Automatyzacja i robotyzacja procesów produkcyjnych

a	b	c	z
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

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

Układy cyfrowe - wprowadzenie
Układy kombinacyjne

Materiały
http://staff.uz.zgora.pl/ipajak

Plan wykładu

Programowalne sterowniki logiczne PLC

architektura sterownika,

programowanie sterowników (norma IEC 61131-3): typy danych, zmiennych,

języki programowania: LD, IL, ST, FBD, metoda SFC.

Układy logiczne: kombinacyjne i sekwencyjne

modele: algebra Boole'a, automaty skończone,

realizacja funkcji logicznych, wybrane sposoby realizacji elementów logicznych.

Podstawy robotyki

klasyfikacja robotów, przegląd podstawowych zagadnień robotyki, zastosowania robotów przemysłowych, układy współrzędnych, podstawowe metody programowania.

Literatura

Sterowniki PLC

Pająk I., Pająk G., *Cyfrowe układy automatyki przemysłowej*, Oficyna Wydawnicza Uniwersytetu Zielonogórskiego 2023

Kasprzyk J., *Programowanie sterowników przemysłowych,* WNT, Warszawa 2006

Mikulczyński T., *Automatyzacja procesów produkcyjnych,* WNT, Warszawa 2006

Układy logiczne

Siwiński J., *Układy przełączające w automatyce,* WNT, Warszawa 1980

Szejach W., *Automatyka, elementy i układy przełączające,* Wydawnictwa Politechniki Warszawskiej, Warszawa 1981

Roboty przemysłowe

Universal Robots, Universal Robots e-Series Podręcznik użytkownika

Universal Robots Academy, E-szkolenia na temat e-Series: https://academy.universal-robots.com/pl/bezplatne-e-szkolenie/e-szkolenia-na-temat-e-series/

Sterowniki PLC

Pająk I., Pająk G., *Cyfrowe układy automatyki przemysłowej*, Oficyna Wydawnicza Uniwersytetu Zielonogórskiego 2023

Kasprzyk J., *Programowanie sterowników przemysłowych,* WNT, Warszawa 2006

Mikulczyński T., *Automatyzacja procesów produkcyjnych,* WNT, Warszawa 2006

Układy logiczne

Siwiński J., *Układy przełączające w automatyce,* WNT, Warszawa 1980

Szejach W., *Automatyka, elementy i układy przełączające,* Wydawnictwa Politechniki Warszawskiej, Warszawa 1981

Roboty przemysłowe

Universal Robots, Universal Robots e-Series Podręcznik użytkownika

Universal Robots Academy, E-szkolenia na temat e-Series: https://academy.universal-robots.com/pl/bezplatne-e-szkolenie/e-szkolenia-na-temat-e-series/

Układy cyfrowe

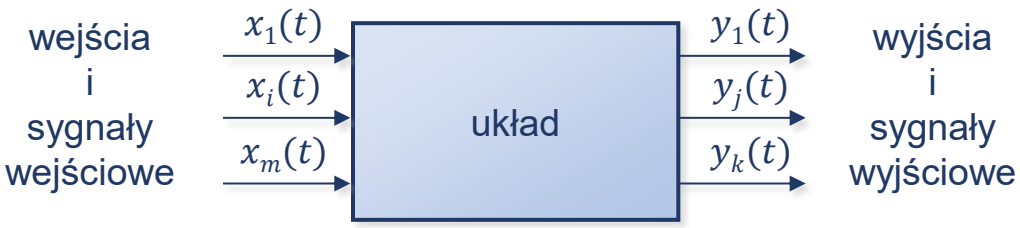
wprowadzenie

Pojęcia podstawowe

Sygnał to przebieg czasowy określonej wielkości fizycznej, za pomocą której przekazywane są informacje, na sygnał składają się:

- treść sygnału – informacja przenoszona przez sygnał,
- nośnik sygnału – wielkość fizyczna, której zmiany umożliwiają przekazanie określonych treści (np. ciśnienie gazu lub cieczy, natężenie lub napięcie prądu, itp.).

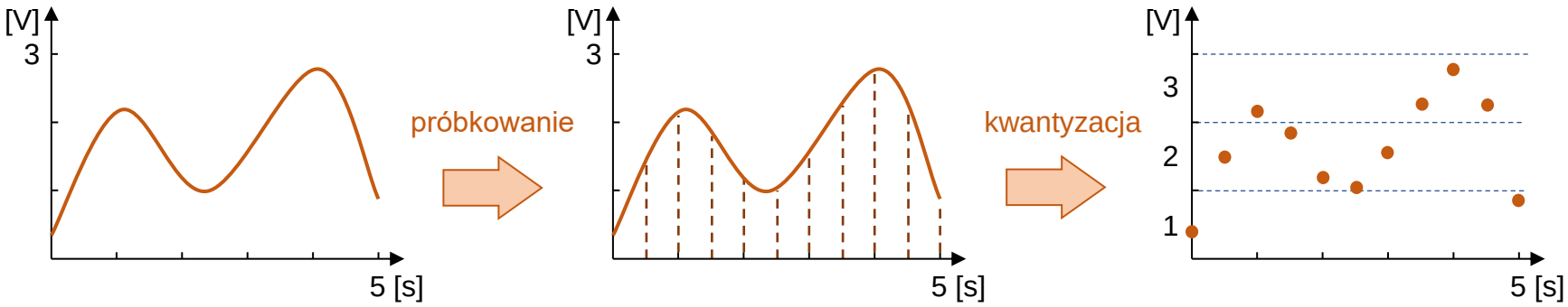
Układ to umownie wyodrębniony fragment rzeczywistości, oddziaływania zewnętrzne, które wpływają na zachowanie układu nazywane są sygnałami wejściowymi, a miejsca ich oddziaływania wejściami układu, wielkości opisujące oddziaływanie układu na środowisko nazywane są sygnałami wyjściowymi układu, a miejsca ich oddziaływania wyjściami układu, układy przetwarzają sygnały wejściowe w sygnały wyjściowe.



Przetwarzanie analogowo-cyfrowe

Przetwornik analogowo-cyfrowy A/C (ang. A/D lub ADC) to układ służący do zamiany sygnału analogowego na cyfrowy, zamiana przebiega w trzech etapach: próbkowanie, kwantowanie i kodowanie (kodowanie to proces polegający na przyporządkowaniu określonej informacji ustalonego zestawu symboli, w tym przypadku wartości liczbowej odpowiadającej otrzymanemu poziomowi kwantowania przypisywany jest jej kod – zwykle naturalny kod dwójkowy czy kod BCD).

Przetwornik cyfrowo-analogowy C/A (ang. D/A lub DAC) to układ przetwarzający sygnał cyfrowy na równoważny mu sygnał analogowy.



sygnał ciągły
$y = f(t)$

sygnał dyskretny
$y_d = \{0.4,1.5,2.2,1.8,1.2,1.0, \ldots\}$

sygnał cyfrowy
$y_c = \{1,2,3,2,2,2,2,3,3,3,1\}$

Układy cyfrowe

Układy cyfrowe to układy których sygnały są sygnałami cyfrowymi (tzn. są dyskretne w czasie i mają dyskretne wartości). Układy te są podstawowymi układami sterowania stosowanymi do automatyzacji procesów produkcyjnych. Wykorzystanie cyfrowych urządzeń sterujących w przypadku występowania sygnałów analogowych wymaga stosowania przetworników A/C i C/A.

Zalety układów cyfrowych

duża odporność na zakłócenia,

większa niezawodność układów,

łatwość zapamiętywania i przechowywania informacji cyfrowych,

duża dokładność przetwarzania (zależy od dokładności informacji wejściowych),

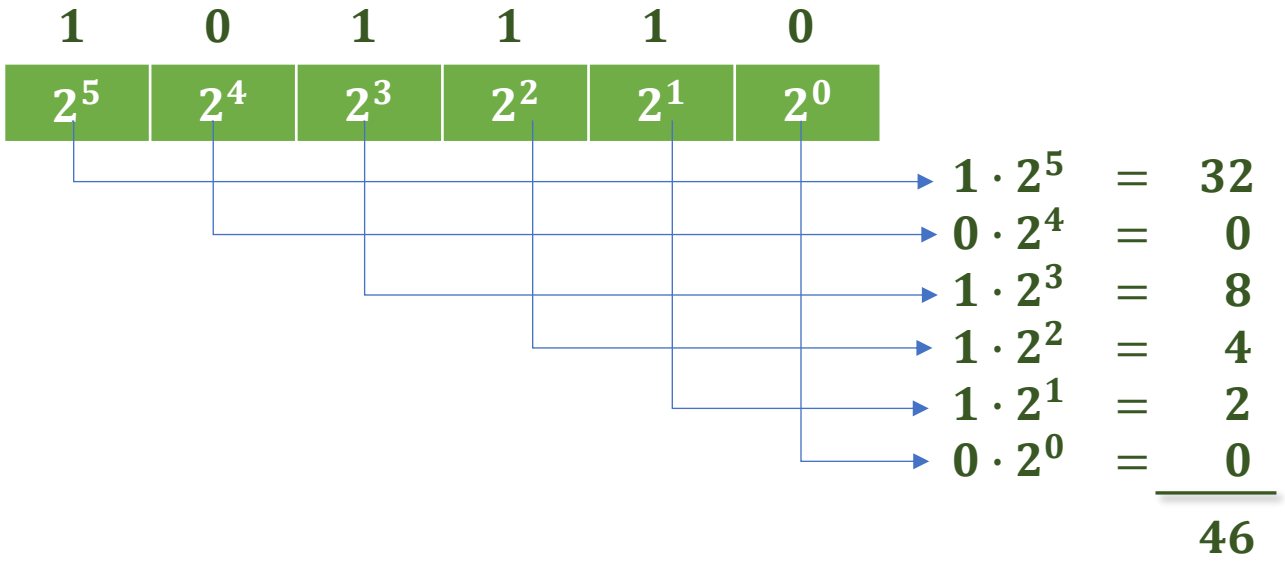
możliwość realizacji złożonych algorytmów przetwarzania sygnałów,

niski koszt w stosunku do realizowanych funkcji.

Kody liczbowe

Kod to dowolnie uporządkowany układ symboli przedstawiających słowa, liczby, informacje. Symbolami wykorzystywanymi przez kody liczbowy są cyfry.

Naturalny kod dwójkowy (binarny) jest kodem wagowym (pozycyjnym) wykorzystującym do kodowania cyfry 0 i 1, każda pozycja (bit) kodu ma określoną wagę: waga i-tej pozycji w n pozycyjnym kodzie wynosi 2^i gdzie $i = 0, 1, 2, \ldots, n-1$.



$$46_{10} = \mathbf{1} \cdot 2^5 + \mathbf{0} \cdot 2^4 + \mathbf{1} \cdot 2^3 + \mathbf{1} \cdot 2^2 + \mathbf{1} \cdot 2^1 + \mathbf{0} \cdot 2^0 = 101110_2$$

Kody BCD

Kody dwójkowo dziesiętne (BCD, ang. *Binary Coded Decimal*) to kody służące do dwójkowego zakodowania cyfr dziesiętnych ($0, 1, .., 9$).

Do kodowania dziesięciu cyfr potrzebny jest co najmniej 4 bitowy kod dwójkowy. W najprostszych kodach BCD każda cyfra jest kodowana przy pomocy dwójkowego kodu wagowego: stosowane są różne kombinacje wag dla poszczególnych pozycji np. 8421, 7421, 2421, 5211.

Kod **BCD 8421** jest nazywany

naturalnym kodem BCD lub po prostu BCD.

$$46_{10} = 0100\,0110_{\text{BCD}}$$

cyfra dziesiętna	wagi kodów			
	8421	**7421**	**2421**	**5211**
0	0000	0000	0000	0000
1	0001	0001	0001	0001
2	0010	0010	0010	0011
3	0011	0011	0011	0101
4	0100	0100	0100	0111
5	0101	0101	1011	1000
6	0110	0110	1100	1001
7	0111	1000	1101	1011
8	1000	1001	1110	1101
9	1001	1010	1111	1111

Kod Graya

Kod Graya należy do grupy kodów refleksyjnych (dwie kolejne liczby kodu refleksyjnego różnią się tylko jedną pozycją).

N–pozycyjny kod Graya można utworzyć z kodu $n-1$ pozycyjnego zgodnie z następującym algorytmem:

do układu symboli kodu $n-1$ pozycyjnego należy dopisać jeszcze raz te same symbole ale w odwrotnej kolejności,

do pierwotnych symboli z przodu dopisać cyfrę „0” a do symboli dopisanych „1”.

kod 1-poz.	*po odbiciu*	*kod 2-poz*
0	0	00
1	1	01
	1	11
	0	10

kod 2-poz.	*po odbiciu*	*kod 3-poz*
00	00	000
01	01	001
11	11	011
10	10	010
	10	110
	11	111
	01	101
	00	100

Konstrukcja 3-pozycyjnego kodu Graya

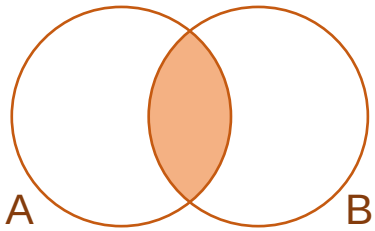
Układy cyfrowe – pojęcia pokrewne

Układ przełączający – układ cyfrowy

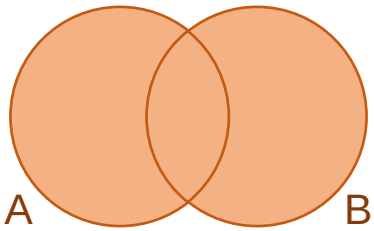
Układ zbudowany z tzw. elementów przełączających, elementy te mogą być w stanie włączenia (przewodzenia) i wyłączenia (blokowania).

Układ logiczny – model matematyczny układu cyfrowego oparty na algebrze Boole'a; często określenie „układ logiczny" jest stosowane zamiennie z określeniem „układ cyfrowy".

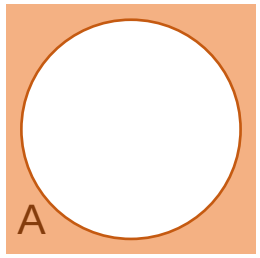
Teoria układów cyfrowych oparta jest na logice matematycznej, głównie na rachunku zdań. Podstawowe znaczenie ma dwuelementowa algebra Boole'a, która jest sformalizowanym uogólnieniem rachunku zdań.



A ∩ B



A ∪ B



¬ A

Układy cyfrowe – pojęcia pokrewne

Automat skończony – model matematyczny układu cyfrowego zdefiniowany w teorii automatów; automat skończony jest układem który może znajdować się w jednym ze skończonej liczby stanów, każda zmiana wejść automatu powoduje zmianę jego stanu, działanie automatu można przedstawić np. przy pomocy grafu którego wierzchołki odpowiadają stanom automatu a gałęzie oznaczają przejście pomiędzy stanami wymuszane określonymi sygnałami wejściowymi.

Graf stanów układu arbitra

Klasyfikacja układów cyfrowych

Klasyfikacja ze względy na sposób przetwarzania sygnałów

- **układy kombinacyjne**

  każda kombinacja wartości sygnałów wejściowych (tzw. stan wejść) określa jednoznacznie kombinację wartości sygnałów wyjściowych (tzw. stan wyjść), układy te nazywane są również układami bez pamięci,

- **układy sekwencyjne**

  istnieje przynajmniej jeden stan wejść któremu odpowiada kilka stanów wyjść, układy nazywane są również układami z pamięcią.

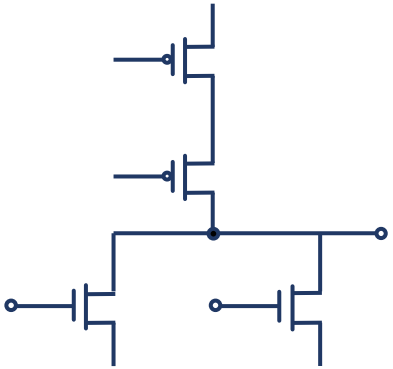
Klasyfikacja ze względu na rodzaj elementów konstrukcyjnych

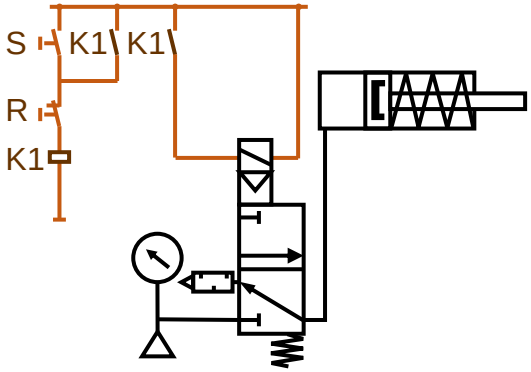
- układy konstruowane z bramek i przerzutników,
- układy konstruowane z tzw. bloków funkcjonalnych (typowe bloki funkcjonalne to multipleksery, demultipleksery, dekodery, sumatory, komparatory, rejestry, liczniki i pamięci).

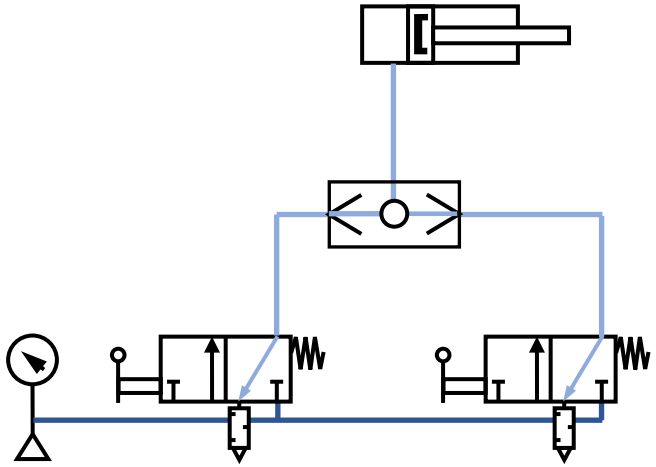
Klasyfikacja układów cyfrowych

Klasyfikacja ze względu na metodę realizacji

- układy stykowe (przekaźnikowe) – elementy przełączające: przyciski, przekaźniki, itp.
- układy złożone z półprzewodnikowych elementów elektronicznych – elementy przełączające budowane z pojedynczych tranzystorów, diod, rezystorów; realizowane jako układy scalone zawierające od kilku do setek tysięcy bramek: skala integracji SSI, MSI, LSI, VLSI, w różnych technologiach: TTL, CMOS, ..., produkowane jako układy uniwersalne lub wg projektu użytkownika (ASIC, PLD), układy procesorowe,
- układy złożone z elementów pneumatycznych,
- układy złożone z elementów hydraulicznych.







Układy kombinacyjne

podstawy teoretyczne

Algebra Boole'a

Dwuelementowa algebra Boole'a jest sformalizowanym uogólnieniem rachunku zdań. Jest ona definiowana jako układ:

$$B = (\{0,1\}, +, \cdot, \bar{}, 0, 1)$$

gdzie: $\{0,1\}$ – zbiór elementów algebry, 0, 1 – stałe algebry; $+$, $\cdot$ – dwuargumentowe operacje alternatywy i koniunkcji, $\bar{}$ – jednoargumentowa operacja negacji.

koniunkcja		
a	b	$a \cdot b$
0	0	0
0	1	0
1	0	0
1	1	1

alternatywa		
a	b	$a + b$
0	0	0
0	1	1
1	0	1
1	1	1

negacja	
a	$\bar{a}$
0	1
1	0

Uwaga: symbol koniunkcji zazwyczaj jest pomijany, stąd $ab = a \cdot b$

Wybrane prawa algebry Boole'a

Prawa istnienia elementu identycznościowego

$$a + 0 = 0 + a = a \qquad a \cdot 1 = 1 \cdot a = a$$

Prawa istnienia dopełnienia

$$a + \overline{a} = 1 \qquad a \cdot \overline{a} = 0$$

Prawa przemienności

$$a + b = b + a \qquad a \cdot b = b \cdot a$$

Prawa łączności

$$a + (b + c) = (a + b) + c \qquad a \cdot (b \cdot c) = (a \cdot b) \cdot c$$

Prawa rozdzielności

$$a \cdot (b + c) = a \cdot b + a \cdot c \qquad a + b \cdot c = (a + b) \cdot (a + c)$$

Prawa de Morgana

$$\overline{a + b} = \overline{a} \cdot \overline{b} \qquad \overline{a \cdot b} = \overline{a} + \overline{b}$$

Prawa idempotentności

$$a + a = a \qquad a \cdot a = a$$

Prawa pochłaniania

$$a + a \cdot b = a \qquad a \cdot (a + b) = a$$

Prawa sklejania

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

$$a \cdot b + a \cdot \overline{b} = a$$

Prawo podwójnej negacji

$$\overline{\overline{a}} = a$$

Prawa elementów neutralnych

$$a + 1 = 1$$

$$a \cdot 0 = 0$$

Funkcje boolowskie

Funkcją boolowską (logiczną, przełączającą) nazywane jest odwzorowanie postaci:

$$f: X \rightarrow Y,$$

$$X \subseteq \{0,1\}^n = \underbrace{\{0,1\} \times \{0,1\} \times \ \ldots \times \{0,1\}}_{n}, \ Y \subseteq \{0,1\}.$$

(elementami dziedziny są n−elementowe ciągi zerojedynkowymi)

Jeżeli $X = \{0,1\}^n$ to funkcja jest nazywana funkcją zupełną, w przeciwnym przypadku funkcją niezupełną lub funkcją nie w pełni określoną.

Reprezentacja funkcji boolowskich

- opis słowny (np. funkcja przyjmuje wartość 1 wtedy i tylko wtedy gdy jej obydwa argumenty są różne),
- tablice prawdy,
- wyrażenia boolowskie (wyrażenie zawierające zmienne boolowskie i operatory $+$, $\cdot$, $\overline{\ }$, np. $\overline{a} \cdot b + a \cdot \overline{b}$)
- tablice Karnaugha (zmodyfikowane tablice prawdy)
- …

a \ b	0	1
0	0	1
1	1	0

a	b	f
0	0	0
0	1	1
1	0	1
1	1	0

Funkcje boolowskie – postacie kanoniczne

Każdą funkcję boolowską n zmiennych $f(x_1, x_2, \dots, xn)$ można zapisać w postaci:

$$f(x_1, x_2, \dots, x_n) = f(1, x_2, \dots, x_n) \cdot x_1 + f(0, x_2, \dots, x_n) \cdot \overline{x}_1 \qquad (1)$$

Dowód

jeśli $x_1 = 1$ to $f(1, x_2, \dots, x_n) = f(1, x_2, \dots, x_n) \cdot 1 + f(0, x_2, \dots, x_n) \cdot 0 = f(1, x_2, \dots, x_n)$

jeśli $x_1 = 0$ to $f(0, x_2, \dots, x_n) = f(1, x_2, \dots, x_n) \cdot 0 + f(0, x_2, \dots, x_n) \cdot 1 = f(0, x_2, \dots, x_n)$

■

Postać (1) jest nazywana rozwinięciem funkcji ze względu na zmienną x_1. Podobnie funkcje $f(1, x_2, \dots, x_n)$ i $f(0, x_2, \dots, x_n)$ można rozwinąć ze względu na zmienną x_2:

$$f(1, x_2, \dots, x_n) = f(1,1, \dots, x_n) \cdot x_2 + f(1,0, \dots, x_n) \cdot \overline{x}_2$$

$$f(0, x_2, \dots, x_n) = f(0,1, \dots, x_n) \cdot x_2 + f(0,0, \dots, x_n) \cdot \overline{x}_2 \qquad (2)$$

Po podstawieniu (2) do (1) otrzymuje się

$$f(x_1, x_2, \dots, x_n) = f(1,1, \dots, x_n) \cdot x_1 \cdot x_2 + f(1,0, \dots, x_n) \cdot x_1 \cdot \overline{x}_2 + f(0,1, \dots, x_n) \cdot \overline{x}_1 \cdot x_2 + f(0,0, \dots, x_n) \cdot \overline{x}_1 \cdot \overline{x}_2$$

Operację rozwijania można przeprowadzić dla wszystkich zmiennych $x_1, x_2, \dots, x_n$, otrzymuje się wtedy kanoniczną postać dysjunkcyjną funkcji.

Funkcje boolowskie – postacie kanoniczne

Kanoniczna postać dysjunkcyjna (alternatywna) funkcji

$$f(x_1, x_2, \dots, x_n) = f(1,1,\dots,1)\cdot x_1\cdot x_2\cdot \dots \cdot x_n + f(0,1,\dots,1)\cdot \overline{x}_1\cdot x_2\cdot \dots \cdot x_n + f(1,0,\dots,1)\cdot x_1\cdot \overline{x}_2\cdot \dots \cdot x_n + \dots + f(1,1,\dots,0)\cdot x_1\cdot x_2\cdot \dots \cdot \overline{x}_n + f(0,0,\dots,1)\cdot \overline{x}_1\cdot \overline{x}_2\cdot \dots \cdot x_n + \dots + f(0,0,\dots,0)\cdot \overline{x}_1\cdot \overline{x}_2\cdot \dots \cdot \overline{x}_n$$

Kanoniczna postać koniunkcyjna funkcji

Podobnie, funkcję n zmiennych $f(x_1, x_2, \dots, x_n)$ można zapisać w postaci

$$f(x_1, x_2, \dots, x_n) = (f(0,0,\dots,0) + x_1 + x_2 + \dots + x_n)\cdot (f(1,0,\dots,0) + \overline{x}_1 + x_2 + \dots + x_n)\cdot (f(0,1,\dots,0) + x_1 + \overline{x}_2 + \dots + x_n)\cdot \dots \cdot (f(0,0,\dots,1) + x_1 + x_2 + \dots + \overline{x}_n)\cdot (f(1,1,\dots,0) + \overline{x}_1 + \overline{x}_2 + \dots + x_n)\cdot \dots \cdot (f(1,1,\dots,1) + \overline{x}_1 + \overline{x}_2 + \dots + \overline{x}_n)$$

Iloczyn pełny to iloczyn wszystkich argumentów funkcji z negacjami lub bez.

Suma pełna to suma wszystkich argumentów funkcji z negacjami lub bez.

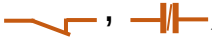
Uwaga! Podczas tworzenia kanonicznej postaci dysjunkcyjnej, można uwzględniać tylko te iloczyny pełne dla których związana z nimi wartość funkcji wynosi 1. Podczas tworzenia kanonicznej postaci koniunkcyjnej, można uwzględniać tylko te sumy pełne dla których związana z nimi wartość funkcji wynosi 0.

Układy cyfrowe

elementy przełączające

Wybrane elementy przełączające

Elementy stykowe to elementy posiadające zestyki lub ich zespoły, wyróżnia się:

- zestyki zwierne (zestyki normalnie otwarte, NO, ),
- zestyki rozwierne (zestyki normalnie zamknięte, NZ, ).

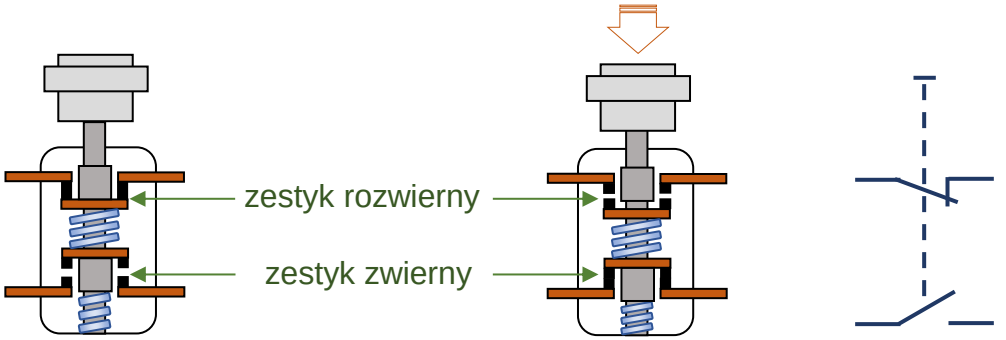
Zestyki mogą być uruchamiane:

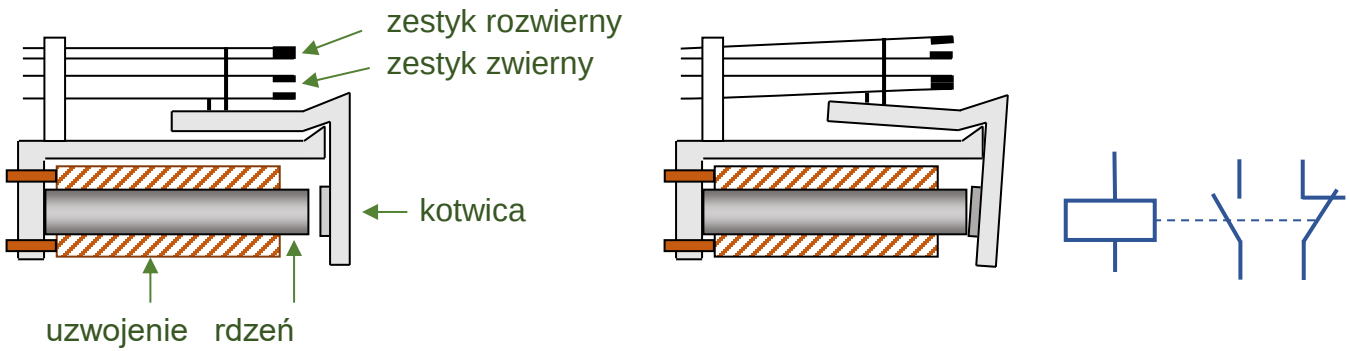
- ręcznie

  np. przyciski,
- mechanicznie

  np. łączniki krańcowe,
- zdalnie

  np. przekaźniki – wykorzystanie siły elektromagnetycznej.

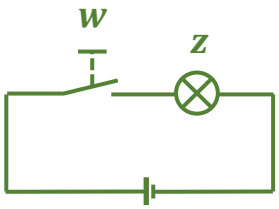




Stykowa realizacja funkcji logicznych

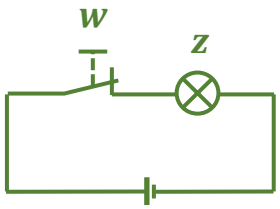
wejście
stan łączników (0 – rozwarte, 1 – zwarte)

wyjście
stan żarówki (0 – nie świeci, 1 – świeci)



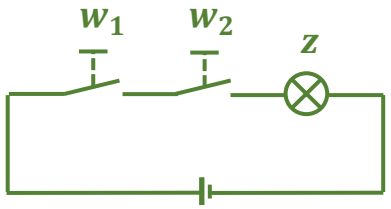
w	z
0	0
1	1

powtórzenie: $z = w$



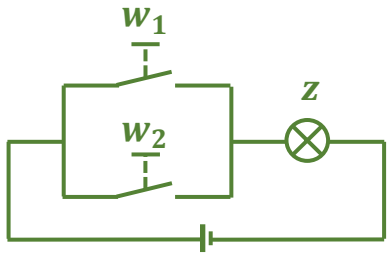
w	z
0	1
1	0

negacja: $z = \bar{w}$



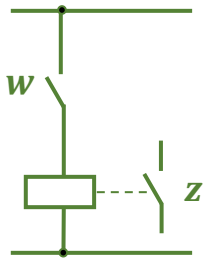
w_1	w_2	z
0	0	0
0	1	0
1	0	0
1	1	1

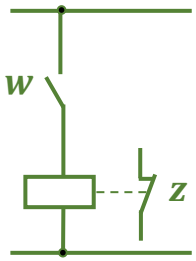
koniunkcja: $z = w_1 \cdot w_2$

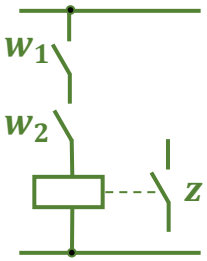


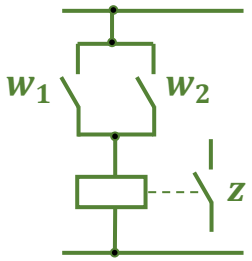
w_1	w_2	z
0	0	0
0	1	1
1	0	1
1	1	1

alternatywa: $z = w_1 + w_2$









wejście
stan zestyków NO (0 – rozwarte, 1 – zwarte)

wyjście
stan styku przekaźnika (0 – rozwarty, 1 – zwarty)

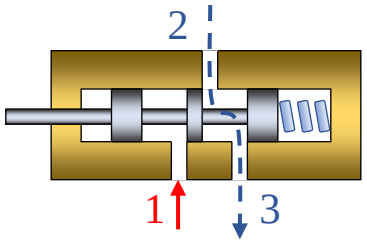
Wybrane elementy przełączające

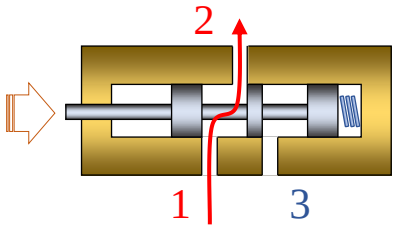
Pneumatyczne elementy przełączające budowane są jako suwakowe czy gniazdowe zawory rozdzielające. Zawory te określają drogę (tzn. początek, koniec i kierunek) przepływu czynnika roboczego, którym jest najczęściej powietrze. Sygnałami wejściowymi elementów są sygnały:

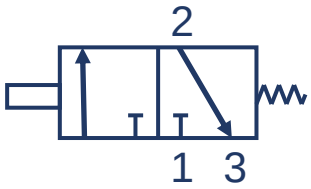
0 – najczęściej ciśnienie atmosferyczne

1 – ciśnienie zasilania.

Zawór rozdzielający 3/2 (o trzech przyłączach i dwóch położeniach)





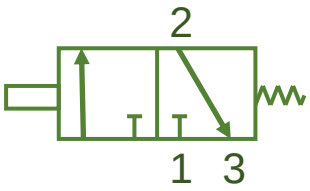


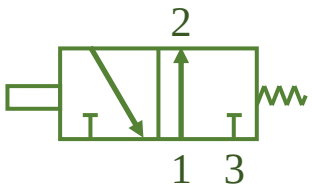
1 – przyłącze zasilania, 2 – przyłącze robocze 3 – odpowietrzenie

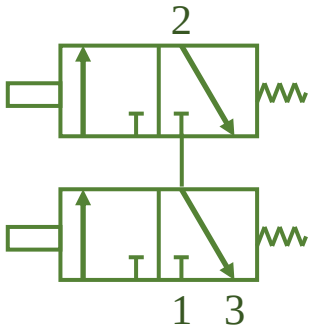
Pneumatyczna realizacja funkcji logicznych

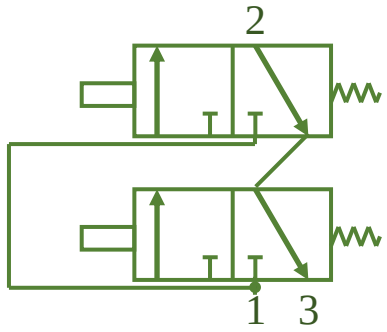
wejście
s – stan zaworu
(0 – stan początkowy, 1 – stan po przełączeniu)

wyjście
c – ciśnienie przyłącza roboczego
(0 – ciśnienie atmosferyczne, 1 – ciśnienie robocze)









s	c
0	0
1	**1**

powtórzenie:
$c = s$

s	c
0	**1**
1	0

negacja:
$c = \bar{s}$

s_1	s_2	c
0	0	0
0	**1**	0
1	0	0
1	**1**	**1**

koniunkcja: $c = s_1 \cdot s_2$

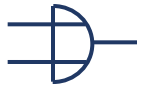
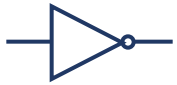
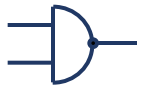
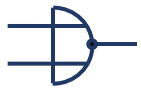
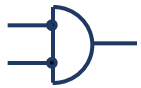
s_1	s_2	c
0	0	0
0	**1**	**1**
1	0	**1**
1	**1**	**1**

alternatywa: $c = s_1 + s_2$

Schematy układów cyfrowych

Schematy logiczne

Przyjmując dla operacji logicznych pewne umowne symbole graficzne można dowolne wyrażenie boolowskie przedstawić w postaci **schematu logicznego**.

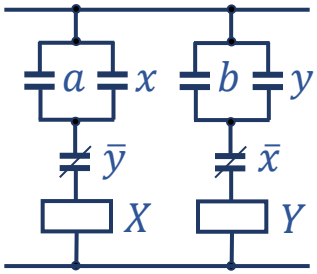
Operacja	1. sposób oznaczeń	2. sposób oznaczeń	3. sposób oznaczeń
koniunkcja (AND)			
alternatywa (OR)			
negacja (NOT)			
negacja koniunkcji (NAND)			 
negacja alternatywy (NOR)	 	 	 

Symbole stosowane do oznaczenia wybranych operacji logicznych

Schematy układów stykowych

Struktura układów realizowanych w technologii stykowo–przekaźnikowej może być przedstawiana na tzw. **schematach obwodowych (drabinkowych)**, na schematach:

- każdy element lub urządzenie obwodu elektrycznego ma swoją gałąź lub tzw. tor prądowy,
- gałęzie rysowane są jako pionowe i umieszczane jedna obok drugiej,
- zestyki przekaźników umieszcza się w gałęziach w zależności od wykonywanego przez nie zadania, nie jest bezpośrednio uwzględniania konstrukcja przekaźnika, tzn. cewka i zestyki przekaźnika mogą leżeć w różnych gałęziach, związek cewki z zestykami przekaźnika wynika z oznaczeń: cewki przekaźników oznaczane są wielkimi literami a ich zestyki tymi samymi literami ale małymi,
- zestyki zwierne oznaczane są małymi literami, zestyki rozwierne małymi literami z kreską nad literą,
- koniunkcja argumentów funkcji logicznej odpowiada szeregowemu połączeniu odpowiednich zestyków,
- alternatywa argumentów funkcji logicznej odpowiada równoległemu połączeniu odpowiednich zestyków.



$$X = (a + x)\overline{y}$$

$$Y = (b + y)\overline{x}$$

Układy kombinacyjne

zadania

Zadania analizy i syntezy

Analiza

Głównym celem analizy jest zrozumienie zasad działania układu, w przypadku gdy znana jest jego struktura; celem analizy może być również chęć wykrycia źródeł niewłaściwego zachowania układu (hazard, wyścig, niestabilność).

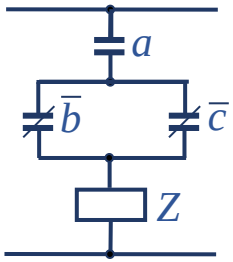
Synteza

Synteza to proces odwrotny do analizy, prowadzi od założeń definiujących sposób działania układu do jego projektu.

Przykład – analiza

Dane

Schemat drabinkowy



Analiza

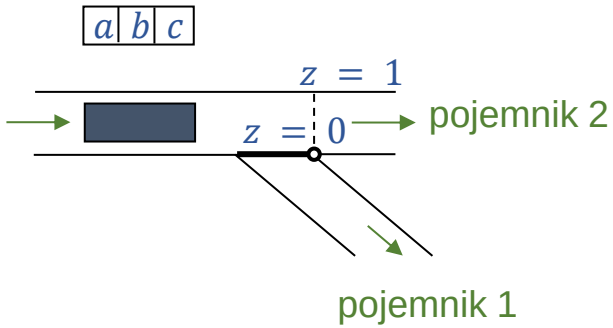
- wyrażenie logiczne: $z = a(\overline{b} + \overline{c})$
- tablica prawdy

a	b	c	$\overline{b}$	$\overline{c}$	$\overline{b} + \overline{c}$	$z = a(\overline{b} + \overline{c})$
0	0	0	1	1	1	0
0	0	1	1	0	1	0
0	1	0	0	1	1	0
0	1	1	0	0	0	0
1	0	0	1	1	1	1
1	0	1	1	0	1	1
1	1	0	0	1	1	1
1	1	1	0	0	0	0

Przykład – synteza

Urządzenie sortujące

- umieszcza detale w jednym z 2 pojemników,
- na stanowisku kontroli badane są 3 cechy (a, b, c) każdego detalu, każda cecha może być 1 (prawidłowa) lub 0 (nieprawidłowa),
- sortowanie realizowane jest z pomocą zwrotnicy, która kieruje detal do pojemnika 1 po otrzymaniu sygnału $z = 1$, do pojemnika 2 gdy $z = 0$,
- do pojemnika 1 przesuwane są detale gdy cecha a jest prawidłowa i nieprawidłowe są cechy b lub c, pozostałe detale przesuwane są do pojemnika 2.



a	b	c	z
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

Funkcje boolowskie – postacie kanoniczne

a	b	c	z
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

Kanoniczna postać koniunkcyjna

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

Kanoniczna postać dysjunkcyjna

$$z = a \cdot \bar{b} \cdot \bar{c} + a \cdot \bar{b} \cdot c + a \cdot b \cdot \bar{c}$$

Funkcje boolowskie

Kanoniczna postać koniunkcyjna

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

prawo idempotentności

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

$(a + b)$ $(a + \bar{b})$ $(\bar{b} + \bar{c})$

a

prawo sklejania

ostatecznie

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

Funkcje boolowskie

Kanoniczna postać dysjunkcyjna

$$z = a \cdot \bar{b} \cdot \bar{c} + a \cdot \bar{b} \cdot c + a \cdot b \cdot \bar{c}$$

prawo idempotentności

$$z = \underbrace{a \cdot \bar{b} \cdot \bar{c} + a \cdot \bar{b} \cdot c}_{a \cdot \bar{b}} + \underbrace{a \cdot \bar{b} \cdot \bar{c} + a \cdot b \cdot \bar{c}}_{a \cdot \bar{c}}$$

prawo sklejania

ostatecznie

$$z = a \cdot \bar{b} + a \cdot \bar{c}$$

prawo rozdzielności

lub

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

Funkcje boolowskie

Kanoniczna postać koniunkcyjna

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

po przekształceniach

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

wszystkie formy funkcji są równoważne

Kanoniczna postać dysjunkcyjna

$$z = a \cdot \bar{b} \cdot \bar{c} + a \cdot \bar{b} \cdot c + a \cdot b \cdot \bar{c}$$

po przekształceniach

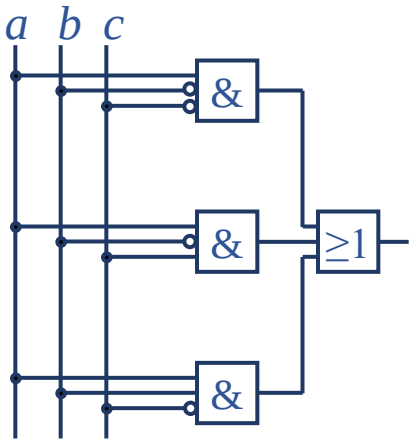
$$z = a \cdot \bar{b} + a \cdot \bar{c}$$

lub

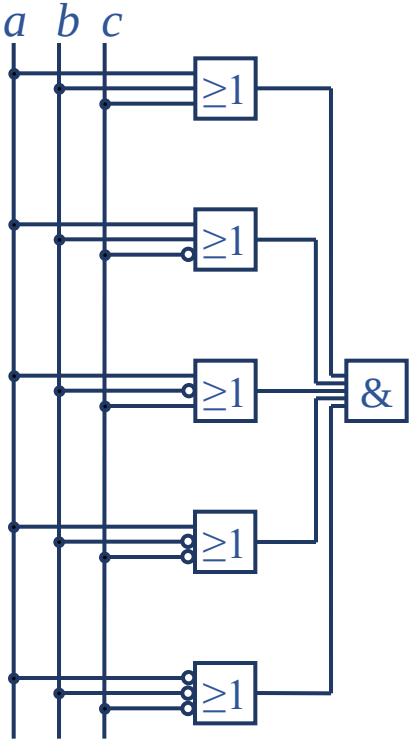
$$z = a \cdot (\bar{b} + \bar{c})$$

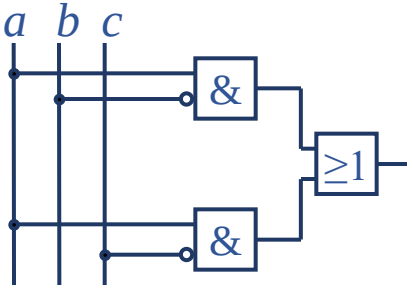
Schematy logiczne

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

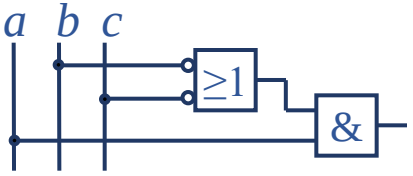


$$z = a\overline{b}\overline{c} + a\overline{b}c + ab\overline{c}$$





$$z = a\overline{b} + a\overline{c}$$



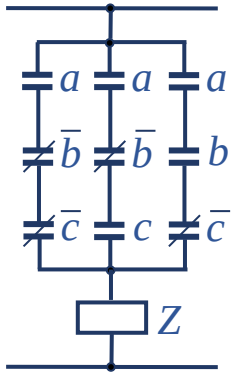
$$z = a(\overline{b} + \overline{c})$$

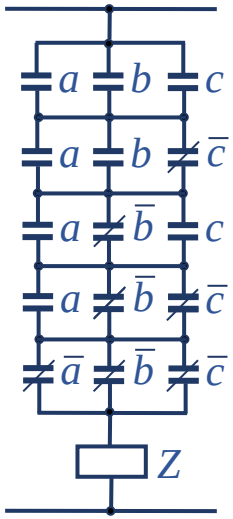
Schematy logiczne urządzenia sortującego (slajd 33)

Schematy układów stykowych

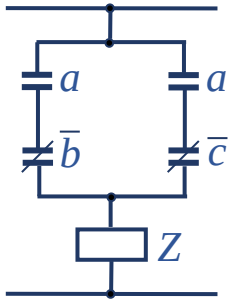
$$z = (a + b + c) \cdot (a + b + \overline{c}) \cdot (a + \overline{b} + c) \cdot (a + \overline{b} + \overline{c}) \cdot (\overline{a} + \overline{b} + \overline{c})$$

$z = a\overline{b}\overline{c} + a\overline{b}c + ab\overline{c}$

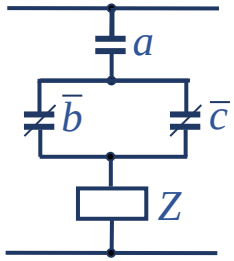




$z = a\overline{b} + a\overline{c}$



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



Schematy obwodowe urządzenia sortującego (slajd 33)

Minimalizacja funkcji logicznych

Minimalizacja funkcji logicznych

Celem minimalizacji funkcji logicznej jest doprowadzenie tej funkcji do postaci o możliwie najmniejszej liczbie argumentów i najmniejszej liczbie operacji logicznych. Minimalizację przeprowadza się wykorzystując:

- prawa algebry Boole'a,
- tablice Karnaugha (inaczej: kraty Veitcha lub cykliczne siatki zależności), w praktyce metoda jest stosowana do minimalizacji funkcji o liczbie argumentów ≤ 6,
- metodę QMC (Quine'a – McCluskeya), metoda może być stosowana do minimalizacji funkcji logicznych o dowolnej liczbie argumentów.

$$a \cdot a = a$$

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

$$a \cdot b + a \cdot \overline{b} = a$$

$$a + a = a$$

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

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

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

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

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

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

Tablice Karnaugha

Tablice Karnaugha ułatwiają stosowanie praw sklejania:

$$ab + a\bar{b} = a, \qquad (a + b)(a + \bar{b}) = a.$$

Sąsiedztwo geometryczne kratek tablicy odpowiada sąsiedztwu logicznemu wyrażeń reprezentowanych przez te kratki.

> Dwa wyrażenia boolowskie określone dla tych samych argumentów z negacjami lub bez, są wyrażeniami sąsiednimi logicznie jeśli różnią się postacią jednego argumentu.

Wyrażenia sąsiednie logicznie można „skleić” np.:

$$\bar{a}bcd + abcd = bcd, \qquad \bar{a}bcd + a\bar{b}cd + abc\bar{d} + a\bar{b}c\bar{d} = bcd + ac\bar{d} = ac.$$

ab \ cd	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	0	0	1	0
10	0	0	1	0

ab \ cd	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	0	0	1	1
10	0	0	1	1

Tablice Karnaugha

Zupełna funkcja boolowska n zmiennych jest określona dla wszystkich możliwych kombinacji sygnałów wejściowych tzn. dziedzina jest zbudowana z 2^n elementów.

Tablica Karnaugha jest kwadratową lub prostokątną siatką zbudowaną z 2^n kratek (dla parzystej liczby sygnałów wejściowych tablica ma kształt kwadratu o wymiarach $2^{0.5n} \times 2^{0.5n}$ dla nieparzystej – prostokąta o wymiarach $2^{0.5(n-1)} \times 2^{0.5(n+1)}$).

Do opisania adresu (nr wiersza i kolumny) kratek wykorzystywany jest **kod Graya**. Adres kratki wyznacza jednoznacznie wartości argumentów funkcji, wartość funkcji (tzn. 0 lub 1) odpowiadającą tym argumentom jest wpisywana we wnętrzu kratki.

a \ b	0	1
0		
1		

a \ bc	00	01	11	10
0				
1				

ab \ cd	00	01	11	10
00				
01				
11				
10				

Tablice Karnaugha dla funkcji o: $n = 2$, $n = 3$ i $n = 4$ argumentach

Tablice Karnaugha

a	b	c	z
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

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

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

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

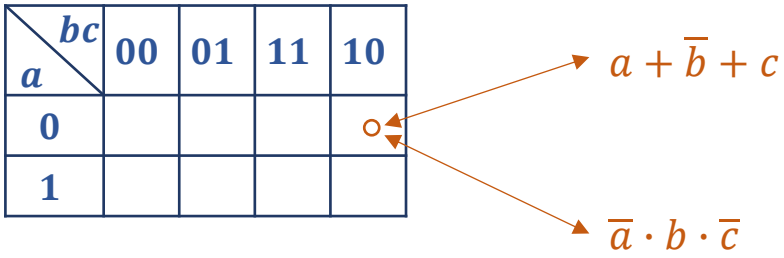
Tablice Karnaugha dla różnej kolejności oznaczeń wierszy i kolumn

Tablice Karnaugha

Każda kratka tablicy odpowiada jednej kombinacji zmiennych wejściowych: jednemu **pełnemu iloczynowi** kanoniczej postaci dysjunkcyjnej lub jednej **pełnej sumie** kanonicznej postaci koniunkcyjnej.

Podobnie jak w przypadku tablic prawdy:

- **iloczyn pełny** odpowiadający wybranej kratce budowany jest w postaci iloczynu argumentów funkcji, które są negowane jeżeli odpowiadają sygnałom o wartości 0,
- **suma pełna** odpowiadająca wybranej kratce budowana jest w postaci sumy argumentów funkcji, które są negowane jeżeli odpowiadają sygnałom o wartości 1.



Tablice Karnaugha

Własności

- sąsiednie kratki odpowiadają wyrażeniom sąsiednim logicznie,

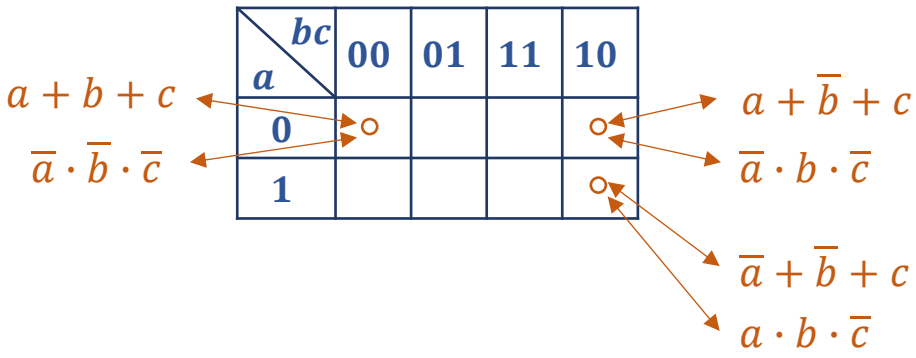
  np. $a + \overline{b} + c$ i $\overline{a} + \overline{b} + c$ czy $\overline{a} \cdot b \cdot \overline{c}$ i $a \cdot b \cdot \overline{c}$

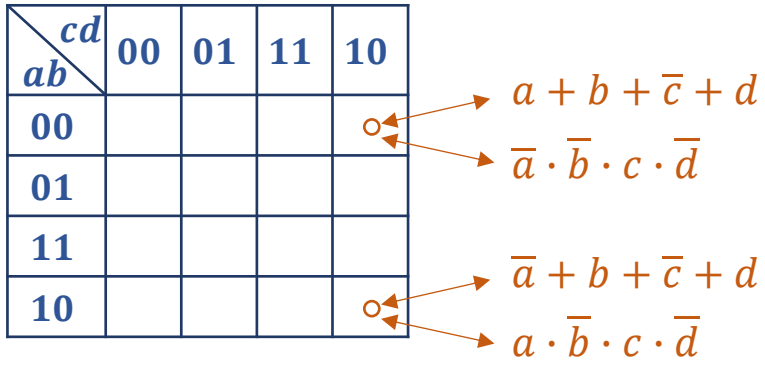
- sąsiednimi są również kratki skrajne: dla 3 zmiennych należy wyobrazić sobie, że sklejone są lewy i prawy brzeg tablicy,

  np. $a + b + c$ i $a + \overline{b} + c$ czy $\overline{a} \cdot \overline{b} \cdot \overline{c}$ i $\overline{a} \cdot b \cdot \overline{c}$

- sąsiedztwo dla 4 zmiennych wprowadzają sklejone brzegi: lewy i prawy oraz górny i dolny,

- dla 5 i 6 zmiennych wprowadzane są dodatkowe linie, które wyznaczają dodatkowe sąsiedztwa.





Minimalizacja funkcji logicznych

Tablice Karnaugha ułatwiają stosowanie praw sklejania: wyodrębniając jak najmniejszą liczbę jak największych grup, które pokrywają wszystkie jedynki lub zera funkcji otrzymuje się postać funkcji zawierającą minimalną liczbę operatorów logicznych.

Sklejanie iloczynów pełnych prowadzi do **normalnej postaci dysjunkcyjnej**, a sklejanie sum pełnych do **normalnej postaci koniunkcyjnej**.

Normalna postać dysjunkcyjna (koniunkcyjna) jest sumą iloczynów (iloczynem sum) dowolnej liczby argumentów funkcji z negacjami lub bez.

Metoda Karnaugha prowadząc do wyznaczenia minimalnej normalnej postaci dysjunkcyjnej (koniunkcyjnej) polega na:

- wyszukiwaniu i zaznaczeniu wśród niezaznaczonych jeszcze kratek tablicy samodzielnych grup jedynek (zer) obejmujących 2^n kratek ($n = \ldots, 3, 2, 1, 0$),
- jeżeli po wyodrębnieniu wszystkich samodzielnych grup pozostają jeszcze niezaznaczone jedynki (zera) to należy je połączyć z kratkami zaznaczonymi tak aby zaznaczona w ten sposób grupa reprezentowała iloczyn (sumę) o najmniejszej liczbie argumentów.

Minimalizacja funkcji logicznych

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

* $a \cdot \overline{b} \cdot \overline{c} + a \cdot \overline{b} \cdot c = a \cdot \overline{b}$

** $a \cdot \overline{b} \cdot \overline{c} + a \cdot b \cdot \overline{c} = a \cdot \overline{c}$

Ostatecznie

$$z = a \cdot \overline{b} + a \cdot \overline{c}$$

lub

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

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

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

Minimalna normalna postać dysjunkcyjna

na 2 sposoby można wyodrębnić 2 –elementową grupę zbudowaną z "1" (załóżmy, że zaznaczona została grupa oznaczona *), pozostała "1" musi tworzyć grupę z już zaznaczoną "1" (grupa **), ostatecznie: $z = a \cdot \overline{b} + a \cdot \overline{c}$.

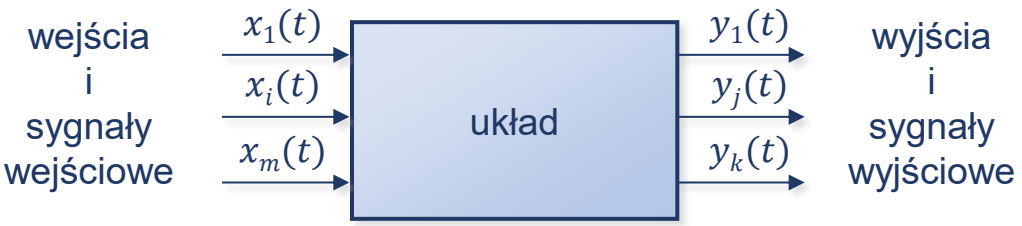
Minimalna normalna postać koniunkcyjna

można wyodrębnić 4 –elementową grupę zbudowaną z "0" (grupa ●), pozostałe "0" musi tworzyć grupę z już zaznaczonym "0" (grupa ●●), ostatecznie: $z = a \cdot (\overline{b} + \overline{c})$.

Układy wielowyjściowe

- generują kilka sygnałów wyjściowych, które zależą od tego samego zbioru sygnałów wejściowych,
- działanie układów jest opisywane układem funkcji logicznych,

  układ o m wejściach i k wyjściach można zapisać w postaci układu k funkcji boolowskich o m argumentach, tzn. każda z funkcji opisuje jedno z wyjść układu,
- minimalizację opisu układu można przeprowadzić oddzielnie dla każdego wyjścia ale cały układ najczęściej może być zapisany jeszcze prościej ponieważ w wyrażeniach mogą występować wspólne elementy (można je wykryć analizując tablice Karnaugha co w przypadku złożonych układów nie jest to proste lub stosując zmodyfikowaną metodę QMC).



Układy wielowyjściowe

Układ włącz-wyłącz

- układ pracuje w 2 trybach ustawianych łącznikiem a (tryb normalny $a = 0$, tryb oszczędzania energii $a = 1$),
- czujniki b i c monitorują stan pewnego parametru systemu,
- w trybie normalnym wysoki stan czujnika b włącza urządzenie u_1 a wysoki stan czujnika c urządzenie u_2,
- w trybie oszczędzania energii urządzenia są włączane tylko gdy stan obydwu czujników jest wysoki.

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

u_1

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

u_2

a	b	c	u_1	u_2
0	0	0	0	0
0	0	1	0	1
0	1	0	1	0
0	1	1	1	1
1	0	0	0	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

Układy wielowyjściowe

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

u_1

$b \cdot c$

$\overline{a} \cdot b$

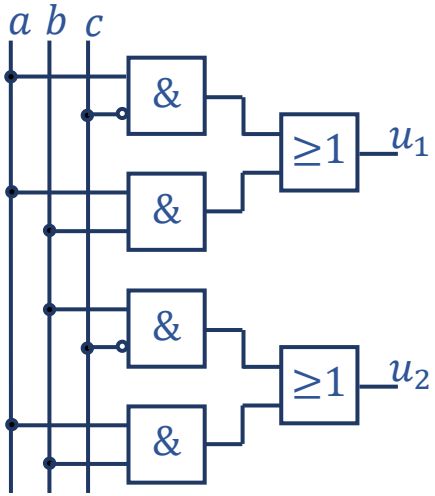
$\overline{a} \cdot c$

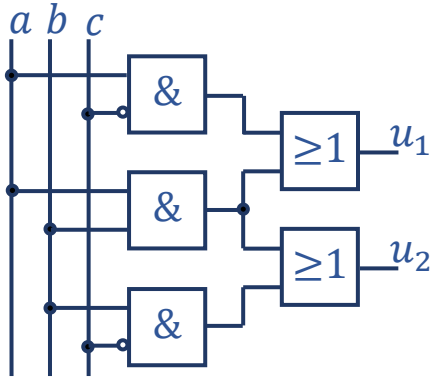
a \ bc	00	01	11	10
0	0	1	1	0
1	0	0	1	0

u_2

$$u_1 = b \cdot c + \overline{a} \cdot b$$

$$u_2 = b \cdot c + \overline{a} \cdot c$$





Stany nieokreślone

Stany nieokreślone

- stany wyjść układu nie zawsze określone są dla wszystkich stanów wejściowych, w takim przypadku układy opisywane są funkcjami logicznymi nie w pełni określonymi (funkcjami niezupełnymi,
- stany nieokreślonych w tablicach prawdy i tablicach Karnaugha opisywane są symbolami: kreska (–), krzyżyk (x), itp.,
- funkcje logiczne mają tylko dwie dopuszczalne wartości: 0 i 1, wartość nieokreślona nie jest trzecią wartością funkcji, minimalizując funkcje logiczne każdą wartość nieokreśloną należy wykorzystać jako 0 lub 1 starając uzyskać jak najprostszą postać funk

a	b	c	z
0	0	0	0
0	0	1	–
0	1	0	–
0	1	1	–
1	0	0	1
1	0	1	–
1	1	0	0
1	1	1	1

a \ bc	00	01	11	10
0	0	–	–	–
1	1	–	1	0

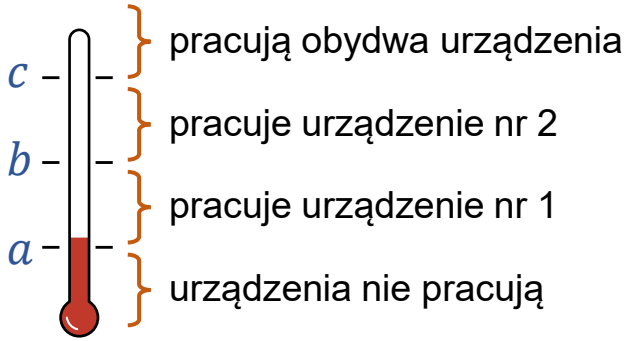
stan nieokreślony potraktowany jak stan 0

stany nieokreślone potraktowane jak stan 1

Stany nieokreślone

Układ włącz-wyłącz

- układ włącza i wyłącza 2 urządzenia w zależności od temperatury mierzonej przy pomocy 3 czujników,
- czujniki a, b i c generują sygnał o wartości 1 jeśli mierzona temperatura przekroczy zadaną wartość progową,
- temperatury progowe są dobrane w taki sposób, że przy wzroście temperatury najpierw reaguje czujnik a, później czujnik b a na końcu czujnik c.



a	b	c	u_1	u_2
0	0	0	0	0
0	0	1	–	–
0	1	0	–	–
0	1	1	–	–
1	0	0	1	0
1	0	1	–	–
1	1	0	0	1
1	1	1	1	1

Stany nieokreślone

a	b	c	u_1	u_2
0	0	0	0	0
0	0	1	–	–
0	1	0	–	–
0	1	1	–	–
1	0	0	1	0
1	0	1	–	–
1	1	0	0	1
1	1	1	1	1

a \ bc	00	01	11	10
0	0	–	–	–
1	1	–	1	0

u_1

a \ bc	00	01	11	10
0	0	–	–	–
1	0	–	1	1

u_2

$$u_1 = a\,\bar{b} + c$$

$$u_2 = b$$