Technika Cyfrowa

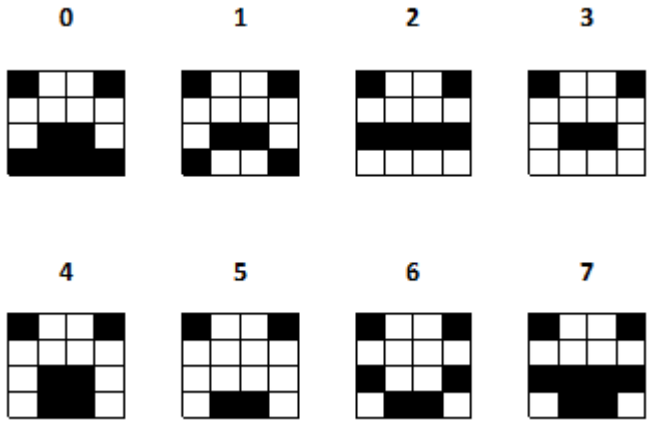
Sprawozdanie z ćwiczenia nr 1

Natalia Curzytek, Marta Stanisławska,
Karolina Nitsch, Szymon Kłodowski

Treść zadania

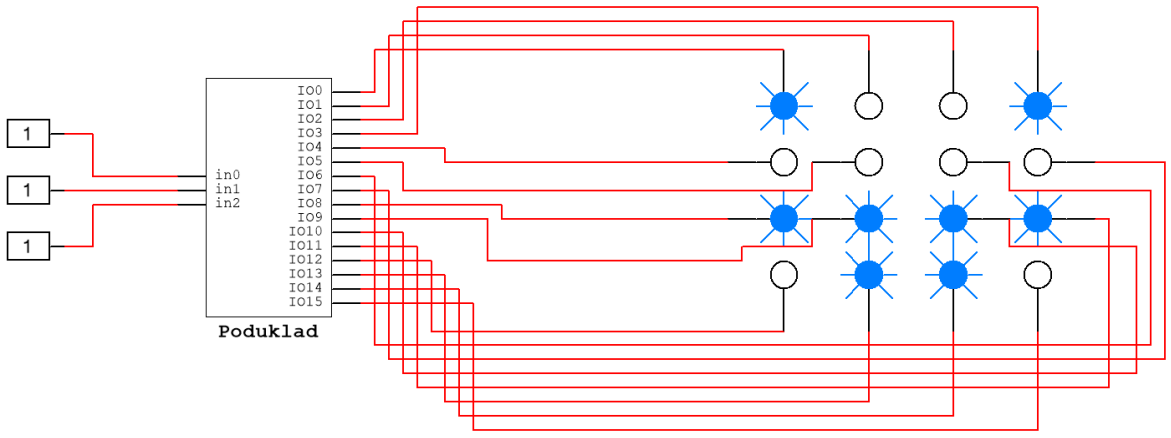
Bazując wyłącznie na bramkach NAND, proszę zaprojektować układ kombinacyjny realizujący transkoder trzybitowej liczby binarnej na układ graficzny emotikony wyświetlanej na szesnastu punktach, zgodnie z poniższym rysunkiem 1:

Rys. 1. Emotikony



Zaprojektowany schemat proszę umieścić w podukładzie (Subcircuit). Wyświetlacz można zrobić z próbników dostępnych w programie Multisim. Przykładowo może to wyglądać tak jak na poniższym schemacie 2:

Rys. 2. Przykładowy projekt układu



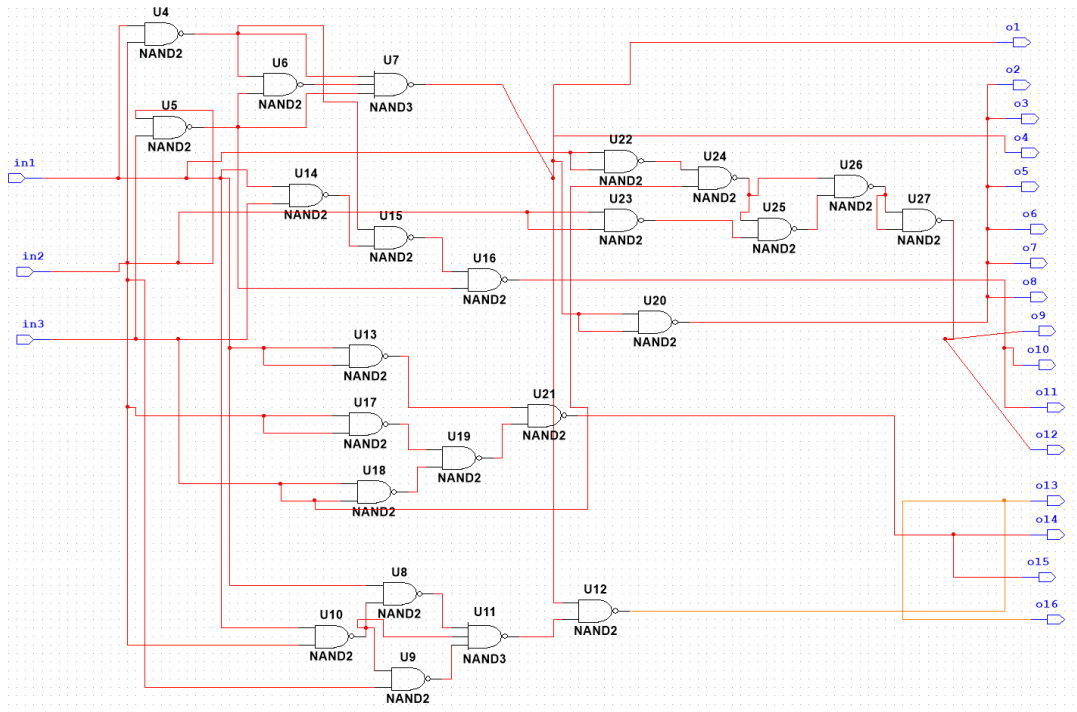
Idea rozwiązania

Rozwiązanie opiera się na analizie wzorca graficznego dla każdej z możliwych wartości wejściowych (od 000 do 111), co odpowiada ośmiu różnym układom punktów na siatce 4x4. Każdy z 16 punktów świetlnych można traktować jako osobną funkcję logiczną, zależną od trzech wejść binarnych.

Aby uprościć proces projektowania, dla każdego punktu świetlnego tworzone są tabele Karnaugh, które umożliwiają minimalizację funkcji logicznych. Ponieważ dozwolone są jedynie bramki NAND, zminimalizowane wyrażenia logiczne muszą zostać przekształcone do postaci wykorzystującej wyłącznie tę bramkę. Po uzyskaniu funkcji logicznych dla każdego punktu, implementowane są one w postaci układu logicznego w programie Multisim.

Trywialne podejście

Pierwotnie rozpoczęliśmy rozwiązywanie zadania od manualnego “ułożenia” bramek logicznych metodą prób i błędów tak, aby określone diody świeciły się przy odpowiednich sygnałach bitowych. Pozwoliło nam to porównać obie metody działania i pokazało, jak bardzo skuteczna jest metoda używająca tablic Karnaugh. Można łatwo zauważyć, że poniższe bramki są nieuporządkowane i zbytecznie rozbudowane. Tym sposobem też bardzo trudno jest określić czy układ jest zminimalizowany. Poza technicznymi różnicami warto też wspomnieć o oczywistej czasochłonności tego podejścia.



Rys. 3. Pierwsze podejście do problemu

Tabela prawdy

Tabela 1. Reprezentacja binarna wejść i wyjść podukładu

Wejścia			Wyjścia															
A	B	C	P11	P12	P13	P14	P21	P22	P23	P24	P31	P32	P33	P34	P41	P42	P43	P44
0	0	0	1	0	0	1	0	0	0	0	0	1	1	0	1	1	1	1
0	0	1	1	0	0	1	0	0	0	0	0	1	1	0	1	0	0	1
0	1	0	1	0	0	1	0	0	0	0	1	1	1	1	0	0	0	0
1	0	0	1	0	0	1	0	0	0	0	0	1	1	0	0	0	0	0
0	1	1	1	0	0	1	0	0	0	0	0	1	1	0	0	1	1	0
1	0	1	1	0	0	1	0	0	0	0	0	0	0	0	0	1	1	0
1	1	0	1	0	0	1	0	0	0	0	1	0	0	1	0	1	1	0
1	1	1	1	0	0	1	0	0	0	0	1	1	1	1	0	1	1	0

Tabele Karnaugh i wyprowadzenie wzorów dla sygnałów wyjściowych

W wyprowadzeniach skorzystano z następujących praw logiki:

- prawo podwójnej negacji: $\overline{\overline{A}} \Leftrightarrow A$,
- II prawo de Morgana: $\overline{A_1 \vee A2 \vee \ldots \vee A_n} \Leftrightarrow \overline{A_1} \wedge \overline{A_2} \wedge \ldots \wedge \overline{A_n}$,

oraz zależności $\overline{A} \Leftrightarrow \overline{AA}$.

Tabela 2. Tabela Karnaugh dla wyjść P11 i P14

	AB = 00	AB = 01	AB = 11	AB = 10
C = 0	1	1	1	1
C = 1	1	1	1	1

Wzór z tabeli Karnaugh:

$$P11 = P14 = 1$$

Rys. 4. Schemat podukładu dla wyjść P11 i P14

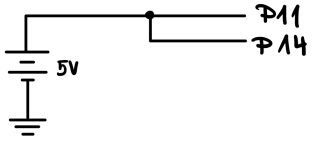


Tabela 3. Tabela Karnaugh dla wyjść P31 i P34

	AB = 00	AB = 01	AB = 11	AB = 10
C = 0	0	1	1	0
C = 1	0	0	1	0

Wzór z tabeli Karnaugh:

$$P31 = P34 = B\overline{C} + AB$$

Wzór na bramkach NAND:

$$P31 = P34 = B\overline{C} + AB = \overline{\overline{B\overline{C}} + \overline{AB}} = \overline{\overline{B\overline{C}} \cdot \overline{AB}} = \overline{\overline{B} \cdot \overline{C\overline{C}} \cdot \overline{AB}}$$

Rys. 5. Schemat podukładu dla wyjść P31 i P34

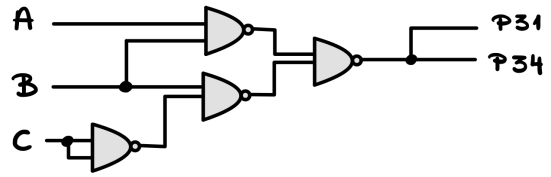


Tabela 4. Tabela Karnaugh dla wyjść P32 i P33

	AB = 00	AB = 01	AB = 11	AB = 10
C = 0	1	1	0	1
C = 1	1	1	1	0

Wzór z tabeli Karnaugh:

$$P32 = P33 = \overline{A} + BC + \overline{B}\,\overline{C}$$

Wzór na bramkach NAND:

$$P32 = P33 = \overline{A} + BC + \overline{B}\,\overline{C} = \overline{\overline{\overline{A} + BC + \overline{B}\,\overline{C}}} = \overline{A \cdot \overline{BC} \cdot \overline{\overline{B}\,\overline{C}}} = \overline{A \cdot \overline{BC} \cdot \overline{B\overline{B}} \cdot \overline{C\overline{C}}}$$

Rys. 6. Schemat podukładu dla wyjść P32 i P33

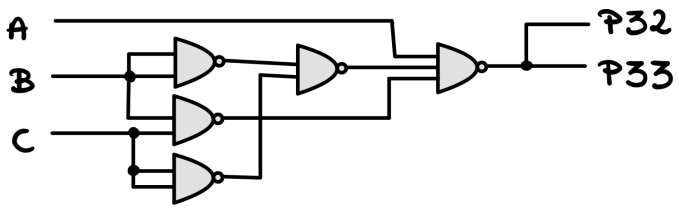


Tabela 5. Tabela Karnaugh dla wyjść P41 i P44

	AB = 00	AB = 01	AB = 11	AB = 10
C = 0	1	0	0	0
C = 1	1	0	0	0

Wzór z tabeli Karnaugh:

$$P41 = P44 = \overline{A}\,\overline{B}$$

Wzór na bramkach NAND:

$$P41 = P44 = \overline{A}\,\overline{B} = \overline{A\,A} \cdot \overline{B\,B} = \overline{\overline{A\,A} \cdot \overline{B\,B}} = \overline{\overline{A\,A} \cdot \overline{B\,B} \cdot \overline{A\,A} \cdot \overline{B\,B}}$$

Rys. 7. Schemat podukładu dla wyjść P41 i P44

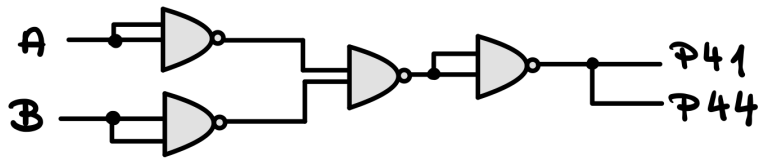


Tabela 6. Tabela Karnaugh dla wyjść P42 i P43

	AB = 00	AB = 01	AB = 11	AB = 10
C = 0	1	0	1	1
C = 1	0	0	1	1

Wzór z tabeli Karnaugh:

$$P42 = P43 = A + \overline{B}\,\overline{C}$$

Wzór na bramkach NAND:

$$P42 = P43 = A + \overline{B}\,\overline{C} = \overline{\overline{A + \overline{B}\,\overline{C}}} = \overline{\overline{A} \cdot \overline{\overline{B}\,\overline{C}}} = \overline{\overline{A\,A} \cdot \overline{\overline{B\,B} \cdot \overline{C\,C}}}$$

Rys. 8. Schemat podukładu dla wyjść P42 i P43

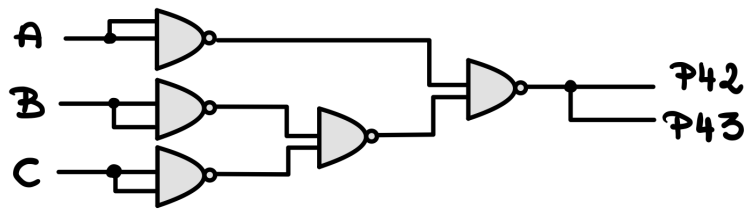


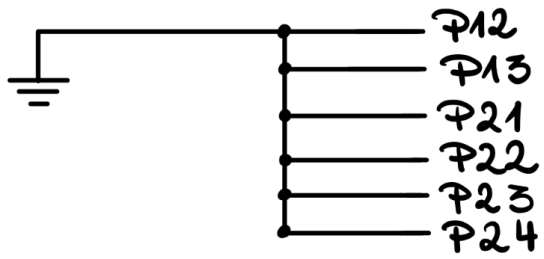
Tabela 7. Tabela Karnaugh dla wyjść P12, P13, P21, P22, P23 i P24

	AB = 00	AB = 01	AB = 11	AB = 10
C = 0	0	0	0	0
C = 1	0	0	0	0

Wzór z tabeli Karnaugh:

$$P12 = P13 = P21 = P22 = P23 = P24 = 0$$

Rys. 9. Schemat podukładu dla wyjść P12, P13, P21, P22, P23 i P24



Sprawdzenie poprawności wyprowadzeń

W celu sprawdzenia poprawności wyprowadzonych wzorów napisany został program w języku python, zaprezentowany poniżej, symulujący działanie projektowanego układu.

1. Implementacja wykorzystanych bramek logicznych NAND

```
def nand2(A, B):
   if A and B: return 0
   return 1

def nand3(A, B, C):
   if A and B and C: return 0
   return 1
```

2. Implementacja otrzymanych funkcji sygnałów wyjściowych

```
def P41_P44(A, B, C):
   return nand2(
                  nand2( nand2( A, A ), nand2( B, B ) ),
                  nand2( nand2( A, A ), nand2( B, B ) )
               )

def P31_P34(A, B, C):
   return nand2(
                  nand2( B, nand2(C, C) ),
                  nand2( A, B )
               )

def P42_P43 (A, B, C):
   return nand2(
                  nand2( A, A ),
                  nand3( nand2(A, A), nand2(B, B), nand2(C, C) )
               )

def P32_P33 (A, B, C):
   return nand3(
                  A,
                  nand2( B, C ),
                  nand2( nand2(B,B), nand2(C,C) )
               )
```

```
def P12_P13_P21_P22_P23_P24 (A, B, C):
   return 0

def P11_P14 (A, B, C):
   return 1
```

3. Test poprawności wyprowadzenia

```
print("A B C | P11 P12 P13 P14 P21 P22 P23 P24 P31 P32 P33 P34 P41 P42 P43 P44")
for a in [0, 1]:
   for b in [0, 1]:
       for c in [0, 1]:
           print(a, b, c, '|',
                 P11_P14(a,b,c), ' ', P12_P13_P21_P22_P23_P24(a,b,c), ' ',
                 P12_P13_P21_P22_P23_P24(a, b, c), ' ', P11_P14(a,b,c), ' ',
                 P12_P13_P21_P22_P23_P24(a,b,c), ' ',
                 P12_P13_P21_P22_P23_P24(a,b,c), ' ',
                 P12_P13_P21_P22_P23_P24(a,b,c), ' ',
                 P12_P13_P21_P22_P23_P24(a,b,c), ' ',
                 P31_P34(a,b,c), ' ', P32_P33(a,b,c), ' ', P32_P33(a,b,c), ' ',
                 P31_P34(a,b,c), ' ',
                 P41_P44(a,b,c), ' ', P42_P43(a,b,c), ' ', P42_P43(a,b,c), ' ',
                 P41_P44(a,b,c), ' ')
```

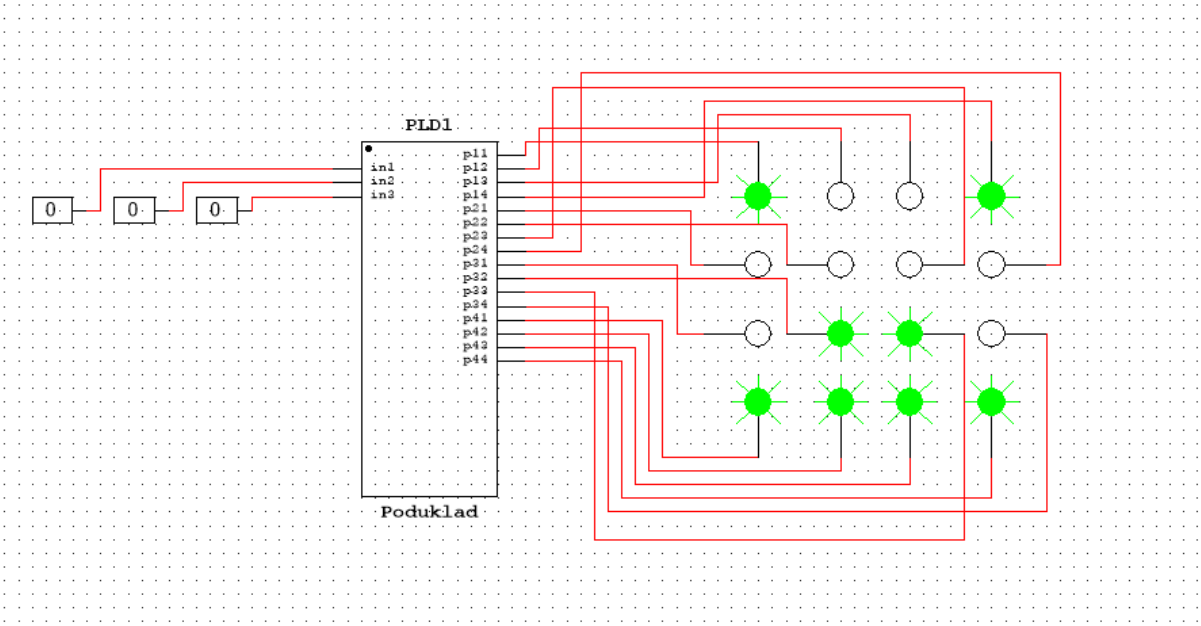
Po uruchomieniu programu otrzymaliśmy tabelę prawdy przedstawioną na rysunku 4. Uzyskane wyniki zgadzają się z przedstawioną wcześniej tabelą 1, co wskazuje na poprawność wyprowadzeń.

```
A B C | P11 P12 P13 P14 P21 P22 P23 P24 P31 P32 P33 P34 P41 P42 P43 P44
0 0 0 | 1   0   0   1   0   0   0   0   0   1   1   0   1   1   1   1
0 0 1 | 1   0   0   1   0   0   0   0   0   1   1   0   1   0   0   1
0 1 0 | 1   0   0   1   0   0   0   0   1   1   1   1   0   0   0   0
0 1 1 | 1   0   0   1   0   0   0   0   0   1   1   0   0   0   0   0
1 0 0 | 1   0   0   1   0   0   0   0   0   1   1   0   0   1   1   0
1 0 1 | 1   0   0   1   0   0   0   0   0   0   0   0   0   1   1   0
1 1 0 | 1   0   0   1   0   0   0   0   1   0   0   1   0   1   1   0
1 1 1 | 1   0   0   1   0   0   0   0   1   1   1   1   0   1   1   0
```

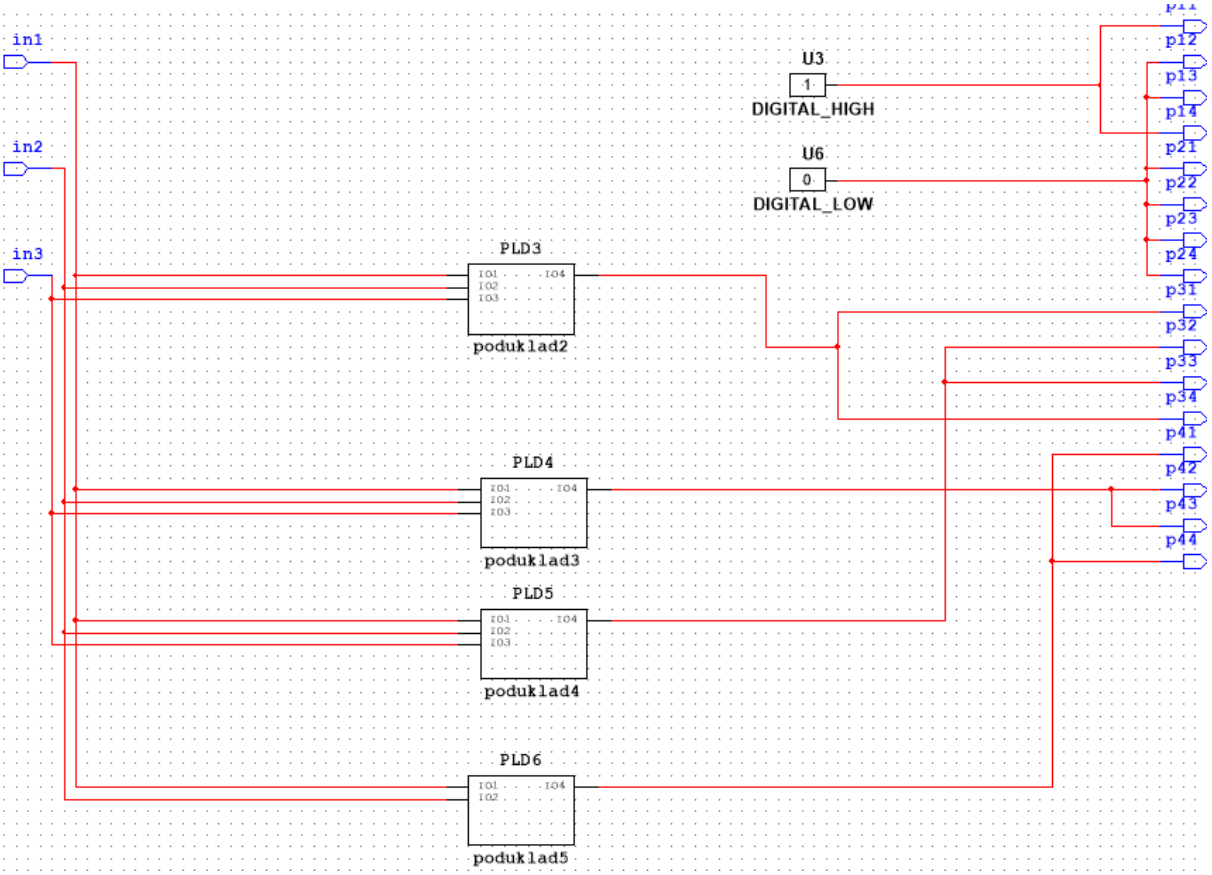
Rys. 10. Tabela prawdy uzyskana w programie testującym wyprowadzone funkcje

Schemat układu na podstawie wyprowadzonych wzorów

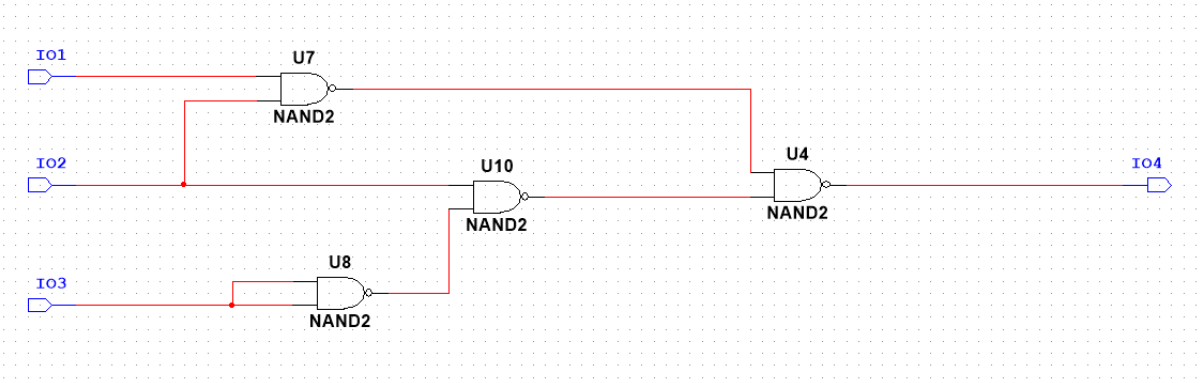
Rys. 11. Ogólny schemat układu



Rys. 12. Schemat podukładu

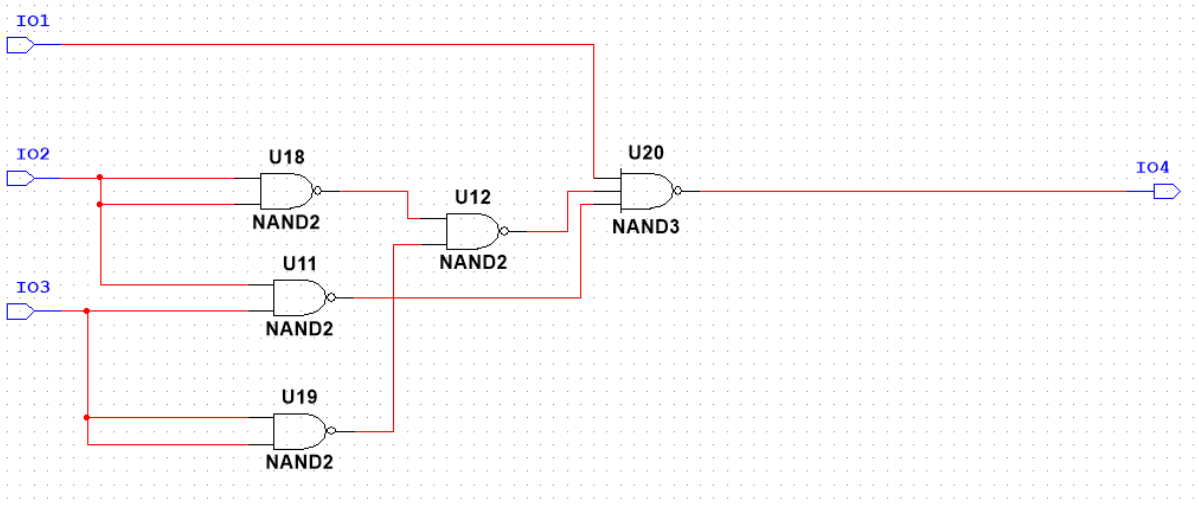


Rys. 8. Schemat podukładu obsługującego wyjścia P31 i P34



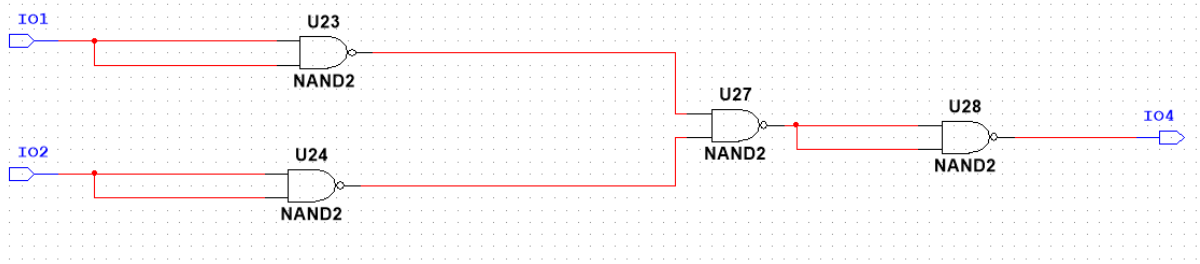
$$P31 = P34 = B\overline{C} + AB = \overline{\overline{B\overline{C}} + \overline{AB}} = \overline{\overline{B\overline{C}} \cdot \overline{AB}} = \overline{\overline{B} \cdot \overline{C\overline{C}} \cdot \overline{AB}}$$

Rys. 9. Schemat podukładu obsługującego wyjścia P32 i P33



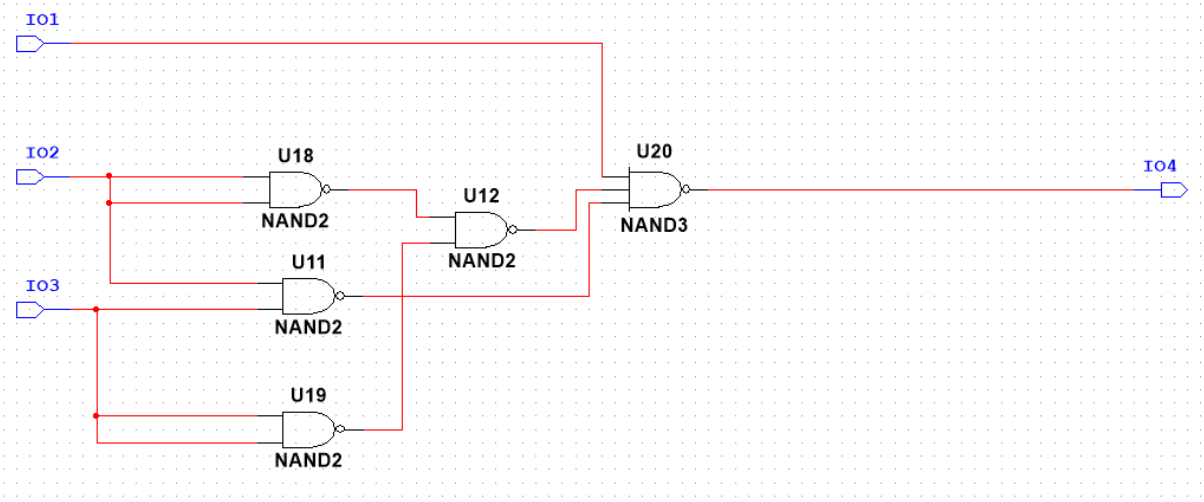
$$P32 = P33 = \overline{A} + BC + \overline{B}\,\overline{C} = \overline{\overline{A} + BC + \overline{B}\,\overline{C}} = \overline{A \cdot \overline{BC} \cdot \overline{\overline{B}\,\overline{C}}} = \overline{A \cdot \overline{BC} \cdot \overline{B\overline{B}} \cdot \overline{C\overline{C}}}$$

Rys. 10. Schemat podukładu obsługującego wyjścia P41 i P44



$$P41 = P44 = \overline{A}\,\overline{B} = \overline{A\,A} \cdot \overline{B\,B} = \overline{\overline{A\,A} \cdot \overline{B\,B}} = \overline{\overline{A\,A} \cdot \overline{B\,B} \cdot \overline{A\,A} \cdot \overline{B\,B}}$$

Rys. 11. Schemat podukładu obsługującego wyjścia P42 i P43



$$P42 = P43 = A + \overline{B}\,\overline{C} = \overline{\overline{A + \overline{B}\,\overline{C}}} = \overline{\overline{A} \cdot \overline{\overline{B}\,\overline{C}}} = \overline{\overline{A\,A} \cdot \overline{\overline{B\,B} \cdot \overline{C\,C}}}$$

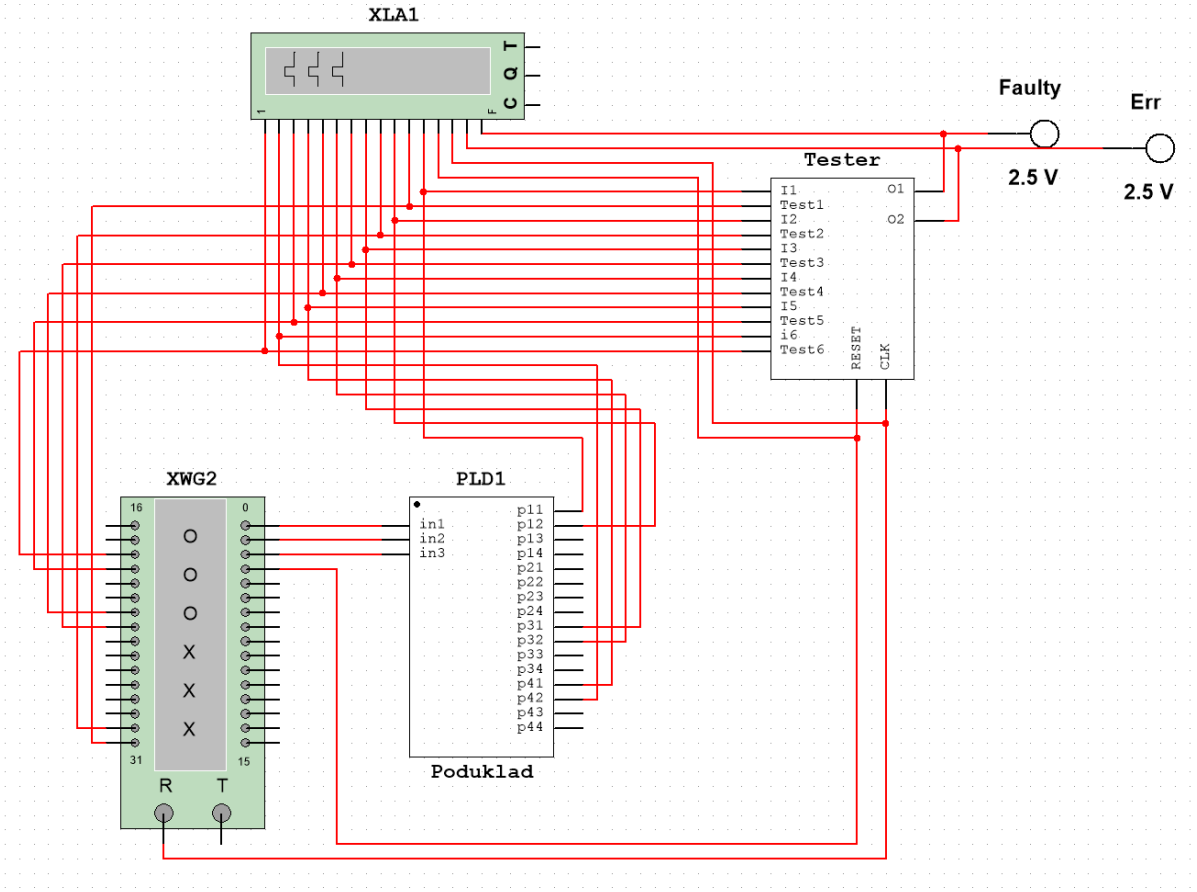
Układ testujący

W celu weryfikacji poprawności działania zaprojektowanego układu zbudowano układ testujący, który umożliwia sprawdzenie, czy wyświetlane na matrycy 4x4 wzory graficzne (emotikony) odpowiadają danym wartościom dla poszczególnych kombinacji wejściowych.

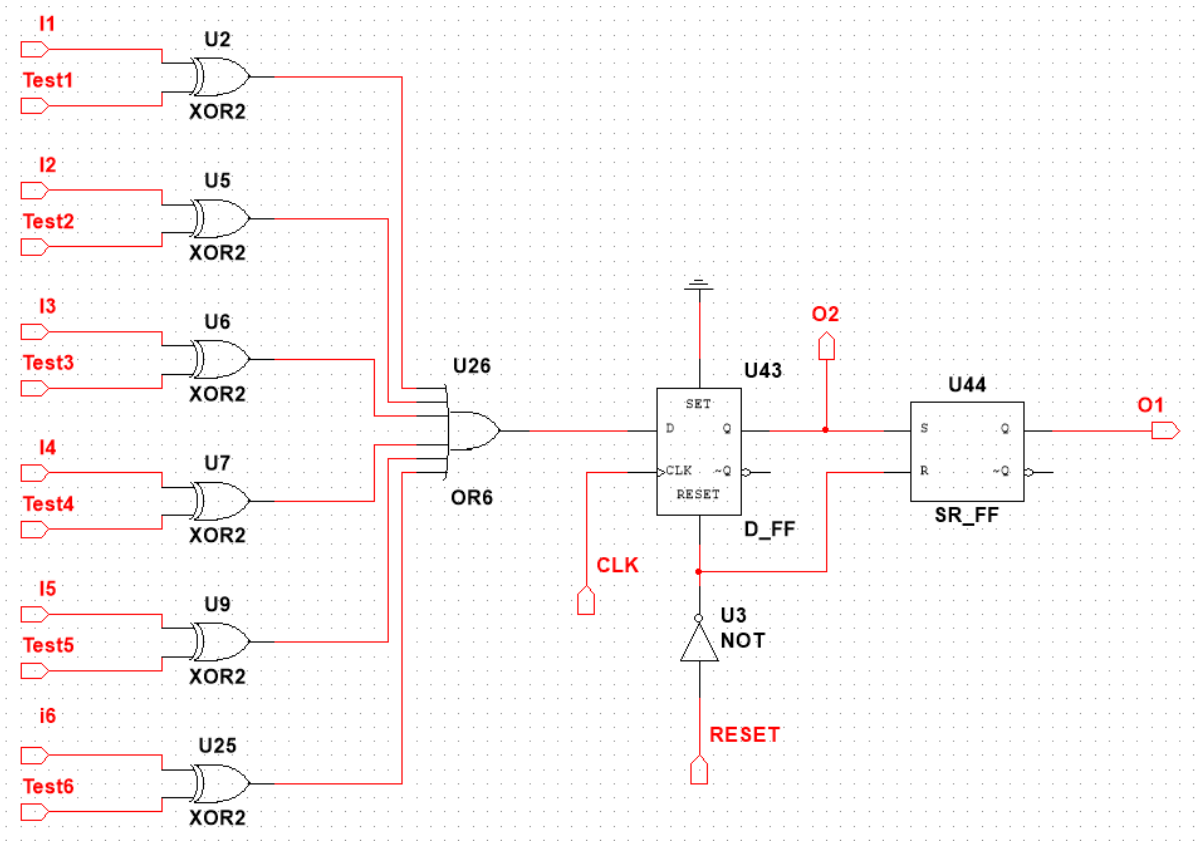
By ułatwić sprawdzanie układu możemy sprawdzać tylko 6 wyjść, każde innego rodzaju. Dla naszych danych wzięliśmy pod uwagę wyjścia P11, P12, P31, P32, P41, P42.

Układ testujący składa się z generatora sygnałów wejściowych, testera stanów logicznych (bramek logicznych), który porównuje uzyskane wyniki z wartościami oczekiwanymi oraz analizatora logicznego. Kluczowym elementem są bramki XOR (wewnątrz testera), które wykrywają ewentualne niezgodności – jeśli wynik nie jest zgodny z prawidłowym wzorem, zapala się dioda sygnalizująca błąd.

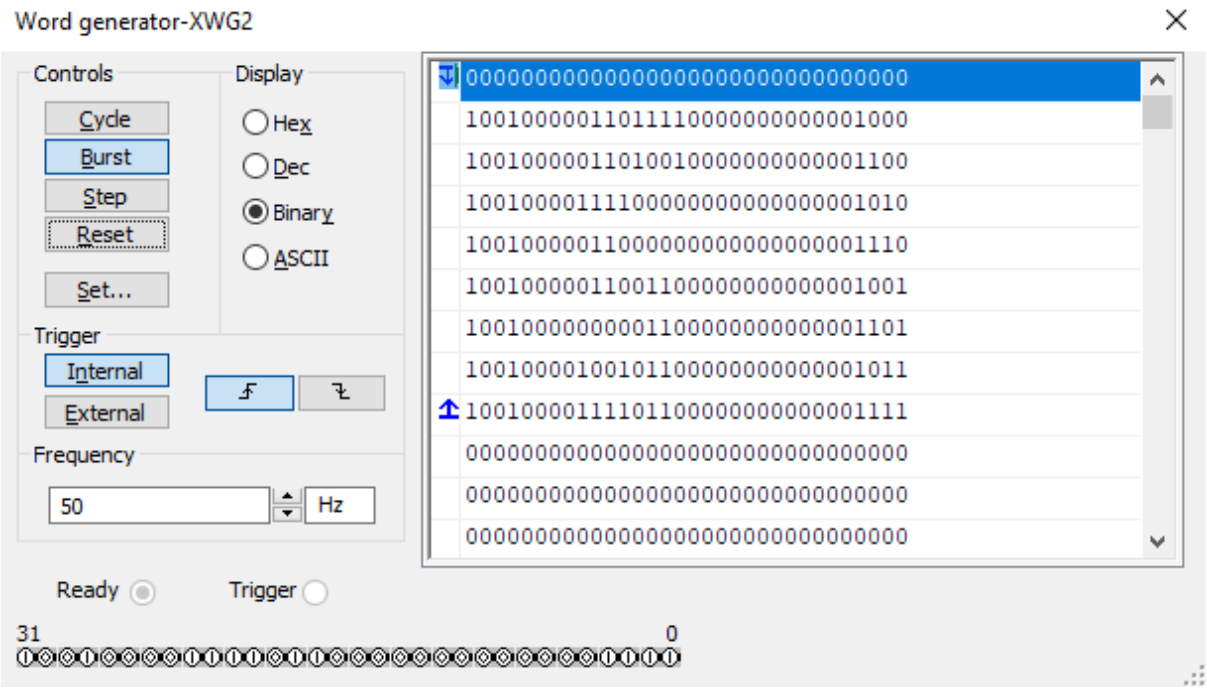
Rys. 12. Schemat układu testującego



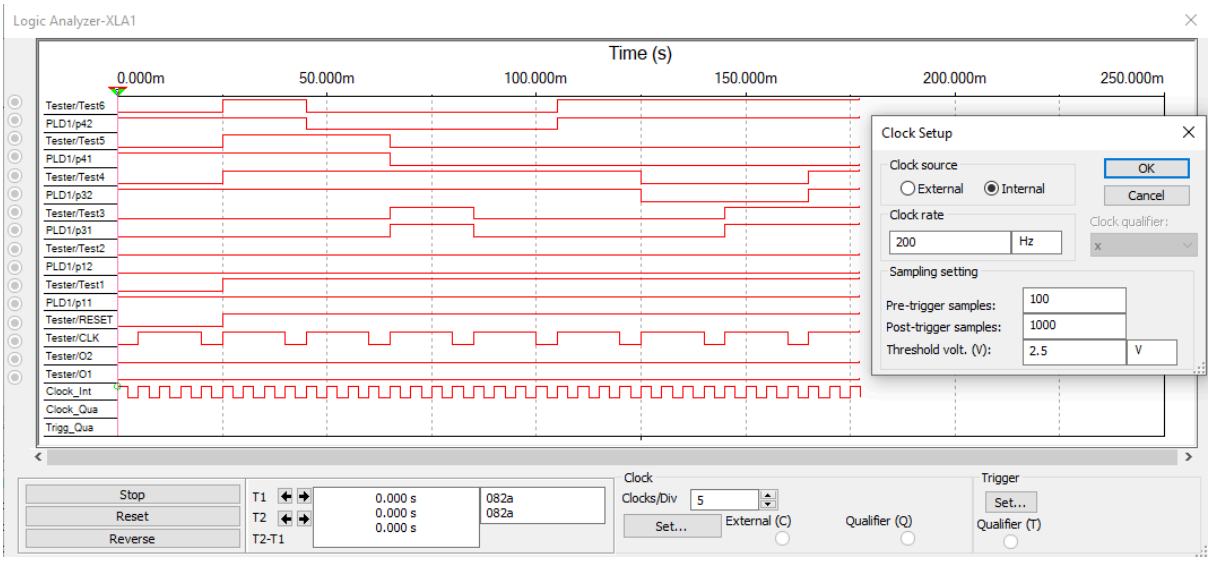
Rys. 13. Schemat podukładu testującego



Rys. 14. Generator słów

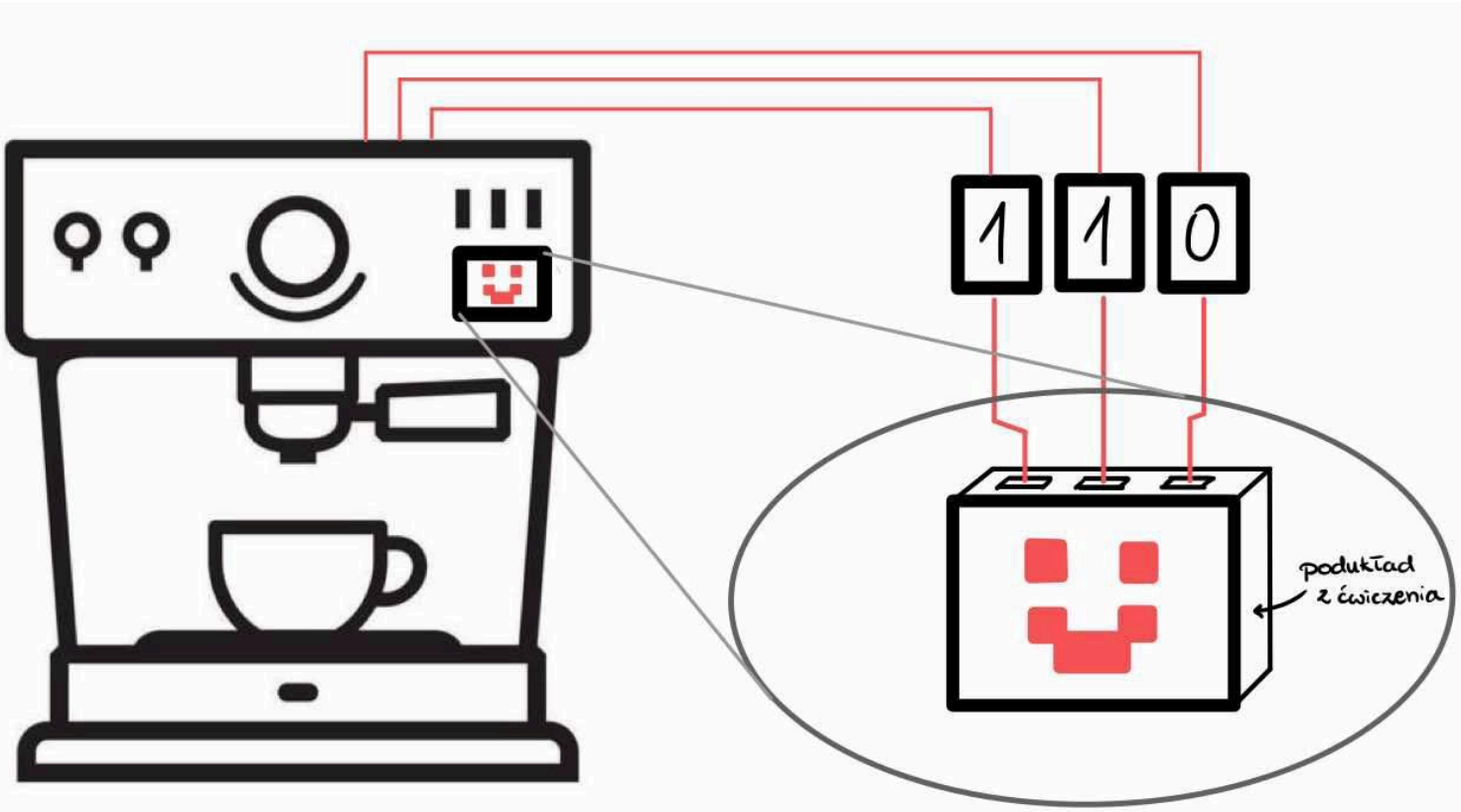


Rys. 15. Analizator logiczny



Zastosowania układu z ćwiczenia

Układ zbudowany w zadaniu może zostać zastosowany w urządzeniach, które komunikują się z użytkownikiem za pomocą prostych emotikon. W urządzeniu takim jak ekspres do kawy czujniki mogłyby kontrolować wyświetlanie różnych emotikon sygnalizujących status urządzenia, na przykład: 😊- wszystko działa, 😠- problem itd. Każdy ze statusów miałby przypisaną wartość od 0 do 7, która byłaby przekazywana do zaprojektowanego układu.



Wnioski

Ćwiczenie pokazało, że możliwe jest zrealizowanie dowolnej funkcji logicznej wyłącznie za pomocą bramek NAND, jednak wiąże się to z koniecznością stosowania dodatkowych przekształceń oraz większej liczby bramek niż w przypadku standardowego podejścia z różnymi typami bramek logicznych.

Minimalizacja funkcji logicznych za pomocą tablic Karnaugh okazała się kluczowa dla uproszczenia układu i zmniejszenia liczby wymaganych operacji. Proces ten umożliwia efektywne uzyskanie równań logicznych, które następnie zostały przekształcone do postaci wykorzystującej wyłącznie bramki NAND.

Dzięki implementacji w programie Multisim możliwa była weryfikacja poprawności działania układu oraz identyfikacja ewentualnych błędów projektowych. Ćwiczenie pozwoliło również na lepsze zrozumienie praktycznych aspektów projektowania układów cyfrowych oraz znaczenia optymalizacji w inżynierii logicznej.