Mechatroniczne systemy wykonawcze, sensoryczne i sterujące

Układy kombinacyjne

dr inż. Grzegorz Góra

D3-4.02

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. *Konwersja liczb DEC-BIN-HEX (Przypomnienie)*
2. *Kodowanie liczb (NKB, BCD, Graya, Z-M, U2, float, double)*
3. *Bramki logiczne*
4. *Algebra Boolea.*
5. *Synteza układów kombinacyjnych*

   - bezpośrednia synteza równań;

   - upraszczanie równań przy pomocy algebry Boolea ;

   - wykorzystanie tablic Karnaugh;
6. *Zjawisko hazardu w układach cyfrowych.*

Konwersja DEC ⇔ BIN

DEC → BIN

Konwersja liczby z systemu **dziesiętnego (DEC)** na **binarny (BIN)** polega na kolejnych dzieleniach liczby przez 2 i zapisywaniu reszt z tych dzieleń. Następnie wynik w systemie binarnym tworzy się z tych reszt czytanych od końca (od ostatniej reszty do pierwszej).

Krok	Operacja	Całość z dzielenia	Reszta z dzielenia
1.	173 : 2	86	1
2.	86 : 2	43	0
3.	43 : 2	21	1
4.	21 : 2	10	1
5.	10 : 2	5	0
6.	5 : 2	2	1
7.	2 : 2	1	0
8.	1 : 2	0	1

$$10101101_{BIN} = 173_{DEC}$$

BIN → DEC

Konwersja liczby z systemu **binarnego (BIN)** na **dziesiętny (DEC)** polega na zsumowaniu wartości kolejnych bitów pomnożonych przez odpowiednie potęgi liczby 2.

$$X_{DEC} = 1 \cdot 2^7 + 0 \cdot 2^6 + 1 \cdot 2^5 + 0 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0$$

$$173_{DEC} = 10101101_{BIN}$$

Konwersja DEC ⇔ HEX

DEC → HEX

Konwersja liczby z systemu **dziesiętnego (DEC)** na **szesnastkowy (HEX)** polega na kolejnych dzieleniach liczby przez 16 i zapisywaniu reszt z tych dzieleń. Następnie wynik w systemie szesnastkowym tworzy się z tych reszt czytanych od końca (od ostatniej reszty do pierwszej).

Step	Operation	Quotient	Remainder
1.	12345678 : 16	771604	14 (E)
2.	771604 : 16	48225	4
3.	48225 : 16	3014	1
4.	3014 : 16	188	6
5.	188 : 16	11	12 (C)
6.	11 : 16	0	11 (B)

$$BC614E_{HEX} = 12345678_{DEC}$$

HEX → DEC

Konwersja liczby z systemu **szesnastkowego (HEX)** na **dziesiętny (DEC)** polega na zsumowaniu wartości kolejnych znaków pomnożonych przez odpowiednie potęgi liczby 16.

$$X_{DEC} = B \cdot 16^5 + C \cdot 16^4 + 6 \cdot 16^3 + 1 \cdot 16^2 + 4 \cdot 16^1 + E \cdot 16^0$$

$$12345678_{DEC} = BC614E_{HEX}$$

Konwersja BIN ⇔ HEX

BIN → HEX

Konwersja liczby z systemu binarnego (**BIN**) na szesnastkowy (**HEX**) polega na podziale liczby zapisanej w systemie binarnym na grupy po cztery bity, zaczynając od prawej strony. Każda taka grupa jest następnie zamieniana na odpowiadającą jej cyfrę w systemie szesnastkowym.

1111000011110111000

↓

0011 | 1100 | 0011 | 1011 |1000

↓

3 | C | 3 | B |8

↓

3C3B8

$3C3B8_{HEX} = 1111000011110111000_{BIN}$

BIN	HEX
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	A
1011	B
1100	C
1101	D
1110	E
1111	F

HEX → BIN

Konwersja liczby z systemu szesnastkowego (**HEX**) na binarny (**BIN**) polega na zastąpieniu każdej cyfry szesnastkowej jej odpowiednikiem w postaci czterech bitów zapisanych w systemie binarnym. Następnie te grupy bitów łączy się w jeden ciąg, który tworzy liczbę w systemie binarnym.

3C3B8

↓

3 | C | 3 | B |8

↓

0011 | 1100 | 0011 | 1011 |1000

↓

1111000011110111000

$1111000011110111000_{BIN} = 3C3B8_{HEX}$

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} … a_2 a_1 a_0], to jej wartość wynosi:

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

- 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

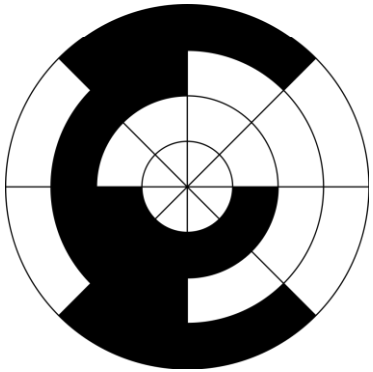
$$5827_{DEC} = \underset{5}{0101}\ \underset{8}{1000}\ \underset{2}{0010}\ \underset{7}{0111}\ _{BCD}$$

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.

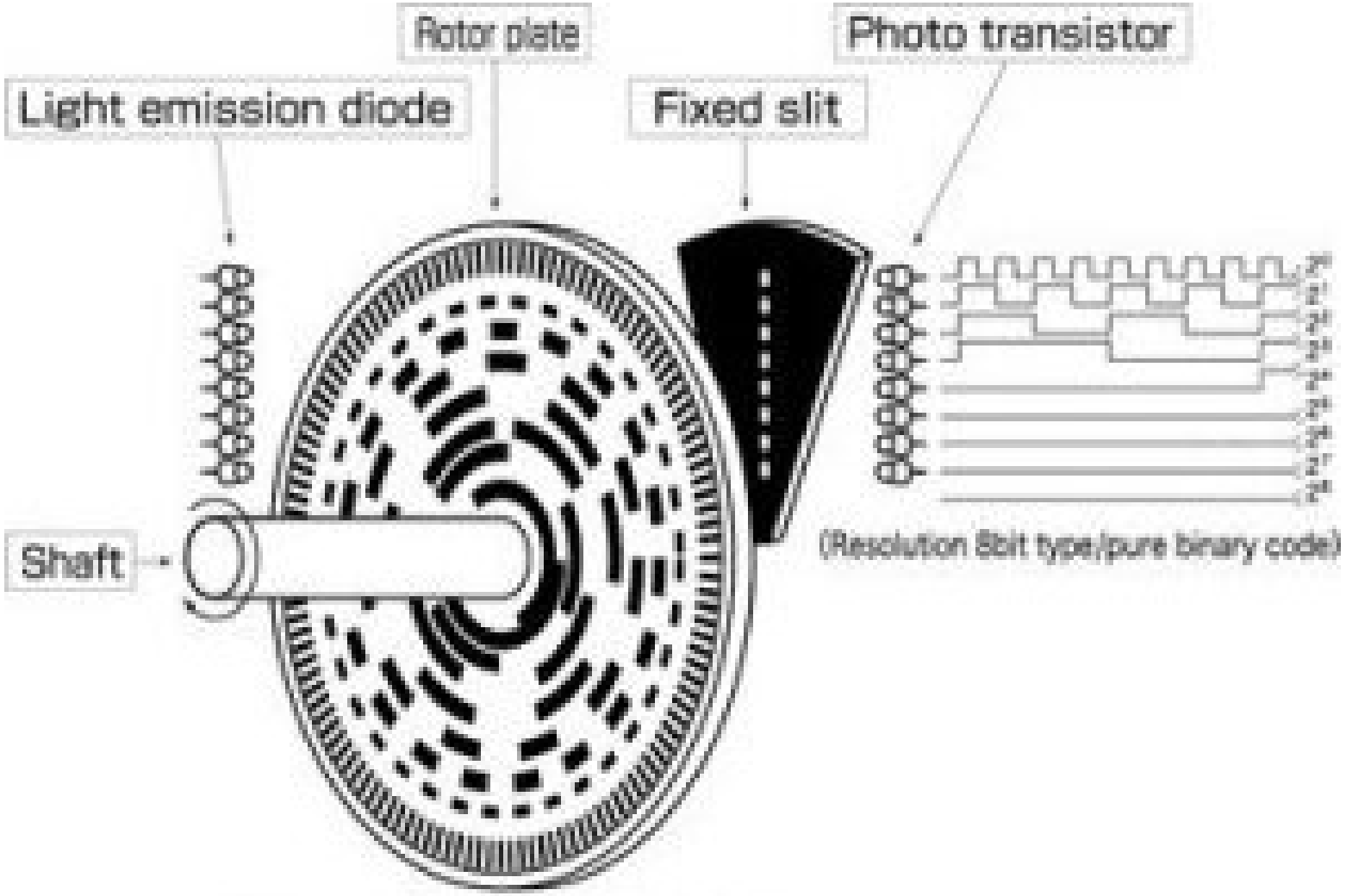
- Kod cykliczny – pierwszy i ostatni wyraz tego kodu także spełniają zasadę różnicy jednego bitu.

- N-bitowy kod Graya tworzy się rekurencyjnie poprzez symetryczne odbicie kodu dla (n-1) bitów oraz uzupełnienie najstarszego bitu zerami (pierwsza połowa) oraz jedynkami (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
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 Graya
(Zastosowanie kodu Graya w konstrukcji tarczy enkodera absolutnego)

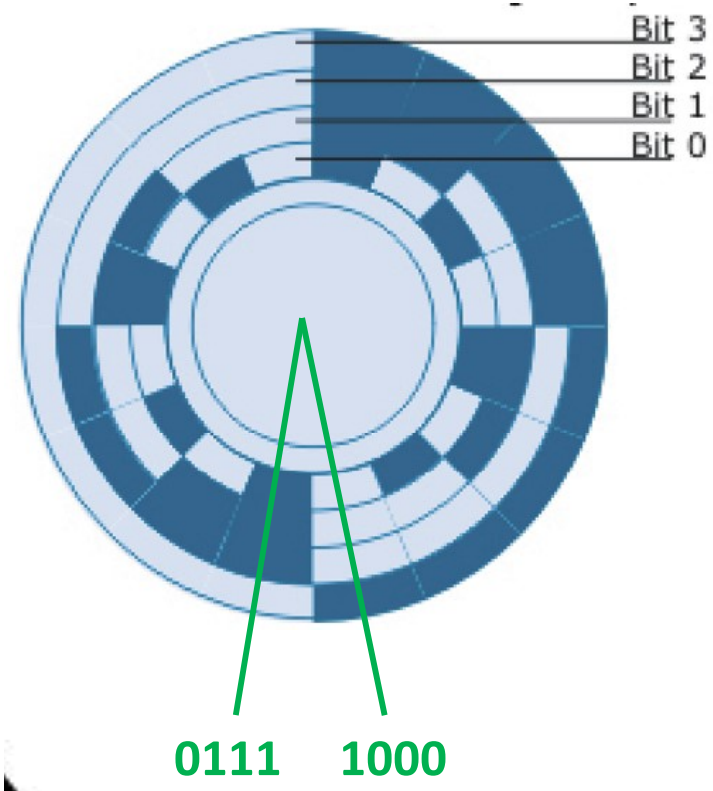


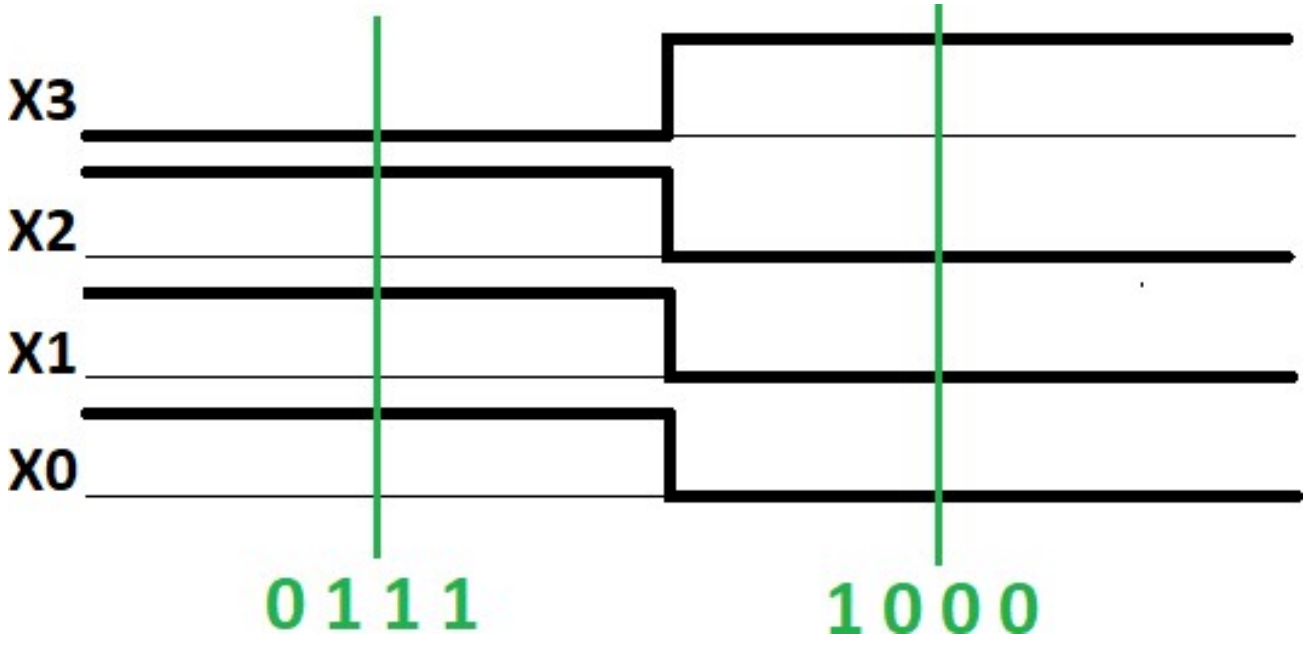
Absolute Encoder Simplified Structure

Kod Graya
(Zastosowanie kodu Graya w konstrukcji tarczy enkodera absolutnego)

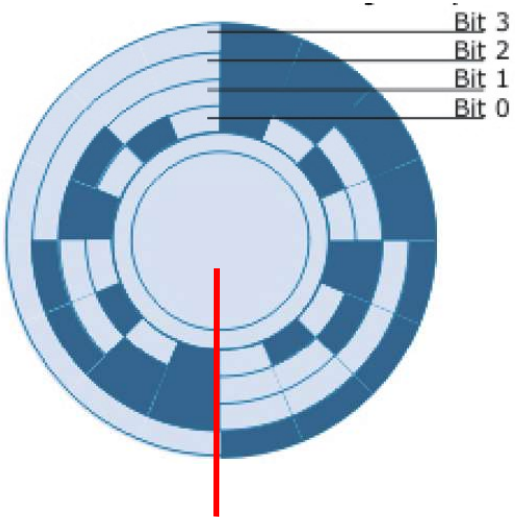
Dlaczego warto wykorzystać kod Graya w budowie tarczy enkodera?
Co się dzieje w enkoderze z tarczą w kodzie binarnym,
gdy następuje zmiana wartości kilku bitów jednocześnie?
Przykład 4-bitowego enkodera:
Zmiana 0111 <=> 1000

Tarcza z kodem binarnym



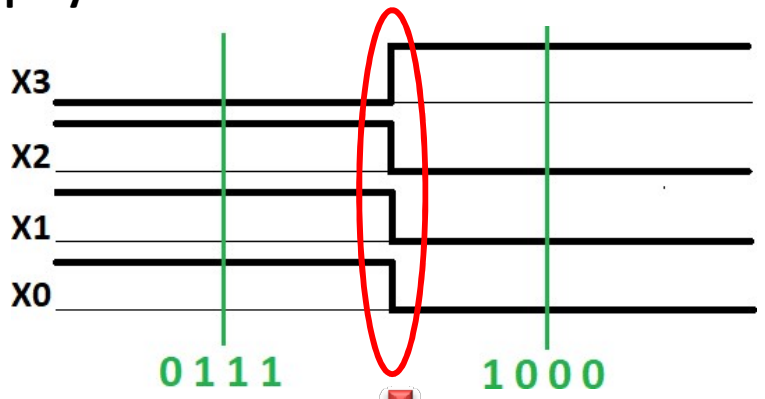


Kod Graya
(Zastosowanie kodu Graya w konstrukcji tarczy enkodera absolutnego)



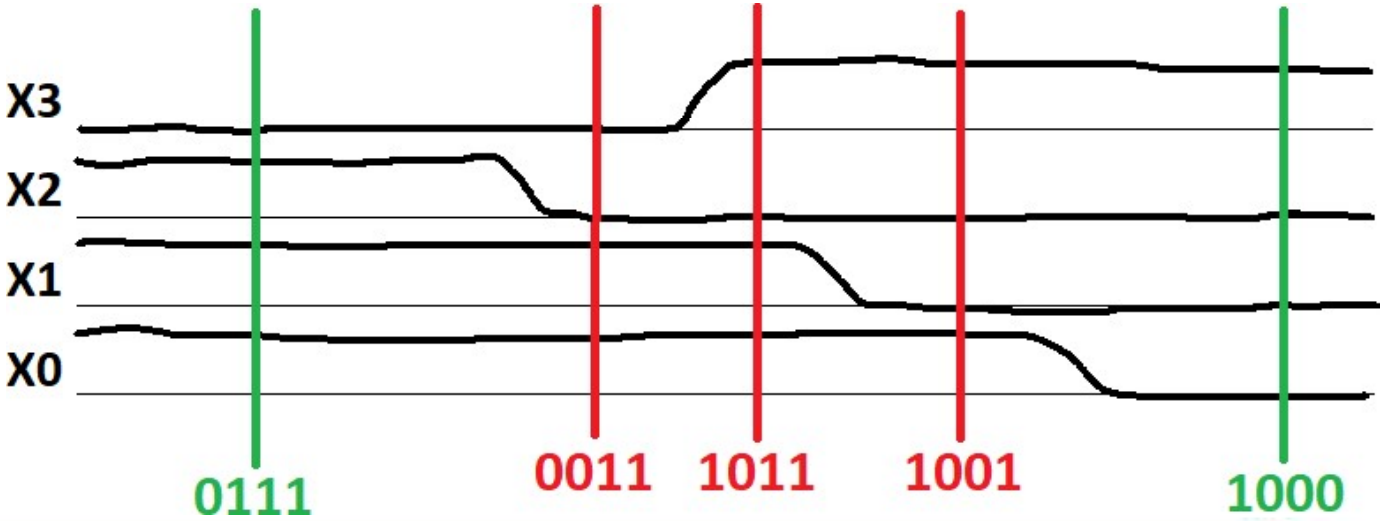
0111 → 0011 → 1011 → 1001 → 1000

W przybliżeniu…



Przyczyny problemów:

- **niedokładności (tolerancja) wykonania tarczy;**
- **różne opóźnienia w torze akwizycji sygnału każdego z bitów;**
- **ograniczona szybkość narastania zbocza sygnału.**



AGH

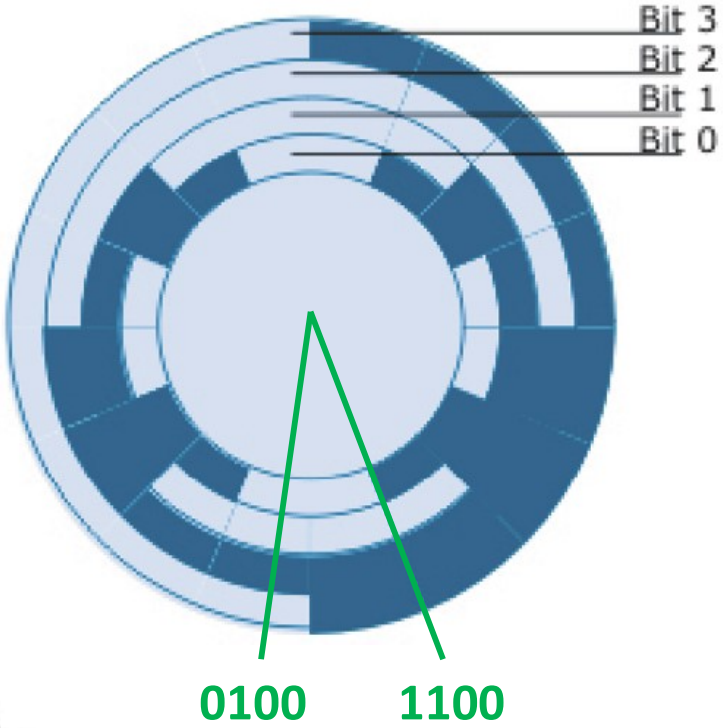
Kod Graya

(Zastosowanie kodu Graya w konstrukcji tarczy enkodera absolutnego)

ROZWIĄZANIE: Tarcza wykonana w kodzie Graya + transkoder Gray => NBC

W przypadku tarczy wykonanej w kodzie Graya zawsze następuje zmiana tylko jednego bitu, zatem sytuacja stanów przejściowych nie występuje.

Tarcza w kodzie Graya:



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 <-127, 127>.

- 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} … a_2 a_1 a_0], to jej wartość wynosi:

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

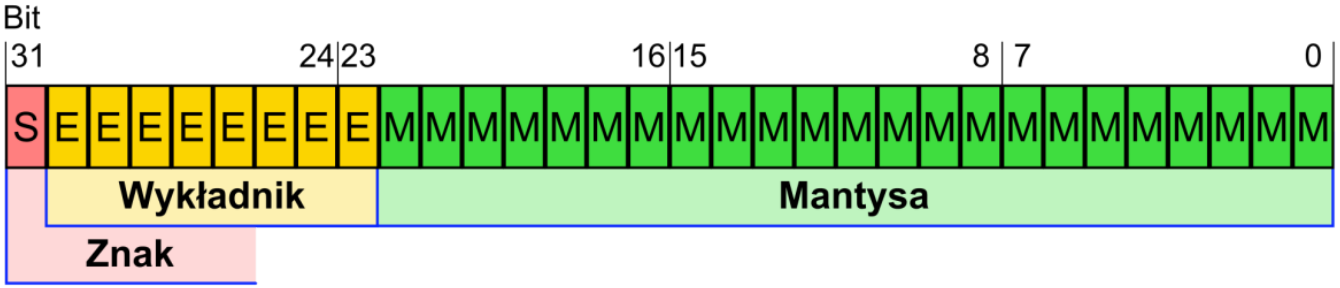
- Zakres n-bitowej liczby wynosi: [-2^{n-1} … $2^{n-1}-1$] , np. dla liczby 8-bitowej <-128, 127>.
- Dla wartości dodatnich liczby zapisane w kodzie U2 mają taką samą postać jak w NKB.
- Znalezienie liczby o przeciwnej wartości:
  - negacja wszystkich bitów;
  - dodanie 1.
- **Operacje arytmetyczne wykonywane 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.

- Pozwala na reprezentację liczba dodatnich, ujemnych, całkowitoliczbowych, ułamkowych oraz znaków specjalnych (np. nieskończoność lub NaN).
- Reprezentacja 32-bitowa, składa się z: 1 bitu znaku, 8 bitów wykładnika, 23-bitów mantysy.
- W języku C/C++ liczba zmiennoprzecinkowa pojedynczej precyzji jest kodowania typem **float**.



$$M = 1.0 + \sum_{i=0}^{22} m_i \cdot 2^{-}$$

$$E = \sum_{i=0}^{7} e_i \cdot 2^i$$

D – wartość liczby
S – bit znaku
m_i – wartości bitów mantysy [m_{22} m_{21} … m_2 m_1 m_0]
e_i – wartości bitów wykładnika [e_7 e_6 e_5 e_4 e_3 e_2 e_1 e_0]

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 ‘00000000000000000000000’	Liczba nieznormalizowana
‘0’	‘00000000’	‘00000000000000000000000’	+0
‘1’	‘00000000’	‘00000000000000000000000’	-0
‘0’	‘11111111’	‘00000000000000000000000’	+ ∞
‘1’	‘11111111’	‘00000000000000000000000’	- ∞
Dowolny	‘11111111’	Różna od ‘00000000000000000000000’	NaN

Tab. Kodowanie liczb zmiennoprzecinkowych pojedynczej precyzji

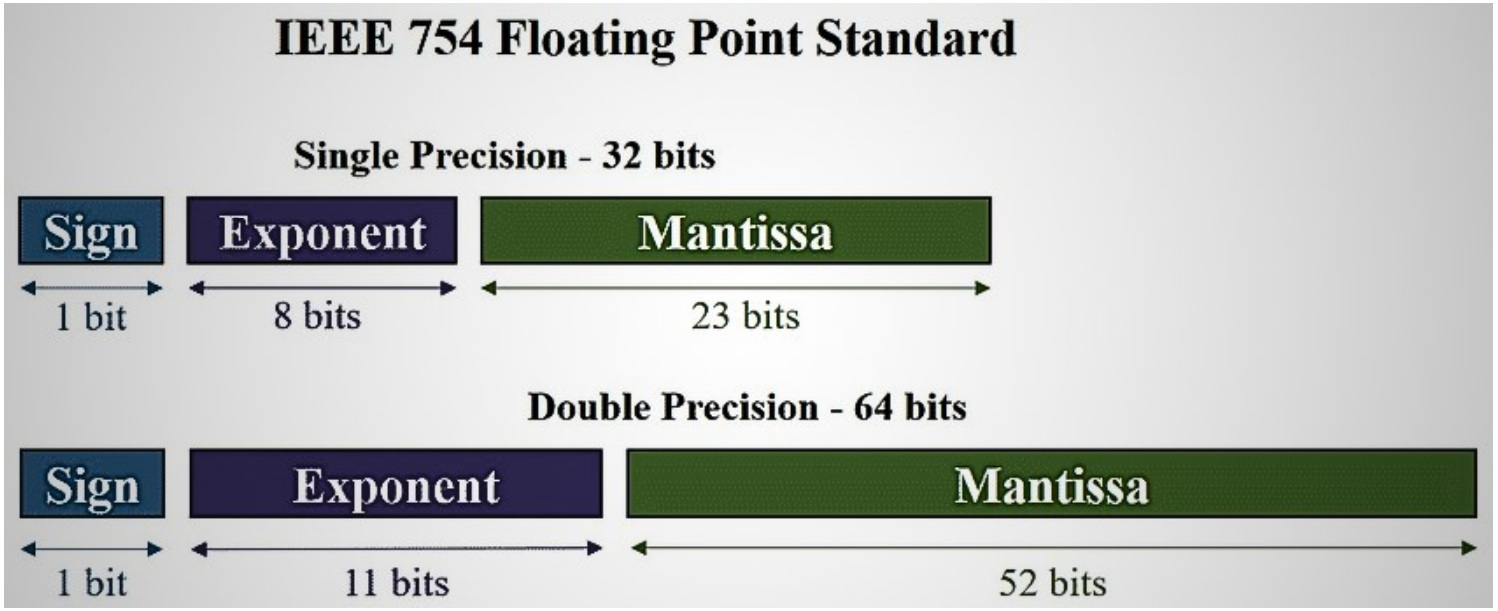
Liczba zmiennoprzecinkowa podwójnej precyzji (double)

Liczba zmiennoprzecinkowa podwójnej precyzji (double) – kodowanie pozwalające na reprezentację liczb zmiennoprzecinkowych. Większa precyzja obliczeń niż w przypadku liczb zmiennoprzecinkowych pojedynczej precyzji (float)

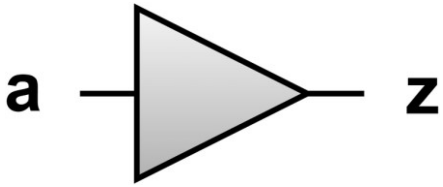
- Podobnie jak float, pozwala na reprezentację liczba dodatnich, ujemnych, całkowitoliczbowych, ułamkowych

oraz znaków specjalnych (np. nieskończoność lub NaN).

- Reprezentacja 64-bitowa, składa się z: 1 bitu znaku, 11 bitów wykładnika, 52-bitów mantysy.
- W języku C/C++ liczba zmiennoprzecinkowa pojedynczej precyzji jest kodowania typem **double**.

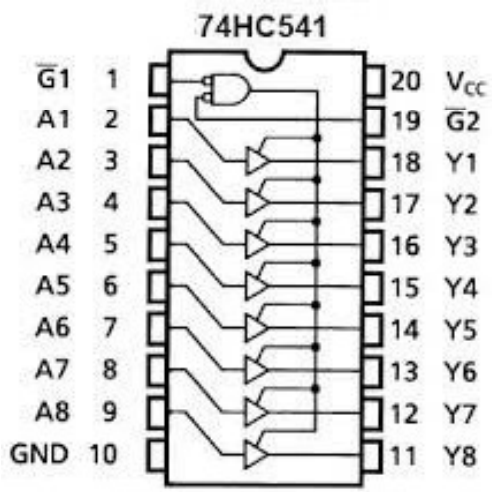


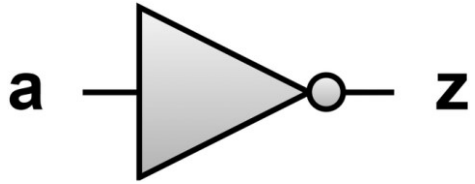
Bramki: BUF, NOT



$z = a$

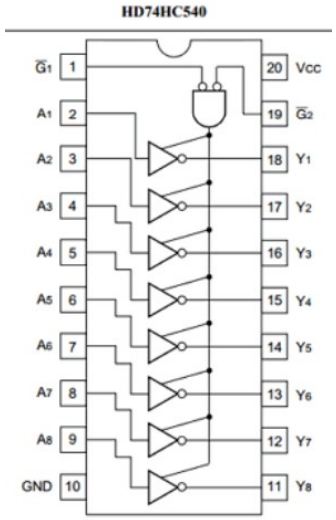
a	z
0	0
1	1



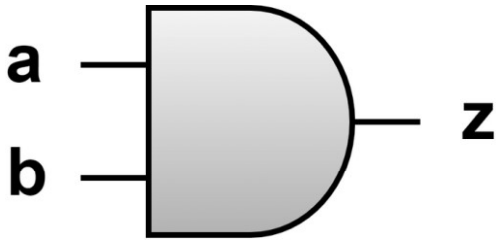


$z = \overline{a}$

a	z
0	1
1	0

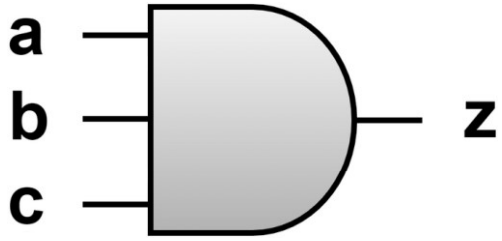


Bramki: AND (iloczyn logiczny, koniunkcja)



$z = ab$

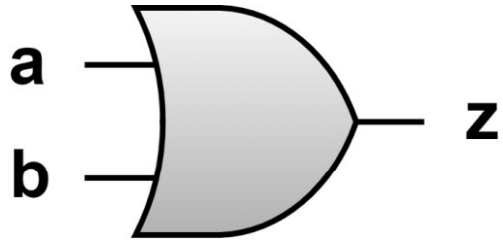
a	b	z
0	0	0
0	1	0
1	0	0
1	1	1



$z = abc$

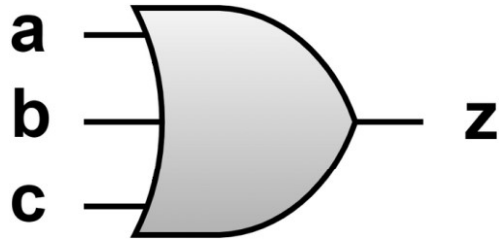
a	b	c	z
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

*Bramki: OR (suma logiczna, a*lternatywa*)*



z = a+b

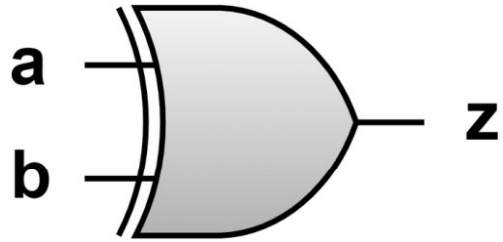
a	b	z
0	0	0
0	1	1
1	0	1
1	1	1



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)

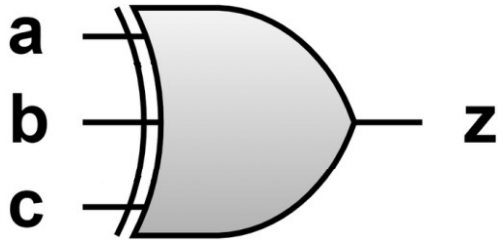


$$z = a \oplus b$$

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

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

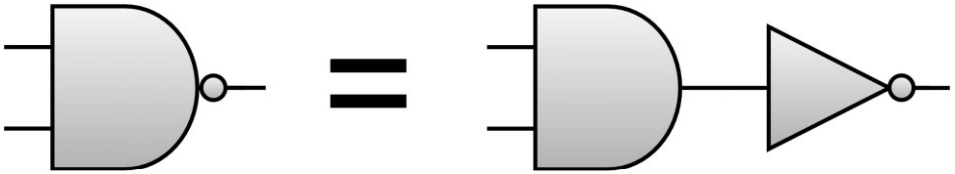
XOR = suma modulo 2

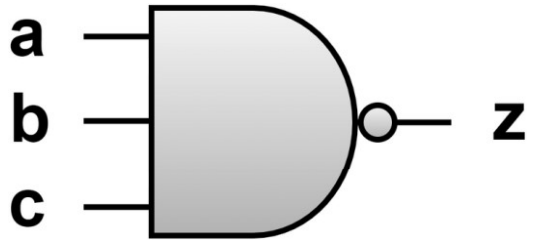


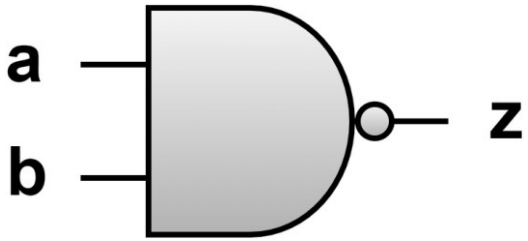
$$z = a \oplus b \oplus c$$

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

Bramki: NAND







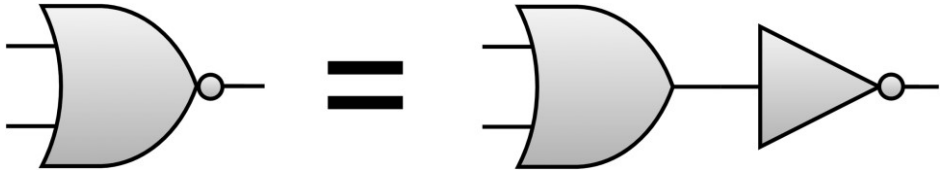
$$z = \overline{ab}$$

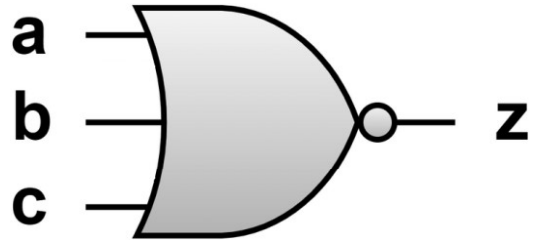
a	b	z
0	0	1
0	1	1
1	0	1
1	1	0

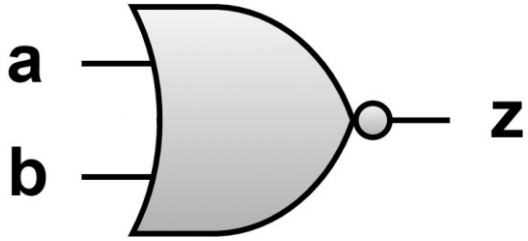
$$z = \overline{abc}$$

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







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

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

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

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

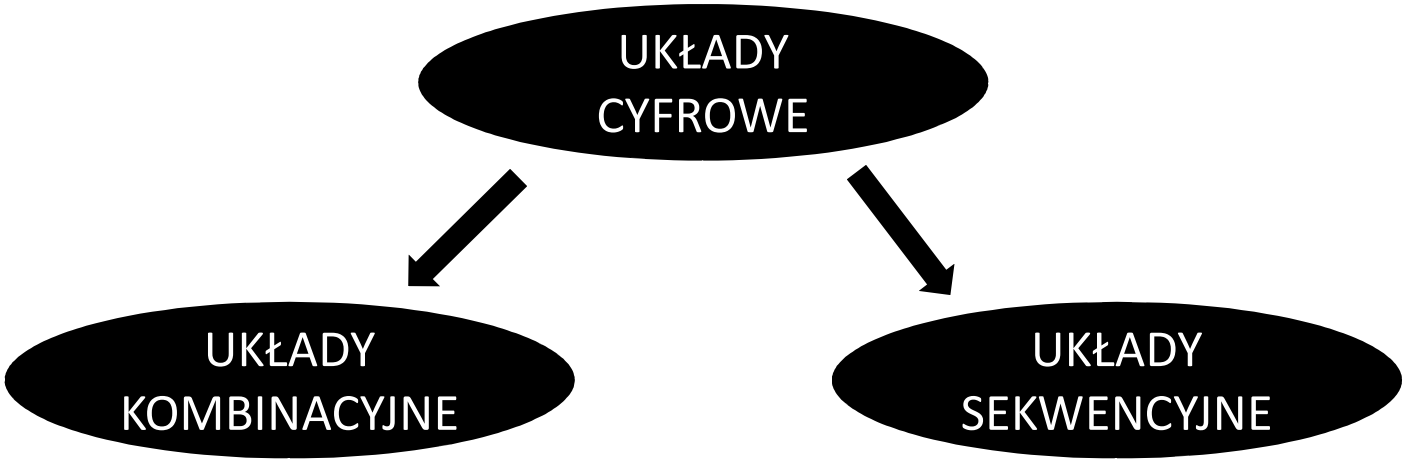
$$\overline{\overline{A}} = A$$

$$A + 0 = A$$
$$A + 1 = 1$$
$$A + A = A$$
$$A + \overline{A} = 1$$

$$A \cdot 0 = 0$$
$$A \cdot 1 = A$$
$$A \cdot A = A$$
$$A \cdot \overline{A} = 0$$

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

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.*

Synteza układów kombinacyjnych

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)]

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

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)]

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

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

Synteza układów kombinacyjnych na podstawie tablicy prawdy

Idx.	Wejścia			Wyjście	Mintermy:	Maxtermy:
i	a	b	c	y		
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):

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

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

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.

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

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

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

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

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

<u>Upraszczanie równań przy pomocy algebry Boolea jest procesem złożonym i pracochłonnym. Poziom skomplikowania szybko rośnie wraz ze wzrostem liczby zmiennych.</u>

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

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 2:

Pogrupowanie 1 (implikanty) lub 0 (implicenty).

- 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 opisania – w konsekwencji uproszczenie równania).

Synteza układów kombinacyjnych: (3) wykorzystanie tablic Karnaugh

CP \ BM	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).

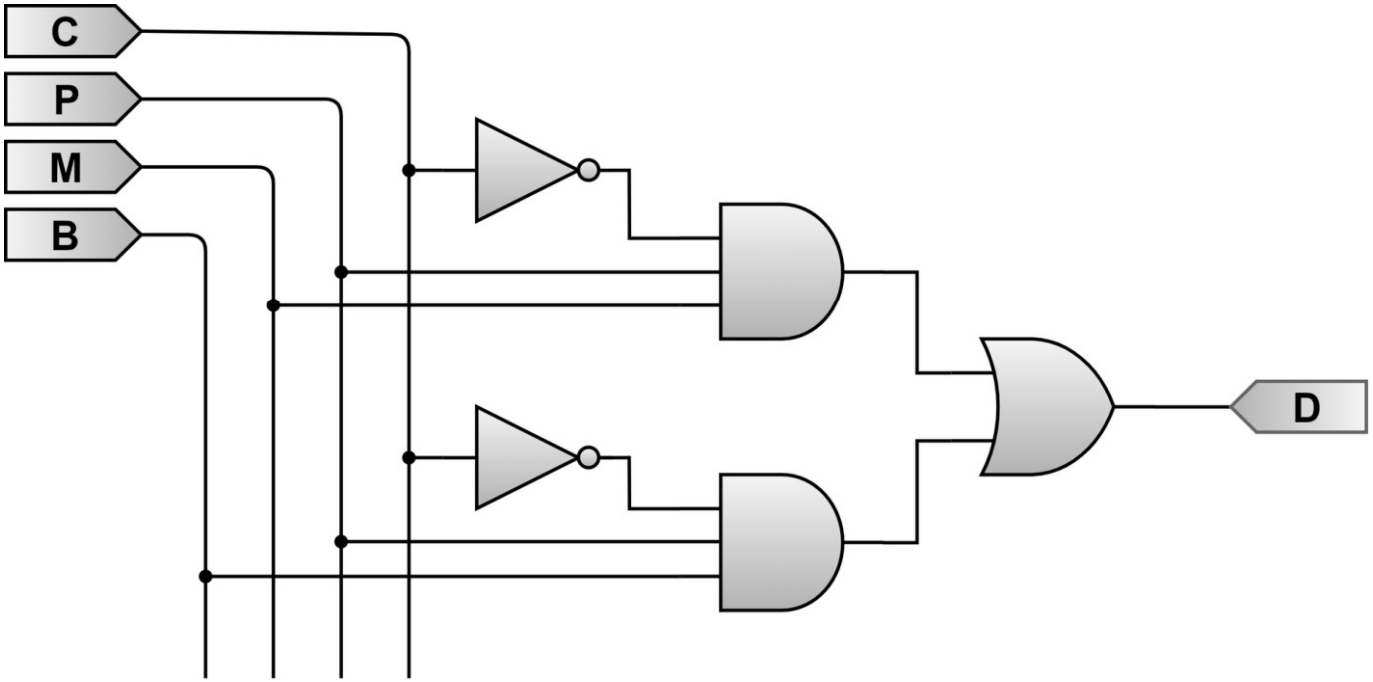
$$D = \overline{C} \cdot P \cdot M + \overline{C} \cdot P \cdot B$$

Synteza układów kombinacyjnych: (3) wykorzystanie tablic Karnaugh

Krok 4:

Narysowanie schematu logicznego na podstawie równania.

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



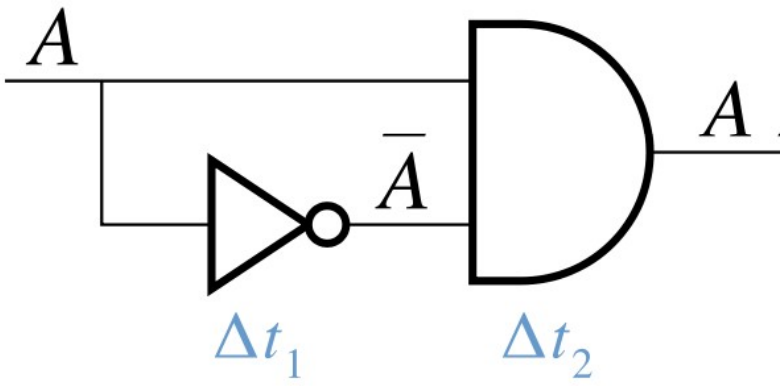
Synteza układów kombinacyjnych: wykorzystanie tablic Karnaugh - Podsumowanie

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 opisania – 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.

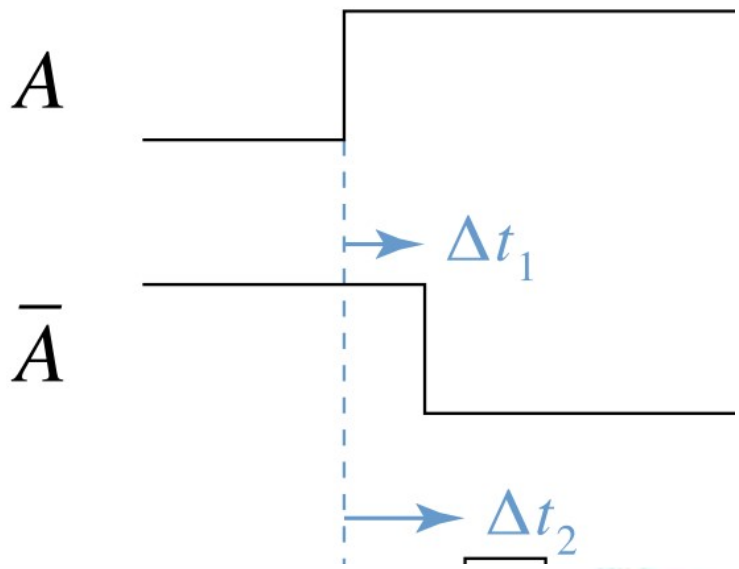
Zjawisko hazardu w układach cyfrowych

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



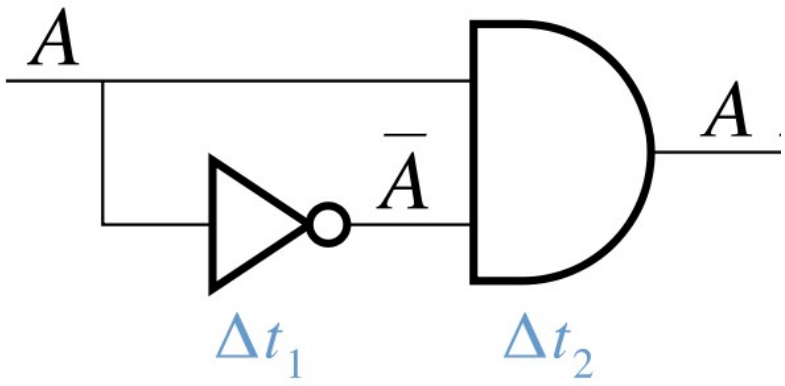
Zazwyczaj czas propagacji jest różny dla zmiany wyjścia ze stanu niskiego na wysoki (t_{LH}) i wysokiego na niski (t_{HL}).

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



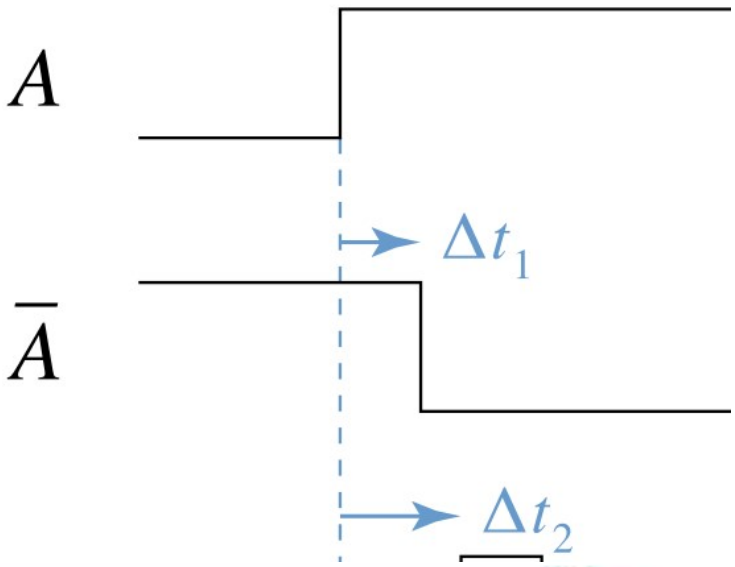
Zjawisko hazardu w układach cyfrowych

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 krótkotrwale 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.

Binarny system liczbowy

Na świecie istnieją 10 typy ludzi:

Ci którzy rozumieją system binarny,

i Ci, którzy go nie rozumieją.

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. in˙z. 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

[8] Understanding Floating-Point Arithmetic: How Computers Handle Decimals and Why It Matters - On-line 27.09.2025: https://medium.com/@yuvindur/understanding-floating-point-arithmetic-how-computers-handle-decimals-and-why-it-matters-1e32aec8d2ea

[9] Absolute Rotary Encoder - On-line 27.09.2025: https://www.omchsmps.com/absolute-rotary-encoder/