Wydział	Imię i nazwisko		Rok	Grupa	Zespół
	1. Michał Rogowski				
WFiIS	2. Ihnatsi Yermakovich		II	10	02
PRACOWNIA ELEKTRONICZNA WFiIS AGH	Temat				Nr ćwiczenia
	Programowalna matryca logiczna - logika kombinacyjna				06
Data wykonania	Data oddania	Zwrot do poprawy	Data oddania	Data zaliczenia	OCENA
26.05.2022	04.06.2022				

Programowalna matryca logiczna - logika kombinacyjna

Ćwiczenie nr 06

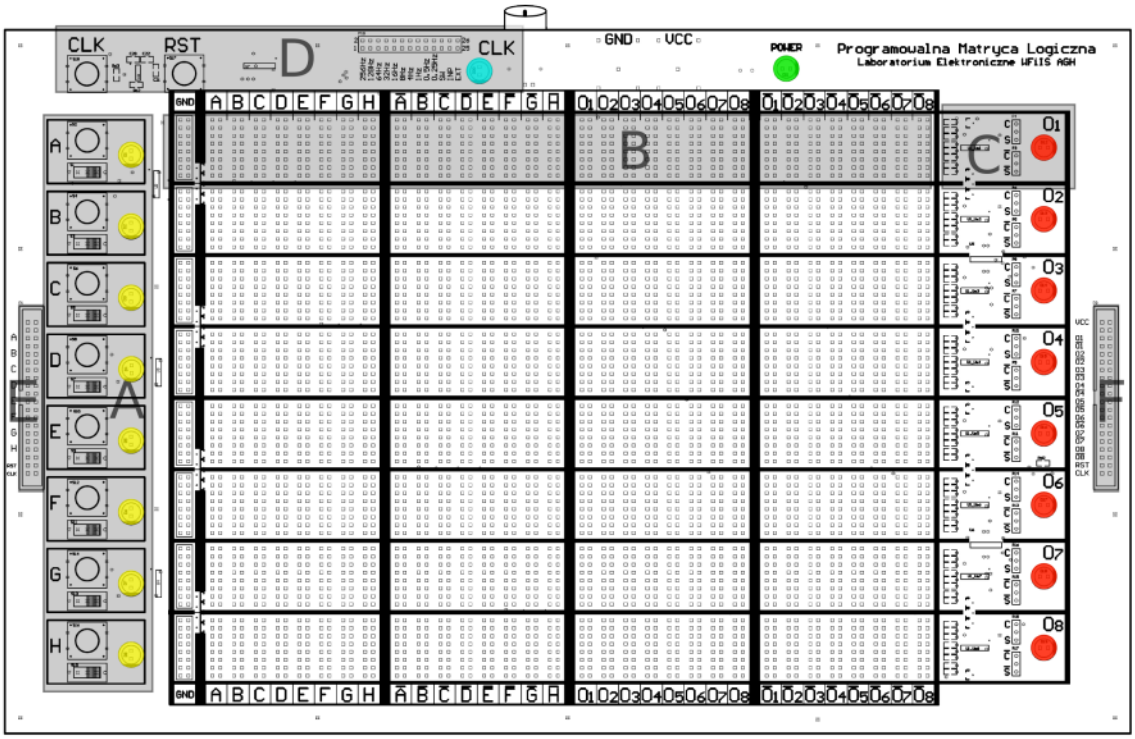
Michał Rogowski

Ihnatsi Yermakovich

1 Cel ćwiczenia

Celem ćwiczenia było zaprojektowanie układów kombinacyjnych oraz sprawdzenie ich działania w praktyce za pomocą matrycy logicznej.

2 Przebieg ćwiczenia



A – Grupa przełączników do ustawiania sygnałów wejściowych
B – Programowalny blok logiczny
C – Zworki wyboru wejścia kombinacyjnego (C) lub sekwencyjnego (S) wraz z diodami sygnalizującymi stan wyjścia danego bloku logicznego
D – Blok wyboru źródła sygnału zegarowego
E – Złącze doprowadzające sygnały wejściowe
F – Złącze wyprowadzające sygnały wyjściowe

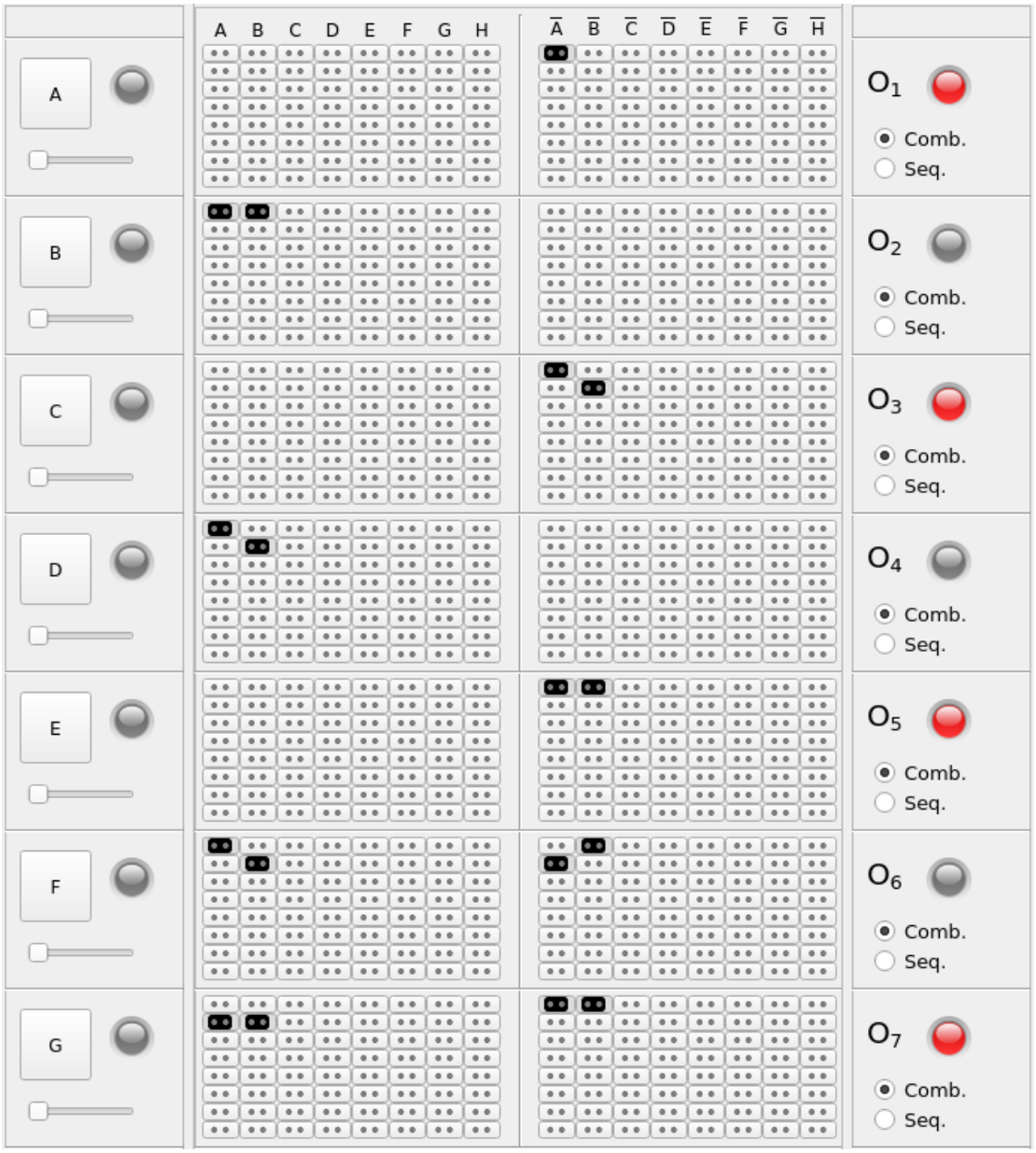
Rysunek 1: Budowa programowalnej matrycy logicznej

Korzystaliśmy z matrycy logicznej (rys. 1) działającej w trybie kombinacyjnym do zaprojektowania poszczególnych układów. Matryca zasilana była napięciem pojedynczym +5V względem masy.

3 Wyniki

3.1 Realizacja bramek logicznych

Korzystając z 7 wyjść matrycy zaprojektowaliśmy w postaci sumy iloczynów następujące bramki logiczne: **NOT**, **AND**, **NAND**, **OR**, **NOR**, **XOR**, **XNOR**. W celu wizualizacji konfiguracji matrycy i sprawdzenia jej poprawności dokonaliśmy symulacji w symulatorze PLD dostępnym na serwerze Taurus:



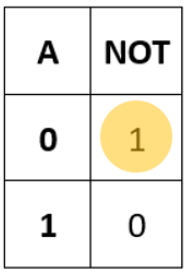
Rysunek 2: Konfiguracja matrycy logicznej dla zadanych bramek logicznych

Poniżej przedstawiona jest tabela prawdy dla wspomnianych bramek:

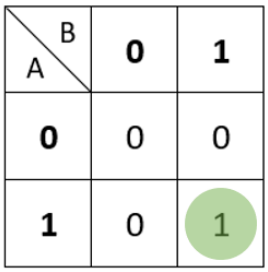
Tabela 1: Tabela prawdy dla wybranych bramek logicznych

A	B	NOT	AND	NAND	OR	NOR	XOR	XNOR
0	-	1	-	-	-	-	-	-
1	-	0	-	-	-	-	-	-
0	0	-	0	1	0	1	0	1
0	1	-	0	1	1	0	1	0
1	0	-	0	1	1	0	1	0
1	1	-	1	0	1	0	0	1

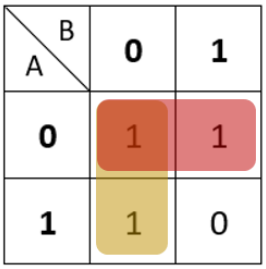
Oraz tablice Karnaugh'a:



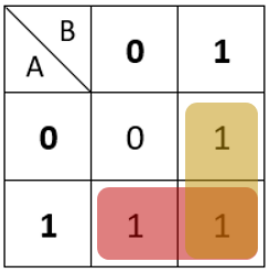
NOT: $\overline{A}$



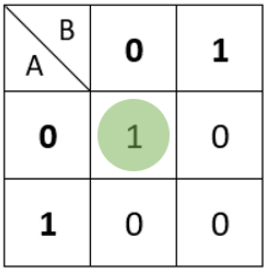
AND: AB



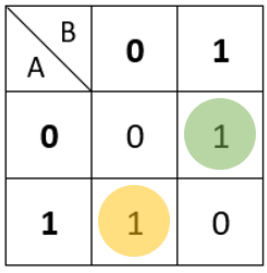
NAND: $\overline{A} + \overline{B}$



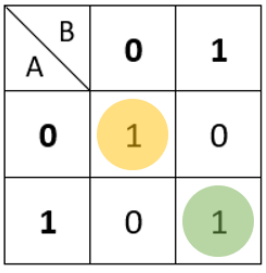
OR: $A + B$



NOR: $\overline{AB}$



XOR: $A\overline{B} + B\overline{A}$



XNOR: $\overline{AB} + AB$

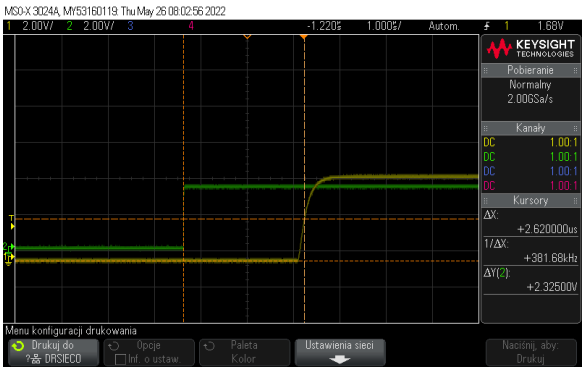
Widzimy, że bramki które posiadamy (**NOT**, **AND**, **OR**) wystarcza, aby zrealizować każdą inną bramkę. Zwróćmy uwagę, że wystarczyłoby użycie bramek **NOT**, **AND** lub samej bramki **NAND**.

3.2 Pomiar czasu propagacji

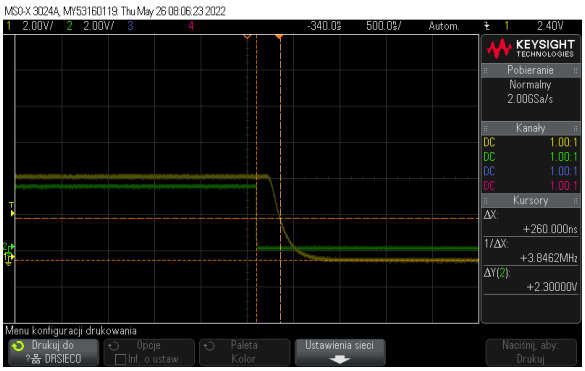
Na wejście bramki logicznej **OR** podaliśmy sygnał prostokątny o następujących parametrach:

- High level = 5 [V].
- Low level = 0 [V].

i zmierzyliśmy czasy przełączenia ze stanu niskiego do wysokiego T_{LH} oraz wysokiego do niskiego T_{HL}:



Rysunek 10: Pomiar czasu T_{LH}



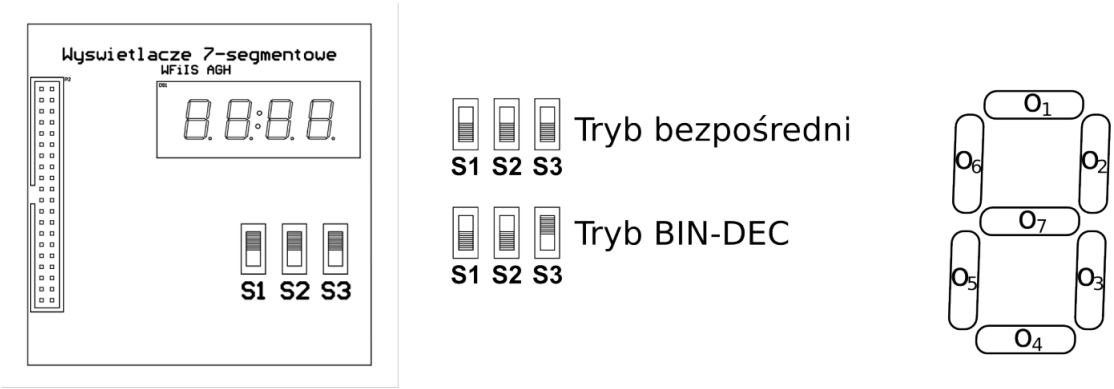
Rysunek 11: Pomiar czasu T_{HL}

Otrzymaliśmy następujące wartości:

- $T_{LH} = 2{,}62\ \mu s$.
- $T_{HL} = 260\ ns$.

Od razu wyciągamy ważny wniosek o tym, że czasy T_{LH} i T_{HL} nie muszą być równe.

3.3 Dekoder kodu binarnego na wyświetlacz 7-segmentowy (wersja 3)



Rysunek 12: Konfiguracja matrycy logicznej dla dekodera kodu binarnego

Naszym zadaniem było zbudowanie układu kombinacyjnego dekodującego liczby w systemie binarnym na system szesnastkowy poprzez przedstawienie liczby na wyświetlaczu siedmiosegmentowym (Rys.12).

W celu realizacji powyższego zadania najpierw sporządzimy następującą tabelę stanów:

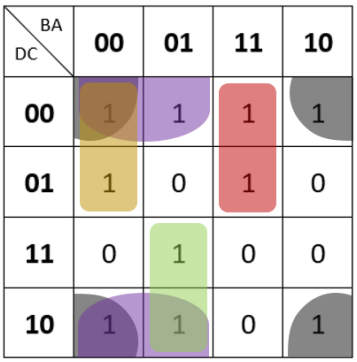
Tabela 2: Tabela stanów (dla słowa wejściowego DCBA przypisane są wartości wyjść dekodera)

	D	**C**	**B**	**A**	$\mathbf{O_1}$	$\mathbf{O_2}$	$\mathbf{O_3}$	$\mathbf{O_4}$	$\mathbf{O_5}$	$\mathbf{O_6}$	$\mathbf{O_7}$	$\mathbf{O_8}$
0	0	0	0	0	1	1	1	1	1	1	0	0
1	0	0	0	1	0	1	1	0	0	0	0	0
2	0	0	1	0	1	1	0	1	1	0	1	0
3	0	0	1	1	1	1	1	1	0	0	1	0
4	0	1	0	0	0	1	1	0	0	1	1	0
5	0	1	0	1	1	0	1	1	0	1	1	0
6	0	1	1	0	1	0	1	1	1	1	1	0
7	0	1	1	1	1	1	1	0	0	0	0	0
8	1	0	0	0	1	1	1	1	1	1	1	0
9	1	0	0	1	1	1	1	1	0	1	1	0
A	1	0	1	0	1	1	1	0	1	1	1	0
B	1	0	1	1	0	0	1	1	1	1	1	0
C	1	1	0	0	1	0	0	1	1	1	0	0
D	1	1	0	1	0	1	1	1	1	0	1	0
E	1	1	1	0	1	0	0	1	1	1	1	0
F	1	1	1	1	1	0	0	0	1	1	1	0

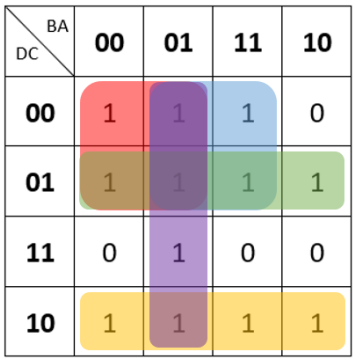
Zauważmy, że w przyjętej konwencji D jest najstarszym bitem a A - najmłodszym.

Następnie dokonaliśmy minimalizacji funkcji logicznych dla każdego z 7 wyjść O_1 - O_7 (O_8 pomijamy ze względu na to, że kolumna zawiera same zera) wykorzystując metodę tablic Karnaugh'a:

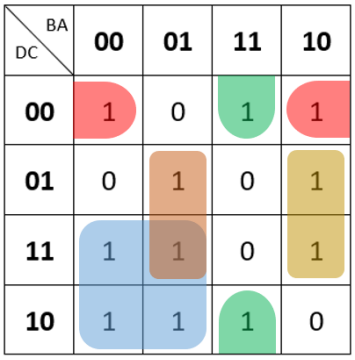
O_1



O_2

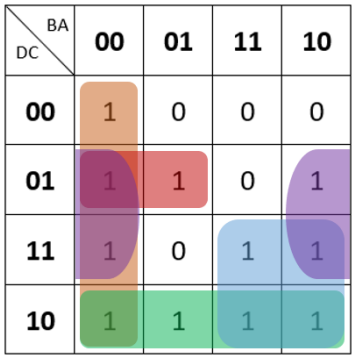


O_3

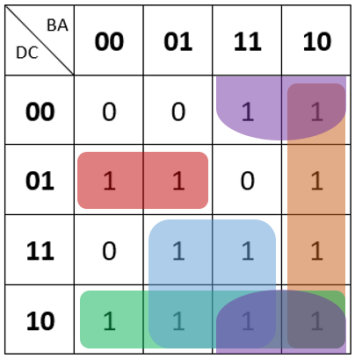


O_4

O_5



O_6



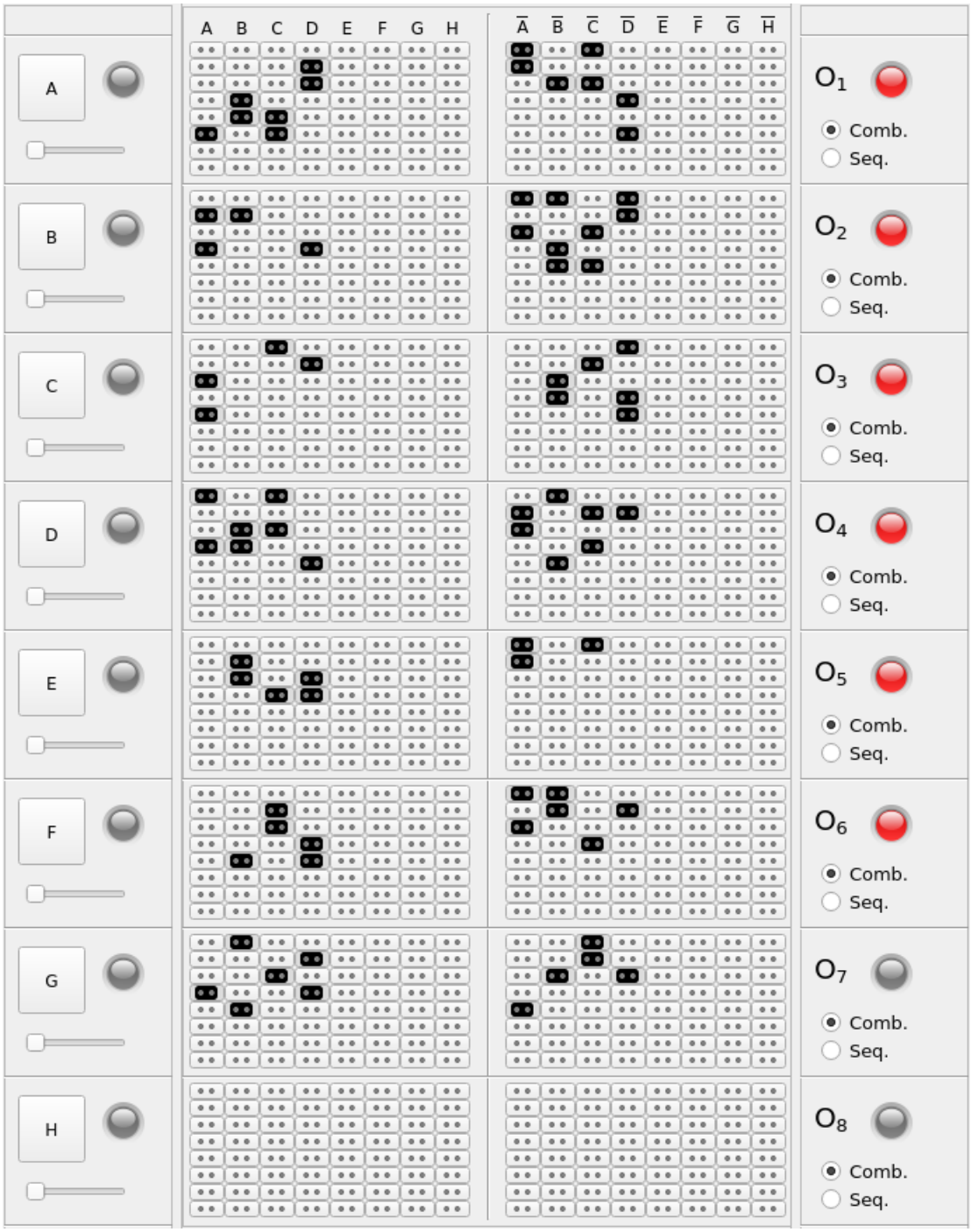
O_7

Teraz korzystając z powyższych map możemy zapisać zminimalizowane funkcje:

Wyjście	Funkcja
$\mathbf{O_1}$	$\overline{C}\overline{A} + D\overline{A} + D\overline{C}\overline{B} + \overline{D}B + CB + \overline{D}CA$
$\mathbf{O_2}$	$\overline{D}\overline{B}\overline{A} + \overline{D}BA + \overline{C}\overline{A} + D\overline{B}A + \overline{C}\overline{B}$
$\mathbf{O_3}$	$\overline{D}C + D\overline{C} + \overline{B}A + \overline{D}\overline{B} + \overline{D}A$
$\mathbf{O_4}$	$C\overline{B}A + \overline{D}\overline{C}\overline{A} + CB\overline{A} + \overline{C}BA + D\overline{B}$

Wyjście	Funkcja
$\mathbf{O_5}$	$\overline{C}\overline{A} + B\overline{A} + DB + DC$
$\mathbf{O_6}$	$\overline{B}\overline{A} + \overline{D}C\overline{B} + C\overline{A} + D\overline{C} + DB$
$\mathbf{O_7}$	$\overline{C}B + D\overline{C} + \overline{D}C\overline{B} + DA + B\overline{A}$

W celu wizualizacji konfiguracji matrycy i sprawdzenia jej poprawności dokonaliśmy symulacji w symulatorze PLD dostępnym na serwerze Taurus:



Rysunek 20: Konfiguracja matrycy logicznej dla dekodera kodu binarnego

Widzimy, jak dokonując prostych kroków możemy zbudować dekoder kodu binarnego na kod szesnastkowy.