Jędrzej Stasik, Kamil Szadkowski, 29.11.2023 r.

Urządzenia Cyfrowe i Systemy Wbudowane

Licznik rewersyjny 0-1-6-2-3-4-5-7

Tabela Prawdy

t				t+1					
DIR	Q2	Q1	Q0	Q2	Q1	Q0	D2	D1	D0
0	0	0	0	0	0	1	0	0	1
0	0	0	1	1	1	0	1	1	0
0	0	1	0	0	1	1	0	1	1
0	0	1	1	1	0	0	1	0	0
0	1	0	0	1	0	1	1	0	1
0	1	0	1	1	1	1	1	1	1
0	1	1	0	0	1	0	0	1	0
0	1	1	1	0	0	0	0	0	0
1	0	0	0	1	1	1	1	1	1
1	0	0	1	0	0	0	0	0	0
1	0	1	0	1	1	0	1	1	0
1	0	1	1	0	1	0	0	1	0
1	1	0	0	0	1	1	0	1	1
1	1	0	1	1	0	0	1	0	0
1	1	1	0	0	0	1	0	0	1
1	1	1	1	1	0	1	1	0	1

Minimalizacja siatkami Karnaugh

DIRQ2\Q1Q0	00	01	11	10
00	0	1	1	0
01	1	1	0	0
11	0	1	1	0
10	1	0	0	1

$D2 = \overline{DIR}\overline{Q2}Q0 + \overline{DIR}Q2\overline{Q1} + DIR\overline{Q2}\overline{Q0} + DIRQ2Q0$

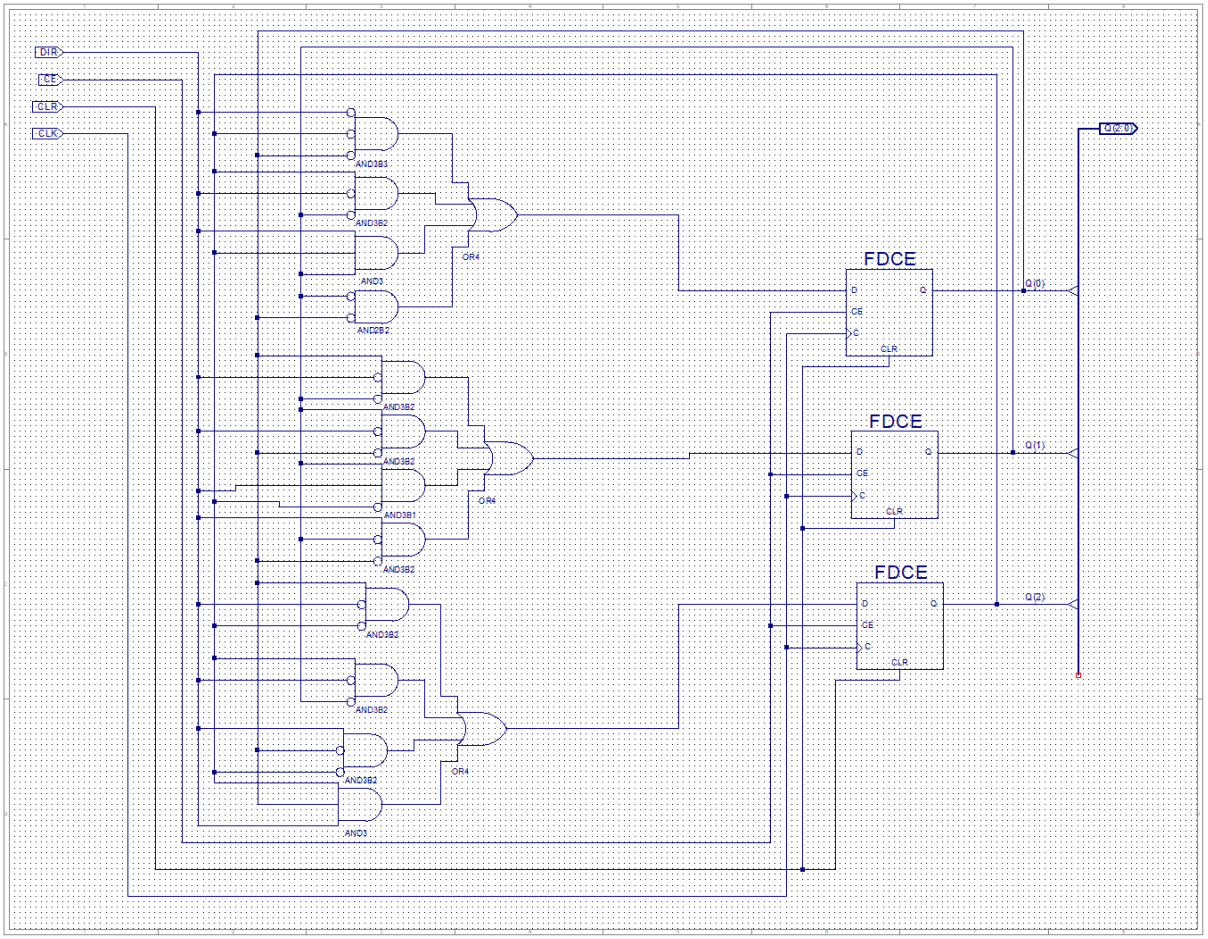
DIRQ2\Q1Q0	00	01	11	10
00	0	1	0	1
01	0	1	0	1
11	1	0	0	0
10	1	0	1	1

$D1 = \overline{DIR}\overline{Q1}Q0 + \overline{DIR}Q1\overline{Q0} + DIR\overline{Q2}Q1 + DIR\overline{Q1}\overline{Q0}$

DIRQ2\Q1Q0	00	01	11	10
00	1	0	0	1
01	1	1	0	0
11	1	0	1	1
10	1	0	0	0

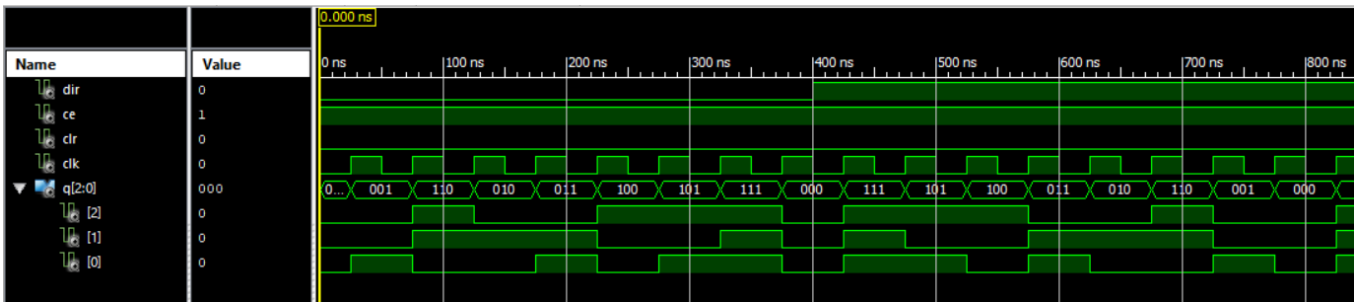
$D0 = \overline{DIR}\overline{Q2}\overline{Q0} + \overline{DIR}Q2\overline{Q1} + DIRQ2Q1 + \overline{Q1}\overline{Q0}$

Schemat połączenia bramek logicznych



Schemat projektu licznika rewersyjnego opartego o przerzutniki typu D

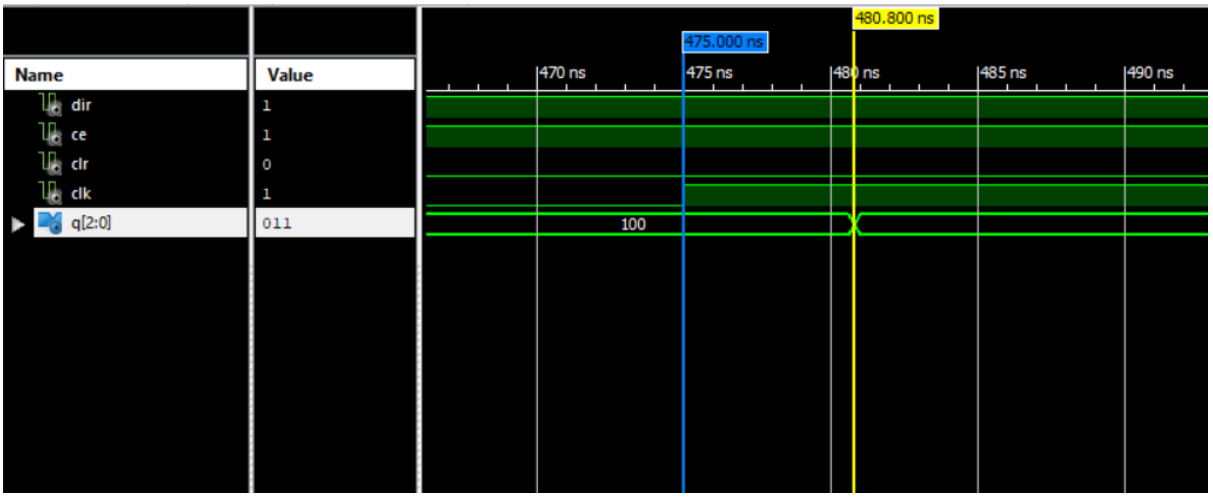
Symulacja behawioralna



Symulacja behawioralna - przebieg czasowy

Symulacja Post-Fit

Na podstawie symulacji post-fit określono czas propagacji który wynosi 5.8 ns



Symulacja Post-fit - określenie czasu propagacji

Plik VHDL

```
-- Vhdl test bench created from schematic C:\Users\lab\Documents\lab2\Lab2\lab2_sch_1.sch - Wed Nov 29 15:25:29 2023
--
-- Notes:
-- 1) This testbench template has been automatically generated using types
-- std_logic and std_logic_vector for the ports of the unit under test.
-- Xilinx recommends that these types always be used for the top-level
-- I/O of a design in order to guarantee that the testbench will bind
-- correctly to the timing (post-route) simulation model.
-- 2) To use this template as your testbench, change the filename to any
-- name of your choice with the extension .vhd, and use the "Source->Add"
-- menu in Project Navigator to import the testbench. Then
-- edit the user defined section below, adding code to generate the
-- stimulus for your design.
--
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;
LIBRARY UNISIM;
USE UNISIM.Vcomponents.ALL;
ENTITY lab2_sch_1_lab2_sch_1_sch_tb IS
END lab2_sch_1_lab2_sch_1_sch_tb;
ARCHITECTURE behavioral OF lab2_sch_1_lab2_sch_1_sch_tb IS

   COMPONENT lab2_sch_1
   PORT( DIR   :  IN STD_LOGIC;
         CE    :  IN STD_LOGIC;
         CLR   :  IN STD_LOGIC;
         CLK   :  IN STD_LOGIC;
         Q :  OUT   STD_LOGIC_VECTOR (2 DOWNTO 0));
   END COMPONENT;

   SIGNAL DIR  :  STD_LOGIC;
   SIGNAL CE   :  STD_LOGIC;
   SIGNAL CLR  :  STD_LOGIC;
   SIGNAL CLK  :  STD_LOGIC:='0';
   SIGNAL Q :  STD_LOGIC_VECTOR (2 DOWNTO 0);

BEGIN

   UUT: lab2_sch_1 PORT MAP(
      DIR => DIR,
      CE => CE,
      CLR => CLR,
      CLK => CLK,
      Q => Q
    );

      CLK <= not CLK after 25 ns;
      DIR <= '0', '1' after 400 ns;
      CE <= '1';
      CLR <='0';


-- *** Test Bench - User Defined Section ***
   tb : PROCESS
   BEGIN
      WAIT; -- will wait forever
   END PROCESS;
-- *** End Test Bench - User Defined Section ***

END;
```

Korzystając z poniższego wzoru obliczono okres sygnału zegarowego:

$$T_{CLK} = \frac{1}{f} = \frac{1}{40MHz} = 25ns$$

Aby pokazać w symulatorze zmianę kierunku odlicznia po 400 ns zmieniana jest wartość DIR.

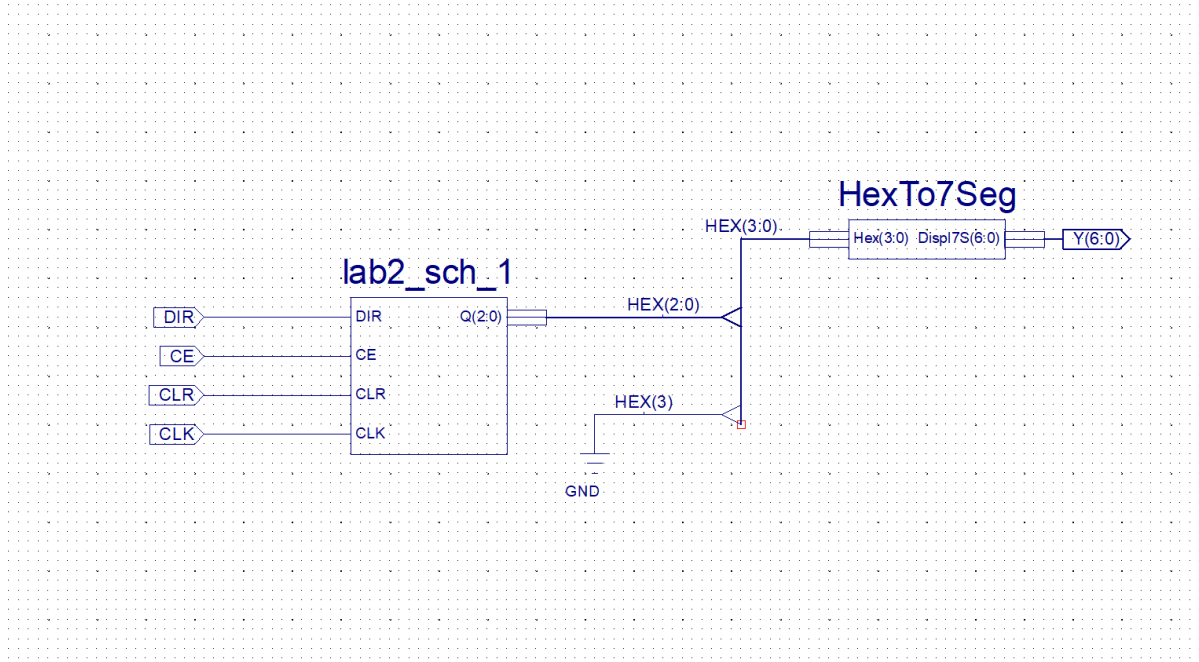
Plik .ucf - przypisanie wejść i wyjść

```
# Clocks
#NET "CLK" LOC = "P7" | BUFG = CLK | PERIOD =   500ms HIGH 50%;
NET "CLK" LOC = "P5" | BUFG = CLK | PERIOD = 500ns HIGH 50%;

# Keys
#NET "CE" LOC = "P42";
#NET "DIR" LOC = "P40";
#NET "DATA_IN(2)" LOC = "P43";
#NET "DATA_IN(3)" LOC = "P38";
#NET "DATA_IN(4)" LOC = "P37";
#NET "DATA_IN(5)" LOC = "P36";  # shared with ROT_A
#NET "DATA_IN(6)" LOC = "P24";  # shared with ROT_B
NET "CLR" LOC = "P39";  # GSR

# LEDS
#NET "Q(0)"  LOC = "P35";
#NET "Q(1)"  LOC = "P29";
#NET "Q(2)"  LOC = "P33";
#NET "Y(3)"  LOC = "P34";
#NET "LED<4>"  LOC = "P28";
#NET "LED<5>"  LOC = "P27";
#NET "LED<6>"  LOC = "P26";
#NET "LED<7>"  LOC = "P25";

#NET "LED<8>"  LOC = "P13";   # shared with seg. B
#NET "LED<9>"  LOC = "P11";   # shared with seg. F
#NET "LED<10>" LOC = "P12";   # shared with seg. A
#NET "LED<11>" LOC = "P18";   # shared with seg. DP
#NET "LED<12>" LOC = "P22";   # shared with seg. C
#NET "LED<13>" LOC = "P20";   # shared with seg. G
#NET "LED<14>" LOC = "P19";   # shared with seg. D
#NET "LED<15>" LOC = "P14";   # shared with seg. E

# DISPL. 7-SEG
#NET "D7S_D(0)" LOC = "P8" | SLEW = "SLOW";
#NET "D7S_D(1)" LOC = "P6" | SLEW = "SLOW";
#NET "D7S_D(2)" LOC = "P4" | SLEW = "SLOW";
#NET "D7S_D(3)" LOC = "P9" | SLEW = "SLOW";
NET "Y(0)" LOC = "P12"; # Seg. A; shared with LED<10>
NET "Y(1)" LOC = "P13"; # Seg. B; shared with LED<8>
NET "Y(2)" LOC = "P22"; # Seg. C; shared with LED<12>
NET "Y(3)" LOC = "P19"; # Seg. D; shared with LED<14>
NET "Y(4)" LOC = "P14"; # Seg. E; shared with LED<15>
NET "Y(5)" LOC = "P11"; # Seg. F; shared with LED<9>
NET "Y(6)" LOC = "P20"; # Seg. G; shared with LED<13>
#NET "D7S_S<7>" LOC = "P18";  # Seg. DP;shared with LED<11>

# Rotary encoder
NET "ROT_A" LOC = "P36";   # shared with Key<5>
NET "ROT_B" LOC = "P24";   # shared with Key<6>

# PS/2
#NET "PS2_Clk"   LOC = "P3";
#NET "PS2_Data"  LOC = "P2";

# RS-232
#NET "RS_RX" LOC = "P1";
#NET "RS_TX" LOC = "P44";

```

Plik .ucf zawierający przypisanie wejść i wyjść dla projektu z zadania 3.

Użycie bloku HexTo7Seg

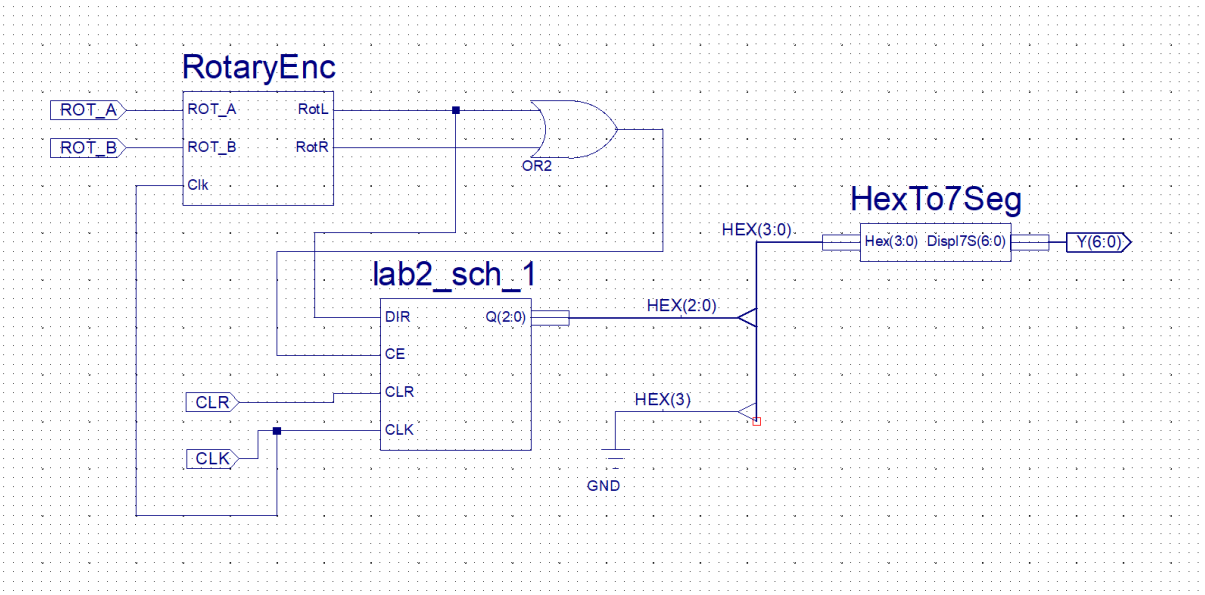
Po zrealizowaniu schematu liczniku, naszym zadaniem było stworzenie projektu hierarchicznego z wykorzystaniem bloku HexTo7Seg w celu przekształcenia liczb z formatu binarnego na szesnastkowy i wyświetlenie ich na wyświetlaczu. W tym celu podłączyliśmy wyjścia Q naszego licznika do wejść Hex bloku HexTo7Seg. Nadmiarowe wejście podpięliśmy do masy. Schemat projektu realizującego zadanie drugie przedstawiono na poniższym rysunku.



Schemat projektu hierarchicznego wykorzystującego blok HexTo7Seg

Użycie bloku RotaryENC

Trzecie zadanie polegało na rozbudowaniu wcześniejszego projektu dodając blok RotaryEnc. Dzięki jego użyciu działanie licznika w obydwie strony realizowane było za pomocą pokrętła. Przekręcenie pokrętła w prawo powodowało standardowe odliczanie w prawo. Natomiast przekręcenie go w lewo skutkowało odliczaniem w kierunku przeciwnym. W tym przypadku (chociaż nie widać różnicy względem poprzedniego schematu) został wykorzystany zegar o wyższej częstotliwości (T=500ns). CE jest pobudzany jeśli wykryto przekręcenie pokrętła w jedną lub drugą stronę, natomiast DIR zależy tylko od tego czy wykryto obrót w lewo co daje nam dwie możliwe wartości. Schemat projektu realizującego zadanie trzecie przedstawiono na poniższym rysunku.



Schemat projektu hierarchicznego wykorzystującego bloki HexTo7Seg oraz RotaryEnc

Wnioski

Na zajęciach zrealizowano 3 zadania. Układ został poprawnie zaprojektowany, wyniki symulacji zgadzały się z założeniami zadania. Po zaprogramowaniu płytka działała poprawnie we wszystkich trzech konfiguracjach.