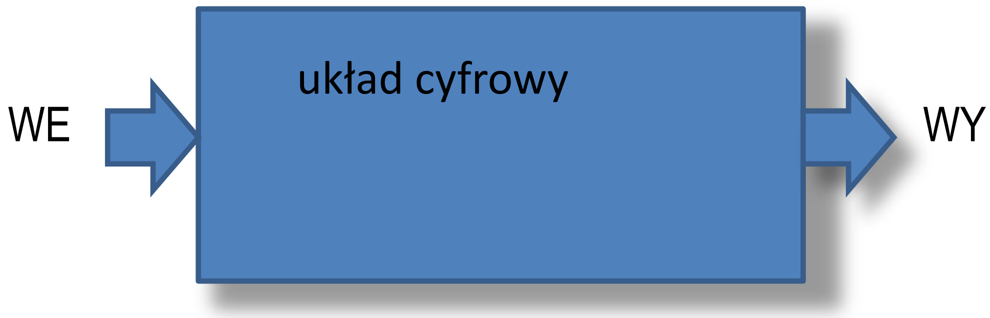
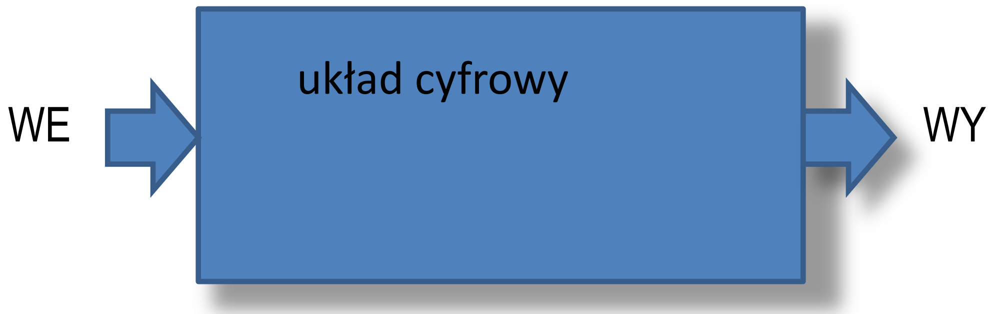
w1

- *układy cyfrowe*
- *algebra Boole'a*
- *funkcje boolowskie*
- *bramki logiczne*
- *projektowanie układów kombinacyjnych*
- *minimalizacja funkcji*

Układy cyfrowe





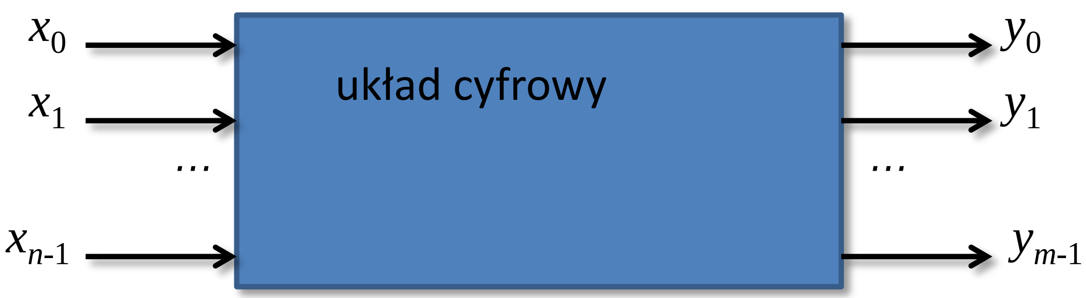
sygnał WE (input)
$X = \langle x_0, \ldots, x_{n-1} \rangle$

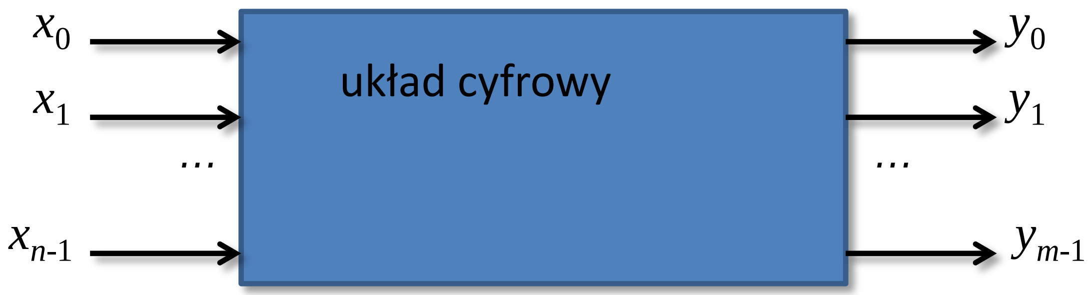
sygnał WY (output)
$Y = \langle y_0, \ldots, y_{m-1} \rangle$

Układy cyfrowe

sygnał WE (input)
$X = \langle x_0, \ldots, x_{n-1} \rangle$

sygnał WY (output)
$Y = \langle y_0, \ldots, y_{m-1} \rangle$





sygnał WE (input)
$X = \langle x_0, \ldots, x_{n-1} \rangle$

sygnał WY (output)
$Y = \langle y_0, \ldots, y_{m-1} \rangle$

sygnały WE oraz WY to sygnały **CYFROWE**

sygnał WE (input)
$X = \langle x_0, \ldots, x_{n-1} \rangle$

sygnał WY (output)
$Y = \langle y_0, \ldots, y_{m-1} \rangle$

sygnały WE oraz WY to sygnały **CYFROWE**

sygnał **cyfrowy** przyjmuje wartości ze <u>skończonego</u> zbioru

Układy cyfrowe

sygnał WE (input)
$X = <x_0, \ldots, x_{n-1}>$

sygnał WY (output)
$Y = <y_0, \ldots, y_{m-1}>$

sygnały WE oraz WY to sygnały **CYFROWE**

sygnał **cyfrowy** przyjmuje wartości ze <u>skończonego</u> zbioru

przeliczalny, **dyskretny**

sygnał WE (input)
$X = <x_0, \ldots, x_{n-1}>$

sygnał WY (output)
$Y = <y_0, \ldots, y_{m-1}>$

sygnały WE oraz WY to sygnały **CYFROWE**

sygnał **cyfrowy** przyjmuje wartości ze skończonego zbioru

… o dwuelementowym zbiorze
wartości, to sygnał binarny

sygnał cyfrowy
o dwuelementowym zbiorze wartości, to sygnał binarny

logiczne zero „0”
logiczne jeden „1”

skoro logiczne, to co reprezentuje w warstwie fizycznej?
jak jest fizyczna reprezentacja wartości logicznych?

w technologii TTL (Transistor – Transistor Logic)

logiczne zero	„0” ⇔ 0 ÷ 0,8 V
logiczne jeden	„1” ⇔ 2,4 ÷ 5 V

zasilanie 5 V	
czas propagacji	1,5 ÷ 3 ns
obciążalność	10

mamy sygnał cyfrowy w postaci ciągu zer i jedynek, 1110100110101011
co reprezentuje taki ciąg?

111010011010101 1

liczbę?
znak?
obraz?
nic?

co dany zestaw (wektor) znaków reprezentuje?
co zostało zakodowane za pomocą tego wektora?

1110
1001
1110 ≡ R
1010
1001

11101001111010101001 = $958\ 121_d$

11101001111010101001 = $E\ 9EA9_h$

11101001111010101001 ≡ bażant

1110	1110
1001	1001
1110	1110
1010	1010
1001	1001

11101001111010101001 = 958 121$_d$

$2^{19} + 2^{18} + 2^{17} + 2^{15} + 2^{12} + 2^{11} + 2^{10} + 2^{9} + 2^{7} + 2^{5} + 2^{3} + 2^{0}$

1110 1001 1110 1010 1001 = E 9EA9$_h$

E 9 E A 9

$x16^4 + x16^3 + x16^2 + x16^1 + x16^0$

111 010 01111 010 101 001 ≡ bażant

11101001111010101001 = 958 121$_d$

$2^{19} + 2^{18} + 2^{17} + 2^{15} + 2^{12} + 2^{11} + 2^{10} + 2^{9} + 2^{7} + 2^{5} + 2^{3} + 2^{0}$

wektor bitowy reprezentuje liczbę w systemie dwójkowym

binarnym

o podstawie 2

1110 1001 1110 1010 1001 = E 9EA9$_h$

$$\left.\begin{array}{ccccc} E & 9 & E & A & 9 \\ \times 16^4 + & \times 16^3 + & \times 16^2 + & \times 16^1 + & \times 16^0 \end{array}\right\} = 958\,121_d$$

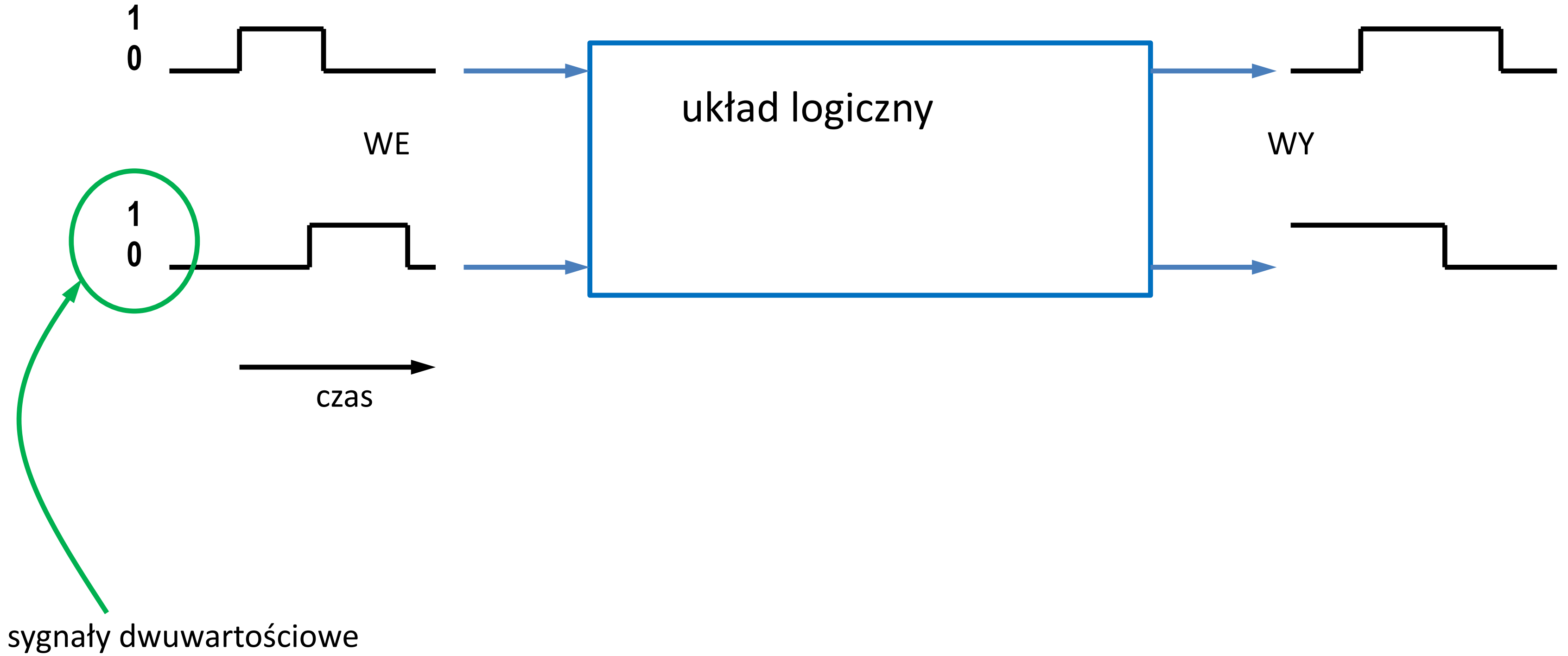
wektor bitowy reprezentuje liczbę w systemie

szesnastkowym

heksadecymalnym

o podstawie 16

Sygnały logiczne



uwaga:

konwencja logiki dodatniej: wyższe napięcie reprezentuje stan logicznej „1”, niższe stan logicznego „0”

konwencja logiki ujemnej: wyższe napięcie reprezentuje stan logicznego „0”, niższe stan logicznej „1”

Algebra Boole'a

- podstawowy aparat matematyczny wspierający opis układów logicznych
- opracowany w 1854 roku przez *Georga Boole'a*, anglika

Definicja

P – zbiór z określonymi operacjami dwuargumentowymi „**+**” i „**•**” ($\cup$ i $\cap$)
0, **1** – wyróżnione elementy zbioru ***P***

Operację „**+**” nazywamy alternatywą (to **lub** to), sumą logiczną
Operację „**•**” nazywamy koniunkcją (to **oraz** to), iloczynem logicznym

0 i **1** są elementami neutralnymi względem sumy i iloczynu

Definicja

P – zbiór z określonymi operacjami dwuargumentowymi „**+**” i „**•**”
0, **1** – wyróżnione elementy zbioru ***P***
$\forall$ ***a***, ***b***, ***c*** $\in$ ***P*** spełnione są aksjomaty:

(A) przemienność

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

(B) rozdzielność

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

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

(C) elementy neutralne działań

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

(D) pochłanianie

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

Algebra Boole'a

Operację „**+**” nazywamy alternatywą (to **lub** to), sumą logiczną
Operację „**•**” nazywamy koniunkcją (to **oraz** to), iloczynem logicznym

0 i **1** $\in$ ***P*** są elementami neutralnymi względem sumy i iloczynu

Można przyjąć, jak w arytmetyce, priorytet iloczynu nad sumą.

Znak „**•**” jest najczęściej pomijany

(Znak * jest/będzie używany do oznaczenia tak zwanej iteracji; stąd „·” a nie *)

UWAGA!
Proszę nie stosować zasad arytmetyki do operacji logicznych bez zastanowienia.
Patrz aksjomat B**!**

Na bazie aksjomatów (A) do (D) można wyprowadzić:

(E) łączność

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

(F) zasada pochłaniania

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

(G) $a + /a = 1 \qquad a \cdot a = a$

(H) Prawa de Morgana

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

(I) $/(/a) = a$

(M) $a\, /b + a \cdot b = ?$ (prawo sklejania)

Funkcja boolowska (~~booleowska~~)

Funkcja boolowska n argumentowa (zmiennych binarnych), to odwzorowanie

$$f : X_n \to Y,$$

lub

$$f : X_n \to \{0, 1\}$$

Funkcja boolowska n zmiennych jest równoważna układowi kombinacyjnemu o n wejściach i jednym wyjściu

Układy wielowejściowe można traktować jako zestawienie tylu funkcji boolowskich, ile jest wyjść w układzie.

Opis funkcji boolowskiej - tabela prawdy*

funkcja jednej zmiennej

x	$f(x)$
0	1
1	0

np. negacja $f(x) = /x$

x	$/x$
0	1
1	0

funkcja dwóch zmiennych

x_1	x_0	$f(x_1, x_0)$
0	0	0
0	1	1
1	0	1
1	1	1

np. alternatywa
$f(x_0, x_1) = x_0 \vee x_1$

x_1	x_0	$x_1 \vee x_0$
0	0	0
0	1	1
1	0	1
1	1	1

* matryca logiczna, opracowana przez Charlesa Sandersa Peirce'a i Emila Leona Posta w XIX w.

Opis funkcji boolowskiej - zbiory zer i jedynek

x_1	x_0	$x_1 \vee x_0$
0	0	0
0	1	1
1	0	1
1	1	1

$f^1 = \begin{bmatrix} 01 \\ 10 \\ 11 \end{bmatrix}$ – zbiór jedynek w postaci binarnej

$f^0 = [00]$ – zbiór zer w postaci binarnej

$f^0 = \{0\}$ – zbiór zer w postaci dziesiętnej

$f^1 = \{1, 2, 3\}$ – zbiór jedynek w postaci dziesiętnej

Opis funkcji boolowskiej - kanoniczna postać sumy (1)

Funkcja $f\,(\,x_{0,}\,x_{1,}\,\ldots,\,x_{n\text{-}1}\,)$ może być rozłożona na czynniki względem dowolnego argumentu x_k.

Rozkładając względem x_0 otrzymujemy:

$$f\,(\,x_{0,}\,x_{1,}\,\ldots,\,x_{n\text{-}1}\,) = x_0\;f_1\,(\,x_{1,}\,x_{2,}\,\ldots,\,x_{n\text{-}1}\,) + /x_0\;f_2\,(\,x_{1,}\,x_{2,}\,\ldots,\,x_{n\text{-}1}\,)$$

dla $x_0 = 1$

$$f\,(\,1_,\,x_{1,}\,\ldots,\,x_{n\text{-}1}\,) = f_1\,(\,x_{1,}\,x_{2,}\,\ldots,\,x_{n\text{-}1}\,) + 0\;f_2\,(\,x_{1,}\,x_{2,}\,\ldots,\,x_{n\text{-}1}\,)$$

dla $x_0 = 0$

$$f\,(\,0_,\,x_{1,}\,\ldots,\,x_{n\text{-}1}\,) = 0\;f_1\,(\,x_{1,}\,x_{2,}\,\ldots,\,x_{n\text{-}1}\,) + f_2\,(\,x_{1,}\,x_{2,}\,\ldots,\,x_{n\text{-}1}\,)$$

Opis funkcji boolowskiej - kanoniczna postać sumy (2)

$$f(x_0, x_1, \ldots, x_{n-1}) = x_0\, f(x_1, x_2, \ldots, x_{n-1}) + /x_0\, f(x_1, x_2, \ldots, x_{n-1})$$

dla $x_0 = 1$

$$f(1, x_1, \ldots, x_{n-1}) = f(x_1, x_2, \ldots, x_{n-1}) + \cancel{0\, f(x_1, x_2, \ldots, x_{n-1})}$$

dla $x_0 = 0$

$$f(0, x_1, \ldots, x_{n-1}) = \cancel{0\, f(x_1, x_2, \ldots, x_{n-1})} + f(x_1, x_2, \ldots, x_{n-1})$$

Zatem

$$f(x_0, x_1, \ldots, x_{n-1}) = x_0\, f(1, x_1, x_2, \ldots, x_{n-1}) + /x_0\, f(0, x_1, x_2, \ldots, x_{n-1})$$

i dalej (tu: względem x_1),

$$f(x_0, x_1, \ldots, x_{n-1}) = x_0\, [\, x_1\, f(1, 1, x_2, \ldots, x_{n-1})] + /x_0\, [\, /x_1\, f(0, 0, x_2, \ldots, x_{n-1})]$$

Po rozłożeniu względem wszystkim x_k powstaje tzw. …

Opis funkcji boolowskiej - kanoniczna postać sumy (3)

Po rozłożeniu względem wszystkim x_k powstaje tzw. ***kanoniczna postać sumy***

$$
\left.
\begin{aligned}
f(x_0, x_1, \ldots, x_{n-1}) = {} & x_0\, x_1, \ldots, x_{n-1}\ f(1, 1, \ldots, 1) + \\
& x_0\, x_1, \ldots, /x_{n-1}\ f(1, 1, \ldots, 0) + \\
& . \\
& . \\
& . \\
& /x_0\, /x_1, \ldots, x_{n-1}\ f(0, 0, \ldots, 1) + \\
& /x_0\, /x_1, \ldots, /x_{n-1}\ f(0, 0, \ldots, 0).
\end{aligned}
\right\} \; 2^n \textit{ składników}
$$

Składnikami tej postaci są tzw. *iloczyny pełne*.
Każdy iloczyn pełny odpowiada wartości funkcji dla określonego wektora wejściowego.

jaki jest związek tabeli prawdy z postacią kanoniczną (tu: sumy iloczynów)?

Opis funkcji boolowskiej - kanoniczna postać sumy (4)

Przejście z tabeli prawdy do wyrażenia w algebrze Boole’a polega na:

wyszukaniu w tabeli prawdy wierszy, w których wartość funkcji wynosi 1,
zapisaniu odpowiadającego jej iloczynu pełnego.

Poprawny zapis algebraiczny funkcji stanowi sumę tych iloczynów pełnych, dla których funkcji przyjmuje wartość 1.

Taką postać funkcji boolowskiej to tzw. pierwsza postać kanoniczna (1pk) lub postać kanoniczna sumy.

x_1	x_0	$x_1 \vee x_0$
0	0	0
0	1	1
1	0	1
1	1	1

$$f(x_{1,}\, x_0) = x_1\, x_0\, f(0, 1) + x_1\, /x_0\, f(1, 0) + /x_1\, x_0\, f(1, 1).$$

Opis funkcji boolowskiej - kanoniczna postać iloczynu (1)

Funkcja $f\,(x_0, x_1, \ldots, x_{n\text{-}1})$ może być rozłożona na czynniki względem dowolnego argumentu x_k.

Rozkładając względem x_0 otrzymujemy:

$$f\,(x_0, x_1, \ldots, x_{n\text{-}1}) = [\,x_0 + f_1\,(x_1, x_2, \ldots, x_{n\text{-}1})][\,/x_0 + f_2\,(x_1, x_2, \ldots, x_{n\text{-}1})]$$

dla $x_0 = 1$

$$f\,(1, x_1, \ldots, x_{n\text{-}1}) = [\,1 + f_1\,(x_1, x_2, \ldots, x_{n\text{-}1})][\,0 + f_2\,(x_1, x_2, \ldots, x_{n\text{-}1})]$$

dla $x_0 = 0$

$$f\,(0, x_1, \ldots, x_{n\text{-}1}) = [\,0 + f_1\,(x_1, x_2, \ldots, x_{n\text{-}1})][\,1 + f_2\,(x_1, x_2, \ldots, x_{n\text{-}1})]$$

Opis funkcji boolowskiej - kanoniczna postać iloczynu(2)

$$f(x_0, x_1, \dots, x_{n-1}) = [x_0 + f_1(x_1, x_2, \dots, x_{n-1})][/x_0 + f_2(x_1, x_2, \dots, x_{n-1})]$$

dla $x_0 = 1$

$$f(1, x_1, \dots, x_{n-1}) = [\cancel{1 + f_1(x_1, x_2, \dots, x_{n-1})}][0 + f_2(x_1, x_2, \dots, x_{n-1})]$$

dla $x_0 = 0$

$$f(0, x_1, \dots, x_{n-1}) = [0 + f_1(x_1, x_2, \dots, x_{n-1})][\cancel{1 + f_2(x_1, x_2, \dots, x_{n-1})}]$$

Zatem

$$f(x_0, x_1, \dots, x_{n-1}) = [x_0 + f(0, x_2, \dots, x_{n-1})][/x_0 + f(1, x_2, \dots, x_{n-1})]$$

i dalej,

$$f(x_0, x_1, \dots, x_{n-1}) =$$

$$= \{x_0 + [x_1 + f(0, 0, x_2, \dots, x_{n-1})][/x_1 + f(0, 1, x_2, \dots, x_{n-1})]\}\{/x_1 + [\dots][\dots]\} =$$

$$= [x_0 + x_1 + f(0, 0, x_2, \dots, x_{n-1})][x_0 + /x_1 + f(0, 1, x_2, \dots, x_{n-1})] \cdot$$
$$\cdot [/x_0 + x_1 + f(1, 0, x_2, \dots, x_{n-1})][/x_0 + /x_1 + f(1, 1, x_2, \dots, x_{n-1})]$$

Po rozłożeniu względem wszystkim x_k powstaje tzw. …

Opis funkcji boolowskiej - kanoniczna postać iloczynu(3)

Po rozłożeniu względem wszystkim x_k powstaje tzw. ***kanoniczna postać iloczynu***

$$
\left.
\begin{aligned}
f(x_0, x_1, \ldots, x_{n-1}) = {} & [x_0 + x_1 + \ldots + x_{n-1} + f(0, 0, \ldots, 0)] + \\
& [x_0 + x_1 + \ldots + /x_{n-1} + f(0, 0, \ldots, 1)] + \\
& . \\
& . \\
& . \\
& [/x_0 + /x_1 + \ldots + x_{n-1} + f(1, 1, \ldots, 0)] + \\
& [/x_0 + /x_1 + \ldots + /x_{n-1} + f(1, 1, \ldots, 1)].
\end{aligned}
\right\} \; 2^n \text{ składników}
$$

Składnikami tej postaci są tzw. *sumy pełne*.
Każda suma pełna odpowiada wartości funkcji dla określonego wektora wejściowego.
Znikają te czynniki iloczynu, dla których $f() = 1$. Zostają zera (wartości) funkcji.

jaki jest związek tabeli prawdy z postacią kanoniczną (tu: iloczynu sum)?

Opis funkcji boolowskiej - kanoniczna postać iloczynu(4)

Przejście z tabeli prawdy do wyrażenia w algebrze Boole'a polega na:

wyszukaniu w tabeli prawdy wierszy, w których wartość funkcji wynosi 0,
zapisaniu odpowiadającego jej sumy pełnej.

Poprawny zapis algebraiczny funkcji stanowi sumę tych iloczynów pełnych, dla których funkcji przyjmuje wartość 0.

Taką postać funkcji boolowskiej to tzw. druga postać kanoniczna (2pk) lub postać kanoniczna iloczynu.

x_1	x_0	$x_1 \wedge x_0$
0	0	0
0	1	0
1	0	0
1	1	1

$$f(x_{1,} x_0) = [\, x_1 + x_0 + f(0, 0)][\, /x_1 + x_0 + f(0, 1)][\, x_1 + /x_0 + f(1, 0)].$$

Systemy funkcjonalne pełne

Zbiór funkcji boolowskich nazywamy *zbiorem funkcjonalnie pełnym*, jeśli dowolna funkcja boolowska daje się przedstawić jako superpozycja funkcji tego zbioru oraz stałych 0 i 1.

1) {NOT, AND, OR} : podstawowy zbiór funkcjonalnie pełny

Aby wykazać, że pewien zbiór jest funkcjonalnie pełny, wystarczy pokazać jak za jego pomocą realizować trzy podstawowe funkcji: {NOT, AND, OR}.

2) {NOT, AND}

$$x_0 + x_1 = /(/x_0 /x_1)$$

3) {NOT, OR}

$$x_0\ x_1 = /(/x_0 + /x_1)$$

z praw de Morgana (J)

4) {NAND}

$$/x_0 = /(x_0\ x_0)$$
$$x_0\ /x_1 = /(/(x_0\ x_1) /(x_0\ x_1))$$
$$x_0 + /x_1 = /(/(x_0\ x_0) /(x_1\ x_1))$$

5) {NOR}

$$/x_0 = /(x_0 + x_0)$$
$$x_0\ /x_1 = /(/(x_0 + x_0) + /(x_1 + x_1))$$
$$x_0 + /x_1 = /(/(x_0 + x_1) /(x_0 + x_1))$$

Zapis skrócony funkcji boolowskiej

Postaci kanoniczne można zapisać skrótowo, podając jedynie indeksy iloczynów bądź sum pełnych niezbędnych w odpowiedniej postaci kanonicznej.

$$y(x_2, x_1, x_0) = \sum (0, 2, 5, 7)$$

$$y(x_2, x_1, x_0) = \prod (1, 3, 4, 6)$$

jak wygląda rozwinięcie poniższych zapisów do postaci 1pk i 2pk?

$$y(x_2, x_1, x_0) = \sum (0, 2, 5, 7)$$

$$y(x_2, x_1, x_0) = \prod (1, 3, 4, 6)$$

Minimalizacja funkcji boolowskich (1)

Układ kombinacyjny może zostać zrealizowany na wiele sposobów.
Układowi opisanemu pewną tabelą prawdy może odpowiadać wiele różnych wyrażeń boolowskich, a tym samym wiele realizacji.

Celem minimalizacji jest realizacja układu z wykorzystaniem minimum środków, czyli

- minimalna liczba bramek (w ogóle),
- minimalna liczba bramek określonego typu,
- wykorzystanie bramek określonego typu,
- eliminacja bramek wielowejściowych,
- minimalizacja liczby użytych układów scalonych (w TTL 4x NAND (1 u.s.) < 2x NAND + 1x OR (2 u.s.)),
- użycie układów o określonej strukturze wewnętrznej: np. układy PLD / FPGA

Minimalizacja funkcji boolowskich (2)

Metoda przekształceń

Polega na wykorzystaniu tożsamości algebry Boole’a do uproszczenia wyrażenia opisującego funkcję działania układu logicznego.

Metoda intuicyjna.

Szybka i poprawna minimalizacja wymaga doświadczenia i w niektórych wypadkach jest sztuką.

Trudność może sprawiać odgadnięcie zależności bądź kolejności użycia zależności prowadzących do uzyskania wyniku minimalnego.

Trudne jest również ustalenie, czy dana postać jest postacią minimalną.

Metoda przekształceń – przykład (1)

funkcja większościowa (*czy tylko?*)

$f(x_1, x_2, x_3) = \sum(011, 101, 110, 111) =$

$= /x_1\, x_2\, x_3 + x_1\, /x_2\, x_3 + x_1\, x_2\, /x_3 + x_1\, x_2\, x_3 =$

(H)(B) $= (/x_1 + x_2)\, x_2\, x_3 + (/\, x_2 + x_2)\, x_1\, x_3 + (/x_3 + x_3)\, x_1\, x_2 =$

(D)(C) $= x_2\, x_3 + x_1\, x_3 + x_1\, x_2$

układ

Jak wygląda schemat układu dla postaci kanonicznej, a jak dla postaci minimalnej?

Metoda przekształceń – przykład (2)

funkcja …

$$f\,(a, b, c_i) = /a\ b\ /c_i + a\ /b\ /c_i + /a\ /b\ c_i + a\ b\ c_i$$

$$(?_1)(?_2) \quad = /c_i\ (\ /a\ b + a\ /b\ ) + c_i\ (\ /a\ /b + a\ b\ ) =$$

$$(?_3)(?_4) \quad = /c_i\ (\ a \oplus b\ ) + c_i\ (\ /a\ /b + a\ b\ ) =$$

$$(?_5)(?_6) \quad = c_i \oplus a \oplus b$$

Jak wygląda schemat układu dla postaci kanonicznej, a jak dla postaci minimalnej?

Metoda siatek (map) *Karnaugha** (1)

Postaci kanoniczne f.b. nie muszą być postaciami minimalnymi/najprostszymi.
Można je uprościć (zredukować liczbę symboli w zapisie bez zmiany realizowanej funkcji) stosując metodę graficznej reprezentacji tabeli prawdy, ułatwiająca minimalizację wyrażenia wg reguły sklejania:

sumę lub iloczyn dwu wyrażeń różniących się tylko znakiem nad jedną zmienną można zastąpić jednym wyrażeniem, redukując zmienną stanowiącą różnicę.

$A\,/x + A\,x = A$ - sklejanie jedynek

$A\,(\,/x + x\,) = A$

$(\,B + x\,)(\,B + /x\,) = B$ - sklejanie zer

$B + (\,x\,/x\,) = B$

* metoda wynaleziona przez Maurice'a Karnaugha w 1950 r.

Siatka *Karnaugha* wypełniana jest w oparciu o tabelę prawdy.

Minimalizacja funkcji boolowskich (6)

Metoda siatek *Karnaugha* (2)

Wygląd siatki zależny jest od liczby zmiennych:

$x_1 \backslash x_0$	0	1
0	00	01
1	10	$f\,(11)$

$x_2 \backslash x_1 x_0$	00	01	11	10
0	000	001	011	010
1	100	101	111	$f\,(111)$

$x_3\, x_2 \backslash x_1 x_0$	00	01	11	10
00				
01				
11				
10				

Metoda siatek *Karnaugha* (3)

Zmienne w opisie wierszy i kolumn uporządkowane są zgodnie z kodem *Graya*.
Ułatwia to stosowanie reguły sklejania wyrażeń sąsiednich.

$x_5\ x_4 x_3$ \ $x_2\ x_1 x_0$	000	001	011	010	110	111	101	100
000								
001								
011								
010								
110								
111								
101								
100								

000
001
011
010
110
111
101
100

Metoda siatek *Karnaugha* (4)

$x_2 \backslash x_1 x_0$	00	01	11	10
0	000	001	011	010
1	100	101	111	$f\,(111)$

Siatka *Karnaugha* wypełniana jest w oparciu o tabelę prawdy.

Każdy element siatki przechowuje wartość funkcji dla odpowiadającego mu wektora WE.

Dla funkcji n zmiennych każda kratka siatki ma dokładnie n sąsiadów.

Metoda siatek *Karnaugha* (5)

Siatka *Karnaugha* wypełniana jest w oparciu o tabelę prawdy.

$$f\,(a, b, c_i) = /a\; b\; /c_i + a\; /b\; /c_i + /a\; /b\; c_i + a\; b\; c_i = c_i \oplus a \oplus b$$

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

c_i \ $a\ b$	00	01	11	10
0	0	1	0	1
1	1	0	1	0

Metoda siatek *Karnaugha* (6)

Proces minimalizacji wykorzystujący regułę sklejania jest równoważny w siatce *Karnaugha* łączeniu sąsiednich jedynek w grupy.

$$f(x_2, x_1, x_0) = /x_2\, x_1\, x_0 + x_2\, /x_1\, x_0 + \underline{x_2\, x_1\, /x_0 + x_2\, x_1\, x_0} =$$

$$x_2\, x_1\, (/x_0 + x_0) = x_2\, x_1$$

$x_2 \backslash x_1 x_0$	00	01	11	10
0	0	0	1	0
1	0	1	**1**	**1**

Minimalizacja funkcji boolowskich (11)

Metoda siatek *Karnaugha* (7)

x_2 \ $x_1 x_0$	00	01	11	10
0	0	0	1	0
1	0	1	**1**	**1**

$x_2 x_1$ *

Dlaczego znika x_0 ?

Metoda siatek *Karnaugha* (8)

Sklejanie zer

- minimalizacja postaci kanonicznej iloczynu sum pełnych do postaci iloczynu sum (niekoniecznie pełnych)

- łączenie sąsiednich zer w grupy oraz tworzenie wektorów opisujących wartości zmiennych w grupie (jak dla jedynek)

- dualizm:
    - jeśli zmienna ma w grupie wartość 1 – występuje w sumie w negacji,
    - jeśli 0 – w pozycji (afirmacji),
    - jeśli * - ulega sklejeniu i nie występuje w zapisie sumy.

Minimalizacja funkcji boolowskich (13)

Metoda siatek *Karnaugha* (9)

$f\,(x_2, x_2, x_0) = \Pi(000, 001, 010, 100) =$

$= (\,x_2 + x_1 + x_0\,)(\,x_2 + x_1 + /x_0\,)(\,x_2 + /x_1 + x_0\,)(\,/x_2 + x_1 + x_0\,) =$

$= (\,x_1 + x_0\,)(\,x_2 + x_1\,)(\,x_2 + x_0\,)$

x_2 \ $x_1 x_0$	00	01	11	10
0	0	0	1	0
1	0	1	1	1

0*0 $\leftrightarrow (\,x_2 + x_0\,)$

00 $\leftrightarrow (\,x_1 + x_0\,)$ 00 $\leftrightarrow (\,x_2 + x_1\,)$

Metoda siatek *Karnaugha* (10)

Podsumowanie zasady sklejania

a) funkcja n zmiennych,
b) siatka *Karnaugha* ma 2^n elementów,
c) każdy element reprezentuje jeden iloczyn pełny (jedynkę) lub sumę pełną (zero) postaci kanonicznej,
d) każdy element ma dokładnie n elementów sąsiednich, z którymi może być sklejony,
e) sklejane grupy mogą liczyć 2^k elementów, $0 \leq k \leq n$,
f) grupa sklejonych 2^k jedynek (zer) jest opisana iloczynem (sumą) $n - k$ zmiennych,
g) grupa sklejonych 2^k elementów ma dokładnie $n - k$ potencjalnych grup sąsiednich, z którymi może być dalej sklejana.

Minimalizacja funkcji boolowskich (15)

Metoda siatek *Karnaugha* (11)

Główne kroki procesu minimalizacji

1) Wypełniamy siatkę *Karnaugha* zerami i jedynkami na podstawie tabeli prawdy, opisu słownego itp..
2) Decydujemy co będzie korzystniejsze: sklejanie *zer* czy sklejanie *jedynek* (ze względu na cel minimalizacji: np. mniejsza liczba literałów, rodzaju bramek).
3) Wybrane symbole (0 bądź 1) sklejamy w możliwie duże grupy wg określonych zasad.
4) Z uzyskanych grup wybieramy zestaw pokrywający wszystkie sklejane elementy złożone z *zer* bądź *jedynek*.
5) Zapisujemy wyrażenie algebraiczne odpowiadające wybranym grupom –
   – *sumę iloczynów* przy sklejaniu *jedynek* bądź *iloczyn sum* przy sklejaniu *zer*.

Metoda siatek *Karnaugha* (12)

Minimalizacja funkcji niezupełnych

Jeżeli dla pewnych wektorów wejściowych funkcja nie jest określona, wpisujemy w odpowiednie pola siatki *Karnaugha* symbole różne od 0 i 1, np.: $-$ lub ϕ, które interpretujemy je jako 0 bądź 1 tak, zależnie od korzystnego wpływu na efekt sklejania.

Wartości nieokreślone można sklejać (z zerami bądź jedynkami), w zależności od potrzeb.

Nie ma konieczności sklejenia wszystkich symboli nieokreślonych.

Minimalizacja funkcji boolowskich (17)

Metoda siatek *Karnaugha* (13)

$x_3 x_2 \backslash x_1 x_0$	00	01	11	10
00	0	1	–	0
01	0	–	–	1
11	0	–	–	1
10	0	1	–	1

$x_3 x_2 \backslash x_1 x_0$	00	01	11	10
00	0	0	0	0
01	0	1	1	1
11	–	–	–	–
10	1	1	–	–

$$f = /x_3\, x_2\, x_0 + /x_3\, x_2\, x_1 + x_3\, /x_2\, /x_1$$

$x_3 x_2 \backslash x_1 x_0$	00	01	11	10
00	0	0	0	0
01	0	1	1	1
11	–	–	–	–
10	1	1	–	–

$$f = x_2\, x_0 + x_2\, x_1 + x_3$$