Technika Cyfrowa - Sprawozdanie 2

Projekt czterobitowego licznika Fibbonaci'ego

Autorzy

- Kacper Feliks
- Robert Raniszewski
- Paweł Czajczyk
- Mateusz Pawliczek

---

1. Treść zadania

Korzystając tylko z konkretnego jednego typu przerzutników oraz z dowolnych bramek logicznych, proszę zaprojektować czterobitowy licznik działający zgodnie z ciągiem Fibonacciego (z nieobowiązkowym upraszczającym zastrzeżeniem, że wartość "1" powinna się pojawiać tylko raz w cyklu). Po uruchomieniu licznika, w kolejnych taktach zegara powinien on zatem przechodzić po wartościach:

0, 1, 2, 3, 5, 8, 13, 0, 1, 2, 3, 5, 8, 13, 0, 1, ... itd.

Aktualna wartość wskazywana przez licznik powinna być widoczna na wyświetlaczach siedmiosegmentowych.

---

2. Wstęp

Zaprojektowany przez nas licznik działa w pętli i wartość jest wyświetlana na wyświetlaczu siedmiosegmentowym. W projekcie wykorzystano **jeden typ przerzutnika** (T) oraz dowolne bramki logiczne. Rysunek poglądowy układu wygląda następująco:

----------------------------------- **Rysunek 1.1** Schemat licznika Fibonacciego -----------------------------------

Układ posiada 2 wejścia: zegarowe i reset oraz 4 wyjścia dla bitów, na których zapisana jest liczba należąca do ciągu

---

3. Budowa schematu przejść

Ciąg Fibbonaciego - Reprezentacja binarna

W celu wyświetlenia ciągu binarnego na liczniku cyfrowym potrzebna jest jego binarna reprezentacja. Fragment ciągu, którego wymaga treść zadania wygląda następująco:

Ciąg :

```
0, 1, 2, 3, 5, 8, 13
```

Reprezentacja binarna ciągu :

Liczba	Q1	Q2	Q3	Q4
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
5	0	1	0	1
8	1	0	0	0
13	1	1	0	1

Przejścia liczb w ciągu

Liczby wyświetlane są wyświetlanie na ekranie cyfrowego licznika zgodnie z kolejnością ciągu fibbonaciego.

```
0 (0000) → 1 (0001) → 2 (0010) → 3 (0011) →
→ 5 (0101) → 8 (1000) → 13 (1101) → 0 (0000) ...
```

Dla każdego z 4 bitów można rozatrzyć przejścia T między każdymi dwoma liczbami w ciągu. Przejście jest rozumiane jako przełączenie przerzutnika

```
0 0 0 0  →  T3 T2 T1 T0
```

Przejście T0 (najmłodszy bit)

stan aktualny	stan następny	Q0 ➔ ~Q0	T0 - zmiana?
000**0**	000**1**	0 ➔ 1	1
000**1**	001**0**	1 ➔ 0	1
001**0**	001**1**	0 ➔ 1	1
001**1**	010**1**	1 ➔ 1	0
010**1**	100**0**	1 ➔ 0	1
100**0**	110**1**	0 ➔ 1	1
110**1**	000**0**	1 ➔ 0	1

Przejście T1

stan aktualny	stan następny	Q1 ➔ ~Q1	T1 - zmiana?
00**0**0	00**0**1	0 ➔ 0	0
00**0**1	00**1**0	0 ➔ 1	1
00**1**0	00**1**1	1 ➔ 1	0
00**1**1	01**0**1	1 ➔ 0	1
01**0**1	10**0**0	0 ➔ 0	0
10**0**0	11**0**1	0 ➔ 0	0
11**0**1	00**0**0	0 ➔ 0	0

Przejście T2

stan aktualny	stan następny	Q2 ➔ ~Q2	T2 - zmiana?
0**0**00	0**0**01	0 ➔ 0	0
0**0**01	0**0**10	0 ➔ 0	0
0**0**10	0**0**11	0 ➔ 0	0
0**0**11	0**1**01	0 ➔ 1	1
0**1**01	1**0**00	1 ➔ 0	1
1**0**00	1**1**01	0 ➔ 1	1
1**1**01	0**0**00	1 ➔ 0	1

Przejście T3 (najstarszy bit)

stan aktualny	stan następny	Q3 ➜ ~Q3	T3 - zmiana?
0000	**0**001	0 ➜ 0	0
0001	**0**010	0 ➜ 0	0
0010	**0**011	0 ➜ 0	0
0011	**0**101	0 ➜ 0	0
0101	**1**000	0 ➜ 1	1
1000	**1**101	1 ➜ 1	0
1101	**0**000	1 ➜ 0	1

4. Tabele Karnaugh

Tabela 4.1 Tabela Karnaugh dla wejścia T_3 w czasie n

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

Z tabeli 4.1 wynika wzór na T_3 : $T_3 = Q_2\bar{Q}_1Q_0$

Tabela 4.2 Tabela Karnaugh dla wejścia T_2 w czasie n

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

Z tabeli 4.2 wynika wzór na T_2 :

$T_2 = Q_2\bar{Q}_1Q_0 + \bar{Q}_3\bar{Q}_2Q_1Q_0 + Q_3\bar{Q}_2\bar{Q}_1\bar{Q}_0$

Tabela 4.3 Tabela Karnaugh dla wejścia T_1 w czasie n

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

Z tabeli 4.3 wynika wzór na T_1 :

$T_1 = \bar{Q}_3\bar{Q}_2Q_0$

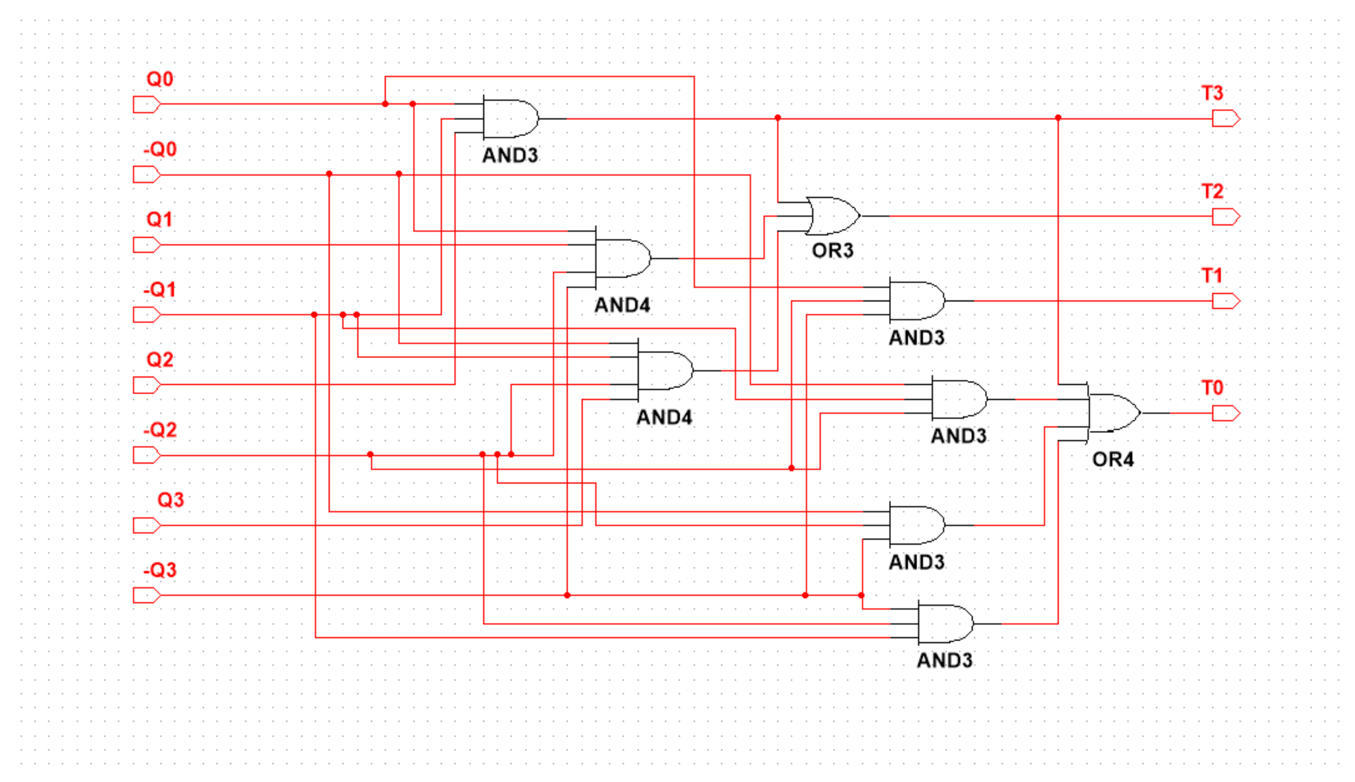
Tabela 4.4 Tabela Karnaugh dla wejścia T_0 w czasie n

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

Z tabeli 4.4 wynika wzór na T_0 :

$T_0 = \bar{Q}_3\bar{Q}_2\bar{Q}_1 + \bar{Q}_3\bar{Q}_2\bar{Q}_0 + Q_2\bar{Q}_1Q_0 + \bar{Q}_2\bar{Q}_1\bar{Q}_0$

Wykorzystując wyprowadzone wzory przygotowaliśmy implementację w multisimie:



Rysunek 4.1 Implementacja podukładu "Logika"

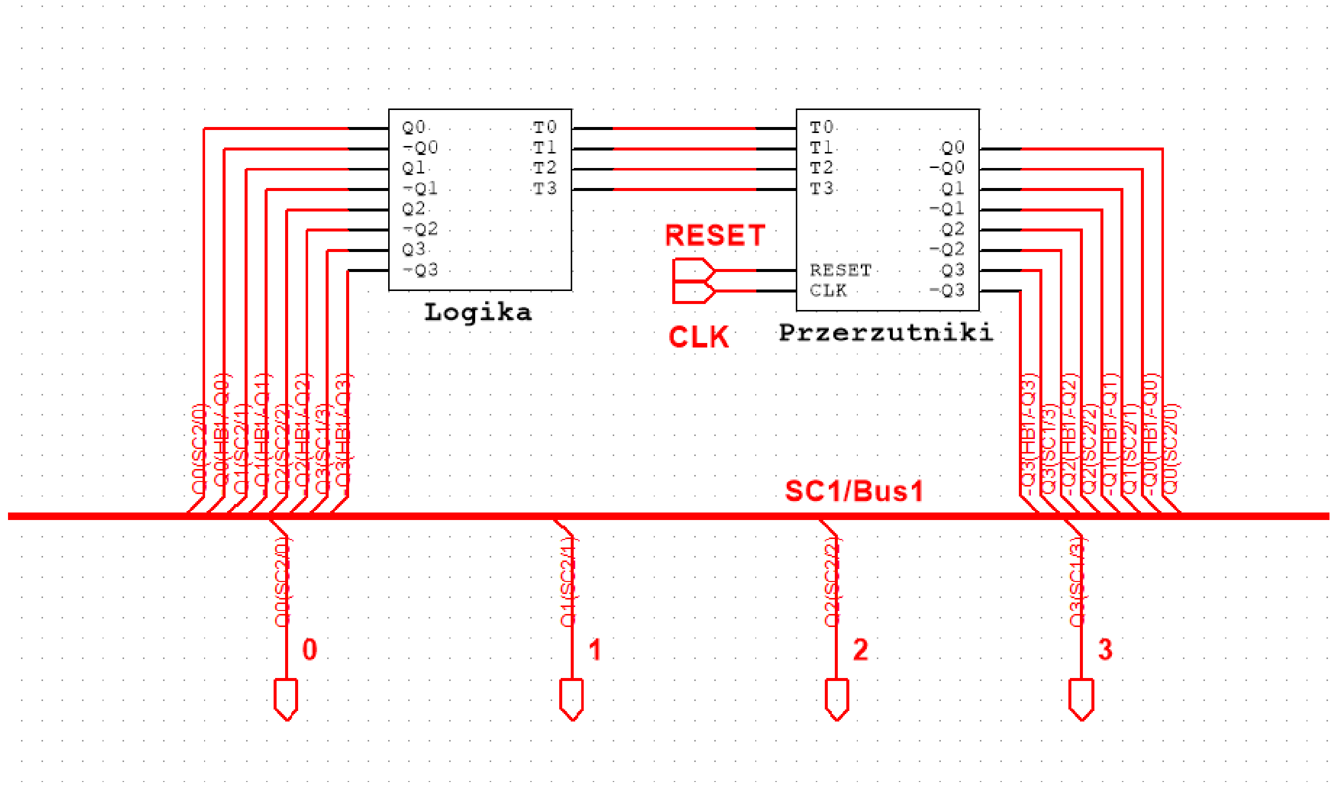
W implementacji uwzględniliśmy powtarzający się fragment wzorów ($Q_2\bar{Q}_1Q_0$ w T3, T2 i T0), co pozwoliło na zmniejszenie liczby bramek z 11 na 9.

5. Schemat układu

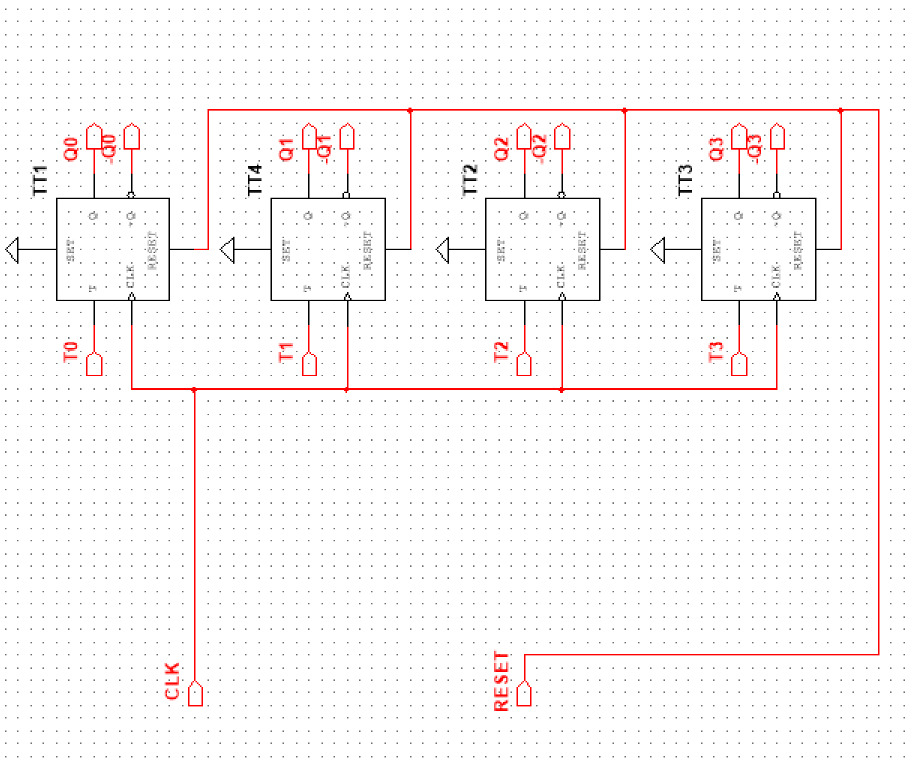
Zaprojektowany licznik wygląda następująco:

Rysunek 5.1 Schemat licznika Fibonacciego w programie Multisim

Poniżej przedstawiona jest implementacja:



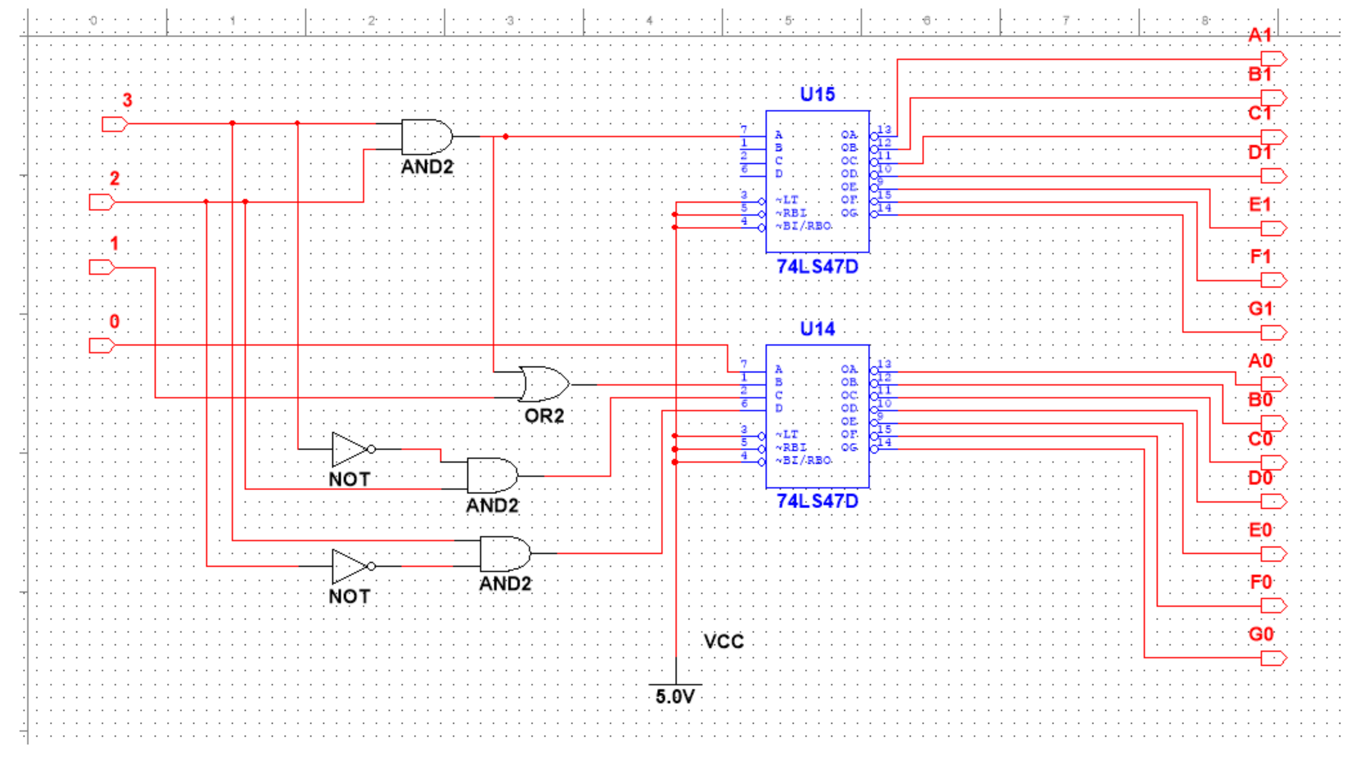
Rysunek 5.2 Implementacja licznika



Rysunek 5.3 Implementacja podukładu "Przerzutniki"

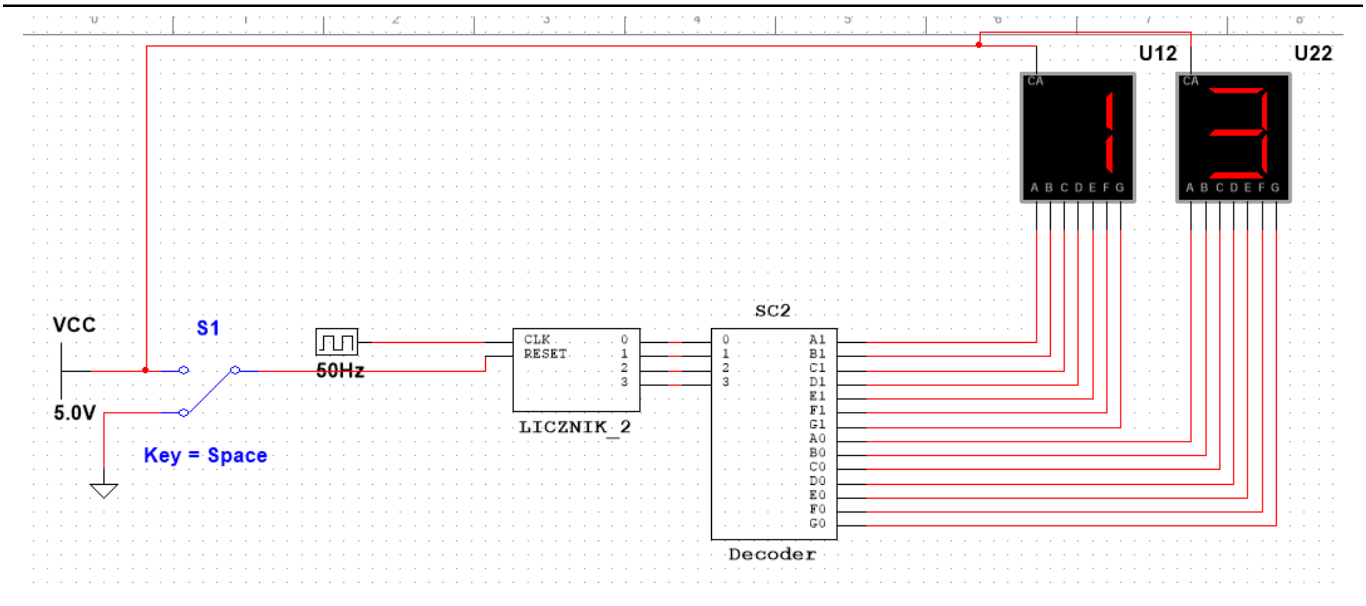
Przedstawione schematy wykorzystują magistrale komunikacyjne. Magistrale te służą do komunikacji między poszczególnymi blokami układu oraz stanowią graficzne uproszczenie układu.

Aby wyświetlić liczby na 2 wyświetlaczach siedmiosegmentowych zaprojektowaliśmy odpowiedni dekoder korzystający z konwerterów BCD-TO-7-SEGMENT-DISPLAY:



Rysunek 4.9 Implementacja podukładu "Decoder"

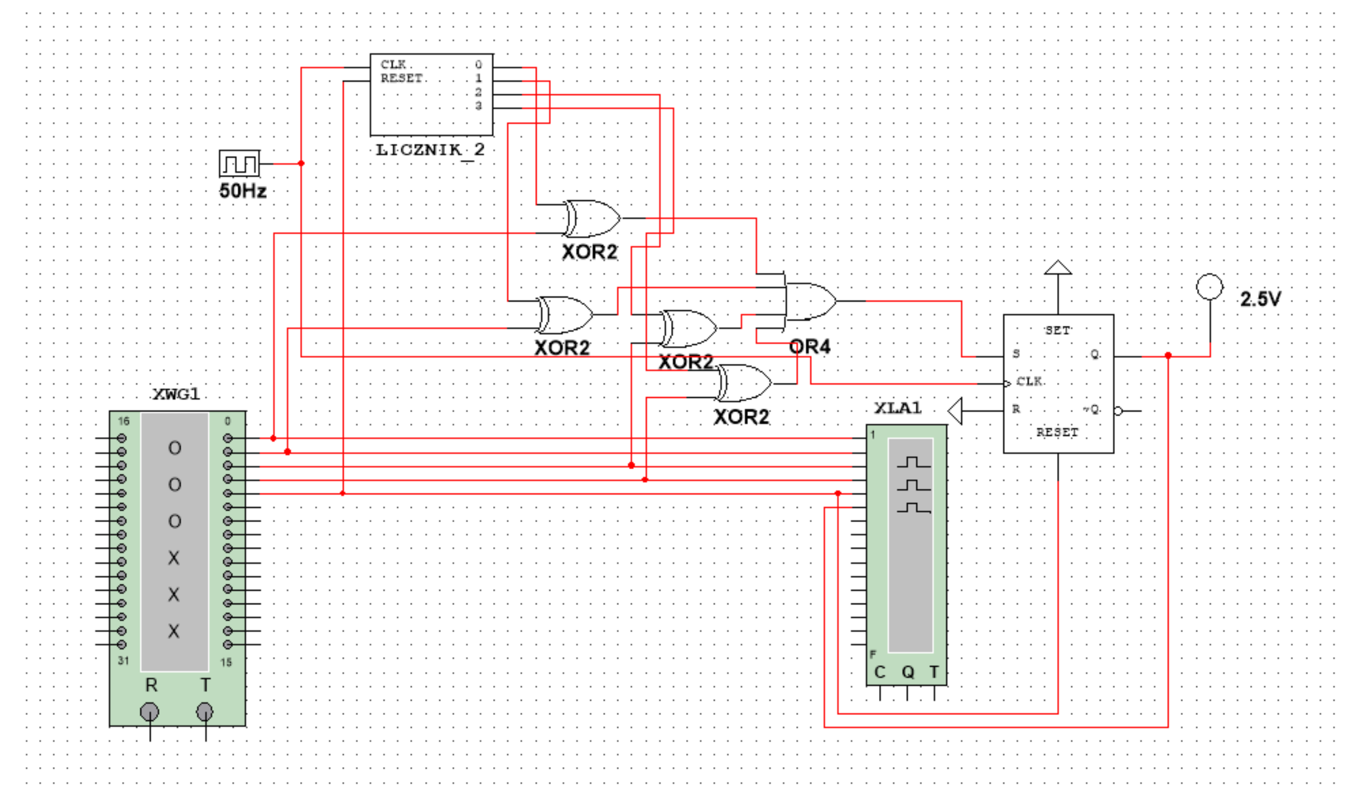
Gotowy układ wraz z wyświetlaczami siedmiosegmentowymi wygląda następująco:



Rysunek 4.10 Implementacja gotowego układu

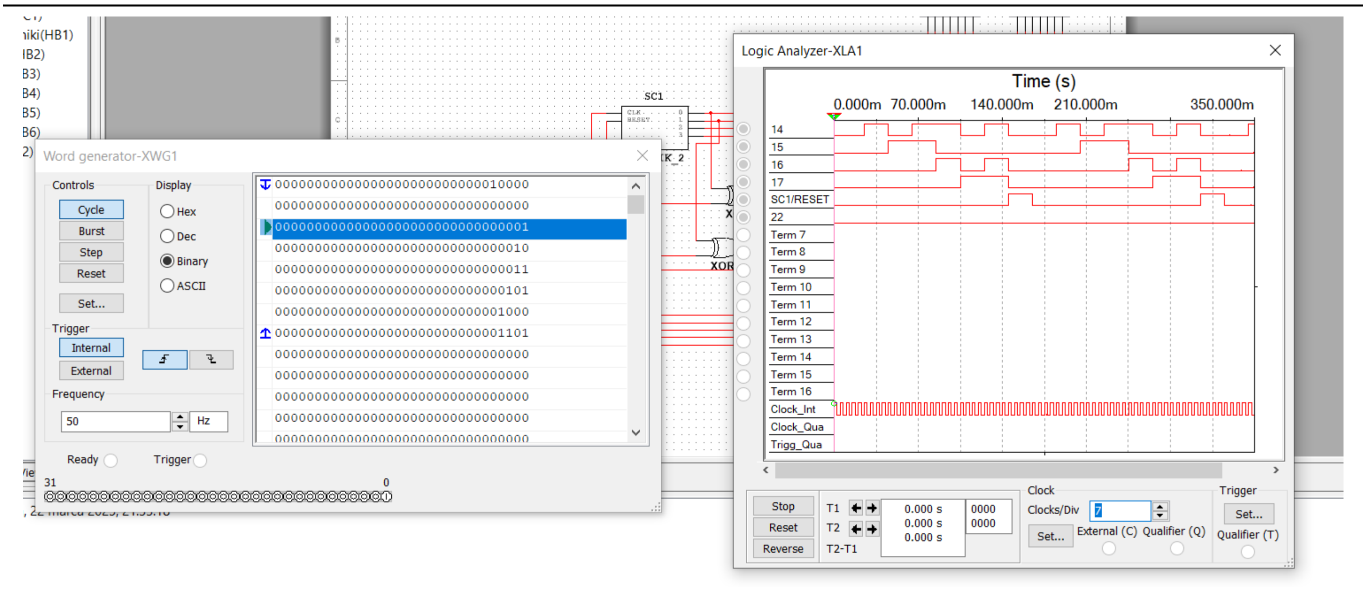
5. Układ testowy

Korzystając z wcześniejszych podukładów zrobiliśmy układ testowy w celu sprawdzenia poprawności naszego licznika, korzystając z generatora słów oraz analizatora stanów logicznych. Gdy układ jest wadliwy dioda zapala się na czerwono



Rysunek 5.1 Układ testowy

Poniżej znajdują się wyniki analizatora logicznego wraz z ustawieniem generatora słów:



Rysunek 5.2 Wyniki testów

Na podstawie analizowanych testów widać, że sekwencja czterech bitów zmienia się zgodnie z oczekiwaniami, bit piąty spełnia funkcję resetowania, a szósty bit pozostaje w stanie niskim, co potwierdza poprawne działanie układu. Generator słów wprowadza kolejne sekwencje testowe, a układ reaguje prawidłowo na wszystkie badane kombinacje wejściowe.

Wnioski

- Układ poprawnie realizuje zapętlony ciąg Fibonacciego.
- Minimalna liczba przerzutników potrzebna do stworzenia 4-bitowego licznika to 4
- Schemat bramek logicznych można bardziej uprościć korzystając ze wspólnych podfragmentów wzorów, co pozwoliłoby na dalsze zmniejszanie liczby bramek,

---

Praktyczne zastosowania

- Tego typu licznik można zastosować w systemach losowych lub efektach świetlnych (np. animacje LED w sekwencji Fibonacciego), gdzie nieregularne sekwencje liczb zapewniają bardziej „naturalny” lub mniej przewidywalny efekt.
- Przedstawiony poniżej system wykorzystuje licznik do kontrolowania dostępu do lodówki. Lodówka automatycznie blokuje się po każdym otwarciu na czas zgodny z sekwencją licznika. Na wyświetlaczu widoczny jest aktualny czas blokady, a diody pokazują liczbę poprzednich otwarć. System można zresetować wrzucając monetę do skarbonki.