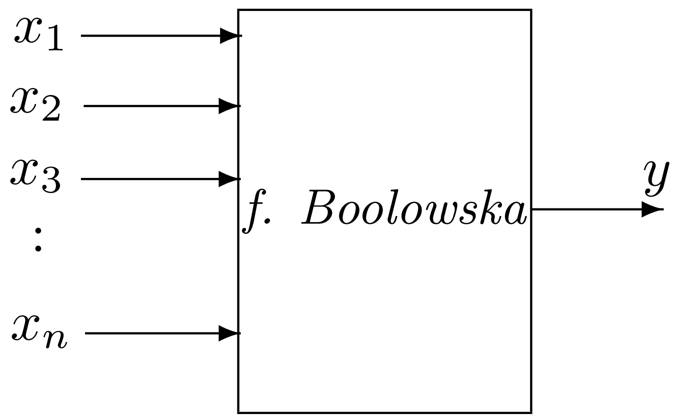
Systemy cyfrowe i podstawy elektroniki

Adam Szmigielski
aszmigie@pjwstk.edu.pl

materiały: $ftp(public)://aszmigie/SYC$

Minimalizacja funkcji Boolowskich - wykład 7

Funkcja Boolowska - przypomnienie



- *Funkcją boolowską* n argumentową nazywamy odwzorowanie $f : B^n \to B$, gdzie $B = \{0, 1\}$ jest zbiorem wartości funkcji.
- *Funkcja boolowska* jest matematycznym modelem *układu kombinacyjnego*.

Opis funkcji Boolowskiej - tabele prawdy

- funkcja jednej zmiennej (np. negacja $f(a) = \neg a$)

a	f(a)
0	1
1	0

- Funkcja dwóch zmiennych (np. funkcja *mod2*: $f(a,b) = a \otimes b$)

a	b	$f(a,b)$
0	0	0
0	1	1
1	0	1
1	1	0

Zbiory zer i jedynek w postaci binarnej i dziesiętnej

a	b	$f(a,b)$
0	0	0
0	1	1
1	0	1
1	1	0

$f^1 = \begin{bmatrix} 01 \\ 10 \end{bmatrix}$ - zbiór jedynek w postaci binarnej

$f^0 = \begin{bmatrix} 00 \\ 11 \end{bmatrix}$ - zbiór zer w postaci binarnej

$f^1 = \{1, 2\}$ -zbiór jedynek w postaci dziesiętnej

$f^0 = \{0, 3\}$ -zbiór zer w postaci dziesiętnej

Postać sumacyjna - DNF (ang. *Disjunctive normal form*)

a	b	$f(a,b)$
0	0	0
0	1	1
1	0	1
1	1	0

Postać sumacyjna: funkcja f jest sumą iloczynów

$f = \ldots(\ldots \wedge \ldots \wedge \ldots) \vee (\ldots \wedge \ldots \wedge \ldots) \vee (\ldots \wedge \ldots \wedge \ldots)\ldots$

Wyrażenie w nawiasie (iloczyn) odpowiada jednej jedynce.

W tym konkretnym przypadku: $f(a,b) = (\overline{a} \wedge b) \vee (a \wedge \overline{b})$.

Zapis dziesiętny: $f(a,b) = \sum(1,2)$

Postać iloczynowa - CNF (ang. *Conjunctive normal form*)

a	b	$f(a,b)$
0	0	0
0	1	1
1	0	1
1	1	0

Postać sumacyjna: funkcja f jest iloczynem sum

$f = \ldots(\ldots \vee \ldots \vee \ldots) \wedge (\ldots \vee \ldots \vee \ldots) \wedge (\ldots \vee \ldots \vee \ldots)\ldots$

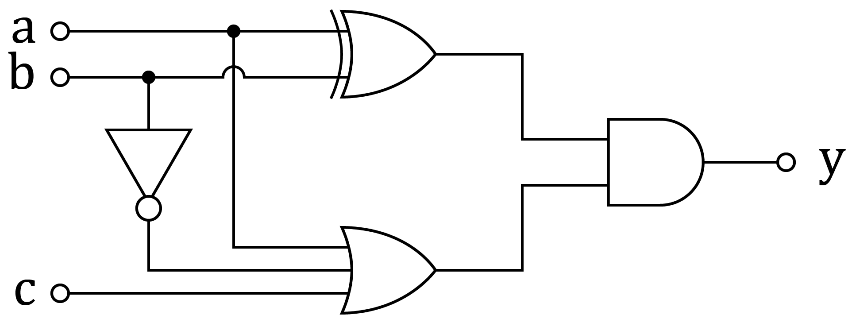
Wyrażenie w nawiasie (suma) odpowiada jednemu zeru.

W tym konkretnym przypadku: $f(a,b) = (a \vee b) \wedge (\overline{a} \vee \overline{b})$[a].

Zapis dziesiętny: $f(a,b) = \prod(0,3)$

[a] należy pamiętać o zanegowaniu zmiennych, tj. Nawiasowi $(a \vee b)$ odpowiada sytuacja, gdy $a = 0$ i $b = 0$.

Schematy układów logicznych



1. Schemat logiczny opisuje logiczną strukturę funkcji boolowskich,
2. Przepływ informacji jest od wejścia do wyjścia, tj. $y = f(a, b, c)$,
3. Kropka oznacza połączenie,
4. Prezentowany schemat realizuje funkcje boolowską:

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

Realizacja funkcji boolowskiej opisanej tabelą prawdy

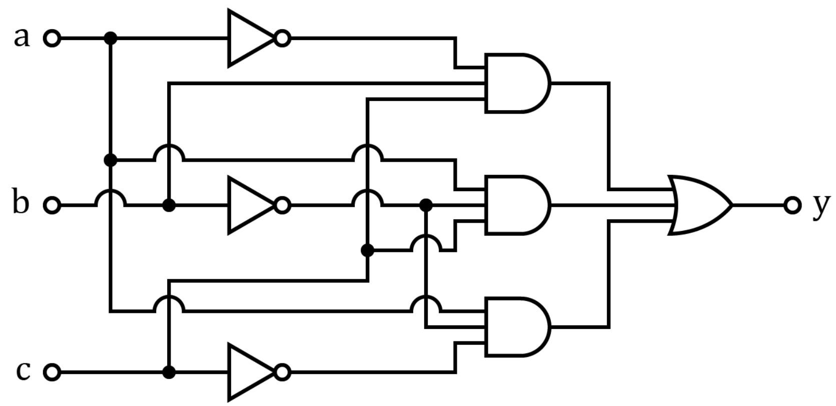
a	b	c	$y = f(a, b, c)$
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

- Sumacyjna postać kanoniczna (szukamy ’1’ na wyjściu):

$$y = f(a, b, c) = \overline{a}bc + a\overline{b}\overline{c} + a\overline{b}c$$

Realizacja funkcji boolowskiej na bramkach

a	b	c	$y = f(a,b,c)$
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0



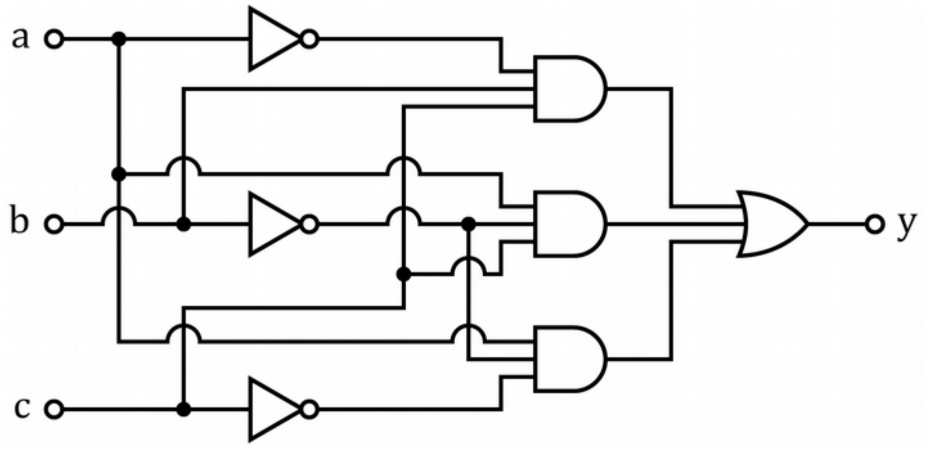
- $y = f(a,b,c) = \overline{a}bc + a\overline{b}\overline{c} + a\overline{b}c$
- Czy można użyć mniejszej liczby bramek ?

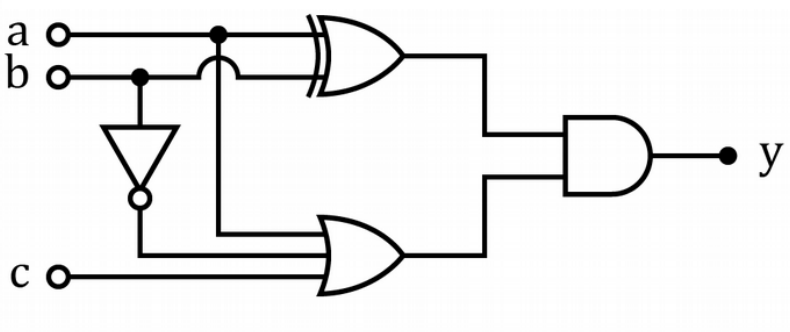
Przekształcenia funkcji boolowskiej

1. $y = f(a, b, c) = \overline{a}bc + a\overline{b}\overline{c} + a\overline{b}c$
2. $\overline{a}bc + a\overline{b}\overline{c} + a\overline{b}c = \overline{a}bc + a\overline{b}\overline{c} + a\overline{b}c + a\overline{b}c$
3. $\overline{a}bc + a\overline{b}\overline{c} + a\overline{b}\overline{c} + a\overline{b}c = \overline{a}bc + a\overline{b}(\overline{c} + c) + a\overline{b}c = \overline{a}bc + a\overline{b} + a\overline{b}c$
4. $\overline{a}bc + a\overline{b} + a\overline{b}c = \overline{a}bc + aa\overline{b} + a\overline{b}\overline{b} + a\overline{b}c$
5. $\overline{a}bc + aa\overline{b} + a\overline{b}\overline{b} + a\overline{b}c = \overline{a}bc + aa\overline{b} + a\overline{b}\overline{b} + a\overline{b}c + a\overline{a}b + ab\overline{b}$
6. $\overline{a}bc + aa\overline{b} + a\overline{b}\overline{b} + a\overline{b}c + a\overline{a}b + ab\overline{b} = a\overline{b}(a + \overline{b} + c) + \overline{a}b(a + \overline{b} + c)$
7. $a\overline{b}(a + \overline{b} + c) + \overline{a}b(a + \overline{b} + c) = (a\overline{b} + \overline{a}b)(a + \overline{b} + c)$

Równoważność funkcji Boolowskich

- Funkcje boolowskie mogą być sobie równoważne
  $\overline{a}bc + a\overline{b}\overline{c} + a\overline{b}c \Leftrightarrow (a\overline{b} + \overline{a}b)(a + \overline{b} + c)$
- Równoważne są więc realizacje tych funkcji





Zadanie optymalizacji funkcji

Przy projektowaniu układów kombinacyjnych dąży się do minimalizacji kosztów układu. Można tego dokonać na kilka sposobów:

- Poprzez minimalizacje liczby bramek,
- Poprzez redukcje liczby wejść bramek,
- Poprzez zmniejszenie różnorodności bramek,
- Poprzez redukcje czasu projektowania układu.

Redukcja różnorodności rodzajów bramek

Jaka jest najmniejsza liczba różnorodności bramek ?

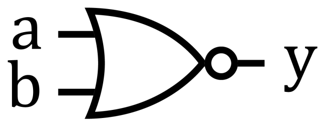
Logika klasyczna (operująca na operatorach koniunkcji $\wedge$, alternatywy $\vee$, implikacji $\Rightarrow$ i negacji $\neg$) jest nadmiarowa, tzn. część operatorów można zdefiniować w oparciu o pozostałe. Najmniejsze systemy to:

- Implikacyjno-negacyjny - operujący negacją i implikacją,
- Koniunkcyjno-negacyjny - operujący negacją i koninkcją,
- Alternatywno-negacyjny - operujący negacją i alternatywą.

NAND i NOR - bramki uniwersalne

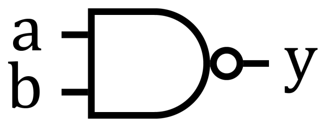
- NOR realizuje zanegowaną sumę logiczną $y = \overline{a \vee b}$,

a	b	NOR(a,b)
0	0	1
0	1	0
1	0	0
1	1	0



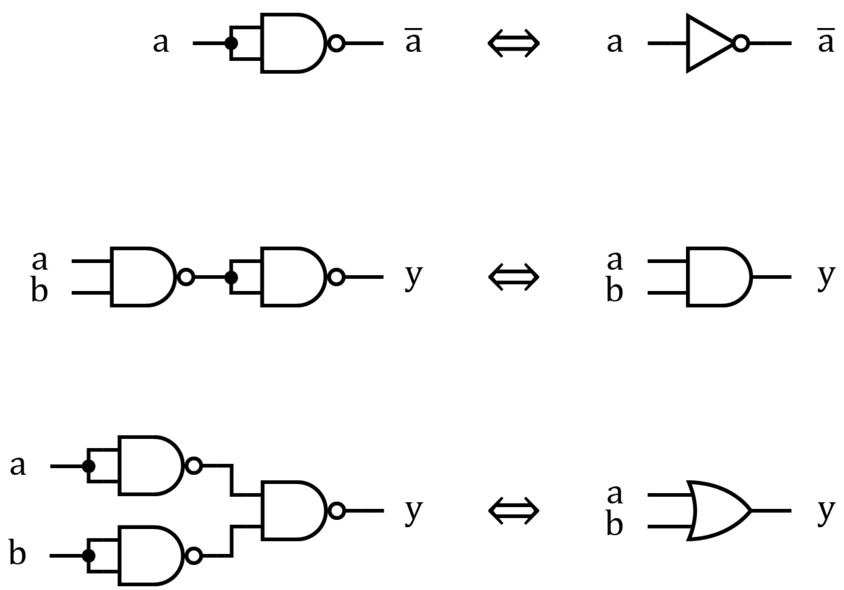
- NAND realizuje zanegowny iloczyn logiczny $y = \overline{a \wedge b}$,

a	b	NAND(a,b)
0	0	1
0	1	1
1	0	1
1	1	0



Realizacja negacji, iloczynu i sumy ma bramkach NAND

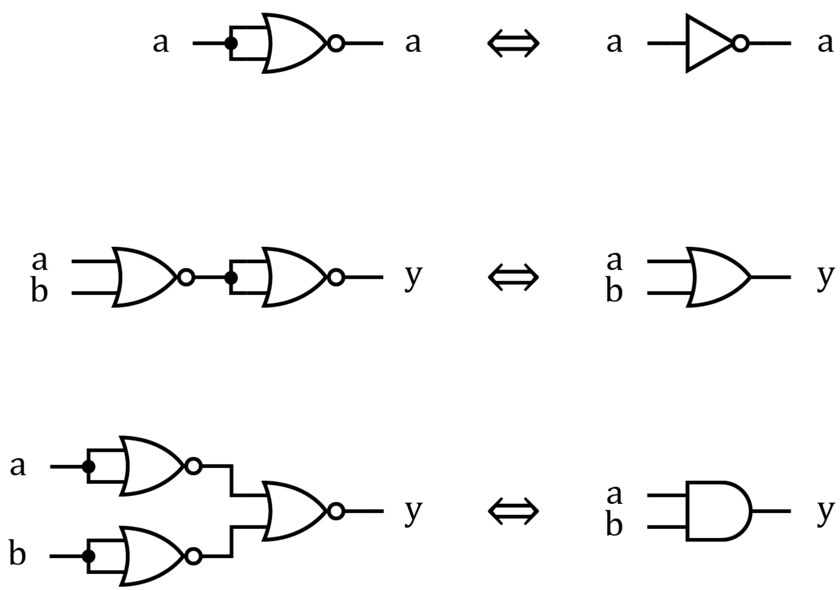
- Za pomocą bramek NAND można zrealizować negację, iloczyn i sumę logiczną,



- Na bramkach NAND można zrealizować dowolną funkcję Boolowską.

Realizacja negacji, sumy i iloczynu ma bramkach NOR

- Za pomocą bramek NOR można zrealizować negację, sumę i iloczyn logiczny,



- Na bramkach NOR można zrealizować dowolną funkcję Boolowską.

Kod Graya

000
001
011
010
110
111
101
100

Kod Graya jest dwójkowym kodem bezwagowym niepozycyjnym, który charakteryzuje się tym, że dwa kolejne słowa kodowe różnią się tylko stanem jednego bitu. Jest również kodem cyklicznym, bowiem ostatni i pierwszy wyraz tego kodu także spełniają w/w zasadę.

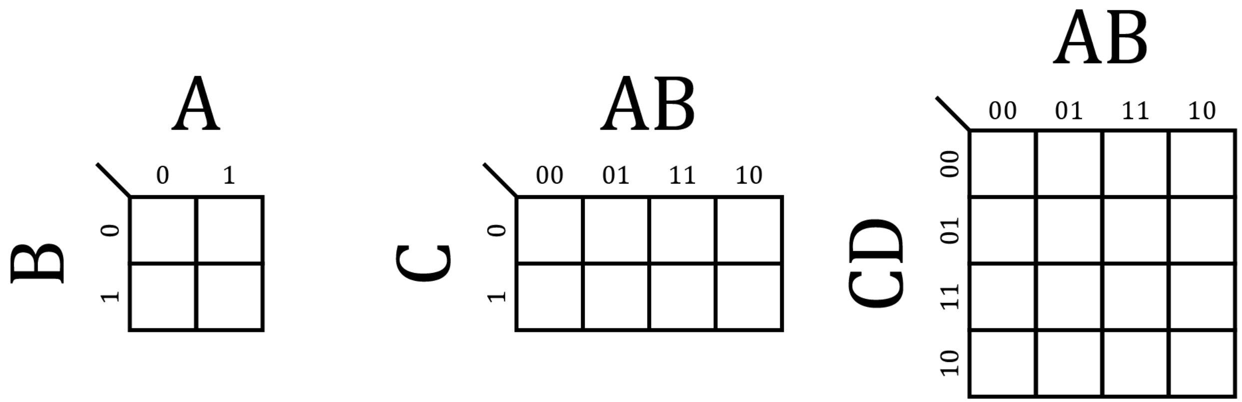
Reguła grupowania a kod Graya

- Reguła grupowania: $a \cdot f(x_1, x_2, \dots x_n) + \overline{a} \cdot f(x_1, x_2, \dots x_n) = (a + \overline{a}) \cdot f(x_1, x_2, \dots x_n) = f(x_1, x_2, \dots x_n)$
-

000	$\overline{a}\overline{b}\overline{c}$
001	$\overline{a}\overline{b}c$
011	$\overline{a}bc$
010	$\overline{a}b\overline{c}$
110	$ab\overline{c}$
111	abc
101	$a\overline{b}c$
100	$a\overline{b}\overline{c}$

- Dwa sąsiadujące wyrażenia zastępujemy jednym, pomijając ten element na którym nastąpiła zmiana np. wyrażenie $\overline{a}bc + \overline{a}b\overline{c}$ jest równoważne wyrażeniu $\overline{a}b$.

Mapy Karnaugha



- Mapy Karnough'a są pomocne przy minimalizacji funkcji boolowskiej,
- Mapa Karnough'a jest wypełniana w oparciu o tablice prawdy,
- Zmienne w wierszach i kolumnach uporządkowane są zgodnie z kodem Graya, co znacznie ułatwia zastosowanie reguły grupowania.

Mapy Karnaugha

a	b	c	$y = f(a,b,c)$
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

i

abc	abc	
000	$\overline{a}\overline{b}\overline{c}$	0
001	$\overline{a}\overline{b}c$	0
011	$\overline{a}bc$	1
010	$\overline{a}b\overline{c}$	0
110	$ab\overline{c}$	0
111	abc	0
101	$a\overline{b}c$	1
100	$a\overline{b}\overline{c}$	1

.

Różne postacie mapy Karnaugh'a

abc	
$\overline{a}\overline{b}\overline{c}$	0
$\overline{a}\overline{b}c$	0
$\overline{a}bc$	1
$\overline{a}b\overline{c}$	0
$ab\overline{c}$	0
abc	0
$a\overline{b}c$	1
$a\overline{b}\overline{c}$	1

$a\backslash bc$	00	01	11	10
0	0	0	1	0
1	1	1	0	0

$ab\backslash c$	0	1
00	0	0
01	0	1
11	0	0
10	1	1

Mapy Karnaugha - grupowanie '1'

abc	
$\overline{a}\overline{b}\overline{c}$	0
$\overline{a}\overline{b}c$	0
$\overline{a}bc$	1
$\overline{a}b\overline{c}$	0
$ab\overline{c}$	0
abc	0
$a\overline{b}c$	1
$a\overline{b}\overline{c}$	1

$a\backslash bc$	00	01	11	10
0	0	0	1	0
1	1	1	0	0

$ab\backslash c$	0	1
00	0	0
01	0	1
11	0	0
10	1	1

- Grupujemy '1' tylko w pionie albo poziomie w ilościach będących krotnością dwójki, tworząc *sumacyjną postać kanoniczną*,
- Pozbywamy się tej zmiennej która się zmienia.
- Minimalna *sumacyjna postać kanoniczną*: $y = a\overline{b} + \overline{a}bc$.

Mapy Karnaugha - grupowanie ’0’

abc	
$\overline{a}\overline{b}\overline{c}$	0
$\overline{a}\overline{b}c$	0
$\overline{a}bc$	1
$\overline{a}b\overline{c}$	0
$ab\overline{c}$	0
abc	0
$a\overline{b}c$	1
$a\overline{b}\overline{c}$	1

$a\backslash bc$	00	01	11	10
0	0	0	1	0
1	1	1	0	0

$ab\backslash c$	0	1
00	0	0
01	0	1
11	0	0
10	1	1

- Sklejamy "0"tylko w pionie albo poziomie w ilościach będących krotnością dwójki, tworząc *iloczynową postać kanoniczną*,
- Pozbywamy się tej zmiennej która się zmienia. Pozostałe zmienne negujemy,
- Minimalna *iloczynową postać kanoniczną*: $y = (a + b) \cdot (\overline{b} + c) \cdot (\overline{a} + \overline{b})$.

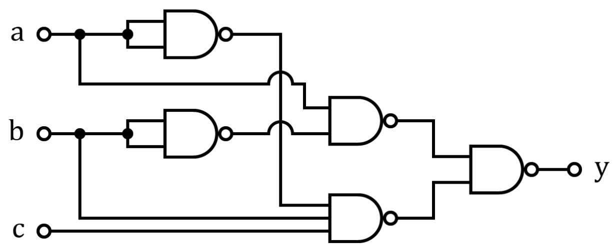
Równoważność postaci sumacyjnej i iloczynowej

- Jak można się domyślać, obie postacie są sobie równoważne, tj.:

$$a\overline{b} + \overline{a}bc \Leftrightarrow (a + b) \cdot (\overline{b} + c) \cdot (\overline{a} + \overline{b})$$

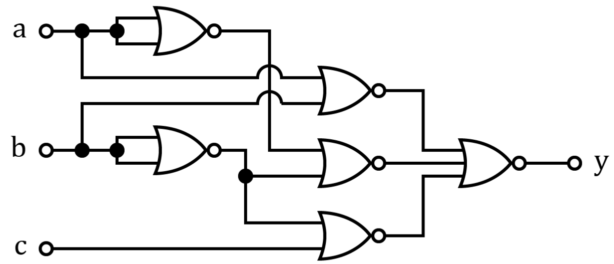
- uzasadnienie:
- $(a + b) \cdot (\overline{b} + c) \cdot (\overline{a} + \overline{b}) \Leftrightarrow (a\overline{b} + ac + b\overline{b} + bc) \cdot (\overline{a} + \overline{b})$
- $(a\overline{b} + ac + b\overline{b} + bc) \cdot (\overline{a} + \overline{b}) \Leftrightarrow a\overline{a}\overline{b} + a\overline{a}c + \overline{a}bc + a\overline{b}\overline{b} + a\overline{b}c + b\overline{b}c$
- $a\overline{a}\overline{b} + a\overline{a}c + \overline{a}bc + a\overline{b}\overline{b} + a\overline{b}c + b\overline{b}c \Leftrightarrow \overline{a}bc + a\overline{b} + a\overline{b}c$
- $\overline{a}bc + a\overline{b} + a\overline{b}c \Leftrightarrow a\overline{b} + \overline{a}bc$

Realizacja *postaci sumacyjnej* **na bramkach NAND**



- Daną funkcję $y = a\overline{b} + \overline{a}bc$ negujemy dwukrotnie
- $y = \overline{\overline{a\overline{b} + \overline{a}bc}}$. Dla "wewnętrznejńegacji stosujemy prawo deMorgana:
- $y = \overline{\overline{a\overline{b}} \cdot \overline{\overline{a}bc}}$

Realizacja *postaci iloczynowej* **na bramkach NOR**



- Daną funkcję $y = (a+b)\cdot(\overline{b}+c)\cdot(\overline{a}+\overline{b})$ negujemy dwukrotnie
- $y = \overline{\overline{(a+b)\cdot(\overline{b}+c)\cdot(\overline{a}+\overline{b})}}$. Dla "wewnętrznejńegacji stosujemy prawo deMorgana:
- $y = \overline{\overline{a+b} + \overline{\overline{b}+c} + \overline{\overline{a}+\overline{b}}}$

Zadania na ćwiczenia

Dana jest funkcja czterech zmiennych wskazana przez prowadzącego[a]
$y = \sum(\ldots, \ldots, \ldots, \ldots,)$.

1. Zrealizuj na bramkach NAND minimalną postać tej funkcji.
2. Posługując się tylko bramkami NOR zrealizuj sterowanie robota mobilnego, realizującą jego bezkolizyjne poruszanie się. Robot wyposażony jest w trzy czujniki, umieszczone jeden z przodu i dwa po bokach. Czujnik identyfikuje przeszkodę - "1- jest przeszkoda, "0- brak przeszkody. Robot posiada różnicowy mechanizm jezdny, tj. dwa niezależne silniki umieszczone na jednej osi, które umożliwiają - *jazdę do przodu* (oba silniki włączone), *skręt w lewo* albo *w prawo* (odpowiednio jeden silnik włączony drugi wyłączony) oraz *zatrzymanie* robota (oba silniki wyłączone). W przypadku braku możliwości jazdy robot powinien zatrzymać się.

[a] dla każdego inna