

Mechatroniczne systemy wykonawcze, sensoryczne i sterujące

dr inż. Grzegorz Góra

D1-Lab 20

ggora@agh.edu.pl

<http://home.agh.edu.pl/~ggora/>

Katedra Robotyki i Mechatroniki

Wydział Inżynierii Mechanicznej i Robotyki

Akademia Górniczo-Hutnicza im. Stanisława Staszica w Krakowie

PLAN WYKŁADU

- 1. Kodowanie liczb (NKB, BCD, Graya, Z-M, U2, float)**
- 2. Bramki logiczne**
- 3. Algebra Boolea.**
- 4. Synteza układów kombinacyjnych**
 - bezpośrednia synteza równań;**
 - upraszczanie równań przy pomocy algebry Boolea ;**
 - wykorzystanie tablic Karnaugh;**
- 5. Zjawisko hazardu w układach cyfrowych.**

NKB – Naturalny Kod Binarny

Naturalny Kod Binarny (*ang. NBC – Natural Binary Code*) – jest najprostszym systemem kodowania liczb, pozwala na zapis liczb **całkowitych nieujemnych**.

- Jeśli n-bitową liczbę A reprezentuje ciąg bitów [$a_{n-1} a_{n-2} \dots a_2 a_1 a_0$], to jej wartość wynosi:

$$A = a_{n-1} \cdot 2^{n-1} + a_{n-2} \cdot 2^{n-2} + \dots + a_2 \cdot 2^2 + a_1 \cdot 2^1 + a_0 \cdot 2^0$$

- Zakres n-bitowej liczby wynosi: [0 ... 2^n-1]
 - 4-bity 0 ... 15
 - 8-bitów 0 ... 255
 - 16-bitów 0 ... 65 535
 - 32-bity 0 ... 4 294 967 295
 - 64-bity 0 ... 18 446 744 073 709 551 615
- NKB nie pozwala na zapis liczb ujemnych.
- W języku C/C++ NKB wykorzystywany jest do kodowania zmiennych typu: ***unsigned char, unsigned short, unsigned int, unsigned long***, itd.

BDC – Binary Coded Decimal

Kod BCD 8421 (ang. BCD – Binary Coded Decimal) – czyli zapis dziesiętny kodowany dwójkowo, pozwala na zapis liczb **całkowitych nieujemnych**.

- Liczba zostaje podzielona na cyfry, a następnie każda z cyfr jest kodowana osobno na 4-bitach.
- Wykorzystywanych jest 10 z 16 kombinacji bitów, pozostałe kombinacje nie są wykorzystywane.
- Liczby w kodzie BCD nie są naturalnymi liczbami dwójkowymi, operacje matematyczne bezpośrednio na kodzie BCD są możliwe, natomiast wymagają stosowania korekt. Klasyczna jednostka arytmetyczno-logiczna (ALU) nie da poprawnych wyników.
- Zastosowanie:
 - w sterownikach wyświetlaczy cyfrowych (np. kalkulatory, mierniki cyfrowe);
 - do przechowywania i obliczeń bezpośrednio na liczbach dziesiętnych, bez konwersji do kodu dwójkowego.



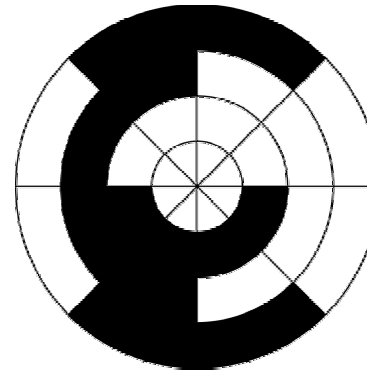
Cyfra dziesiętna	BCD
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
-	1010
-	1011
-	1100
-	1101
-	1110
-	1111

$$5827_{\text{DEC}} = \begin{matrix} 0101 & 1000 & 0010 & 0111 \\ \downarrow & \downarrow & \downarrow & \downarrow \\ 5 & 8 & 2 & 7 \end{matrix} \text{ BCD}$$

Kod Graya

Kod Graya (*ang. Gray Code*) – binarny kod niepozycyjny, pozwala na zapis liczb **całkowitych nieujemnych**. Jego podstawową cechą jest to, że dwie kolejne liczby zapisane w tym kodzie różnią się od siebie zawsze tylko jednym bitem.

- Kod cykliczny – pierwszy i ostatni wyraz tego kodu także spełniają zasadę różnicy jednego bitu.
- N-bitowy kod Graya tworzy się rekurencyjnie poprzez symetryczne odbicie kodu dla (n-1) bitów oraz uzupełnienie najstarszego bitu zerami (pierwsza połowa) oraz jedynekami (druga połowa).
- Zastosowanie (w celu unikania stanów przejściowych w układach elektroniki cyfrowej):
 - przetworniki ADC;
 - enkodery (przykład 3-bitowej tarczy enkodera na rys. obok);
 - **minimalizacja funkcji logicznych przy pomocy tablica Karnaugh.**



	3-bity	
	2-bity	
	1-bit	1-bit
0	0	0
0	0	1
0	1	1
0	1	0
1	1	0
1	1	1
1	0	1
1	0	0

Kod Znak-Moduł (Z-M)

Kod Znak-Moduł (ang. *S-M – Sign-Magnitude*) – jest najprostszym rodzajem kodowania pozwalającym na reprezentację **liczb całkowitych ze znakiem, czyli dodatnich i ujemnych**.

- W kodzie Znak-Moduł:
 - jeden bit (zazwyczaj najstarszy) jest bitem znaku:
0 – liczba dodatnia, 1 – liczba ujemna;
 - wszystkie pozostałe bity reprezentują wartość bezwzględną kodowanej liczby (w naturalnym kodzie binarnym).
- Jeśli n-bitową liczbę A reprezentuje ciąg bitów $[a_{n-1} a_{n-2} \dots a_2 a_1 a_0]$, to jej wartość wynosi:

$$A = (-1)^{a_{n-1}} \cdot (a_{n-2} \cdot 2^{n-2} + \dots + a_2 \cdot 2^2 + a_1 \cdot 2^1 + a_0 \cdot 2^0).$$

- Zakres n-bitowej liczby wynosi: $[-2^{n-1}+1 \dots 2^{n-1}-1]$, np. dla liczby 8-bitowej $\langle -127, 127 \rangle$.
- Znalezienie liczby o przeciwnej wartości polega na negacji bitu znaku (najstarszego bitu).
- Podwójne kodowanie zera np. dla liczby 8-bitowej: 00000000 oraz 10000000 oba ciągi bitów oznaczają **wartość 0 (czyli +0 oraz -0)**.
- Kod Z-M jest koncepcyjnie prosty, lecz stwarza poważne problemy przy wykonywaniu operacji arytmetycznych. Bit znaku posiada zupełnie inne znaczenie od pozostałych bitów i nie uczestniczy bezpośrednio w operacjach arytmetycznych. **W konsekwencji operacje wykonywane przez standardową jednostkę arytmetyczno-logiczną (ALU) na liczbach zapisanych w tym kodzie nie dają poprawnych wyników!!!**

Kod U2 (Kod uzupełnień do 2)

Kod U2 (ang. Two's complement) – kodowanie pozwalające na reprezentację liczb całkowitych ze znakiem, czyli dodatnich i ujemnych.

- Najstarszy bit ma wagę o wartości ujemnej, pozostałe bity są dodatnie.
- Jeśli n-bitową liczbę A reprezentuje ciąg bitów $[a_{n-1} a_{n-2} \dots a_2 a_1 a_0]$, to jej wartość wynosi:

$$A = (-1) \cdot a_{n-1} \cdot 2^{n-1} + a_{n-2} \cdot 2^{n-2} + \dots + a_2 \cdot 2^2 + a_1 \cdot 2^1 + a_0 \cdot 2^0$$

- Zakres n-bitowej liczby wynosi: $[-2^{n-1} \dots 2^{n-1}-1]$, np. dla liczby 8-bitowej $\langle -128, 127 \rangle$.
- Dla wartości dodatnich liczby zapisane w kodzie U2 mają taką samą postać jak w NKB.
- Znalezienie liczby o przeciwnej wartości:
 - negacja wszystkich bitów;
 - dodanie 1.
- Operacje arytmetyczne wykonywane przez standardową jednostkę arytmetyczno-logiczną (ALU) na liczbach zapisanych w kodzie U2 dają poprawne wyniki.
- W języku C/C++ U2 wykorzystywany jest do kodowania zmiennych typu: *char, short, int, long*, itd.

0010 0011	= 35 _(U2)
~	
1101 1100	
+1	
1101 1101	= -35 _(U2)
~	
0010 0010	
+1	
0010 0011	= 35 _(U2)

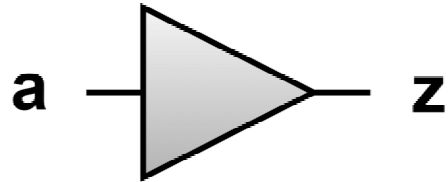
Liczba zmiennoprzecinkowa pojedynczej precyzji (float)

Liczba zmiennoprzecinkowa pojedynczej precyzji (float) – kodowanie pozwalające na reprezentację liczb zmiennoprzecinkowych.

Znak	Wykładnik	Mantysa	Rodzaj liczby
Dowolny	Różny od '00000000' Oraz '11111111'	Dowolna	Liczba znormalizowana
Dowolny	'00000000'	Różna od '000000000000000000000000'	Liczba nieznormalizowana
'0'	'00000000'	'000000000000000000000000'	+0
'1'	'00000000'	'000000000000000000000000'	-0
'0'	'11111111'	'000000000000000000000000'	+∞
'1'	'11111111'	'000000000000000000000000'	-∞
Dowolny	'11111111'	Różna od '000000000000000000000000'	NaN

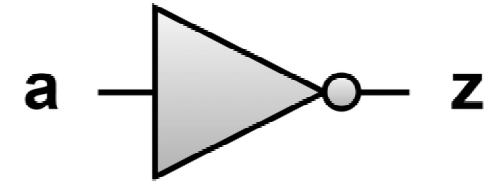
Tab. Kodowanie liczb zmiennoprzecinkowych pojedynczej precyzji

Bramki: BUF, NOT



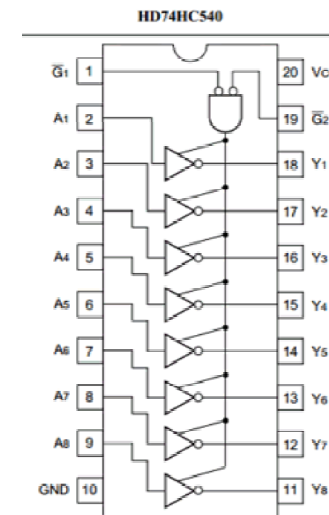
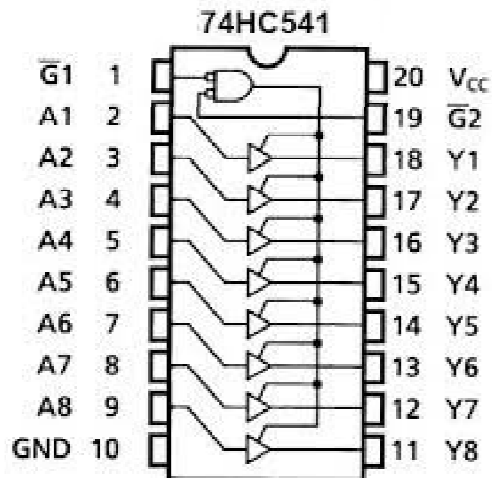
$$z = a$$

a	z
0	0
1	1

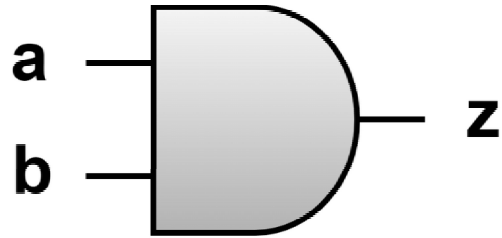


$$z = \bar{a}$$

a	z
0	1
1	0

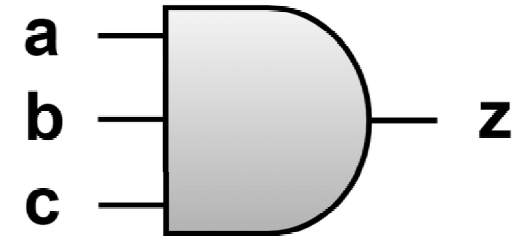


Bramki: AND (iloczyn logiczny, koniunkcja)



$$z = ab$$

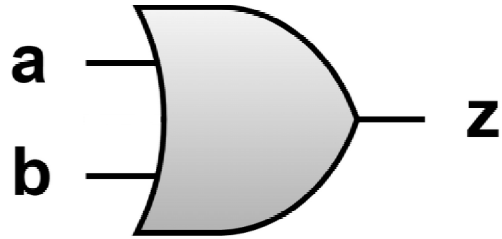
a	b	z
0	0	0
0	1	0
1	0	0
1	1	1



$$z = abc$$

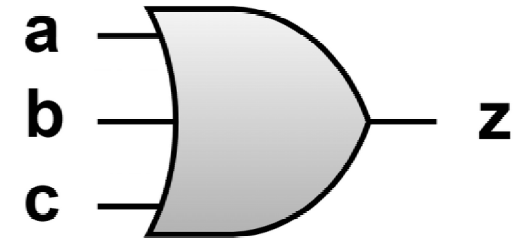
a	b	c	z
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

Bramki: OR (suma logiczna, alternatywa)



$$z = a + b$$

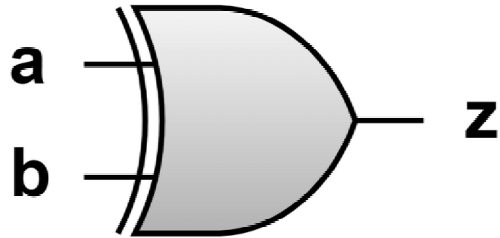
a	b	z
0	0	0
0	1	1
1	0	1
1	1	1



$$z = a + b + c$$

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

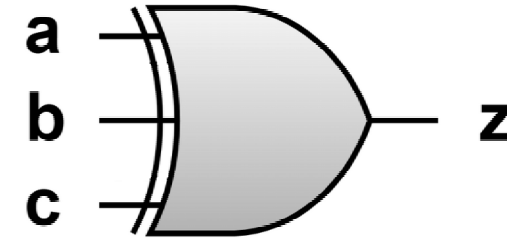
Bramki: XOR (alternatywa wykluczająca)



$$z = a \oplus b$$

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

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

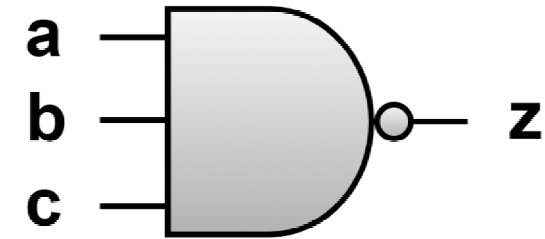
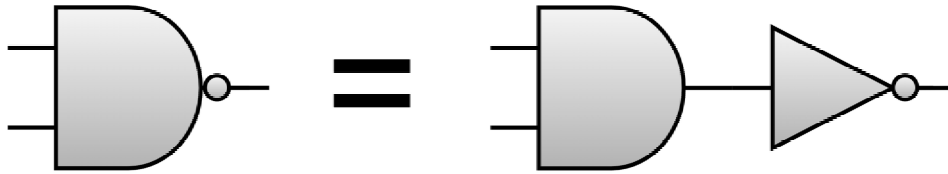


$$z = a \oplus b \oplus c$$

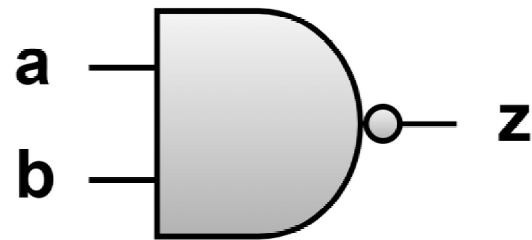
a	b	c	z
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

XOR = suma modulo 2

Bramki: NAND



$$z = \overline{abc}$$

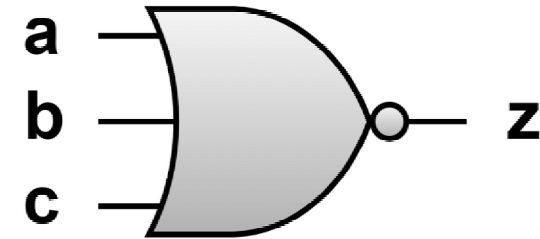
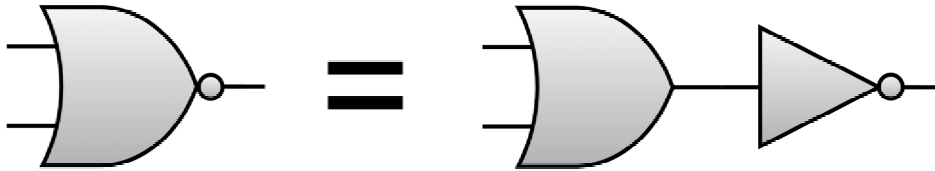


$$z = \overline{ab}$$

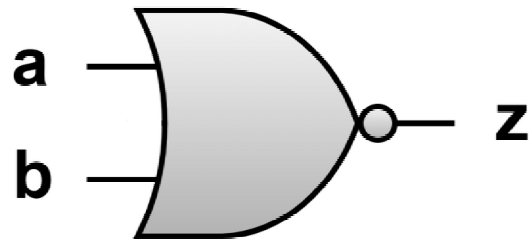
a	b	z
0	0	1
0	1	1
1	0	1
1	1	0

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

Bramki: NOR



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



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

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

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

Algebra Boole'a

$$\overline{\overline{A}} = A$$

$$A + 0 = A$$

$$A + 1 = 1$$

$$A + A = A$$

$$A + \overline{A} = 1$$

$$A \cdot 0 = 0$$

$$A \cdot 1 = A$$

$$A \cdot A = A$$

$$A \cdot \overline{A} = 0$$

$$A + B = B + A$$

$$A + (A \cdot B) = A$$

$$A + (B + C) = (A + B) + C$$

$$A \cdot (B + C) = (A \cdot B) + (A \cdot C)$$

$$\overline{A + B} = \overline{A} \cdot \overline{B}$$

$$A + \overline{A} \cdot B = A + B$$

$$A \cdot B + \overline{A} \cdot C + B \cdot C = A \cdot B + \overline{A} \cdot C \quad (A + B) \cdot (\overline{A} + C) \cdot (B + C) = (A + B) \cdot (\overline{A} + C)$$

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

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

$$A \cdot B = B \cdot A$$

$$A \cdot (A + B) = A$$

$$A \cdot (B \cdot C) = (A \cdot B) \cdot C$$

$$A + (B \cdot C) = (A + B) \cdot (A + C)$$

$$\overline{A \cdot B} = \overline{A} + \overline{B}$$

$$A \cdot (\overline{A} + B) = A \cdot B$$

Prawo przemienności

Prawo pochłaniania

Prawo łączenia

Prawo rozdzielności

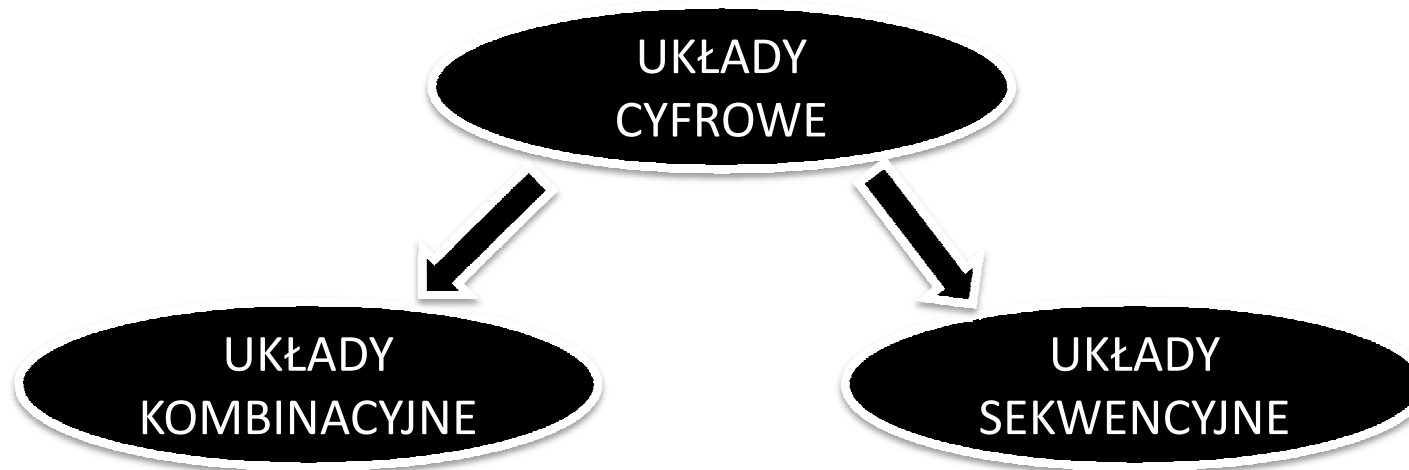
Prawa DeMorgana

Prawo eliminacji

Prawo zgody

Prawo wymiany

**Układy cyfrowe:
układy kombinacyjne i sekwencyjne**



UKŁAD KOMBINACYJNY – rodzaj układu cyfrowego, w którym *stan wyjść tego układu zależy jedynie od stanu jego wejść*. Układami kombinacyjnymi są np. sumator, substraktor, multiplekser, komparator, transkodery, itd.

UKŁAD SEKWENCYJNY – rodzaj układu cyfrowego, w którym *stan wyjść tego układu zależy od stanu wejść oraz poprzedniego stanu tego układu*. Układami sekwencyjnymi są np. liczniki, automaty.

ZADANIE: Znaleźć równanie (możliwie proste) oraz schemat logiczny (z bramek) pozwalający na implementację funkcji logicznej dla uproszczonego sterownika drzwi windy.

Opis działania uproszczonego sterownika drzwi windy

Drzwi zostaną zamknięte gdy:

- obwód fotokomórki jest zamknięty (nikt nie stoi w drzwiach), oraz
- upłynął minimalny czas otwarcia drzwi (sygnał z zegara – osobnego urządzenia – jest wyłączony), oraz
- przycisk przywołania windy na piętrze jest naciśnięty
lub
- przycisk wyboru piętra w windzie jest naciśnięty.

**Synteza układów kombinacyjnych:
(1) bezpośrednia synteza równań**

ZADANIE: Znaleźć równanie (możliwie proste) oraz schemat logiczny (z bramek) pozwalający na implementację funkcji logicznej dla uproszczonego sterownika drzwi windy.

Drzwi "D" zostaną zamknięte gdy
(zegar "C" **NIE** jest włączony) **I**
(obwód fotokomórki "P" jest zamknięty) **I**
[(przycisk przywołania "B" jest naciśnięty) **LUB**
(przycisk wyboru piętra "M" jest naciśnięty)]

$$D = \bar{C} \cdot P \cdot (B + M)$$

**Synteza układów kombinacyjnych:
(1) bezpośrednia synteza równań**

ZADANIE: Znaleźć równanie (możliwie proste) oraz schemat logiczny (z bramek) pozwalający na implementację funkcji logicznej dla uproszczonego sterownika drzwi windy.

Drzwi "D" zostaną zamknięte gdy
(zegar "C" **NIE** jest włączony) **I**
(obwód fotokomórki "P" jest zamknięty) **I**
[(przycisk przywołania "B" jest naciśnięty) **LUB**
(przycisk wyboru piętra "M" jest naciśnięty)]

$$D = \bar{C} \cdot P \cdot (B + M)$$

Bezpośrednia synteza równań możliwa jest tylko dla prostych przypadków, najczęściej o małej liczbie zmiennych (sygnałów), gdzie nie występują między nimi złożone zależności.

Synteza układów kombinacyjnych na podstawie tablicy prawdy

Idx.	Wejścia			Wyjście	Mintermy:	Maxtermy:
	i	a	b	c		
0	0	0	0	1	$m_0 = \bar{a}\bar{b}\bar{c}$	$M_0 = a + b + c$
1	0	0	1	0	$m_1 = \bar{a}\bar{b}c$	$M_1 = a + b + \bar{c}$
2	0	1	0	1	$m_2 = \bar{a}b\bar{c}$	$M_2 = a + \bar{b} + c$
3	0	1	1	x	$m_3 = \bar{a}bc$	$M_3 = a + \bar{b} + \bar{c}$
4	1	0	0	1	$m_4 = a\bar{b}\bar{c}$	$M_4 = \bar{a} + b + c$
5	1	0	1	0	$m_5 = a\bar{b}c$	$M_5 = \bar{a} + b + \bar{c}$
6	1	1	0	1	$m_6 = ab\bar{c}$	$M_6 = \bar{a} + \bar{b} + c$
7	1	1	1	x	$m_7 = abc$	$M_7 = \bar{a} + \bar{b} + \bar{c}$

Postać kanoniczna funkcji w formie SOP
(Sum Of Products - Suma Iloczynów):

$$SOP = \sum_{i|y(i)=1} m_i$$

Postać kanoniczna funkcji w formie POS
(Product of Sums – Iloczyn Sum):

$$POS = \prod_{i|y(i)=0} M_i$$

Stan dowolny (oznaczany jako x lub -) oznacza, że dana kombinacja bitów nie pojawi się na wejściu układu opisanego przez tablicę. Taka kombinacja bitów nie należy do dziedziny funkcji.

Stan dowolny może być użyty jako 0 lub 1, aby powiększyć istniejącą grupę zer lub jedynek.

Synteza układów kombinacyjnych:

(2) tablica prawdy + upraszczanie równań przy pomocy algebry Boolea

C (zegar)	P (fotokomórka)	B (przycisk przywołania)	M (przycisk w windzie)	D (zamknięcie drzwi)
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

Uproszczony sterownik drzwi windy: Drzwi zostaną zamknięte gdy obwód fotokomórki jest zamknięty (nikt nie stoi w drzwiach), oraz upłynął minimalny czas otwarcia drzwi (sygnał z zegara – osobnego urządzenia – jest wyłączony), oraz przycisk przywołania windy na piętrze jest naciśnięty lub przycisk wyboru piętra w windzie jest naciśnięty.

Postać kanoniczna – suma iloczynów (SOP – Sum Of Products):

$$\left. \begin{aligned}
 D &= \bar{C} \cdot P \cdot \bar{B} \cdot M \\
 &+ \bar{C} \cdot P \cdot B \cdot \bar{M} \\
 &+ \bar{C} \cdot P \cdot B \cdot M
 \end{aligned} \right\}$$

$$\begin{aligned}
 D &= (\bar{C} \cdot P) \cdot (\bar{B} \cdot M + B \cdot \bar{M} + B \cdot M) \\
 &= (\bar{C} \cdot P) \cdot (\bar{B} \cdot M + B(\bar{M} + M)) \\
 &= (\bar{C} \cdot P) \cdot (\bar{B} \cdot M + B) \\
 &= \bar{C} \cdot P \cdot (B + M)
 \end{aligned}$$

Synteza układów kombinacyjnych:
(2) tablica prawdy + upraszczanie równań przy pomocy algebry Boolea

C (zegar)	P (fotokomórka)	B (przycisk przywołania)	M (przycisk w windzie)	D (zamknięcie drzwi)
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

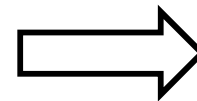
Uproszczony sterownik drzwi windy: Drzwi zostaną zamknięte gdy obwód fotokomórki jest zamknięty (nikt nie stoi w drzwiach), oraz upłynął minimalny czas otwarcia drzwi (sygnał z zegara – osobnego urządzenia – jest wyłączony), oraz przycisk przywołania windy na piętrze jest naciśnięty lub przycisk wyboru piętra w windzie jest naciśnięty.

$$\begin{aligned}
 D &= \bar{C} \cdot P \cdot \bar{B} \cdot M + \bar{C} \cdot P \cdot B \cdot \bar{M} + \bar{C} \cdot P \cdot B \cdot M \\
 &= (\bar{C} \cdot P) \cdot (\bar{B} \cdot M + B \cdot \bar{M} + B \cdot M) \\
 &= (\bar{C} \cdot P) \cdot (\bar{B} \cdot M + B(\bar{M} + M)) \\
 &= (\bar{C} \cdot P) \cdot (\bar{B} \cdot M + B) \\
 &= \bar{C} \cdot P \cdot (B + M)
 \end{aligned}$$

Upraszczenie równań przy pomocy algebry Boolea jest procesem złożonym i pracochłonnym. Poziom skomplikowania szybko rośnie wraz ze wzrostem liczby zmiennych.

Synteza układów kombinacyjnych: (3) wykorzystanie tablic Karnaugh

C (zegar)	P (fotokomórka)	B (przycisk przywołania)	M (przycisk w windzie)	D (zamknięcie drzwi)
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0



Krok 1:

Przeniesienie informacji z tablicy prawdy do tablicy Karnaugh.

BM CP	00	01	11	10
00	0	0	0	0
01	0	1	1	1
11	0	0	0	0
10	0	0	0	0

Synteza układów kombinacyjnych: (3) wykorzystanie tablic Karnaugh

Krok 2:

Pogrupowanie 1 (implikanty) lub 0 (implicenty).

BM CP	00	01	11	10
00	0	0	0	0
01	0	1	1	1
11	0	0	0	0
10	0	0	0	0

- rozmiar grupy musi być potęgą 2 (np. 1x1, 1x2, 2x1, 1x4, 4x1, 2x4, 4x2, itd.);
- należy utworzyć minimalną konieczną ilość grup zawierającą wszystkie 1 (lub 0);
- grupy mogą na siebie „nachodzić” (tzn. jedna komórka może być elementem dwóch lub więcej grup);
- grupy powinny być jak największe (duża ilość komórek w grupie oznacza mniejszą liczbę zmiennych do jej opisanie – w konsekwencji uproszczenie równania).

Synteza układów kombinacyjnych: (3) wykorzystanie tablic Karnaugh

BM CP	00	01	11	10
00	0	0	0	0
01	0	1	1	1
11	0	0	0	0
10	0	0	0	0

Krok 3:

Napisanie równań w formie:

sum iloczynów SOP (w przypadku grupowania 1)
lub

iloczynów sum POS (w przypadku grupowania 0).

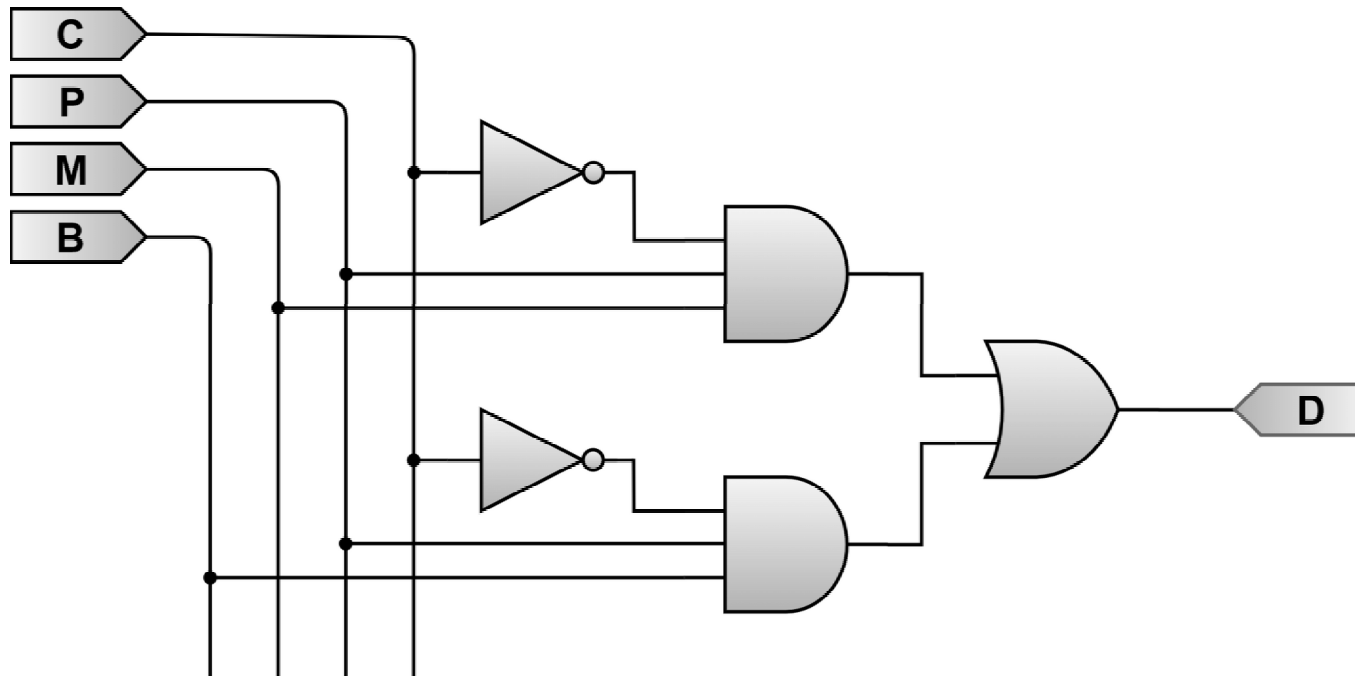
$$D = \underline{\overline{C}} \cdot P \cdot M + \overline{\underline{C}} \cdot P \cdot B$$

**Synteza układów kombinacyjnych:
(3) wykorzystanie tablic Karnaugh**

Krok 4:

Narysowanie schematu logicznego na podstawie równania.

$$D = \bar{C} \cdot P \cdot M + \bar{C} \cdot P \cdot B$$



Synteza układów kombinacyjnych: wykorzystanie tablic Karnaugh - Podsumowanie

Tablice Karnaugh są graficzną metodą upraszczania równań algebry Boolea. Pozwalają na dostanie od razu uproszczonej formy równania.

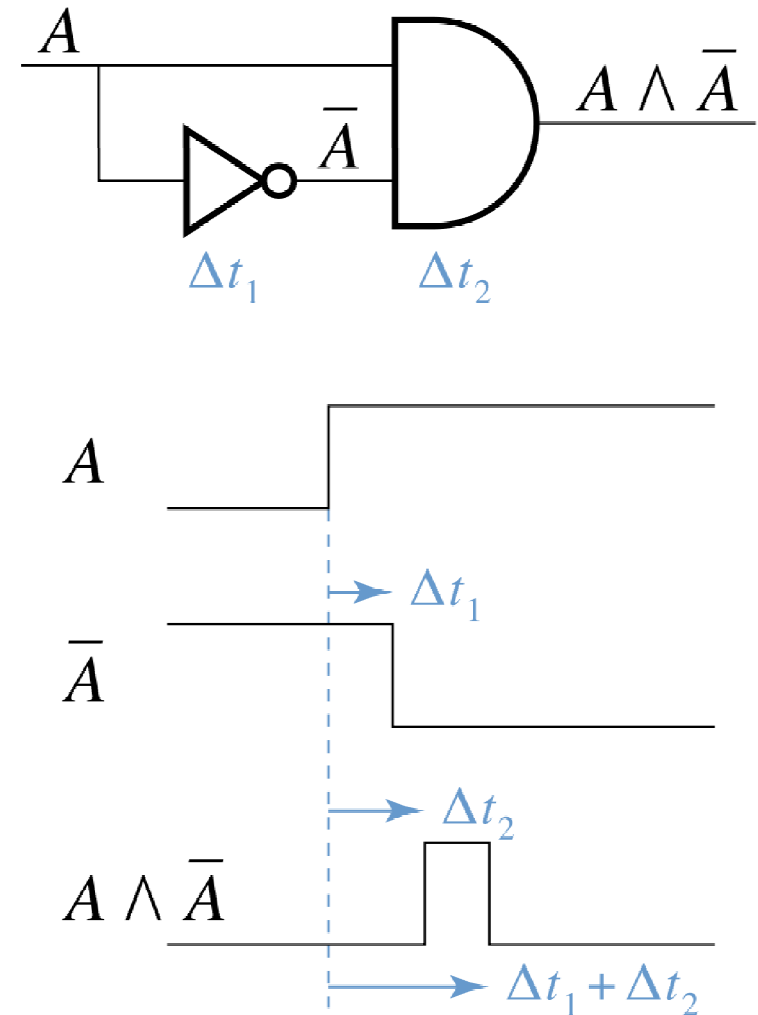
1. Przeniesienie informacji z tablicy prawdy do tablicy Karnaugh.
2. Pogrupowanie 1 lub 0:
 - rozmiar grupy musi być potęgą 2 (np. 1x1, 1x2, 2x1, 1x4, 4x1, 2x4, 4x2, itd.);
 - należy utworzyć minimalną konieczną ilość grup zawierającą wszystkie 1 (lub 0);
 - grupy mogą na siebie „nachodzić” (tzn. jedna komórka może być elementem dwóch lub więcej grup);
 - grupy powinny być jak największe (duża ilość komórek w grupie oznacza mniejszą liczbę zmiennych do jej opisanie – w konsekwencji uproszczenie równania).
3. Napisanie równań w formie sum iloczynów (w przypadku grupowania 1) lub iloczynów sum (w przypadku grupowania 0).
4. Narysowanie schematu logicznego na podstawie równań algebraicznych.

Zjawisko hazardu w układach cyfrowych

Rzeczywiste bramki wprowadzają opóźnienie w odpowiedzi sygnałów wyjściowych na zmiany sygnałów wejściowych – czas ten nazywa się **czasem propagacji sygnału przez bramkę**.

Zazwyczaj czas propagacji jest różny dla zmiany wyjścia ze stanu niskiego na wysoki (t_{LH}) i wysokiego na niski (t_{HL}).

Producenci bramek podają maksymalny gwarantowany czas propagacji dla swoich bramek. Jest to czas propagacji bramki w najgorszym możliwym przypadku.



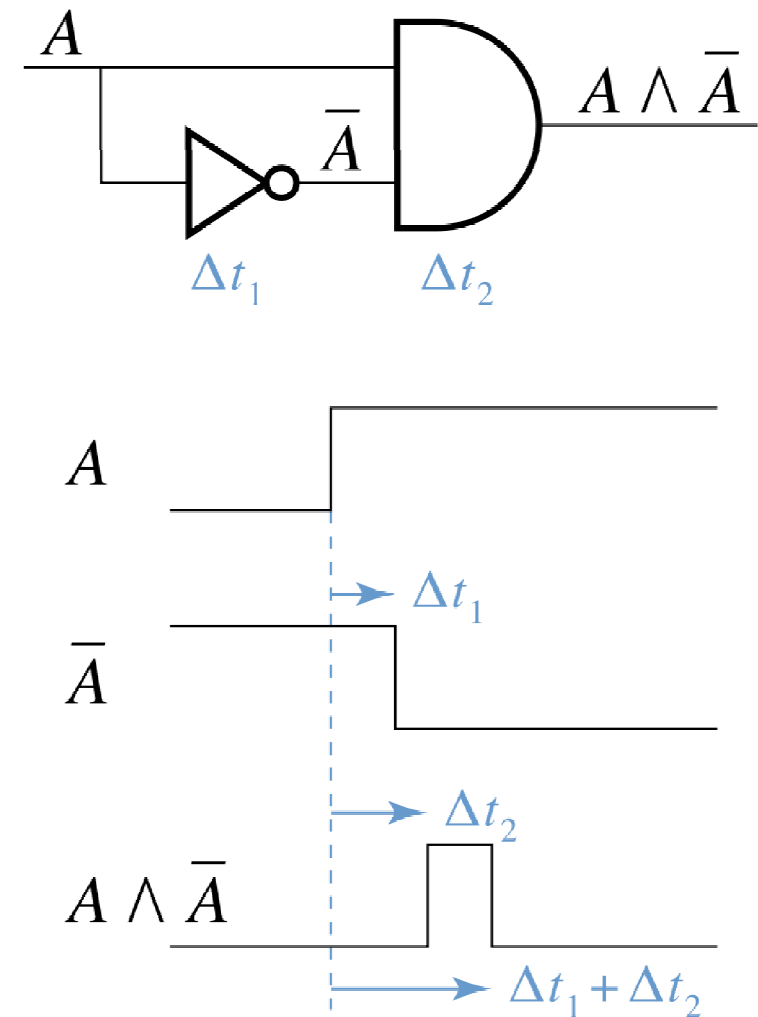
Zjawisko hazardu w układach cyfrowych

Hazard to krótkotrwała ‘szpilka’ (występowanie 0 w stanie 1 lub występowanie 1 w stanie 0), która powstała na skutek nierównych czasów propagacji dla różnych ścieżek w układzie.

Wyróżniamy hazardy:

- statyczne; kiedy wyjście które ma nie zmienić stanu krótkotrwanie zmienia stan na przeciwny;
- dynamiczne - kiedy występują wielokrotne zmiany stanów na wyjściach które zmieniają stan.

Hazardy mogą powodować nieprawidłowe działanie układów, jeżeli wyjścia na których się pojawiają są interpretowane asynchronicznie.



LITERATURA

- [1] Barry Wilkinson: *Układy cyfrowe*; WKŁ
- [2] Halina Kamionka-Mikuła, Henryk Małysiak, Bolesław Pochopień: *Układy cyfrowe - teoria i przykłady*; WPKJS;
- [3] Wojciech Głocki: *Układy cyfrowe*; WSiP;
- [4] Andrzej Skorupski: *Podstawy techniki cyfrowej*; WKŁ;
- [5] prof. dr hab. inż. Joanna Józefowska: *Kodowanie informacji - Reprezentacja liczb*; Poznań; rok akademicki 2010/2011;
- [6] dr hab. inż. Mikołaj Morzy: *Wykład - kodowanie liczb*;
- [7] Politechnika Łódzka: *Kodowanie liczb całkowitych w systemach komputerowych*;
http://neo.dmcs.p.lodz.pl/ak/kodowanie_liczb_calkowitych.pdf