Realizacja logicznych układów kombinacyjnych z bramek.
Wprowadzenie do ćwiczeń E51 i E52.

Plan prezentacji:

1. Algebra Boole’a
2. Podstawowe bramki logiczne
3. Układy kombinacyjne - wprowadzenie
4. Synteza układu kombinacyjnego metodą tablic Karnaugha
   - przykład pełnego projektu.
5. Literatura

1. Algebra Boole’a

Definicja

Algebrą lub **algebrą abstrakcyjną** lub **algebrą uniwersalną** nazywamy strukturę w postaci

$$A = (U, f_1, f_2, \ldots, f_n), \tag{1.1}$$

gdzie:

U jest niepustym zbiorem,

$f_1, f_2, \ldots, f_n$ są działaniami określonymi w zbiorze U,
tzn. nie wyprowadzają poza zbiór U.

Algebra Boole’a

Algebra Boole’a została rozwinięta przez angielskiego matematyka George Boole’a (1815-1864) w celu analizy zdań logicznych które mogą zostać określone wyłącznie jako prawdziwe (1) albo fałszywe (0).

Podstawowe operacje algebry Boole’a to:

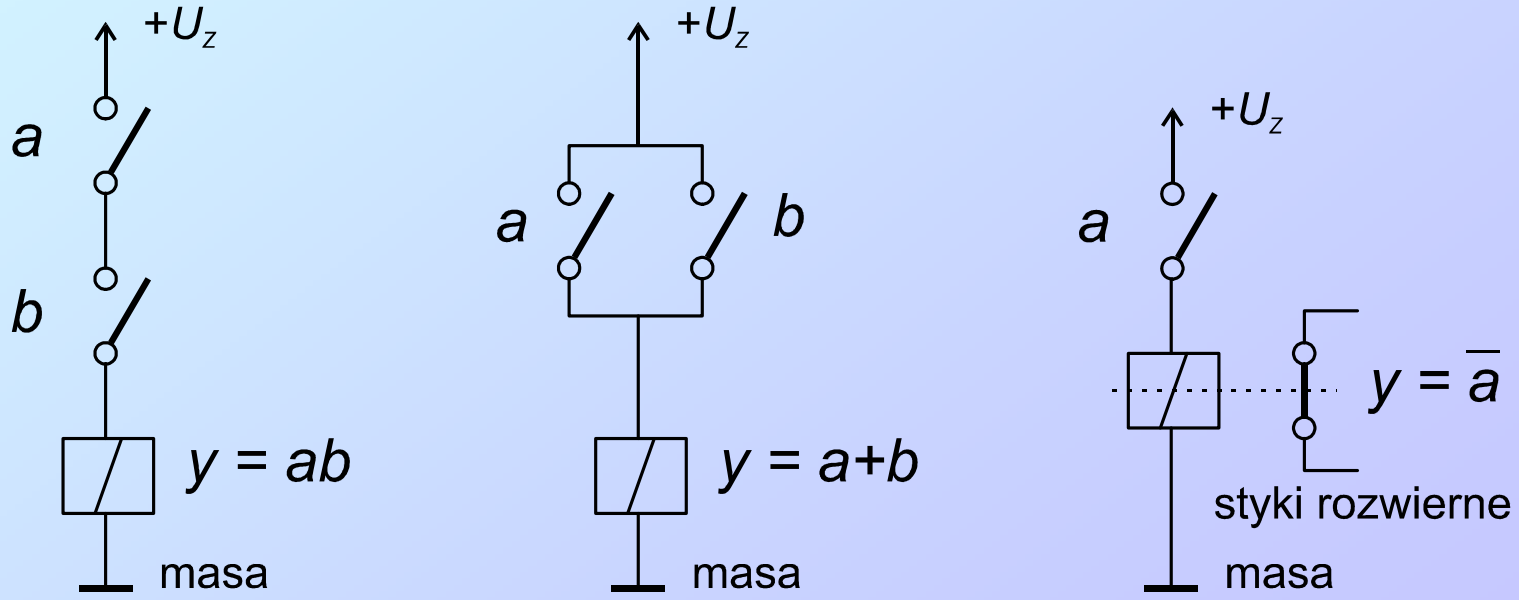
- iloczyn logiczny (koniunkcja) $y = a \cdot b = ab = a \wedge b$
- suma logiczna (alternatywa) $y = a + b = a \vee b$
- negacja $y = \overline{a} = \neg a = \sim a$

Algebra Boole’a jest zatem strukturą w postaci:

$$A = (U = \{0, 1\}, \neg, \wedge, \vee), \qquad (1.2)$$

Tabela 1.1. Definicje podstawowych operacji algebry Boole'a.

a	b	$a + b$	ab	$\neg a$
0	0	0	0	1
0	1	1	0	1
1	0	1	0	0
1	1	1	1	0



Rys. 1.1. Elektrotechniczna realizacja operacji algebry Boole'a.

Tabela 1.2. Podstawowe prawa i tożsamości algebry Boole'a.

Nazwa przekształcenia	Dla iloczynu logicznego	Dla sumy logicznej
Prawa przemienności	$a \cdot b = b \cdot a$	$a + b = b + a$
Prawa łączności	$a \cdot (b \cdot c) = (a \cdot b) \cdot c$	$a + (b + c) = (a + b) + c$
Prawa rozdzielności	$a \cdot (b + c) = a \cdot b + a \cdot c$	$a + b \cdot c = (a + b) \cdot (a + c)$
Prawa De Morgana	$\overline{a \cdot b \cdot \ldots} = \overline{a} + \overline{b} + \ldots$	$\overline{a + b + \ldots} = \overline{a} \cdot \overline{b} \cdot \ldots$
Prawa pochłaniania	$a \cdot (a + b) = a$	$a + a \cdot b = a$
Tożsamości podstawowe	$a \cdot 0 = 0$	$a + 1 = 1$
	$a \cdot 1 = a$	$a + 0 = a$
	$a \cdot a = a$	$a + a = a$
	$a \cdot \overline{a} = 0$	$a + \overline{a} = 1$
Tożsamości dodatkowe	$a \cdot (a + b) = a$	$a + a \cdot b = a$
	$a + \overline{a} \cdot b = a + b$	$a \cdot (\overline{a} + b) = a \cdot b$
	$(a + b) \cdot (\overline{a} + b) = b$	$a \cdot b + \overline{a} \cdot b = b$
Prawo podwójnego przeczenia	$\overline{\overline{a}} = a$	

Prawa algebry Boole'a oznaczone ciemnożółtym tłem nie maja swoich odpowiedników wśród działań w zbiorze liczb rzeczywistych.

Tabela 1.3. Przykład sprawdzenia prawa rozdzielności sumy względem iloczynu $a + b \cdot c = (a + b)(a + c)$.

a	b	c	bc	$a + bc$	$a + b$	$a + c$	$(a + b)(a + c)$
0	0	0	0	0	0	0	0
0	0	1	0	0	0	1	0
0	1	0	0	0	1	0	0
0	1	1	1	1	1	1	1
1	0	0	0	1	1	1	1
1	0	1	0	1	1	1	1
1	1	0	0	1	1	1	1
1	1	1	1	1	1	1	1

Wniosek: kolumny $a + bc$ oraz $(a + b)(a + c)$ są zgodne.

Tabela 1.4. Przykład sprawdzenia prawa De Morgana $\overline{a \cdot b} = \overline{a} + \overline{b}$.

a	b	ab	$\overline{ab}$	$\overline{a}$	$\overline{b}$	$\overline{a}+\overline{b}$
0	0	0	1	1	1	1
0	1	0	1	1	0	1
1	0	0	1	0	1	1
1	1	1	0	0	0	0

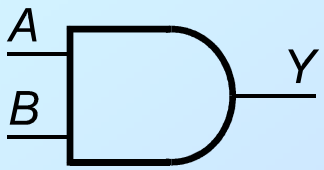
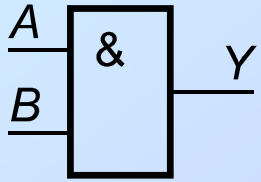
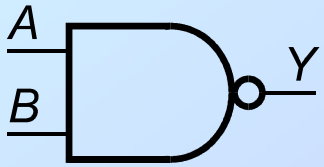
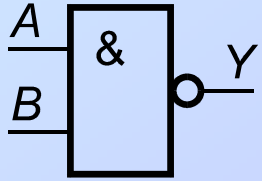
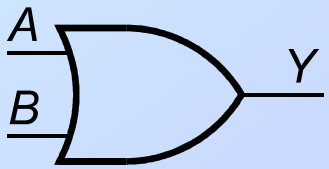
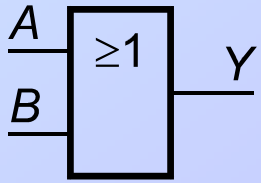
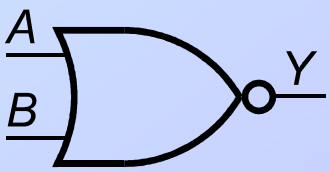
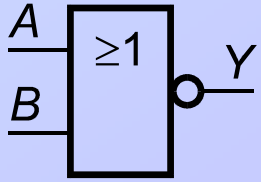
Wniosek : kolumny $\overline{ab}$ oraz $\overline{a}+\overline{b}$ są zgodne.

Tabela 1.5. Rozszerzony zestaw operacji logicznych

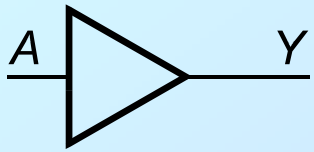
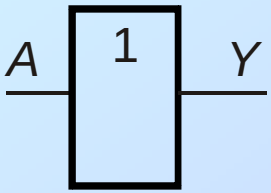
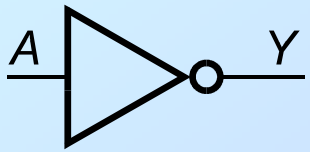
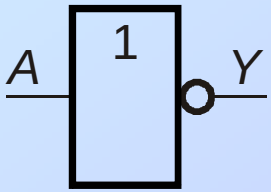
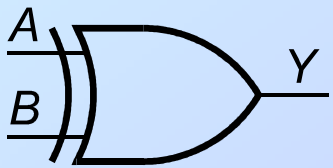
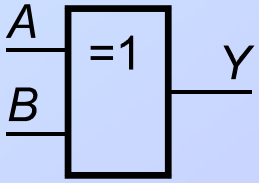
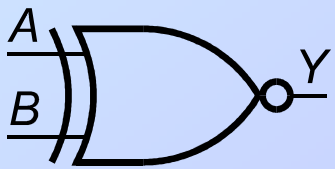
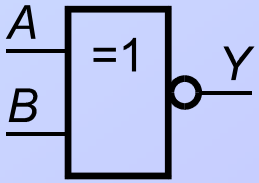
nazwa operacji	symbol	zapis	nazwa bramki
suma logiczna, alternatywa	$+, \vee$	$a + b$	OR
iloczyn logiczny, koniunkcja	brak, $\wedge$	$a\ b$	AND
negacja	$\overline{\ }, \neg, \sim$	$\overline{a}$	NOT
operator Pierce'a	$\downarrow$	$a \downarrow b = \overline{a + b} = \overline{a}\ \overline{b}$	NOR
opertor Sheffera	$\mid$	$a \mid b = \overline{a b} = \overline{a} + \overline{b}$	NAND
nierównoażność, suma modulo 2	$\oplus$	$a \oplus b = a\overline{b} + \overline{a}b$	EXOR, XOR
równoważność, parzystość	$\otimes$, $\Leftrightarrow$	$a \otimes b = ab + \overline{a}\,\overline{b}$	EXNOR, XNOR
implikacja	$\Rightarrow$	$a \Rightarrow b = \overline{a} + b$ $b \Rightarrow a = a + \overline{b}$	*tranz. bipolarny*

2. Podstawowe bramki logiczne

Tabela 2.1. Symbole podstawowych bramek logicznych i ich tablice prawdy.

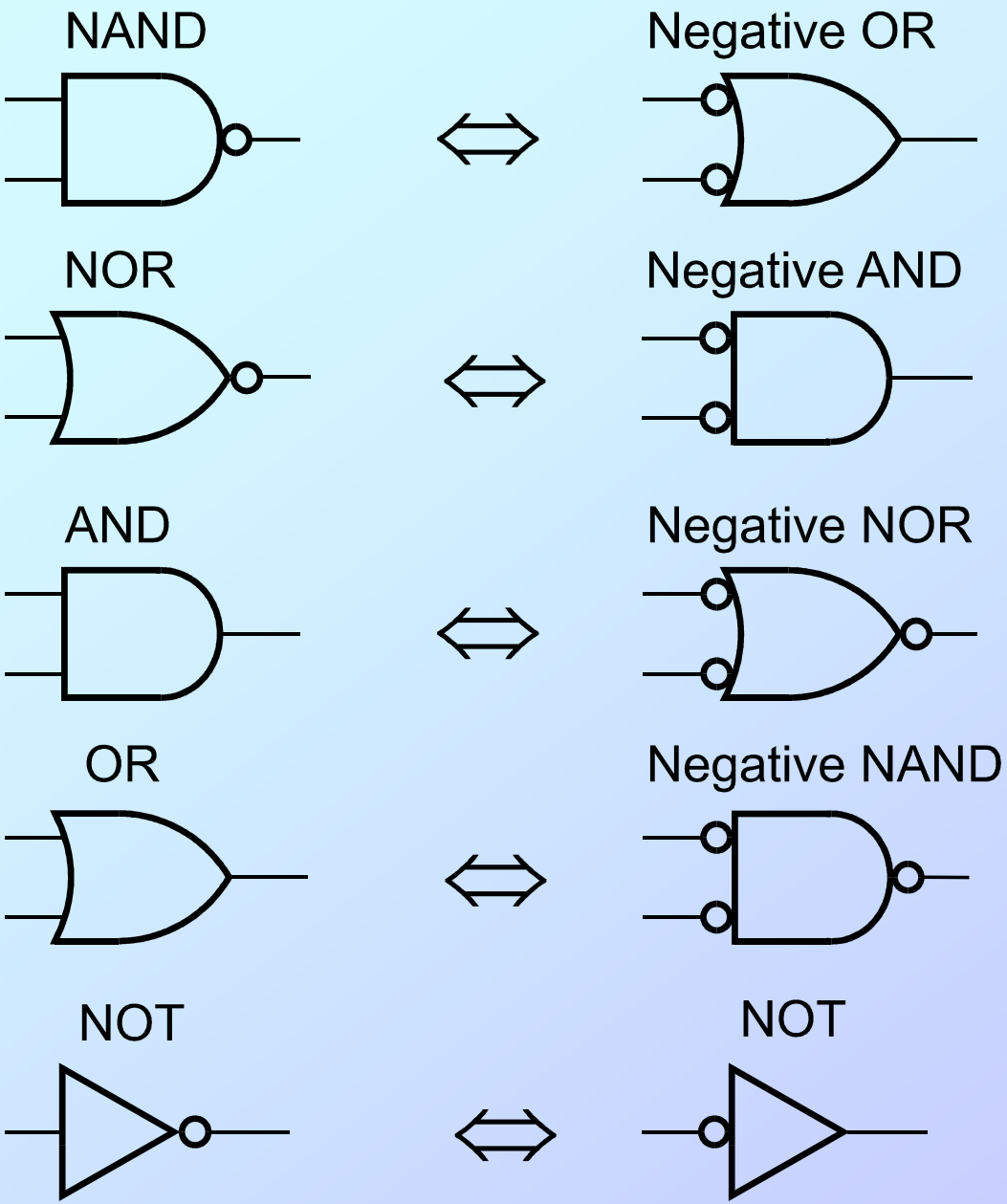
Nazwa elementu ang. / pl.	symbol ANSI/IEEE	symbol IEC, ANSI/IEEE	Tabela prawdy
AND I			A | B | $Y = A \cdot B$ 0 0 0 0 1 0 1 0 0 1 1 1
NAND I-NIE			A | B | $Y = \overline{A \cdot B}$ 0 0 1 0 1 1 1 0 1 1 1 0
OR, LUB			A | B | $Y = A + B$ 0 0 0 0 1 1 1 0 1 1 1 1
NOR, LUB-NIE			A | B | $Y = \overline{A + B}$ 0 0 1 0 1 0 1 0 0 1 1 0

c.d. podstawowe bramki logiczne

Nazwa elementu ang. / pl.	symbol ANSI/IEEE	symbol IEC, ANSI/IEEE	Tabela prawdy
BUFFER BUFOR			A | $Y = A$ 0 | 0 1 | 1
NOT NIE			A | $Y = \overline{A}$ 0 | 1 1 | 0
EXOR (XOR) ALBO, WYŁĄCZNIE LUB			A | B | $Y = A \oplus B$ 0 | 0 | 0 0 | 1 | 1 1 | 0 | 1 1 | 1 | 0
EXNOR (XNOR) ALBO-NIE			A | B | $Y = A \otimes B$ 0 | 0 | 1 0 | 1 | 0 1 | 0 | 0 1 | 1 | 1

Obecnie dopuszczone do użytkowania są oba zestawy symboli wprowadzone w normach: IEEE Std 91/91a-1991; IEC 60617-12:1997.

System wskaźników negacji

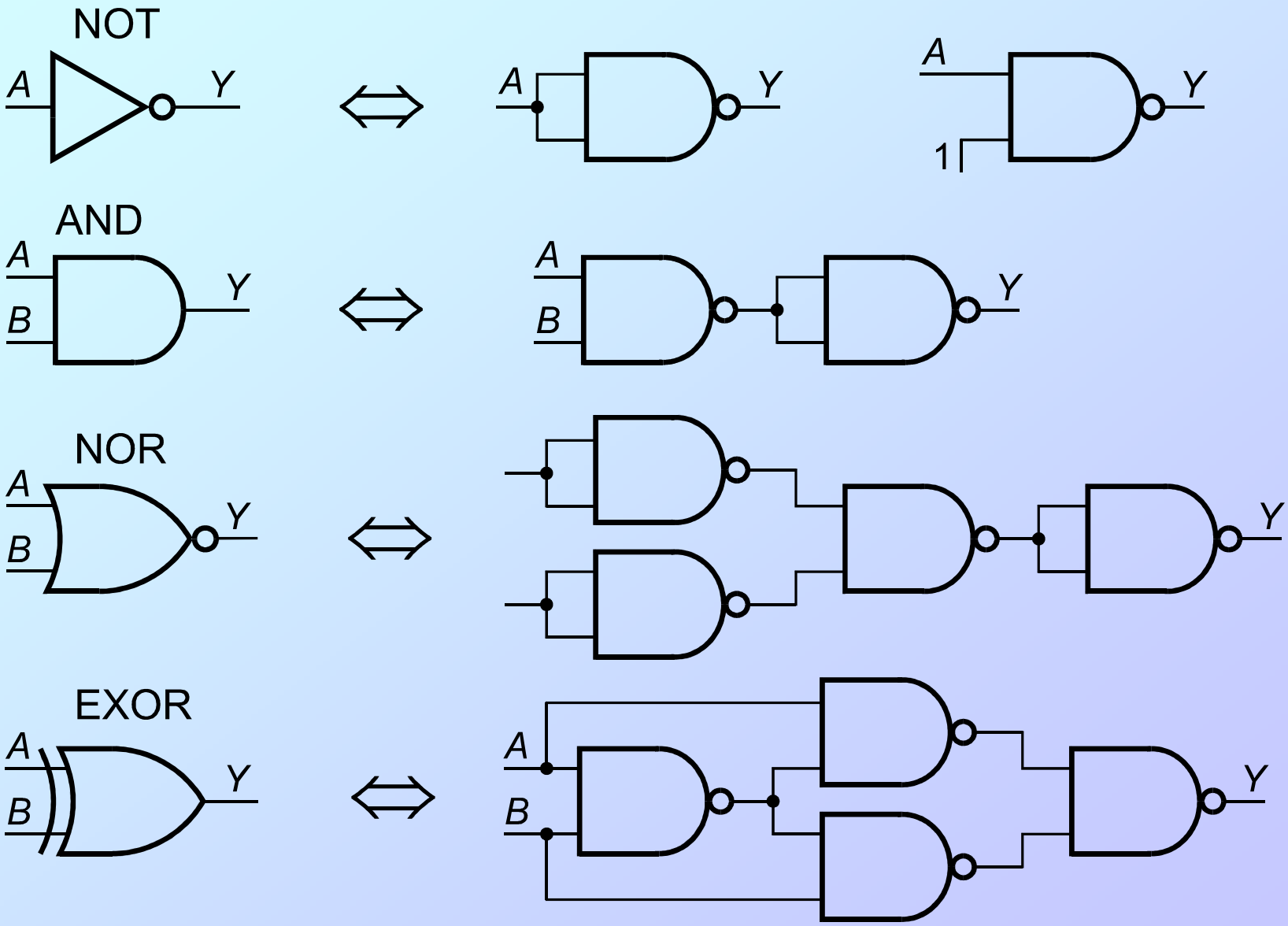


- System wskaźników negacji stosowany jest z obydwoma rodzajami symboli bramek (ANSI/IEEE, IEC).
- Przyjmuje się, że miejsce oznaczenia negacji opisuje sposób funkcjonowania realnych układów. Obecność wskaźnika negacji oznacza, że stan wewnętrzny jest negacją stanu zewnętrznego.
- Obecność lub brak wskaźnika negacji nie ma żadnego związku z poziomami wielkości fizycznych (napięcie, prąd) na kocówkach realnych układów.

Rys. 2.1. Równoważne symbole graficzne elementów logicznych.

Uniwersalność bramek NAND

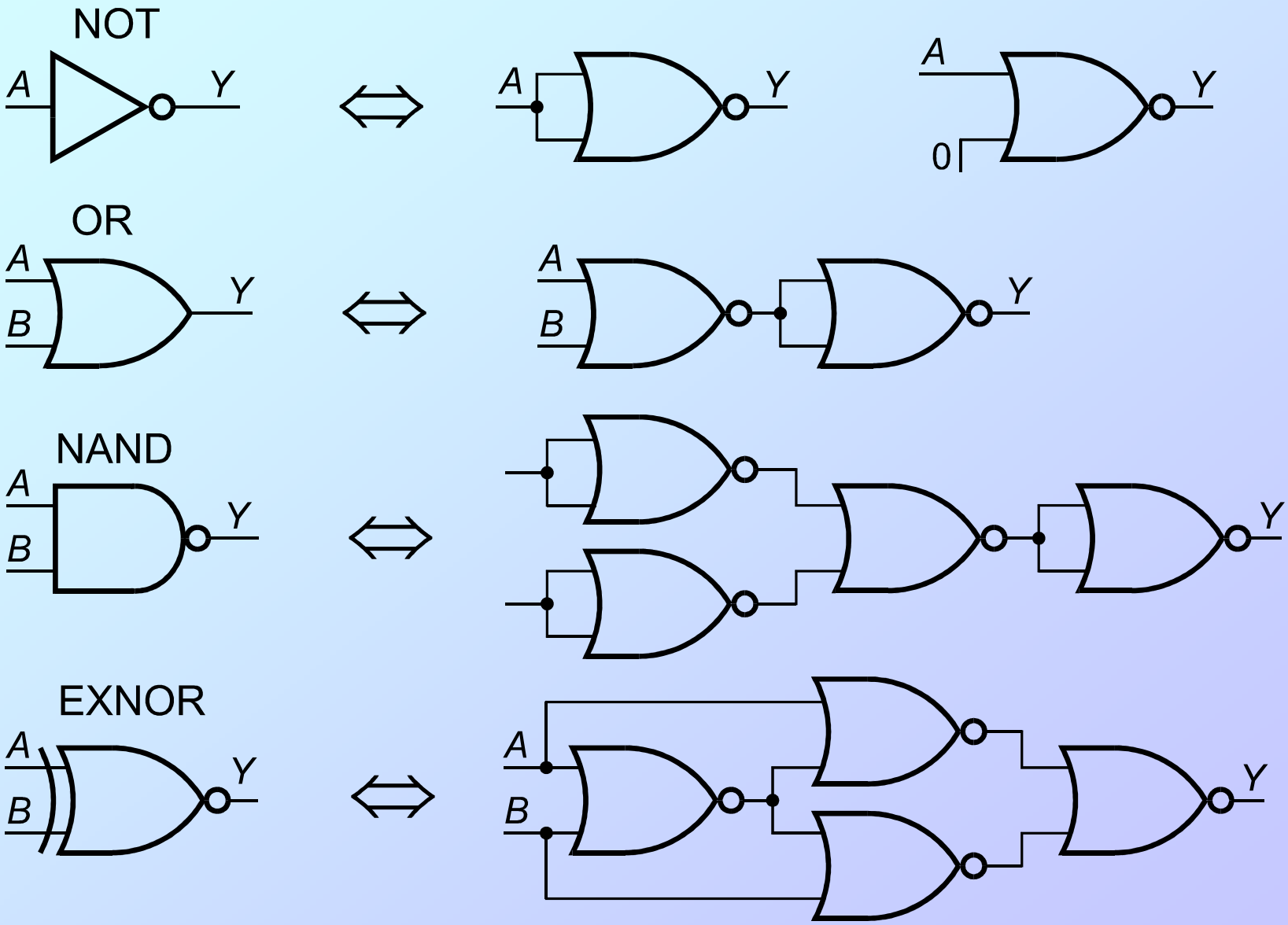
Wszystkie funktory logiczne można zrealizować przy użyciu samych bramek NAND.



Rys. 2.2. Przykłady realizacji innych funktorów logicznych przy użyciu bramek NAND.

Uniwersalność bramek NOR

Wszystkie funktory logiczne można zrealizować przy użyciu samych bramek NOR.

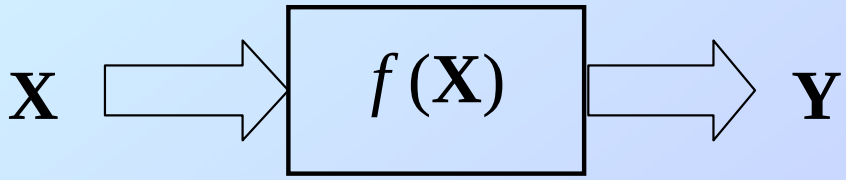


Rys. 2.3. Przykłady realizacji innych funktorów logicznych przy użyciu bramek NOR.

3. Układy kombinacyjne - 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 $x_1, x_2, \ldots, x_n$,

$$\mathbf{X} = (x_1, x_2, \ldots, x_n) \qquad (3.1)$$

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

$$\mathbf{Y} = (y_1, y_2, \ldots, y_m) \qquad (3.2)$$

f – funkcja boolowska (przełączająca, …)

$$\mathbf{Y} = f(\mathbf{X}) \qquad (3.3)$$

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).
4). Znalezienie najprostszej (minimalnej) postaci funkcji.
5). Schemat układu.
6). Ocena kosztu układu i ewentualnie korekta schematu.

Metody opisu układów kombinacyjnych:

- opis słowny
- tablica prawdy
- zbiór jedynek funkcji F^1 (albo zer F^0)
- funkcja boolowska (przełączająca, logiczna)
- - postać kanoniczna sumy,
- - postać kanoniczna iloczynu,
- - postać zminimalizowana.
- schemat układu złożonego z podstawowych funktorów logicznych,
- inne.

Metody minimalizacji wyrażeń logicznych:

- 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.

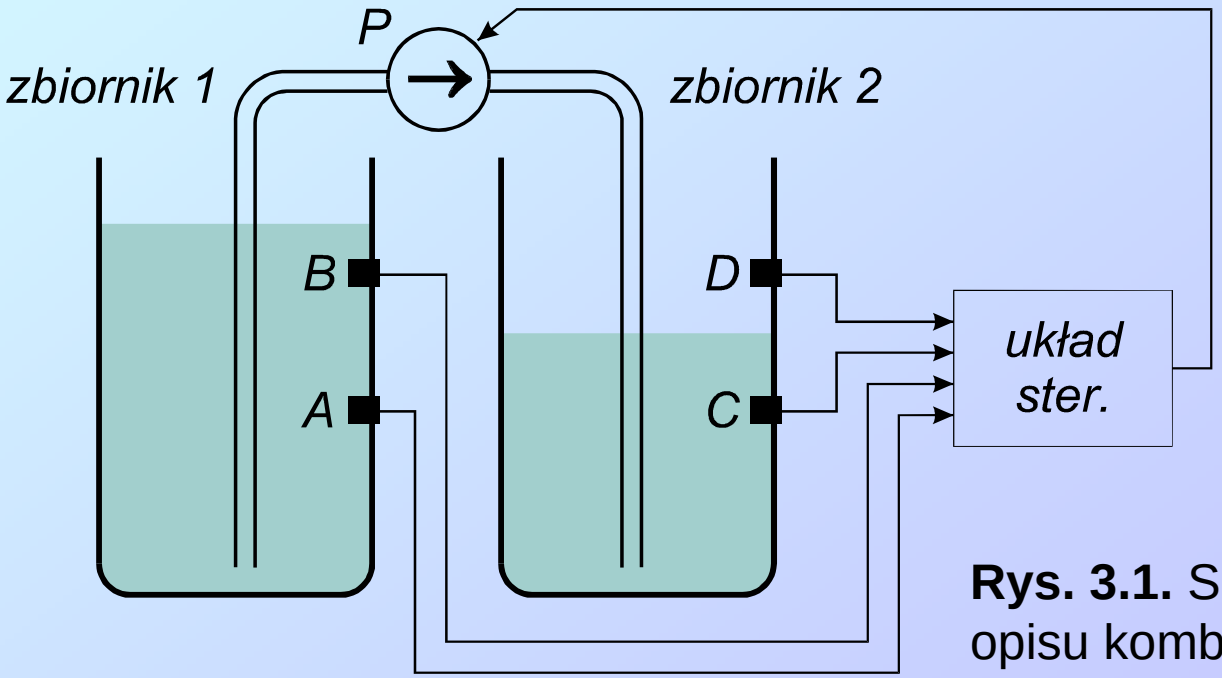
Opis słowny układu kombinacyjnego

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:



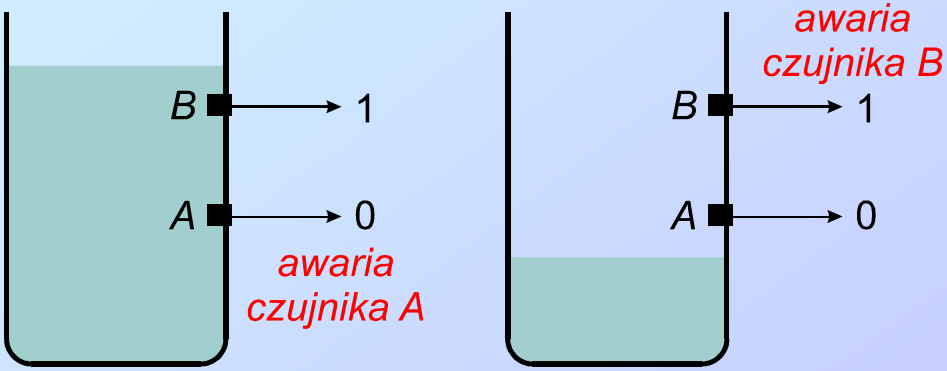
Rys. 3.1. Schemat instalacji do słownego opisu kombinacyjnego układu sterowania.

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. 3.2. 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 3.1. Tablica prawdy dla układu sterowania z rys. 3.1.

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

$$P = A\overline{B}\,\overline{C}\,\overline{D} + AB\overline{C}\;\overline{D} + ABC\overline{D}. \tag{3.4}$$

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

$$P = (A+B+C+D)(A+B+\overline{C}+D)(A+B+\overline{C}+\overline{D})\cdot \tag{3.5}$$
$$\cdot(\overline{A}+B+\overline{C}+D)(\overline{A}+B+\overline{C}+\overline{D})(\overline{A}+\overline{B}+\overline{C}+\overline{D}).$$

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,

$$
\begin{aligned}
P &= A\overline{B}\,\overline{C}\,\overline{D} + AB\overline{C}\,\overline{D} + ABC\overline{D} = && \text{przemienność iloczynu logicznego, rozdzielność iloczynu logicznego względem sumy} \\
&= (\overline{B} + B)A\overline{C}\,\overline{D} + ABC\overline{D} = && \text{prawo wyłączonego środka} \\
&= A\overline{C}\,\overline{D} + ABC\overline{D} = && \text{przemienność iloczynu logicznego, rozdzielność iloczynu logicznego względem sumy} \\
&= (\overline{C} + BC)A\overline{D} = && \text{tożsamość pomocnicza (Shannona)} \\
&= (B + \overline{C})A\overline{D} && (3.6)
\end{aligned}
$$

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.

4. Synteza układu kombinacyjnego metodą 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.

P \ CD	00	01	11	10
00	0	–	0	0
01	–	–	–	–
11	1	–	0	1
10	1	–	0	0

(AB)

Tabela 4.1. Tablica Karnaugha odpowiadająca tabeli prawdy 3.1.

Etap 1. Kopiowanie danych do tablicy Karnaugha.

Kopia tabeli 3.1 (od zadania z pompą).

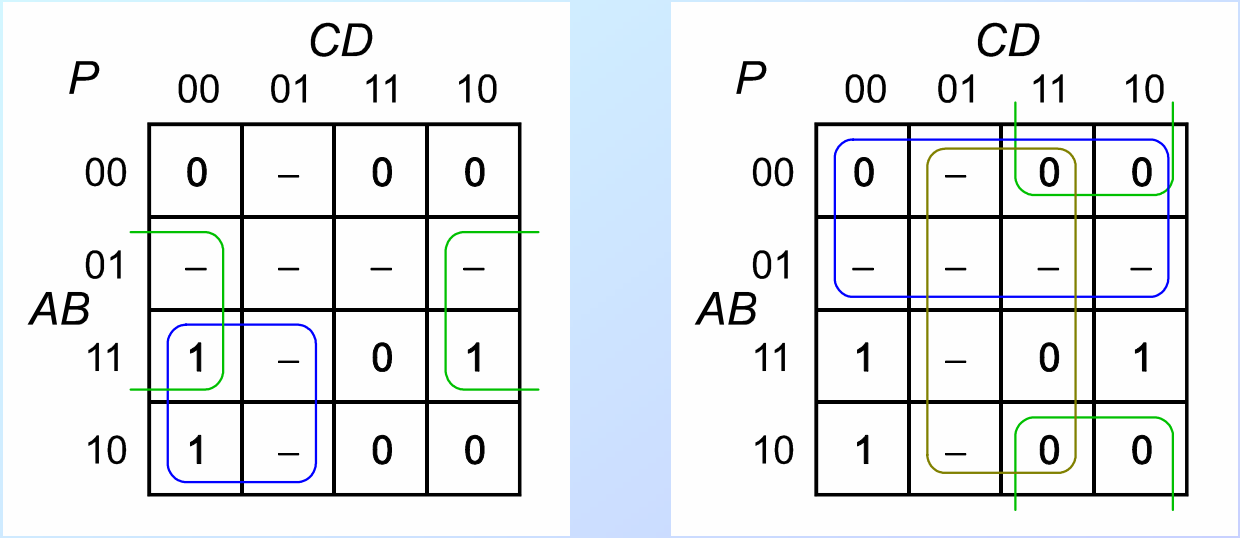
A	B	C	D	P
0	0	0	0	0
0	0	0	1	–
0	0	1	0	0
0	0	1	1	0
0	1	0	0	–
0	1	0	1	–
0	1	1	0	–
0	1	1	1	–

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

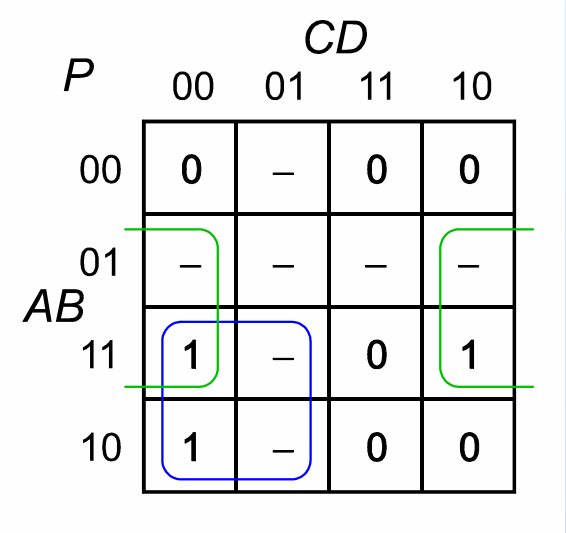
Zasady OPTYMALNEGO grupowania:

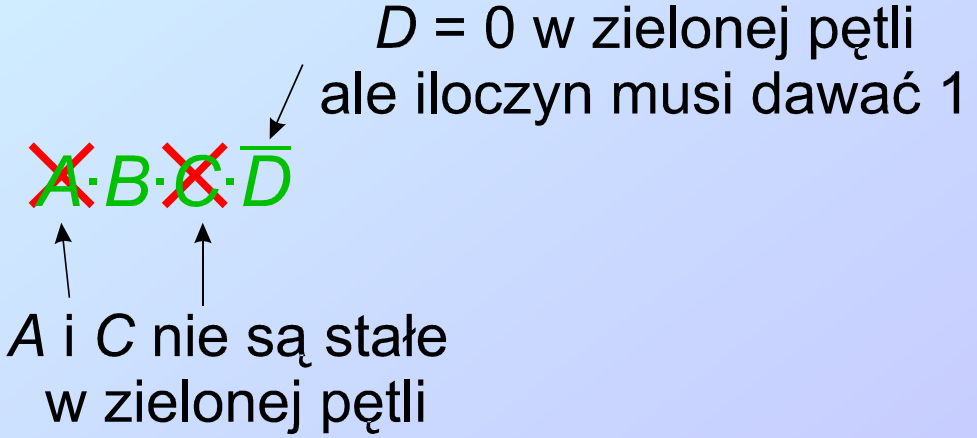
- staramy się zgrupować wszystkie „1” (alternatywnie „0”) wewnątrz możliwie najmniejszej liczby pętli,
- pętle o większych rozmiarach są korzystniejsze od pętli mniejszych.

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.





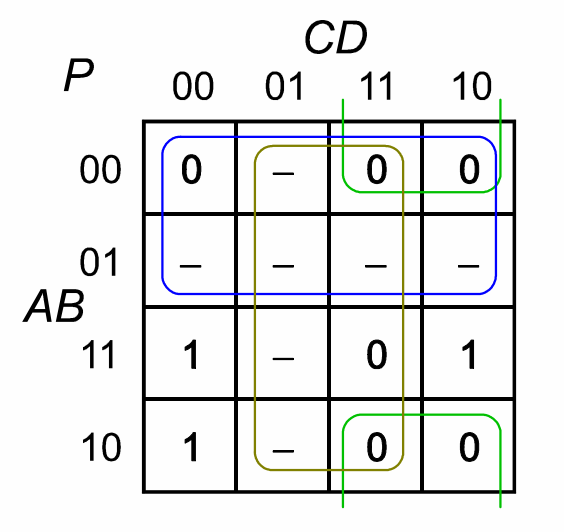
Rys. 4.2. Przykład konstrukcji wyrażenia odpowiadającego pętli zielonej grupującej jedynki.

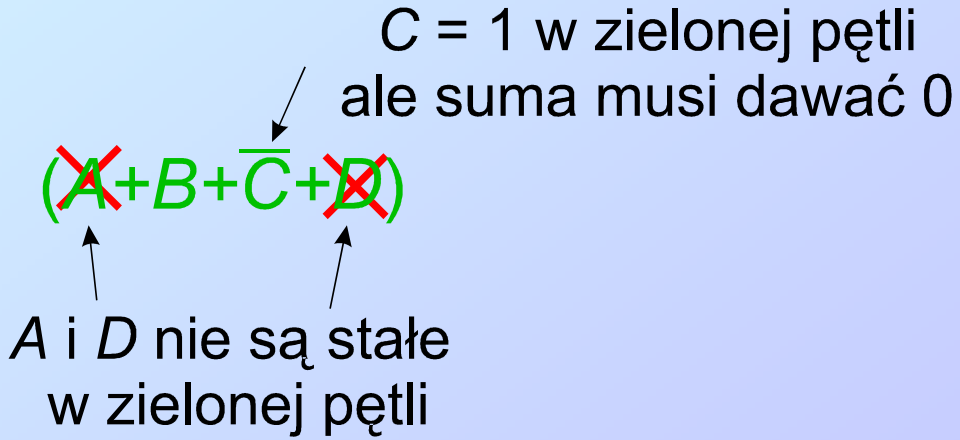
Razem dla obu zaznaczonych pętli otrzymujemy

$$P = A\overline{C} + B\overline{D} \qquad (4.1)$$

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.





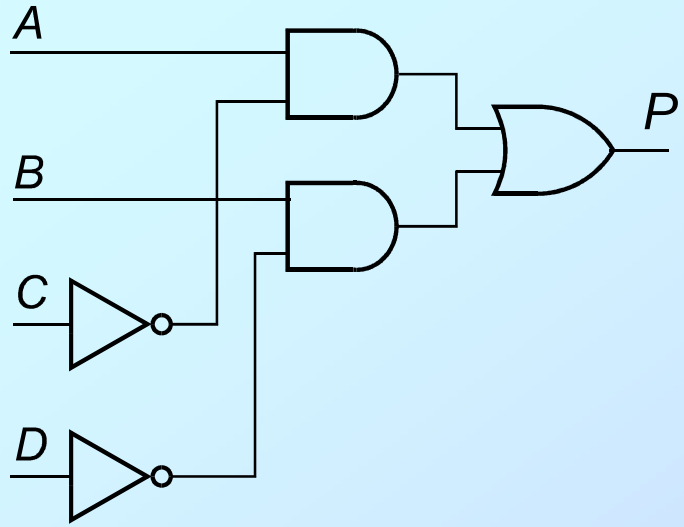
Rys. 4.3. Przykład konstrukcji wyrażenia odpowiadającego pętli zielonej grupującej zera.

Razem dla wszystkich zaznaczonych pętli otrzymujemy

$$P = (B + \overline{C})A\overline{D} \qquad (4.2)$$

Porównanie schematów układów kombinacyjnych

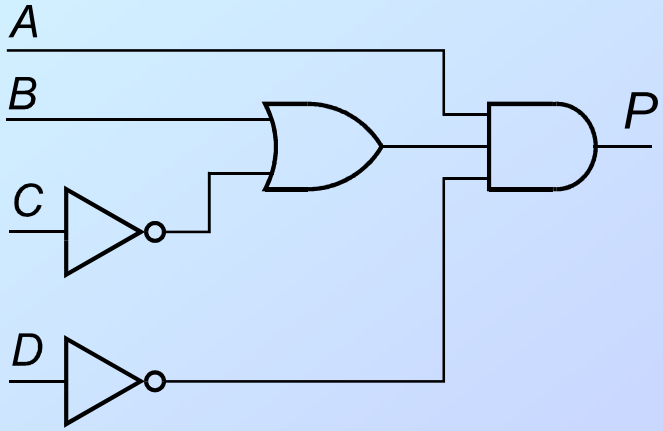
Przypadek grupowania jedynek



Układ zbudowano na podstawie funkcji (2.7):

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

Przypadek grupowania zer



Układ zbudowano na podstawie funkcji (2.8):

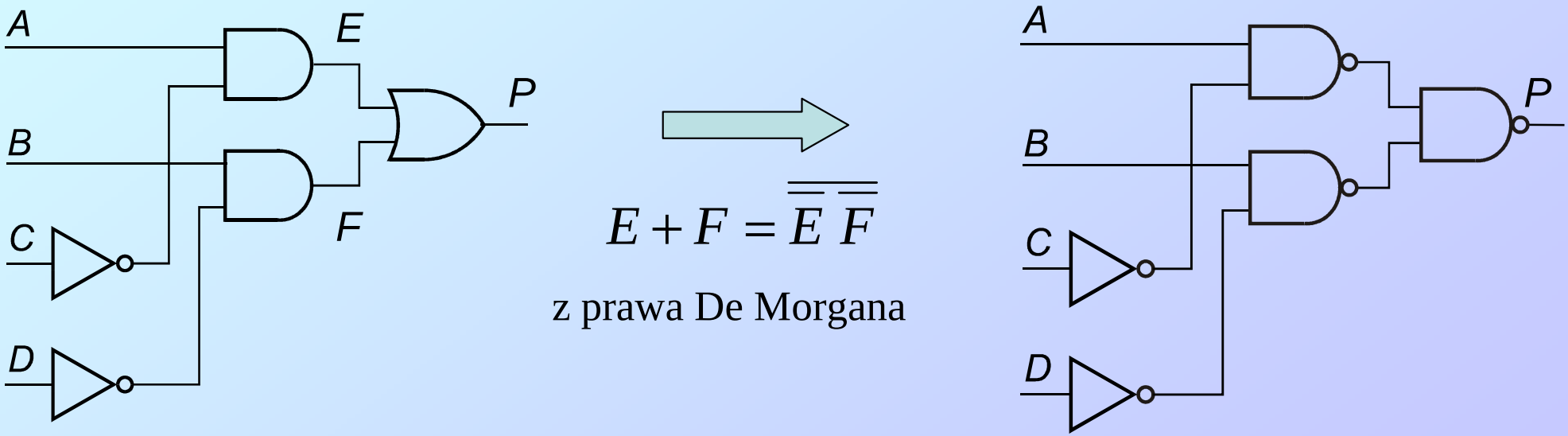
$$P = (B + \overline{C})A\overline{D}$$

Rys. 4.4. 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.



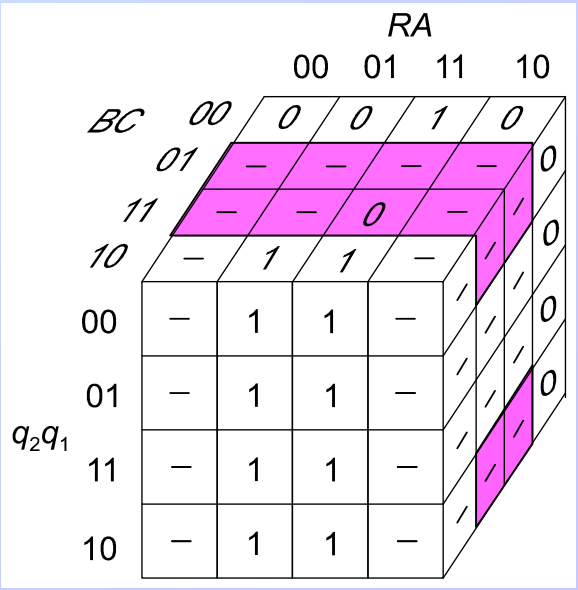
Rys. 4.5. Porównanie schematów przed i po ocenie kosztów układu.

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. 4.6. 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.

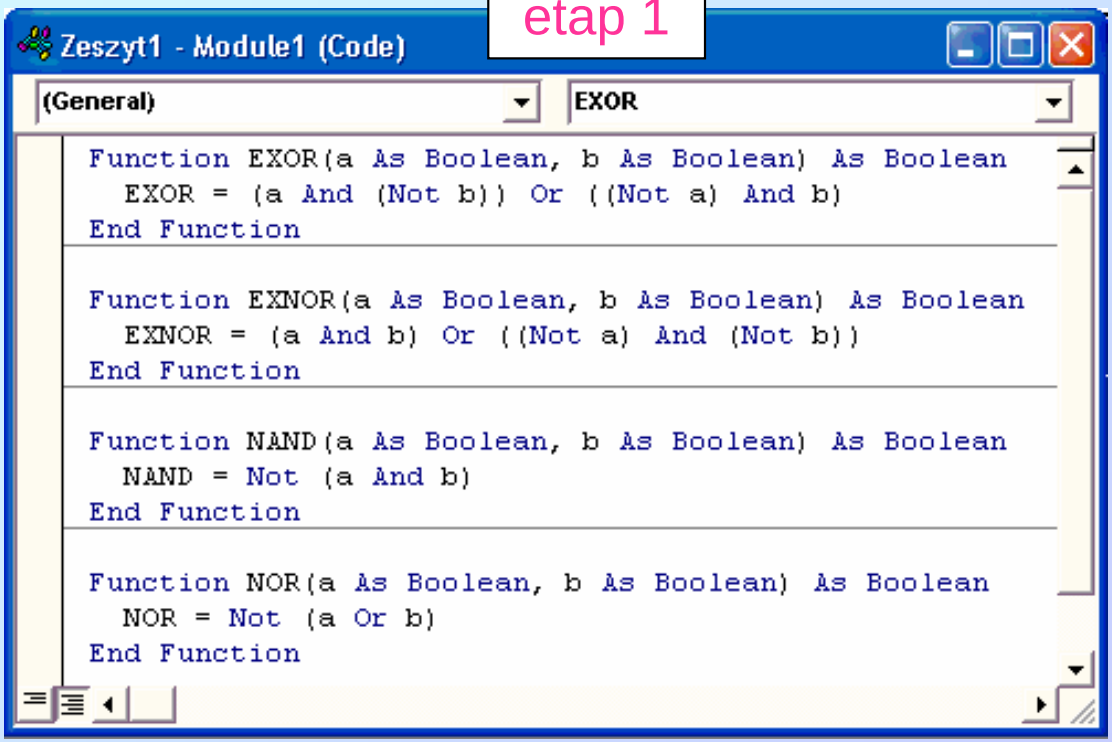
Zastosowanie arkusza MS Excel do testowania wyrażeń logicznych

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

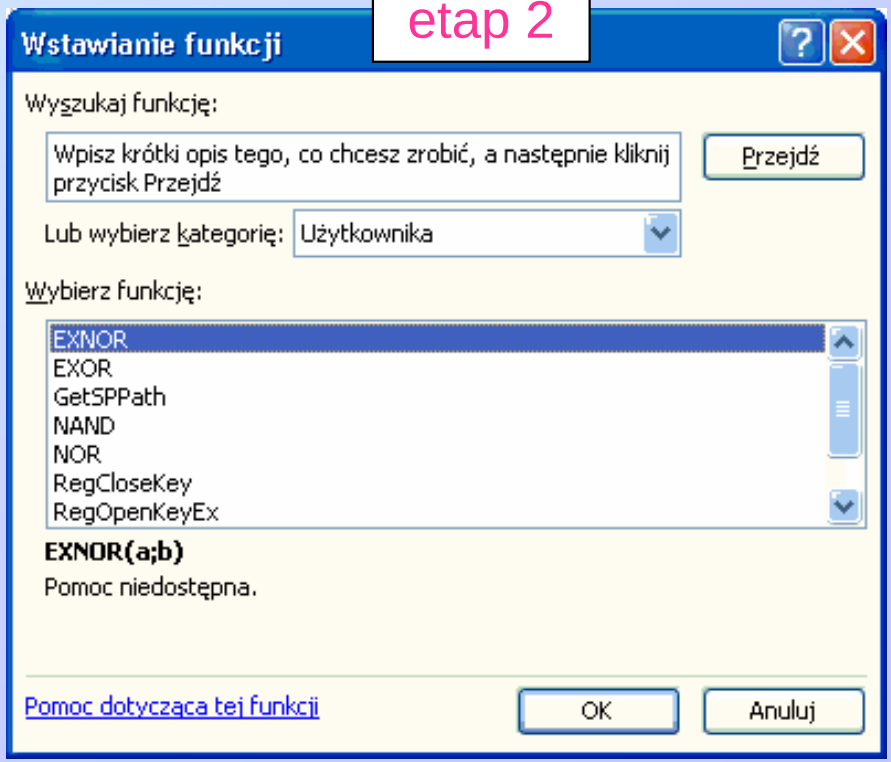
funkcja Excel PL	równoważna bramka log.
=nie(a1)	NOT
=lub(a1; a2; …)	OR
=oraz(a1; a2; …)	AND

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



5. 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 Sasal, Układy scalone serii UCA64/UCY74, WKiŁ, Warszawa 1990.

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

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

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

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

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

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

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

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

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

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