Sprawozdanie
bramki i funkcje logiczne

Technika Cyfrowa 2021/22

Dominik Grzesik, Marcin Mikuła, Sebastian Kozak
Szymon Słota, Jakub Łubkowski

mmikula@student.agh.edu.pl szymonslota@student.agh.edu.pl lubkowski@student.agh.edu.pl
domgrze@student.agh.edu.pl sebkozak@student.agh.edu.pl

Spis treści

Bramki i funkcje logiczne

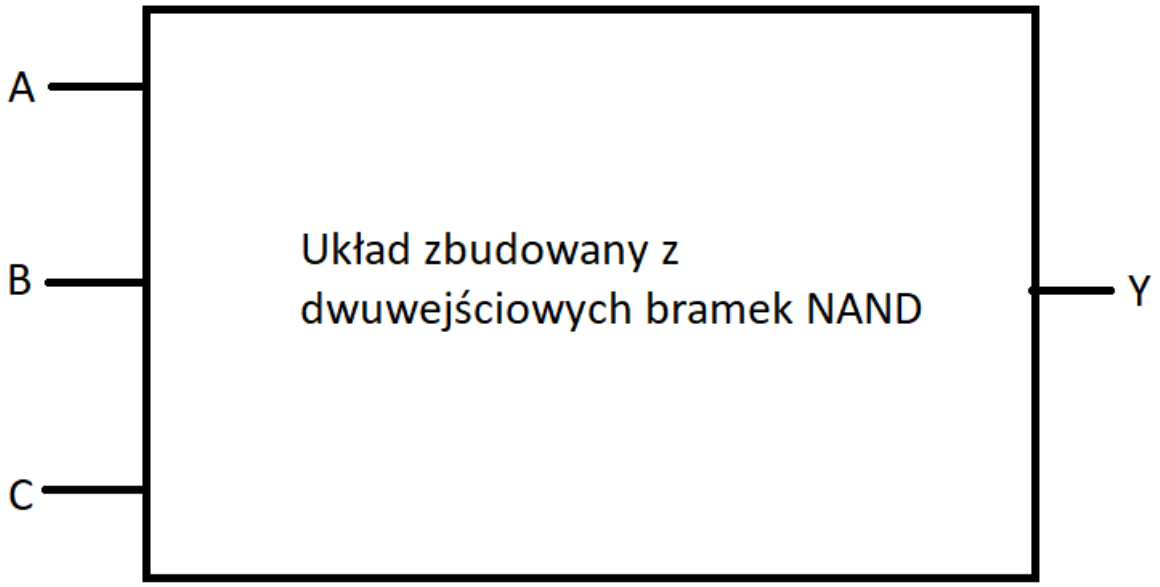
1. Zadanie 1

Za pomocą bramek NAND zrealizować funkcję logiczną

$$Y = \bar{A}C + B(A + B)$$

1.1. Wprowadzenie

Układ będzie przyjmował 3 zmienne na wejściu, oraz zwracał 1 na wyjściu.



Rysunek 1. Układ poglądowy.

Właściwości bramki NAND:

A	B	$\overline{AB}$
0	0	1
0	1	1
1	0	1
1	1	0

Tabela 1. Tablica bramki NAND.

.

1.2. Wyprowadzenie

$$Y = \overline{A}C + B(A + B)$$

Wiemy, że:

$$(1)\ BB = B$$

$$(2)\ A = AA$$

Z prawa rozdzielności koniunkcji względem alternatywy:

$$Y = \overline{A}C + BA + BB$$

Z (1):

$$Y = \overline{A}C + BA + B$$

Z prawa rozdzielności alternatywy względem koniunkcji:

$$Y = \overline{A}C + B(A + 1)$$

$$A + 1 = 1$$

$$Y = \overline{A}C + B$$

Z prawa podwójnego zaprzeczenia:

$$Y = \overline{\overline{\overline{A}C + B}}$$

Z pierwszego prawa de Morgana:

$$Y = \overline{\overline{\overline{A}C}\,\overline{B}}$$

Z (2):

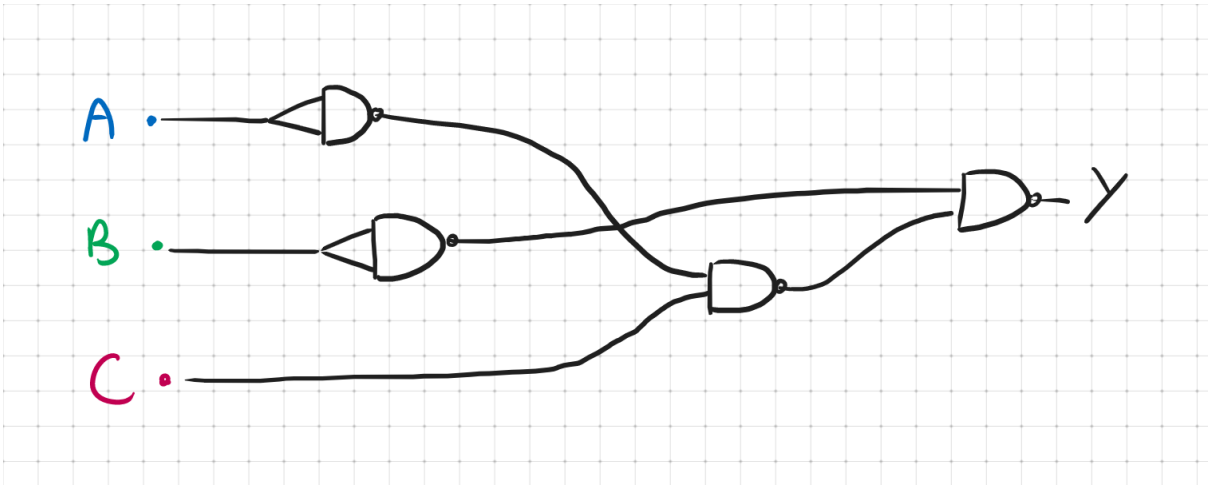
$$Y = \overline{\overline{\overline{A}\overline{A}C}\overline{B}}$$

Z definicji nand:

$$\overline{XY} = X \text{nand} Y$$

W związku z tym:

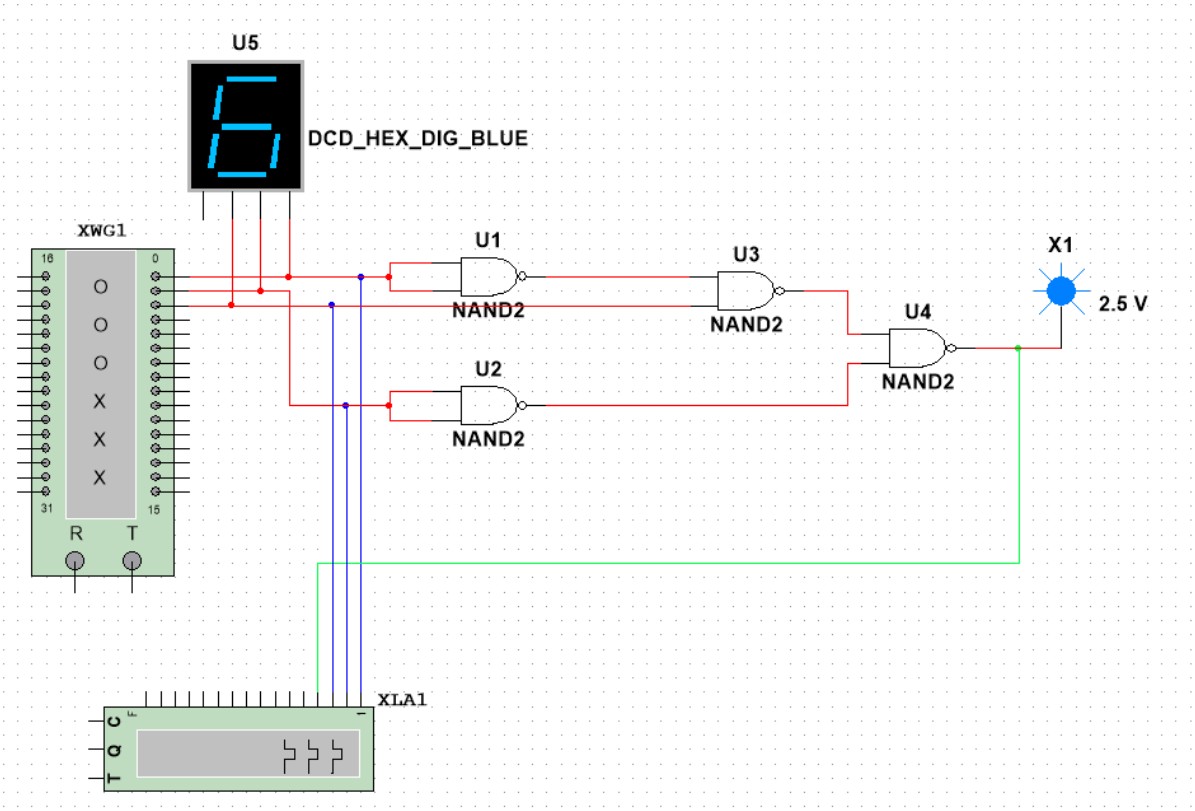
$$Y = ((A \text{nand} A) \text{nand} C) \text{nand} (B \text{nand} B)$$



Rysunek 2. Poglądowy rysunek układu.

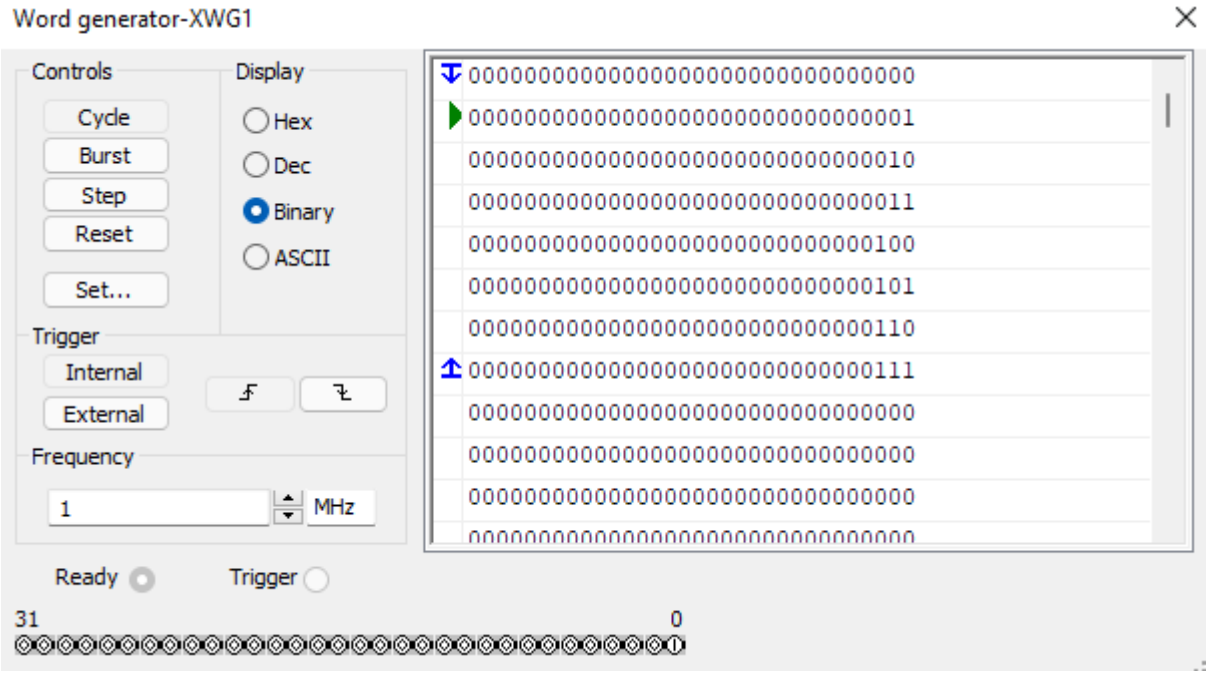
1.3. Implementacja układu z bramkami NAND w programie Multisim

Układ przedstawiony wyżej został podpięty do generatora słów logicznych, oraz wyjście Y do diody. Następnie sprawdzono czy wyniki otrzymane zgadzają się z tymi obliczonymi na kartce.



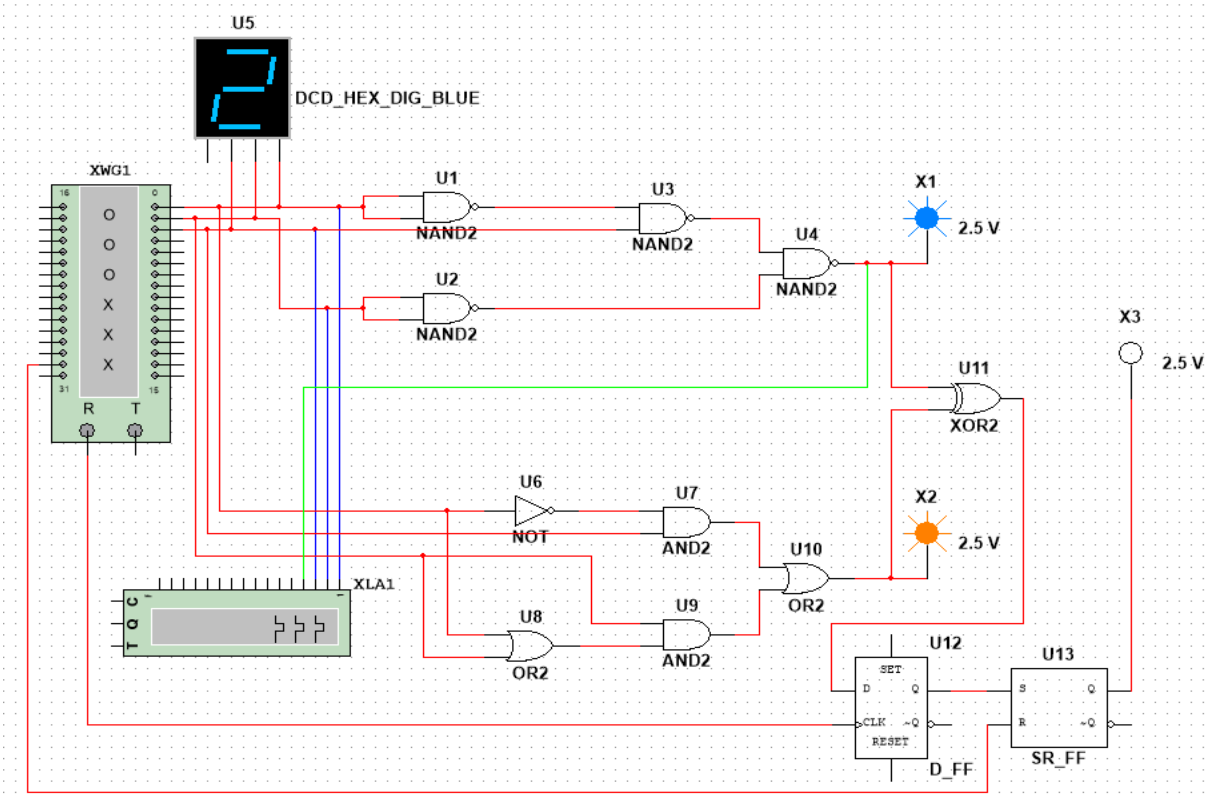
Rysunek 3. Układ w Multisimie

1.4. Generator słów



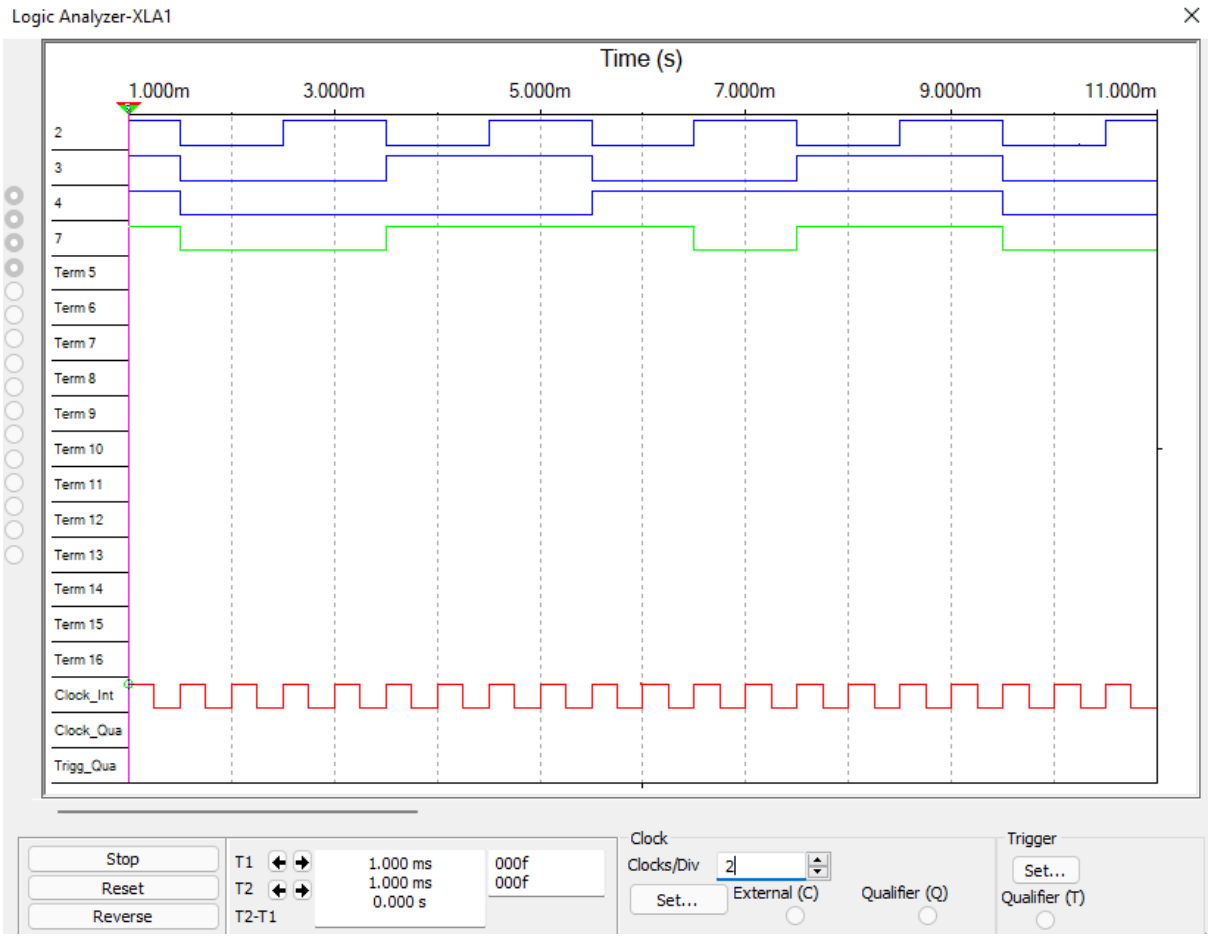
Rysunek 4. Generator słów

1.5. Implementacja układu testującego w programie Multisim



Rysunek 5. Układ testujący w Multisimie

1.6. Analizator słów



Rysunek 6. Analizator słów

1.7. Implementacja w języku Python

Za pomocą krótkiego programu wyliczamy wartości funkcji logicznej Y

```
for a in range(2):
   if a==1: aa=True
   else: aa=False
   for b in range(2):
       if b==1: bb=True
       else: bb=False
       for c in range(2):
           if c==1: cc=True
           else: cc=False

           if ((not aa) and cc) or (bb and (aa or bb)): y=1
           else: y=0

           print("A:", a, "| B:", b, "| C:", c, "| Y:", y)
```

Kod 1. Program wyliczający wartości równania zadania

```
A: 0 | B: 0 | C: 0 | Y: 0
A: 0 | B: 0 | C: 1 | Y: 1
A: 0 | B: 1 | C: 0 | Y: 1
A: 0 | B: 1 | C: 1 | Y: 1
A: 1 | B: 0 | C: 0 | Y: 0
A: 1 | B: 0 | C: 1 | Y: 0
A: 1 | B: 1 | C: 0 | Y: 1
A: 1 | B: 1 | C: 1 | Y: 1
```

Tabela 2. Wartości funkcji Y

1.8. Wnioski

- Przez przekształcenia równoważne danej funkcji logicznej zmniejszyliśmy liczbę używanych bramek logicznych z 5 do 4 w układzie zbudowanym jedynie z bramek NAND.

- Możliwe jest testowanie poprawności szybszego układu poprzez projektowanie wersji przed i po przekształceniu.

1.9. Zastosowanie układu w praktyce.

Układ może znaleźć zastosowanie w zespołach rajdowych jako system pomagający w podjęciu decyzji w sytuacji, gdy ciężko stwierdzić, czy kierowca powinien kontynuować jazdę.

$$Y = \bar{A}C + B(A + B)$$

A - czy zabraknie mu paliwa
B - czy wyścig dalej trwa
C - czy nie skończył jeszcze okrążenia

Jeśli kierowcy **nie zabraknie paliwa** (Ā) i **nie skończył on jeszcze okrążenia** (C) lub **wyścig wciąż trwa** (B), a do tego **zabraknie mu paliwa** (A) lub **wyścig dalej trwa** (B) to oznacza, że **kierowca powinien kontynuować jazdę** (Y).

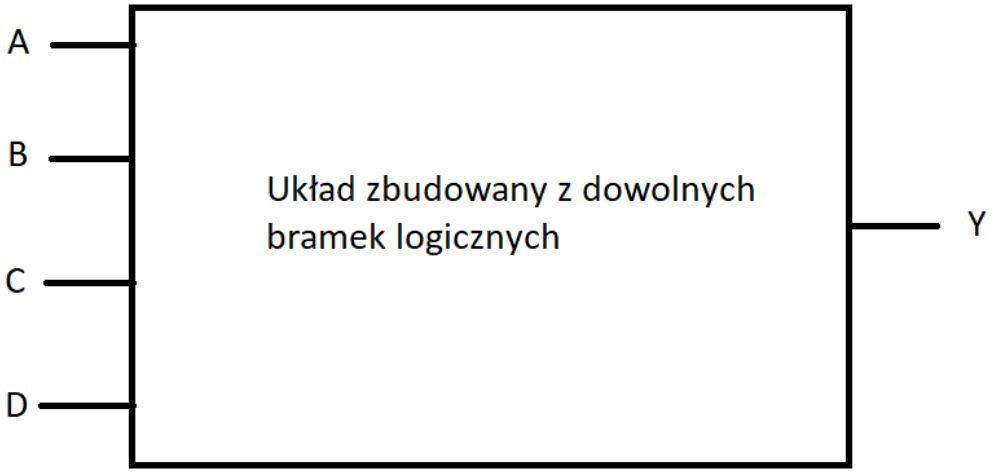
2. Zadanie 2

Bazując na dowolnych bramkach logicznych, proszę od podstaw zaprojektować, zbudować i przetestować układ detekcji liczby pierwszej w binarnym słowie czterobitowym

2.1. Wprowadzenie

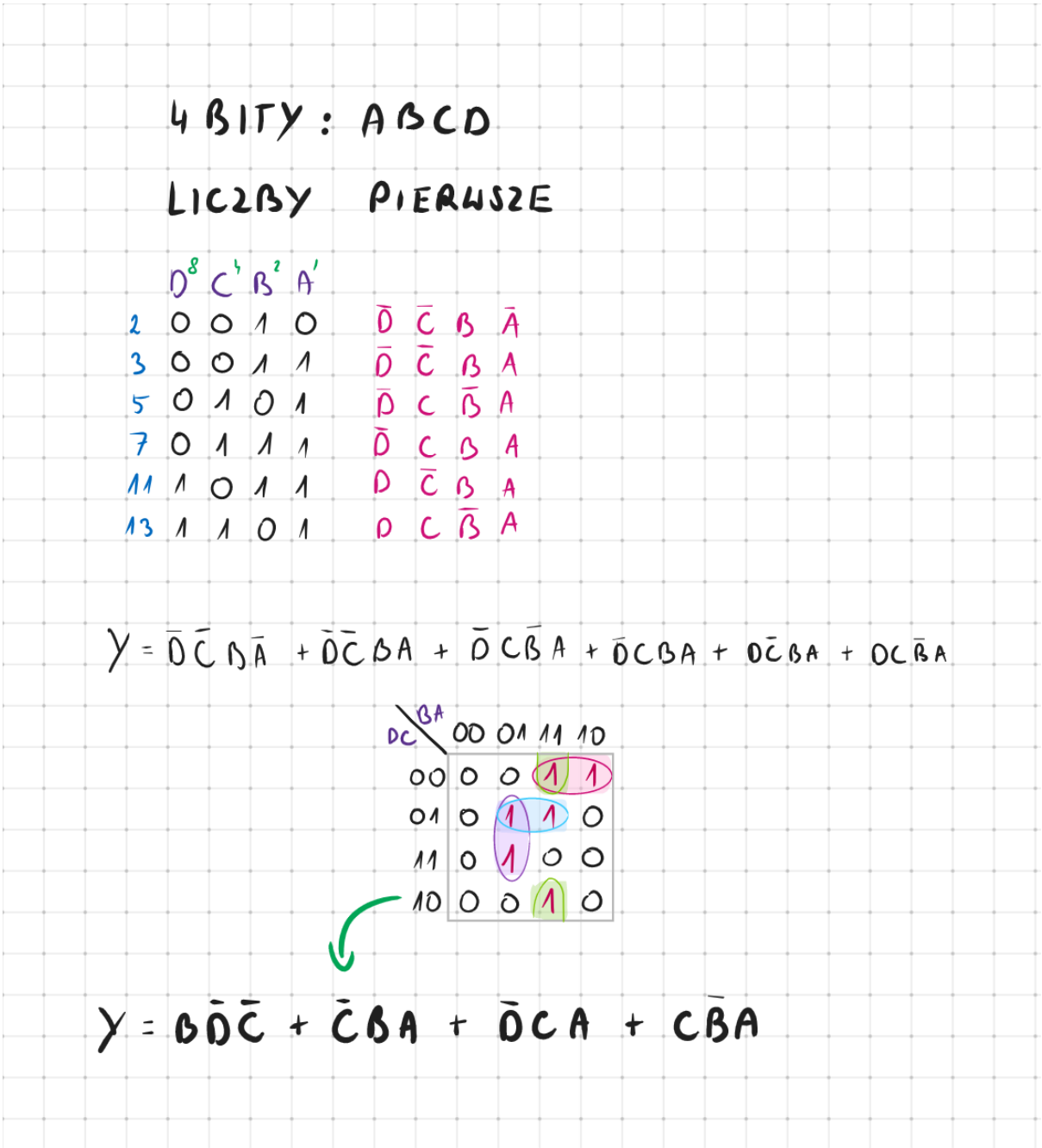
Należy zwrócić uwagę, że na 4 bitach zmieszczą się liczby od 0 do 15. Wśród nich liczbami pierwszymi są:

2, 3, 5, 7, 11, 13

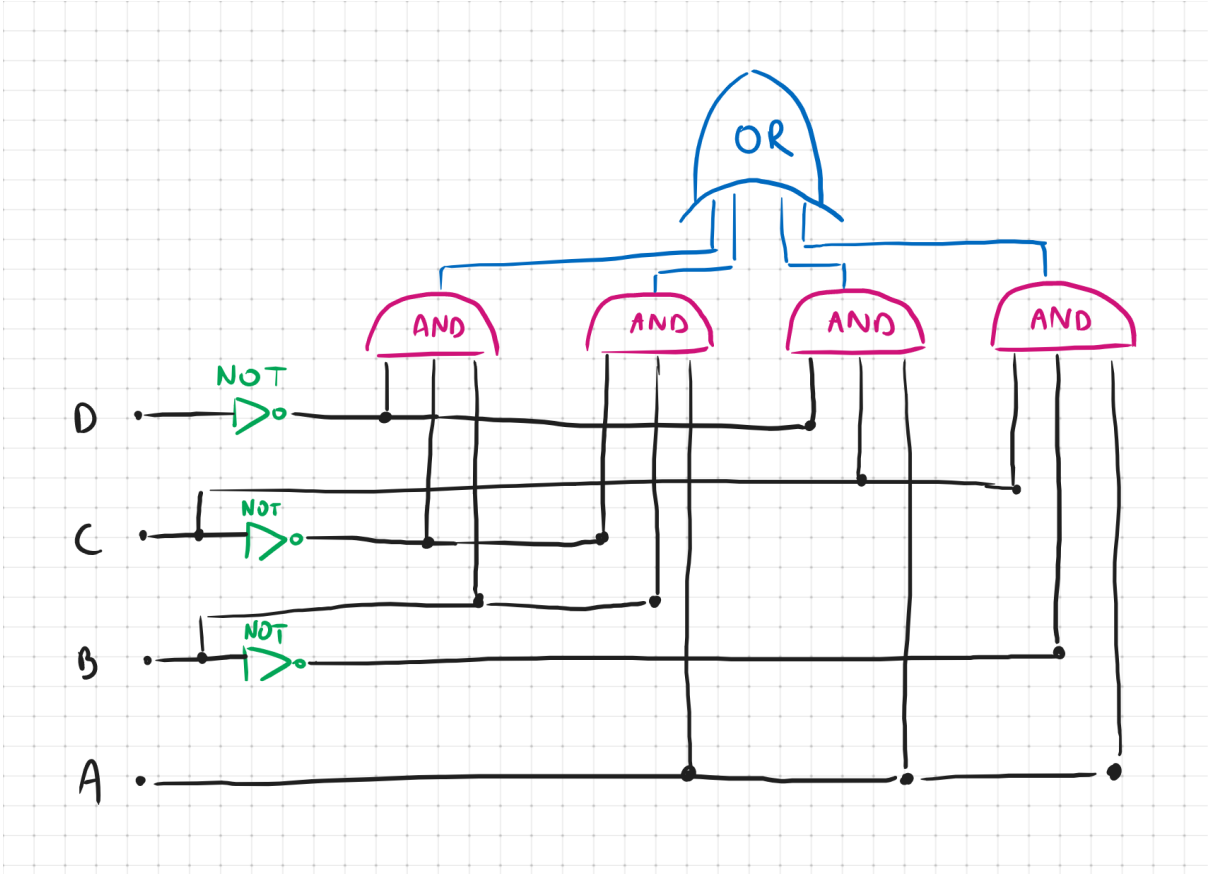


Rysunek 7. Ogólna idea planowanego rozwiązania

2.2. Wyprowadzenie

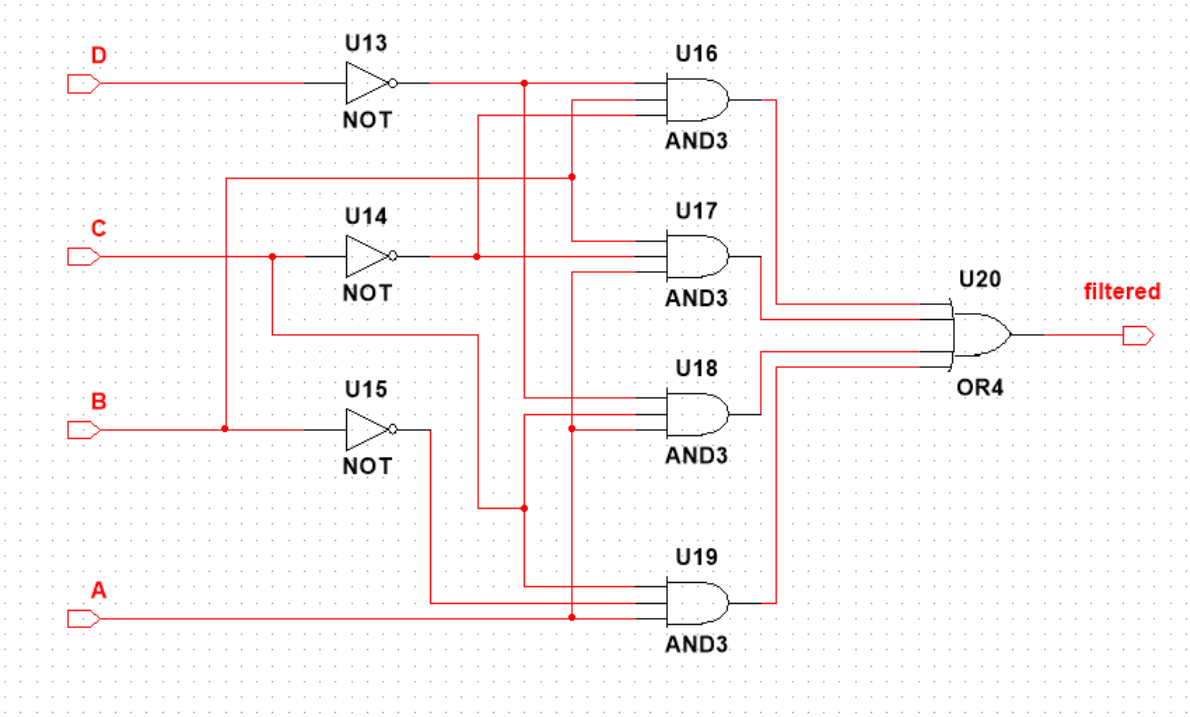


Rysunek 8. Obliczanie funkcji wejściowej Y w zależności od parametrów A B C D



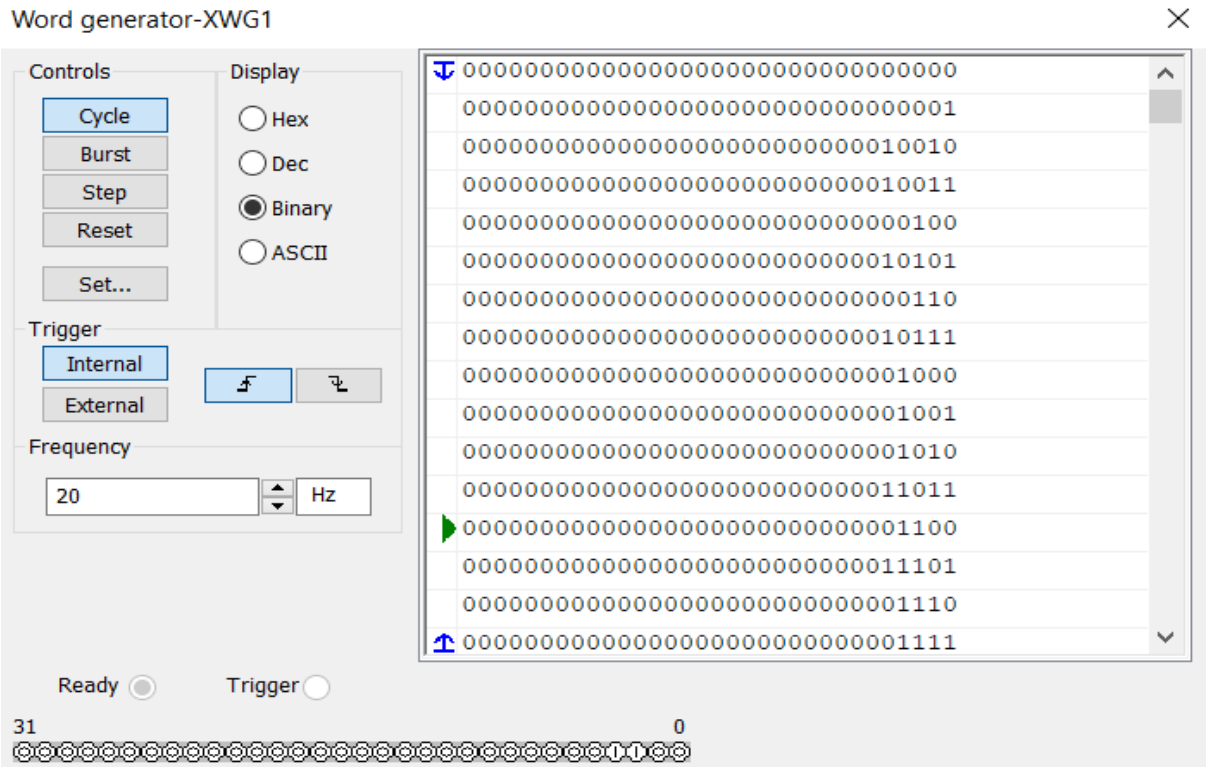
Rysunek 9. Poglądowy rysunek układu

2.3. Implementacja układu z bramkami NAND w programie Multisim



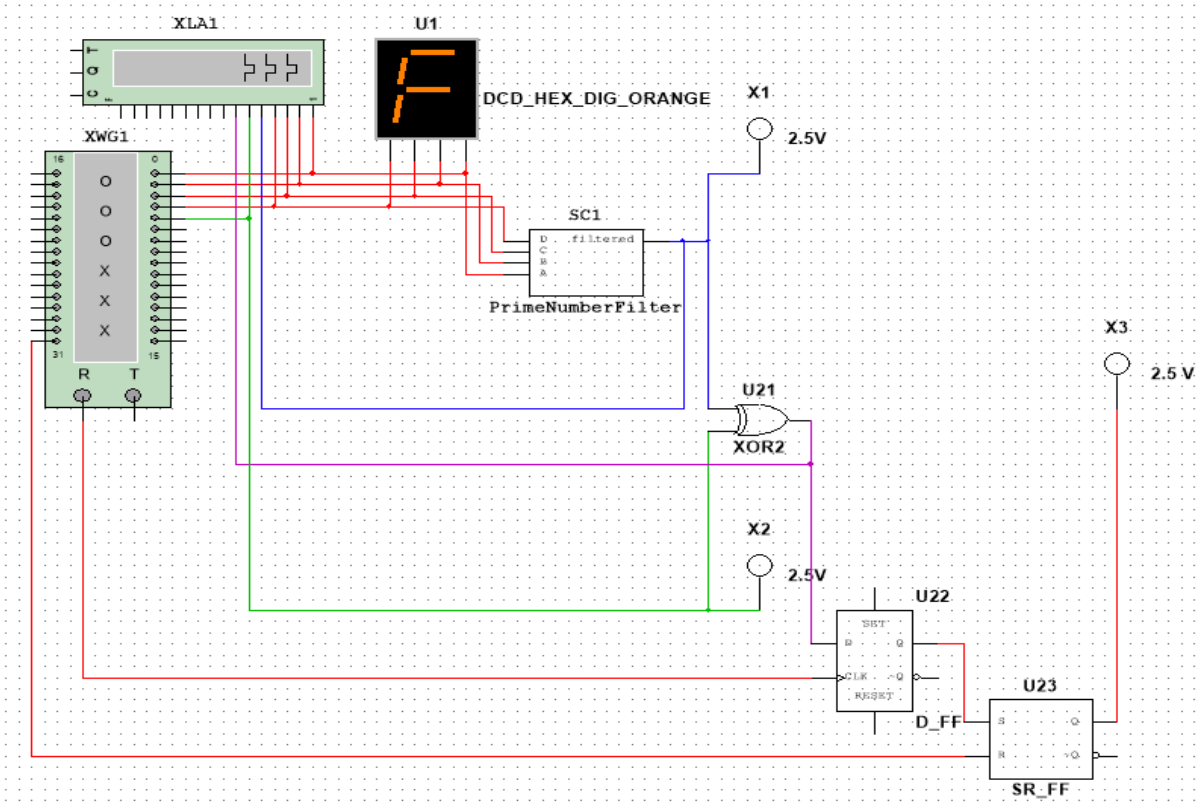
Rysunek 10. Zaprojektowany układ w Multisimie

2.4. Generator słów



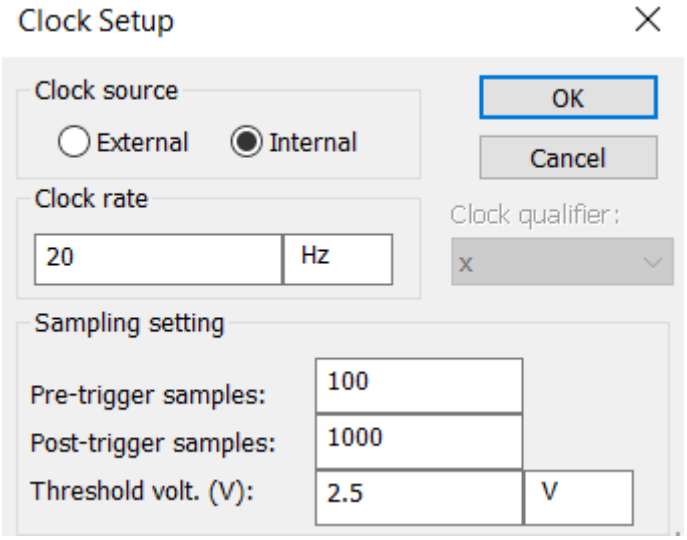
Rysunek 12. Generator słów

2.5. Implementacja układu testującego w programie Multisim

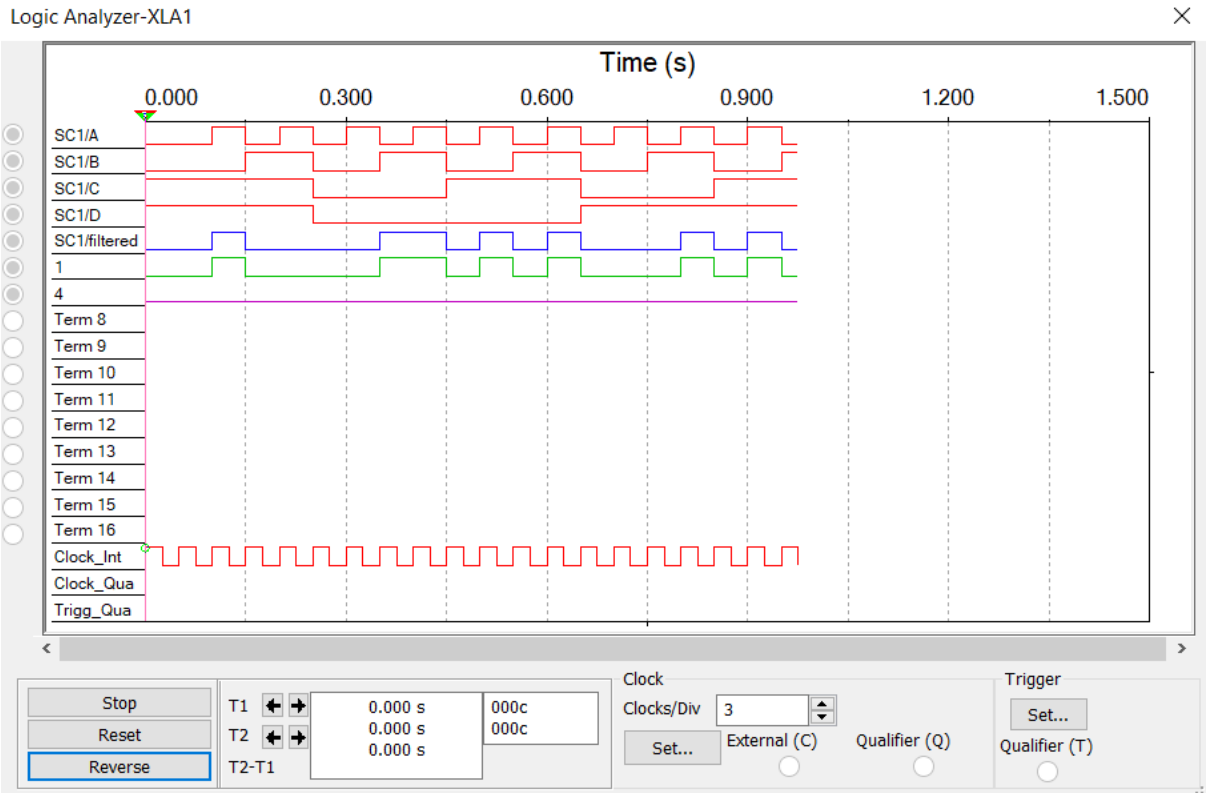


Rysunek 11. Zaprojektowany układ testowy w Multisimie

2.6. Analizator słów



Rysunek 12. Analizator słów



Rysunek 13. Analizator słów

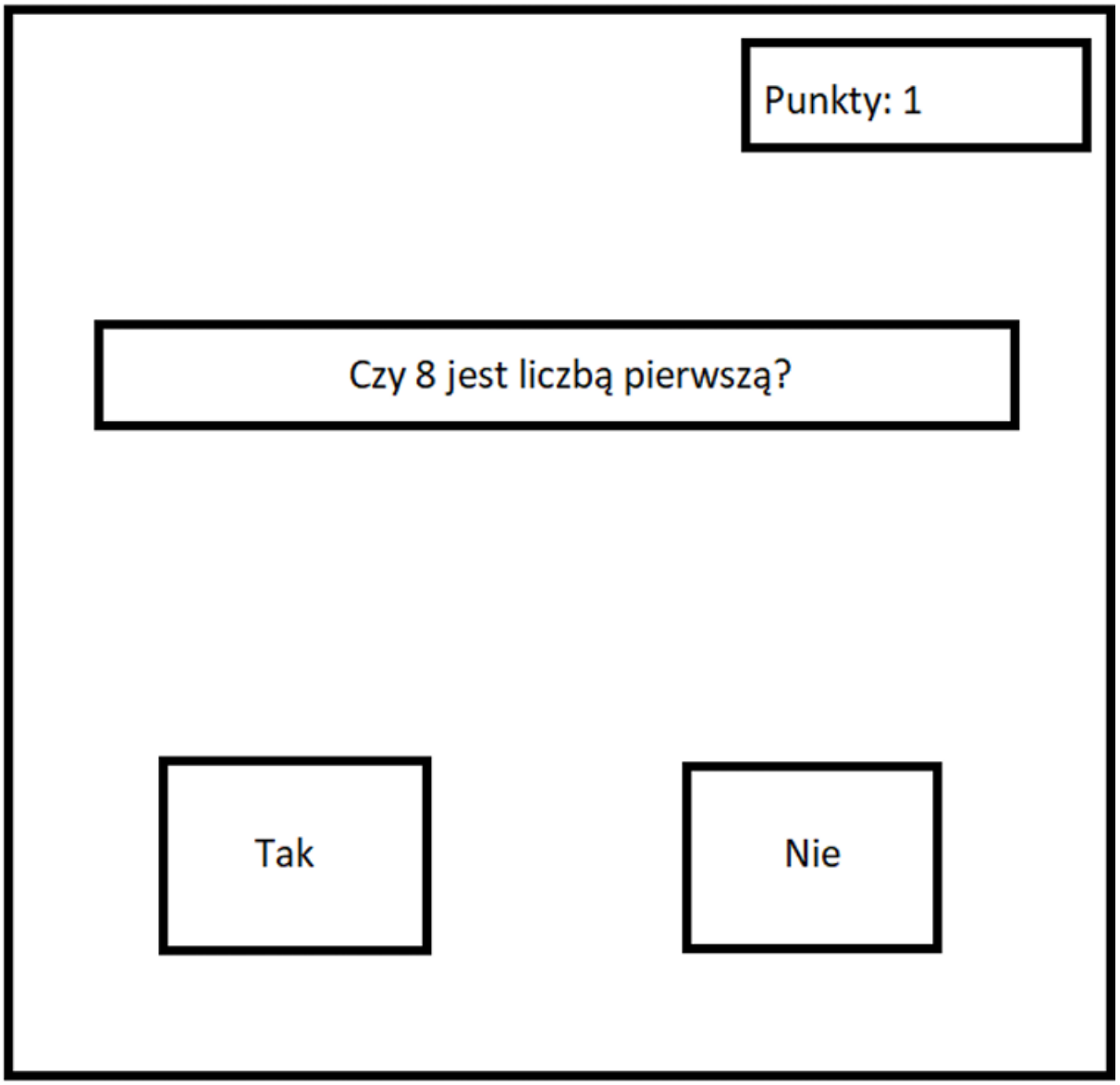
2.7. Wnioski

- Przy użyciu tablic Karnaugh uprościliśmy reprezentację 6 pierwszych liczb pierwszych przedstawionych jako 4-bitowe słowa, gdzie każdy bit jest podany jako zmienna logiczna.

- Widać na analizatorze, że bramka XOR(linia fioletowa) jest ciągle w stanie niskim,gdyż 5 bit(linia zielona), wykorzystany do wyróżnienia liczb pierwszych oraz wyniki filtrowania(linia niebieska) są w tym samym stanie w tych samych odstępach czasowych. W związku z tym można stwierdzić, że układ działa poprawnie.

2.8. Zastosowanie układu w praktyce.

Ciekawym zastosowaniem jest **gra edukacyjna** dla małych dzieci, która sprawdza znajomość liczb pierwszych(np.: czy 8 jest liczbą **pierwszą** lub czy 7 jest liczbą **złożoną** itp.). W momencie **generowania** liczby przechodzi ona przez utworzony **układ** i zostanie przypisana odpowiednio do zbioru liczb **złożonych** lub **pierwszych**. Następnie dziecko dokonuje wyboru i odpowiednio gra odpowiada czy wybrana odpowiedź była poprawna lub nie.



Rysunek 14. Poglądowa wizualizacja gry edukacyjnej