

Maciej Byczko Bartosz Matysiak	Prowadzący: dr inż. Jacek Mazurkiewicz	Numer ćwiczenia 2
PN 10:50 TP	Temat ćwiczenia: Układy Sekwencyjne	Ocena:
Grupa: B	Data wykonania: 10 Października 2021	

## Spis treści

<b>1</b>	<b>Zadanie 1</b>	<b>2</b>
1.1	Polecenie . . . . .	2
1.2	Rozwiązanie . . . . .	2
1.2.1	Schemat stanów . . . . .	2
1.2.2	Tabela prawdy . . . . .	2
1.2.3	Siatki Karnaugh . . . . .	3
1.2.4	Schemat układu . . . . .	4
1.2.5	Kod VHDL . . . . .	4
1.2.6	Symulacja . . . . .	5
<b>2</b>	<b>Zadanie 2</b>	<b>5</b>
2.1	Polecenie . . . . .	5
2.2	Rozwiązanie . . . . .	5
2.2.1	Opis symboliki . . . . .	5
2.2.2	Schemat grafowy . . . . .	6
2.2.3	Tabela prawdy . . . . .	7
2.2.4	Siatka Karnaugh . . . . .	8
2.2.5	Schemat układu . . . . .	8
2.2.6	Kod VHDL . . . . .	8
2.2.7	Symulacja . . . . .	10
<b>3</b>	<b>Wnioski</b>	<b>10</b>

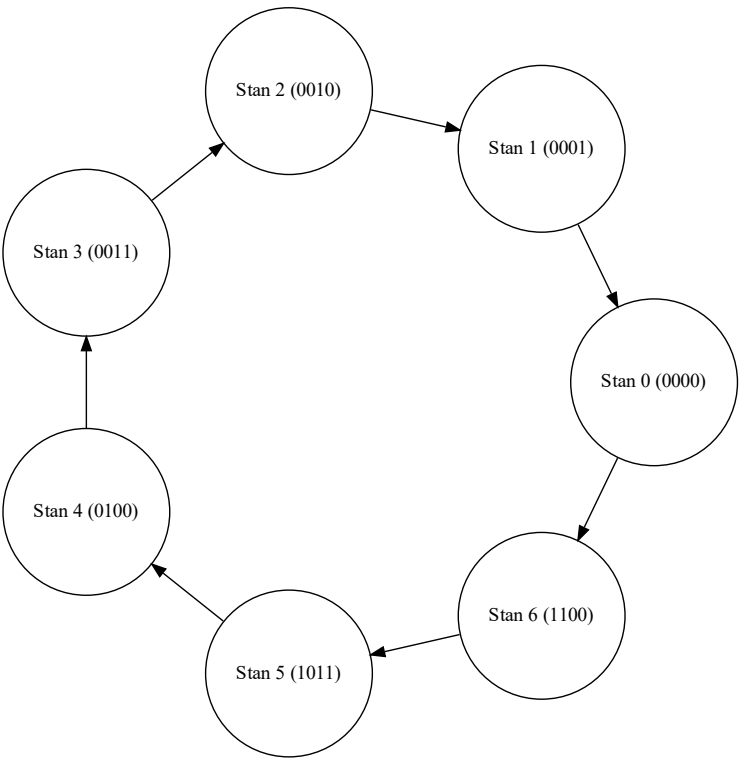
# 1 Zadanie 1

## 1.1 Polecenie

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

## 1.2 Rozwiązanie

### 1.2.1 Schemat stanów



### 1.2.2 Tabela prawdy

n	Q(t)				Q(t+1)				JK							
	$Q_3$	$Q_2$	$Q_1$	$Q_0$	$Q_3$	$Q_2$	$Q_1$	$Q_0$	$J_3$	$K_3$	$J_2$	$K_2$	$J_1$	$K_1$	$J_0$	$K_0$
0	0	0	0	0	1	1	0	0	1	-	1	-	0	-	0	-
1	0	0	0	1	0	0	0	0	0	-	0	-	0	-	-	1
2	0	0	1	0	0	0	0	1	0	-	0	-	-	1	1	-
3	0	0	1	1	0	0	1	0	0	-	0	-	-	0	-	1
4	0	1	0	0	0	0	1	1	0	-	-	1	1	-	1	-
5	1	0	1	1	0	1	0	0	-	1	1	-	-	1	-	1
6	1	1	0	0	1	0	1	1	-	0	-	1	1	-	1	-

## 1.2.3 Siatki Karnaugh

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

$$J_3 = \overline{Q_2} \overline{Q_1} \overline{Q_0}$$

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

$$J_2 = \overline{Q_1} \overline{Q_0} + Q_3$$

		00	01	11	10
$Q_3Q_2$	00	0	0	-	-
	01	1	-	-	-
	11	1	-	-	-
	10	-	-	-	-

$$J_1 = Q_2$$

		00	01	11	10
$Q_3Q_2$	00	0	-	-	1
	01	1	-	-	-
	11	1	-	-	-
	10	-	-	-	-

$$J_0 = Q_1 + Q_2$$

		00	01	11	10
$Q_3Q_2$	00	-	-	-	-
	01	-	-	-	-
	11	0	-	-	-
	10	-	-	1	-

$$K_3 = Q_1$$

		00	01	11	10
$Q_3Q_2$	00	-	-	-	-
	01	1	-	-	-
	11	1	-	-	-
	10	-	-	-	-

$$K_2 = 1$$

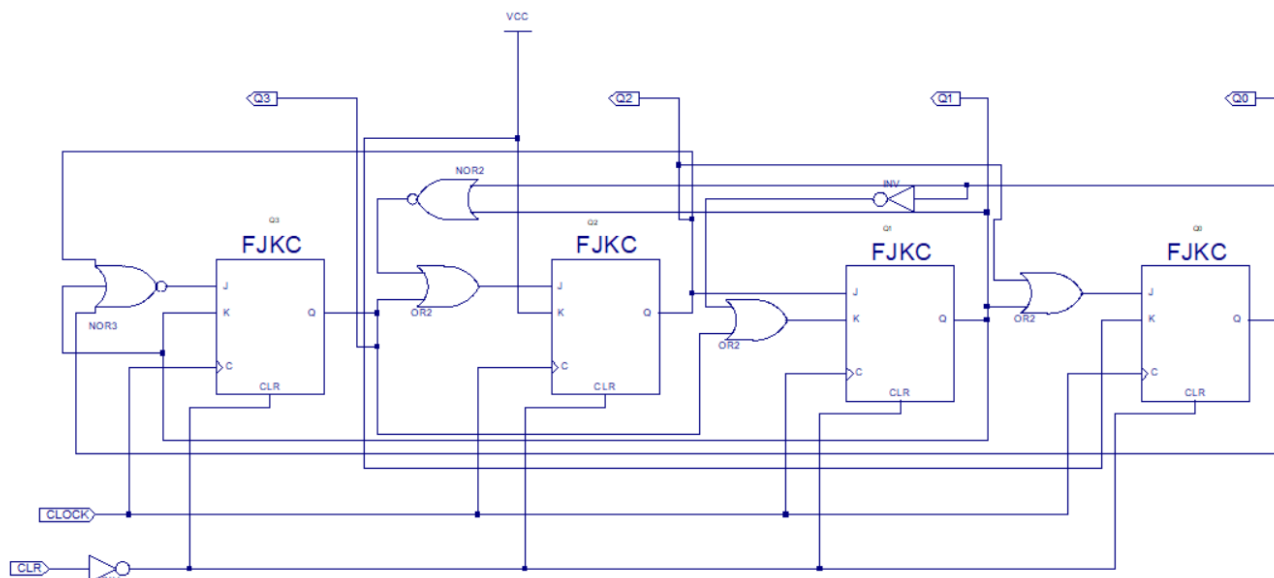
		00	01	11	10
$Q_3Q_2$	00	-	-	0	1
	01	-	-	-	-
	11	-	-	-	-
	10	-	-	1	-

$$K_1 = \overline{Q_0} + Q_3$$

		00	01	11	10
$Q_3Q_2$	00	-	1	1	-
	01	-	-	-	-
	11	-	-	-	-
	10	-	-	1	-

$$K_0 = 1$$

### 1.2.4 Schemat układu



### 1.2.5 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 AikenCounterScheme_AikenCounterScheme_sch_tb IS
7  END AikenCounterScheme_AikenCounterScheme_sch_tb;
8  ARCHITECTURE behavioral OF
    AikenCounterScheme_AikenCounterScheme_sch_tb IS
9
10     COMPONENT AikenCounterScheme
11     PORT( Q2 : OUT STD_LOGIC;
12           Q1  : OUT STD_LOGIC;
13           Q0  : OUT STD_LOGIC;
14           CLOCK : IN  STD_LOGIC;
15           Q3   : OUT STD_LOGIC;
16           CLR  : IN  STD_LOGIC);
17     END COMPONENT;
18
19     SIGNAL Q2  : STD_LOGIC;
20     SIGNAL Q1  : STD_LOGIC;
21     SIGNAL Q0  : STD_LOGIC;
22     SIGNAL CLOCK : STD_LOGIC := '0';
23     SIGNAL Q3   : STD_LOGIC;
24     SIGNAL CLR  : STD_LOGIC;
25
26 BEGIN
27
28     UUT: AikenCounterScheme PORT MAP(
29         Q2 => Q2,
30         Q1 => Q1,

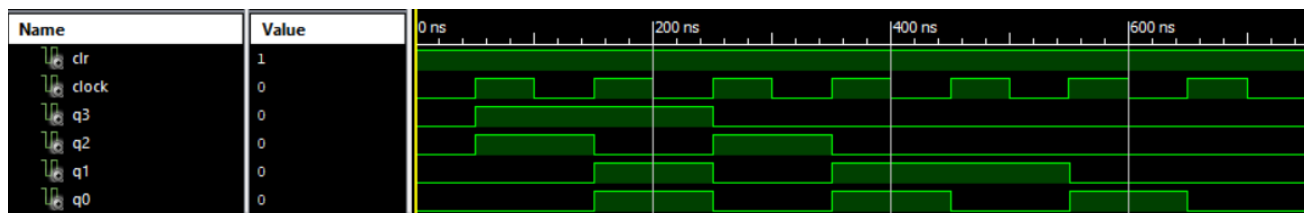
```

```

31      Q0 => Q0,
32      CLOCK => CLOCK,
33      Q3 => Q3,
34      CLR => CLR
35  );
36  CLR <= '1';
37  CLOCK <= not CLOCK after 50 ns;
38
39  END;

```

### 1.2.6 Symulacja



## 2 Zadanie 2

## 2.1 Polecenie

Detektor sekwencji 11011, automat Mealy-ego, jedno wejście, jedno wyjście, brak resetu, sekwencja prawidłowa 5-bitowa.

## 2.2 Rozwiązanie

### 2.2.1 Opis symboliki

## Alfabet wejściowy

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

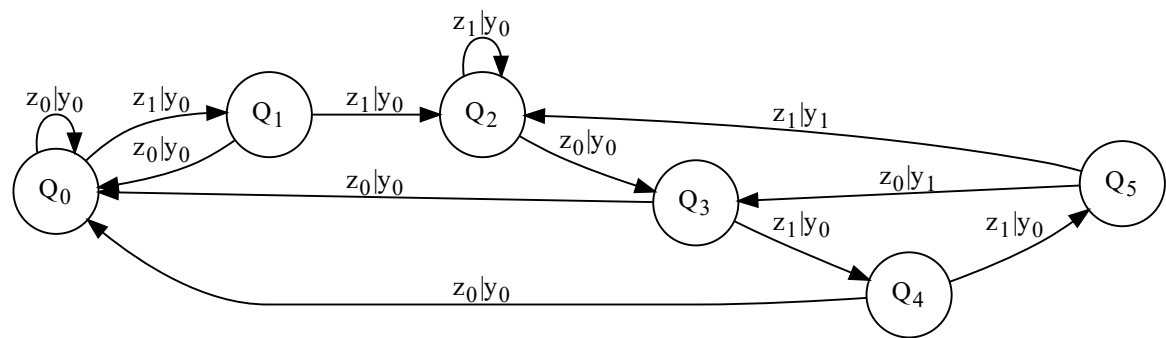
## Stany wewnętrzne

- $q_0$  - 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

2.2.2 Schemat grafowy



## 2.2.3 Tabela prawdy

S	Q(t)			Z	Q(t+1)			Y	D(t)		
	$Q_2$	$Q_1$	$Q_0$		$Q_2$	$Q_1$	$Q_0$		$T_2$	$T_1$	$T_0$
$Q_0$	0	0	0	0	0	0	0	0	0	0	0
$Q_0$	0	0	0	1	0	0	1	0	0	0	1
$Q_1$	0	0	1	0	0	0	0	0	0	0	1
$Q_1$	0	0	1	1	0	1	0	0	0	1	1
$Q_2$	0	1	0	0	0	1	1	0	0	0	1
$Q_2$	0	1	0	1	0	1	0	0	0	0	0
$Q_3$	0	1	1	0	0	0	0	0	0	1	1
$Q_3$	0	1	1	1	1	0	0	0	1	1	1
$Q_4$	1	0	0	0	0	0	0	0	1	0	0
$Q_4$	1	0	0	1	1	0	1	0	0	0	1
$Q_5$	1	0	1	0	0	1	1	1	1	1	0
$Q_5$	1	0	1	1	0	1	0	1	1	1	1
-	1	1	0	0	-	-	-	-	-	-	-
-	1	1	0	1	-	-	-	-	-	-	-
-	1	1	1	0	-	-	-	-	-	-	-
-	1	1	1	1	-	-	-	-	-	-	-

### 2.2.4 Siatka Karnaugh

		$Q_0Z$			
		00	01	11	10
$Q_2Q_1$	00	0	0	0	0
	01	0	0	1	0
	11	-	-	-	-
	10	1	0	1	1

$$T_2 = Q_1Q_0Z + Q_2\bar{Z} + Q_2Q_0$$

		$Q_0Z$			
		00	01	11	10
$Q_2Q_1$	00	0	0	1	0
	01	0	0	1	1
	11	-	-	-	-
	10	0	0	1	1

$$T_1 = Q_0Z + Q_1Q_0 + Q_2Q_0$$

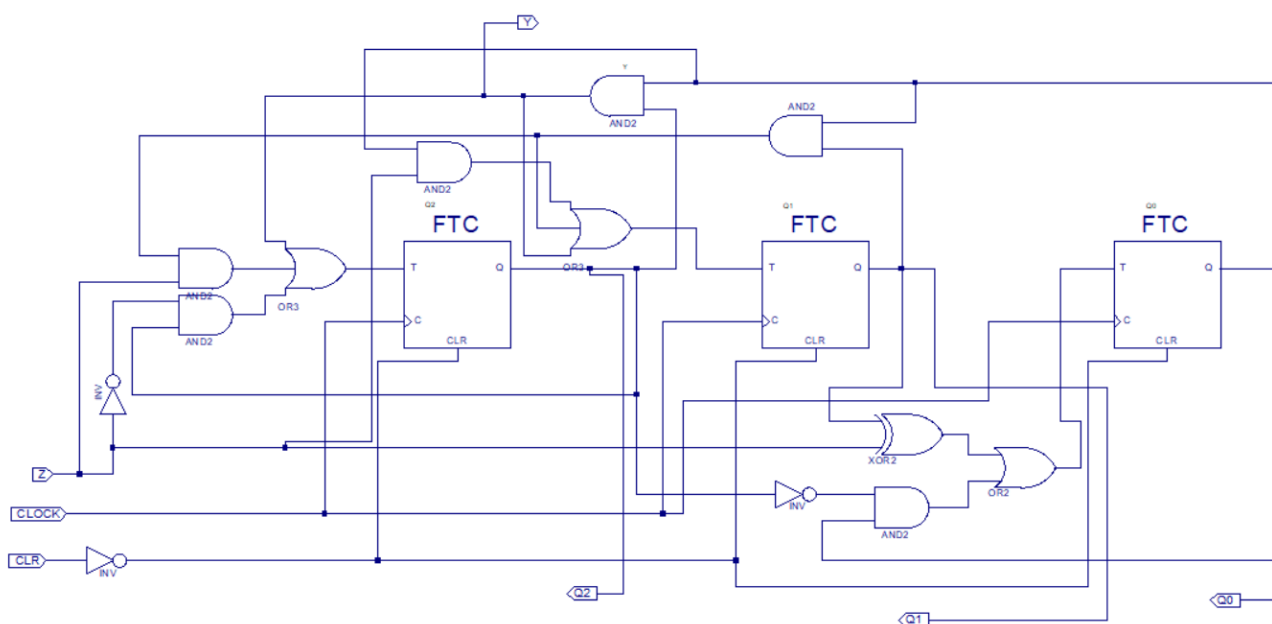
		$Q_0Z$			
		00	01	11	10
$Q_2Q_1$	00	0	1	1	1
	01	1	0	1	1
	11	-	-	-	-
	10	0	1	1	0

$$T_0 = \bar{Q}_2Q_0 + Q_1\bar{Z} + \bar{Q}_1Z$$

		$Q_0Z$			
		00	01	11	10
$Q_2Q_1$	00	0	0	0	0
	01	0	0	0	0
	11	-	-	-	-
	10	0	0	1	1

$$Y = Q_2Q_0$$

### 2.2.5 Schemat układu



### 2.2.6 Kod VHDL

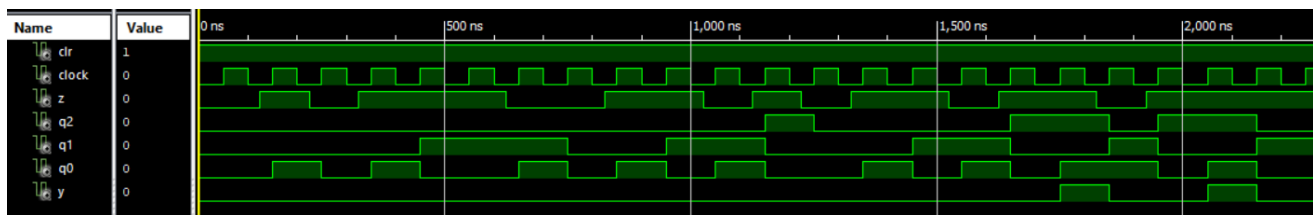


```

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 MealyDetectorScheme_MealyDetectorScheme_sch_tb IS
7  END MealyDetectorScheme_MealyDetectorScheme_sch_tb;
8  ARCHITECTURE behavioral OF
    MealyDetectorScheme_MealyDetectorScheme_sch_tb IS
9
10     COMPONENT MealyDetectorScheme
11     PORT( Y   : OUT STD_LOGIC;
12           Q0  : OUT STD_LOGIC;
13           Q2  : OUT STD_LOGIC;
14           Z   : IN  STD_LOGIC;
15           CLOCK : IN  STD_LOGIC;
16           Q1  : OUT STD_LOGIC;
17           CLR  : IN  STD_LOGIC);
18     END COMPONENT;
19
20     SIGNAL Y : STD_LOGIC;
21     SIGNAL Q0 : STD_LOGIC;
22     SIGNAL Q2 : STD_LOGIC;
23     SIGNAL Z : STD_LOGIC;
24     SIGNAL CLOCK : STD_LOGIC := '0';
25     SIGNAL Q1 : STD_LOGIC;
26     SIGNAL CLR : STD_LOGIC ;
27
28 BEGIN
29
30     UUT: MealyDetectorScheme PORT MAP(
31         Y => Y,
32         Q0 => Q0,
33         Q2 => Q2,
34         Z => Z,
35         CLOCK => CLOCK,
36         Q1 => Q1,
37         CLR => CLR
38     );
39     CLR <= '1';
40     CLOCK <= not CLOCK after 50 ns;
41
42     Z <= '0', '1' after 125 ns, '0' after 225 ns, '1' after 325 ns,
43         '0' after 625 ns, '1' after 825 ns, '0' after 1025 ns, '1' after
44         1125 ns, '0' after 1225 ns, '1' after 1325 ns, '0' after 1525
45         ns, '1' after 1625 ns, '0' after 1825 ns, '1' after 1925 ns;
46
47 END;

```

## 2.2.7 Symulacja



## 3 Wnioski

Przy wykonywaniu zadań niezbędna okazała się wiedza na temat typów przerzutników i ich działania. Oprócz tego, ćwiczenia wymagały także zapoznania się z opracowaniem dotyczącym zestawu ZL-9572, a konkretniej - generatorów sygnałów prostokątnych **Clk\_XT** oraz **Clk\_LF** oraz modułów impulsatora (Rotary Encoder) oraz czterocyfrowego wyświetlacza siedmiosegmentowego. Pierwsze zadanie dotyczyło stworzenia licznika negatywnego modulo 7 zliczającego w kodzie Aike-na. Wykorzystaliśmy w jego rozwiązaniu przerzutniki JK, a konkretnie komponenty FJKC z biblioteki programu Xilinx, przyjmując przy okazji założenie, że praca licznika rozpoczyna się od jego resetu. Możliwą alternatywą były stworzenie schematu opartego na komponentach FJKP, FJKCP lub rozwiązanie mieszane. Dodatkowe wejście **Prereset** pełniłoby wówczas rolę stanu domyślnego dla przerzutnika, i pozwalałoby mu przyjąć określony stan na początku symulacji, bez konieczności resetu.

Podobne założenie dotyczące pochodzenia stanu początkowego przyjęliśmy także w zadaniu 2. Należało w nim stworzyć układ detektora sekwencji "11011" oparty na automacie Mealy'ego. Przyjęliśmy, że układ prawidłowo reaguje na dowolnie długą sekwencję, o ile kończy się ona wcześniej wspomnianą, 5-bitową; możliwe jest w szczególności kilkukrotne wykrycie poprawnej sekwencji w trakcie pracy układu.

W rozszerzonej wersji, zadanie 2 należało rozszerzyć o obsługę wejścia przy pomocy modułu Rotary Encoder, a także prezentację aktualnego stanu układu przy pomocy wyświetlacza siedmiosegmentowego. Niestety, nie udało się zrealizować tej wersji zadania w całości: powiodło się zaimportowanie plików z gotowymi modułami do projektu. Wykorzystujący je schemat okazał się jednak niesyntezowalny. Podejrzewamy, że może mieć to związek z niepoprawnym użyciem przez nas magistrali i błędnym przyporządkowaniem pojedynczych ścieżek w niej zawartych.