UNIWERSYTET ŚLĄSKI

Projektowanie układów kombinacyjnych

Zawartość

1 Teoria

Podstawowe twierdzenia i własności algebry Boole'a:

Postulat 2	$x+0=x$	$x\cdot 1=x$
Postulat 5	$x+\bar{x}=1$	$x\cdot\bar{x}=0$
Twierdzenie 1	$x+x=x$	$x\cdot x=x$
Twierdzenie 2	$x+1=1$	$x\cdot 0=0$
Twierdzenie 3 – INWOLUCJA	$\bar{\bar{x}}=x$	
Postulat 3 – PRZEMIENNOŚĆ	$x+y=y+x$	$x\cdot y=y\cdot x$
Twierdzenie 4 – ŁĄCZNOŚĆ	$x+(y+z)=(x+y)+z$	$x\cdot(y\cdot z)=(x\cdot y)\cdot z$
Postulat 4 – ROZDZIELNOŚĆ	$x\cdot(y+z)=xy+xz$	$x+(y\cdot z)=(x+y)(x+z)$
Twierdzenie 5 – PRAWA DE MORGANA	$\overline{(x+y)}=\bar{x}\cdot\bar{y}$	$\overline{(x\cdot y)}=\bar{x}+\bar{y}$
Twierdzenie 6 – ABSORPCJA	$x+xy=x$	$x\cdot(x+y)=x$

Przykłady minimalizacji funkcji:

1) stosowanie „sztuczek”

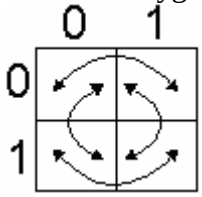
$$xy+\bar{x}z+yz=$$
$$xy+\bar{x}z+yz(x+\bar{x})=$$
$$xy(1+z)+\bar{x}z+\bar{x}yz=$$
$$xy+\bar{x}z(1+y)=$$
$$xy+\bar{x}z$$

2) zastosowania praw De Morgana

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

Metoda Karnaugh to sposób minimalizacji funkcji boolowskich. Został odkryty w 1950 roku przez Maurice Karnaugha. W ogólnym przypadku znalezienie formuły minimalnej dla zadanej funkcji boolowskiej jest bardzo skomplikowanym problemem. Jednak jeśli funkcja ma małą liczbę zmiennych (do sześciu) i zostanie zapisana w specjalnej tablicy zwanej tablicą Karnaugh, wówczas znalezienie minimalnej formuły odbywa się na drodze intuicyjnej.

Siatka Karnaugh'a dla automatu czterech stanów wygląda następująco:



Nie stanowi to jeszcze żadnej filozofii. Automat poruszając się wewnątrz po polach siatki zmienia stan tylko na jednym bicie. Automat nie może poruszać się po skosie. Jedynie przeskakiwać do sąsiednich kratek.

Powyższy opis siatki odpowiada takiemu zakodowaniu pól stanów:

a \ b	0	1
0	00	01
1	10	11

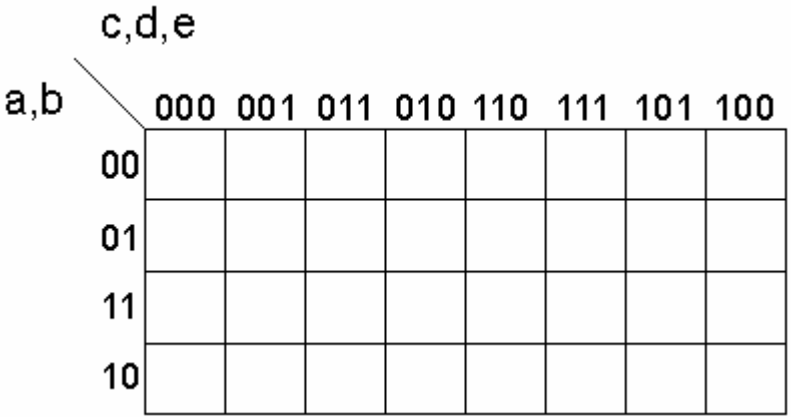
Litery „a" i „b" – to nazwy zmiennych logicznych (czyli po prostu sygnałów wejściowych układu). Stany zakodowano czytając najpierw współrzędną „a", następnie współrzędną „b".

Zwróć uwagę, że przeskakując po sąsiednich kratkach poruszamy się w ten sposób po sąsiadujących ze sobą stanach kodu Graya, a więc zmieniamy kod stanu zawsze tylko na 1 pozycji (a przecież o to nam chodzi).

W przypadku, gdy zmiennych logicznych mamy 3 (3 sygnały wejściowe), mamy 8 możliwych kombinacji stanów. Siatka Karnaugh'a dla układu z trzema sygnałami wejściowymi wygląda tak:

A \ B,C	00	01	11	10
0				
1				

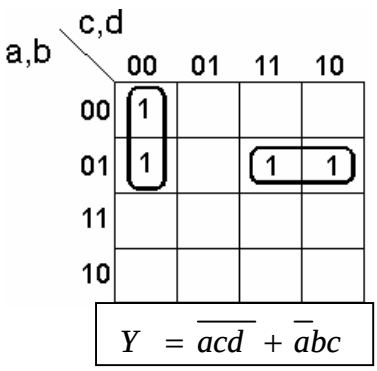
Siatka Karnaugha dla układu pięciu zmiennych wygląda następująco:



Gdyby ująć w kilka zasad projektowanie przy pomocy siatek Karnaugh'a można by przedstawić je następująco:

1) Tworzymy siatki o rozmiarze takim by ilość opisujących je zmiennych była równa ilości sygnałów wejściowych.
2) Tworzymy tyle siatek, ile układ posiada wyjść.
3) Wewnątrz kratek danej siatki wpisujemy stany wewnętrzne jakie mają pojawiać się na danym wyjściu przy odpowiadającej im kombinacji sygnałów wejściowych opisujących siatkę.
4) Te trzy zasady mówią nam jak zbudować siatki i jak zakodować w nich stany. Ale to dopiero połowa projektu. Kiedy już to zrobimy – czeka nas ciekawsze zadanie. Na podstawie zakodowanych odpowiednio stanów trzeba stworzyć funkcję logiczną. Mając ją w postaci równania potrafimy już zbudować układ logiczny.
5) Teraz jednak zajmijmy się uzyskiwaniem funkcji logicznej na podstawie zakodowanych w tablicy Karnaugh'a stanów.

Zakodowane w siatce stany zakreślamy w grupy. W zależności od tego czy zakreślamy grupy jedynek czy grupy zer – otrzymujemy funkcję logiczną w postaci sumy iloczynów lub iloczynu sum.



Obok zostało przedstawione przykładowe zakreślenie dwóch grup jedynkowych.

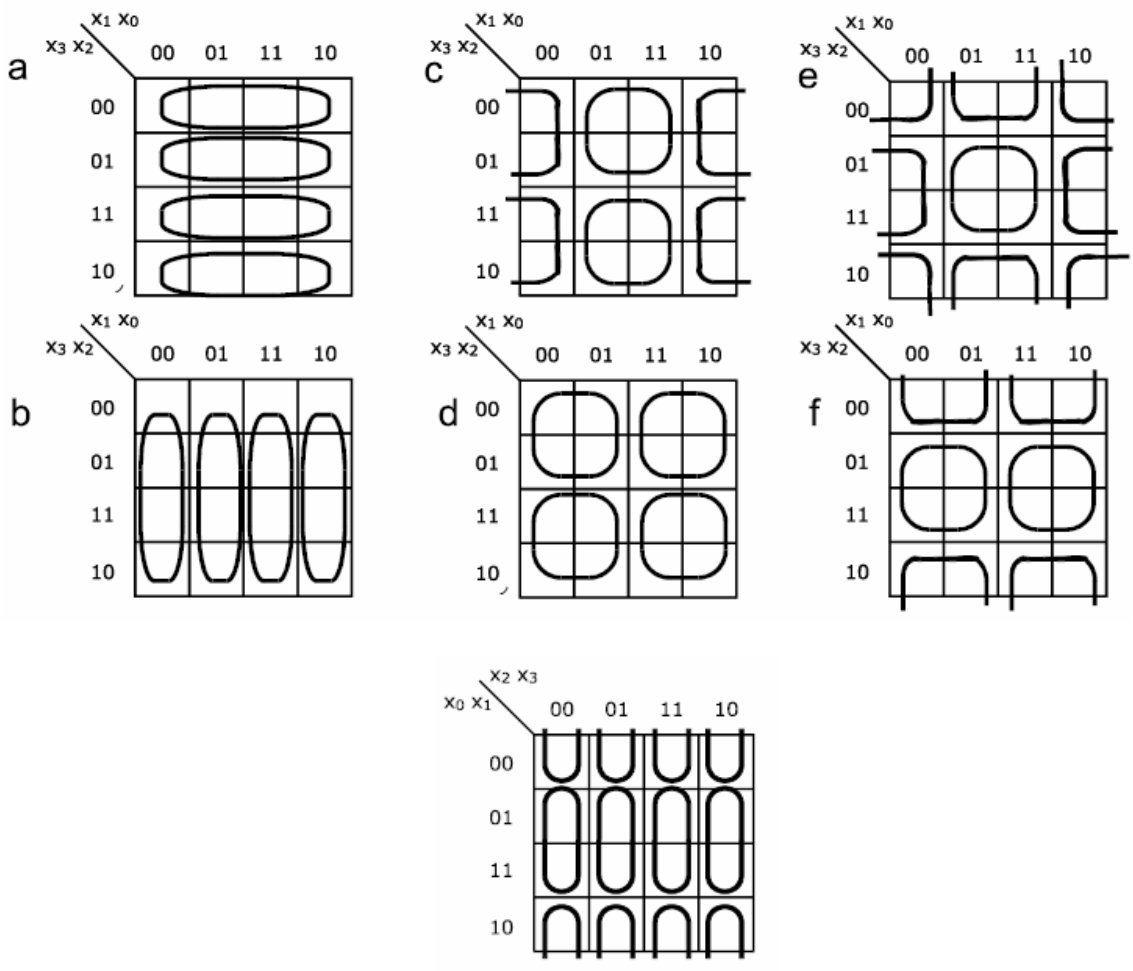
Jedynki wpisane zostają w te miejsca siatki, w których dla sygnałów wejściowych (a,b,c,d) ma występować jedynka na wyjściu „y".

Wyżej opisana jest funkcja wyjściowa. Robi się to w ten sposób, że patrzy się na współrzędne zakreślonej grupy i do wyrażenia iloczynowego wchodzą tylko te zmienne, które w obrębie grupy nie zmieniają swojej wartości z 0 na 1 (lub z 1 na 0).

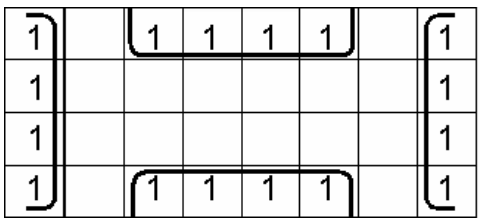
Nie wolno nam jednak zakreślać grup w dowolny sposób. Aby układ miał szansę działać poprawnie musimy pilnować następujących reguł:

1) Ilość zakreślonych w jednej grupie kratek musi wynosić zawsze 2^k (dwa do k-tej) czyli wolno zakreślać tylko 1, 2, 4, 8, 16, 32 jedynki.... nie możemy w jedną grupę objąć np. 6.
2) Zakreślona grupa musi być symetryczna względem którejś z głównych (bądź podgłównych) osi siatki Karnaug'a.

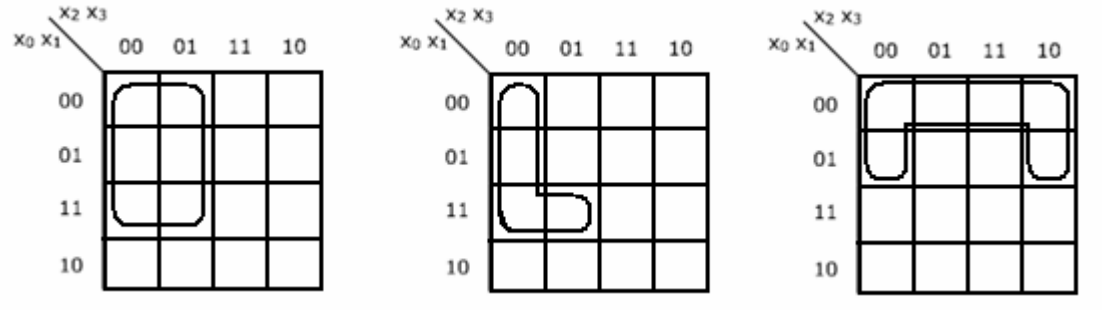
Drugi punkt wymaga dodatkowego wyjaśnienia. Bez niego nie będziemy w stanie robić tego poprawnie. Najlepszym wyjaśnieniem będzie kilka zakreślonych poprawnie kratek;



Rysunek 1. Przykładowe poprawne zakreślenia grup w siatkach dla 4 zmiennych.



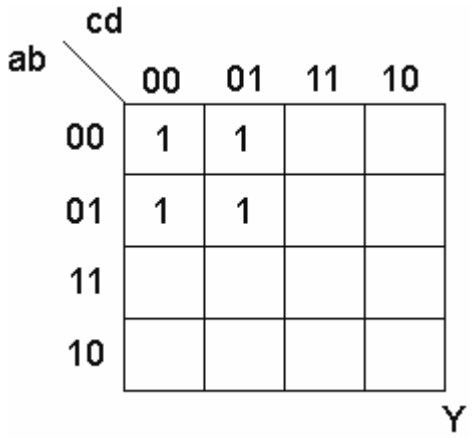
Rysunek 2. Przykładowe poprawne zakreślenie grup w siatce 5 zmiennych.



Rysunek 3. Przykładowe niepoprawne zakreślenie grup w siatce 4 zmiennych.

Kolejną zasadą zakreślania grup jest to aby tworzyć grupy o jak największej ilości jedynek (lub zer). Robiąc to pamiętajmy jednak o pierwszym warunku mówiącym o ilości zakreślonych kratek. Im liczniejszą grupę zakreślimy – tym mniej zmiennych (zgodnie z zasadą tworzenia funkcji) wejdzie do wyrażenie logicznego opisującego wyjście układu.

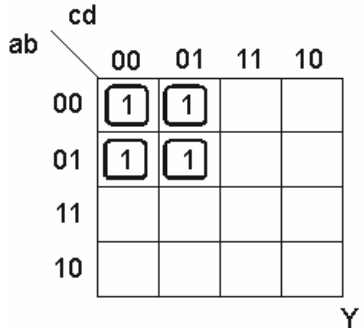
Aby lepiej wyjaśnić zasady tworzenia wyrażenia logicznego opisującego wyjście na podstawie zakreślonych wewnątrz siatki grup, przyjrzyjmy się następującej siatce:



Rysunek 4. Przykładowa siatka 4 zmiennych z wpisanymi wartościami wyjścia dla określonych kombinacji sygnałów na wejściu.

Innymi słowy na wyjściu Y układu kombinacyjnego opisanego taką siatką pojawią się jedynki gdy na wejściach będą kombinacja (dla a,b,c,d): 0000, 0001, 0100, 0101.

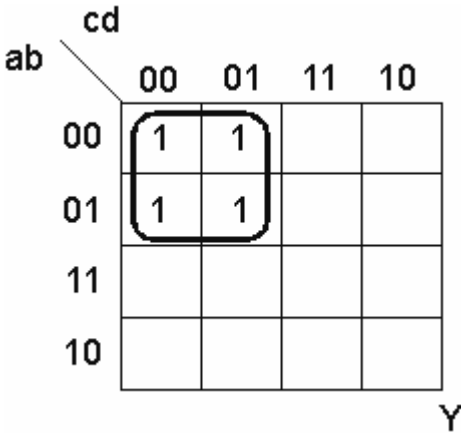
Układ ten możemy zbudować zakreślając jedynki w grupy na różne sposoby. Najgorszy z możliwych ruchów wykonamy, gdy zrobimy to w ten sposób (choć zaznaczenie będzie poprawne):



Zakreśliliśmy w ten sposób 4 grupy. Są one poprawne i formalnie układ mógłby działać poprawnie. Skoro jednak mamy 4 implikanty, to funkcja logiczna opisująca wyjście Y musi posiadać 4 składniki iloczynowe, a ponieważ w obszarze żadnego zakreślonego implikantu nie zmienia się ani jedna zmienna wejściowa (a,b,c,d) do każdego składniku iloczynowego wejdą wszystkie cztery. W ten sposób uzyskana funkcja wygląda następująco: $Y = \overline{abcd} + \overline{abcd} + \overline{abcd} + \overline{abcd}$

Zbudowany w oparciu o tę funkcję układ logiczny potrzebowałby czterech wejść, 5 bramek negacji, czterech cztero-wejściowych iloczynów oraz jedną cztero-wejściową sumę. Jak na swoją klasę byłby drogi i zbyt skomplikowany. Pomijając inne czynniki niemal uniemożliwiające jego działanie (które opanujemy nieco później) – byłby niezbyt opłacalny.

Znając jednak zasady poprawnego zakreślania grup możemy ten sam układ zbudować najbardziej poprawnie. Zastanówmy się jak wyglądałyby sprawy, gdybyśmy zakreślili funkcję w następujący sposób:



Tak zakreślona grupa jest poprawna (liczba jedynek i osie symetrii się zgadzają) oraz optymalny. Nie da się zrealizować tego układu prościej. Opisująca go funkcja dla tego zakreślenia wygląda następująco;

$$Y = \overline{ac}$$

Jak więc widać jest nieporównywalnie prostsza niż poprzednia. Zbudowany tak układ można by zrealizować nawet na jednej bramce NOR.

Przykład minimalizacji

Weźmy funkcję: f(x1,x2,x3,x4) = Σ[2,3,6,7,8,10,11,15,(0,13)]

x1 x2 \ x3 x4	00	01	11	10
00	0000	0001	0011	0010
01	0100	0101	0111	0110
11	1100	1101	1111	1110
10	1000	1001	1011	1010

Każda kratka tablicy odpowiada jednemu, konkretnemu wektorowi zmiennych binarnych. W kratkach zapisywane są wartości funkcji dla odpowiadających im wektorów.

Tabela pawdy dla funkcji f wygląda następująco:

x1	x2	x3	x4	f
0	0	0	0	-
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	-
1	1	1	0	0
1	1	1	1	1

Tworzymy tablicę Karnaugha przypisując zmienne x1, x2 wierszom a zmienne x3, x4 kolumnom. Następnie sklejamy ze sobą jak największe grupy jedynek i kresek tak, by każda

jedynka z tablicy znalazła się choć raz w jednej z grup. Niektóre grupy będą po prostu pojedynczymi kratkami - w nich znajdują się jedynki których nie można skleić wg poniższych reguł:

- sklejamy kratki bezpośrednio stykające się ze sobą np.: 2-3-6-7 lub 13-15; przy czym traktujemy brzegi tablicy Karnaugh jako stykające się ze sobą np.: 3-11 lub 0-2-8-10.
- sklejony obszar musi być prostokątem o bokach będących potęgami 2; należy pamiętać że brzegi stykają się ze sobą, dlatego sklejone np. cztery rogi tablicy tworzą kwadrat o boku 2

Przykładami tabel uzyskanych wg powyższego przepisu są tabele 3 i 4. Warto zauważyć, że kratka 13 (kreska nie obwiedziona żadnym kolorem) nie musi być w żadnej grupie, a jej włączenie spowodowałoby niepotrzebny przyrost ilości grup.

x1 x2 \ x3 x4	00	01	11	10
00	-	0	1	1
01	0	0	1	1
11	0	-	1	0
10	1	0	1	1

Rysunek 5. Tabela nieoptymalna Kratki 8 i 10 (niebieskie) tworzą jeden obszar, zostały sklejone poprzez brzeg tabeli.

x1 x2 \ x3 x4	00	01	11	10
00	-	0	1	1
01	0	0	1	1
11	0	-	1	0
10	1	0	1	1

Rysunek 6. Tabela nieoptymalna. W tym wypadku sklejone przez brzegi tablicy są wszystkie cztery rogi (żółte) oraz kratki 3 i 2 (czerwone u góry) z 11 i 10 (czerwone na dole).

Obydwa przedstawione wyżej sposoby sklejania są nieoptymalne. Żeby otrzymać optymalną tablicą należy wybierć najmniejszą liczbę jak największych grup pokrywającą wszystkie jedynki. W tabeli 1 było zbyt dużo grup, a w tabeli 2 obszary czerwone były niepotrzebne bo te same jedynki zawierały się w innych sklejonych obszarach. Tablicą optymalną jest tablica poniżej.

x1 x2 \ x3 x4	00	01	11	10
00	-	0	1	1
01	0	0	1	1
11	0	-	1	0
10	1	0	1	1

Formułą minimalną, równoważną funkcji pierwotnej jest suma iloczynów odpowiadających wybranym grupom.

Funkcja zminimalizowana na podstawie tabeli powyżej wygląda tak:

$$f(x1, x2, x3, x4) = (\overline{x_1} \cdot x_3) + (\overline{x_2} \cdot \overline{x_4}) + (x_3 \cdot x_4)$$

2 Ćwiczenie nr 1

Dana jest następująca siatka Karnaugha:

AB \ CD	00	01	11	10
00	1	1	0	1
01	0	1	0	1
11	1	1	1	0
10	1	-	0	0

1) Za pomocą elementów NAND zrealizować funkcję zadaną siatką Karnaugha.
2) Za pomocą elementów NOR zrealizować funkcję zadaną siatką Karnaugha.

Ad.1.

Tworzymy odpowiednio grupy, pamiętając przy tym o zasadach według których trzeba postępować (zostały nadmienione w teorii). Jeśli minimalizujemy funkcję używając bramki AND musimy pamiętać o tym, że wtedy grupujemy „1”.

Na poniższym rysunku przedstawione są prawidłowo zakreślone grupy. Po kolei minimalizujemy każdą zaznaczoną funkcję i otrzymujemy:

AB \ CD	00	01	11	10
00	1	1	0	1
01	0	1	0	1
11	1	1	1	0
10	1	-	0	0

$F_1 = \overline{B}\,\overline{C}$

$F_2 = A\overline{C}$

$F_3 = \overline{C}D$

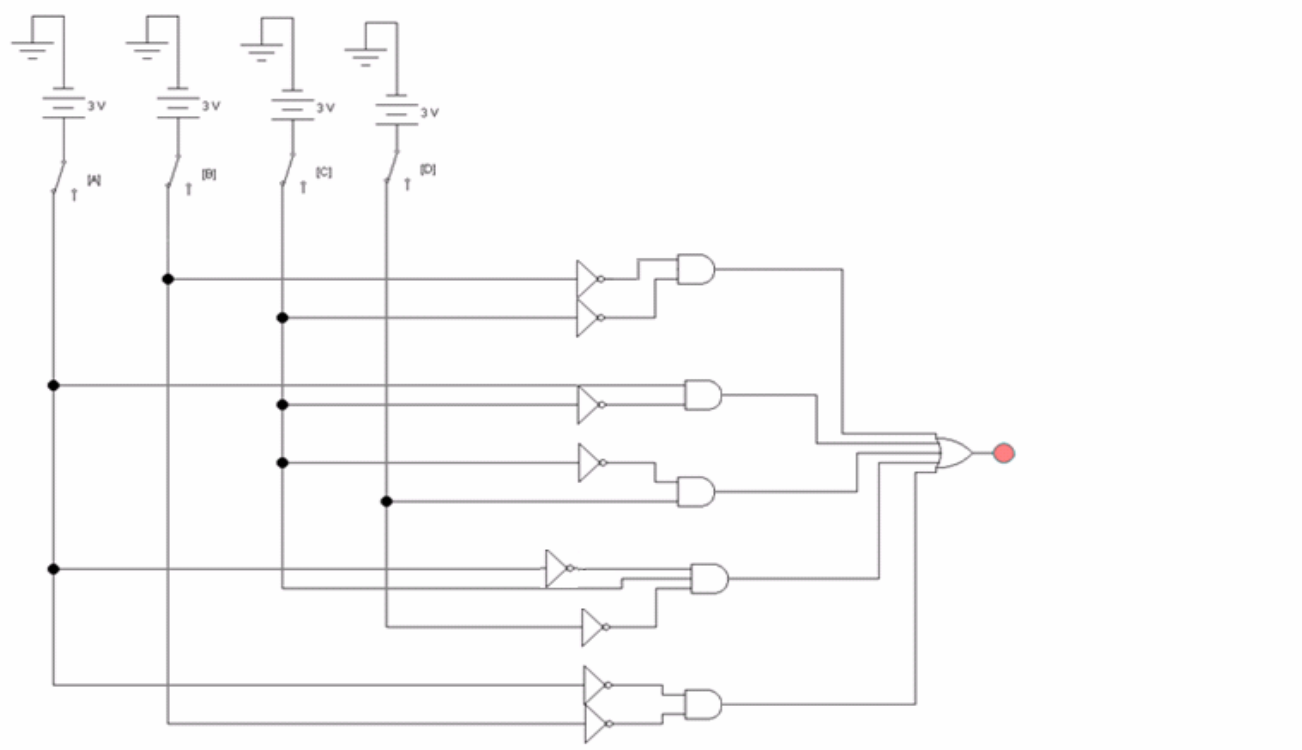
$F_4 = \overline{A}C\overline{D}$

$F_5 = \overline{A}\overline{B}$

Widać, że żadnej z podanych wyżej funkcji nie da się już bardziej zminimalizować. Oznacza to tyle, że grupy zostały odpowiednio dobrane. Teraz wszystko wystarczy do siebie dodać i otrzymujemy ostateczną, zminimalizowaną funkcję:

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

Schemat podanej funkcji:



Ad. 2.

Musimy stworzyć nowe grupy, ale tym razem bierzemy pod uwagę „0”, bo to je będziemy grupować. Zasada jest taka sama jak podczas grupowania „1”.

Jedyna różnica w minimalizacji jest taka, że tutaj mnożymy dodawane do siebie zanegowane zmienne.

Co to oznacza? Jeśli z tabelki odczytamy wartość A=1, to przy funkcji wpisujemy jakby A przyjmowało wartość 0, tzn. zanegowane A.

Gdybyśmy grupowali jedynki, pierwsza funkcja wyglądałaby tak:

$F_{1x} = \overline{A}B\overline{C}\overline{D}$

Zauważmy, że przy grupowaniu jedynek mnożyliśmy zmienne. Tym razem musimy się przestawić na dodawanie i należy pamiętać o zanegowaniu zmiennych:

AB \ CD	00	01	11	10
00	1	1	2: 0	1
01	1: 0	1	0	1
11	1	1	1	3: 0
10	1	-	4: 0	0

$F_1 = A + \overline{B} + C + D$

$F_2 = A + \overline{C} + \overline{D}$

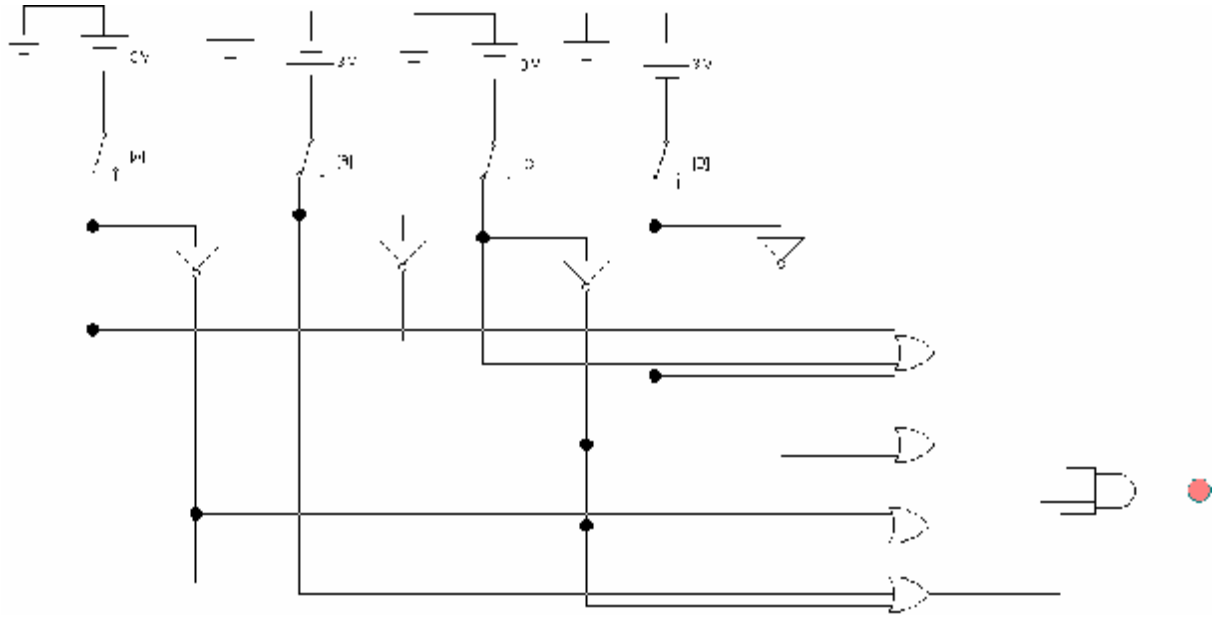
$F_3 = \overline{A} + \overline{C} + D$

$F_4 = \overline{A} + B + \overline{C}$

Ostateczna, zminimalizowana funkcja:

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

Schemat:



3 Ćwiczenie nr 2

Dane są postaci:

$$F = \sum (0,1,4,5,6)_{ABC} + f(3)_{ABC}$$
$$F = \prod (1,2,3,7)_{ABC} + f(0,5)_{ABC}$$

Należy zapisać podane funkcje w postaci tabeli Karnaugha.

Zajmijmy się pierwszą funkcją, tzn.:

$$F = \sum (0,1,4,5,6)_{ABC} + f(3)_{ABC}$$

Pierwszą rzeczą jaką należy wiedzieć, jest to, że jeśli przy funkcji widzimy znak sumy to interesują nas wyłącznie „1”.

Ważna jest również kolejność liter zapisanych w indeksach dolnych. Tutaj nie obędzie się bez znajomości zamiany cyfr dziesiętnych na binarne.

Ważne jest to, że w tym wypadku litera A odpowiada cyfrze najbardziej znaczącej, a litera C – najmniej.

Liczba (0)10 to binarnie 0, a, że musimy zapisać ją na trzech bitach, bo mamy 3 literki w ineksie to we wszystkich kolumnach zapisujemy 0.

Liczba (1)10 to binarnie 1, a na trzech bitach będzie wyglądać następująco: 001.

Liczba (3)10 to 011, itd.

Możemy sobie rozpisać kolejne liczby w tabelce:

Liczba	A(22)	B (21)	C (20)
0	0	0	0
1	0	0	1
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0

Tworzymy tabelę Karnaugha na trzy zmienne (A,B,C):

A / BC	00	01	11	10
0				
1				

Teraz w odpowiednie komórki będziemy wpisywać kolejne liczby. Zaczynamy od cyfry „0”. Sprawdzamy, jakie wartości przyjmuje A, B oraz C i wpisujemy „1” w odpowiednie miejsce:

A / BC	00	01	11	10
0	1			
1				

Kolejną cyfrą jaką musimy wpisać do tabelki jest „1”, która dla A i B przyjmuje wartość „0”, a dla C – wartość „1”. Wybieramy odpowiednie miejsce i wpisujemy kolejną „1”.

A / BC	00	01	11	10
0	1	1		
1				

Postępujemy tak z kolejnymi cyframi: „4”, „5” i „6”. Z cyfrą „3” na razie się wstrzymujemy, gdyż przed nią stoi znak f.
Tabelka po uzupełnieniu powyższych liczb powinna wyglądać tak:

A / BC	00	01	11	10
0	1 (0)	1 (1)		
1	1 (4)	1 (5)		1 (6)

Przejdźmy teraz do wspomianego znaku f. Tym razem w miejsce kratki, zamiast „1” wpisujemy „-” (stan bierny) i na tym polega cała różnica.
„3” to A=0, B=1 i C=1, uzupełniamy więc ponowanie tabelkę:

A / BC	00	01	11	10
0	1	1		-
1	1	1		1

Miejsca, które zostały puste uzupełniamy zerami:

A / BC	00	01	11	10
0	1	1	0	-
1	1	1	0	1

Pozostała druga funkcja:
$F = \prod (1,2,3,7)_{ABC} + f(0,5)_{ABC}$
Od pierwszej, różni się jedynie tym, że zamiast „1” teraz wpisujemy w tabelkę najpierw „0”, ponieważ pojawił się znak iloczynu.
Znowu dla ułatwienia rozpisujemy sobie wszystkie liczby:

Liczba	A(22)	B (21)	C (20)
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
5	1	0	1
7	1	1	1

Tabelka uzupełniona o podane wyżej cyfry włącznie z miejscami biernymi:

A / BC	00	01	11	10
0	-	0	0	0
1		-	0	

Tym razem w puste miejsca wpisujemy „1”.

A / BC	00	01	11	10
0	-	0	0	0
1	1	-	0	1

4 Ćwiczenie nr 3

Zaprojektuj układ sterowania samochodzikiem, który posiada 4 przyciski:
L – odpowiadający za jazdę w lewo,
P – w prawo,
T –do tyłu,
N – naprzód.

Samochodzik jednak jeździ TYLKO w wypadku gdy:
- jest wciśnięty tylko jeden z przycisków,
- są wciśnięte 2 przyciski jednocześnie: Naprzód + Lewo lub Naprzód + Prawo.

Pierwszym krokiem do rozwiązania tego zadania będzie stworzenie tabeli prawdy dla przycisków sterowania.
Tworzymy więc cztery kolumny odpowiadające za Lewo, Przód, Tył oraz Naprzód. Teraz uzupełniamy te kolumny o wszystkie możliwe kombinacje „0” i „1”.
Dodajemy piątą kolumnę – „S”. Wpisujemy w niej jedynki w odpowiednie miejsca, tzn. takie, w których samochodzik ma jechać.
Tabelka wygląda następująco:

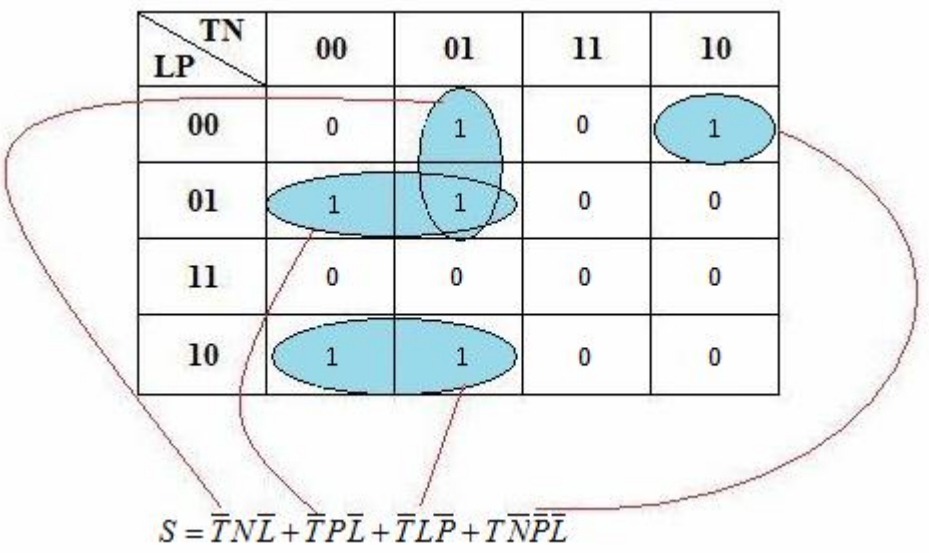
L	P	T	N	S
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	1
0	1	1	0	0
0	1	1	1	0
1	0	0	0	1
1	0	0	1	1
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

Gdy wykonaliśmy już tabelkę to czas na zbudowanie na tej podstawie siatki Karnaugha. Budujemy ją w następujący sposób:

- patrzymy na pierwszy wiersz tabelki, mamy w nim: 0000 → 0, więc wpisujemy wartość 0 w odpowiednią kratkę w siatce, czyli tam gdzie mamy wartość dla LP→00 TN→00.
- kolejna wartość w tabelce to 0001→1, więc w odpowiedniej kratce wpisujemy 1. Powtarzamy ten algorytm, aż do wypełnienia całej siatki.

Gdy to zrobimy należy pogrupować wartości w tabelce. Można grupować albo „0” albo „1”, należy jednak pamiętać jakich bramek będziemy musieli użyć w budowaniu schematu. My grupujemy jedynki, więc w schemacie użyjemy bramek typu NAND.

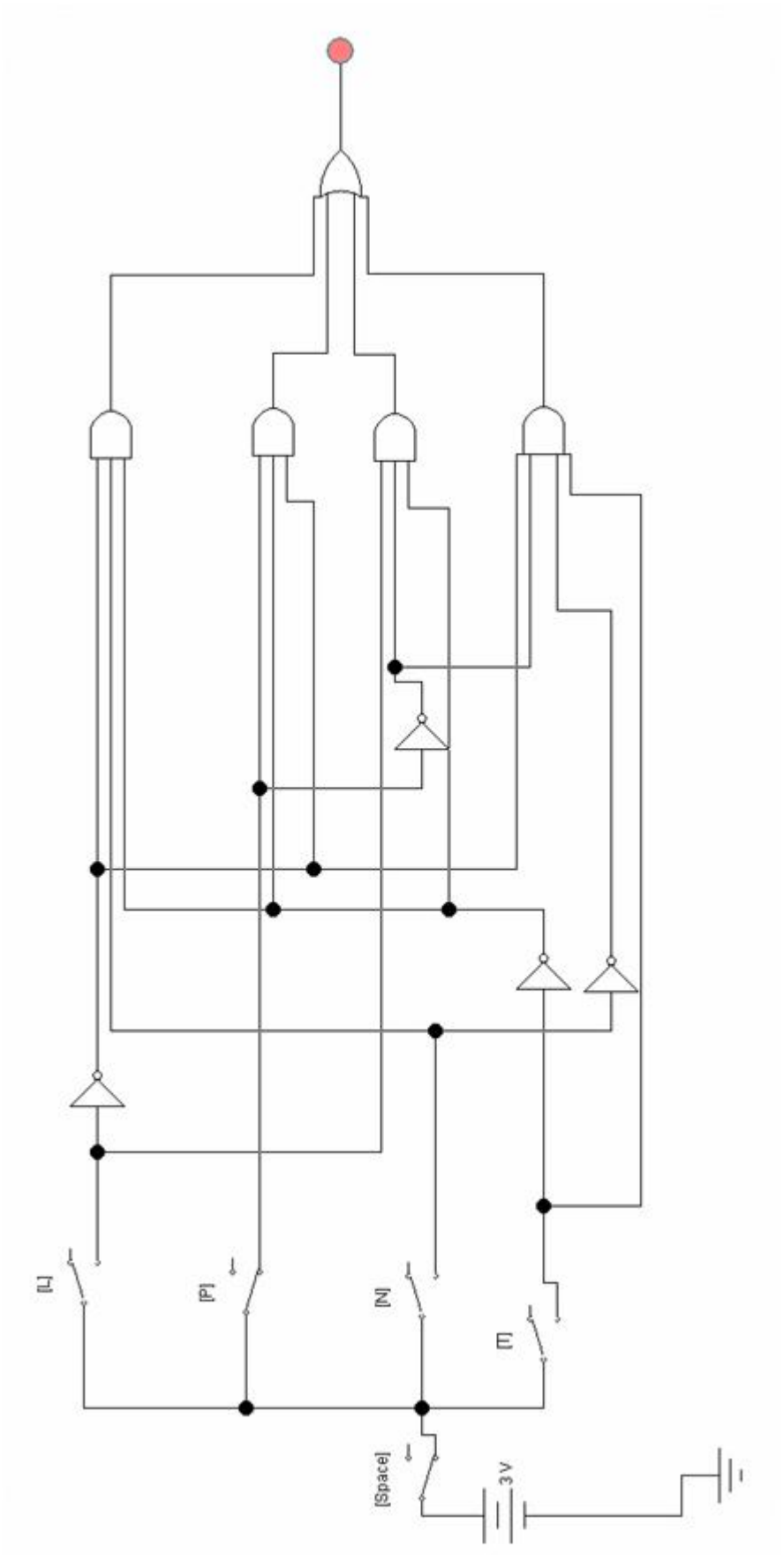
Na poniższym rysunku pokazane są grupy oraz ich wartości.



Dzięki temu możemy teraz zbudować schemat. Tworzymy 4 zasilania, które będą naszymi przyciskami oraz 4 bramki typu AND, do których podłączymy przyciski.

Dlaczego 4? Ponieważ w siatce wyszły nam 4 grupy, które trzeba połączyć za pomocą odpowiednich bramek, aby schemat był kompletny i miał szanse działać.

Oto działający, poprawnie skonstruowany układ:


[L]
[P]
[N]
[T]
[Space]
3 V