Synteza układów kombinacyjnych metodą tablic Karnaugha - ćwiczenie 7

1. Cel ćwiczenia:

Celem ćwiczenia jest praktyczna realizacja układu kombinacyjnego na podstawie funkcji boolowskich wyznaczonych na zajęciach. Celem drugorzędnym jest nabycie umiejętności praktycznych w zakresie symulacji układów kombinacyjnych za pomocą programu LabVIEW.

2. Wprowadzenie teoretyczne

2.1. Metoda tablic Karnaugha

Metoda tablicy Karnaugha należy do grupy najszybszych metod minimalizacji funkcji przełączających małej liczby zmiennych, co wynika z dużej komplikacji samego zapisu następującej wraz ze wzrostem ilości zmiennych.

Upraszczając funkcję przełączającą przy wykorzystaniu tablicy Karnaugha, należy pamiętać o następujących problemach:

a) wiersze i kolumny tablicy Karnaugha opisane są w kodzie Greya, tzn. każdy kolejny wiersz i kolumna różnią się od siebie o negację jednej zmiennej,
b) zakreślając jedynki (zera), tworzy się grupy liczące 2, 4, 8, 16 ... elementów,
c) zawsze zakreśla się grupy z największą możliwą ilością jedynek (zer), przy czym należy pamiętać o możliwości sklejenia ze sobą krawędzi równoległych tablicy,
d) grupy mogą posiadać części wspólne,
e) liczba grup jedynek (zer) odpowiada liczbie składników sumy (iloczynu) poszukiwanej funkcji,
f) w przypadku kiedy istnieje możliwość zakreślenia grup na kilka sposobów, arbitralnie wybiera się jeden z nich,
g) dana grupa reprezentuje iloczyn (sumę) tych zmiennych, które nie zmieniają swojej wartości,
h) w przypadku gdy funkcja przełączająca posiada elementy o wartości nieokreślonej elementy te wpisujemy do tabeli wprowadzając dla nich specjalne oznaczenie np. – a następnie wykorzystujemy lub pomijamy w zależności od potrzeby przy tworzeniu grup (patrz punkt b).

Przykład
Zminimalizować funkcję $y = \sum(0, 2, 4, 6, 12, 14)_{x_1 x_2 x_3 x_4}$ metodą tablicy Karnaugha.

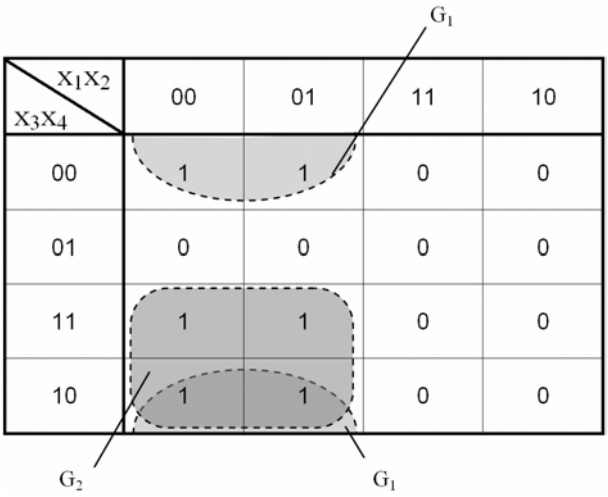
Rozwiązanie
Po rozwinięciu otrzymamy funkcję przełączającą w postaci:

$$y = \bar{x}_1\bar{x}_2\bar{x}_3\bar{x}_4 + \bar{x}_1 x_2\bar{x}_3\bar{x}_4 + \bar{x}_1\bar{x}_2 x_3\bar{x}_4 + \bar{x}_1 x_2 x_3\bar{x}_4 + \bar{x}_1\bar{x}_2 x_3 x_4 + \bar{x}_1 x_2 x_3 x_4$$

Stosując zasadę a) tworzymy tablicę Karnaugha, wypełniając ją jedynkami dla elementów funkcji a pozostałe pola uzupełniamy zerami otrzymując:

x_3x_4 \ x_1x_2	00	01	11	10
00	1	1	0	0
01	0	0	0	0
11	1	1	0	0
10	1	1	0	0

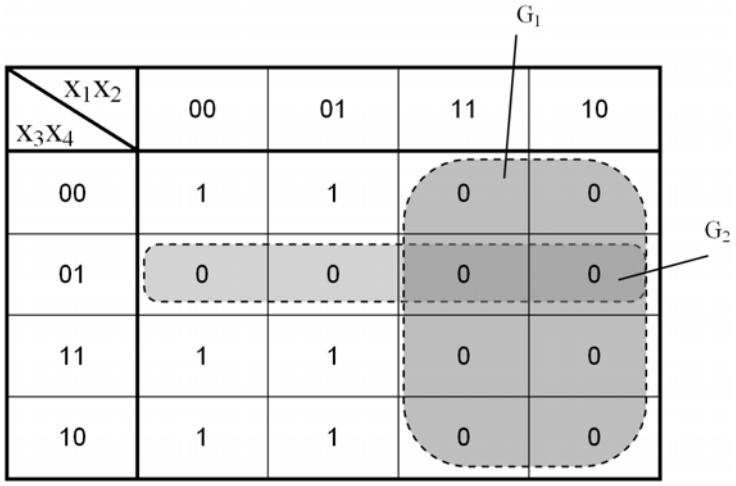
W kolejnym kroku tworzymy dwie grupy zawierające po cztery elementy - według zasady b), c), d):



Postępując według wytycznych e), f), g) odczytujemy zminimalizowaną postać funkcji przełączającej. W czteroelementowej grupie G_1 wartości zmiennych $\bar{x}_1$ i $\bar{x}_4$ nie ulegają zmianie ponieważ zakreślono grupę jedynek. Funkcja ta przyjmie postać iloczynu $G_1 = \bar{x}_1\bar{x}_4$. Dla grupy G_2 niezmienne wartości przyjmują parametry $\bar{x}_1$, x_3 więc grupa ta przyjmie postać $G_1 = \bar{x}_1 x_3$. Po zsumowaniu z grupą G_1 otrzymamy ostatecznie poszukiwaną przez nas funkcję w postaci dysjunkcyjnej:

$$y = \bar{x}_1 x_3 + \bar{x}_1\bar{x}_4$$

W celu wyznaczenia zminimalizowanej funkcji w postaci koniunkcyjnej należy zacząć od początku wypisywanie tablicy dla $y = \Pi(0,2,4,6,12,14)_{x_1x_2x_3x_4}$ lub też skorzystać z tablicy wypisanej dla postaci dysjunkcyjnej zakreślając w tym przypadku grupy zer:



Postępując według wytycznych e), f), g) otrzymamy następujące grupy $G_1 = \bar{x}_1$ (ponieważ jedynym niezmiennym parametrem w grupie jest x_1 i przyjmuje on wartość jeden) oraz $G_2 = x_3 + \bar{x}_4$. Po pomnożeniu obu grup otrzymamy zminimalizowaną funkcję w postaci koniunkcyjnej:

$$y = \bar{x}_1(x_3 + \bar{x}_4)$$

2.2. LabVIEW

LabVIEW (**Laboratory Virtual Instrument Engineering Workbench)** umożliwia tworzenie programów za pomocą języka graficznego (tzw. język G).
Programowanie w LabVIEW polega na budowie schematu blokowego i korespondującego z nim panelu stanowiącego interfejs użytkownika. Budowa tego interfejsu jest możliwa dzięki dostępnym bibliotekom gotowych elementów takich, jak: wyświetlacze cyfrowe, mierniki, potencjometry, termometry, diody LED, tabele, wykresy itp. Elementy te konfiguruje się w zależności od zastosowania. Panel użytkownika umożliwia zbudowanie wirtualnego przyrządu obsługiwanego: z klawiatury, za pomocą myszy lub innego urządzenia wejściowego służącego do komunikacji komputera z użytkownikiem.

Następnie, przy pomocy graficznego języka konstruuje się odpowiedni schemat blokowy, będący równocześnie kodem źródłowym. Budowany schemat blokowy można porównać z grafem przepływu informacji, a jego elementy to funkcje zawarte w bibliotekach, np. algebraiczne, boolowskie, statystyczne, związane z obsługą plików, przetwarzaniem sygnałów lub obsługą urządzeń we/wy itp. Relacje między blokami funkcyjnymi reprezentowane są przez połączenia o różnych kolorach i grubościach. Rodzaj połączenia świadczy o typie przekazywanych danych. Można łączyć ze sobą tylko elementy tego samego typu. Tworzone aplikacje nazywane są virtual instruments (VI), ponieważ ich wygląd i operacje imitują działanie rzeczywistych przyrządów. Program zawiera wszystkie narzędzia niezbędne do akwizycji, analizy i prezentacji danych.

Wszystkie aplikacje używają struktury hierarchicznej i modularnej. Oznacza to, że można ich używać również jako podprogramy. Aplikacje użyte w innej aplikacji nazywane są subVI.

3. Przebieg ćwiczenia

3.1. Synteza układu kombinacyjnego na przykładzie dekodera kodu naturalnego na kod 1 z 4.

Pierwszym krokiem jest wyznaczenie funkcji boolowskich dla zadanego układu. W tym celu przedstawiamy działanie zadanego układu za pomocą tabeli stanów.

wejścia				wyjścia			
b_1	b_2	$\overline{b}_1$	$\overline{b}_2$	d_1	d_2	d_3	d_4
0	0	1	1	1	0	0	0
0	1	1	0	0	0	1	0
1	0	0	1	0	1	0	0
1	1	0	0	0	0	0	1

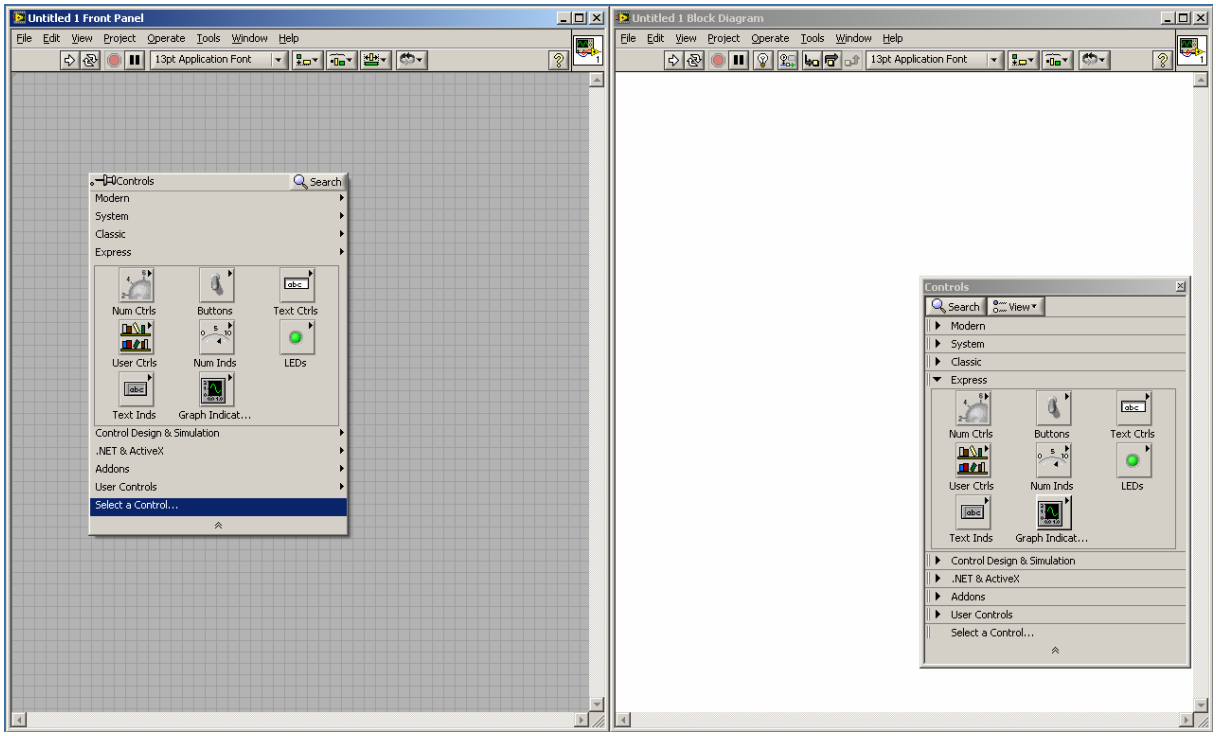
Jeżeli jest to możliwe, zapisujemy równania bezpośrednio z tabeli. W tym celu wygodnie jest w tabeli stanów uwzględnić negację sygnałów wejściowych.

$d_1 =$
$d_2 =$
$d_3 =$
$d_4 = b_1 \cdot b_2$

Symulacja dekodera kodu naturalnego na kod 1 z 4 za pomocą programu LabWIEV.
Uruchamiamy program z menu Start > Programy > National Instruments LabVIEW 8.x.
Po uruchomieniu wybieramy opcję *new* i po rozwinięciu klikamy *BlankVI.*

W celu optymalnego rozmieszczenia okien programu na monitorze komputera, należy w oknie o tle szarym, (jest to okno gdzie będzie budowany panel sterujący) wybrać w górnym pasku narzędzi opcję *Window>Tile Left and Right.*

Okno Panelu użytkownika *Okno schematu blokowego*

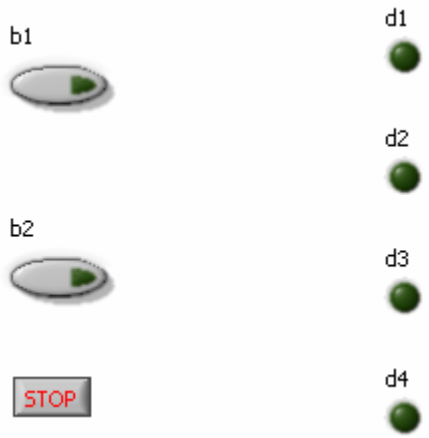


Rys.1. Okna programu LabVIEW

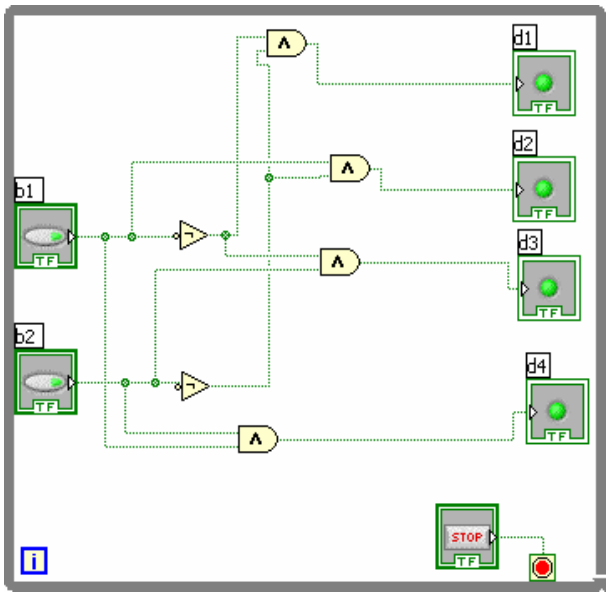
Okno Block Diagram służy do tworzenia schematu blokowego w celu realizacji dekodera – umieszczamy tu bramki, wejścia, wyjścia itp. Okno Front Panel posłuży do wizualizacji i kontroli napisanego programu umieścimy tu wszystkie przyciski (wejścia) i diody (wyjścia).
W celu zbudowania dekodera umieszczamy dwa przyciski oraz cztery diody. Nadajemy im nazwy b_1, b_2 dla przycisków i d_1 do d_4 dla diod. Klikamy prawym przyciskiem myszy w szare pole i najeżdżamy kursorem na ikonę *Leds.* Następnie wybieramy *Round LED* i w ten sposób tworzymy 4 diody. Dalej tworzymy 2 przyciski klikając prawym przyciskiem myszy na szarym polu oraz kursorem najeżdżamy na ikonę *Buttons* i wybieramy *PushButton.* Należy zauważyć, że wszystkie elementy panelu użytkownika mają swoje odpowiedniki w oknie schematu blokowego.
Kolejnym etapem jest budowa schematu blokowego w oknie Block Diagram.
Dla przykładu przedstawiono odzwierciedlenie pierwszego równania realizującego funkcję dla diody numer 1.

W celu zapewnienia cyklicznej pracy programu w oknie schematu blokowego wstawiamy pętlę *While Loop.* W oknie schematu blokowego klikamy prawym przyciskiem myszy na białym polu i wybieramy ikonę *Exec Ctrl>While Loop.* Trzymając lewy przycisk myszy tworzymy pole działania (rysujemy prostokąt) tak, aby objąć nim wszystkie wstawione do tej pory ikony. Następnie znów klikamy prawym przyciskiem myszy (wewnątrz pętli *While Loop*) i wybieramy ikonę *Not* z menu *Programming >Boolean* oraz ikonę *And* z tego samego menu.

W kolejnym kroku łączymy elementy schematu blokowego. Myszką najeżdżamy na prawą krawędź przycisku b_1, gdzie znajduje się mały trójkącik (wyjście przycisku). Pojawia się mała rolka – klikamy lewy przycisk myszy i trzymając łączymy z wejściem bramki *Not*. Dalej wyjście bramki *Not* łączymy z pierwszym wejściem bramki *And* i wyjście tej bramki do podłączamy do diody d_1. Następnie wstawiamy drugą bramkę *Not* z menu *Programming >Boolean*. Drugi przycisk b_2 łączymy z wejściem drugiej bramki *Not,* a wyjście tej bramki z drugim (wolnym) wejściem bramki *And*. W ten sposób zrealizowano funkcję d_1. Podobnie należy zbudować funkcje d_2 do d_4. Sposób realizacji całego dekodera przedstawiono na rysunkach 2 i 3.



Rys.2. Panel sterowania



Rys.3. Schemat blokowy budowanego dekodera

Tak zbudowany program będzie działał cyklicznie, tzn. pętla *While Loop* będzie ponownie uruchamiana zaraz po zakończeniu poprzedniego cyklu. Powoduje to spowolnienie pracy komputera, gdyż jednostka centralne stara się jak najczęściej wykonywać pętlę. W celu zmniejszenia liczby wykonywanych cykli do istniejącego schematu blokowego dodajemy opóźnienie. Z menu *Programming >Timing* wybieramy funkcję *Wait Until Next ms Multiple* i umieszczamy ją wewnątrz pętli. Definicji czasu oczekiwania dokonujemy poprzez kliknięcie prawym przyciskiem myszy nad wejściem funkcji (lewa krawędź) i wybraniu opcji *Constant* z podręcznego menu *Create*. W powstałe pole należy wpisać czas oczekiwania w mili sekundach np. 10 ms.

Ostatnim krokiem jest sprawdzenie, czy zbudowany układ działa poprawnie. W tym celu należy go uruchomić. W oknie Panelu sterowania klikamy ikonę strzałki *Run (Ctrl+R)*, która znajduje się pod górnym paskiem menu. Sprawdzamy poprawność zbudowanego układu wykorzystując wszystkie możliwości przycisków włączając je i wyłączając. Po sprawdzeniu naciskamy przycisk STOP.

W ten oto sposób możemy zbudować dowolny dekoder kodu naturalnego 1 z *n*.

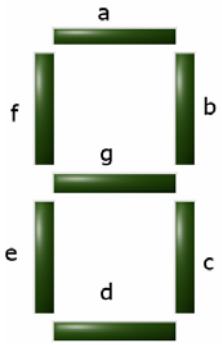
3.2. Synteza transkodera kodu binarnego na kod wskaźnika siedmiosegmentowego.

Transkoder ten ma 4 wejścia (A, B, C, D) reprezentujące wyświetlaną w kodzie binarnym cyfrę. Siedem wyjść (a, b, c, d, e, f, g) służy do sterowania poszczególnymi segmentami wskaźnika. Każde z wyjść może być w jednym z dwóch stanów: „włączony” (świeci) i „wyłączony” (nie świeci). Liczba możliwych kombinacji logicznych stanów wyjściowych to 2^7=128. Liczba kombinacji stanów wejściowych jest równa 16, z których tylko 10 jest wykorzystanych do kodowania dziesięciu cyfr (0..9), a 6 pozostałych kombinacji może służyć do kodowania dodatkowych znaków. Sposób rozmieszczenia segmentów wskaźnika przedstawiono na rysunku 4.
Zadany transkoder ma współpracować ze wskaźnikiem o wspólnej anodzie. Zatem, w celu zaświecenia danego segmentu należy podać stan niski. Przykładowo – w celu wyświetlenia cyfry „0” należy zapalić wszystkie segmenty z wyjątkiem segmentu „g”. Na wyjścia **a, b, c, d, e, f,** podajemy „0” logiczne, natomiast wyjście **g** ustawiamy na poziomie wysokim „1”.

Tabela stanów transkodera

Wartość dziesiętna	Stany wejść				Stany wyjść						
	D	C	B	A	a	b	c	d	e	f	g
0	0	0	0	0	0	0	0	0	0	0	1
1	0	0	0	1	1	0	0	1	1	1	1
2	0	0	1	0	0	0	1	0	0	1	0
3	0	0	1	1	0	0	0	0	1	1	0
4	0	1	0	0	1	0	0	1	1	0	0
5	0	1	0	1	0	1	0	0	1	0	0
6	0	1	1	0	0	1	0	0	0	0	0
7	0	1	1	1	0	0	0	1	1	1	1
8	1	0	0	0	0	0	0	0	0	0	0
9	1	0	0	1	0	0	0	0	1	0	0
10	1	0	1	0	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ
11	1	0	1	1	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ
12	1	1	0	0	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ
13	1	1	0	1	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ
14	1	1	1	0	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ
15	1	1	1	1	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ

ϕ – oznacza dowolny stan



Rys.4. Rozkład segmentów dla wskaźnika siedmiosegmentowego

Kolejnym krokiem jest minimalizacja funkcji dla poszczególnych wyjść **a** do **g**. W tym celu dla każdego z wyjść wypełniamy tabelę Karnaugha.

Uzupełnimy tabele:

DC \ BA	00	01	11	10
00	0	1	0	0
01	1	0	0	0
11	ϕ	ϕ	ϕ	ϕ
10	0	0	ϕ	ϕ

a = $\overline{A}\,\overline{B}\,C + A\,\overline{B}\,\overline{C}\,\overline{D}$

DC \ BA	00	01	11	10
00				
01				
11	ϕ	ϕ	ϕ	ϕ
10			ϕ	ϕ

b =

DC \ BA	00	01	11	10
00				
01				
11	ϕ	ϕ	ϕ	ϕ
10			ϕ	ϕ

c =

DC \ BA	00	01	11	10
00				
01				
11	ϕ	ϕ	ϕ	ϕ
10			ϕ	ϕ

d =

DC \ BA	00	01	11	10
00				
01				
11	ϕ	ϕ	ϕ	ϕ
10			ϕ	ϕ

e =

DC \ BA	00	01	11	10
00				
01				
11	ϕ	ϕ	ϕ	ϕ
10			ϕ	ϕ

f =

DC \ BA	00	01	11	10
00				
01				
11	ϕ	ϕ	ϕ	ϕ
10			ϕ	ϕ

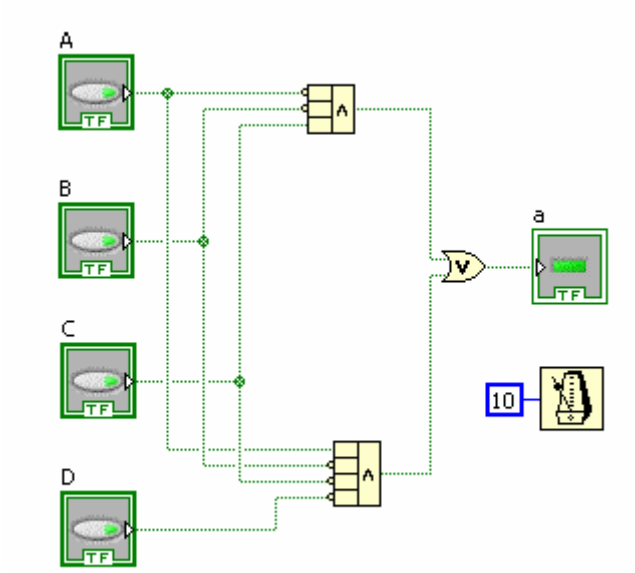
g =

Po tym etapie zadania przechodzimy do budowy układów za pomocą programu LabVIEW.

Budowy układu transkodera dokonujemy podobnie jak układu dekodera, omawianego poprzednio. W oknie panelu tworzymy wyświetlacz siedmiosegmentowy. Do tego celu używamy diod LED. Po kliknięciu prawym przyciskiem myszy w szare pole panelu frontowego należy wybrać kursorem ikonę *Leds,* a następnie *Square LEDs*. Wybrana dioda będzie stanowić jeden z segmentów wskaźnika siedmiosegmentowego. Należy tu pamiętać o odpowiedniej konfiguracji diody (poprzez kliknięcie na diodzie prawym przyciskiem myszy i wybranie opcji *properties*). Zgodnie z założeniem o wspólnej anodzie wskaźnika, segmenty powinny „świecić" (mieć odpowiedni kolor) gdy na wejściu jest poziom niski i nie świecić (mieć kolor tła) gdy na wejściu jest poziom wysoki. Diodę kopiujemy i wklejamy sześć razy w celu uzyskania siedmiu takich samych segmentów, które układamy tak – aby tworzyły cyfrę „osiem". Segmentom nadajemy nazwy **a, ..., g**. Po zbudowaniu „ósemki" dodajemy 4 przyciski *PushButton* i nazywamy je odpowiednio **A, B, C, D**.

W programie LabVIEW możliwe jest stosowanie wielowejściowych funkcji logicznych nazywanych *Compound Arithmetic* z menu *Programming >Boolean.* Omawiany bloczek w zależności od potrzeb można rozszerzyć na odpowiednią ilość wejść. W celu dokonania tej operacji kursorem myszy najeżdżamy na dolną krawędź bloczka i rozszerzamy go Funkcje *Compound Arithmetic* umożliwiają, również negację poszczególnych wejść poprzez wybór opcji *Invert* z menu podręcznego dla danego wejścia funkcji. Lewa strona bloczka *Compound Arithmetic* podzielona jest na obszary (liczba obszarów zależy od liczby wejść), które możemy zanegować w zależności od potrzeb programu. Kolejnym krokiem jest wybór realizowanej funkcji przez *Coumpound Arithmetic.* Wyboru tej funkcji dokonujemy z menu podręcznego (klikając prawym przyciskiem myszy ponad ikoną) *Change Mode>AND*. Przykład realizacji funkcji dla segmentu przedstawiono na rysunku 5.

$$a = \overline{A}\,\overline{B}\,C + A\,\overline{B}\,\overline{C}\,\overline{D}$$



Rys. 5. Przykład realizacji zminimalizowanej funkcji

W podobny sposób budujemy pozostałe zminimalizowane funkcje.

4. Sprawozdanie

Sprawozdanie powinno zawierać: schematy symulacyjne zbudowanych układów, tablice stanów, tablice Karnaugha w postaci dysjunkcyjnej, funkcje po zminimalizowaniu w postaci dysjunkcyjnej oraz wnioski końcowe. Należy również wyznaczyć postać koniunkcyjną zadanych na zajęciach (dwóch) funkcji, korzystając z tablicy zapisanej dla postaci dysjunkcyjnej. Należy tu zapisać funkcje wykorzystując grupy zer.