# Czterobitowy licznik działający zgodnie z ciągiem Fibonacciego

Szymon Borusiewicz, Jakub Zając, Dawid Szłapa, Radosław Szepielak

Maj 2025

## 1 Treść zadania

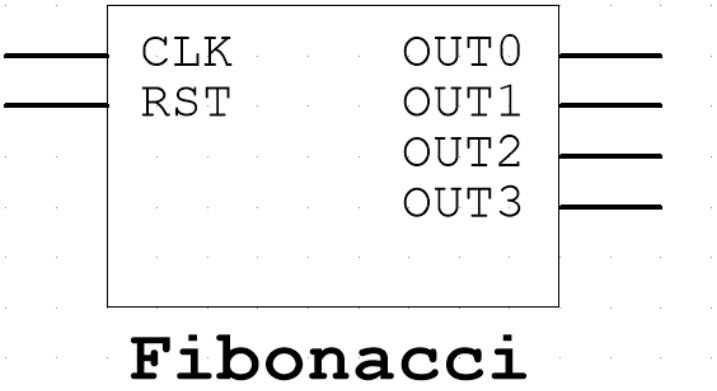
Korzystając tylko z konkretnego jednego typu przerzutników oraz z dowolnych bramek logicznych, proszę zaprojektować czterobitowy licznik działający zgodnie z ciągiem Fibonacciego (z nieobowiązkowym upraszczającym zastrzeżeniem, że wartość "1"powinna się pojawiać tylko raz w cyklu). Po uruchomieniu licznika,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

$0, 1, 2, 3, 5, 8, 13, 0, 1, 2, 3, 5, 8, 13, 0, 1, \ldots$ itd.

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

## 2 Czarna skrzynka

Układ, do którego dążymy, prezentuje się w następujący sposób:




Rysunek 1

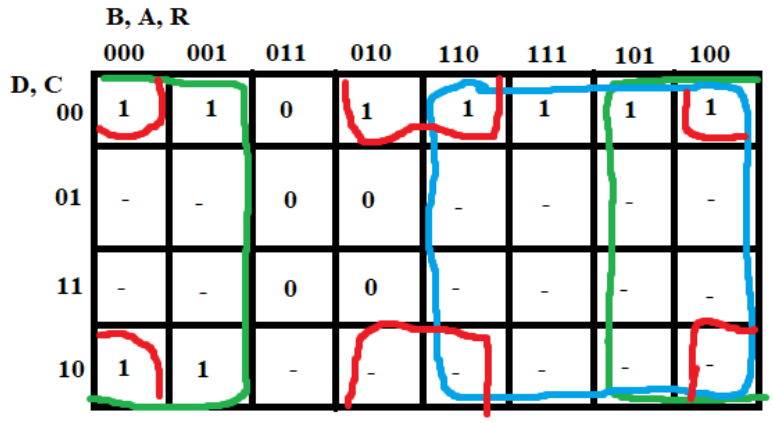
Gdzie wejściami są:

1. CLK - sygnał z zewnętrznego zegara
2. RST - sygnał służący zresetowaniu stanu licznika
3. OUT0, OUT1, OUT2, OUT3 - wyjścia odpowiedzialne za kolejną liczbę

## 3 Tablica przejść automatu

| R | D | C | B | A | Out3 | Out2 | Out1 | Out0 |
|---|---|---|---|---|---|---|---|---|
| X | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| X | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 |
| X | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 |
| X | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 |
| X | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 |
| X | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 |

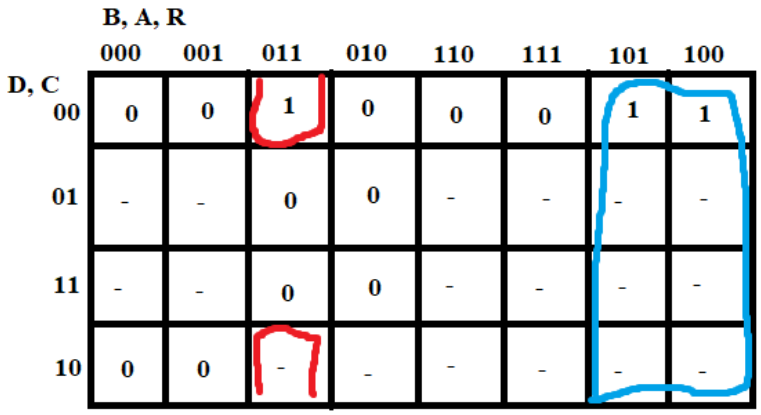
## 4 Tablice Karnaugh




| R | D | C | B | A | Out3 | Out2 | Out1 | Out0 |
|---|---|---|---|---|---|---|---|---|
| X | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| X | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 |
| X | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 |
| X | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 |
| X | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 |
| X | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 |

Rysunek 2: Tablica Karnaugh dla wyjścia Out0

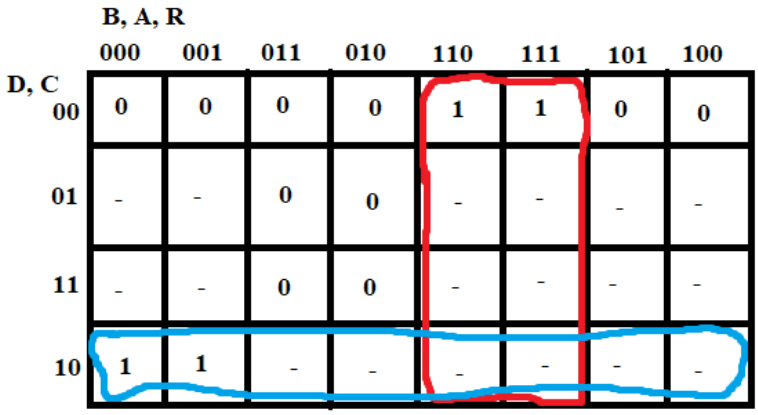
$\overline{A} + B + \overline{CR}$




| R | D | C | B | A | Out3 | Out2 | Out1 | Out0 |
|---|---|---|---|---|---|---|---|---|
| X | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| X | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 |
| X | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 |
| X | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 |
| X | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 |
| X | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 |

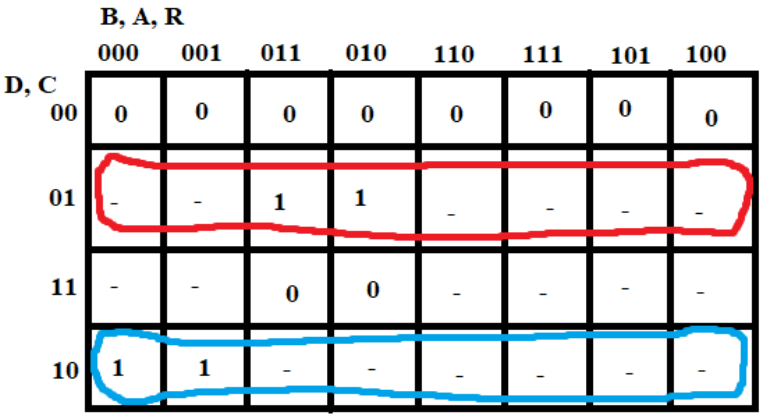
Rysunek 3: Tablica Karnaugh dla wyjścia Out1

$\overline{A}B + A\overline{B}\overline{C}R$




| R | D | C | B | A | Out3 | Out2 | Out1 | Out0 |
|---|---|---|---|---|---|---|---|---|
| X | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| X | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 |
| X | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 |
| X | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 |
| X | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 |
| X | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 |

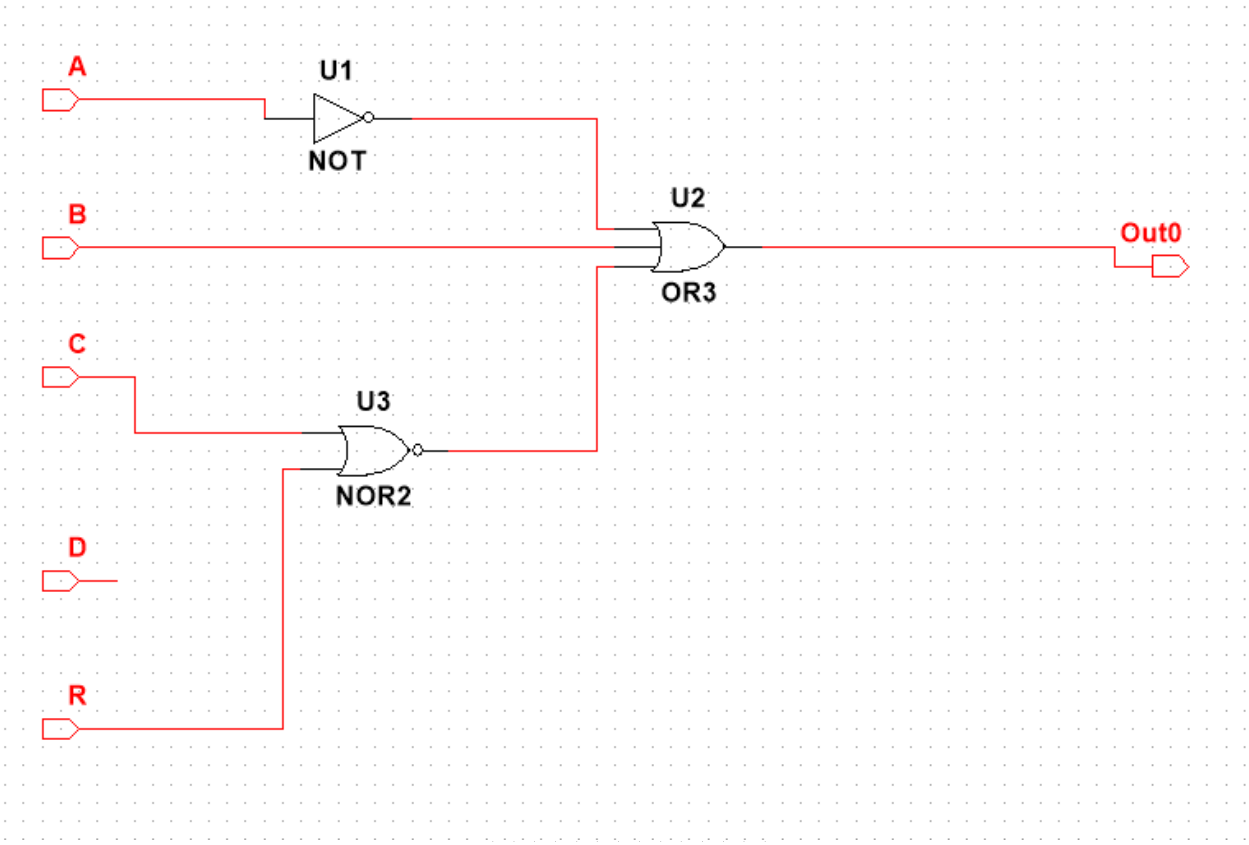
Rysunek 4: Tablica Karnaugh dla wyjścia Out2
$AB + D\overline{C}$

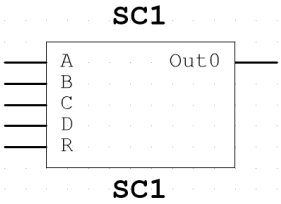



| R | D | C | B | A | Out3 | Out2 | Out1 | Out0 |
|---|---|---|---|---|---|---|---|---|
| X | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| X | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 |
| X | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 |
| X | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 |
| X | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 |
| X | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 |

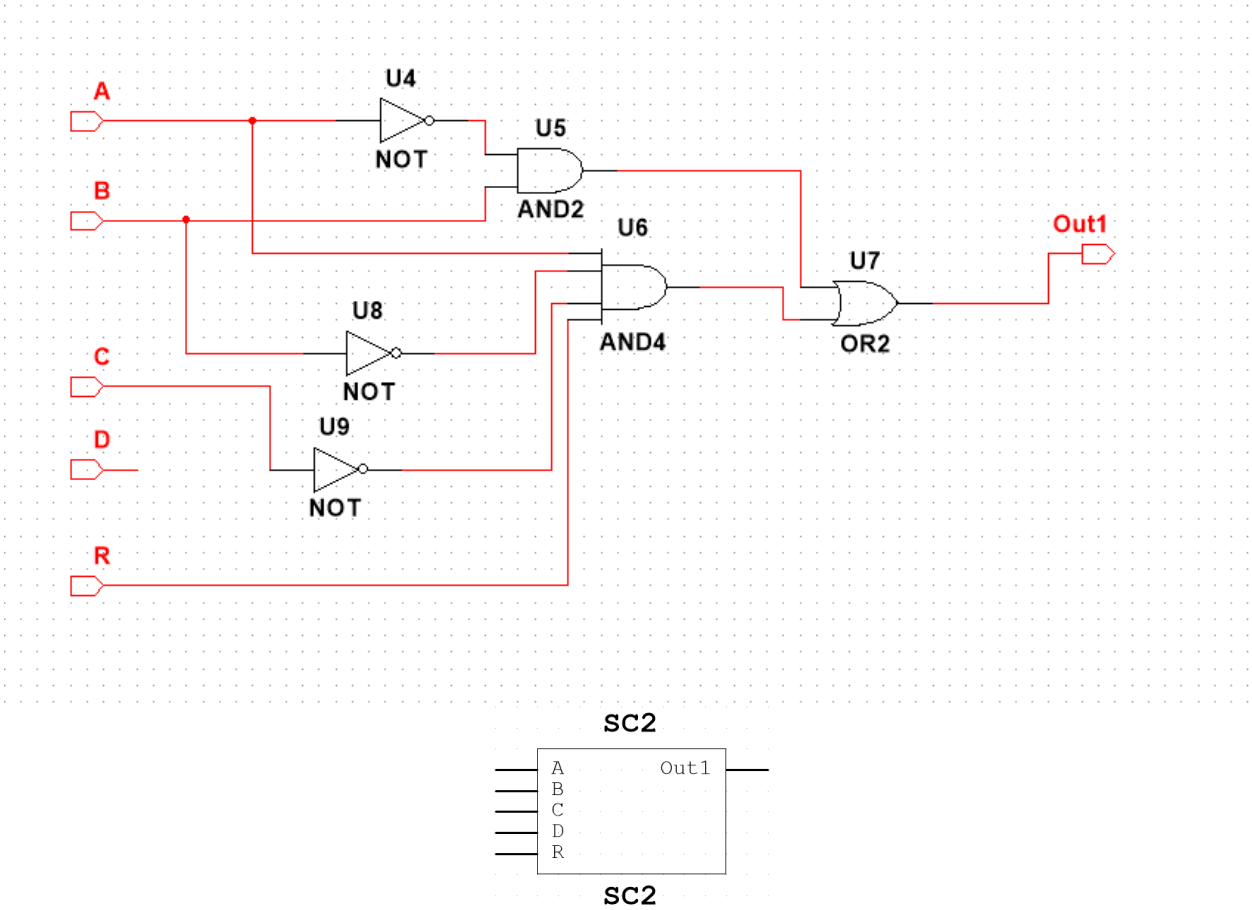
Rysunek 5: Tablica Karnaugh dla wyjścia Out3
$\overline{C}D + C\overline{D}$

# 5 Implementacja w programie Multisim

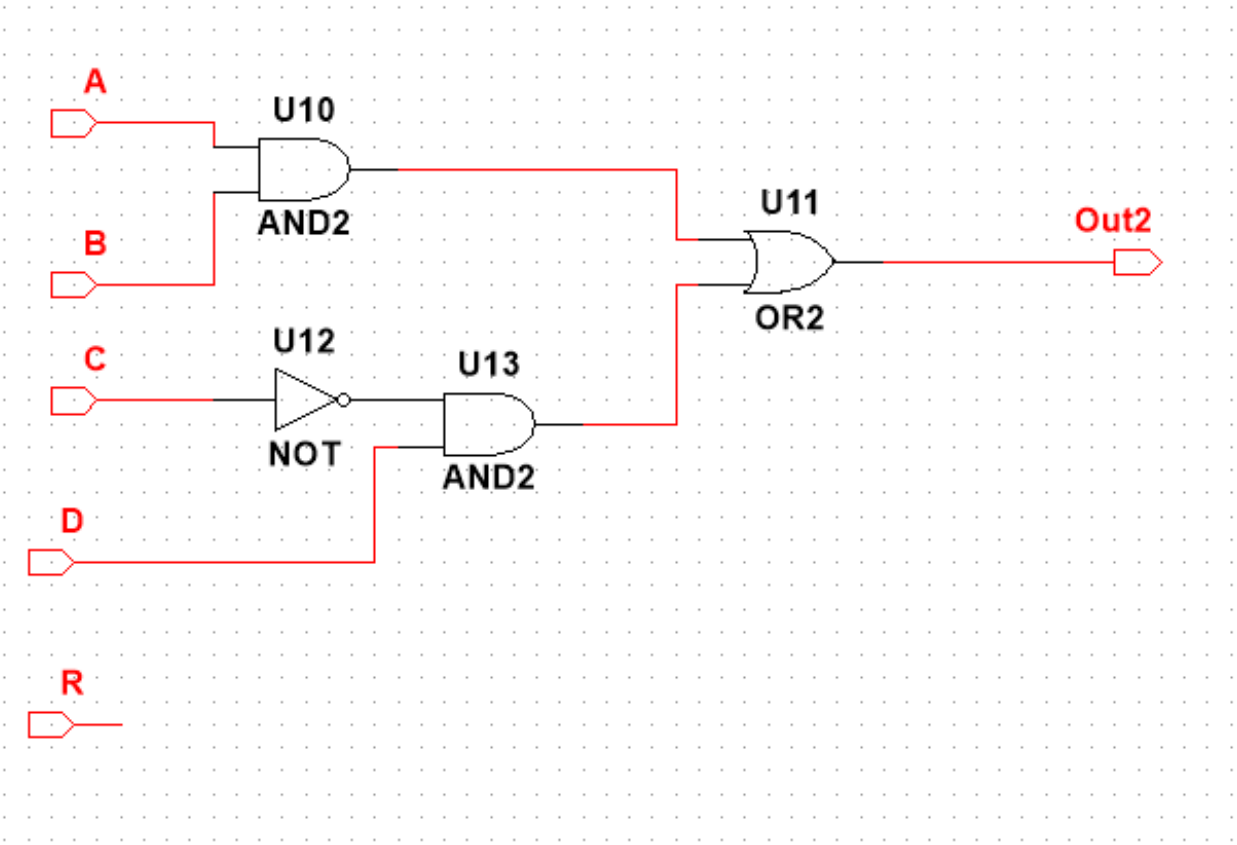


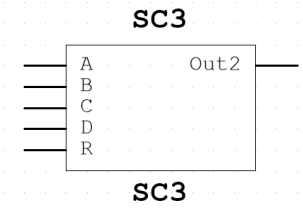




Rysunek 6: Implementacja podukładu SC1

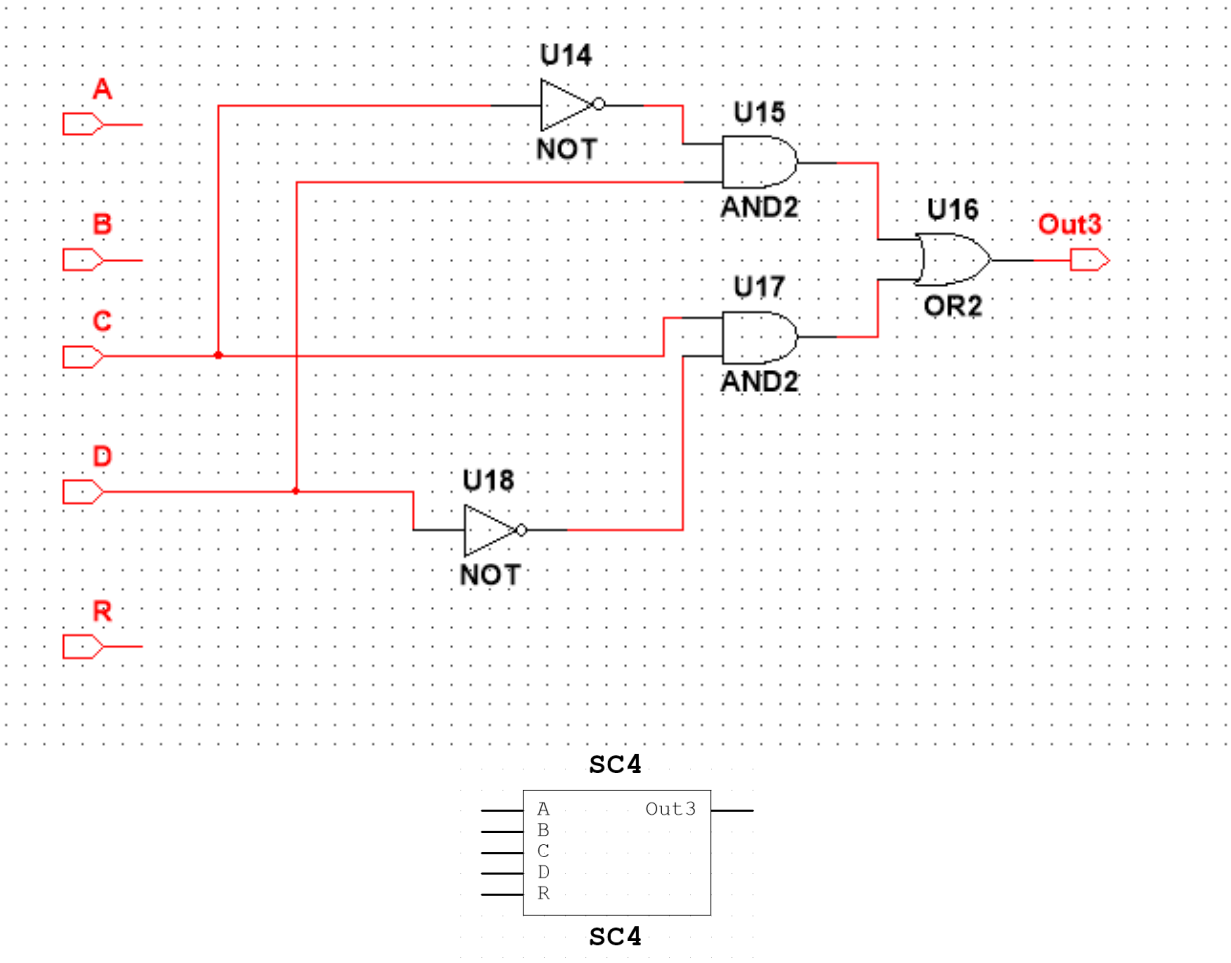



Rysunek 7: Implementacja podukładu SC2

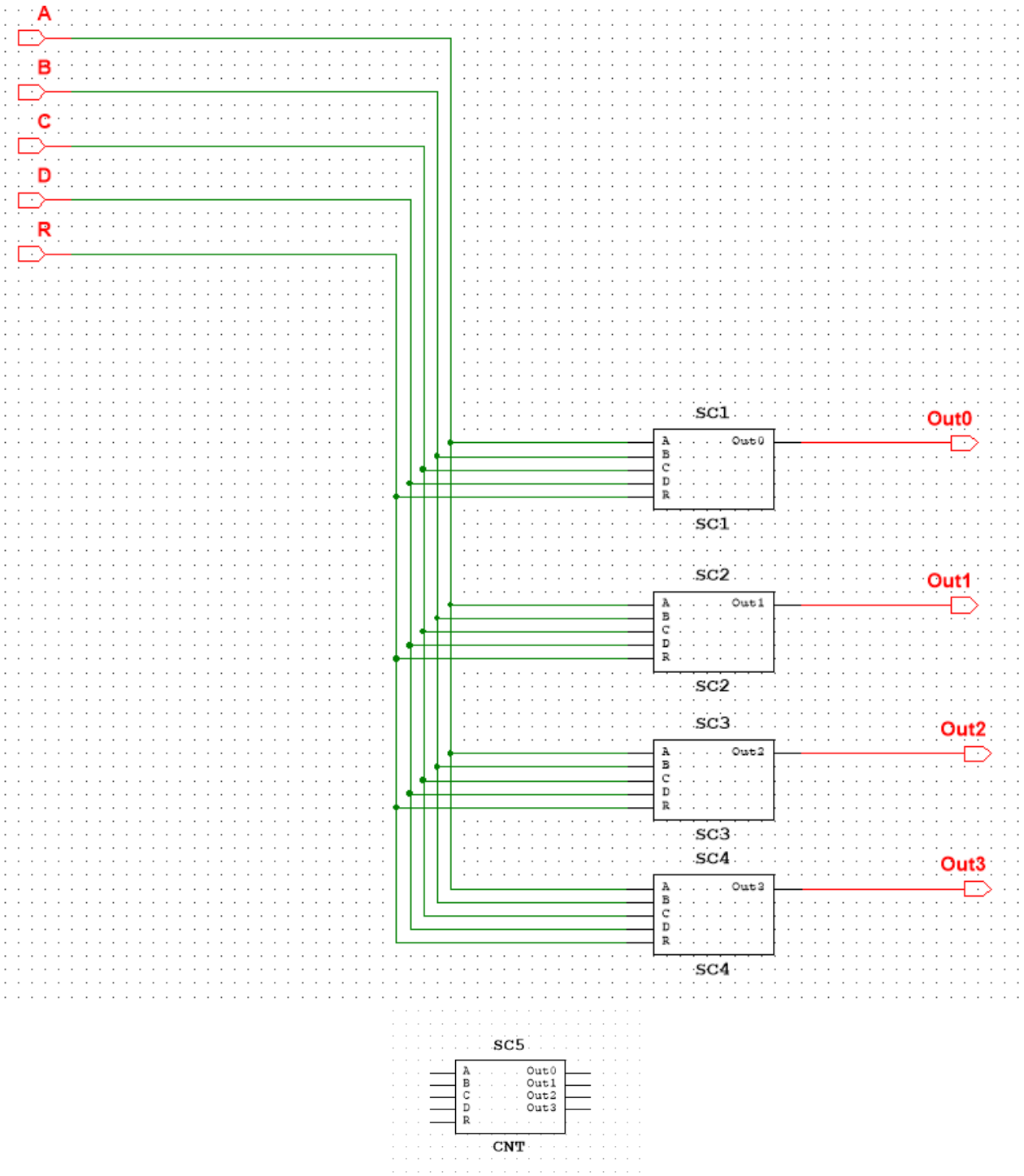


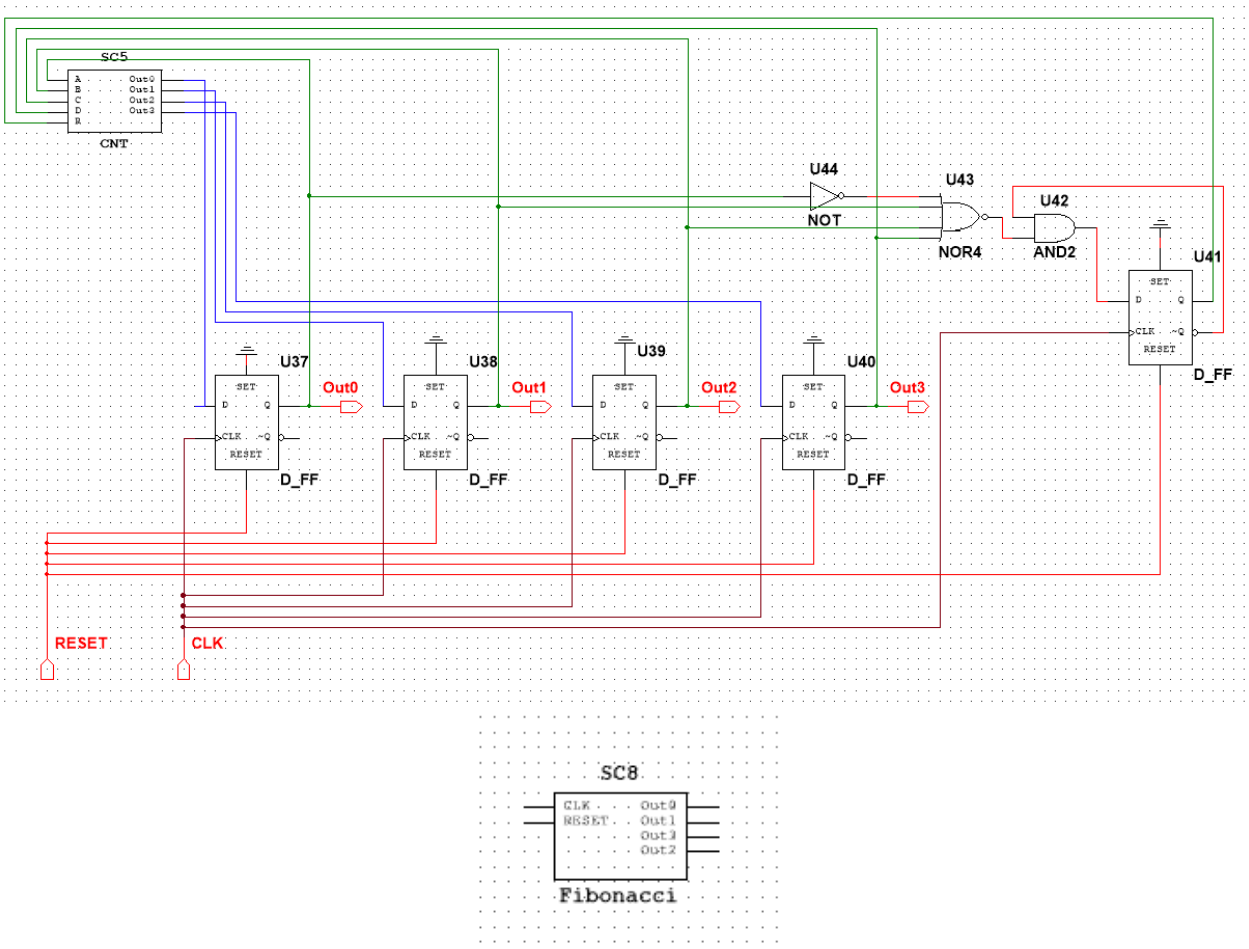

Rysunek 8: Implementacja podukładu SC3




Rysunek 9: Implementacja podukładu SC4




Rysunek 10: Implementacja podukładu CNT



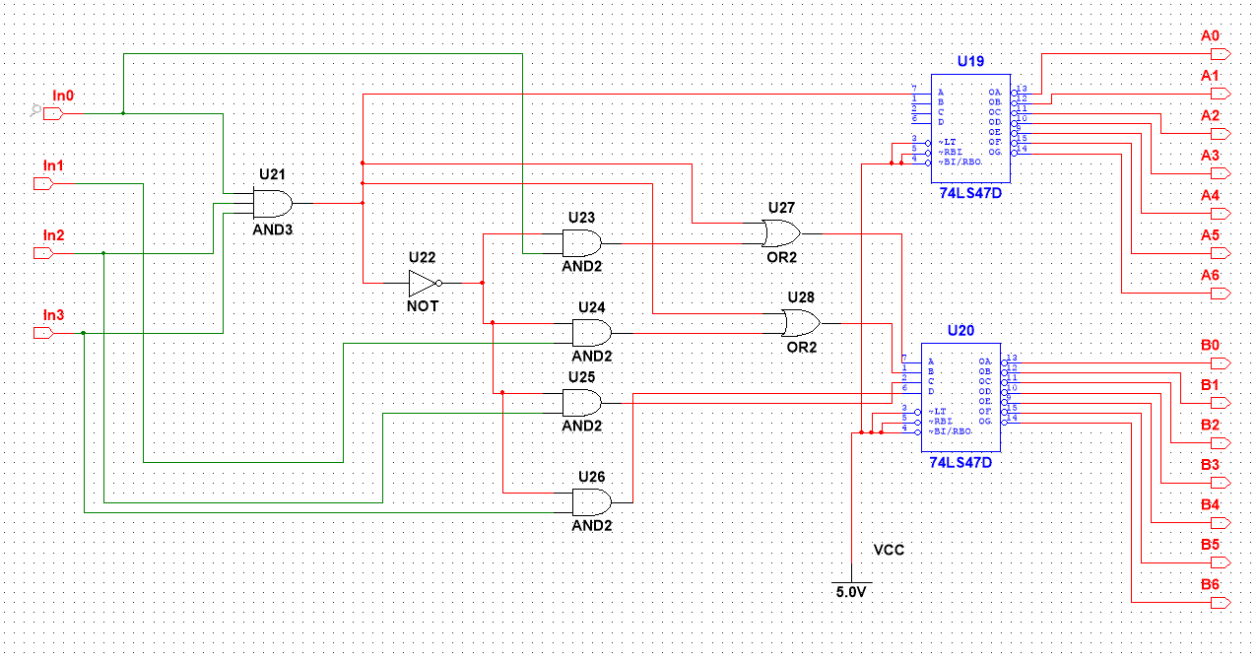

Rysunek 11: Implementacja podukładu Fibonacci

| D | C | B | A | Q | $D_o$ |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | X | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 | 1 |
| 0 | 0 | 1 | 0 | X | 0 |
| 0 | 0 | 1 | 1 | X | 0 |
| 0 | 1 | 0 | 1 | X | 0 |
| 1 | 0 | 0 | 0 | X | 0 |
| 1 | 1 | 0 | 1 | X | 0 |

Tabela 1: Tabela prawdy dla układu bramek U44, U43, U42

| CLK | D | SET | RESET | $Q_{n+1}$ |
|---|---|---|---|---|
| ↑ | 0 | 0 | 0 | 0 |
| ↑ | 1 | 0 | 0 | 1 |
| X | X | 0 | 0 | $Q_n$ |
| X | X | 1 | 0 | 1 |
| X | X | 0 | 1 | 0 |
| X | X | 1 | 1 | X |

Tabela 2: Tabela prawdy dla przerzutnika D




Rysunek 12: Implementacja transkodera

| D | C | B | A | A1 | B1 | C1 | D1 | A2 |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 |

Tabela 3: Tabela prawdy dla układu bramek U21, U22... U28

Wejścia układu U20 to A1, B1, C1, D1, natomiast wejście układu U19 to A2

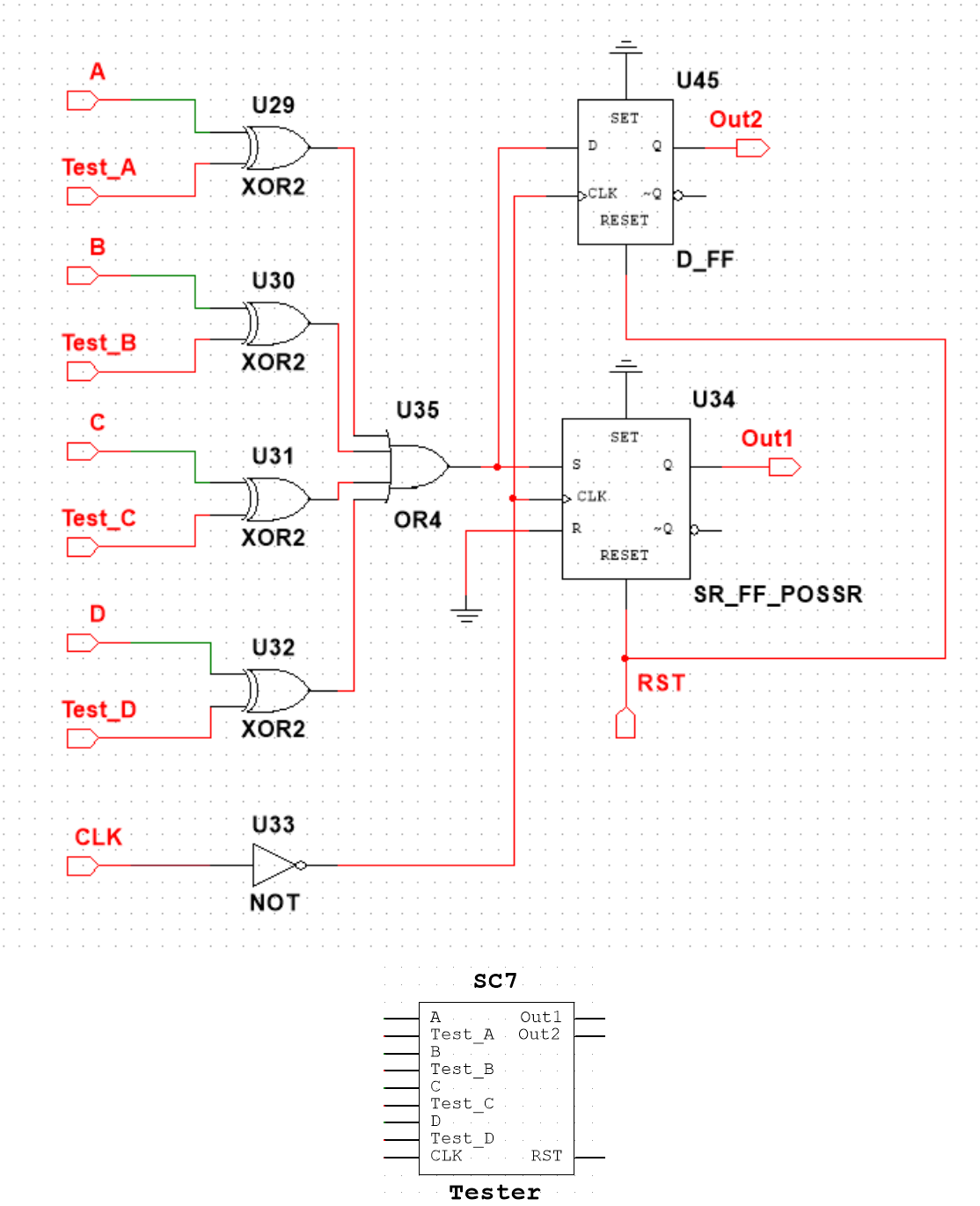
$$A_1 : \overline{ACD}A + ACD$$
$$B_1 : \overline{ACD}B + ACD$$
$$C_1 : \overline{ACD}C$$
$$D_1 : \overline{ACD}D$$
$$A_2 : ACD$$

| **D** | **C** | **B** | **A** | **A0** | **A1** | **A2** | **A3** | **A4** | **A5** | **A6** | **B0** | **B1** | **B2** | **B3** | **B4** | **B5** | **B6** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 1 |
| 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 |

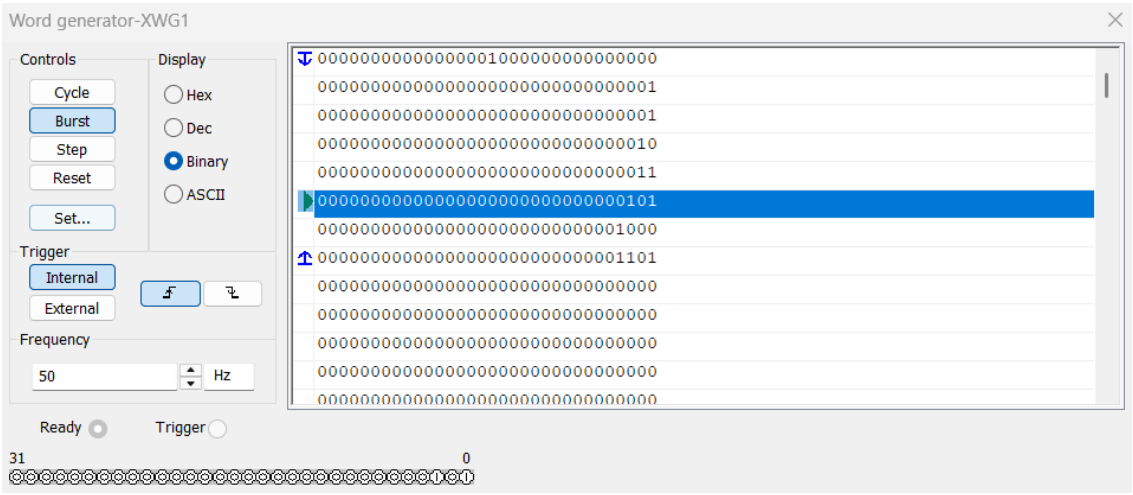
Tabela 4: Tabela prawdy dla całego transkodera

# 6 Układ testujący

Układ testujący analizuje dane wyjściowe pochodzące z naszego układu i porównuje je z przewidywanymi kolejnymi liczbami, które powinny się pojawić w naszym wyświetlaczu 7-segmentowym.




Rysunek 13: Schemat układu testującego




Rysunek 14: Schemat układu testującego

W generatorze słowa są postaci: 0000000000000000

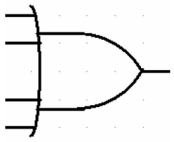
0 – Bit resetujący
00000000000 – Bity nieużywane
0000 – Bity testujące poprawność

Układ testujący najpierw porównuje bit otrzymany z podukładu Fibonacci z bitem przewidywanym przez XWG-1. Wykorzystaliśmy tutaj bramki XOR ($\sim (A \Leftrightarrow B)$).

| A | B | O |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |


Rysunek 15: Tabela prawdy dla bramki XOR

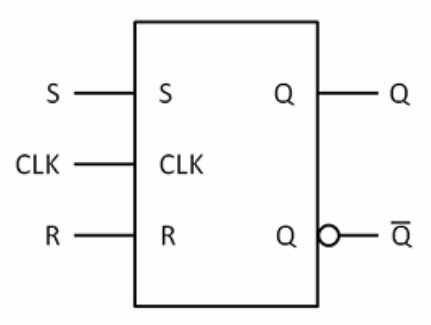
Następnie wyjście z bramek XOR przekazujemy do bramki OR:



| A | B | C | D | O |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 |

Tabela 5: Tabela prawdy i symbol dla bramki 4-wejściowej OR

Sygnał błędu jeśli błąd wystąpił w jakimkolwiek teście rejestrowany jest przez przerzutnik synchroniczny RS (dodatkowe wejście CLK) i zapalana jest dioda X1.

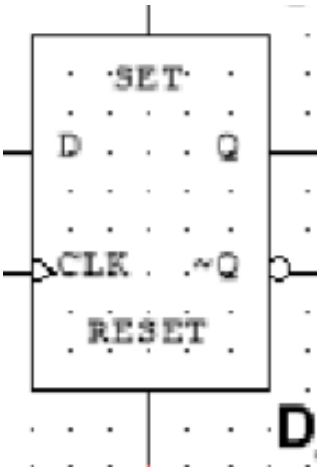



Rysunek 16: Przerzutnik RS

| CLK | S | R | $Q_{n+1}$ | $-Q_{n+1}$ |
|---|---|---|---|---|
| ↑ | 0 | 0 | $Q_n$ | $-Q_n$ |
| ↑ | 0 | 1 | 0 | 1 |
| ↑ | 1 | 0 | 1 | 0 |
| ↑ | 1 | 1 | X | X |
| X | X | X | $Q_n$ | $-Q_n$ |

Tabela 6: Tabela prawdy przerzutnika RS

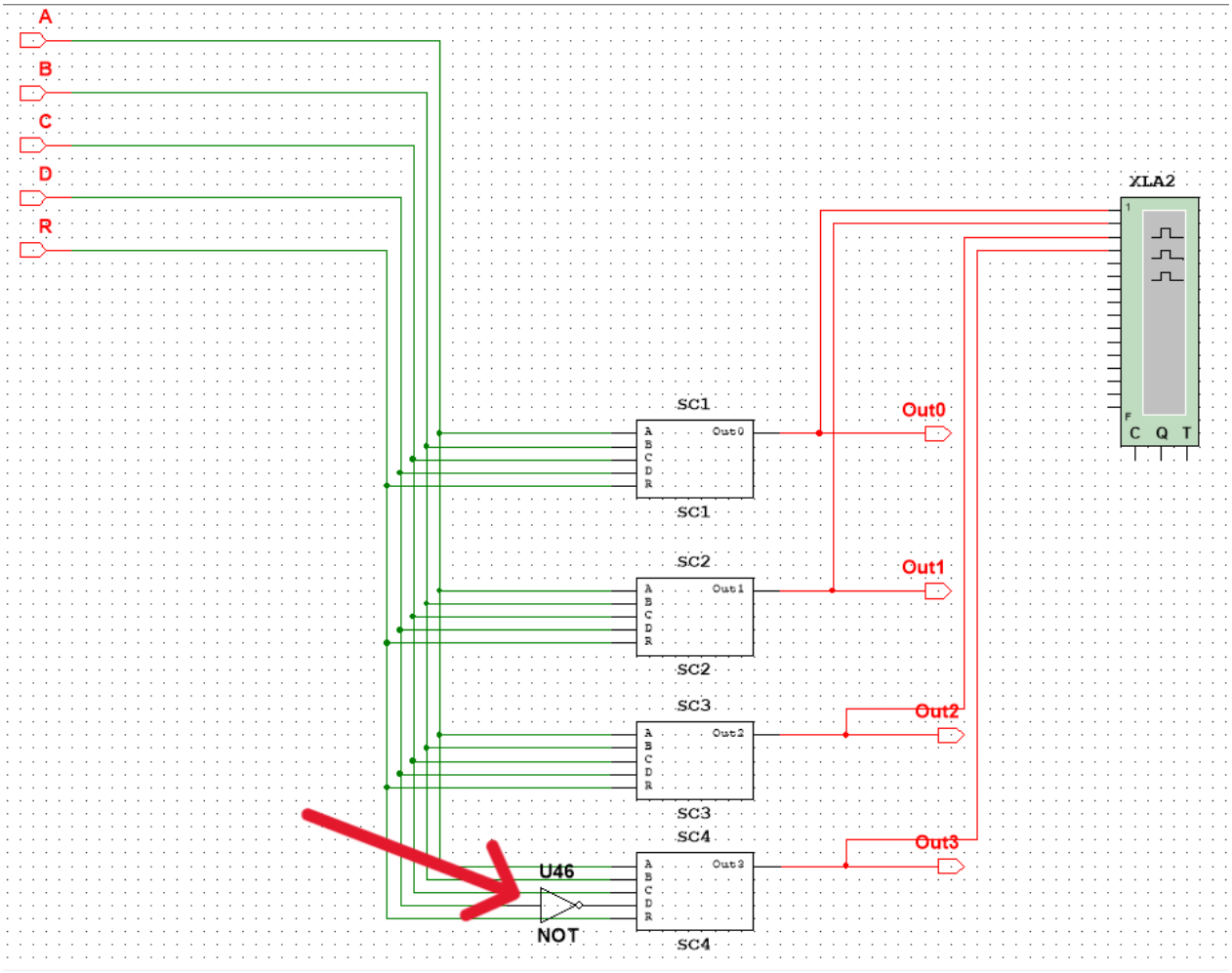
Natomiast sygnał błędu jeśli błąd wystąpił w danym teście rejestrowany jest przez przerzutnik synchroniczny D i zapalana jest dioda X2:



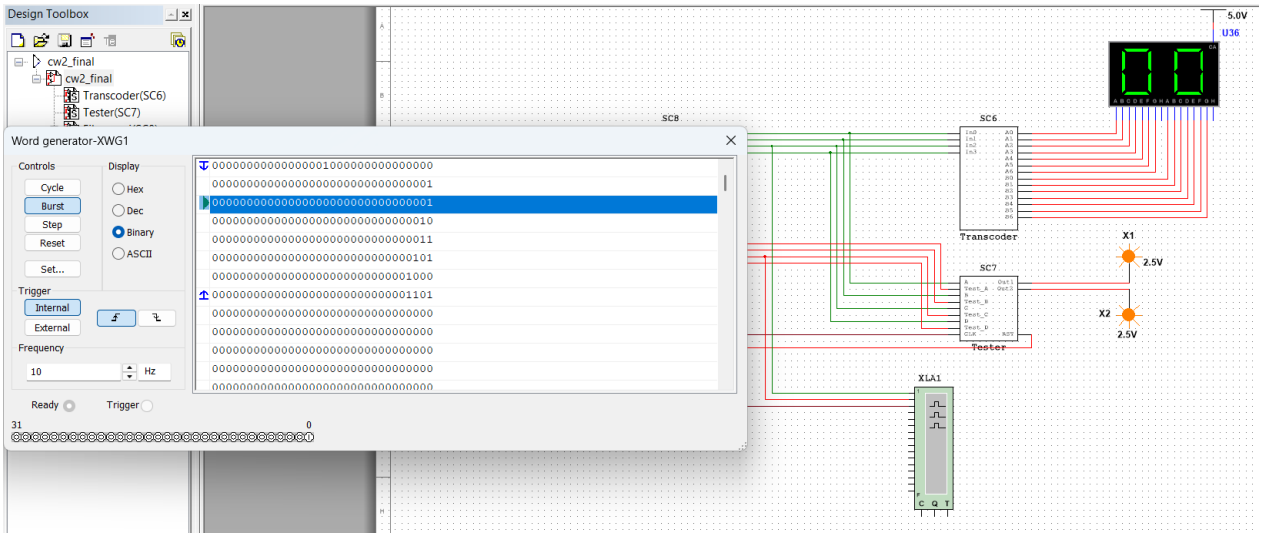
Rysunek 17: Przerzutnik D

| **CLK** | **D** | **SET** | **RESET** | $Q_{n+1}$ |
|---|---|---|---|---|
| ↑ | 0 | 0 | 0 | 0 |
| ↑ | 1 | 0 | 0 | 1 |
| X | X | 0 | 0 | $Q_n$ |
| X | X | 1 | 0 | 1 |
| X | X | 0 | 1 | 0 |
| X | X | 1 | 1 | X |

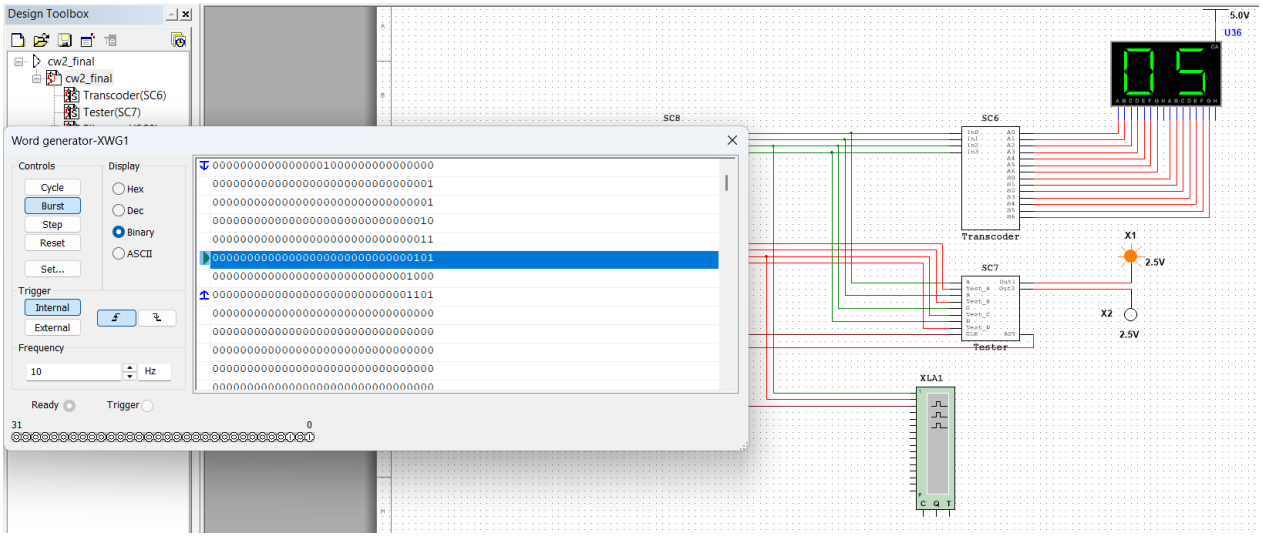
Tabela 7: Tabela prawdy dla przerzutnika D




Rysunek 18: Uszkodzony układ

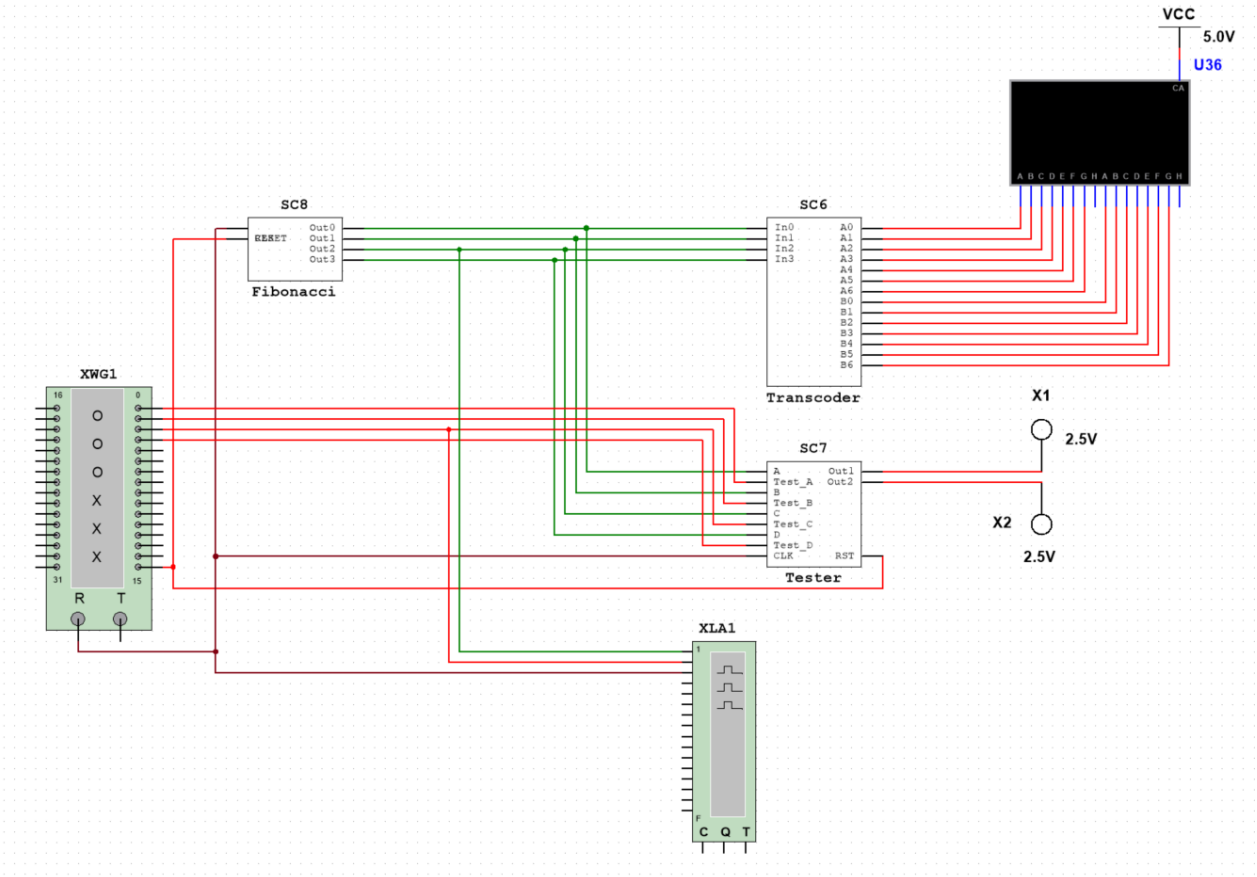



Rysunek 19: Testowanie uszkodzonego układu 1

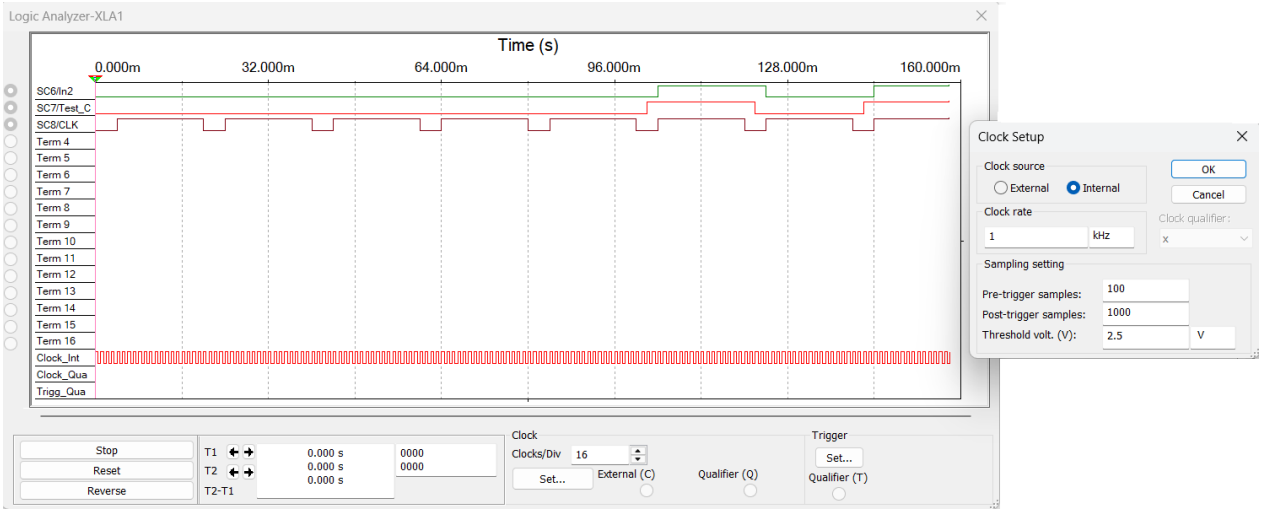



Rysunek 20: Testowanie uszkodzonego układu 1

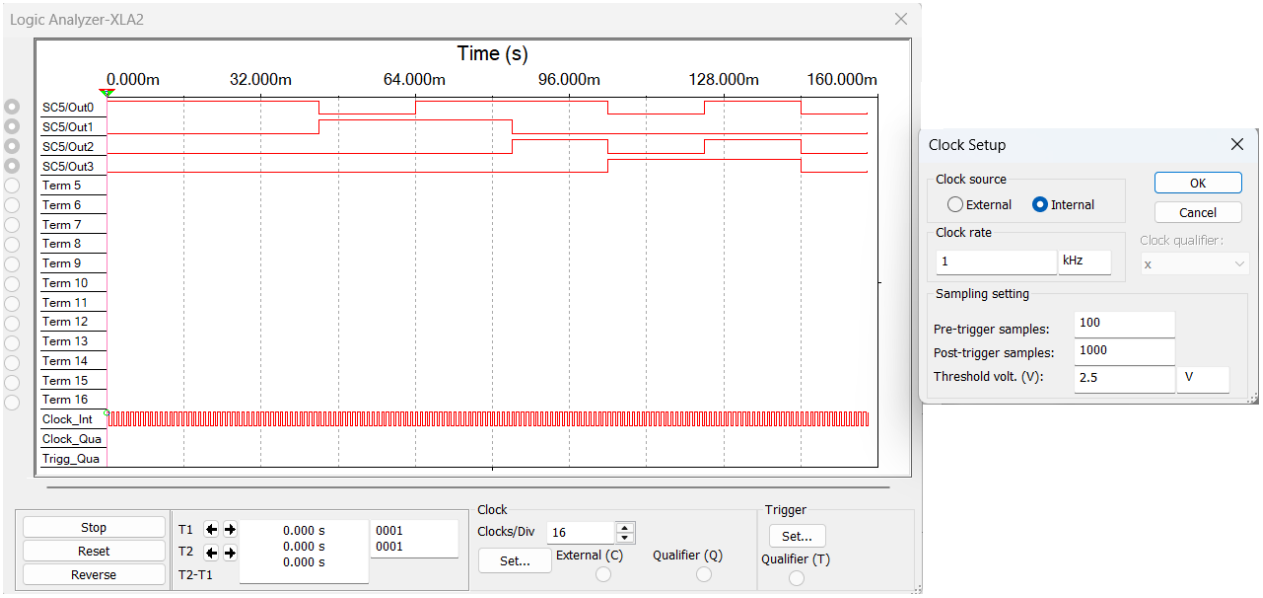
# 7 Schemat całego układu




Rysunek 21: Schemat całego układu




Rysunek 22: Wykres analizatora XLA1




Rysunek 23: Wykres analizatora XLA2 dla nieuszkodzonego podukładu CNT

# 8 Wnioski

W trakcie realizacji projektu wykorzystanie narzędzi do analizy tablic Karnaugh znacząco ułatwiło optymalizację funkcji logicznych. Dzięki temu proces upraszczania wyrażeń logicznych przebiegł sprawnie i pozwolił uzyskać wersje funkcji o minimalnej liczbie składników, co w praktyce przekłada się na mniejsze zużycie zasobów sprzętowych i uproszczenie układu.

Na początkowym etapie zdecydowaliśmy się wprowadzić modyfikację polegającą na uproszczeniu zbioru wejść — poprzez eliminację jednej z jedynek w ciągu — co ograniczyło liczbę wymaganych wejść do czterech. Taki zabieg znacząco uprościł dalsze działania projektowe, w tym analizę i syntezę funkcji sterujących.

W projekcie kluczową rolę odegrały przerzutniki typu D, które zostały wykorzystane do realizacji zegara. Ich zastosowanie uprościło układ sekwencyjny, ponieważ wyeliminowano konieczność szczegółowego rozważania wszystkich możliwych kombinacji stanów wejściowych.

Projekt można łatwo przekształcić do realizacji z użyciem bramek NAND lub NOR, co umożliwia jego implementację na gotowych układach scalonych, takich jak:

CMOS: np. 4011 (NAND) lub 4001 (NOR),

TTL: np. 7400 (NAND) lub 7402 (NOR).

Zaletą takiego podejścia jest możliwość fizycznego odwzorowania układu z użyciem popularnych komponentów, co zwiększa jego dostępność i praktyczność.

Co można było zrobić inaczej? Podczas projektowania rozważaliśmy kilka alternatywnych podejść. Jednym z nich było użycie dwóch niezależnych rejestrów do przechowywania danych oraz realizacja operacji dodawania przy pomocy sumatora. Wynik byłby następnie przesyłany do jednego z rejestrów, zastępując poprzednią wartość. Taka koncepcja dawałaby możliwość łatwej rozbudowy układu i byłaby zgodna z potencjalnym rozszerzeniem projektu. Wadą tego podejścia byłaby jednak jego złożoność oraz większy koszt wykonania, zarówno pod względem sprzętowym, jak i projektowym.

Inną rozważaną opcją było stworzenie automatu sekwencyjnego, który cyklicznie przechodziłby przez wartości od 0 do 7. Każdej liczbie można przypisać inną wartość wyjściową (np. ciąg Fibonacciego: 0, 1, 1, 2, 3, 5, 8, 13), realizując tym samym pewne dodatkowe założenia projektu. Taki automat można by było łatwo zaimplementować z użyciem enkodera, a dzięki swojej deterministycznej naturze mógłby być zarówno tani w produkcji, jak i stosunkowo prosty do zaprogramowania.