Sławomir Kulesza

Technika cyfrowa

Minimalizacja form boolowskich

Wykład dla studentów III roku Informatyki

Wersja 2.0, 15/10/2018

Komparator 2-bitowy

A1	A0	B1	B0	Y1 (A > B)	Y2 (A = B)	Y3 (A < B)
0	0	0	0	0	1	0
0	0	0	1	0	0	1
0	0	1	0	0	0	1
0	0	1	1	0	0	1
0	1	0	0	1	0	0
0	1	0	1	0	1	0
0	1	1	0	0	0	1
0	1	1	1	0	0	1
1	0	0	0	1	0	0
1	0	0	1	1	0	0
1	0	1	0	0	1	0
1	0	1	1	0	0	1
1	1	0	0	1	0	0
1	1	0	1	1	0	0
1	1	1	0	1	0	0
1	1	1	1	0	1	0

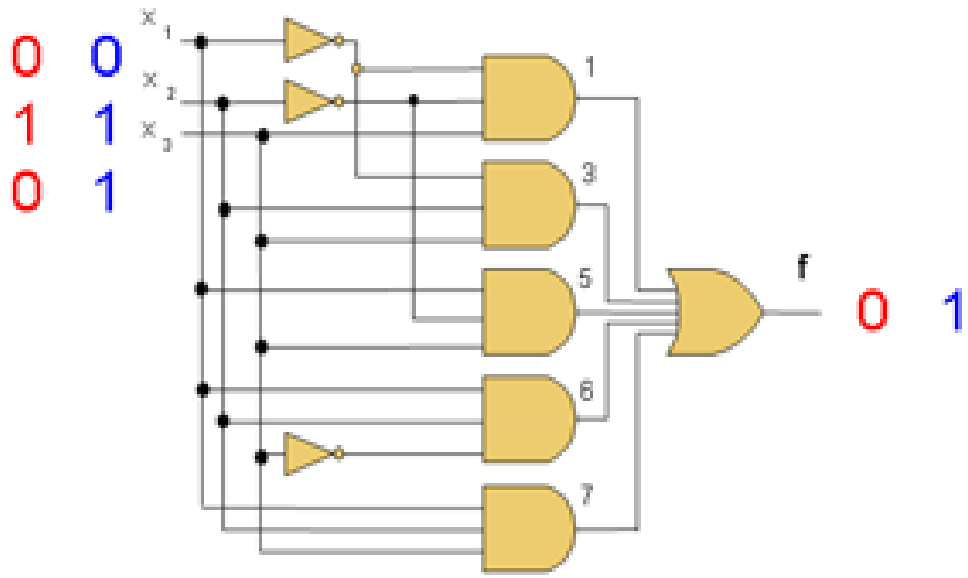
Minimalizacja form boolowskich

Minimalizacja – proces przekształcania form boolowskich w celu otrzymania możliwie najprostszych form równoważnych.

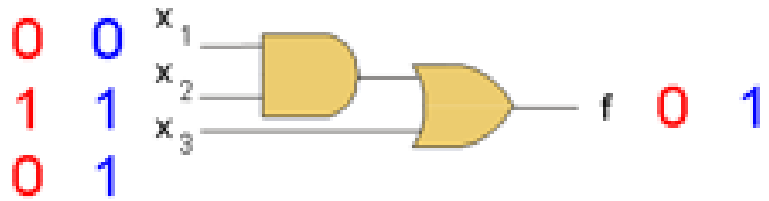
Kryterium minimalizacji – obniżenie kosztu układu cyfrowego, skrócenie ścieżek propagacji sygnałów, ograniczenie liczby sygnałów w układzie.

Drogi minimalizacji – (1) zmniejszanie liczby bramek, (2) zmniejszanie liczby wejść bramek.

Sens minimalizacji



	x_1	x_2	x_3	f
0	0	0	0	0
1	0	0	1	1
2	0	1	0	0
3	0	1	1	1
4	1	0	0	0
5	1	0	1	1
6	1	1	0	1
7	1	1	1	1



Metody minimalizacji

(1) M. algebraiczne: bazują na tożsamościach i prawach algebry Boole'a.
Wady: podejście niesystematyczne (brak algorytmu), trudno określić moment osiągnięcia formy minimalnej, tylko do form o niewielkiej złożoności.

(2) M. graficzne: siatki Karnaugha.
Wady: ograniczona liczba zmiennych wejściowych (do 5).

(3) M. numeryczne: metoda Quine'a-McCluskeya, metoda Espresso.
Wady: minimalizowane formy zwykle muszą posiadać określoną postać, problemy NP-trudne (Q-MC).
Zalety: pozwalają w systematyczny sposób dojść do formy minimalnej, dostępne oprogramowanie niekomercyjne.

Implikanty funkcji logicznej

Funkcja logiczna g jest implikantem funkcji f wtedy i tylko wtedy, gdy zachodzi implikacja:

$$\forall\ x_1x_2...x_k \in X\ [g(x_1x_2...x_k) = 1] \Rightarrow [f(x_1x_2...x_k) = 1]$$

Każda kombinacja liniowa termów w formie boolowskiej jest więc implikantem funkcji opisywanej tą formą.

Implikant prosty w formie sumacyjnej n-zmiennych jest takim iloczynem m-literałów ($m \leq n$), że po odrzuceniu choćby jednego literału przestaje być implikantem tej funkcji

Nieredukowalne formy boolowskie

Twierdzenie 1:

Każdą formę boolowską można przekształcić do postaci sumy zawierającej wyłącznie implikanty proste.

Twierdzenie 2:

Suma implikantów prostych formy boolowskiej, która po odrzuceniu któregokolwiek z nich nie opisuje funkcji logicznej f jest nieredukowalną formą boolowską.

W trakcie minimalizacji można otrzymać jedną lub więcej nieredukowalnych form boolowskich. Wybiera się wówczas formę o najmniejszej złożoności Z.

Istotne implikanty proste i jądro formy

Niech dana jest forma:

$$f(a,b,c) = a'b'c + a'bc + a'bc' + abc' + ab'c'$$

Wówczas można pokazać, że:

$f(a,b,c) = a'c + a'b + ac'$

$f(a,b,c) = a'c + bc' + ac'$

Otrzymaliśmy 2 nieredukowalne formy boolowskie o Z = 8.

Wspólne implikanty proste a'c oraz ac' są tzw. istotnymi implikantami prostymi, które tworzą jądro formy boolowskiej.

Wszystkie nieredukowalne formy boolowskie danej funkcji logicznej zawierają to samo jądro.

Ogólny schemat minimalizacji

Określ wszystkie implikanty proste formy boolowskiej

Wyznacz jądro formy (istotne implikanty proste)

Wyznacz nieredukowalne formy boolowskie

Wybór formy nieredukowalnej w oparciu o założenia projektowe

Narzędzia minimalizacji

Minimalizacja ilości literałów w formie boolowskiej korzysta z praw i tożsamości algebry Boole'a:

$F(a,b) = ab + ab' = a(b + b') = a$

A	B	F
0	0	0
0	1	0
1	0	1
1	1	1

$G(a,b) = a'b' + ab' = (a' + a)b' = b'$

A	B	G
0	0	1
0	1	0
1	0	1
1	1	0

Siatki Karnaugha

Siatka Karnaugh dla funkcji n-zmiennych składa się z 2^n–pól, w które wpisuje się wartości funkcji dla wszystkich termów.

Współrzędne kratek opisuje się w kodzie Graya, stąd termy z sąsiednich kratek różnią się stanem jednej zmiennej.

Sklejanie sąsiednich kratek pozwala na eliminację zmiennej występującej w stanie 1 oraz 0: (a+a')b = b.

a \ b	0	1
0	0 a′b′	1 a′b
1	2 ab′	3 ab

a \ bc	00	01	11	10
0	0 a′b′c′	1 a′b′c	3 a′bc	2 a′bc′
1	4 ab′c′	5 ab′c	7 abc	6 abc′

ab \ cd	00	01	11	10
00	0 a′b′c′d′	1 a′b′c′d	3 a′b′cd	2 a′b′cd′
01	4 a′bc′d′	5 a′bc′d	7 a′bcd	6 a′bcd′
11	12 abc′d′	13 abc′d	15 abcd	14 abcd′
10	8 ab′c′d′	9 ab′c′d	11 ab′cd	10 ab′cd′

Siatki Karnaugha

Siatki Karnaugha można konstruować dla dowolnej ilości literałów, aczkolwiek możliwości ich analizy kończą się w przypadku form o więcej niż 6 zmiennych (niezanegowanych).
Wypełnianie oczek jest łatwiejsze po opisaniu ich kodem NB.

x_1 \ x_2x_3	00	01	11	10
0	0	1	3	2
1	4	5	7	6

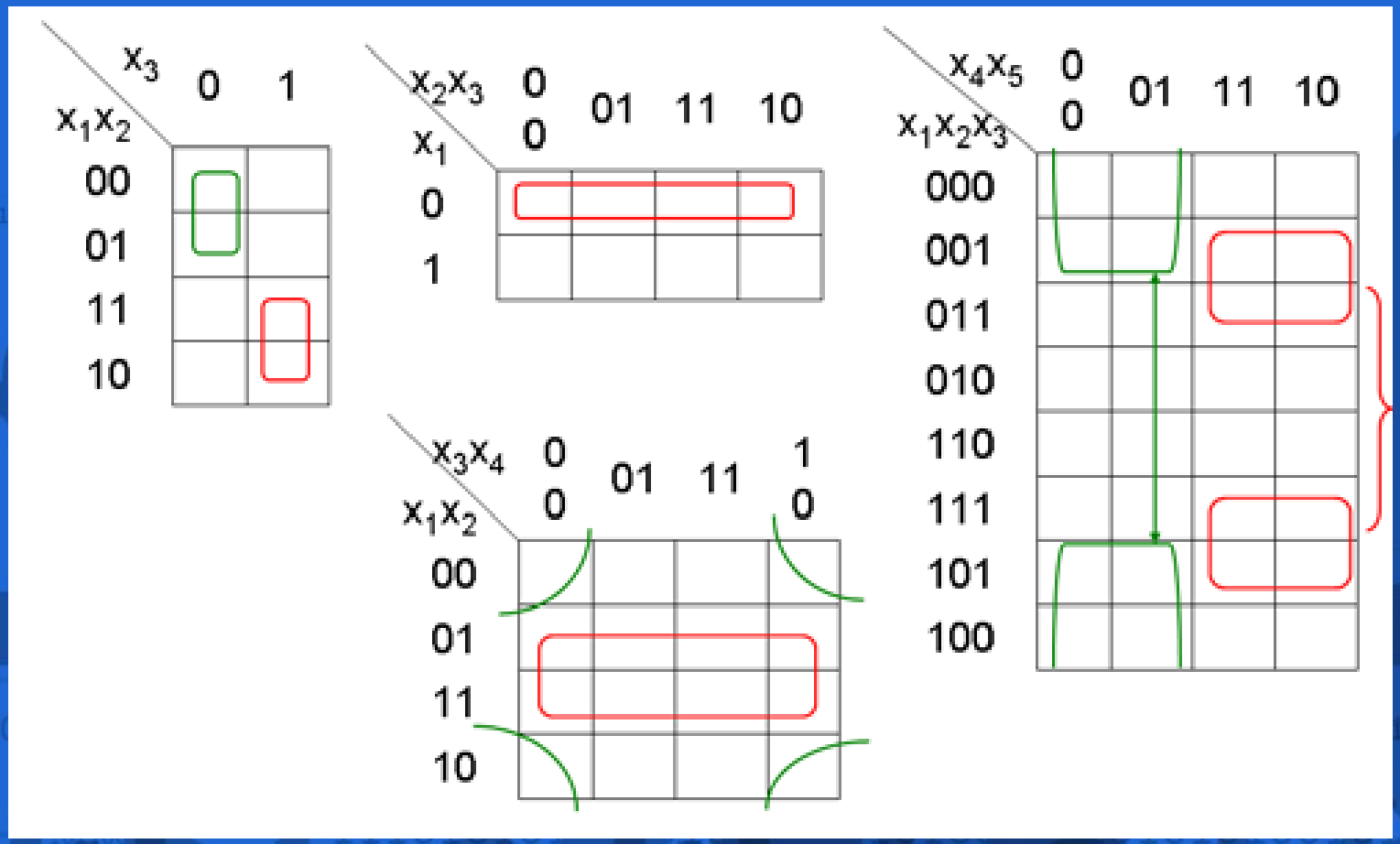
x_1x_2 \ x_3	0	1
00	0	1
01	2	3
11	6	7
10	4	5

x_1x_2 \ x_3x_4	00	01	11	10
00	0	1	3	2
01	4	5	7	6
11	12	13	15	14
10	8	9	11	10

$x_1x_2x_3$ \ x_4x_5	00	01	11	10
000	0	1	3	2
001	4	5	7	6
011	12	13	15	14
010	8	9	11	10
110	24	25	27	26
111	28	29	31	30
101	20	21	23	22
100	16	17	19	18

Sklejanie oczek siatki

Dopuszczalne jest sklejanie sąsiednich oczek jedynie w prostokątne grupy liczące 2^n-pól, przy czym można sklejać pola na brzegach siatki.



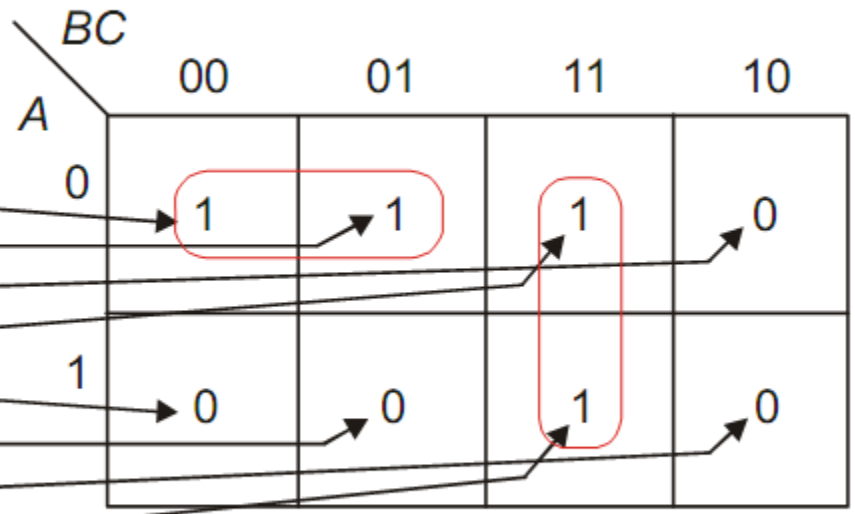
Ex. 1. Minimalizacja na siatce Karnaugh

$$f(A,B,C) = A'B'C' + A'B'C + A'BC + ABC \qquad Z = 4 + 12 = 16$$

Tablica stanów

A	B	C	f
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

Siatka Karnaugh



Po minimalizacji: $f(A,B,C) = A'B' + BC \qquad Z = 2 + 4 = 6$

Ex. 2. Minimalizacja na siatce Karnaugh

	x_1	x_2	x_3	f
0	0	0	0	0
1	0	0	1	1
2	0	1	0	0
3	0	1	1	1
4	1	0	0	0
5	1	0	1	1
6	1	1	0	1
7	1	1	1	1

1) Wpisanie funkcji do tablicy

2) Zakreślanie pętelek

Z pętelkami kojarzymy iloczyn zmiennych (prostych lub zanegowanych)

x_1x_2 \ x_3	0	1
00	0	1
01	0	1
11	1	1
10	0	1

$$f = x_1x_2 + x_3$$

Ex. 3. Minimalizacja na siatce Karnaugh

$f = \Sigma[0, 5, 6, 7, 10, (2, 3, 11, 12)]$

x_1x_2 \ x_3x_4	00	01	11	10
00	0	1	3	2
01	4	5	7	6
11	12	13	15	14
10	8	9	11	10

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

$$f = \bar{x}_1x_3 + \bar{x}_2x_3 + \bar{x}_1\bar{x}_2\bar{x}_4 + \bar{x}_1x_2x_4$$

Ex. 4. Minimalizacja na siatce Karnaugh

$$y = ab'c'd' + a'b'cd + a'bc'd + ab'cd' + a'b'c'd + a'bcd + abc'd' + abc'd$$

$Z = 8 + 8 \cdot 4 = 40$

Opis na siatce Karnaugh ►

ab \ cd	00	01	11	10
00		1	1	
01		1	1	
11	1	1		
10	1			1

Różne pola mogą się pokrywać, gdyż $x + x = x$

Dwie sąsiednie kratki jedynkowe (1-kostka) ► eliminacja jednej zmiennej:

Np. $ab'c'd' + ab'cd' = ab'd'(c' + c) = ab'd'$

Cztery sąsiednie kratki (2-kostka) ► eliminacja dwu zmiennych:

Np. $a'b'cd + a'bc'd + a'b'c'd + a'bcd = a'd$

Ex. 4. Minimalizacja na siatce Karnaugh

Wszystkie możliwe sklejenia ► forma o pięciu termach:

$$y = ab'd' + a'd + abc' + bc'd + ac'd'$$

► *Jeśli przynajmniej jedna kratka w polu danego implikanta prostego nie jest również objęta polem innego implikanta prostego, to taki implikant prosty jest ISTOTNY. Zbiór tych implikantów tworzy JĄDRO.*

Jądro formy stanowią zatem implikanty $ab'd'$ i $a'd$.

► *Należy wybrać minimalną liczbę pozostałych implikantów prostych, które pokrywają wszystkie pola nieobjęte przez jądro.*

Wystarczy jeden implikant abc'.

Trzy sklejenia ► forma minimalna o trzech termach:

$$y = ab'd' + a'd + abc' \qquad Z = 3 + 2\cdot 3 + 2 = 11$$

Implikanty proste na siatce Karnaugha

W interpretacji tablic Karnaugha implikant prosty odpowiada grupie jedynek (i kresek), której nie można powiększyć.

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

Implikant $\bar{x}_1x_3x_4$

Prosty implikant $\bar{x}_1x_3$

To nie jest Implikant!

Sklejanie zer

Na siatce Karnaugha można także sklejać zera tworząc kanoniczną postać iloczynową funkcji logicznej. Pamiętać należy o odwrotnym wartościowaniu

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

A, B, C, D

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

zastąp F przez F', 0 zamień na 1 i na odwrót

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

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

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

Układy wielowyjściowe

W przypadku układów o wielu wyjściach dobrze jest poszukiwać implikantów wspólnych dla form wyjściowych, a nie poszukiwać implikantów prostych dla każdej formy z osobna.

$y_1 = \Sigma(2,3,5,7,8,9,10,11,13,15)$

$y_2 = \Sigma(2,3,5,6,7,10,11,14,15)$

$y_3 = \Sigma(6,7,8,9,13,14,15)$

ab \ cd	00	01	11	10
00	0	1	3	2
01	4	5	7	6
11	12	13	15	14
10	8	9	11	10

ab \ cd	00	01	11	10
00			1	1
01		1	1	
11		1	1	
10	1	1	1	1

$y_1 = a\bar{b} + bd + \bar{b}c$

ab \ cd	00	01	11	10
00			1	1
01		1	1	1
11			1	1
10			1	1

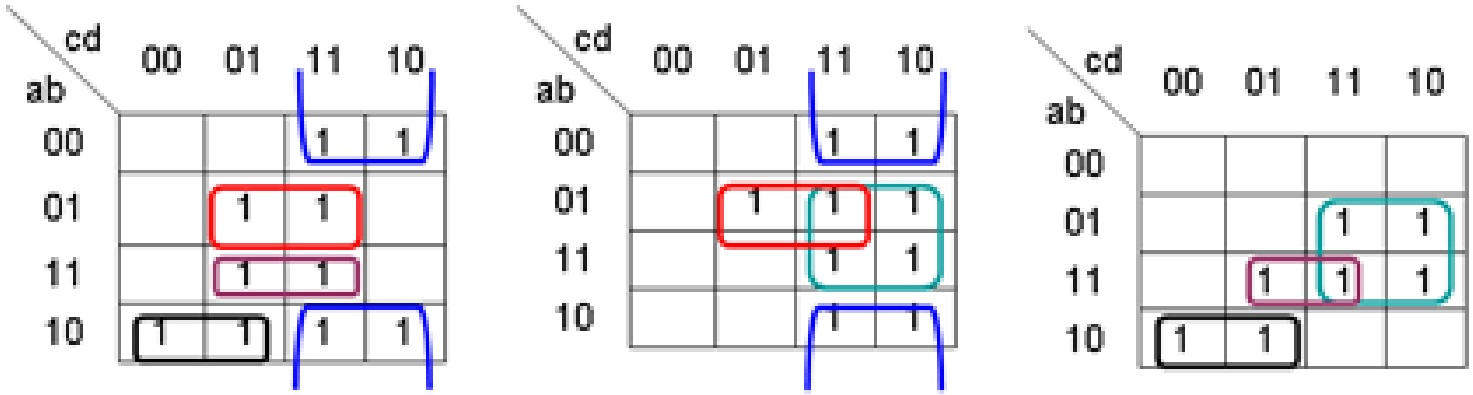
$y_2 = c + \bar{a}bd$

ab \ cd	00	01	11	10
00				
01			1	1
11		1	1	1
10	1	1		

$y_3 = bc + a\bar{c}d + a\bar{b}\bar{c}$

7 bramek AND

Układy wielowyjściowe



$$y_1 = \overset{1}{\bar{b}c} + \overset{2}{\bar{a}bd} + \overset{3}{abd} + \overset{4}{ab\bar{c}}$$

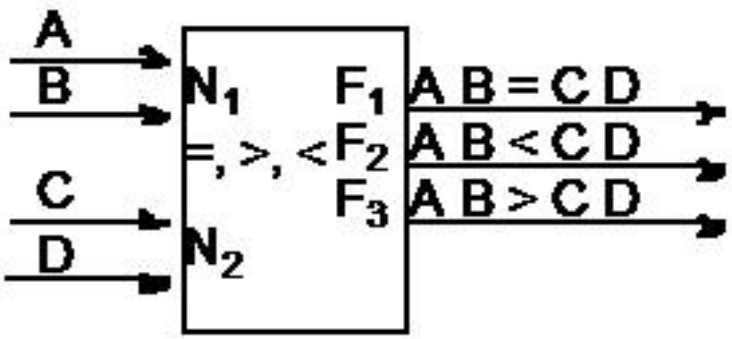
$$y_2 = \overset{1}{\bar{b}c} + \overset{2}{\bar{a}bd} + \overset{5}{bc}$$

$$y_3 = \overset{3}{abd} + \overset{4}{ab\bar{c}} + \overset{5}{bc}$$

5 bramek AND

... a poprzednio było 7 bramek AND!!!

Ex. Komparator 2-bitowy



Schemat blokowy
i tablica prawdy

A	B	C	D	F_1	F_2	F_3
0	0	0	0	1	0	0
		0	1	0	1	0
		1	0	0	1	0
		1	1	0	1	0
0	1	0	0	0	0	1
		0	1	1	0	0
		1	0	0	1	0
		1	1	0	1	0
1	0	0	0	0	0	1
		0	1	0	0	1
		1	0	1	0	0
		1	1	0	1	0
1	1	0	0	0	0	1
		0	1	0	0	1
		1	0	0	0	1
		1	1	1	0	0

4-wejściowe mapy Karnaugha
dla każdej z 3 funkcji wyjściowych

Ex. Komparator 2-bitowy

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

K-map for F_1

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

K-map for F_2

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

K-map for F_3

$$F1 = A' B' C' D' + A' B C' D + A B C D + A B' C D'$$

$$F2 = A' B' D + B' C D + A' C$$

$$F3 = B C' D' + A C' + A B D'$$

= (A xnor C) (B xnor D) ← forma prostsza, ale nie w postaci kanonicznej
1 na przekątnych map Karnaugha odpowiadają funkcjom XOR/XNOR

Algorytm minimalizacji na siatce K.

Krok 1:
Wybierz element na mapie Karnaugh nie powiązany z żadnym implikantem (nie sklejony).

Krok 2:
Znajdź wszystkie sąsiedzkie pokrycia danego pola o maksymalnej powierzchni (2^n), czyli implikanty proste.

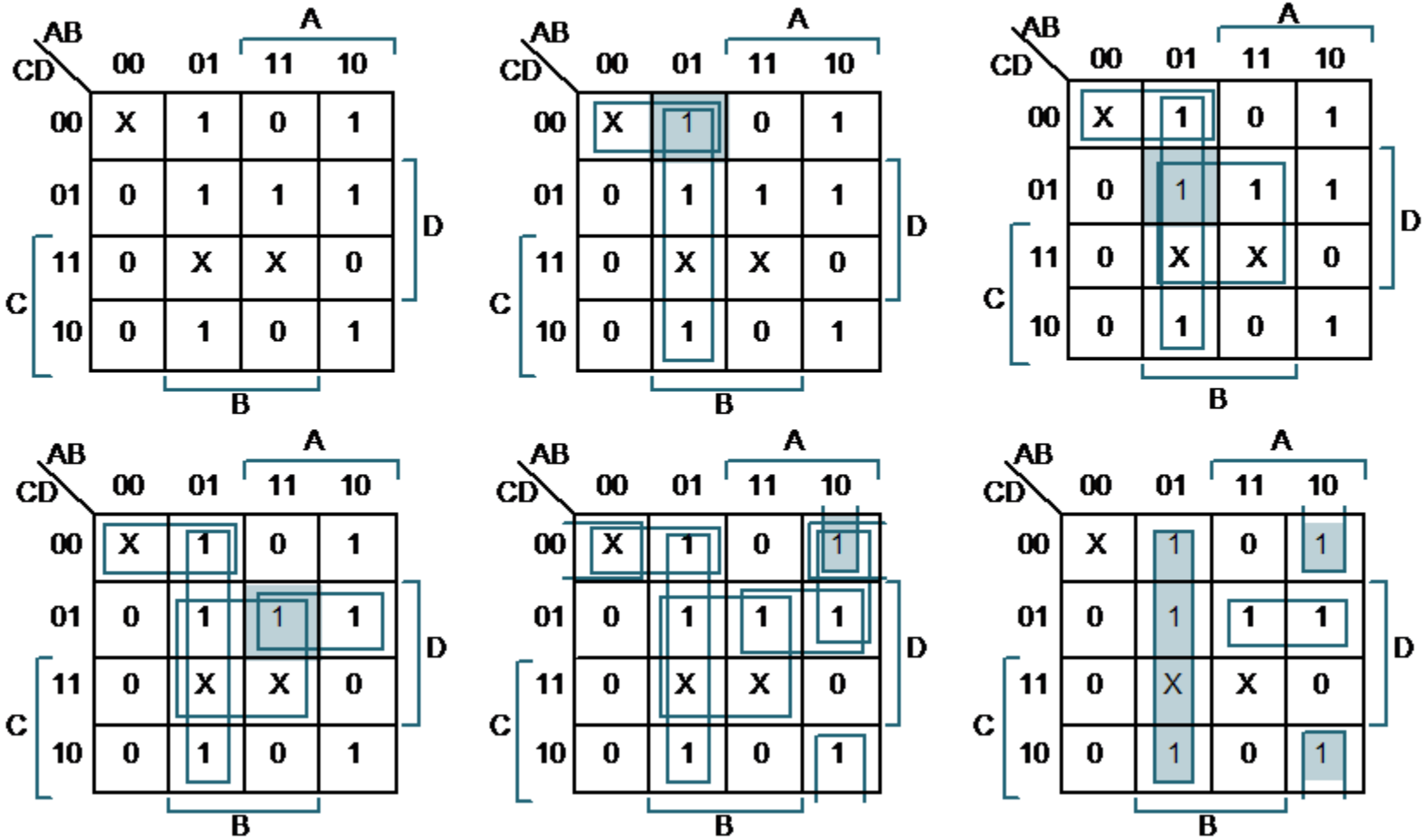
Powtórz krok 1 i 2 aż znajdziesz wszystkie implikanty proste.

Krok 3:
Wyszukaj wszystkie istotne implikanty proste (pola pokrywane tylko przez 1 implikant prosty).

Krok 4:
Pola niepokryte przez istotne implikanty proste pokryj najmniejszą możliwą liczbą implikantów prostych.

Ex. Budowanie minimalnego pokrycia

$$f(A,B,C,D) = \sum(4,5,6,8,9,10,13(0,7,15))$$



Metoda Quine'a-McCluskeya

Metoda Q-McC jest metodą algorytmiczną, dającą się zaimplementować numerycznie. Przebieg minimalizacji:

Selekcja prostych implikantów na grupy w zależności od ilości '1'

↓

Wyszukanie wszystkich możliwych par różniących się zawartością tylko 1 bitu

↓

Wybór optymalnego pokrycia minimalnego

Ex. 1. Minimalizacja Q-McC

Zminimalizujmy funkcję 4 zmiennych:

$$f(x_1,x_2,x_3,x_4) = \sum(3,7,10,11,15)$$

Selekcja prostych implikantów na grupy w zależności od liczby 1

0011	(3)
1010	(10)
0111	(7)
1011	(11)
1111	(15)

Wyszukanie wszystkich możliwych par różniących się zawartością na 1 pozycji

0x11	(3,7)
x011	(3,11)
101x	(10,11)
x111	(7,15)
1x11	(11,15)

xx11	**(3,7,11,15)**
	(3,11,7,15)

Poszukiwane rozwiązanie:

$$f(x_1,x_2,x_3,x_4) = x_1x_2'x_3 + x_3x_4$$

Ex. 2. Minimalizacja Q-McC

Zminimalizujmy funkcję 4 zmiennych:

$$f(x_1,x_2,x_3,x_4) = \sum(1,3,4,6,7,12,14,15)$$

Selekcja prostych implikantów na grupy w zależności od liczby 1

~~0001~~	(1)
~~0100~~	(4)
~~0011~~	(3)
~~0110~~	(6)
~~1100~~	(12)
~~0111~~	(7)
~~1110~~	(14)
~~1111~~	(15)

Wyszukanie wszystkich możliwych par różniących się zawartością na 1 pozycji

00x1	(1,3)
~~01x0~~	(4,6)
~~x100~~	(4,12)
0x11	(3,7)
~~011x~~	(6,7)
~~x110~~	(6,14)
~~11x0~~	(12,14)
~~x111~~	(7,15)
~~111x~~	(14,15)

x1x0	(4,6,12,14)
x11x	(6,7,14,15)

Następny krok – selekcja najlepszego minimalnego pokrycia:

Ex. 2. Minimalizacja Q-McC

Generacja tablicy Quine'a

	x1x0	x11x	00x1	0x11
0001	—	—	(v)	—
0100	(v)	—	—	—
0011	—	—	v	v
0110	v	v	—	—
1100	(v)	—	—	—
0111	—	v	—	v
1110	v	v	—	—
1111	—	(v)	—	—

Poszukiwane pokrycie minimalne:

$$f(x_1,x_2,x_3,x_4) = x_2x_4' + x_2x_3 + x_1'x_2'x_4$$