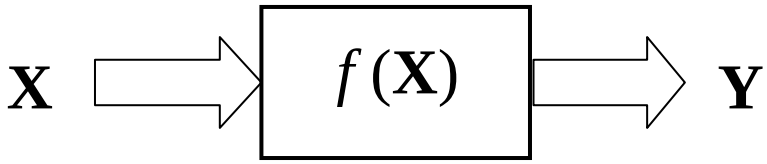
Temat 2. Synteza układów kombinacyjnych z bramek logicznych

Spis treści do tematu 2

2.1. Wprowadzenie

Układ logiczny (ang. *logic circuit*)
– zbiór funktorów logicznych połączonych ze sobą w określony sposób, tak aby realizować przyjętą funkcję.

Układ kombinacyjny (ang. *combinational logic circuit*)
– szczególny typ układu logicznego w którym bieżący stan wyjść **Y** układu jest jednoznacznie określony przez bieżący stan jego wejść **X**



gdzie:
X - wektor wejść złożony ze stanów sygnałów wejściowych $x_1, x_2, \ldots, x_n$,

$$\mathbf{X} = (x_1, x_2, \ldots, x_n) \tag{2.1}$$

Y - wektor wyjść złożony ze stanów sygnałów wyjściowych $y_1, y_2, \ldots, y_m$,

$$\mathbf{Y} = (y_1, y_2, \ldots, y_m) \tag{2.2}$$

f – funkcja boolowska (przełączająca, …)

$$\mathbf{Y} = f(\mathbf{X}) \tag{2.3}$$

Etapy projektowania układu kombinacyjnego

1). Opis problemu/układu kombinacyjnego.
2). Ustalenie sygnałów wejściowych x_i i wyjściowych y_i
(jeśli nie podano bezpośrednio w opisie).
3). Ustalenie funkcji przełączającej f (jeśli nie podano bezpośrednio w opisie)
- tablica prawdy,
- zbiór jedynek funkcji F^1 albo zer F^0,
- postać kanoniczna sumy albo postać kanoniczna iloczynu,
4). Znalezienie najprostszej (minimalnej) postaci funkcji.
5). Schemat układu.
6). Ocena kosztu układu i ewentualnie korekta schematu.

<u>Metody minimalizacji wyrażeń logicznych:</u>
- przekształcenia na podst. praw algebry Boole'a,
- metoda tablic (siatek) Karnaugha,
- metoda Quine'a-McCluskey'a (Q-MC),
- metoda Kazakowa,
- metoda Tablic Niezgodności (TN),
- metoda bezpośredniego przeszukiwania (BP),
- inne.

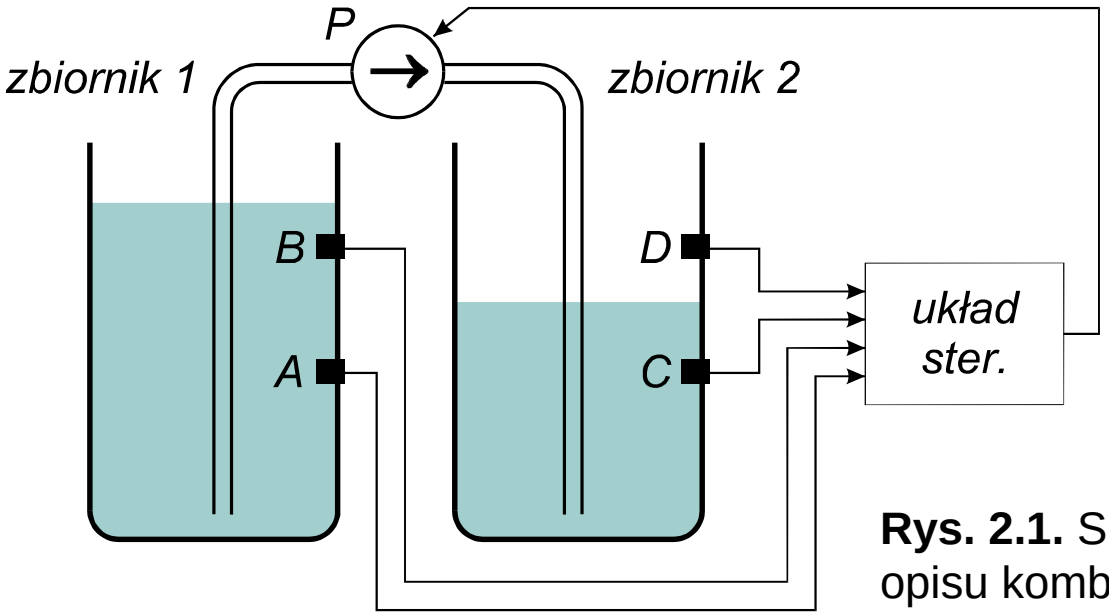
Opis słowny układu kombinacyjnego

W instalacji przemysłowej dane są dwa podobne zbiorniki, z których każdy zawiera po dwa czujniki poziomu cieczy. Czujnik znajdujący się powyżej poziomu cieczy wysyła sygnał o wartości 0, natomiast sygnał 1 oznacza zanurzenie czujnika w cieczy.

Zaprojektować układ, który steruje pompą *P*, według następujących reguł:

- jeżeli w zb. 1 jest więcej cieczy niż w zbiorniku 2, to włącz pompę,
- jeżeli w zb. 1 jest mniej cieczy lub poziomów nie można odróżnić, to wyłącz pompę.

Odbiór cieczy ze zbiornika 2 zapewnia inny układ, który nie jest przedmiotem tego projektu. Oznaczenia czujników przedstawiono na rysunku:



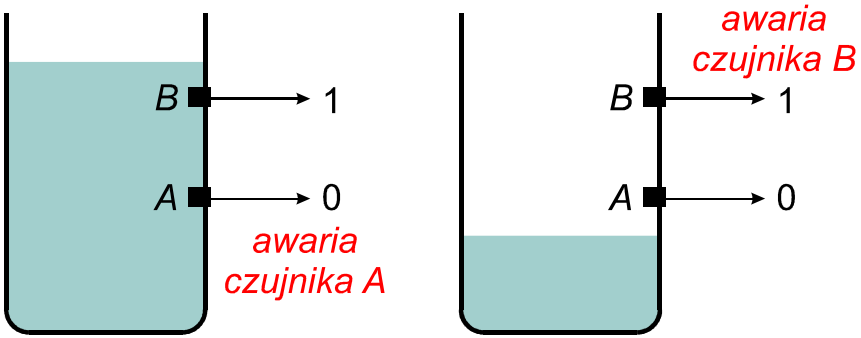
Rys. 2.1. Schemat instalacji do słownego opisu kombinacyjnego układu sterowania.

Opis słowny układu nie zawsze zapewnia projektantowi pełne i jednoznaczne dane. Zleceniodawca może nie mieć wystarczającej wiedzy technicznej, by uzupełnić opis.

W opisanym układzie może dojść do awarii czujnika, która przejawi się poprzez załączenie czujnika leżącego wyżej przy niezałączonym czujniku leżącym niżej. Potrzebna jest decyzja dotycząca zachowania się układu sterowania w sytuacji awaryjnej, np.:

- nie rozważamy stanów awaryjnych (wariant najprostszy),
- wyłączenie pompy w każdej możliwej do wykrycia sytuacji awaryjnej,
- dodajemy do układu sterowania wyjście do sygnalizowania stanów awaryjnych dla nadzoru technicznego,
- dodajemy do układu sterowania dwa wyjścia do sygnalizowania stanów awaryjnych osobno dla zbiornika *A* i osobno dla *B*,
- inne.

Dalej rozważymy najprostszy wariant



Rys. 2.2. Przykładowe sytuacje, w których możliwe jest wykrycie awarii czujnika poziomu cieczy.

Ustalenie sygnałów wejściowych i wyjściowych

Dla rozważanego problemu sygnały wejściowe *A*, *B*, *C*, *D* wynikają wprost z opisu słownego układu. Załóżmy brak dodatkowych nieoczywistych wejść.

Jedno wyjście układu *P* wynika z analizy opisu słownego oraz dodatkowego założenia projektowego. Przyjęcie do realizacji bardziej złożonego wariantu może wymagać dodatkowych wyjść.

Ustalenie funkcji przełączającej

Odpowiedź układu na poszczególne kombinacje stanów wejściowych wynika bezpośrednio z analizy opisu i dodatkowych założeń projektowych.

Tabela 2.1. Tablica prawdy dla układu sterowania z rys. 2.1.

A	B	C	D	P
0	0	0	0	0
0	0	0	1	–
0	0	1	0	0
0	0	1	1	0
0	1	0	0	–
0	1	0	1	–
0	1	1	0	–
0	1	1	1	–

A	B	C	D	P
1	0	0	0	1
1	0	0	1	–
1	0	1	0	0
1	0	1	1	0
1	1	0	0	1
1	1	0	1	–
1	1	1	0	1
1	1	1	1	0

gdzie symbolem (–) oznaczono stany nieokreślone.

Postać kanoniczna sumy, to suma iloczynów pełnych zmiennych lub ich negacji. Można ją otrzymać wprost z tablicy prawdy, biorąc pod uwagę jedynie te wiersze, dla których wartość funkcji $P = 1$ i przypisując:
wartościom 1 argumentu – zmienne niezanegowane (A, B, C, D),
wartościom 0 argumentu – zmienne zanegowane ($\overline{A}$, $\overline{B}$, $\overline{C}$, $\overline{D}$)

$$P = A\overline{B}\,\overline{C}\,\overline{D} + AB\overline{C}\;\overline{D} + ABC\overline{D}. \tag{2.4}$$

Postać kanoniczna iloczynu, to iloczyn sum pełnych ze zmiennych lub ich negacji. Każda suma pełna odpowiada jednej linii w tablicy prawdy, dla której funkcja przyjmuje wartość zero

$$P = (A+B+C+D)(A+B+\overline{C}+D)(A+B+\overline{C}+\overline{D})\cdot \tag{2.5}$$
$$\cdot(\overline{A}+B+\overline{C}+D)(\overline{A}+B+\overline{C}+\overline{D})(\overline{A}+\overline{B}+\overline{C}+\overline{D}).$$

Każą funkcję logiczną można przedstawić w postaci kanonicznej, jednakże zazwyczaj nie jest to postać minimalna tej funkcji.

Przykład minimalizacji postaci kanonicznej sumy przez bezpośrednie wykorzystanie praw algebry Boole'a,

$$
\begin{aligned}
P &= A\overline{B}\,\overline{C}\,\overline{D} + AB\overline{C}\,\overline{D} + ABC\overline{D} = && \text{przemienność iloczynu logicznego, rozdzielność iloczynu logicznego względem sumy} \\
&= (\overline{B} + B)A\overline{C}\,\overline{D} + ABC\overline{D} = && \text{prawo wyłączonego środka} \\
&= A\overline{C}\,\overline{D} + ABC\overline{D} = && \text{przemienność iloczynu logicznego, rozdzielność iloczynu logicznego względem sumy} \\
&= (\overline{C} + BC)A\overline{D} = && \text{tożsamość pomocnicza (Shannona)} \\
&= (B + \overline{C})A\overline{D} && (2.6)
\end{aligned}
$$

Funkcję nazywamy zupełną jeżeli jest jednoznacznie określona dla wszystkich kombinacji swoich argumentów.

Funkcję nieokreśloną dla niektórych kombinacji argumentów nazywamy funkcją niezupełną (np. funkcja z tabeli 3.1).

Uwaga:
Postać kanoniczna sumy i postać kanoniczna iloczynu dla funkcji niezupełnej nie są równoważne.

2.1. Metoda tablic Karnaugha

Metoda tablic Karanauga wykorzystuje zdolność ludzi do rozpoznawania geometrycznych wzorów. Tablica Karanugha jest to specyficznie ułożona tablica prawdy

Wartości argumentów uporządkowane wg. kodu Graya (BRGC), w którym każde dwie kolejne wartości różnią się dokładnie jednym bitem.

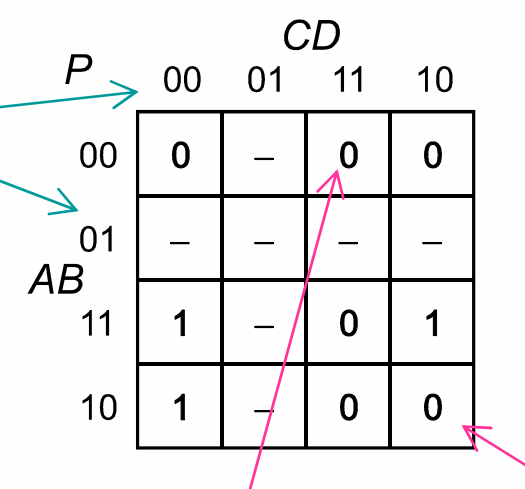


Tabela 2.2. Tablica Karnaugha odpowiadająca tabeli prawdy 2.1.

Etap 1. Kopiowanie danych do tablicy Karnaugha.

Kopia tabeli 2.1 (od zadania z pompą).

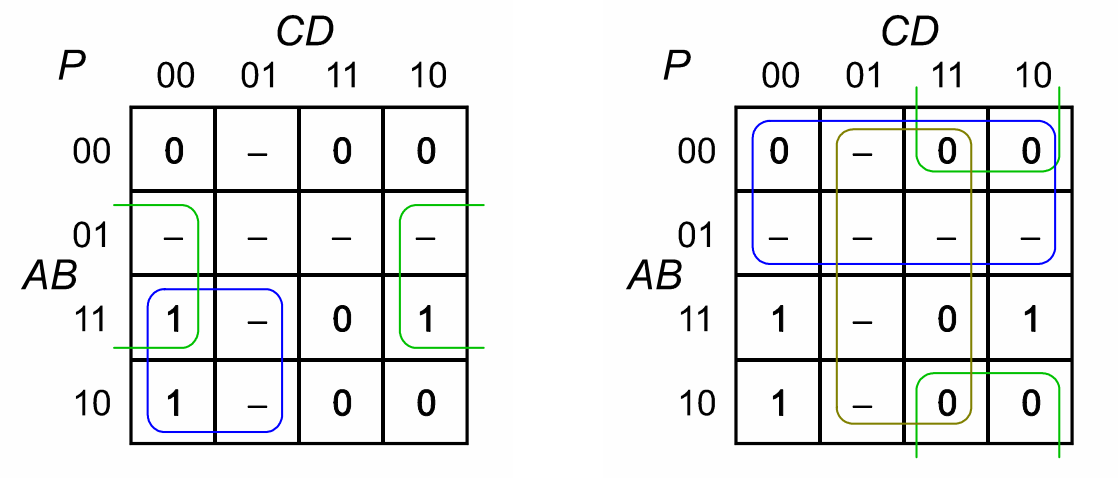
A	*B*	*C*	*D*	*P*
0	0	0	0	0
0	0	0	1	–
0	0	1	0	0
0	0	1	1	0
0	1	0	0	–
0	1	0	1	–
0	1	1	0	–
0	1	1	1	–

A	*B*	*C*	*D*	*P*
1	0	0	0	1
1	0	0	1	–
1	0	1	0	0
1	0	1	1	0
1	1	0	0	1
1	1	0	1	–
1	1	1	0	1
1	1	1	1	0

Etap 2. Grupowanie „1” (lub „0”).

Kolejnym etapem jest grupowanie jedynek (alternatywnie zer) w sąsiednich komórkach.
Obowiązują następujące reguły:

- tworzymy tylko prostokątne obszary zawierające 2^n sąsiednich komórek (n - liczba naturalna),
- wszystkie „1” (alternatywne „0”) trzeba włączyć do któregoś obszaru,
- daną komórkę można włączyć do więcej niż jednego obszaru,
- komórki ze stanem nieokreślonym (–) można ale nie trzeba łączyć z jedynkami albo zerami.
- tablica Karanugha nie ma brzegów, przeciwlegle krawędzie widoczne na rysunku tablicy traktujemy jak sklejone ze sobą.



Rys. 2.3. Przykłady optymalnego grupowania jedynek oraz zer na tablicy Karnaugha dotyczącej zadania z pompą.

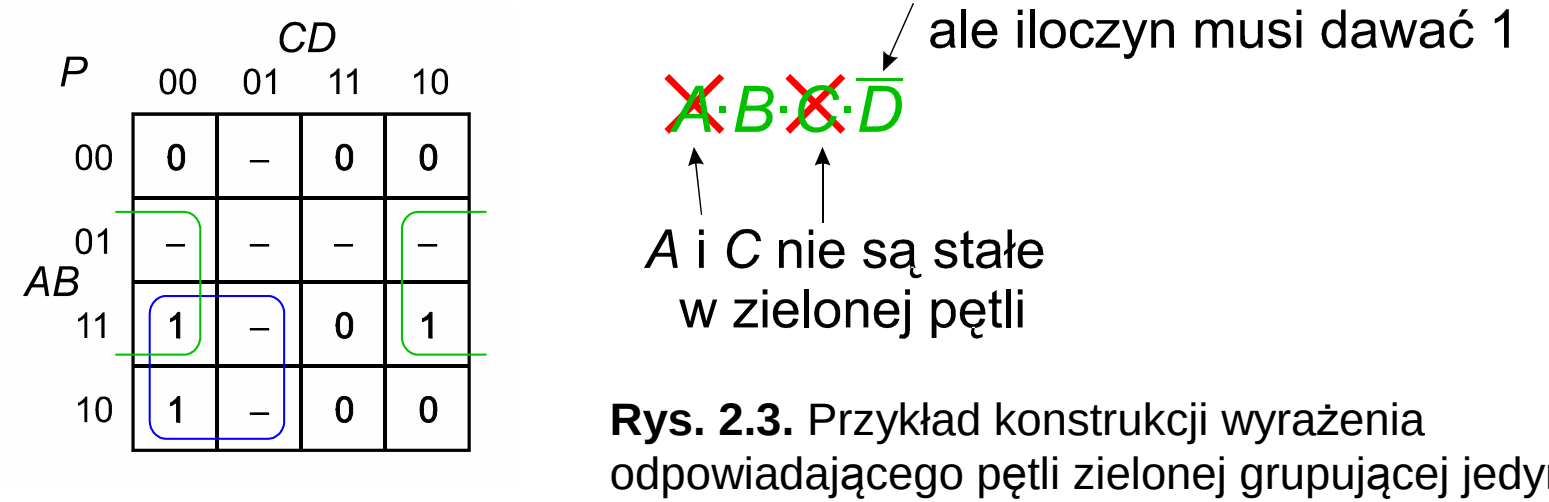
Zasady OPTYMALNEGO grupowania:

- staramy się zgrupować wszystkie „1” (alternatywnie „0”) wewnątrz możliwie najmniejszej liczby pętli,
- pętle o większych rozmiarach są korzystniejsze od pętli mniejszych.

Etap 3. Synteza funkcji logicznej odpowiadającej zakreślonym obszarom.

Przypadek grupowania jedynek

- Każda grupa jedynek i znaków (–) odpowiada iloczynowi tych spośród argumentów *A*, *B*, *C*, *D*, które mają ustaloną wartość na wszystkich komórkach w pętli.
- Zmienne do iloczynu podstawiamy wprost albo zanegowane, tak by iloczyn dawał wynik „1” dla wszystkich komórek w odpowiadającej mu pętli.
- Kompletną funkcję logiczną budujemy jako sumę iloczynów odpowiadających wszystkim pętlom.



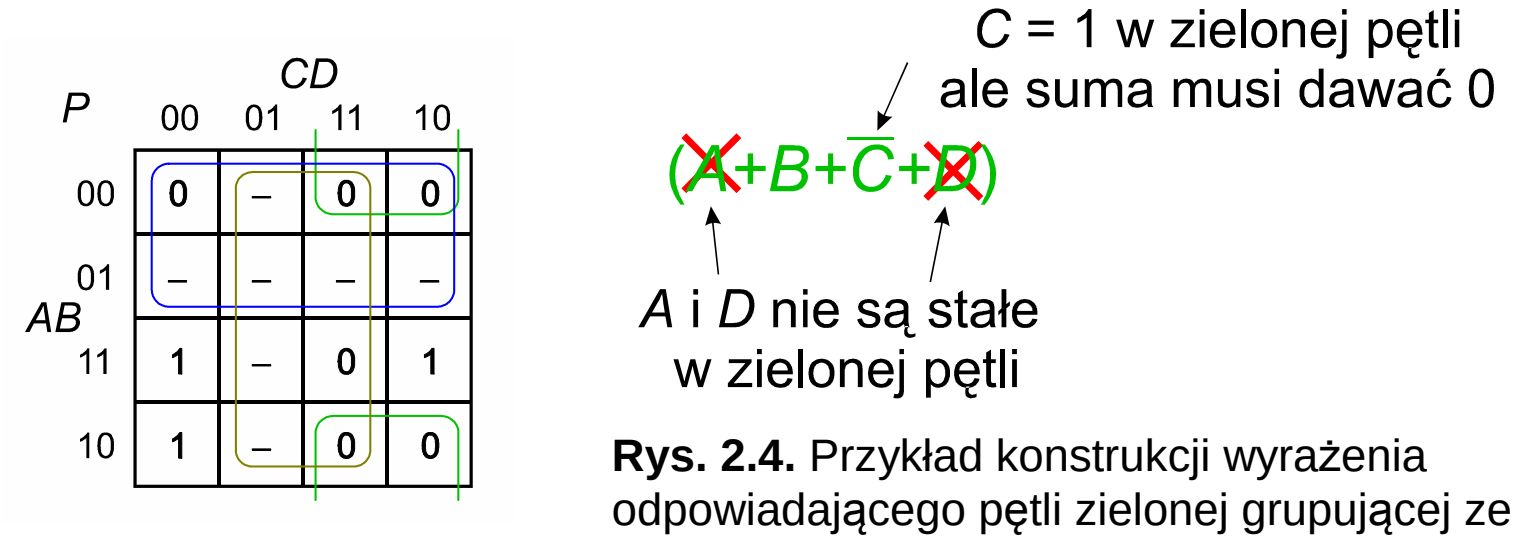
Rys. 2.3. Przykład konstrukcji wyrażenia odpowiadającego pętli zielonej grupującej jedynki.

Razem dla obu zaznaczonych pętli otrzymujemy

$$P = A\overline{C} + B\overline{D} \qquad (2.7)$$

Przypadek grupowania zer

- Każda grupa zer i znaków (–) odpowiada sumie tych spośród argumentów *A*, *B*, *C*, *D*, które mają ustaloną wartość na wszystkich komórkach w pętli.
- Zmienne do sumy podstawiamy wprost albo zanegowane, tak by suma dawała wynik „0” dla wszystkich komórek w odpowiadającej jej pętli.
- Kompletną funkcję logiczną budujemy jako iloczyn sum odpowiadających wszystkim pętlom.



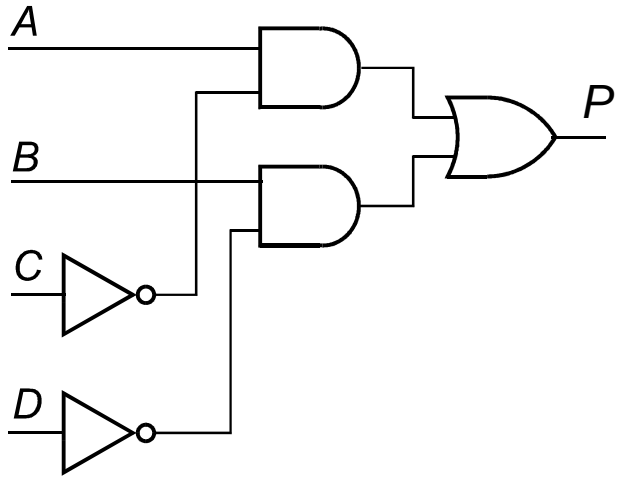
Rys. 2.4. Przykład konstrukcji wyrażenia odpowiadającego pętli zielonej grupującej zera.

Razem dla wszystkich zaznaczonych pętli otrzymujemy

$$P = (B + \overline{C})A\overline{D} \tag{2.8}$$

Porównanie schematów układów kombinacyjnych

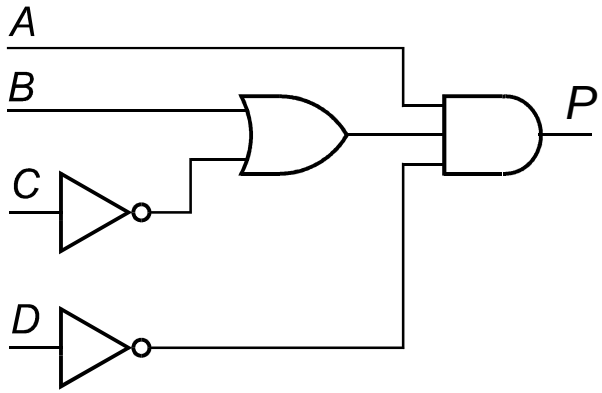
Przypadek grupowania jedynek



Układ zbudowano na podstawie funkcji (2.7):

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

Przypadek grupowania zer



Układ zbudowano na podstawie funkcji (2.8):

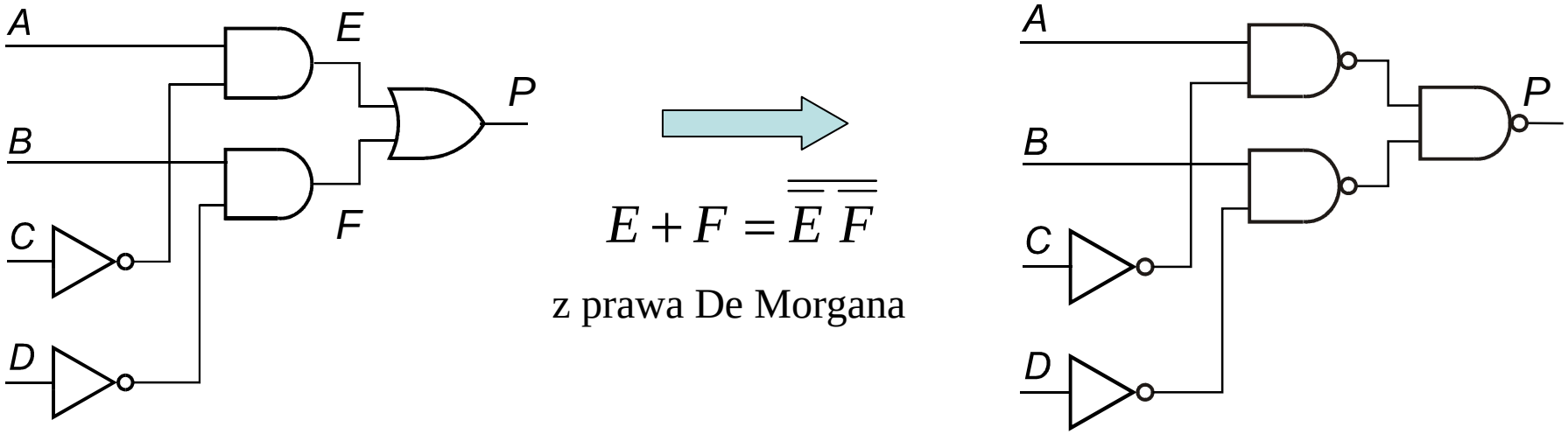
$$P = (B + \overline{C})A\overline{D}$$

Rys. 2.5. Schematy układów sterowania pompą otrzymane metodą tablic Karnaugha dla przypadków grupowania jedynek oraz grupowania zer.

Ocena kosztu układu

Bezpośrednia realizacja układu odpowiadającego funkcji (2.7) wymaga użycia 3-ech układów scalonych zawierających bramki NOT, AND oraz OR.

Po korekcie układ wymaga użycia 2-ch układów scalonych z bramkami NOT i NAND.

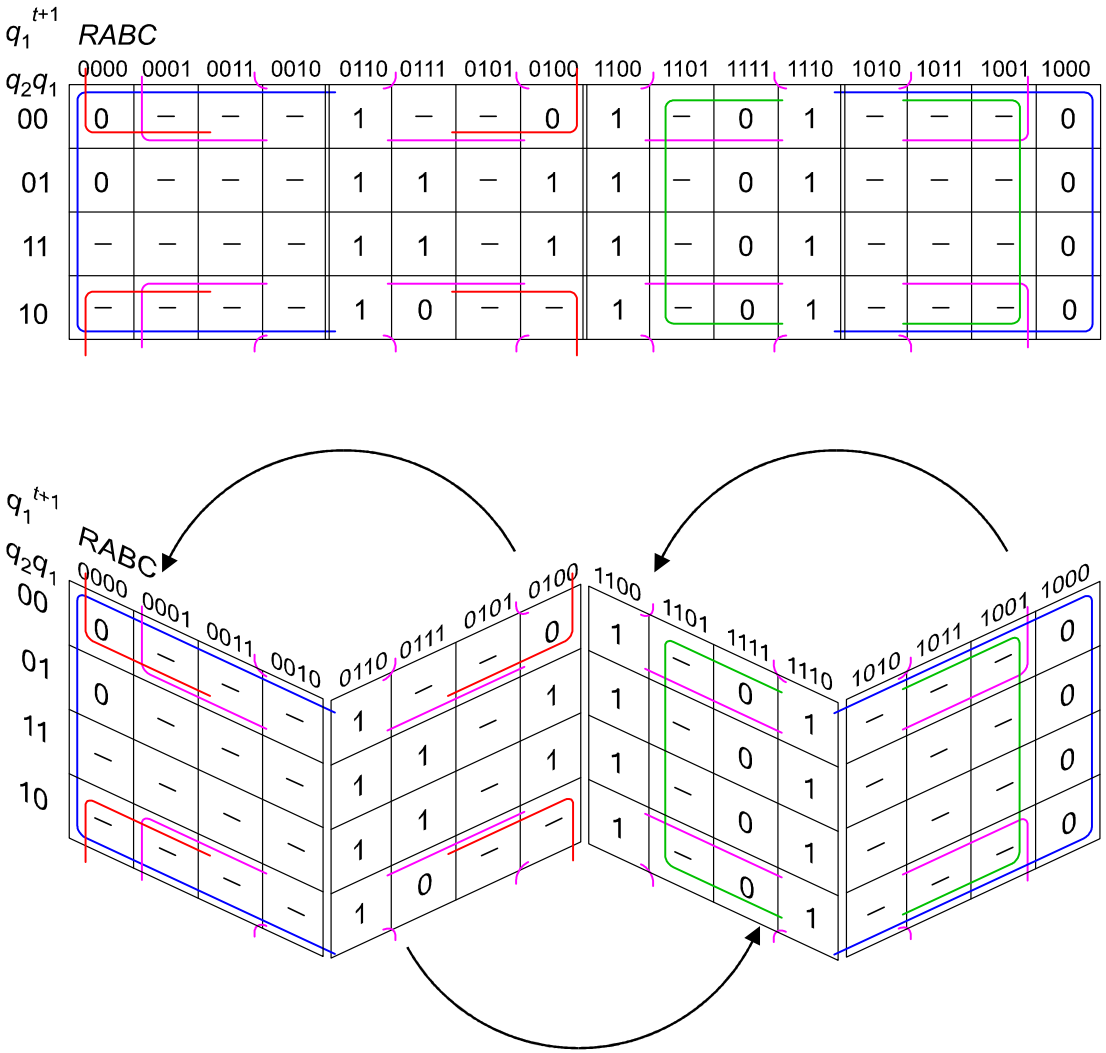


Rys. 2.6. Porównanie schematów przed i po ocenie kosztów układu.

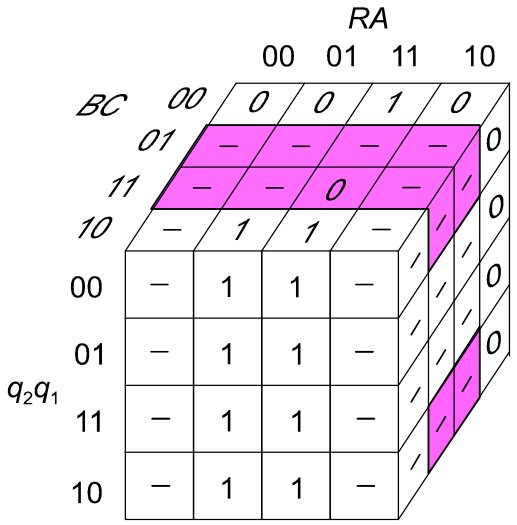
Jeżeli układ ma być zrealizowany z bramek TTL, to zastąpienie bramek NOT przez bramki NAND nie przyniesie oszczędności, bo 2-wejściowe bramki NAND są oferowane po 4 bramki na układ scalony.

Tablice Karnaugha dla dużej liczby zmiennych

Dla 5 i więcej zmiennych nie można na płaszczyźnie narysować obok siebie wszystkich komórek sąsiadujących w tablicy Karnaugha. Tablicę przedstawiamy wtedy w postaci rozciętej na kilka płaszczyzn 4 × 4.



Rys. 2.7. Przykład transformacji tablicy Karnaugha dla 6-ciu zmiennych między rozkładem na płaszczyźnie a kostką w 3-trzech wymiarach przestrzennych.



Kostka 4 × 4 × 4. Ograniczono się do grupowania komórek z pętli liliowej.

2.3. Metoda Quine’a-McCluskey’a (Q-MC)

Tabela 2.2. Porównanie cech metody tablic Karnaugha i metody Quine’a-McCluskey’a (Q-MC)

Cecha	Metoda Karnaugha	Metoda Q-MC
Przystępność dla człowieka	czytelna forma graficzna	mało przystępna
Opis w algorytmie komputerowym	trudny	względnie łatwy
liczba zmiennych	powyżej 6 wymagana wyobraźnia w przestrzeniach >3 wymiarowych	algorytm uniwersalny ale dla dużej liczby zmiennych koszt wykonania rośnie bardzo szybko
minimalizacja funkcji niezupełnych	łatwa	przypadki stanów nieokreślonych nie są uwzględnione; wstępnie łączymy (–) z F^1 albo F^0

Przykład minimalizacji funkcji metodą Q-MC

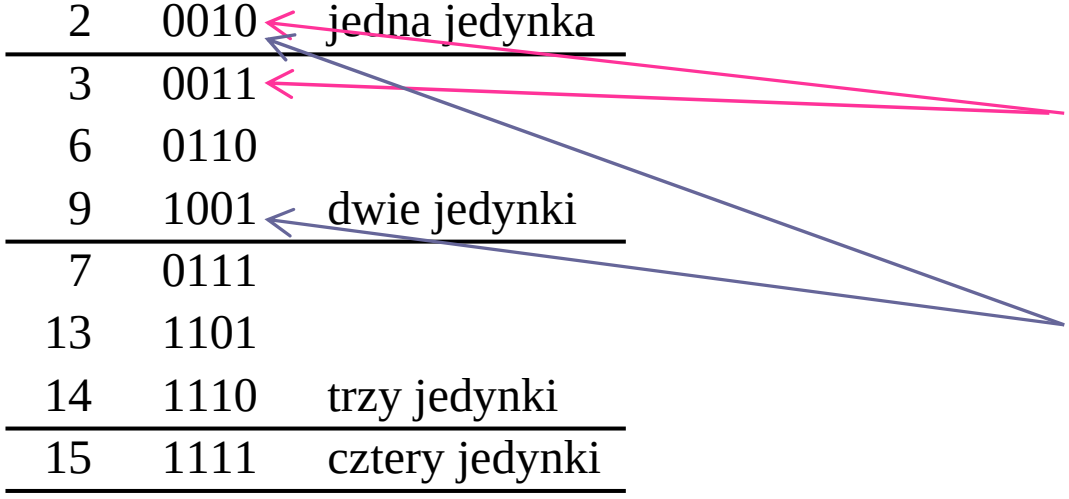
Niech dana będzie funkcja logiczna $F(X, Y, Z, W)$ określona zbiorem jedynek $F^1=\{2,3,6,7,9,13,14,15\}$. Rozwiązanie przebiega w następujących krokach:

1). Wypisujemy wszystkie wektory zbioru F^1

F^1 (dziesiętnie)	$XYZW$ (binarnie)
2	0010
3	0011
6	0110
7	0111
9	1001
13	1101
14	1110
15	1111

2). Grupujemy wektory wg. liczby jedynek

2	0010	jedna jedynka
3	0011	
6	0110	
9	1001	dwie jedynki
7	0111	
13	1101	
14	1110	trzy jedynki
15	1111	cztery jedynki



różnica na tylko jednej pozycji
- kombinacje do połączenia

więcej różnic
- nie można połączyć

3). Łączenie kombinacji, które na jednej pozycji mają różne cyfry 0 i 1.

2,3	001-	*u*	
2,6	0-10	*u*	jedna jedynka
3,7	0-11	*u*	
6,7	011-	*u*	
6,14	-110	*u*	
9,13	1-01		dwie jedynki
7,15	-111	*u*	
13,15	11-1		
14,15	111-	*u*	trzy jedynki

u – użyte (pochłonięte) podczas łączenia

Kopia tabeli z pkt. 2

2	0010	*u*	jedna jedynka
3	0011	*u*	
6	0110	*u*	
9	1001	*u*	dwie jedynki
7	0111	*u*	
13	1101	*u*	
14	1110	*u*	trzy jedynki
15	1111	*u*	cztery jedynki

u – użyte (pochłonięte) podczas łączenia

przykład łączenia

Każdy element może być sklejany dowolną liczbę razy

Zbiór prostych implikantów danej funkcji (elementów które nie uległy połączeniom)

Następny cykl łączenia

2,3,6,7	0-1-	jedna jedynka
6,7,14,15	-11-	dwie jedynki

nie ma już nic do połączenia

Powtarzamy sklejanie kombinacji aż do wyczerpania możliwości dalszego łączenia

4). Poszukiwanie minimalnego zbioru prostych implicentów.

Tworzymy tablicę pokrycia, której wiersze odpowiadają prostym implikanom G_i (znalezionym na poprzednim etapie), zaś kolumny wszystkim wektorom funkcji F^1 (czyli implikantom elementarnym). We wnętrzu tabeli stawiamy „1” tam gdzie wektor funkcji F^1 jest zgodny z implikantem.

$F^1(XYZW)$ / G_i		0010	0011	0110	0111	1001	1101	1110	1111
		2	3	6	7	9	13	14	15
0-1-	2,3,6,7	(1)	(1)	1	1				
-11-	6,7,14,15			1	1			(1)	1
1-01	9,13					(1)	1		
11-1	13,15						1		1

(1) - elementy konieczne, wchodzą w skład wierszy zasadniczych.

Najpierw wybieramy wiersze zasadnicze, których obecność jest konieczna. Następnie decydujemy o wyborze innych wierszy koniecznych do zapewnienia „1” we wszystkich kolumnach tabeli. W rozważonym przypadku wiersz 4 w tabeli pokrycia nie jest potrzebny. Funkcję budujemy analogicznie jak dla pętli na tablicy Karnaugha

$$F(X,Y,Z,W) = G_1 + G_2 + G_3 = \overline{X}Z + YZ + X\overline{Z}W \qquad (2.9)$$

Metodę Q-MC można zastosować także do zbioru zer F^0 danej funkcji (zmiany analogiczne jak w metodzie tablic Karanugha).

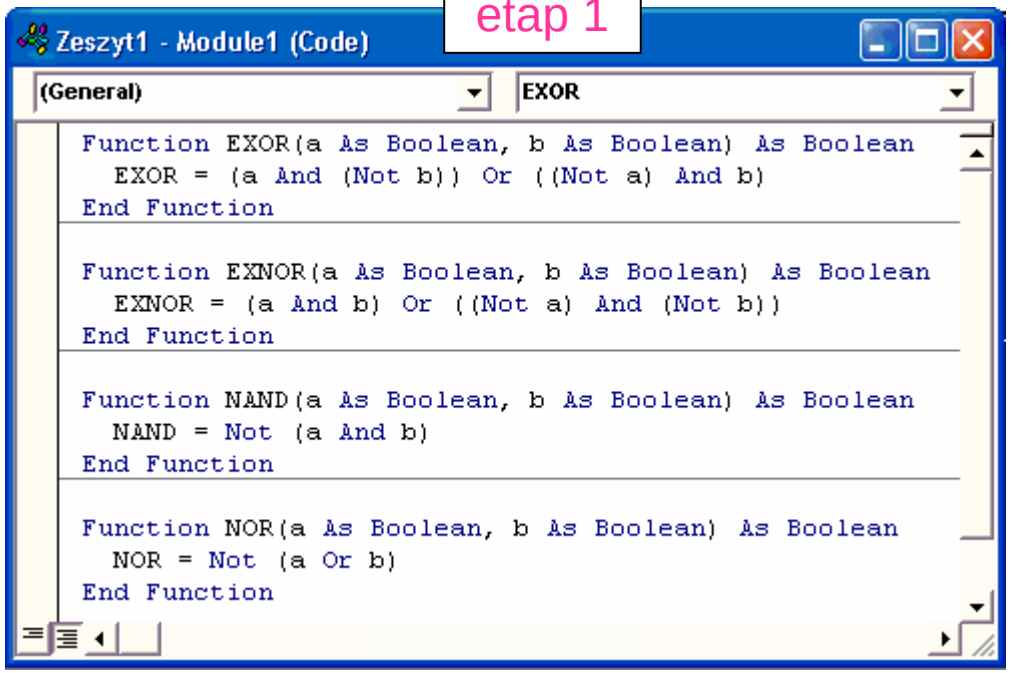
Zastosowanie arkusza MS Excel do testowania wyrażeń logicznych

Arkusz kalkulacyjny Excel posiada tylko 3 podstawowe predefiniowane funkcje Boole'a:

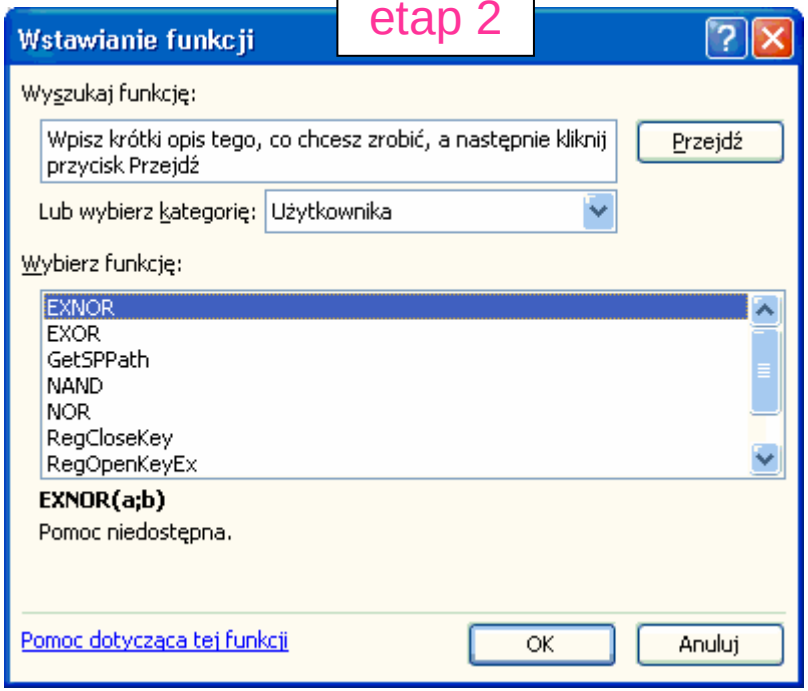
funkcja Excel PL	równoważna bramka log.
=nie(a1)	NOT
=lub(a1; a2; …)	OR
=oraz(a1; a2; …)	AND

Edytor makrodefinicji w języku Visual BASIC pozwala na łatwe rozszerzenie zestawu dostępnych funkcji. Przykład definicji funkcji EXOR, EXNOR, NAND i NOR:

etap 1



etap 2



2.4. Literatura

[1] P. Misiurewicz, *Układy automatyki cyfrowej*, Wydawnictwa Szkolne i Pedagogiczne, Warszawa, 1984.

[2] H. Kamionka-Mikuła, H. Małysiak, B. Pochopień, *Synteza i analiza układów cyfrowych*, Wydawnictwo Pracowni Komputerowej Jacka Skamierskiego, Gliwice 2006.

[3] W. Głocki, *Układy cyfrowe*, Wydawnictwa Szkolne i Pedagogiczne, Warszawa, 2008.

[4] A. Skorupski, *Podstawy techniki cyfrowej*, WKiŁ, Warszawa 2004.

[5] C. Zieliński, *Podstawy projektowania układów cyfrowych*, PWN, Warszawa 2003.

[6] W. Traczyk, *Układy cyfrowe. Podstawy teoretyczne i metody syntezy*, WNT, Warszawa 1986.

[7] M. Molski, *Wstęp do techniki cyfrowej*, WKiŁ, Warszawa 1989.

[8] R. Ćwirko, M. Rusek, W. Marciniak, Układy scalone w pytaniach i odpowiedziach, WNT, Warszawa, 1987.

[9] J. Kalisz, *Podstawy elektroniki cyfrowej*, WKiŁ, Warszawa 2002.

[10] P. Horowitz, W. Hill, Sztuka elektroniki, WKiŁ, Warszawa 2001.

[11] U. Tietze, Ch. Schenk, Układy półprzewodnikowe, WNT, Warszawa 2009.

[12] A. Barczak, J. Florek, T. Sydoruk, *Elektroniczne techniki cyfrowe*, VIZJA PRESS&IT Sp. z o.o., Warszawa 2006.