Sprawozdanie z układów logicznych	Rok 2021
Jakub Samulski (260407)	Ćwiczenie nr 2
Kacper Suchanek (260468)	Temat: Układy kombinacyjne
Grupa laboratoryjna nr Z01-45u Prowadzący: mgr inż. Karol Stasiński	Piątek
	17.05-18.35

Spis treści

Zagadnienia do opracowania

Funkcja przełączająca f(x_1, x_2, … , x_n), to takie odwzorowanie, które dla kombinacji argumentów x_1, x_2, …, x_n przyjmujących wartości „0” lub „1” przyporządkowuje rozwiązanie ze zbioru {0, 1}.

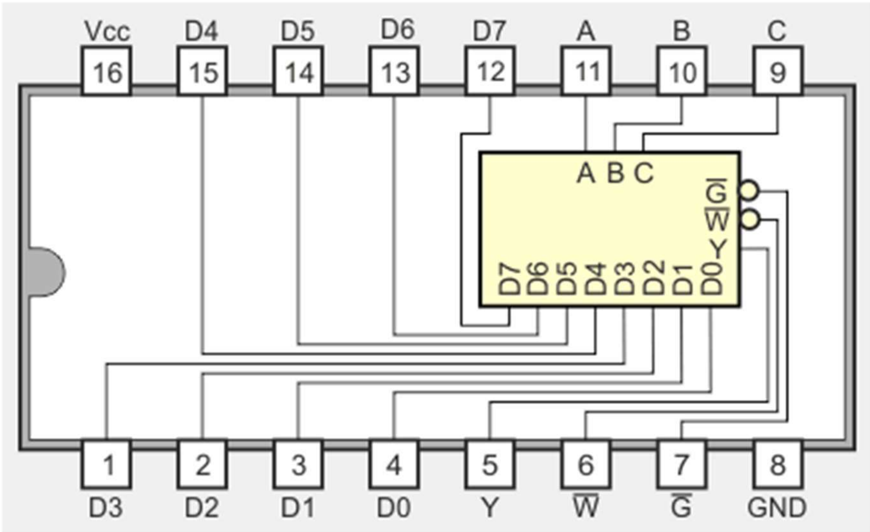
Tablica prawdy – układ tabelaryczny zero-jedynkowych kombinacji wartości logicznych argumentów danej funkcji zdaniowej i odpowiadających im wartości logicznych tejże funkcji, w którym prawdzie odpowiada wartość 1, a fałszowi przypisuje się wartość 0.

I prawo De Morgana – prawo zaprzeczania koniunkcji: negacja koniunkcji jest równoważna alternatywie negacji.

II prawo De Morgana – prawo zaprzeczenia alternatywy: negacja alternatywy jest równoważna koniunkcji negacji.

Tablica Karnaugha jest uporządkowaną w specyficzny sposób postacią zapisu tablicy wartości funkcji logicznej. Korzysta się z niej w procesie minimalizacji funkcji logicznych. Tablica Karnaugha ma strukturę prostokątną, złożoną z 2n elementarnych pól. Każde pole reprezentuje iloczyn pełny w odniesieniu do zmiennych wejściowych, czyli zmiennych niezależnych (argumentów) danej funkcji. Na marginesach tablicy wpisuje się w określonym porządku (wg kodu Graya) wartości argumentów. Ułożenie tablicy Karnaugha polega na takim zgrupowaniu wszystkich kombinacji wartości argumentów, aby zawsze przy przejściu z danego pola do pola sąsiedniego zmieniała się wartość tylko jednego argumentu. Zasada sąsiedztwa obowiązuje również dla pól leżących przy krawędziach tablicy. Przy minimalizacji korzysta się z faktu, że dwa człony iloczynowe wyrażenia, różniące się jedną negacją, można zastąpić jednym członem, bez literału różnicującego. Działanie takie nosi nazwę sklejania, a sklejane człony to wyrażenia sąsiednie. Główną trudność w procesie sklejania (minimalizacji) stanowi wyszukiwanie wyrażeń sąsiednich. W metodzie Karnaugha rozwiązuje się to w ten sposób, że różniącym się tylko o negację pełnym iloczynom przyporządkowuje się leżące obok siebie pola tablicy, do których następnie wpisuje się wartość funkcji. Jeśli w zbudowanej tablicy dwa symbole 1 leżą obok siebie, to odpowiadają one wyrażeniom sąsiednim, które można skleić. Im większe pole sklejeń uda się odszukać i oznaczyć w tablicy, tym prostsze będzie wyrażenie, opisujące zawarte w tym polu symbole, więc w procesie minimalizacji powinno się tworzyć pola największe z możliwych. W przypadku funkcji niezupełnych bardzo pomocne w zwiększaniu rozmiarów pola mogą być wartości nieokreślone funkcji, którym dowolnie można przypisywać wartości 0 lub 1 tak, by wynik minimalizacji był najlepszy.

Specyfikacja i schemat układu scalonego 74151



Wejścia i wyjścia układu 74151:

- D_1 - D_7 – wejścia danych
- y, ⌐ – wyjście
- G – Wyjście strobujące (blokujące)
- GND – uziemienie
- A, B, C – wejścia informacyjne (adresowe)
- V_{cc} – zasilanie

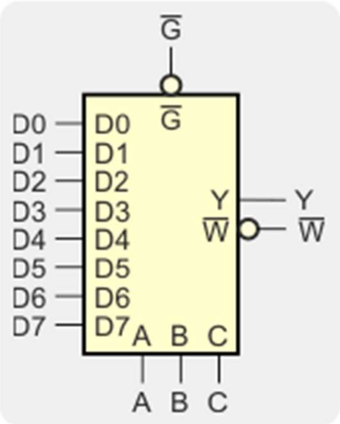
Układ scalony 74151 to multiplekser 8-bitowy (8 do 1). Multiplekser to cyfrowy układ kombinacyjny służący do wyboru jednego z kilku dostępnych sygnałów wejściowych i przekazania go na wyjście. Na wejścia danych (D1-D7) przekazywane są odpowiednie wartości logiczne. Do wyboru konkretnego sygnału wejściowego służą wejścia informacyjne (A, B, C), na które należy podać nr konkretnego wejścia w postaci binarnej (np. dla C=0, B=0, A=1 na wyjście zostanie przekazany sygnał z wejścia D1, dla C=0, B=1, A=0 zostanie przekazany sygnał D2 itd. Multiplekser posiada również wejście strobujące (blokujące działanie). Gdy na wejściu strobującym zostanie podana wartość 1, to wówczas na wyjście zostanie przekazana wartość 0.

Zastosowania

Jednym z zastosowań multiplekserów jest możliwość prostej realizacji przy ich pomocy układów kombinacyjnych. Używając ich można zrealizować dowolny układ kombinacyjny o ilości wejść równej ilości wejść adresujących i jednym wyjściu. Minimalnie rozbudowując układ można zrealizować także układ o ilości wejść większej o jeden niż ilość wejść adresujących.

Tablicę funkcji logicznej możemy interpretować w ten sposób, że dla każdej kombinacji wejść należy wybrać 0 lub 1. Jeśli więc x_1, x_2, i x_3 zinterpretujemy jako wejście wybierające A, B i C, to wartość funkcji Y, staje się równa wejściom odpowiadającym poszczególnym adresom wejść. Multiplekser można więc traktować jako układ realizujący dowolną funkcję n zmiennych, gdzie n jest liczbą wejść adresowych. Realizacja funkcji polega na odpowiednim przypisaniu wejściom informacyjnym wartości 0 i 1.

Schemat bloku



Schemat sieci logicznej

Schemat sieci logicznej multipleksera typu 74151

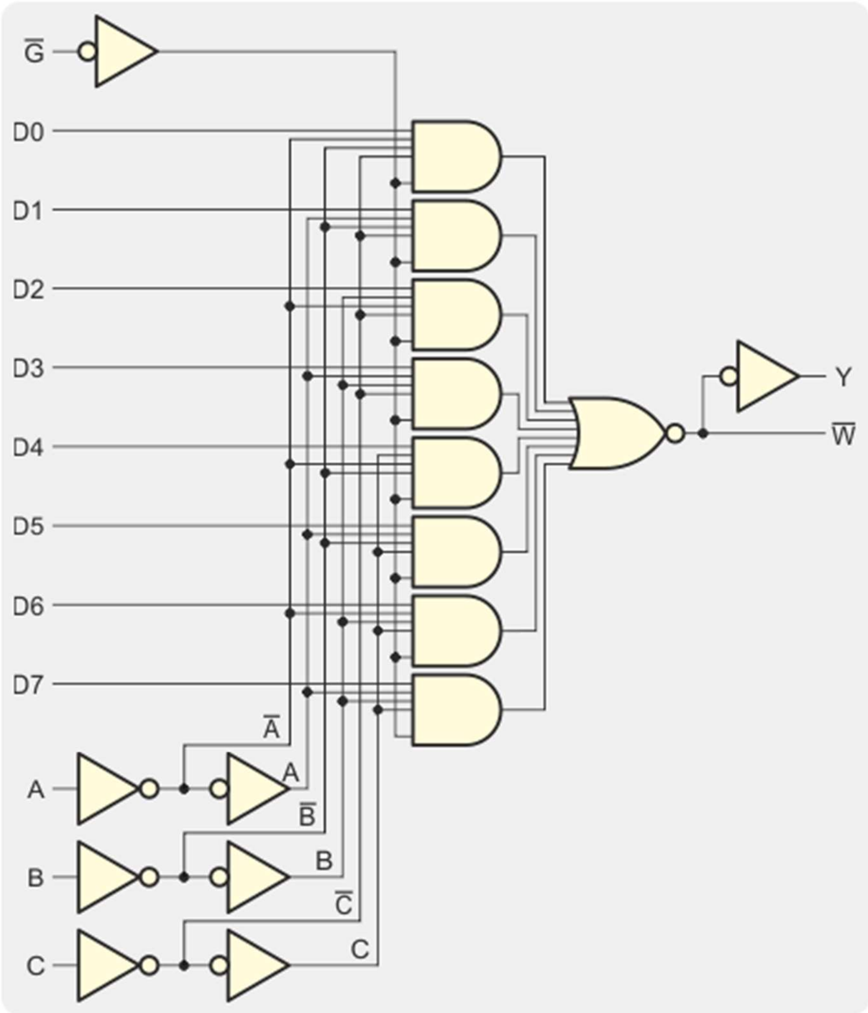


Tabela Stanów multipleksera 74151

Wejścia				Wyjścia	
Wybór			STROBE		
C	B	A	$\overline{G}$	Y	$\overline{W}$
X	X	X	1	0	1
0	0	0	0	D0	$\overline{D0}$
0	0	1	0	D1	$\overline{D1}$
0	1	0	0	D2	$\overline{D2}$
0	1	1	0	D3	$\overline{D3}$
1	0	0	0	D4	$\overline{D4}$
1	0	1	0	D5	$\overline{D5}$
1	1	0	0	D6	$\overline{D6}$
1	1	1	0	D7	$\overline{D7}$

0 – Stan Low

1- Stan High

X-Stan obojętny(nie wpływa na prace układu)

Syntezy funkcji

Synteza funkcji a)

Funkcja z punktu a) ma postać:

$$\mathrm{f(a, b, c, d)} = \overline{(\overline{\overline{a+b}+\bar{c}}) + d}$$

Tablica prawdy funkcji a) wygląda następująco

a	b	c	d	$\overline{a+b}$	$\bar{c}$	$\overline{\overline{a+b}+\bar{c}}$	$\overline{(\overline{\overline{a+b}+\bar{c}})+d}$
0	0	0	0	1	1	0	1
0	0	0	1	1	1	0	0
0	0	1	0	1	0	0	1
0	0	1	1	1	0	0	0
0	1	0	0	0	1	0	1
0	1	0	1	0	1	0	0
0	1	1	0	0	0	1	0
0	1	1	1	0	0	1	0
1	0	0	0	0	1	0	1
1	0	0	1	0	1	0	0
1	0	1	0	0	0	1	0
1	0	1	1	0	0	1	0
1	1	0	0	0	1	0	1
1	1	0	1	0	1	0	0
1	1	1	0	0	0	1	0
1	1	1	1	0	0	1	0

Następnie tworzę tablicę Karnaugha w celu zminimalizowania funkcji

ab \ Cd	00	01	11	10
00	1	0	0	1
01	1	0	0	0
11	1	0	0	0
10	1	0	0	0

W grupy można połączyć całą 1. kolumnę oraz pierwszy i ostatni element 1. wiersza. Zatem ostateczna, zminimalizowana, postać funkcji f to:

$$f(a,b,c,d) = \bar{c}\bar{d} + \overline{abd}$$

Aby zrealizować funkcję jedynie przy pomocy bramek NAND, należy skorzystać z następujących praw algebry Boole'a:

$$\overline{a+b} = \bar{a} \cdot \bar{b}$$

$$\bar{\bar{a}} = a$$

$$f(a,b,c,d) = \bar{c}\bar{d} + \overline{abd} = \overline{\overline{\bar{c}\bar{d} + \overline{abd}}} = \overline{\overline{\bar{c}\bar{d}} \cdot \overline{\bar{a}\bar{b}\bar{d}}}$$

Powyższa postać funkcji f zawiera jedynie operacje koniunkcji i negacji, zatem da się zrealizować jedynie za pomocą bramek NAND.

Synteza funkcji b)

Funkcja b ma postać

$$f(a, b, c) = \overline{ab\bar{c} + \bar{a}\bar{b}c}$$

Tablica prawdy funkcji a) wygląda następująco

A	B	C	$ab\bar{c}$	$\bar{a}\bar{b}c$	$\overline{ab\bar{c} + \bar{a}\bar{b}c}$
0	0	0	0	0	1
0	0	1	0	1	0
0	1	0	0	0	1
0	1	1	0	0	1
1	0	0	0	0	1
1	0	1	0	0	1
1	1	0	1	0	0
1	1	1	0	1	1

Następnie tworzę tablicę Karnaugha w celu zminimalizowania funkcji

bc / a	00	01	11	10
0	1	0	1	1
1	1	1	1	0

Na tablicy zostały zaznaczone kolorami 3 grupy, które można połączyć ze sobą. Ostateczna postać funkcji:

$$f(a, b, c) = \bar{b}\bar{c} + ac + \bar{a}b$$

Korzystam z praw algebry Boole'a, aby zrealizować funkcję tylko przy pomocy bramek NAND

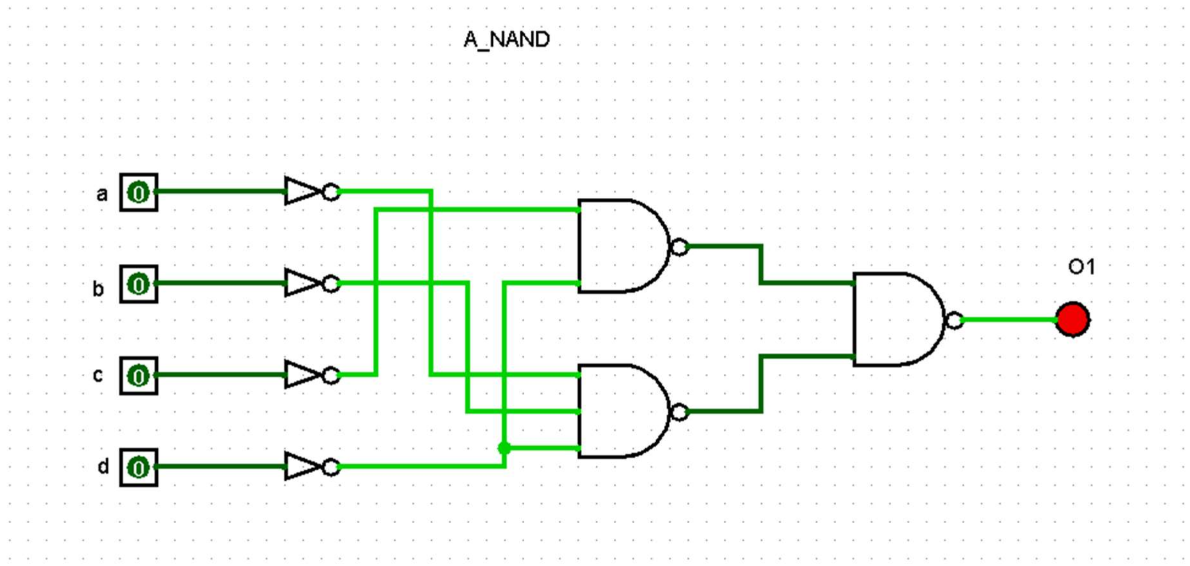
$$f(a, b, c) = \bar{b}\bar{c} + ac + \bar{a}b = \overline{\overline{\bar{b}\bar{c} + ac + \bar{a}b}} = \overline{\overline{\bar{b}\bar{c}} \cdot \overline{ac} \cdot \overline{\bar{a}b}}$$

Powyższa postać funkcji f zawiera jedynie operacje koniunkcji i negacji, więc jest możliwa do zrealizowania jedynie za pomocą bramek NAND.

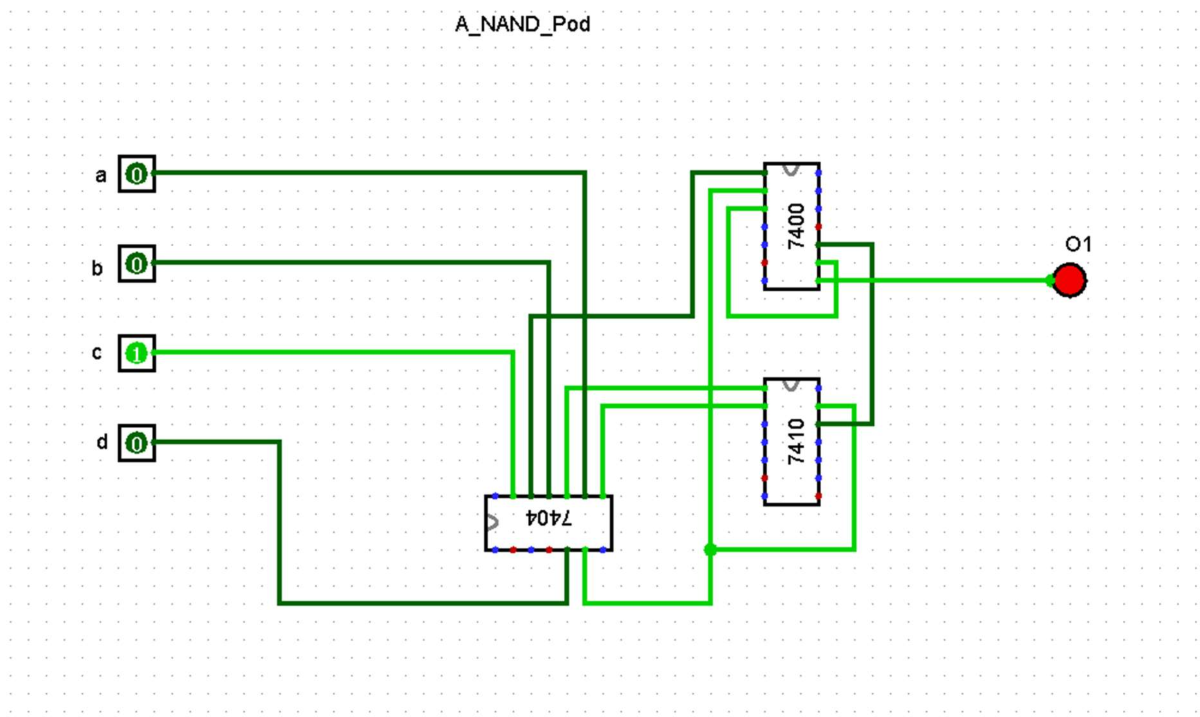
Realizacja funkcji

Funkcja a) przy pomocy bramek NAND

Schemat Logiczny



Schemat podłączenia



Funkcja a) przy pomocy multipleksera

Aby zrealizować funkcję a) za pomocą multipleksera, należy sprawdzić, dla jakich wartości zmiennych a, b, c, d, wartość funkcji wynosi 1. Z tablicy prawdy odczytujemy, że dzieje się tak dla (odpowiednio a, b, c, d): 0000, 0010, 0100, 1000, 1100. Są to binarne postaci liczb: 0, 2, 4, 8, 12. Zatem funkcja a) ma postać:

$$f(a,b,c,d) = \Sigma(0,2,4,8,12)$$

Napotykamy tutaj jednak pewien problem. Funkcja określona była dla 4 zmiennych, zaś do dyspozycji mamy multiplekser 8-bitowy z 3 wejściami adresowymi. Musimy więc zminimalizować naszą funkcję względem jednej ze zmiennych, np. a

Tworzę więc siatkę względem zmiennej a

bcd / a	000	001	011	010	110	111	101	100
0	1	0	0	1	0	0	0	1
1	1	0	0	0	0	0	0	1
wejscie	D_0	D_1	D_3	D_2	D_6	D_7	D_5	D_4

Numer wejścia danych określa nam binarnie zapisana liczba w poszczególnych kolumnach nagłówka, zaś wartości logiczne, które należy podać na wejście te wejścia określają wartości funkcji w poszczególnych kolumnach. Mamy więc:

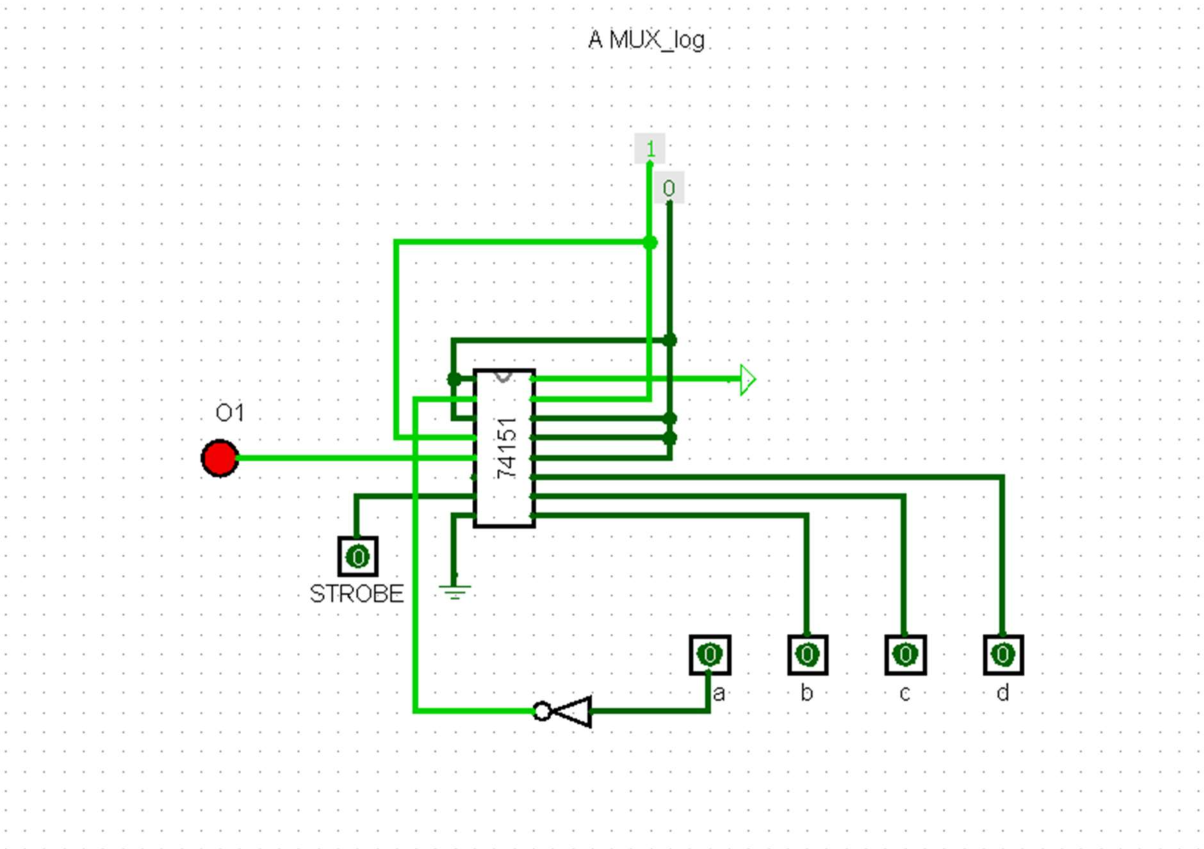
$D_0 = D_4 = 1$

$D_1 = D_3 = D_2 = D_6 = D_7 = D_5 = 0$

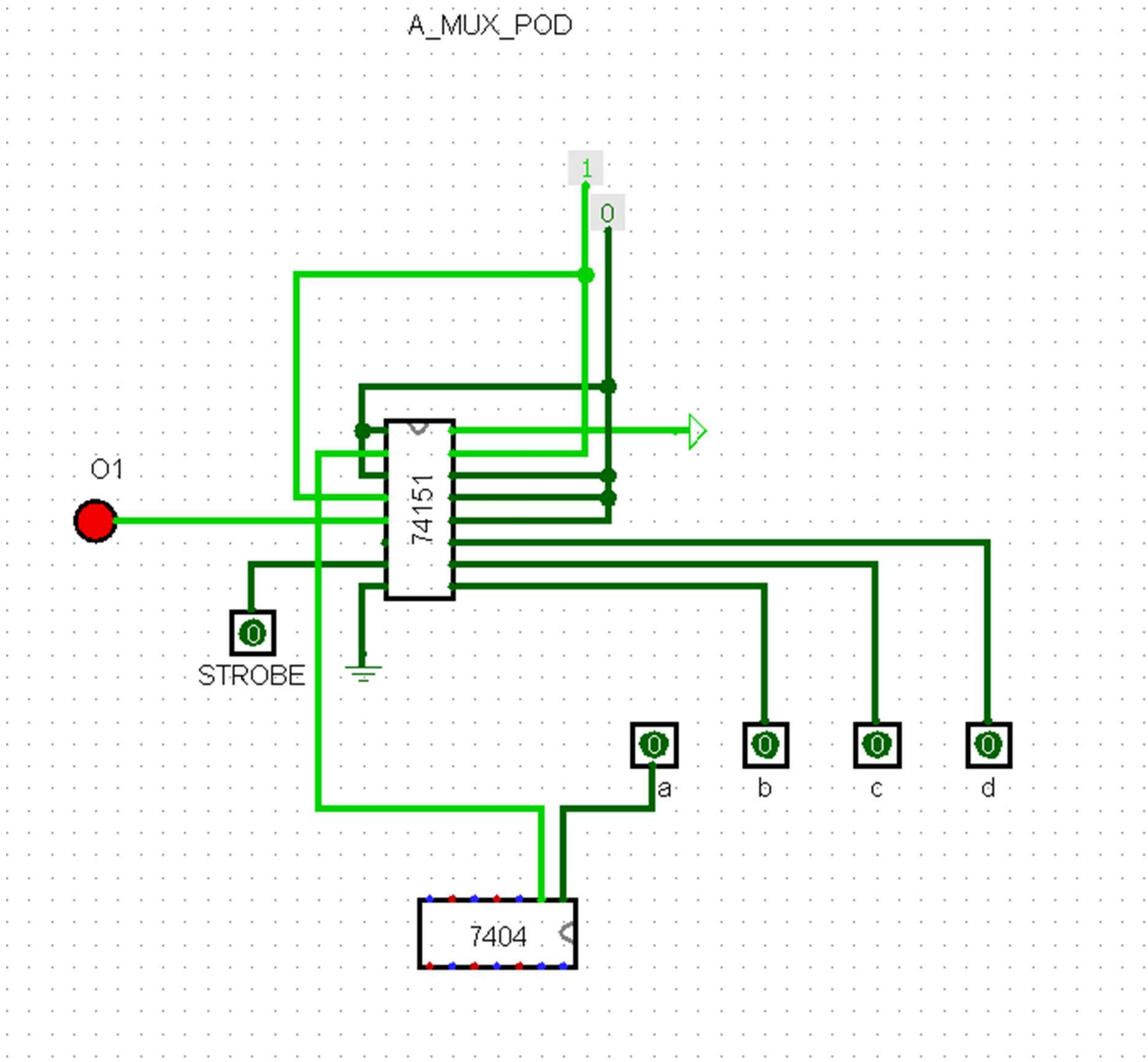
$D_2 = \overline{a}$

Do wejść D_0, D_4 należy podłączyć wartość 1, do D_1, D_3, D_2, D_6, D_7, D_5 wartość 0, do D_2 – $\overline{a}$

Schemat Logiczny

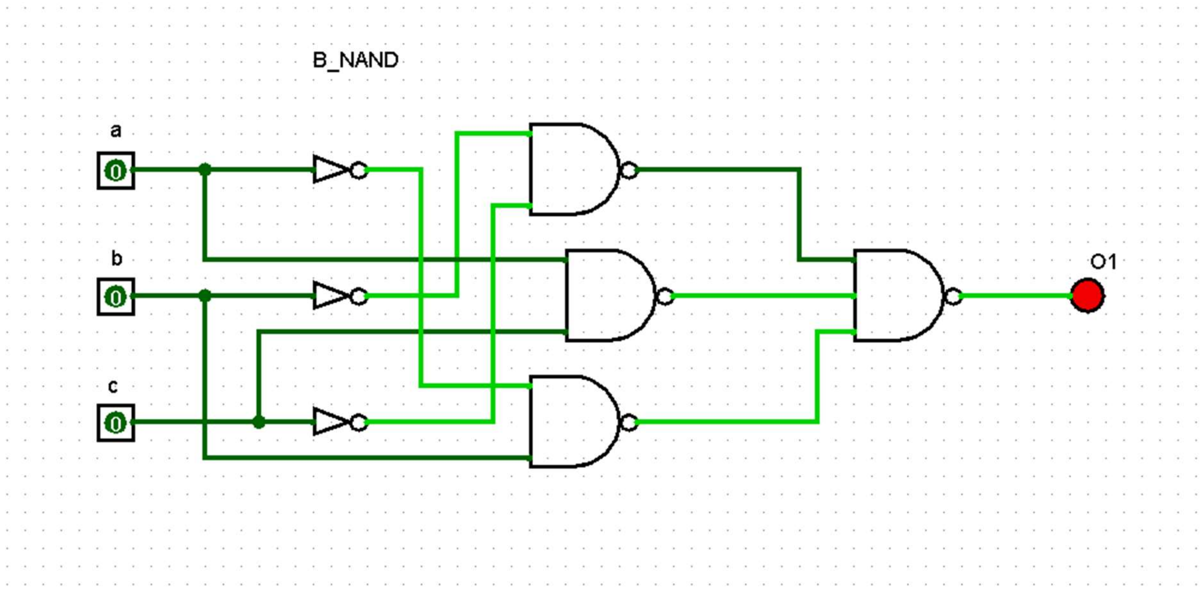


Schemat podłączenia

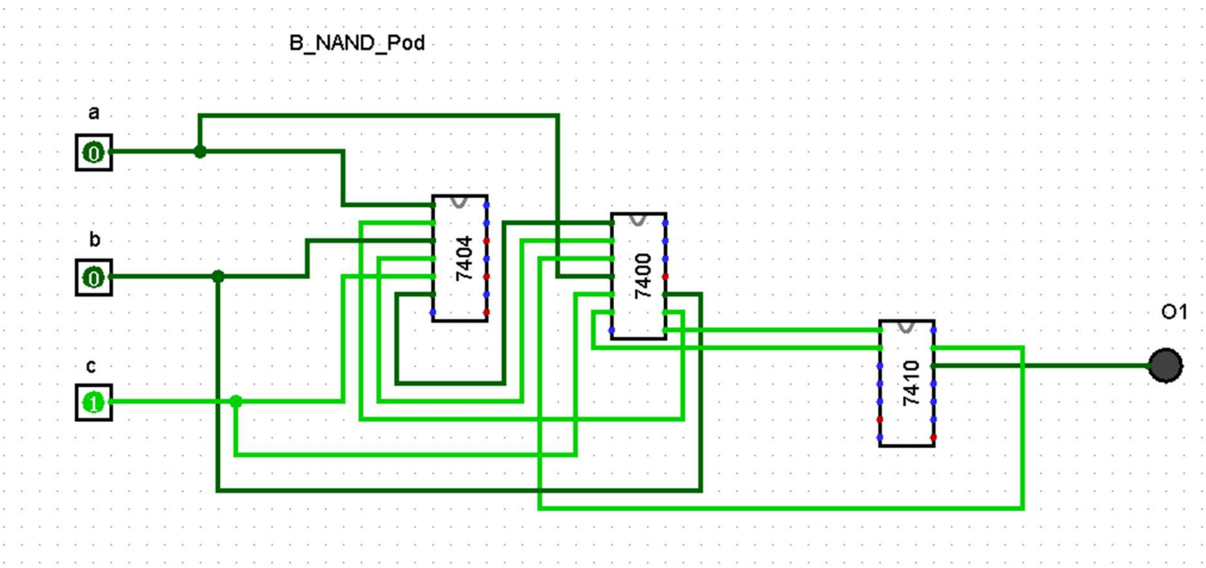


Funkcja b) przy pomocy bramek NAND

Schemat Logiczny



Schemat podłączenia



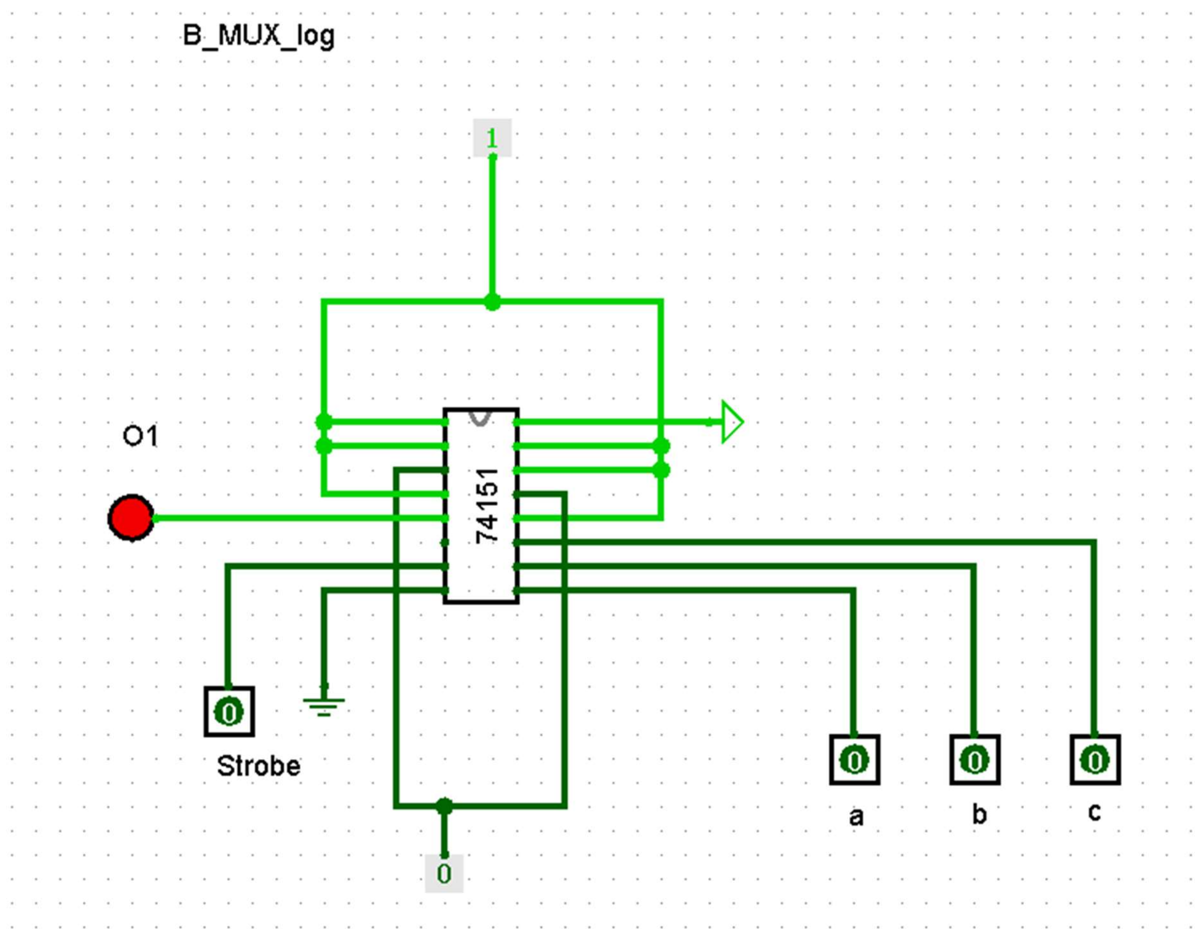
Funkcja b) przy pomocy multipleksera

Aby zrealizować funkcję b) za pomocą multipleksera, postępujemy analogicznie jak poprzednio. Funkcja jest już określona dla 3 zmiennych, zatem nie będzie potrzeby jej dalszego upraszczania. Z tablicy prawdy odczytujemy, że wartość logiczna 1 jest przyjmowana dla (odpowiednio a, b, c): 000, 010, 011, 100, 101, 111. Są to binarne postaci liczb: 0, 2, 3, 4, 5, 7. Zatem funkcja b) ma postać:

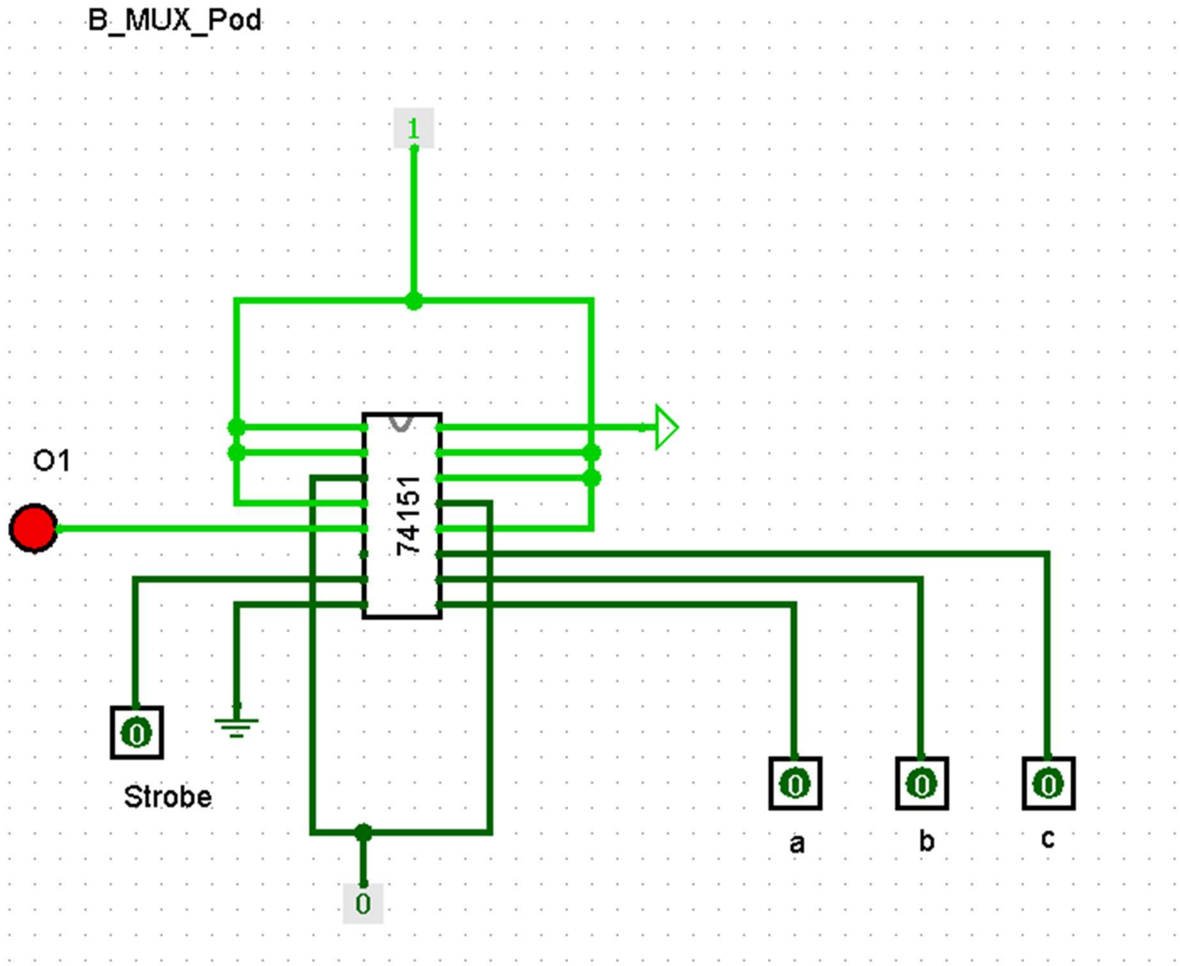
$$f(a, b, c) = \Sigma(0, 2, 3, 4, 5, 7)$$

Do wejść D_0, D_2, D_3, D_4, D_5, D_7 należy więc podłączyć wartość 1, do pozostałych wartość 0

Schemat Logiczny



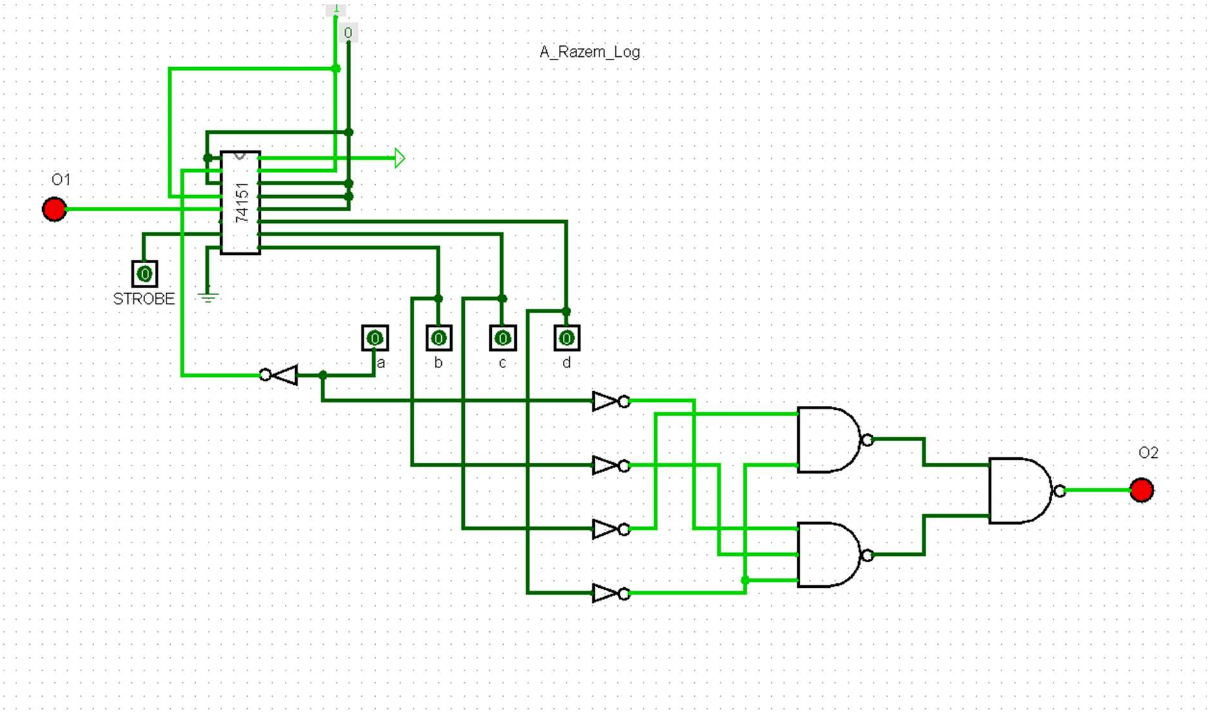
Schemat podłączenia



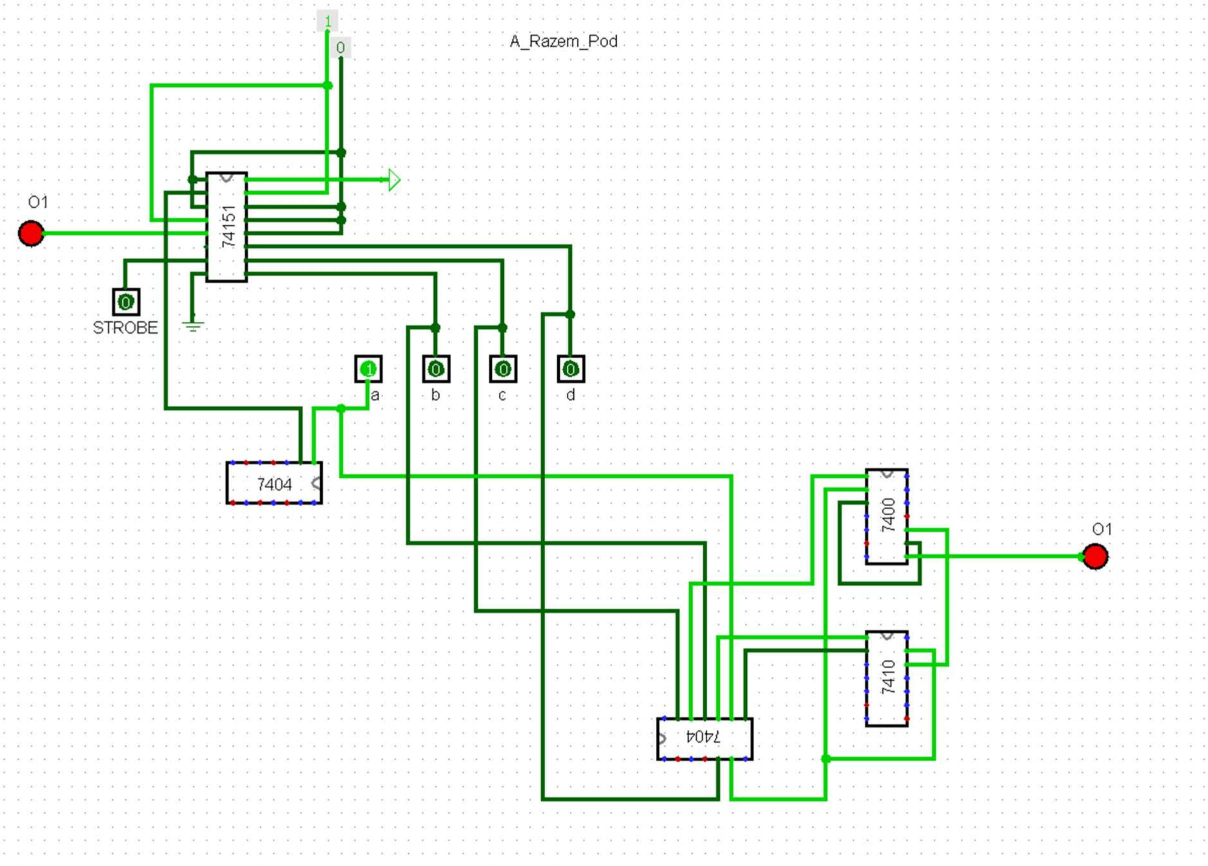
Połączenie układów

Funkcja a)

Schemat logiczny



Schemat podłączenia



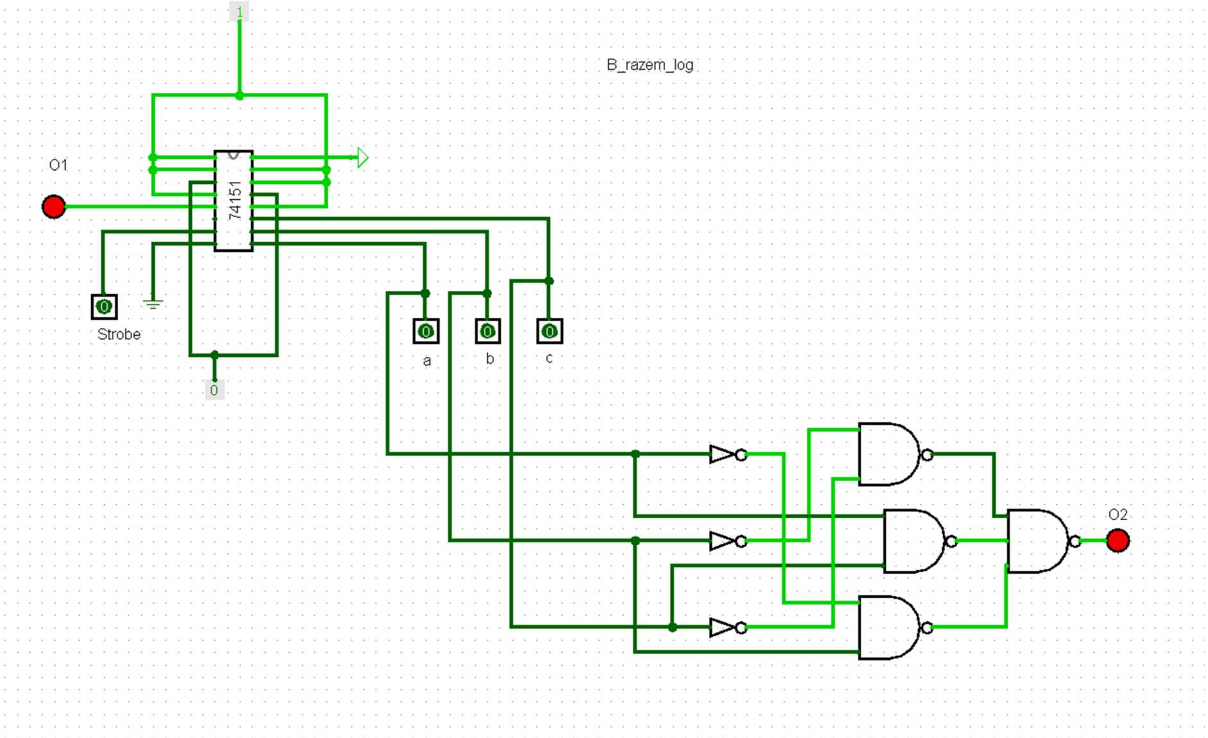
Tablica Prawdy

W tej i kolejnych tabelach 1 oznacza logiczną „jedynkę” (w układzie sygnał HIGH) analogicznie 0 oznacza sygnał LOW

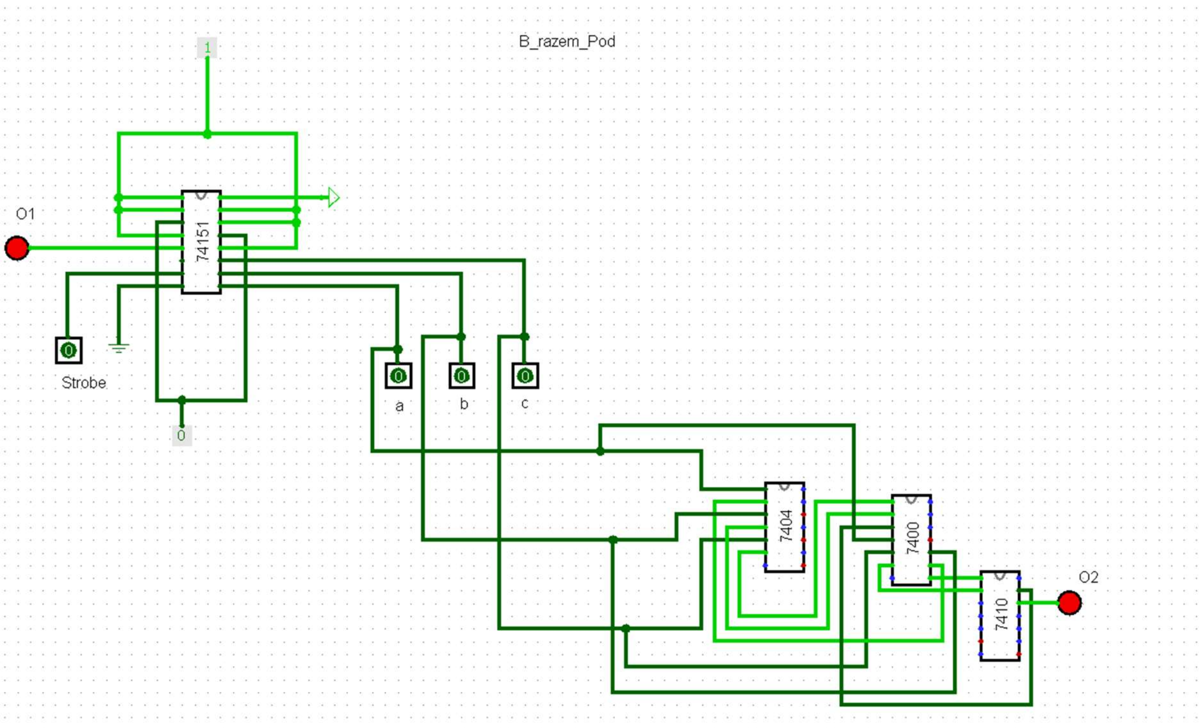
A	b	C	D	O1	O2
0	0	0	0	1	1
0	0	0	1	0	0
0	0	1	1	0	0
0	0	1	0	1	1
0	1	1	0	0	0
0	1	1	1	0	0
0	1	0	1	0	0
0	1	0	0	1	1
1	1	0	0	1	1
1	1	0	1	0	0
1	1	1	1	0	0
1	1	1	0	0	0
1	0	1	0	0	0
1	0	1	1	0	0
1	0	0	1	0	0
1	0	0	0	1	1

Funkcja B

Schemat logiczny



Schemat podłączenia



Tablica prawdy

A	B	C	O1	O2
0	0	0	1	1
0	0	1	0	0
0	1	1	1	1
0	1	0	1	1
1	1	0	0	0
1	1	1	1	1
1	0	1	1	1
1	0	0	1	1