LOGIC;
39
40  BEGIN
41
42     UUT: zad1sch PORT MAP(
43          Q2 => Q2,
44          Q3 => Q3,
45          Q1 => Q1,
46          Q0 => Q0,
47          clock => clock,
48          reset => reset
49     );
50
51  -- *** Test Bench - User Defined Section ***
52     clock <= not clock after 100 ns;
53      reset <= '0', '1' after 1800 ns;
54  -- *** End Test Bench - User Defined Section ***
55
56  END;
```

Rysunek 1-3 Kod w języku VHDL wraz z wektorami pobudzeń

W pliku testowym, aby wygenerować falę prostokątną o wypełnieniu 50% i stałym okresie, dopisano inicjalizację wartości początkowej sygnału „clock” w części deklaracyjnej architektury, a w treści architektury zastosowano przypisanie sygnału, negując wejście zegarowe co 100 ns.

```
1  #-----------------------------------------------
2  #  ZL-9572 CPLD board,  J.Sugier 2009
3  #-----------------------------------------------
4
5  # Clocks
6  NET "clock" LOC = "P7" | BUFG = CLK;
7  NET "clock" PERIOD = 5ms HIGH 50%;
8
9  #NET "Clk_XT" LOC = "P5" | BUFG = CLK;
10 #NET "Clk_XT" PERIOD = 500ns HIGH 50%;
11
12 # Keys
13 NET "reset" LOC = "P42";
14 #NET "Key<1>" LOC = "P40";
15 #NET "Key<2>" LOC = "P43";
16 #NET "Key<3>" LOC = "P38";
17 #NET "Key<4>" LOC = "P37";
18 #NET "Key<5>" LOC = "P36";  # shared with ROT_A
19 #NET "Key<6>" LOC = "P24";  # shared with ROT_B
20 #NET "Key<7>" LOC = "P39";  # GSR
21
22 # LEDS
23 NET "Q0"  LOC = "P35";
24 NET "Q1"  LOC = "P29";
25 NET "Q2"  LOC = "P33";
26 NET "Q3"  LOC = "P34";
27 #NET "LED<4>"  LOC = "P28";
28 #NET "LED<5>"  LOC = "P27";
29 #NET "LED<6>"  LOC = "P26";
30 #NET "LED<7>"  LOC = "P25";
```

Rysunek 1-4 Fragment pliku .ucf z przypisanymi wyjściami i wejściami do wyprowadzeń

Natomiast w pliku *.ucf* dodatkowo odkomentowano linijkę 6 i 7 i przypisano wejście „clock”

Licznik zlicza impulsy zegarowe generowane w przystawce, przez co na płytce można zobaczyć całą sekwencję licznika. Zgaszona dioda oznacza „1” na wyjściu, zapalona natomiast „0” na wyjściu.

2. Detektor sekwencji „11100” w postaci automatu Mealy-ego

Proces syntezy:

Wykonano na przerzutnikach JK.

	n	Q2	Q1	Q0	X	Q2'	Q1'	Q0'	Y	J2	K2	J1	K1	J0	K0
1	0	0	0	0	0	0	0	0	0	0	\	0	\	0	\
2	0	0	0	0	1	0	0	1	0	0	\	0	\	1	\
3	1	0	0	1	0	0	0	0	0	0	\	0	\	\	1
4	1	0	0	1	1	0	1	0	0	0	\	1	\	\	1
5	2	0	1	0	0	0	0	0	0	0	\	\	1	0	\
6	2	0	1	0	1	0	1	1	0	0	\	\	0	1	\
7	3	0	1	1	0	1	0	0	0	1	\	\	1	\	1
8	3	0	1	1	1	0	1	1	0	0	\	\	0	\	0
9	4	1	0	0	0	1	0	1	1	\	0	0	\	1	\
10	4	1	0	0	1	0	0	1	0	\	1	0	\	1	\
11	5	1	0	1	0	0	0	0	0	\	1	0	\	\	1
12	5	1	0	1	1	0	0	1	0	\	1	0	\	\	0
13	6	1	1	0	0	\	\	\	\	\	\	\	\	\	\
14	6	1	1	0	1	\	\	\	\	\	\	\	\	\	\
15	7	1	1	1	0	\	\	\	\	\	\	\	\	\	\
16	7	1	1	1	1	\	\	\	\	\	\	\	\	\	\

Minimalizacja metodą siatek Karnaugha:

Q_2Q_1 \ Q_0X	00	01	11	10
00	0	0	0	0
01	0	0	0	1
11	\	\	\	\
10	\	\	\	\

$$J_2 = Q_1 Q_0 \bar{X}$$

Q_2Q_1 \ Q_0X	00	01	11	10
00	\	\	\	\
01	\	\	\	\
11	\	\	\	\
10	0	1	1	1

$$K_2 = X + Q_0$$

Q_2Q_1 \ Q_0X	00	01	11	10
00	0	0	1	0
01	\	\	\	\
11	\	\	\	\
10	0	0	0	0

$$J_1 = \overline{Q_2} Q_0 X$$

Q_2Q_1 \ Q_0X	00	01	11	10
00	\	\	\	\
01	1	0	0	1
11	\	\	\	\
10	\	\	\	\

$$K_1 = \bar{X}$$

Q_2Q_1 \ Q_0X	00	01	11	10
00	0	1	\	\
01	0	1	\	\
11	\	\	\	\
10	1	1	\	\

$$J_0 = Q_2 + X$$

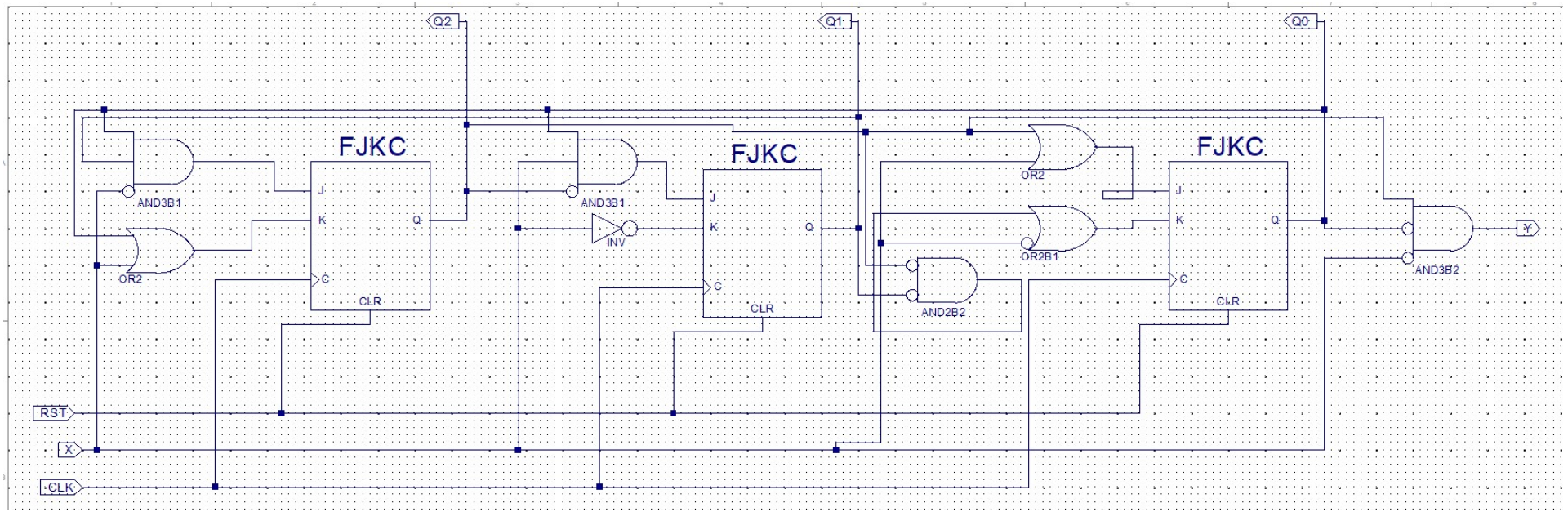
Q_2Q_1 \ Q_0X	00	01	11	10
00	\	\	1	1
01	\	\	0	1
11	\	\	\	\
10	\	\	0	1

$$K_0 = \bar{X} + \overline{Q_2}\,\overline{Q_1}$$

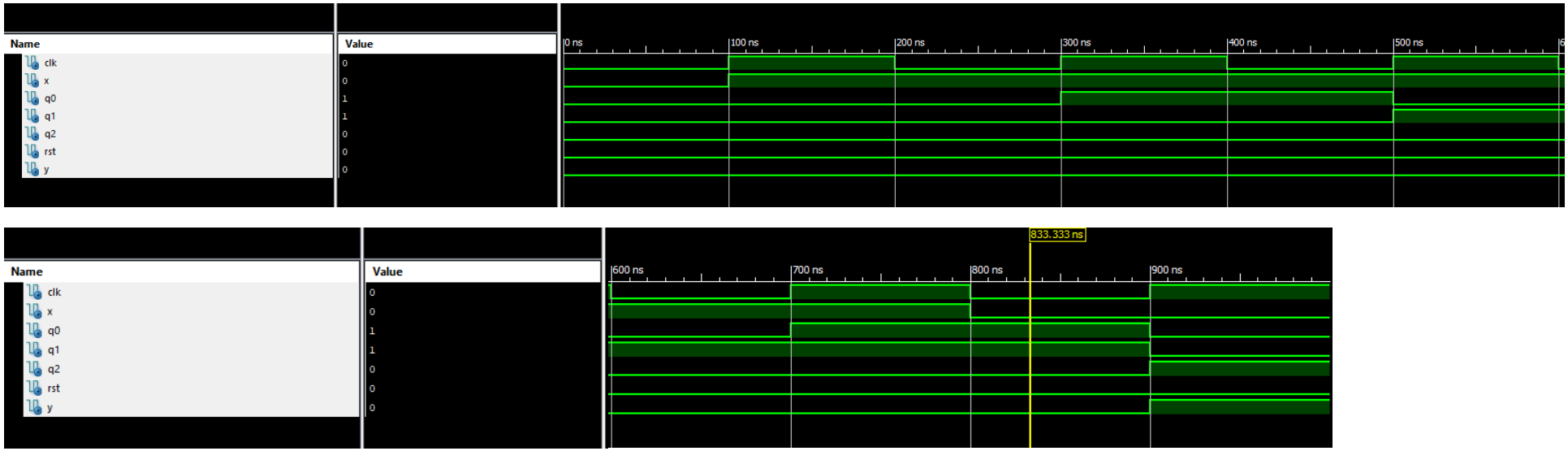
Q_2Q_1 \ Q_0X	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	\	\	\	\
10	1	0	0	0

$$Y = Q_2 \overline{Q_0}\,\bar{X}$$

Proces implementacji:



Rysunek 2-1 Układ detektora sekwencji: "11100" w postaci automatu Mealy-ego



Rysunek 2-2 Przebiegi czasowe w symulatorze ISim

Kod:

```
USE ieee.numeric_std.ALL;
LIBRARY UNISIM;
USE UNISIM.Vcomponents.ALL;
ENTITY zad2sch_zad2sch_sch_tb IS
END zad2sch_zad2sch_sch_tb;
ARCHITECTURE behavioral OF zad2sch_zad2sch_sch_tb IS

    COMPONENT zad2sch
    PORT( CLK     :   IN  STD_LOGIC;
          wej     :   IN  STD_LOGIC;
          Q0      :   OUT STD_LOGIC;
          Q1      :   OUT STD_LOGIC;
          Q2      :   OUT STD_LOGIC;
          RST     :   IN  STD_LOGIC;
          wyj     :   OUT STD_LOGIC);
    END COMPONENT;

    SIGNAL CLK    :   STD_LOGIC :='0';
    SIGNAL wej    :   STD_LOGIC;
    SIGNAL Q0     :   STD_LOGIC;
    SIGNAL Q1     :   STD_LOGIC;
    SIGNAL Q2     :   STD_LOGIC;
    SIGNAL RST    :   STD_LOGIC;
    SIGNAL wyj    :   STD_LOGIC;

BEGIN

    UUT: zad2sch PORT MAP(
         CLK => CLK,
         wej => wej,
         Q0 => Q0,
         Q1 => Q1,
         Q2 => Q2,
         RST => RST,
         wyj => wyj
    );

-- *** Test Bench - User Defined Section ***
    CLK <= not CLK after 100 ns;
     RST <= '0', '1' after 1800 ns;
     wej <= '0', '1' after 100 ns, '0' after 800 ns;
-- *** End Test Bench - User Defined Section ***

END;
```

Rysunek 2-3 Kod w języku VHDL wraz z wektorami pobudzeń

Wprowadzamy sekwencję „0111111100000000000”, zmieniając zbocze zegara co 100 ns i sczytując jeden bit co 100 ns z portu *wej*. Analizując przebiegi czasowe symulatora ISim, można zauważyć, że po 400. nanosekundzie sekwencja „01111” nie została wykryta, a także: po 500., 600. i 700. nanosekundzie sekwencja „11111” ani sekwencja „11110” po 800. nanosekundzie. Dopiero po 900. nanosekundzie *y* zmienia wartość na 1, gdy wykryta zostaje pożądana sekwencja „11100”.

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

#NET "Clk_XT" LOC = "P5" | BUFG = CLK;
#NET "Clk_XT" PERIOD = 500ns HIGH 50%;

# Keys
NET "wej" LOC = "P42";
#NET "Key<1>" LOC = "P40";
#NET "Key<2>" LOC = "P43";
#NET "Key<3>" LOC = "P38";
#NET "Key<4>" LOC = "P37";
#NET "Key<5>" LOC = "P36";  # shared with ROT_A
#NET "Key<6>" LOC = "P24";  # shared with ROT_B
NET "RST" LOC = "P39";  # GSR

# LEDS
NET "wyj"  LOC = "P35";
#NET "wej"  LOC = "P29";
#NET "LED<2>"  LOC = "P33";
#NET "LED<3>"  LOC = "P34";
NET "Q0"  LOC = "P28";
NET "Q1"  LOC = "P27";
NET "Q2"  LOC = "P26";
#NET "Q3"  LOC = "P25";
```

Rysunek 2-4 Fragment pliku .ucf z przypisanymi wyjściami i wejściami do wyprowadzeń

Układ gdy wykryje sekwencję „11100”, zgasza diodę P35, która symbolizuje jej detekcję.

3. Wnioski

Układ licznika i detektora sekwencji działały poprawnie. Niestety, z powodu „drobnych” problemów ze środowiskiem Xilinx w czasie implementacji drugiego układu, nie udało się zaimplementować „ulepszonej” wersji układu z Rotary Encoderem i wyświetlaczem 7-segmentowym.