

UKŁADY CYFROWE – przewodnik po wykładzie

Materiały pomocnicze dla studentów kierunku Informatyka

* * *

Uwaga! Niniejszy materiał zawiera jedynie zarys treści podawanych na wykładzie i w żadnym razie nie zastępuje odpowiednich pozycji literaturowych.

Podstawowe zagadnienia:

- system pozycyjny zapisu liczb,
- podstawowe kody, ich własności i zastosowania,
- aksjomaty i twierdzenia algebry Boole’a oraz przekształcanie funkcji logicznych,
- kanoniczne postaci funkcji logicznych,
- minimalizacja funkcji logicznych,
- podstawowe funktory logiczne i kanoniczna realizacja funkcji logicznych,
- realizacja funkcji logicznych na multiplexerach,
- matematyczny opis układów logicznych,
- opis i synteza układów kombinacyjnych,
- opis i synteza układów iteracyjnych,
- przerzutniki synchroniczne, zasada działania i sposoby wyzwalania,
- projektowanie układów sekwencyjnych synchronicznych,
- wyścigi i hazardy w układach sekwencyjnych asynchronicznych,
- projektowanie układów sekwencyjnych asynchronicznych,
- przykładowe testy egzaminacyjne.

Literatura:

1. Barski M., Jędruch W.: Układy cyfrowe i mikroprocesory – skrypt. Wyd. PG 1985 (lub 2007).
2. Barski M., Jędruch W., Niedźwiecki M., Raczyński P., Sarzyński B.: Układy cyfrowe i mikroprocesory – zadania. Wyd. PG 1984.
3. Baranowski J., Kalinowski B., Nosal Z.: Układy elektroniczne cz. III – układy i systemy cyfrowe. Wyd. 2 (seria Podręczniki Akademickie), WNT 1998.
4. Kalisz J.: Podstawy elektroniki cyfrowej. Wyd. 3, WKŁ 1988.
5. Majewski W.: Układy logiczne. Wyd. 6 (seria Podręczniki Akademickie), WNT 1999.
6. Traczyk W.: Układy cyfrowe. Podstawy teoretyczne i metody syntezy (seria Elektronika-Informatyka-Telekomunikacja), WNT 1982.
7. Nelson V.P., Nagle H.T., Carroll B.D., Irwin J.D.: Digital Logic Circuit Analysis and Design. Prentice Hall 1985.
8. De Micheli G.: Synteza i optymalizacja układów cyfrowych.

System pozycyjny zapisu liczb

- **System o podstawie 10 (dec):** cyfry znaczące $\{ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 \}$

Przykład: $L = (3954,287)_{10}$

wagi	10^3	10^2	10^1	10^0	,	10^{-1}	10^{-2}	10^{-3}
cyfry	d_3	d_2	d_1	d_0	,	d_{-1}	d_{-2}	d_{-3}
$L =$	3	9	5	4	,	2	8	7

$$L = 3 \cdot 10^3 + 9 \cdot 10^2 + 5 \cdot 10^1 + 4 \cdot 10^0 + 2 \cdot 10^{-1} + 8 \cdot 10^{-2} + 7 \cdot 10^{-3}$$

- **System o podstawie 2 (bin):** cyfry znaczące $\{ 0, 1 \}$

Przykład: $L = (1011,011)_2$

wagi	2^3	2^2	2^1	2^0	,	2^{-1}	2^{-2}	2^{-3}
cyfry	b_3	b_2	b_1	b_0	,	b_{-1}	b_{-2}	b_{-3}
$L =$	1	0	1	1	,	0	1	1

$$L = 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 + 0 \cdot 2^{-1} + 1 \cdot 2^{-2} + 1 \cdot 2^{-3}$$

- **System o podstawie 16 (hex):** cyfry znaczące $\{0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F\}$

Przykład: $L = (12AF,C53)_{16}$

wagi	16^3	16^2	16^1	16^0	,	16^{-1}	16^{-2}	16^{-3}
cyfry	h_3	h_2	h_1	h_0	,	h_{-1}	h_{-2}	h_{-3}
$L =$	1	2	A	F	,	C	5	3

$$L = 1 \cdot 16^3 + 2 \cdot 16^2 + A \cdot 16^1 + F \cdot 16^0 + C \cdot 16^{-1} + 5 \cdot 16^{-2} + 3 \cdot 16^{-3}$$

- **System o podstawie "g":** cyfry znaczące $\{ 0, 1, \dots, g-1 \}$

$$L = \dots c_3 \cdot g^3 + c_2 \cdot g^2 + c_1 \cdot g^1 + c_0 \cdot g^0 + c_{-1} \cdot g^{-1} + c_{-2} \cdot g^{-2} + c_{-3} \cdot g^{-3} + \dots$$

Ogólnie liczbę można zapisać w postaci:

$$L = \sum_{i=0}^n c_i \cdot g^i + \sum_{j=1}^m c_{-j} \cdot g^{-j}$$

gdzie część całkowita zawiera $(n + 1)$ cyfr, a część ułamkowa zawiera (m) cyfr.

Ile cyfr znaczących posiada liczba zapisana w systemie o podstawie "g"?

Przyjmijmy oznaczenia: L – liczba naturalna,

g – podstawa systemu,

N_g^k – tyle pozycji zajmuje liczba k .

Wyrażając liczbę (L) jako potęgę podstawy liczenia (g) można zapisać: $L = g^p$.

Logarytmując następnie powyższe wyrażenie otrzymuje się: $\log L = p \cdot \log g$.

Wreszcie pozbywając się części ułamkowej z p dostajemy: $N_g^k = \lfloor p \rfloor + 1 = \lfloor \log k / \log g \rfloor + 1$.

Przykład: zapis liczby 100 w różnych systemach.

$$N_2^{100} = \left\lfloor \frac{\log 100}{\log 2} \right\rfloor + 1 = 7 \text{ cyfr} \quad (\text{zapis binarny: } 1100100_B)$$

$$N_{10}^{100} = \left\lfloor \frac{\log 100}{\log 10} \right\rfloor + 1 = 3 \text{ cyfry} \quad (\text{zapis dziesiętny: } 100)$$

$$N_{16}^{100} = \left\lfloor \frac{\log 100}{\log 16} \right\rfloor + 1 = 2 \text{ cyfry} \quad (\text{zapis heksadecymalny: } 64_H)$$

Zamiana liczby dziesiętnej na binarną:

- **zamiana części całkowitej**: dzielimy liczbę przez dwa i notujemy kolejne reszty
- **zamiana części ułamkowej**: mnożymy liczbę przez dwa i notujemy wartości całkowite

Uzasadnienie: liczba jest sumą części całkowitej i ułamkowej ($L = L_C + L_U$)

Część całkowita: $L_C = \dots b_4 \cdot 2^4 + b_3 \cdot 2^3 + b_2 \cdot 2^2 + b_1 \cdot 2^1 + b_0 \cdot 2^0$

Dzielenie przez 2: $L_C / 2 = \dots b_4 \cdot 2^3 + b_3 \cdot 2^2 + b_2 \cdot 2^1 + b_1 \cdot 2^0 + b_0 \cdot 2^{-1}$ (reszta b_0)

Część ułamkowa: $L_U = b_{-1} \cdot 2^{-1} + b_{-2} \cdot 2^{-2} + b_{-3} \cdot 2^{-3} + \dots$

Mnożenie przez 2: $L_U \cdot 2 = b_{-1} \cdot 2^0 + b_{-2} \cdot 2^{-1} + b_{-3} \cdot 2^{-2} + \dots$ (część całkowita b_{-1})

Zatem dzieląc L_C przez 2 otrzymujemy bity (b_0, b_1, \dots) jako kolejne reszty.

Z kolei mnożąc L_U przez 2 otrzymujemy bity (b_{-1}, b_{-2}, \dots) jako wynikowe wartości całkowite.

Przykład: zamiana liczby dziesiętnej $L = 23,375$ na binarną ($L = 10111,011$)₂.

Część całkowita – dzielenie przez 2				Część ułamkowa – mnożenie przez 2		
dzielne	23	$1 = b_0$	reszty	całkowite	0,	375 ułamki
	11	$1 = b_1$			$b_{-1} = 0,$	750
	5	$1 = b_2$			$b_{-2} = 1,$	500
	2	$0 = b_3$			$b_{-3} = 1,$	000
	1	$1 = b_4$				
	0					

Podstawowe kody

- NKB (naturalny kod binarny)
- BCD (*binary coded decimal*: cyfra dziesiętna zapisana na 4-bitach)
- Z-M (znak-moduł: moduł NKB i bit znaku 0 = plus, 1 = minus)
- U1 (uzupełnienie do 1: negacja liczby $-x = \bar{x}$)
- U2 (uzupełnienie do 2: negacja liczby $-x = \bar{x} + 1$)

Ważne: dodawanie i odejmowanie liczb w kodzie U2 wykonuje się identycznie jak w NKB!

Przykład: interpretacje kodu czterobitowego.

Kod (4 bity)	NKB	BCD	Z-M	U1	U2
0 0 0 0	0	0	+ 0	+ 0	+ 0
0 0 0 1	1	1	+ 1	+ 1	+ 1
0 0 1 0	2	2	+ 2	+ 2	+ 2
0 0 1 1	3	3	+ 3	+ 3	+ 3
0 1 0 0	4	4	+ 4	+ 4	+ 4
0 1 0 1	5	5	+ 5	+ 5	+ 5
0 1 1 0	6	6	+ 6	+ 6	+ 6
0 1 1 1	7	7	+ 7	+ 7	+ 7
1 0 0 0	8	8	− 0	− 7	− 8
1 0 0 1	9	9	− 1	− 6	− 7
1 0 1 0	10	−	− 2	− 5	− 6
1 0 1 1	11	−	− 3	− 4	− 5
1 1 0 0	12	−	− 4	− 3	− 4
1 1 0 1	13	−	− 5	− 2	− 3
1 1 1 0	14	−	− 6	− 1	− 2
1 1 1 1	15	−	− 7	− 0	− 1

Ważne: na n – bitach możemy zapisać liczby:

NKB z zakresu $\langle 0, 2^n - 1 \rangle$

U2 z zakresu $\langle -2^{n-1}, 2^{n-1} - 1 \rangle$

U1 z zakresu $\langle -2^{n-1} + 1, 2^{n-1} - 1 \rangle$

Z-M z zakresu $\langle -2^{n-1} + 1, 2^{n-1} - 1 \rangle$

Przykład: interpretacje pewnego kodu binarnego (1 0 0 1 0 1 1 1).

$L = (1\ 0\ 0\ 1\ 0\ 1\ 1\ 1)_{\text{NKB}} = 151$ (czyli 97_{H})

$L = (1\ 0\ 0\ 1\ 0\ 1\ 1\ 1)_{\text{BCD}} = 97$

$L = (1\ 0\ 0\ 1\ 0\ 1\ 1\ 1)_{\text{Z-M}} = -23$

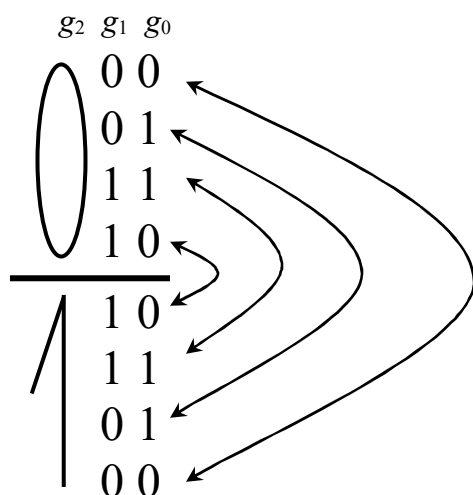
$L = (1\ 0\ 0\ 1\ 0\ 1\ 1\ 1)_{\text{U1}} = -104$

$L = (1\ 0\ 0\ 1\ 0\ 1\ 1\ 1)_{\text{U2}} = -105$

Kod refleksyjny (Gray'a) – sposoby tworzenia

- Metoda lustrzanego odbicia bitów
- Metoda przeplatania ciągów bitowych
- Metoda analityczna

W metodzie lustrzanego odbicia zapisujemy kod Gray'a na znanej liczbie bitów (np. na dwóch bitach, czyli: 00, 01, 11, 10). Następnie przepisujemy symetrycznie ("lustrzanie") kody poniżej poziomej kreski. Nad kreską dopisujemy bity "0", a pod kreską dopisujemy bity "1". Opisaną metodę ilustruje poniższy rysunek.



W metodzie przeplatania ciągów bitowych zapisujemy kolejne kolumny tablicy wg zasady:

w kolumnie bitu g_0 zapisujemy na przemian ciągi $\{0,1\}$ i $\{1,0\}$,

w kolumnie bitu g_1 zapisujemy na przemian ciągi $\{0,0,1,1\}$ i $\{1,1,0,0\}$,

:

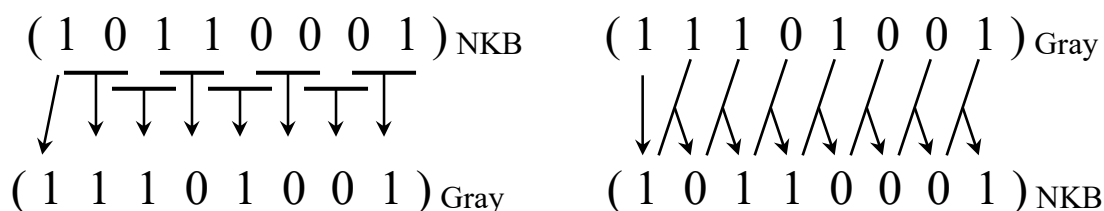
w kolumnie bitu g_n zapisujemy na przemian ciągi $\{2^n \text{ zer}, 2^n \text{ jedynek}\}$ i $\{2^n \text{ jedynek}, 2^n \text{ zer}\}$.

W metodzie analitycznej zamianę NKB na kod Gray'a (i odwrotnie) dokonujemy wg wzorów:

$$\text{NKB} \rightarrow \text{Gray: } g_N = b_N \quad \text{i} \quad g_i = b_i \oplus b_{i+1} \quad \text{dla} \quad i = 0, 1, \dots, N-1,$$

$$\text{Gray} \rightarrow \text{NKB: } b_N = g_N \quad \text{i} \quad b_i = g_i \oplus b_{i+1} \quad \text{dla} \quad i = N-1, \dots, 0,$$

gdzie symbol \oplus oznacza "sumę modulo dwa" ("0" dla bitów równych i "1" dla bitów różnych). Opisaną metodę ilustruje poniższy rysunek (kreską połączono bity sumowane modulo 2).



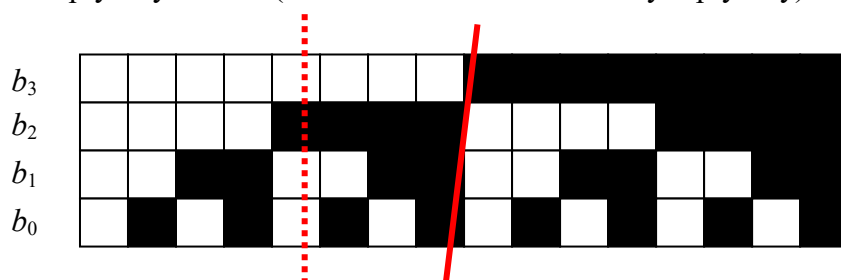
Ważne: w kodzie Gray'a sąsiednie liczby różnią się dokładnie na jednym bicie.

Przykład: tabela czterobitowego kodu NKB i Gray'a.

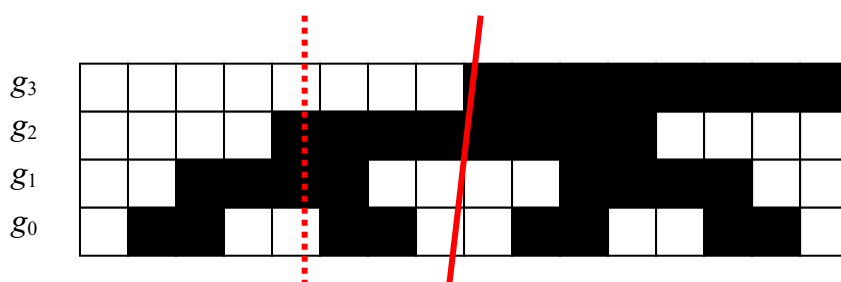
Liczba zapis dziesiętny	NKB $b_3 b_2 b_1 b_0$	Kod Gray'a $g_3 g_2 g_1 g_0$
0	0 0 0 0	0 0 0 0
1	0 0 0 1	0 0 0 1
2	0 0 1 0	0 0 1 1
3	0 0 1 1	0 0 1 0
4	0 1 0 0	0 1 1 0
5	0 1 0 1	0 1 1 1
6	0 1 1 0	0 1 0 1
7	0 1 1 1	0 1 0 0
8	1 0 0 0	1 1 0 0
9	1 0 0 1	1 1 0 1
10	1 0 1 0	1 1 1 1
11	1 0 1 1	1 1 1 0
12	1 1 0 0	1 0 1 0
13	1 1 0 1	1 0 1 1
14	1 1 1 0	1 0 0 1
15	1 1 1 1	1 0 0 0

Zastosowanie kodu Gray'a w pomiarach optycznych (liniały i tarcze kodowe)

- Liniał optyczny z NKB (czerwona linia oznacza odczyt optyczny):



- Liniał optyczny z kodem Gray'a (czerwona linia oznacza odczyt optyczny):



Ważne: w przypadku pochylenia linii odczytu optycznego (linia ciągła na rysunkach):

- w NKB może wystąpić chwilowe przekłamanie (na rys. po stanie $0111_B = 7$ jest $1111_B = 15$),
- w kodzie Gray'a odczyt jest ciągle poprawny (na rys. po stanie $0100_G = 7$ jest $1100_G = 8$).

Algebra Boole'a

Stałe logiczne: 0 – fałsz , 1 – prawda

Podstawowe działania	notacja klasyczna	notacja w układach cyfrowych
negacja logiczna – NOT (zaprzeczenie)	$\sim x$	\bar{x}
suma logiczna – OR (alternatywa zwykła)	$x \vee y$	$x + y$
suma modulo 2 – XOR (alternatywa rozłączna)	$x \dot{\vee} y$	$x \oplus y$
iloczyn logiczny – AND (koniunkcja)	$x \wedge y$	$x \cdot y$
implikacja – IF (jeżeli ... to)	$x \Rightarrow y$	$x \rightarrow y$
równoważność – EQ (wtedy i tylko wtedy)	$x \Leftrightarrow y$	$x \equiv y$

Tabela podstawowych funkcji

funkcja $x \quad y$	NOT \bar{x}	OR $x + y$	XOR $x \oplus y$	AND $x \cdot y$	IF $x \rightarrow y$	EQ $x \equiv y$	NOR $\overline{x + y}$	NAND $\overline{x \cdot y}$
0 0	1	0	0	0	1	1	1	1
0 1		1	1	0	1	0	0	1
1 0	0	1	1	0	0	0	0	1
1 1		1	0	1	1	1	0	0

Ważne: z tabeli wynikają następujące tożsamości:

- $\overline{x + y} = \bar{x} \cdot \bar{y}$; NOR nazywane jest też funkcją Pierce'a (ozn. $\overline{x + y} = x \downarrow y$),
- $\overline{x \cdot y} = \bar{x} + \bar{y}$; NAND nazywane jest też funkcją Sheffera (ozn. $\overline{x \cdot y} = x / y$),
- $\overline{x \oplus y} = (x \equiv y)$; XOR zanegowany oznacza równoważność,
- $\overline{x \rightarrow y} = x \cdot \bar{y}$; IF jest fałszywe wtedy i tylko wtedy gdy z prawdy wynika fałsz,
- $x \rightarrow y = \bar{x} + y$; IF jest prawdziwe, gdy x jest fałszem lub y jest prawdą.

Ile można zbudować różnych funkcji n -zmiennych ?

- argumenty x_1, x_2, \dots, x_n : liczba wierszy w tabeli 2^n ,
- funkcje $f(x_1, x_2, \dots, x_n)$: liczba różnych funkcji 2^{2^n} .

Przykład: liczba funkcji n -zmiennych.

- 1 – zmienna: 4 – funkcje (tj. $f(x)=0$, $f(x)=1$, $f(x)=x$, $f(x)=\bar{x}$),
- 2 – zmienne: 16 – funkcji,
- 3 – zmienne: 256 – funkcji.

Aksjomaty i twierdzenia:

Aksjomaty	Twierdzenia
$x + 0 = x$ $x \cdot 1 = x$ $x + \bar{x} = 1$ $x \cdot \bar{x} = 0$ przemienność sumy logicznej $x + y = y + x$ przemienność iloczynu logicznego $x \cdot y = y \cdot x$ rozdzielność sumy względem iloczynu $x \cdot y + x \cdot z = x \cdot (y + z)$ rozdzielność iloczynu względem sumy $(x + y) \cdot (x + z) = x + y \cdot z$	$x + 1 = 1$, $x + x = x$ $x \cdot 0 = 0$, $x \cdot x = x$ $\bar{\bar{x}} = x$ $x + \bar{x} \cdot y = x + y$ $x \cdot (\bar{x} + y) = x \cdot y$ prawa de Morgana $\overline{x \cdot y} = \bar{x} + \bar{y}$ $\overline{x + y} = \bar{x} \cdot \bar{y}$ reguły pochłaniania $x + x \cdot y = x$ $x \cdot (x + y) = x$ reguły sklejania $x \cdot y + x \cdot \bar{y} = x$ $(x + y) \cdot (x + \bar{y}) = x$

Metody dowodzenie tożsamości logicznych:

- tabela binarna (sprawdzenie wszystkich możliwości),
- schemat Venna (interpretacja operacji logicznych w kategorii rachunku zbiorów),
- przekształcenie analityczne (upraszczanie wyrażeń z wykorzystaniem aksjomatów i twierdzeń).

Zadania: uprościć wyrażenia:

$$x + x \cdot y = \dots$$

$$(x + y) \cdot (x + \bar{y}) = \dots$$

$$x \cdot \bar{y} + z + (\bar{x} + y) \cdot \bar{z} = \dots$$

$$x \cdot y + \bar{x} \cdot y \cdot \bar{z} + y \cdot z = \dots$$

Zadania: podane operacje logiczne wyrazić za pomocą sumy, iloczynu i negacji:

$$x \oplus y = \dots$$

$$x \equiv y = \dots$$

$$x \rightarrow y = \dots$$

Zadania: zilustrować na schematach Venna podane tożsamości:

$$x + \bar{x} \cdot y = x + y$$

$$x \oplus y = \bar{x} \cdot y + x \cdot \bar{y}$$

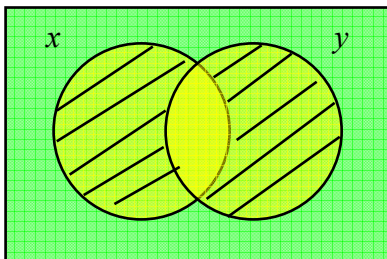
$$x \equiv y = \bar{x} \cdot \bar{y} + x \cdot y$$

$$x \rightarrow y = \bar{x} + y$$

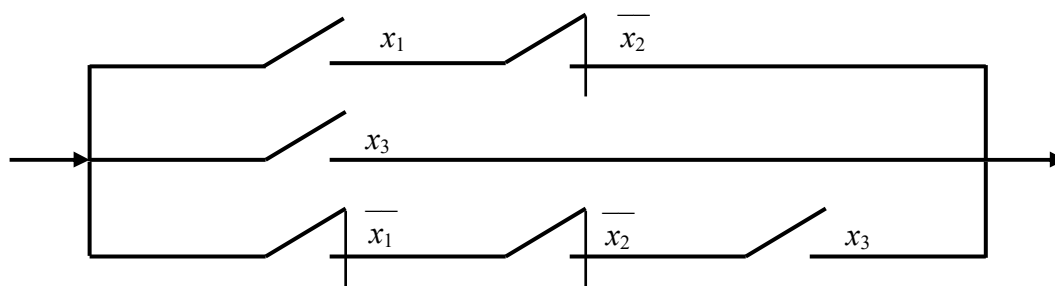
$$x \oplus y \oplus z = \bar{x} \cdot \bar{y} \cdot \bar{z} + x \cdot y \cdot z + \bar{x} \cdot y \cdot \bar{z} + x \cdot \bar{y} \cdot \bar{z}$$

Przykłady: wybrane rozwiązania.

- $(x + y) \cdot (x + \bar{y}) = x \cdot x + x \cdot \bar{y} + y \cdot x + y \cdot \bar{y} = x + x \cdot (y + \bar{y}) + 0 = x + x + 0 = x$.
- $x \rightarrow y = \dots$. Negacja implikacji jest prawdziwa, gdy x jest prawdą i y jest fałszem (tabela). Stąd:
 $x \rightarrow y = x \cdot \bar{y}$. Negując obydwie strony tej równości i stosując prawo de Morgana otrzymujemy:
 $\overline{x \rightarrow y} = \overline{x \cdot \bar{y}} = \bar{x} + \bar{\bar{y}} = \bar{x} + y$.
- $x \oplus y = \bar{x} \cdot y + x \cdot \bar{y}$ (na schemacie Venna obszar zakreskowany ilustruje sumę modulo dwa).



Zadanie: uprościć sieć zestyków stosując metodę ścieżek i metodę cięć:



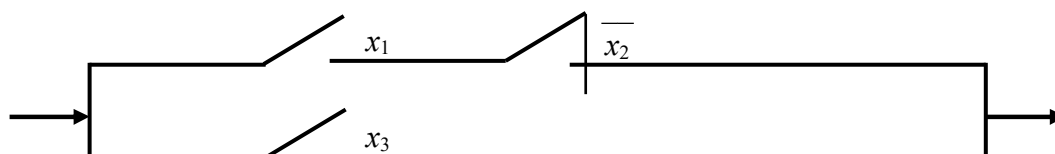
- W metodzie ścieżek funkcja ma postać sumy, przy czym każdy składnik jest iloczynem zmiennych na danej ścieżce prowadzącej od wejścia do wyjścia.

$$f(x_1, x_2, x_3) = x_1 \bar{x}_2 + x_3 + \bar{x}_1 \bar{x}_2 x_3 = x_1 \bar{x}_2 + x_3 (1 + \bar{x}_1 \bar{x}_2) = x_1 \bar{x}_2 + x_3.$$

- W metodzie cięć funkcja ma postać iloczynu, przy czym każdy czynnik jest sumą zmiennych na przecięciu powodującym rozwarcie całej sieci.

$$\begin{aligned} f(x_1, x_2, x_3) &= (x_1 + x_3 + \bar{x}_1)(x_1 + x_3 + \bar{x}_2)(x_1 + x_3 + x_3)(\bar{x}_2 + x_3 + \bar{x}_1)(\bar{x}_2 + x_3 + \bar{x}_2)(\bar{x}_2 + x_3 + x_3) = \\ &= (x_1 + x_3 + \bar{x}_2)(x_1 + x_3)(\bar{x}_2 + x_3 + \bar{x}_1)(\bar{x}_2 + x_3) = (x_1 + x_3)(1 + \bar{x}_2)(\bar{x}_2 + x_3)(1 + \bar{x}_1) = \\ &= (x_1 + x_3)(\bar{x}_2 + x_3) = x_1 \bar{x}_2 + x_1 x_3 + x_3 \bar{x}_2 + x_3 = x_1 \bar{x}_2 + x_3 (x_1 + \bar{x}_2 + 1) = \\ &= x_1 \bar{x}_2 + x_3. \end{aligned}$$

Z przeprowadzonej minimalizacji widać, że ostatnia gałąź sieci jest zbędna. Stąd sieć optymalna:



Kanoniczna Postać Sumacyjna (KPS lub ZNPS)

Reguła przekształcania: $f(x_1, x_2, \dots, x_n) = \bar{x}_1 \cdot f(0, x_2, \dots, x_n) + x_1 \cdot f(1, x_2, \dots, x_n)$

Rozkładamy kolejne składniki wg podanej reguły:

$$\begin{aligned} f(x_1, x_2, \dots, x_n) &= \bar{x}_1 \cdot \bar{x}_2 \cdot \dots \cdot \bar{x}_n \cdot f(0, 0, \dots, 0) + \\ &\quad + \bar{x}_1 \cdot \bar{x}_2 \cdot \dots \cdot x_n \cdot f(0, 0, \dots, 1) + \\ &\quad : \\ &\quad + x_1 \cdot x_2 \cdot \dots \cdot x_n \cdot f(1, 1, \dots, 1). \end{aligned}$$

KPS: $f(x_1, x_2, \dots, x_n) = \sum_{i=0}^{2^n-1} I_i \phi_i = \sum_{\phi_i=1} I_i$, gdzie

I_i iloczyn zmiennych, np. dla $i = 0$: $I_0 = \bar{x}_1 \cdot \bar{x}_2 \cdot \dots \cdot \bar{x}_n$

ϕ_i próbka funkcji $f()$, np. dla $i = 0$: $\phi_0 = f(0, 0, \dots, 0)$

Kanoniczna Postać Iloczynowa (KPI lub ZNPI)

Reguła: $f(x_1, x_2, \dots, x_n) = [x_1 + f(0, x_2, \dots, x_n)] \cdot [\bar{x}_1 + f(1, x_2, \dots, x_n)]$

Rozkładamy kolejne czynniki wg. podanej reguły:

$$\begin{aligned} f(x_1, x_2, \dots, x_n) &= [x_1 + x_2 + \dots + x_n + f(0, 0, \dots, 0)] \cdot \\ &\quad \cdot [x_1 + x_2 + \dots + \bar{x}_n + f(0, 0, \dots, 1)] \cdot \\ &\quad : \\ &\quad \cdot [\bar{x}_1 + \bar{x}_2 + \dots + \bar{x}_n + f(1, 1, \dots, 1)] \end{aligned}$$

KPI: $f(x_1, x_2, \dots, x_n) = \prod_{i=0}^{2^n-1} (S_i + \phi_i) = \prod_{\phi_i=0} S_i$, gdzie

S_i suma zmiennych, np. dla $i = 0$: $S_0 = x_1 + x_2 + \dots + x_n$

ϕ_i próbka funkcji $f()$, np. dla $i = 0$: $\phi_0 = f(0, 0, \dots, 0)$

Przykład: doprowadzić do postaci kanonicznych funkcję $f(x_1, x_2, x_3) = x_1 + \bar{x}_2 \cdot x_3$.

KPS: $f(x_1, x_2, x_3) = x_1 + \bar{x}_2 \cdot x_3 = x_1 \cdot 1 \cdot 1 + 1 \cdot \bar{x}_2 \cdot x_3 = x_1(\bar{x}_2 + x_2)(\bar{x}_3 + x_3) + (\bar{x}_1 + x_1)\bar{x}_2 \cdot x_3 =$
 $= x_1 \cdot \bar{x}_2 \cdot \bar{x}_3 + x_1 \cdot \bar{x}_2 \cdot x_3 + x_1 \cdot x_2 \cdot \bar{x}_3 + x_1 \cdot x_2 \cdot x_3 + \bar{x}_1 \cdot \bar{x}_2 \cdot x_3 + x_1 \cdot \bar{x}_2 \cdot x_3.$

Składniki kodujemy wg zasady: zmienna prosta $x_i \leftrightarrow 1$, zmienna negowana $\bar{x}_i \leftrightarrow 0$. Stąd:

$$f(x_1, x_2, x_3) = \sum (1, 4, 5, 6, 7) \text{ oraz przez dopełnienie } f(x_1, x_2, x_3) = \prod (0, 2, 3).$$

KPI: $f(x_1, x_2, x_3) = x_1 + \bar{x}_2 \cdot x_3 = (x_1 + \bar{x}_2)(x_1 + x_3) =$
 $= (x_1 + \bar{x}_2 + \bar{x}_3)(x_1 + \bar{x}_2 + x_3)(x_1 + \bar{x}_2 + x_3)(x_1 + x_2 + x_3).$

Składniki kodujemy wg zasady: zmienna prosta $x_i \leftrightarrow 0$, zmienna negowana $\bar{x}_i \leftrightarrow 1$. Stąd:

$$f(x_1, x_2, x_3) = \prod (0, 2, 3) \text{ oraz przez dopełnienie } f(x_1, x_2, x_3) = \sum (1, 4, 5, 6, 7).$$

Implikant funkcji $f(x_1, \dots, x_n)$ to taka funkcja $g(x_1, \dots, x_n)$, że jeżeli $g = 1$, to $f = 1$.

Implikant prosty: taki iloczyn zmiennych, który jest implikantem i który zmniejszony o dowolną zmienną przestaje nim być.

Implicent funkcji $f(x_1, \dots, x_n)$ to taka funkcja $h(x_1, \dots, x_n)$, że jeżeli $h = 0$, to $f = 0$.

Implicent prosty: taka suma zmiennych, która jest implicentem i która zmniejszona o dowolną zmienną przestaje nim być.

Wnioski:

- Implikant g pokrywa (część lub wszystkie) jedynki funkcji f .
- Implicent h pokrywa (część lub wszystkie) zera funkcji f .
- Każdą funkcję można przedstawić w postaci sumy wszystkich jej implikantów prostych:

$$f(x_1, \dots, x_n) = \sum g_i.$$
- Każdą funkcję można przedstawić w postaci iloczynu wszystkich jej implicentów prostych:

$$f(x_1, \dots, x_n) = \prod h_i.$$

Własności implikantów i implicentów:

Funkcja logiczna $f(x_1, \dots, x_n)$	
Implikant $g(x_1, \dots, x_n)$	Implicent $h(x_1, \dots, x_n)$
$g \rightarrow f \equiv 1$	$\bar{h} \rightarrow \bar{f} \equiv 1$
$\bar{g} + f \equiv 1$	$h + \bar{f} \equiv 1$
$g \cdot \bar{f} \equiv 0$	$\bar{h} \cdot f \equiv 0$
$g \cdot f \equiv g$	$h + f \equiv h$

Przykład: funkcję $f(x_1, x_2, x_3) = \bar{x}_1 \cdot x_3 + x_1 \cdot x_2$ doprowadzić do KPS i KPI; zbadać, czy $g = x_2 \cdot x_3$ jest implikantem prostym tej funkcji; zbadać, czy $h = x_1 + x_3$ jest implicentem prostym tej funkcji.

$$\begin{aligned} f(x_1, x_2, x_3) &= \bar{x}_1 \cdot x_3 + x_1 \cdot x_2 = \bar{x}_1 \cdot (\bar{x}_2 + x_2) \cdot x_3 + x_1 \cdot x_2 \cdot (\bar{x}_3 + x_3) = \\ &= \bar{x}_1 \cdot \bar{x}_2 \cdot x_3 + \bar{x}_1 \cdot x_2 \cdot x_3 + x_1 \cdot x_2 \cdot \bar{x}_3 + x_1 \cdot x_2 \cdot x_3 = \sum(1,3,6,7) = \prod(0,2,4,5). \end{aligned}$$

$$\begin{aligned} \text{Sprawdzenie implikantu: } \bar{g} + f &= \overline{x_2 \cdot x_3} + \bar{x}_1 \cdot x_3 + x_1 \cdot x_2 = \bar{x}_2 + \bar{x}_3 + \bar{x}_1 \cdot x_3 + x_1 \cdot x_2 = \\ &= (\bar{x}_2 + x_2 \cdot x_1) + (\bar{x}_3 + x_3 \cdot \bar{x}_1) = (\bar{x}_2 + x_1) + (\bar{x}_3 + \bar{x}_1) = \\ &= \bar{x}_1 + x_1 + \bar{x}_2 + \bar{x}_3 = 1 + \bar{x}_2 + \bar{x}_3 \equiv 1, \end{aligned}$$

$$\text{dla } g_1 = x_2: \bar{x}_2 + f = \bar{x}_2 + \bar{x}_1 \cdot x_3 + x_1 \cdot x_2 = x_1 + \bar{x}_2 + x_3 \neq 1,$$

$$\text{dla } g_2 = x_3: \bar{x}_3 + f = \bar{x}_3 + \bar{x}_1 \cdot x_3 + x_1 \cdot x_2 = \bar{x}_1 + x_2 + \bar{x}_3 \neq 1, \text{ czyli implikant prosty.}$$

$$\begin{aligned} \text{Sprawdzenie implicentu: } \bar{h} \cdot f &= \overline{(x_1 + x_3)} \cdot (\bar{x}_1 \cdot x_3 + x_1 \cdot x_2) = \bar{x}_1 \cdot \bar{x}_3 \cdot (\bar{x}_1 \cdot x_3 + x_1 \cdot x_2) = \\ &= \bar{x}_1 \cdot \bar{x}_3 \cdot x_3 + \bar{x}_1 \cdot x_1 \cdot x_2 \cdot \bar{x}_3 = \bar{x}_1 \cdot 0 + 0 \cdot x_2 \cdot \bar{x}_3 \equiv 0, \end{aligned}$$

$$\text{dla } h_1 = x_1: \bar{x}_1 \cdot f = \bar{x}_1 \cdot (\bar{x}_1 \cdot x_3 + x_1 \cdot x_2) = \bar{x}_1 \cdot \bar{x}_3 \neq 0$$

$$\text{dla } h_2 = x_3: \bar{x}_3 \cdot f = \bar{x}_3 \cdot (\bar{x}_1 \cdot x_3 + x_1 \cdot x_2) = x_1 \cdot x_2 \cdot \bar{x}_3 \neq 0, \text{ czyli implicent prosty.}$$

Przykład: dla funkcji $f(x_1, x_2, x_3) = \bar{x}_1 \cdot x_3 + x_1 \cdot x_2$ pokazać w tabeli, że $g = x_2 \cdot x_3$ jest implikantem prostym tej funkcji; pokazać w tabeli, że $h = x_1 + x_3$ jest implicantem prostym tej funkcji.

x_1	x_2	x_3	f	$g = x_2 \cdot x_3$	$h = x_1 + x_3$
0	0	0	0	0	0
0	0	1	1	0	1
0	1	0	0	0	0
0	1	1	1	1	1
1	0	0	0	0	1
1	0	1	0	0	1
1	1	0	1	0	1
1	1	1	1	1	1

Wnioski z tabeli:

Jedynki funkcji $g = x_2 \cdot x_3$ pokrywają niektóre jedynki funkcji f (czerwone strzałki).

Uproszczona funkcja $g_1 = x_2$ nie zawsze wskazuje jedynki funkcji f .

Uproszczona funkcja $g_2 = x_3$ nie zawsze wskazuje jedynki funkcji f .

Funkcja $g = x_2 \cdot x_3$ jest zatem implikantem prostym funkcji f .

Zera funkcji $h = x_1 + x_3$ pokrywają niektóre zera funkcji f (niebieskie strzałki).

Uproszczona funkcja $h_1 = x_1$ nie zawsze wskazuje zera funkcji f .

Uproszczona funkcja $h_2 = x_3$ nie zawsze wskazuje zera funkcji f .

Funkcja $h = x_1 + x_3$ jest zatem implicantem prostym funkcji f .

Przykład: funkcję f doprowadzić do KPS i KPI; wskazać jej dowolne implikanty i implicenty.

$$\begin{aligned}
 f(x_1, x_2, x_3) &= \overline{(x_1 \oplus x_2)} \rightarrow \bar{x}_2 \cdot x_3 = (x_1 \oplus x_2) \cdot (\bar{x}_2 \cdot x_3) = (x_1 \oplus x_2) \cdot (\bar{x}_2 + \bar{x}_3) = \\
 &= (\bar{x}_1 \cdot x_2 + x_1 \cdot \bar{x}_2) \cdot (\bar{x}_2 + \bar{x}_3) = \bar{x}_1 \cdot x_2 \cdot \bar{x}_2 + \bar{x}_1 \cdot x_2 \cdot \bar{x}_3 + x_1 \cdot \bar{x}_2 \cdot \bar{x}_2 + x_1 \cdot \bar{x}_2 \cdot \bar{x}_3 = \\
 &= \bar{x}_1 \cdot x_2 \cdot (\bar{x}_3 + x_3) + \bar{x}_1 \cdot x_2 \cdot \bar{x}_3 + x_1 \cdot 0 + x_1 \cdot \bar{x}_2 \cdot \bar{x}_3 = \\
 &= \bar{x}_1 \cdot x_2 \cdot \bar{x}_3 + \bar{x}_1 \cdot x_2 \cdot x_3 + \bar{x}_1 \cdot x_2 \cdot \bar{x}_3 + x_1 \cdot \bar{x}_2 \cdot \bar{x}_3 = \sum(2,3,4) = \prod(0,1,5,6,7).
 \end{aligned}$$

Implikantami tej funkcji (nie koniecznie prostymi) są jej poszczególne składniki, np.:

$$g_1 = \bar{x}_1 \cdot x_2 \cdot \bar{x}_3, \quad g_2 = \bar{x}_1 \cdot x_2 \cdot x_3, \quad g_3 = x_1 \cdot \bar{x}_2 \cdot \bar{x}_3.$$

Implicantami tej funkcji (nie koniecznie prostymi) są jej poszczególne czynniki, np.:

$$h_1 = x_1 + x_2 + x_3, \quad h_2 = x_1 + x_2 + \bar{x}_3, \quad h_3 = \bar{x}_1 + x_2 + \bar{x}_3, \quad h_4 = \bar{x}_1 + \bar{x}_2 + x_3, \quad h_5 = \bar{x}_1 + \bar{x}_2 + x_3.$$

Minimalizacja funkcji logicznych – tablice Karnaugh

- wiersze i kolumny opisujemy kodami Gray’a
- stosujemy reguły sklejania (po jedynkach lub po zerach)

Przykłady: tablice Karnaugh dla funkcji o różnej liczbie zmiennych.

$x_1 \backslash x_2$	0	1
0	0	1
1	2	3

$x_1 \backslash x_2 x_3$	0 0	0 1	1 1	1 0
0	0	1	3	2
1	4	5	7	6

$x_1 x_2 \backslash x_3 x_4$	0 0	0 1	1 1	1 0
0 0	0	1	3	2
0 1	4	5	7	6
1 1	12	13	15	14
1 0	8	9	11	10

$x_1 x_2 \backslash x_3 x_4 x_5$	0 0 0	0 0 1	0 1 1	0 1 0	1 1 0	1 1 1	1 0 1	1 0 0
0 0	0	1	3	2	6	7	5	4
0 1	8	9	11	10	14	15	13	12
1 1	24	25	27	26	30	31	29	28
1 0	16	17	19	18	22	23	21	20

Przykłady: zminimalizować podane funkcje.

- Funkcja $f(x_1, x_2, x_3, x_4) = \sum(4, 8, 11, 13, (1, 5, 10, 12, 14))$ minimalizowana po *jedynkach*.

$x_3 \ x_4$ $x_1 \ x_2$	0 0	0 1	1 1	1 0
0 0		x		
0 1	1	x		
1 1	x	1		x
1 0	1		1	x

Wynik minimalizacji: $f(x_1, x_2, x_3) = x_2 \cdot \bar{x}_3 + x_1 \cdot \bar{x}_4 + x_1 \cdot \bar{x}_2 \cdot x_3$.

- Funkcja $f(x_1, x_2, x_3, x_4) = \sum(0, 2, 8, 11, (1, 3, 10))$ minimalizowana po *jedynkach*.

$x_3 \ x_4$ $x_1 \ x_2$	0 0	0 1	1 1	1 0
0 0	1	x	x	1
0 1				
1 1				
1 0	1		1	x

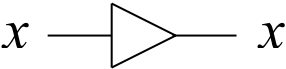
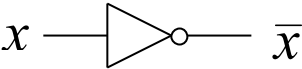
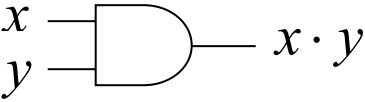
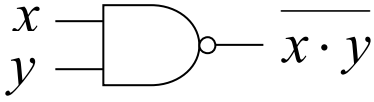
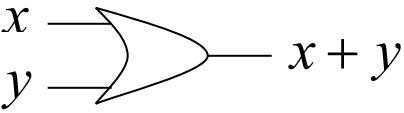
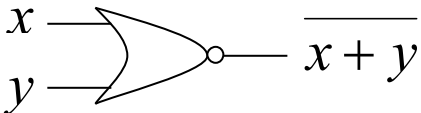
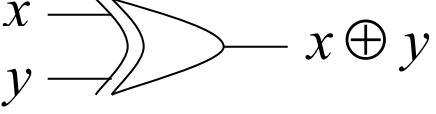
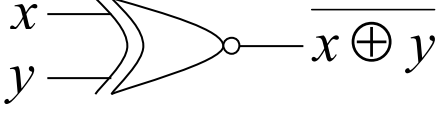
Wynik minimalizacji: $f(x_1, x_2, x_3, x_4) = \bar{x}_2 \cdot \bar{x}_4 + \bar{x}_2 \cdot x_3$.

- Funkcja $f(x_1, x_2, x_3, x_4) = \prod(0, 2, 6, 7, 9, 15, (1, 3, 5, 10, 12, 14))$ minimalizowana po *zerach*.

$x_3 \ x_4$ $x_1 \ x_2$	0 0	0 1	1 1	1 0
0 0	0	x	x	0
0 1		x	0	0
1 1	x		0	x
1 0		0		x

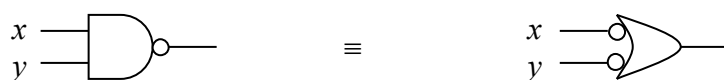
Wynik minimalizacji: $f(x_1, x_2, x_3) = (\bar{x}_2 + \bar{x}_3) \cdot (x_1 + x_2) \cdot (x_2 + x_3 + \bar{x}_4)$.

Bramki logiczne

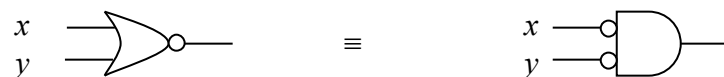
<p style="text-align: center;">BUF</p> 	<p style="text-align: center;">NOT</p> 
<p style="text-align: center;">AND</p> 	<p style="text-align: center;">NAND</p> 
<p style="text-align: center;">OR</p> 	<p style="text-align: center;">NOR</p> 
<p style="text-align: center;">XOR</p> 	<p style="text-align: center;">NXOR</p> 

Ważne:

- negacja iloczynu (NAND) to suma zanegowanych sygnałów: $\overline{x \cdot y} = \bar{x} + \bar{y}$,



- negacja sumy (NOR) to iloczyn zanegowanych sygnałów: $\overline{x + y} = \bar{x} \cdot \bar{y}$,



- negacja sumy modulo 2 (XOR) to równoważność sygnałów: $\overline{x \oplus y} = (x \equiv y)$,



- bramka BUF jest jedynie wzmacniaczem (buforem) i nie zmienia wartości logicznej sygnału.

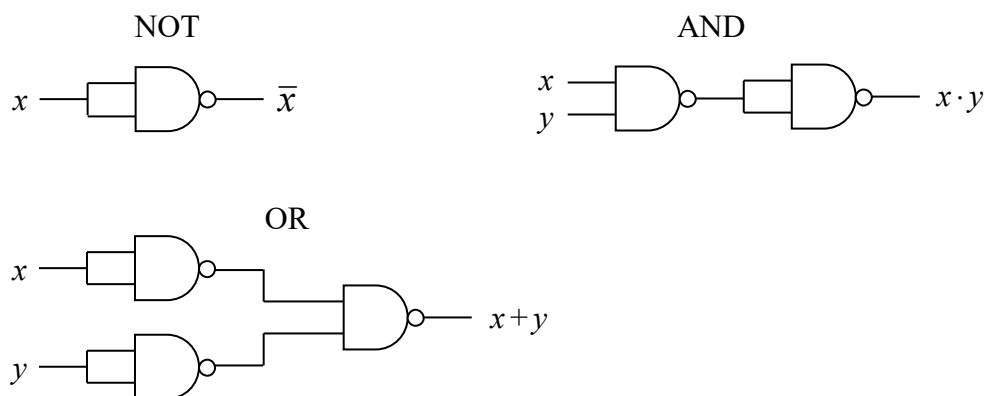
System funkcjonalnie pełny (SFP): system, w którym można zbudować każdą funkcję logiczną.

Różne SFP:

- system realizujący operacje: **AND**, **OR**, **NOT**,
- system realizujący operację **NAND**,
- system realizujący operację **NOR**,
- system realizujący operację **IF** oraz stała logiczna **falsz** ("0").

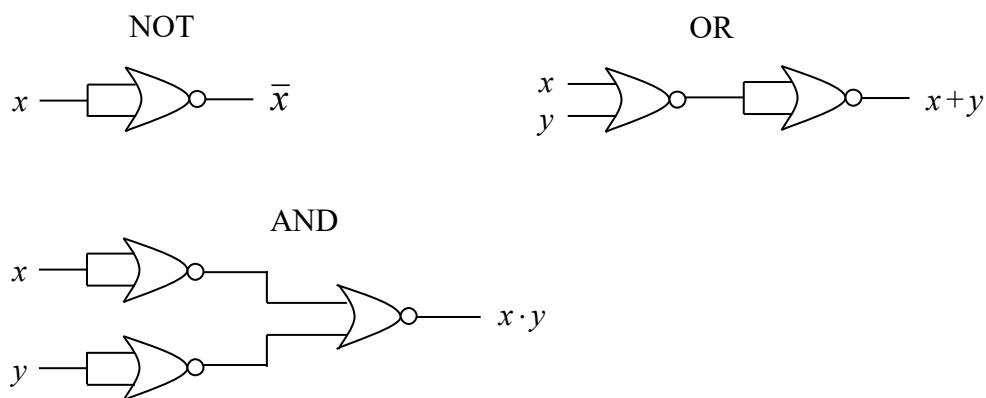
Przykład: wykazać, że system NAND jest funkcjonalnie pełny.

Wystarczy pokazać, że za pomocą bramek NAND można wykonać operacje NOT, AND i OR.



Przykład: wykazać, że system NOR jest funkcjonalnie pełny.

Wystarczy pokazać, że za pomocą bramek NOR można wykonać operacje NOT, OR i AND.



Przykład: wykazać, że system IF ze stałą logiczną "0" jest funkcjonalnie pełny.

Wykorzystamy zależności: $(x \rightarrow y) = \bar{x} + y$ oraz $\overline{(x \rightarrow y)} = x \cdot \bar{y}$.

- realizacja NOT: $(x \rightarrow 0) = \bar{x} + 0 = \bar{x}$,
- realizacja OR: $((x \rightarrow 0) \rightarrow y) = (\bar{x} \rightarrow y) = \bar{\bar{x}} + y = x + y$,
- realizacja AND: $((x \rightarrow (y \rightarrow 0)) \rightarrow 0) = \overline{(x \rightarrow \bar{y})} = x \cdot \bar{\bar{y}} = x \cdot y$.

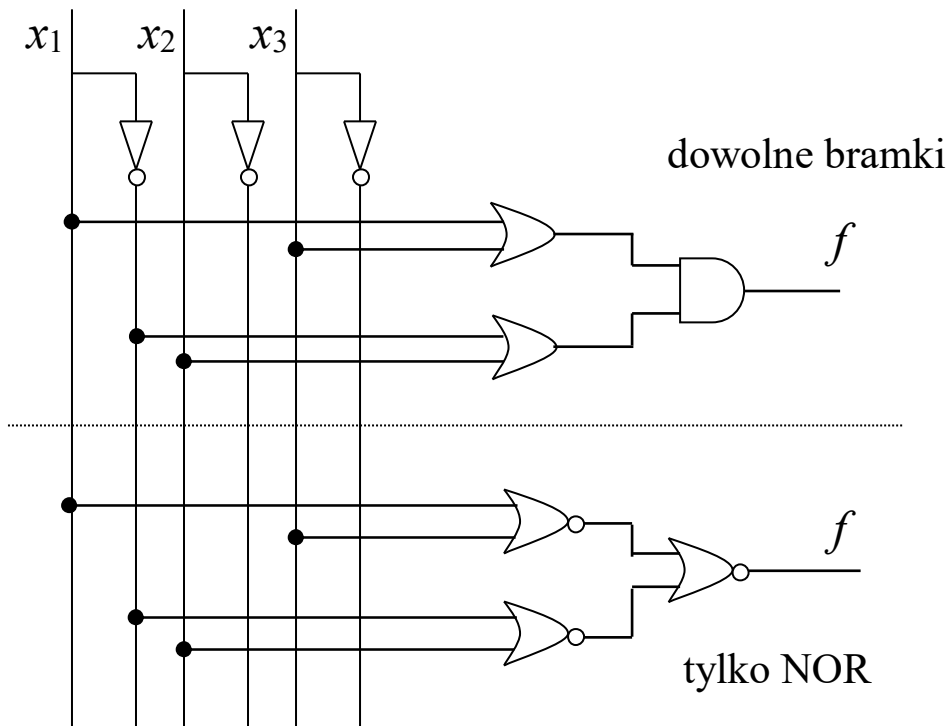
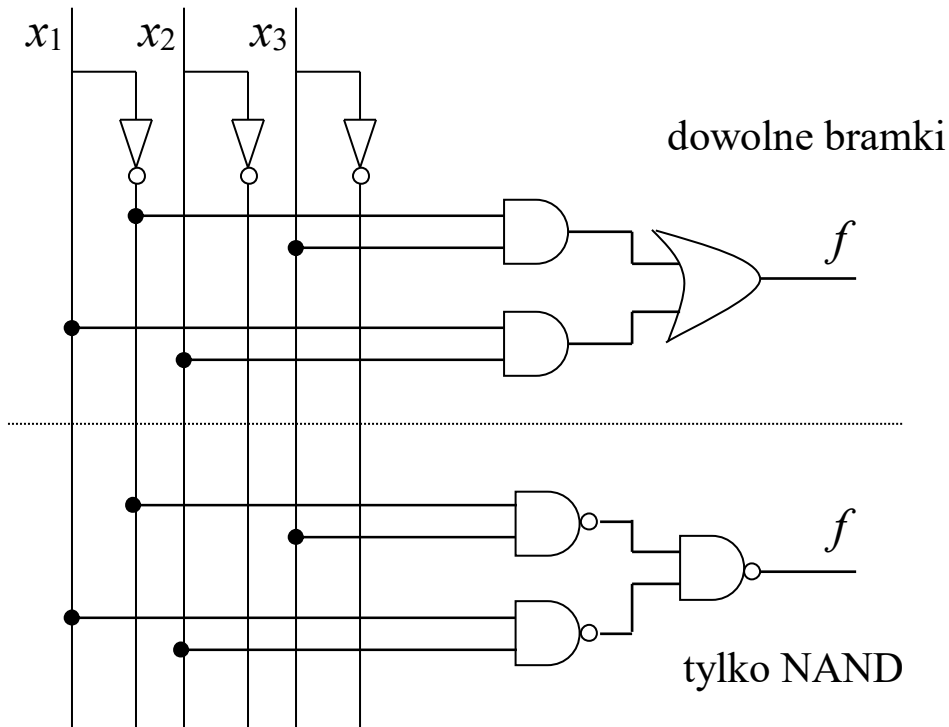
Realizacje kanoniczne funkcji logicznych

Jako ilustracja pokazana zostanie realizacja funkcji $f(x_1, x_2, x_3) = \sum(1, 3, 6, 7) = \prod(0, 2, 4, 5)$.

$x_1 \backslash x_2 x_3$	0 0	0 1	1 1	1 0
0	0	1	1	0
1	0	0	1	1

Minimalizacja po *jedynkach* (linie czerwone): $f(x_1, x_2, x_3) = \bar{x}_1 \cdot x_3 + x_1 \cdot x_2$.

Minimalizacja po *zerach* (linie niebieskie): $f(x_1, x_2, x_3) = (x_1 + x_3) \cdot (\bar{x}_1 + x_2)$.



Działanie poszczególnych sieci bramek:

- Sieć (2×AND i OR): $\bar{x}_1 \cdot x_3 + x_1 \cdot x_2 \equiv f$
- Sieć (3×NAND): $\overline{((\bar{x}_1 \cdot x_3) \cdot (x_1 \cdot x_2))} = \overline{(\bar{x}_1 \cdot x_3)} + \overline{(x_1 \cdot x_2)} = \bar{x}_1 \cdot x_3 + x_1 \cdot x_2 \equiv f$
- Sieć (2×OR i AND): $(x_1 + x_3) \cdot (\bar{x}_1 + x_2) \equiv f$
- Sieć (3×NOR): $\overline{((x_1 + x_3) + (\bar{x}_1 + x_2))} = \overline{(x_1 + x_3)} \cdot \overline{(\bar{x}_1 + x_2)} = (x_1 + x_3) \cdot (\bar{x}_1 + x_2) \equiv f$

Analiza schematów

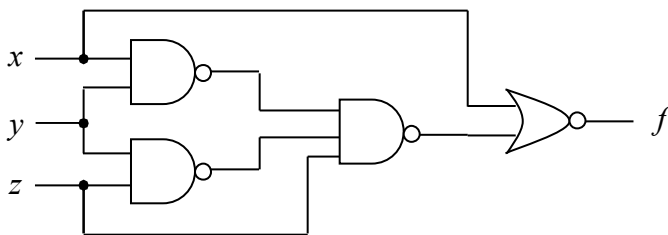
Dla sieci zawierającej bramki NAND i NOR tworzymy schemat zastępczy wg podanych reguł:

- licząc od strony wyjścia: nieparzysty (pierwszy, trzeci, ...) NAND zastępujemy bramką OR,
- licząc od strony wyjścia: nieparzysty (pierwszy, trzeci, ...) NOR zastępujemy bramką AND,
- licząc od strony wyjścia: parzysty (drugi, czwarty, ...) NAND zastępujemy bramką AND,
- licząc od strony wyjścia: parzysty (drugi, czwarty, ...) NOR zastępujemy bramką OR,
- licząc od strony wyjścia: zmienna dochodząca do nieparzystej bramki będzie zanegowana,
- licząc od strony wyjścia: zmienna dochodząca do parzystej bramki nie będzie modyfikowana.

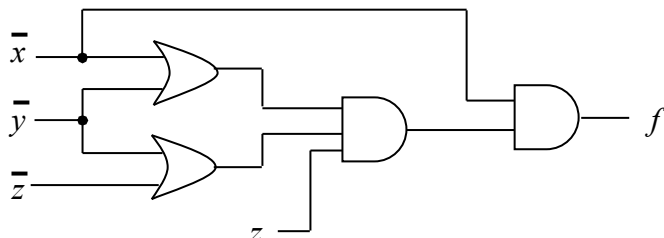
Podany sposób postępowania można zestawić w tabeli:

licząc od strony wyjścia:	bramka nieparzysta	bramka parzysta
NAND	zastąp bramką OR	zastąp bramką AND
NOR	zastąp bramką AND	zastąp bramką OR
zmienna	zastąp zmienną zanegowaną	pozostaw zmienną

Przykład: narysować schemat zastępczy podanej sieci i podać postać funkcji $f(x, y, z)$.

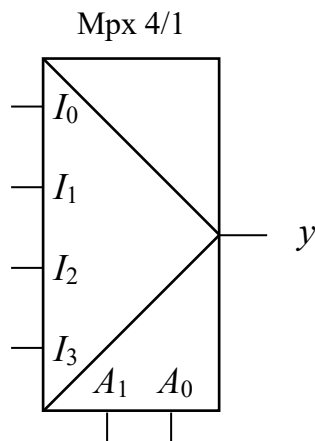
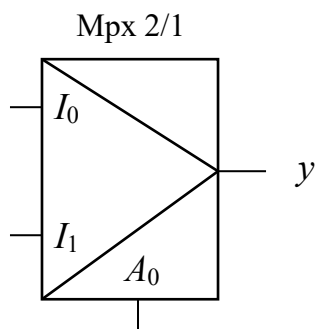


Stosując podane reguły dostaje się sieć zastępczą.



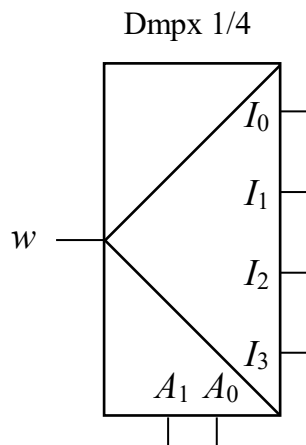
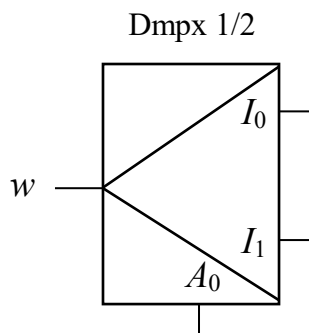
Stąd funkcja: $f(x, y, z) = (\bar{x} + \bar{y}) \cdot (\bar{y} + \bar{z}) \cdot z \cdot \bar{x} = (\bar{x} \cdot \bar{y} + \bar{x} \cdot \bar{z} + \bar{y} \cdot \bar{y} + \bar{y} \cdot \bar{z}) \cdot z \cdot \bar{x} = \bar{x} \cdot \bar{y} \cdot z$.

Multipleksery (2/1 , 4/1 , 8/1 , 16/1 , ...)



działanie Mpx 2/1	działanie Mpx 4/1
$y = \begin{cases} I_0 & \text{dla } A_0 = 0 \\ I_1 & \text{dla } A_0 = 1 \end{cases}$	$y = \begin{cases} I_0 & \text{dla } A_1 A_0 = 00 \\ I_1 & \text{dla } A_1 A_0 = 01 \\ I_2 & \text{dla } A_1 A_0 = 10 \\ I_3 & \text{dla } A_1 A_0 = 11 \end{cases}$
$y = I_0 \cdot \bar{A}_0 + I_1 \cdot A_0$	$y = I_0 \cdot \bar{A}_1 \cdot \bar{A}_0 + I_1 \cdot \bar{A}_1 \cdot A_0 + I_2 \cdot A_1 \cdot \bar{A}_0 + I_3 \cdot A_1 \cdot A_0$
$y = (I_0 + A_0) \cdot (I_1 + \bar{A}_0)$	$y = (I_0 + A_1 + A_0) \cdot (I_1 + A_1 + \bar{A}_0) \cdot (I_2 + \bar{A}_1 + A_0) \cdot (I_3 + \bar{A}_1 + \bar{A}_0)$

Demultipleksery (1/2 , 1/4 , 1/8 , 1/16 , ...)



działanie Dmpx 1/2	działanie Dmpx 1/4
$\begin{cases} I_0 = w & \text{dla } A_0 = 0 \\ I_1 = w & \text{dla } A_0 = 1 \end{cases}$	$\begin{cases} I_0 = w & \text{dla } A_1 A_0 = 00 \\ I_1 = w & \text{dla } A_1 A_0 = 01 \\ I_2 = w & \text{dla } A_1 A_0 = 10 \\ I_3 = w & \text{dla } A_1 A_0 = 11 \end{cases}$

Realizacja funkcji logicznych na multiplexerach

Jako ilustracja pokazana zostanie realizacja funkcji $f(a,b,c,d) = \sum (0,1,2,8,9,13)$.

$a \backslash b \backslash c \ d$	0 0	0 1	1 1	1 0
0 0	1	1		1
0 1				
1 1		1		
1 0	1	1		

minimalizacja po *jedynkach*:

$$f = \bar{b}\bar{c} + a\bar{c}d + \bar{a}\bar{b}\bar{d}$$

• Rozwiązanie z Mpx 4/1 (zmiennne adresowe: $A_1 = a$, $A_0 = b$)

Multiplexer realizuje funkcję f , zatem: $I_0\bar{a}\bar{b} + I_1\bar{a}b + I_2a\bar{b} + I_3ab = \bar{b}\bar{c} + a\bar{c}d + \bar{a}\bar{b}\bar{d}$.

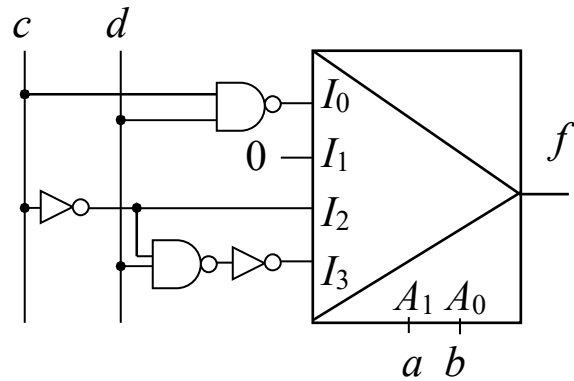
Podstawiając do obu stron równania różne wartości zmiennych a i b otrzymujemy:

– dla $a=0, b=0$: $I_0 = f(0,0,c,d) = \bar{c} + \bar{d}$

– dla $a=0, b=1$: $I_1 = f(0,1,c,d) = 0$

– dla $a=1, b=0$: $I_2 = f(1,0,c,d) = \bar{c}$

– dla $a=1, b=1$: $I_3 = f(1,1,c,d) = \bar{c} \cdot d$



• Rozwiązanie z Mpx 8/1 (zmiennne adresowe: $A_2 = a$, $A_1 = b$, $A_0 = c$)

Zachodzi: $I_0\bar{a}\bar{b}\bar{c} + I_1\bar{a}\bar{b}c + I_2\bar{a}b\bar{c} + I_3\bar{a}bc + I_4a\bar{b}\bar{c} + I_5a\bar{b}c + I_6ab\bar{c} + I_7abc = \bar{b}\bar{c} + a\bar{c}d + \bar{a}\bar{b}\bar{d}$.

Podstawiając do obu stron równania różne wartości zmiennych a , b i c otrzymujemy:

– dla $a=0, b=0, c=0$: $I_0 = f(0,0,0,d) = 1$

– dla $a=0, b=0, c=1$: $I_1 = f(0,0,1,d) = \bar{d}$

– dla $a=0, b=1, c=0$: $I_2 = f(0,1,0,d) = 0$

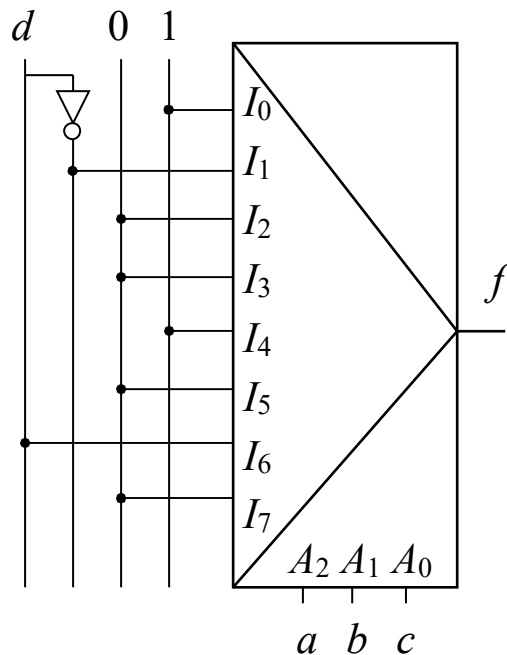
– dla $a=0, b=1, c=1$: $I_3 = f(0,1,1,d) = 0$

– dla $a=1, b=0, c=0$: $I_4 = f(1,0,0,d) = 1$

– dla $a=1, b=0, c=1$: $I_5 = f(1,0,1,d) = 0$

– dla $a=1, b=1, c=0$: $I_6 = f(1,1,0,d) = d$

– dla $a=1, b=1, c=1$: $I_7 = f(1,1,1,d) = 0$

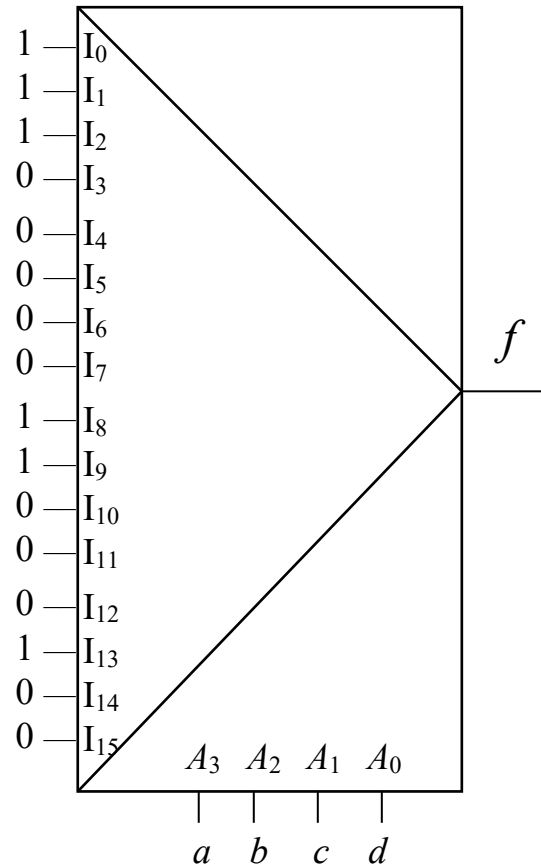


• **Rozwiązanie z Mpx 16/1** (zmienne adresowe: $A_3 = a$, $A_2 = b$, $A_1 = c$, $A_0 = d$)

$$\begin{aligned} \text{Zachodzi: } \bar{b}\bar{c} + a\bar{c}d + \bar{a}\bar{b}d &= I_0\bar{a}\bar{b}\bar{c}\bar{d} + I_1\bar{a}\bar{b}\bar{c}d + I_2\bar{a}\bar{b}c\bar{d} + I_3\bar{a}\bar{b}cd + \\ &+ I_4\bar{a}b\bar{c}\bar{d} + I_5\bar{a}b\bar{c}d + I_6\bar{a}bcd + I_7\bar{a}bcd + \\ &+ I_8a\bar{b}\bar{c}\bar{d} + I_9a\bar{b}\bar{c}d + I_{10}a\bar{b}c\bar{d} + I_{11}a\bar{b}cd + \\ &+ I_{12}ab\bar{c}\bar{d} + I_{13}ab\bar{c}d + I_{14}abcd + I_{15}abcd. \end{aligned}$$

Podstawiając do obu stron równania różne wartości zmiennych a , b , c i d otrzymujemy:

- dla $a=0, b=0, c=0, d=0$: $I_0 = f(0,0,0,0) = 1$
- dla $a=0, b=0, c=0, d=1$: $I_1 = f(0,0,0,1) = 1$
- dla $a=0, b=0, c=1, d=0$: $I_2 = f(0,0,1,0) = 1$
- dla $a=0, b=0, c=1, d=1$: $I_3 = f(0,0,1,1) = 0$
- dla $a=0, b=1, c=0, d=0$: $I_4 = f(0,1,0,0) = 0$
- dla $a=0, b=1, c=0, d=1$: $I_5 = f(0,1,0,1) = 0$
- dla $a=0, b=1, c=1, d=0$: $I_6 = f(0,1,1,0) = 0$
- dla $a=0, b=1, c=1, d=1$: $I_7 = f(0,1,1,1) = 0$
- dla $a=1, b=0, c=0, d=0$: $I_8 = f(1,0,0,0) = 1$
- dla $a=1, b=0, c=0, d=1$: $I_9 = f(1,0,0,1) = 1$
- dla $a=1, b=0, c=1, d=0$: $I_{10} = f(1,0,1,0) = 0$
- dla $a=1, b=0, c=1, d=1$: $I_{11} = f(1,0,1,1) = 0$
- dla $a=1, b=1, c=0, d=0$: $I_{12} = f(1,1,0,0) = 0$
- dla $a=1, b=1, c=0, d=1$: $I_{13} = f(1,1,0,1) = 1$
- dla $a=1, b=1, c=1, d=0$: $I_{14} = f(1,1,1,0) = 0$
- dla $a=1, b=1, c=1, d=1$: $I_{15} = f(1,1,1,1) = 0$



Wniosek: ponieważ $f(a,b,c,d) = \sum (0,1,2,8,9,13)$ i adresujemy $A_3 = a$, $A_2 = b$, $A_1 = c$, $A_0 = d$, wystarczy do wejść I_n ($n=0,1,2,8,9,13$) doprowadzić sygnał "1", a do pozostałych wejść I_m ($m=3,4,5,6,7,10,11,12,14,15$) doprowadzić sygnał "0".

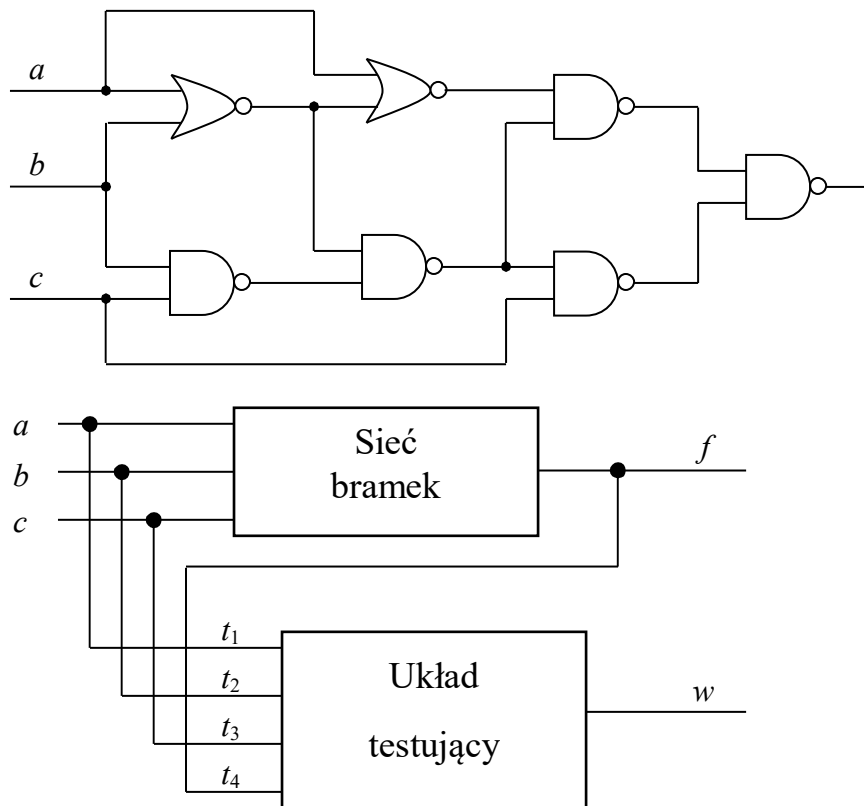
Zadania:

- Zbudować Mpx 4/1 na bramkach NAND.
- Zbudować Dmpx 4/1 na bramkach NAND.
- Z kilku Mpx 4/1 zbudować Mpx 16/1.
- Z kilku Dmpx 4/1 zbudować Dmpx 16/1.
- Funkcję $f(a,b,c,d) = \sum (1,3,6,7,8,(5,10,11,13,15))$ zrealizować na Mpx 4/1.

Układy kombinacyjne

Zadanie: zbudować układ kombinacyjny testujący podaną sieć logiczną:

- podać funkcję $f(a, b, c)$ realizowaną przez sieć logiczną,
- podać tabelę projektowanego układu testującego,
- układ zrealizować na Mpx 4/1 i bramkach NAND.



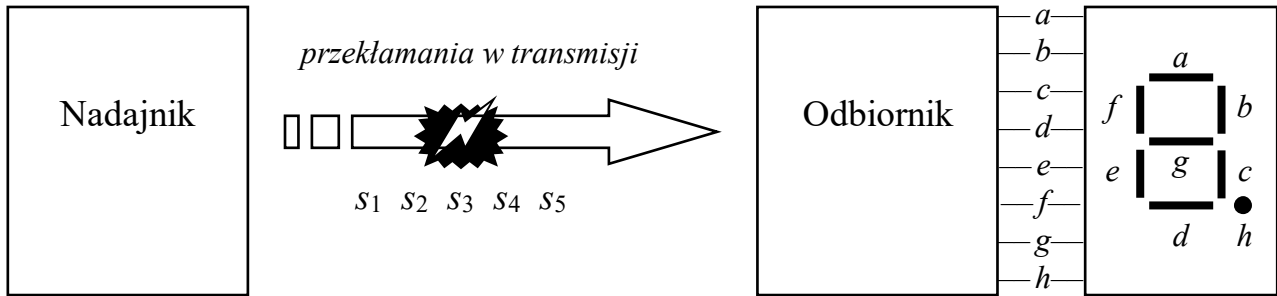
Rozwiązanie: należy stworzyć schemat zastępczy sieci. Ze schematu łatwo odczytać funkcję:

$f(a, b, c, d) = \bar{a} \cdot b + a \cdot c + b \cdot c = \sum(2, 3, 5, 7)$. Stąd wynika natychmiast tabela układu testującego.

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

t_1	t_2	t_3	t_4	w
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	0
0	1	0	0	0
0	1	0	1	1
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	1
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

Zadanie: zbudować układ dekodowania czterech znaków (A,b,C,d) z możliwością poprawiania pojedynczych błędów; rozpoznane znaki eksponować na wyświetlaczu.



Protokół transmisji	S_1	S_2	S_3	S_4	S_5	a	b	c	d	e	f	g	h
(kodowanie nadmiarowe)	A	0	0	0	0	0	0	0	0	0	0	0	0
	b	0	1	1	1	0	0	0	1	1	1	1	0
	C	1	0	1	1	1	1	0	0	1	1	1	0
	d	1	1	0	0	1	0	1	1	1	1	0	1
–	błąd gruby					0	0	0	0	0	0	1	0
.	dioda h zapalana dodatkowo, gdy poprawiono błąd					0	0	0	0	0	0	0	1

Rozwiązanie: kod A = 00000 oraz pojedyncze błędy A. = {00001, 00010, 00100, 01000, 10000},
kod b = 01110 oraz pojedyncze błędy b. = {01111, 01100, 01010, 00110, 11110},
kod C = 10111 oraz pojedyncze błędy C. = {10110, 10101, 10011, 11111, 00111},
kod d = 11001 oraz pojedyncze błędy d. = {11000, 11011, 11101, 10001, 01001}.

Stąd natychmiast wynika tabela układu dekodującego znaki z możliwością korekty błędów.

S_1	S_2	S_3	S_4	S_5	znak
0	0	0	0	0	A
0	0	0	0	1	A.
0	0	0	1	0	A.
0	0	0	1	1	–
0	0	1	0	0	A.
0	0	1	0	1	–
0	0	1	1	0	b.
0	0	1	1	1	C.
0	1	0	0	0	A.
0	1	0	0	1	d.
0	1	0	1	0	b.
0	1	0	1	1	–
0	1	1	0	0	b.
0	1	1	0	1	–
0	1	1	1	0	b
0	1	1	1	1	b.

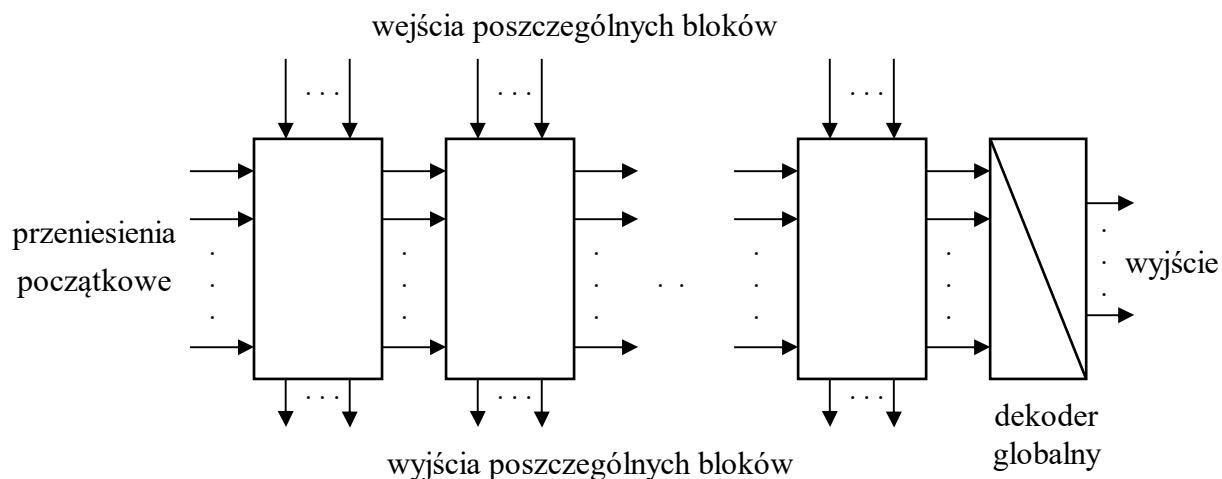
S_1	S_2	S_3	S_4	S_5	znak
1	0	0	0	0	A.
1	0	0	0	1	d.
1	0	0	1	0	–
1	0	0	1	1	C.
1	0	1	0	0	–
1	0	1	0	1	C.
1	0	1	1	0	C.
1	0	1	1	1	C
1	1	0	0	0	d.
1	1	0	0	1	d
1	1	0	1	0	–
1	1	0	1	1	d.
1	1	1	0	0	–
1	1	1	0	1	d.
1	1	1	1	0	b.
1	1	1	1	1	C.

Układy iteracyjne – projektowanie układów kombinacyjnych opisanych rekurencyjnie

Etapy projektu:

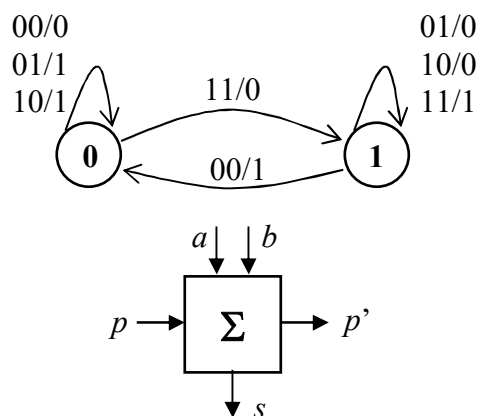
- podanie tabeli (lub grafu) pojedynczego bloku,
- minimalizacja funkcji przeniesień i wyjść,
- realizacja bloku na bramkach lub multiplekserach,
- określenie przeniesień początkowych.

Układ iteracyjny ma postać kaskady identycznych bloków:



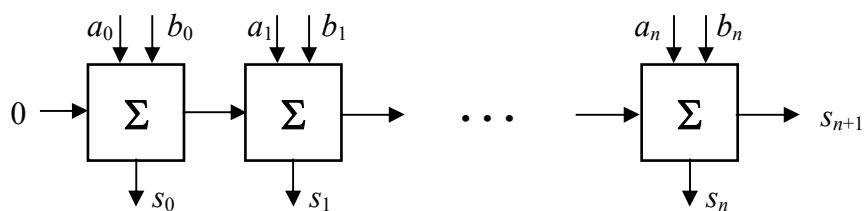
Zadanie: zbudować iteracyjny sumator liczb binarnych.

p	a	b	p'	s
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1



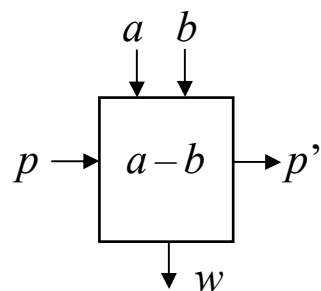
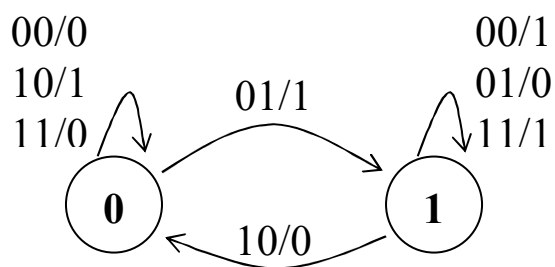
Synteza pojedynczego bloku: najłatwiej na dwóch Mpx 8/1 (adresowanie $A_2=p$, $A_1=a$, $A_0=b$):

- realizacja bitu sumy s : $I_0=I_3=I_5=I_6=0$, $I_1=I_2=I_4=I_7=1$,
- realizacja przeniesienia p' : $I_0=I_1=I_2=I_4=0$, $I_3=I_5=I_6=I_7=1$,
- przeniesienie początkowe: $p=0$ (na pierwszym bloku).

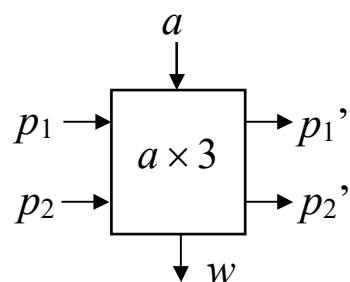
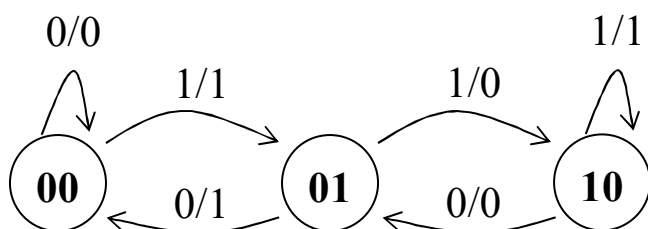


Przykłady: układy iteracyjne wykonujące różne operacje arytmetyczne i logiczne:

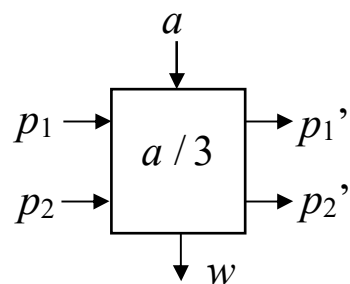
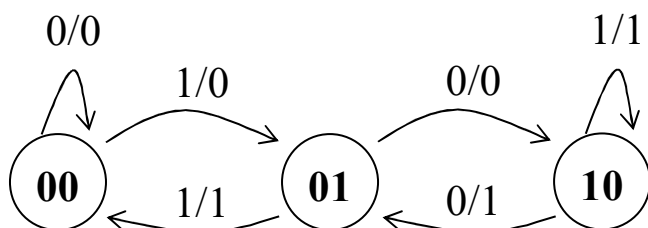
a) układ iteracyjny odejmujący dwie liczby binarne ($a - b$)



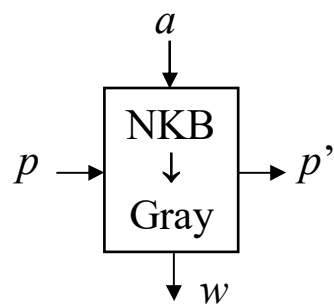
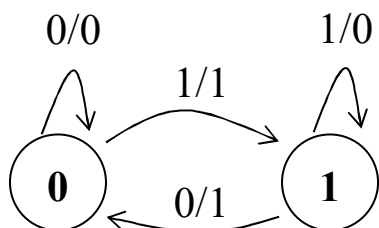
b) układ iteracyjny mnożący liczbę binarną przez 3 ($a \times 3$)



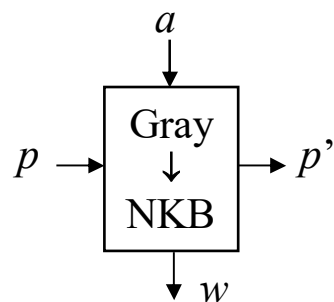
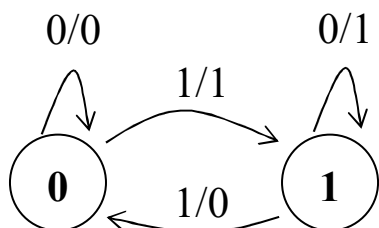
c) układ iteracyjny dzielący liczbę binarną przez 3 ($a / 3$)



d) układ iteracyjny zamieniający NKB na Gray'a (NKB \rightarrow Gray)

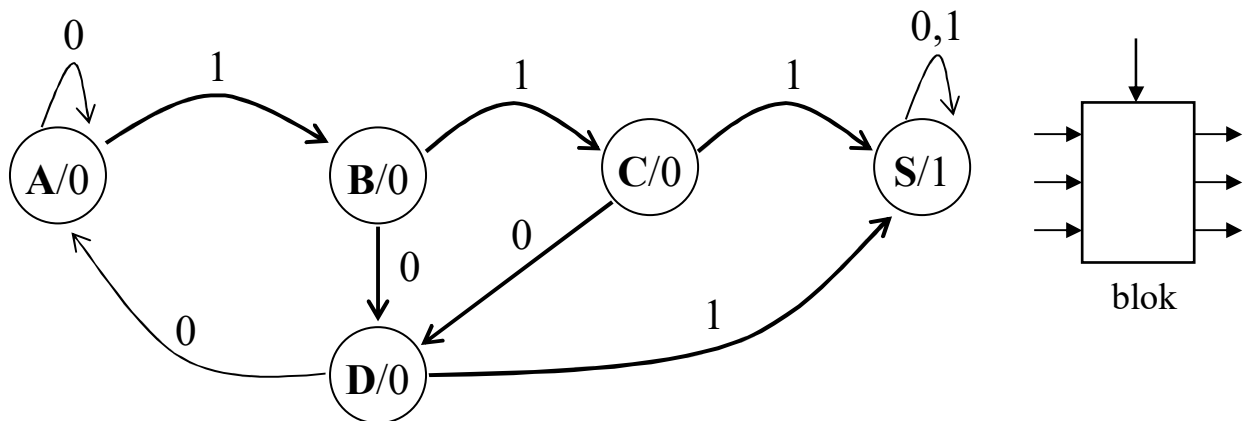


e) układ iteracyjny zamieniający Gray'a na NKB (Gray \rightarrow NKB)



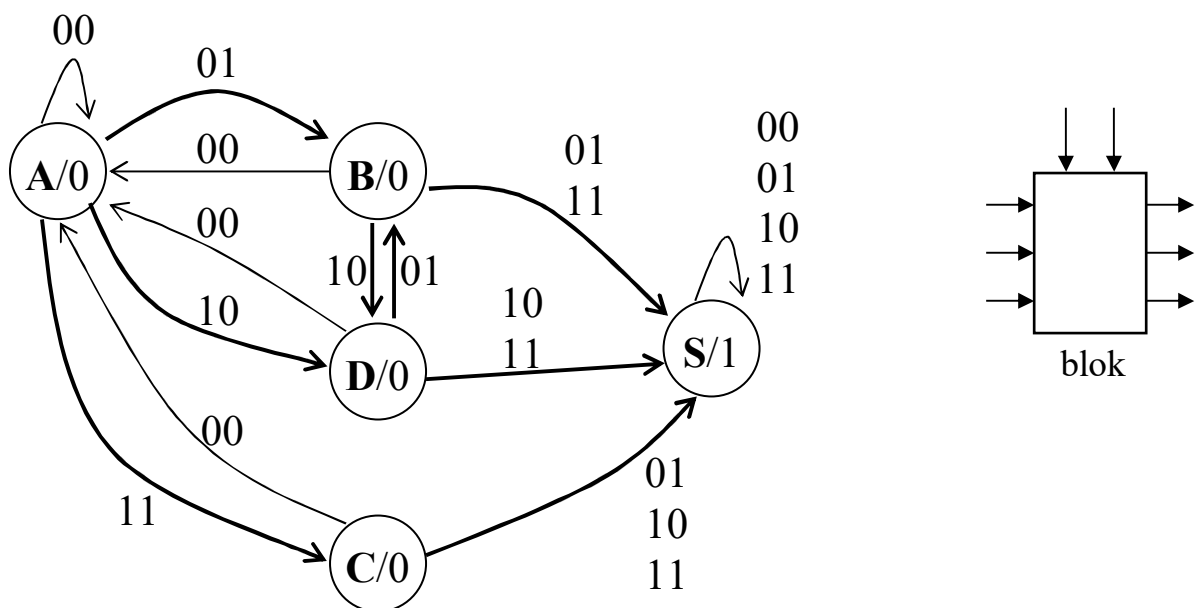
Zadanie: zaprojektować układ iteracyjny wykrywający w n -bitowym słowie wejściowym sekwencję kolejnych bitów 1×1 ; po wykryciu sekwencji wyjście dekodera globalnego zostaje ustawione na stałe w stan wysoki.

Rozwiązanie pierwsze – jeden bit na każdy blok (łącznie n bloków):



Stany w grafie: A – nie znaleziono początku sekwencji (...0),
 B – znaleziono pierwszy element (...1),
 C – znaleziono dwa elementy (...11),
 D – znaleziono dwa elementy (...10),
 S – sukces, sekwencja znaleziona (...111 lub ...101).

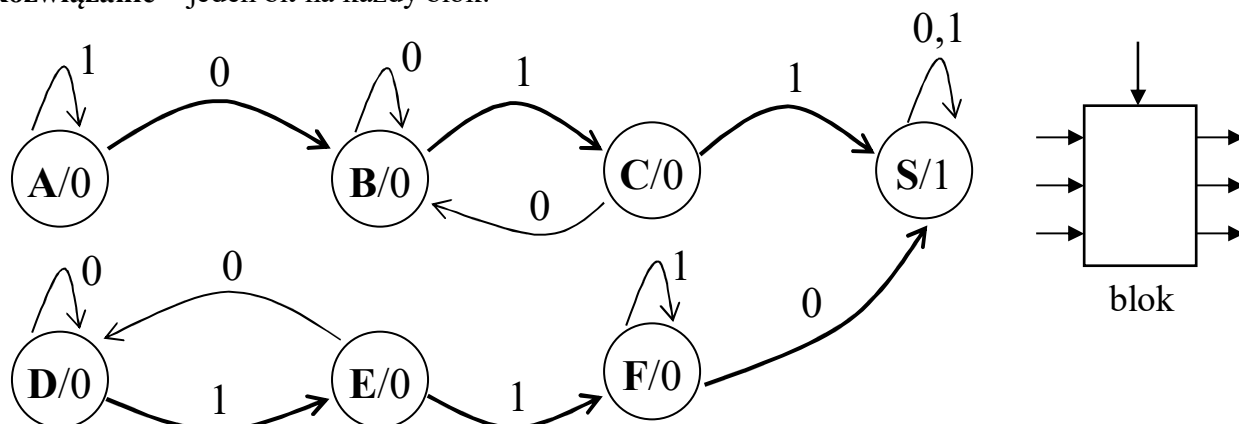
Rozwiązanie drugie – dwa kolejne bity na każdy blok (łącznie $n/2$ bloków):



Stany w grafie: A – nie znaleziono początku sekwencji (...0),
 B – znaleziono pierwszy element (...1),
 C – znaleziono dwa elementy (...11),
 D – znaleziono dwa elementy (...10),
 S – sukces, sekwencja znaleziona (...111 lub ...101).

Zadanie: zbudować układ iteracyjny wykrywający w n -bitowym słowie wejściowym sekwencje:
0 1 1 gdy przeniesienia początkowe są równe 0 0 0,
1 1 0 gdy przeniesienia początkowe są równe 1 1 1.
 Po wykryciu wyjście dekodera globalnego pozostaje na stałe w stanie wysokim.

Rozwiązanie – jeden bit na każdy blok:



Stany w grafie: **A** – nie znaleziono początku sekwencji **011**,
B – znaleziono pierwszy element sekwencji **011**,
C – znaleziono dwa elementy sekwencji **011**,
D – nie znaleziono początku sekwencji **110**,
E – znaleziono pierwszy element sekwencji **110**,
F – znaleziono dwa elementy sekwencji **110**,
S – sukces, znaleziono sekwencję **011** lub **110**.

Kodowanie stanów: **A** = 0 0 0 (obowiązkowo)

B = 0 0 1

C = 0 1 0

D = 1 1 1 (obowiązkowo)

E = 1 1 0

F = 1 0 1

S = 1 0 0

p_1	p_2	p_3	x	p_1'	p_2'	p_3'
0	0	0	0	0	0	1
0	0	0	1	0	0	0
0	0	1	0	0	0	1
0	0	1	1	0	1	0
0	1	0	0	0	0	1
0	1	0	1	1	0	0
0	1	1	0	-	-	-
0	1	1	1	-	-	-
1	0	0	0	1	0	0
1	0	0	1	1	0	0
1	0	1	0	1	0	0
1	0	1	1	1	0	1
1	1	0	0	1	1	1
1	1	0	1	1	0	1
1	1	1	0	1	1	1
1	1	1	1	1	1	0

p_1	p_2	p_3	y
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	-
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

Realizacja: 3 x Mpx 16/1 i Mpx 8/1

p_1' ($A_3=p_1, A_2=p_2, A_1=p_3, A_0=x$)

$I_0=I_1=I_2=I_3=I_4=0$,
 $I_5=I_8=I_9=I_{10}=1, I_6=I_7=$ dowolne,
 $I_{11}=I_{12}=I_{13}=I_{14}=I_{15}=1$.

p_2' ($A_3=p_1, A_2=p_2, A_1=p_3, A_0=x$)

$I_0=I_1=I_2=0, I_6=I_7=$ dowolne,
 $I_4=I_5=I_8=I_9=I_{10}=I_{11}=I_{13}=0$,
 $I_3=I_{12}=I_{14}=I_{15}=1$.

p_3' ($A_3=p_1, A_2=p_2, A_1=p_3, A_0=x$)

$I_1=I_3=I_5=0, I_6=I_7=$ dowolne,
 $I_8=I_9=I_{10}=I_{15}=0$,
 $I_0=I_2=I_4=I_{11}=I_{12}=I_{13}=I_{14}=1$.

y ($A_3=p_1, A_2=p_2, A_1=p_3, A_0=x$)

$I_0=I_1=I_2=I_5=I_6=I_7=0$,
 $I_4=1, I_3=$ dowolne.

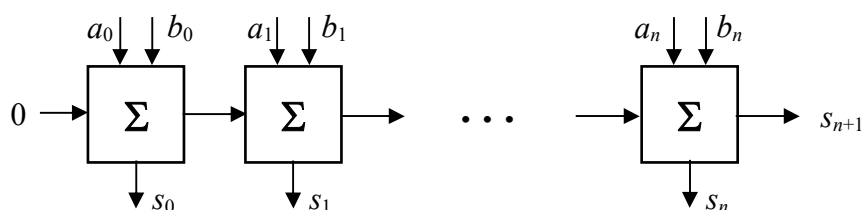
Układy sekwencyjne – synchroniczne

- układ sekwencyjny, czyli zawierający stany (pamięć),
- układ synchroniczny, czyli zmiany stanu dokonują się w takt sygnału zegarowego,
- układ opisuje się za pomocą grafu lub tabeli przejść stanów (modele: Moore’a i Mealy’ego).

Związek między układami iteracyjnymi i synchronicznymi

Sumator iteracyjny – dane podawane równolegle:

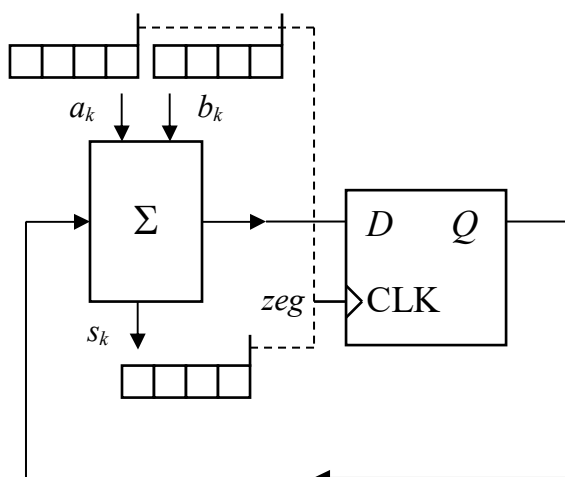
- kaskada identycznych bloków,
- przeniesienie początkowe $p=0$ (na pierwszym bloku).



p	a	b	p'	s
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Sumator synchroniczny (dane podawane szeregowo):

- sumowane liczby i wynik dodawania zawarte w rejestrach przesuwanych,
- przeniesienie wychodzące podawane na wejście sumatora przez przerzutnik D.



przerzutnik D

Q	Q'	D
0	0	0
0	1	1
1	0	0
1	1	1

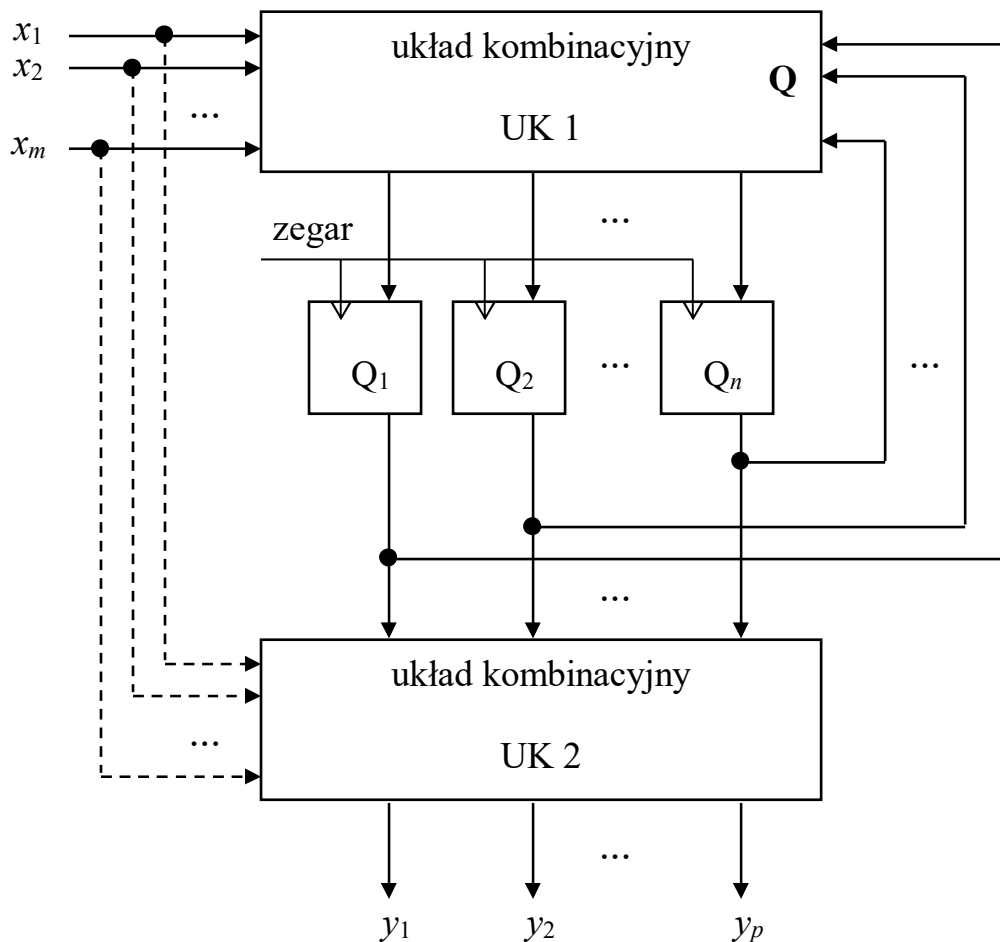
Ważne:

- zadania z układów iteracyjnych i układów synchronicznych mogą być identyczne,
- w projekcie identyczny jest graf oraz tabela (odpowiednio przeniesień lub przejść stanów),
- przeniesienia początkowe w ukł. iteracyjnym to stan początkowy w ukł. synchronicznym.

Porównanie układów iteracyjnych i synchronicznych:

	układy iteracyjne	układy synchroniczne
podawanie danych:	równolegle	szeregowo
szybkość działania:	$n \times$ czas propagacji bloku	$n \times$ okres zegara
wartości inicjalne:	przeniesienia początkowe	początkowy stan przerzutników

Układy sekwencyjne synchroniczne – schemat blokowy



Oznaczenia:

wejsie: $\mathbf{x} = x_1, x_2, \dots, x_m$

stan: $\mathbf{Q} = Q_1, Q_2, \dots, Q_n$

wyjsie: $\mathbf{y} = y_1, y_2, \dots, y_p$

Modele układów sekwencyjnych:

• układ Moore'a:

- stan następny (\mathbf{Q}^{k+1}) zależy od stanu bieżącego (\mathbf{Q}^k) i wejścia bieżącego (\mathbf{x}^k),
- wyjście (\mathbf{y}^{k+1}) zależy tylko od stanu (\mathbf{Q}^{k+1}),

• układ Mealy'ego:

- stan następny (\mathbf{Q}^{k+1}) zależy od stanu bieżącego (\mathbf{Q}^k) i wejścia bieżącego (\mathbf{x}^k),
- wyjście (\mathbf{y}^{k+1}) zależy od stanu (\mathbf{Q}^{k+1}) i wejścia (\mathbf{x}^{k+1}).

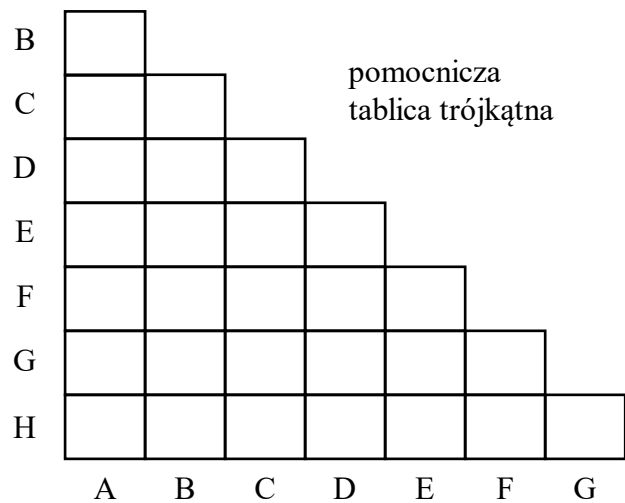
układ Moore'a	układ Mealy'ego
$\mathbf{Q}^{k+1} = f(\mathbf{Q}^k, \mathbf{x}^k)$ $\mathbf{y}^{k+1} = g(\mathbf{Q}^{k+1})$	$\mathbf{Q}^{k+1} = f(\mathbf{Q}^k, \mathbf{x}^k)$ $\mathbf{y}^{k+1} = g(\mathbf{Q}^{k+1}, \mathbf{x}^{k+1})$

Minimalizacja liczby stanów wewnętrznych

Minimalizacja układu: zastąpienie danego układu sekwencyjnego innym identycznie działającym układem, ale posiadającym mniejszą liczbę stanów.

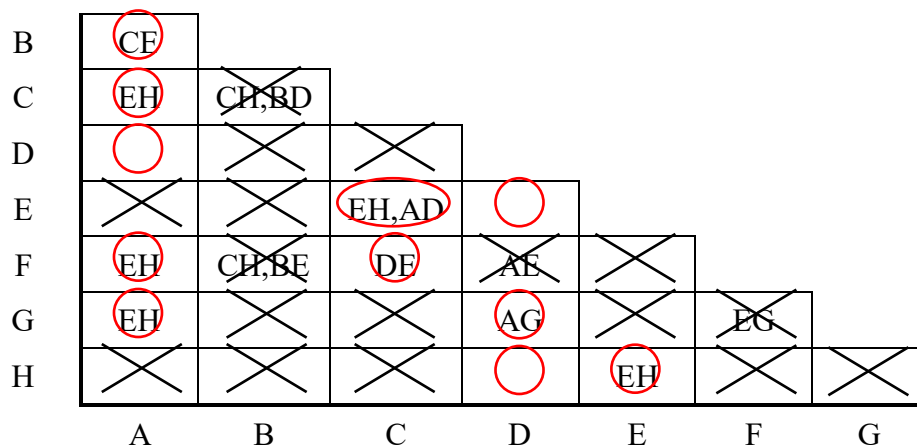
Jako ilustracja pokazana zostanie minimalizacja liczby stanów pewnego układu Mealy'ego.

$Q \backslash x$	0	1
A	E/1	-/-
B	C/1	B/0
C	H/-	D/0
D	-/-	A/1
E	E/0	A/-
F	H/1	E/-
G	H/1	G/1
H	H/0	A/1



Tablica trójkątna – analiza par stanów:

- stany **sprzeczne**, gdy mają sprzeczne wyjścia (oznaczamy \times)
- stany **niesprzeczne**, gdy mają zgodne wyjścia i identyczne przejścia (oznaczamy **O**)
- stany **niesprzeczne warunkowo**, gdy tylko zgodne wyjścia (np. warunek EC dla pary A i B)

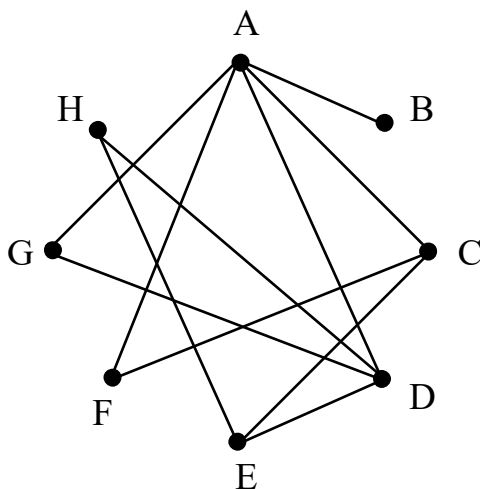


Wypisywanie par stanów niesprzecznych:

AB, AC, AD, AF, AG, CE, CF, DE, DG, DH, EH.

Grupowanie stanów niesprzecznych:

- łączymy odcinkami znalezione pary stanów niesprzecznych,
- kilka par tworzy jedną grupę stanów niesprzecznych, gdy wielokąt ma wszystkie przekątne.



Grupy stanów niesprzecznych:

ACF, ADG, DEH, AB, CE

Wykreślanie – minimalna rodzina stanów (musi być spełniony warunek zamkniętości):

- w grupach można skreślać tylko stany powtarzające się,
- otrzymane po skreśleniach grupy muszą zawierać wszystkie stany z oryginalnej tabeli,
- po skreśleniach stany wewnątrz tabeli muszą być reprezentowane w opisie jej kolumn.

Pierwszy wariant skreśleń

$Q \backslash x$	A	C	F	A	D	G	D	E	H	A	B	C	E
0	E	H	H	E	-	H	-	E	H	E	C	H	E
1	-	D	E	-	A	G	A	A	A	-	B	D	A

$Q \backslash x$	0	1
$CF = \alpha$	$\gamma/1$	$\gamma/0$
$AG = \beta$	$\gamma/1$	$\beta/1$
$DEH = \gamma$	$\gamma/0$	$\beta/1$
$B = \delta$	$\alpha/1$	$\delta/0$

Drugi wariant skreśleń

$Q \backslash x$	A	C	F	A	D	G	D	E	H	A	B	C	E
0	E	H	H	E	-	H	-	E	H	E	C	H	E
1	-	D	E	-	A	G	A	A	A	-	B	D	A

$Q \backslash x$	0	1
$ACF = \alpha$	$\gamma/1$	$\gamma/0$
$G = \beta$	$\gamma/1$	$\beta/1$
$DEH = \gamma$	$\gamma/0$	$\alpha/1$
$B = \delta$	$\alpha/1$	$\delta/0$

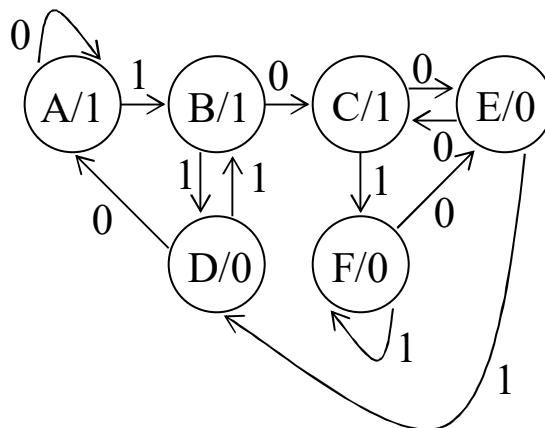
Ważne:

- układ minimalny ma 4 stany (kodowanie stanu na 2 bitach),
- układ minimalny można zrealizować na 2 przerzutnikach.

Zmiana rodzaju opisu: układ Moore'a \leftrightarrow układ Mealy'ego

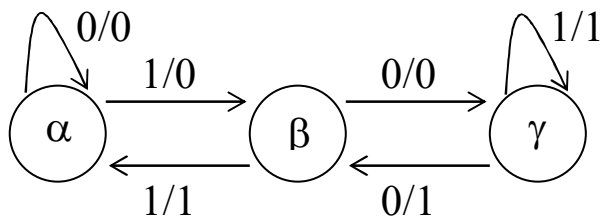
Przykład: zamienić podany (minimalny) układ Moore'a na układ Mealy'ego.

$Q \backslash x$	0	1	y
A	A	B	1
B	C	D	1
C	E	F	1
D	A	B	0
E	C	D	0
F	E	F	0



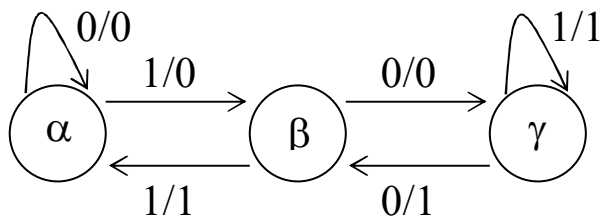
Rozwiązanie: należy wpisać wyjścia do środka tablicy i zminimalizować układ.

$Q \backslash x$	0	1	y
A	A/1	B/1	1
B	C/1	D/0	1
C	E/0	F/0	1
D	A/1	B/1	0
E	C/1	D/0	0
F	E/0	F/0	0



Przykład: zamienić podany (minimalny) układ Mealy'ego na układ Moore'a.

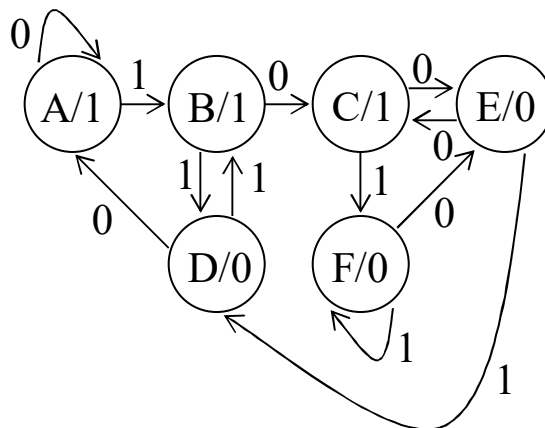
$Q \backslash x$	0	1
α	$\alpha/0$	$\beta/0$
β	$\gamma/0$	$\alpha/1$
γ	$\beta/1$	$\gamma/1$



Rozwiązanie: każdy stan układu Mealy'ego to dwa stany układu Moore'a:

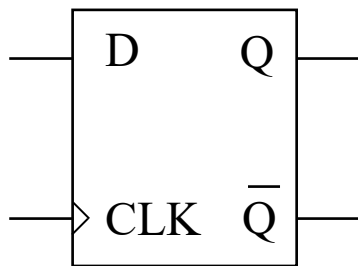
$\alpha/0 = A/0$, $\alpha/1 = D/1$, $\beta/0 = B/0$, $\beta/1 = E/1$, $\gamma/0 = C/0$, $\gamma/1 = F/1$.

$Q \backslash x$	0	1	y
A	A	B	1
B	C	D	1
C	E	F	1
D	A	B	0
E	C	D	0
F	E	F	0

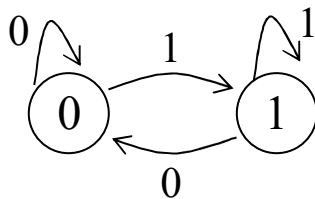


Przerzutniki synchroniczne

przerzutnik typu **D**



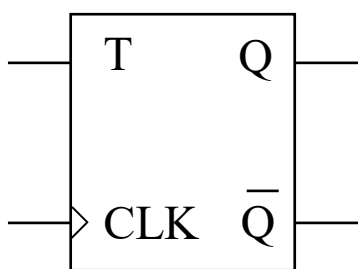
graf



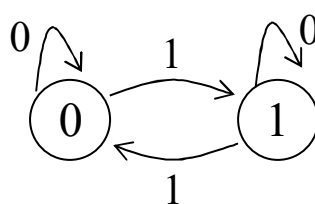
tabela

Q	Q'	D
0	0	0
0	1	1
1	0	0
1	1	1

przerzutnik typu **T**



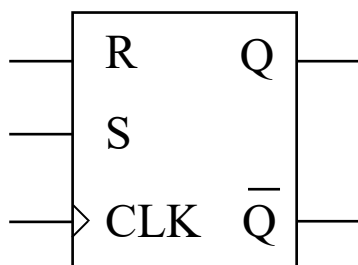
graf



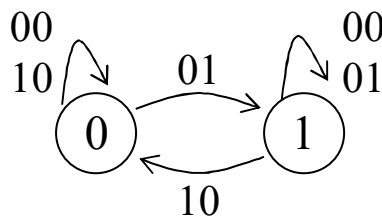
tabela

Q	Q'	T
0	0	0
0	1	1
1	0	1
1	1	0

przerzutnik typu **RS**



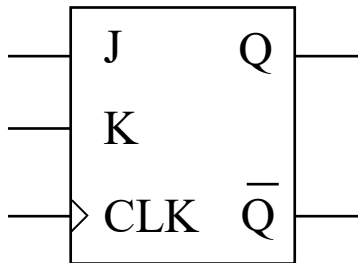
graf



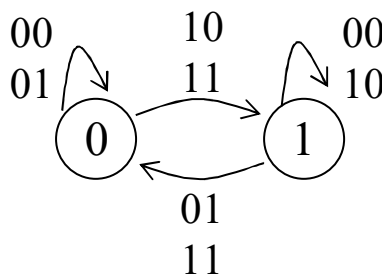
tabela

Q	Q'	R	S
0	0	×	0
0	1	0	1
1	0	1	0
1	1	0	×

przerzutnik typu **JK**



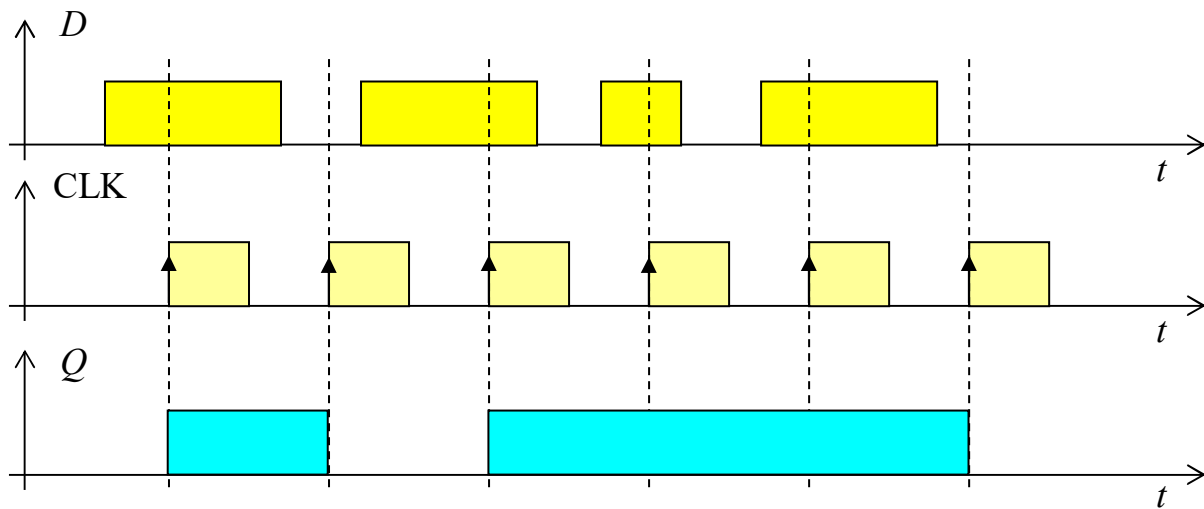
graf



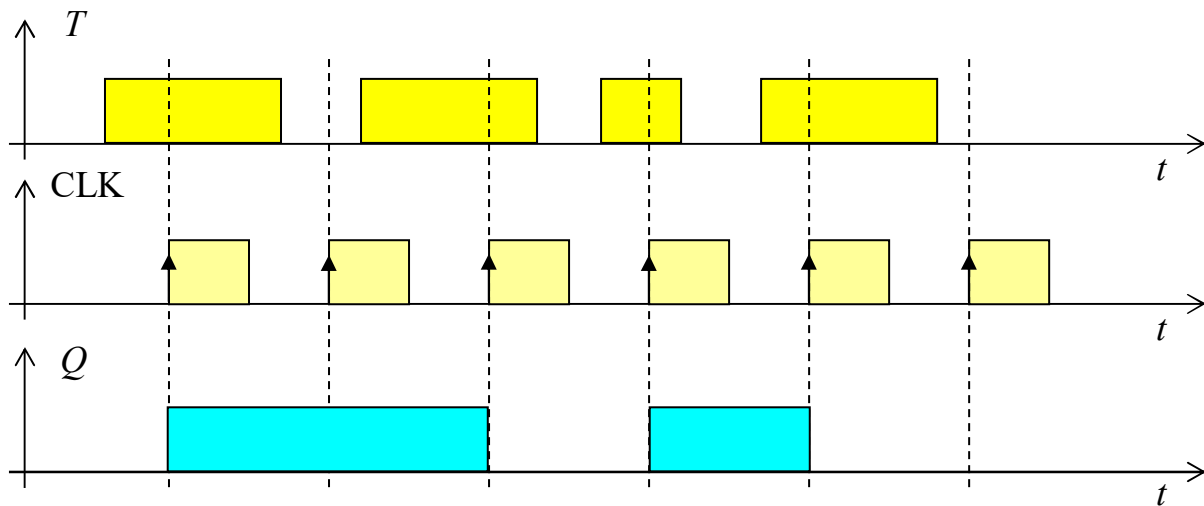
tabela

Q	Q'	J	K
0	0	0	×
0	1	1	×
1	0	×	1
1	1	×	0

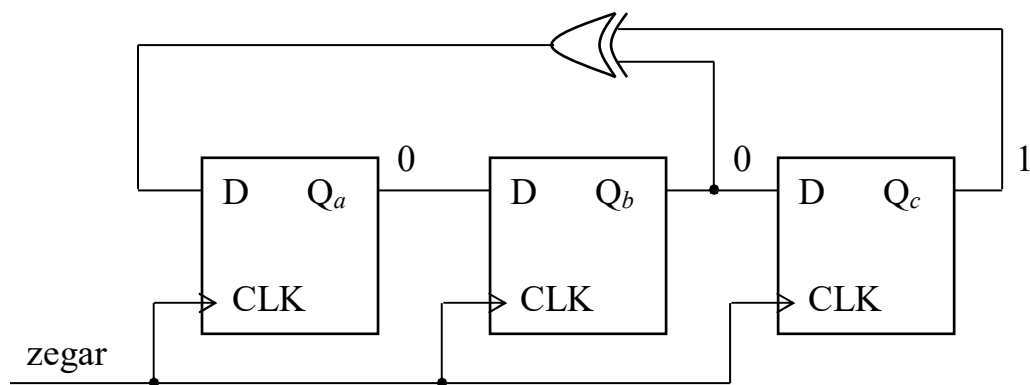
Zadanie: uzupełnić przebiegi czasowe na wyjściu Q przerzutnika D.



Zadanie: uzupełnić przebiegi czasowe na wyjściu Q przerzutnika T.



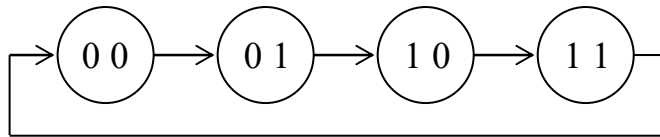
Zadanie: podać sekwencję (Q_a Q_b Q_c) kolejnych stanów układu począwszy od 001.



Rozwiązanie:

→ 001 → 100 → 010 → 101 → 110 → 111 → 011 →

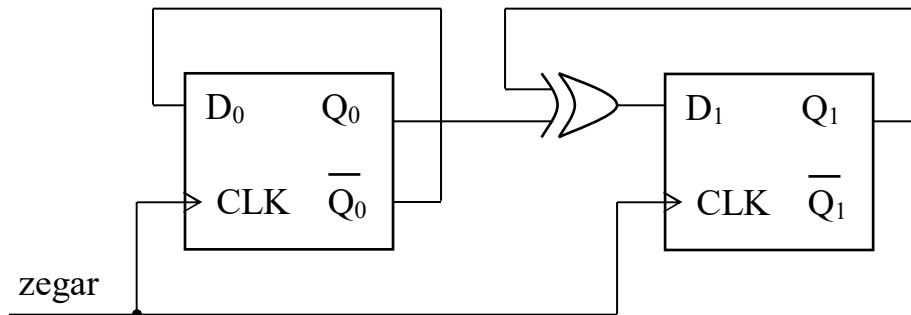
Zadanie: zaprojektować licznik *modulo 4*.



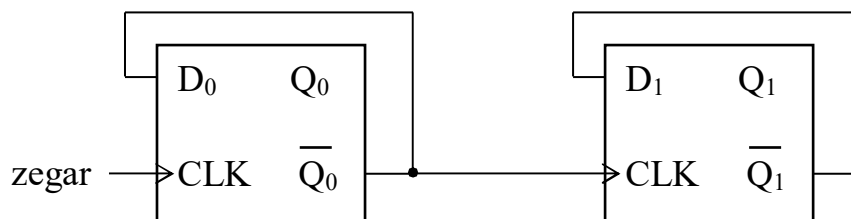
$q_1 \backslash q_0$	0	1
0	0 1	1 0
1	1 1	0 0

Rozwiązanie pierwsze: licznik jako układ sekwencyjny synchroniczny.

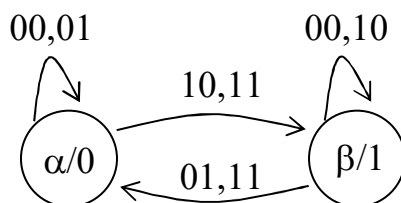
$$D_0 = q_0' = \bar{q}_0 \quad , \quad D_1 = q_1' = q_1 \bar{q}_0 + \bar{q}_1 q_0 = q_1 \oplus q_0$$



Rozwiązanie drugie: licznik asynchroniczny złożony z "dwójek liczących".

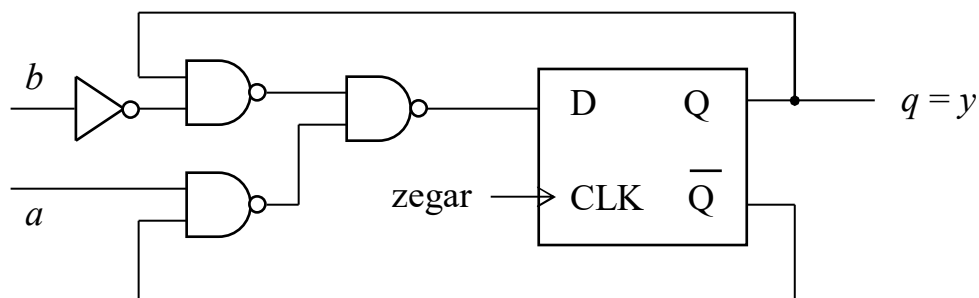


Zadanie: zaprojektować podany układ na przerzutniku D (stan układu q jest wyjściem y).



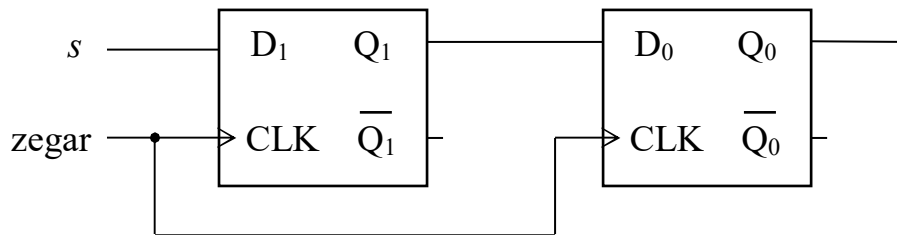
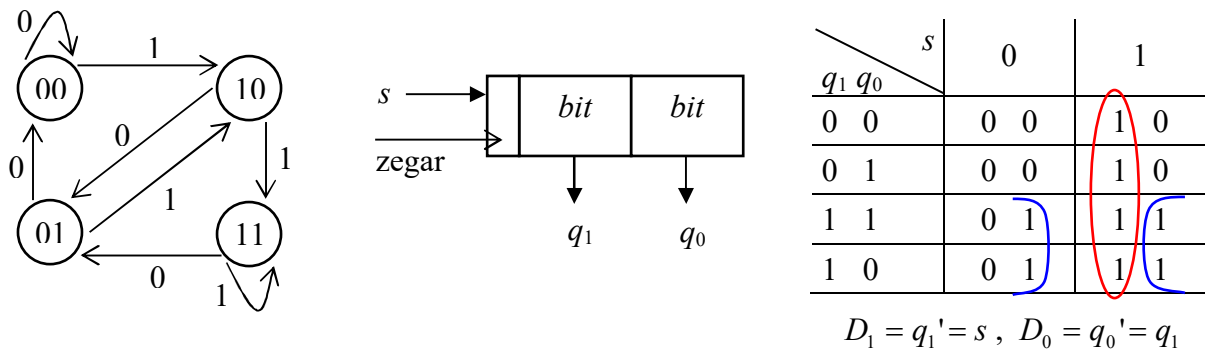
$a \ b$	0 0	0 1	1 1	1 0
q				
$\alpha = 0$	0	0	1	1
$\beta = 1$	1	0	0	1

$$D = q' = \bar{q}a + q\bar{b}$$



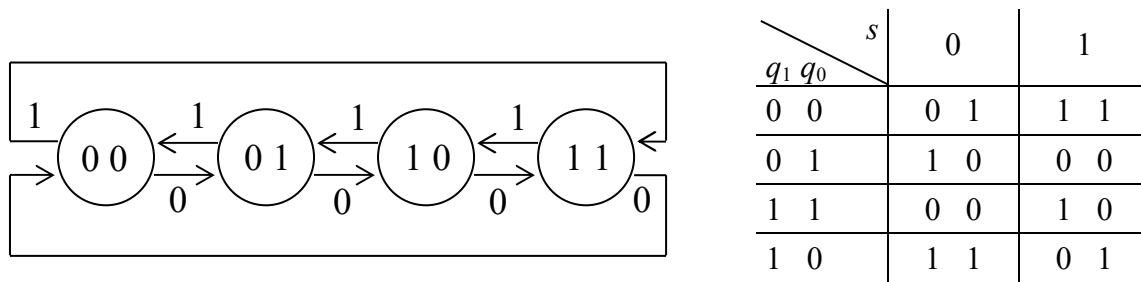
Ważne: otrzymany układ to przerzutnik JK zrealizowany na przerzutniku D (tj. $J=a$, $K=b$).

Zadanie: zbudować dwubitowy rejestr *serial-input-parallel-output* (SIPO) z przesuwem w prawo.



Ważne: łącząc w podany sposób n przerzutników D otrzymujemy n -bitowy rejestr przesuwany.

Zadanie: zaprojektować licznik rewersyjny *modulo 4* na różnych przerzutnikach (np. D, T, JK).



sterowanie przerzutnikiem J_1K_1 dla q_1

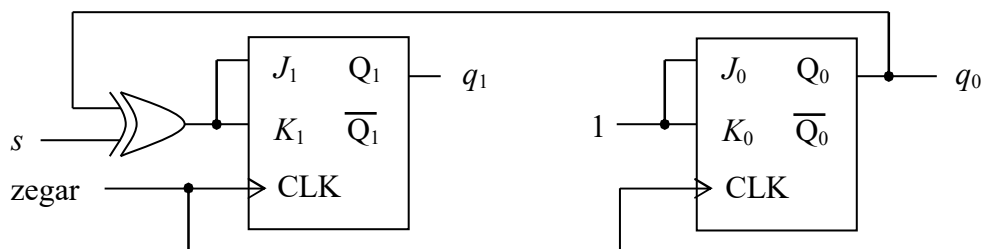
q_1	q_0	s	0	1
0	0	0	x	1
0	1	1	x	0
1	1	x	1	0
1	0	x	0	x

$$J_1 = K_1 = q_0 \cdot \bar{s} + \bar{q}_0 \cdot s = q_0 \oplus s$$

sterowanie przerzutnikiem J_0K_0 dla q_0

q_1	q_0	s	0	1
0	0	1	x	1
0	1	x	1	x
1	1	x	1	x
1	0	1	x	1

$$J_0 = K_0 = 1$$



Zadanie: zbudować zadany przerzutnik korzystając z innego przerzutnika i bramek logicznych:

- korzystając z przerzutnika T i bramek logicznych zbudować przerzutnik D,
- korzystając z przerzutnika T i bramek logicznych zbudować przerzutnik JK,
- korzystając z przerzutnika JK i bramek logicznych zbudować przerzutnik D,
- korzystając z przerzutnika JK i bramek logicznych zbudować przerzutnik T.

Zadanie: zaprojektować układ sekwencyjny synchroniczny obliczający wyrażenia arytmetyczne:

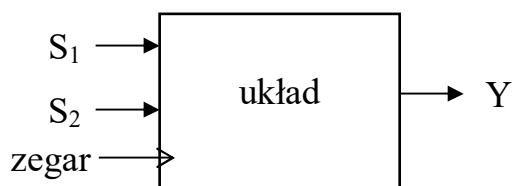
- $y = 3 \cdot a + b$
- $y = 3 \cdot a - b$
- $y = 3 \cdot a + 2 \cdot b$

Zadanie: zmierzchowe sterowanie miejskimi lampami ulicznymi.

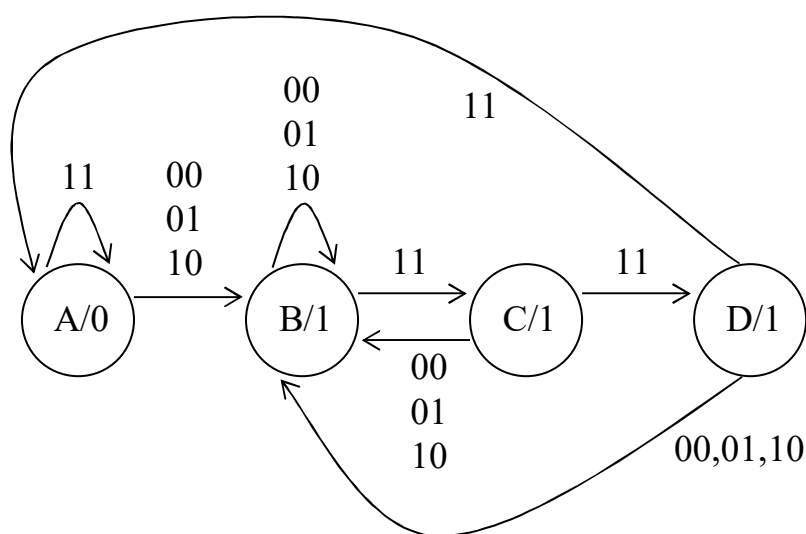
Zaprojektować układ sekwencyjny synchroniczny zmierzchowego sterowania oświetleniem ulic. Układ w kolejnych taktach sygnału zegarowego odczytuje stany logiczne czujników światła S_1 i S_2 . Załączanie lamp odbywa się poprzez ustawienie wyjścia układu w stan wysoki ($Y = 1$), a wyłączenie lamp następuje w momencie ustawienia wyjścia układu w stan niski ($Y = 0$). Pracujące niezależnie czujniki światła słonecznego sygnalizują stanem niskim ($S_i = 0$) porę nocną, zaś stanem wysokim ($S_i = 1$) porę dzienną. Włączanie i wyłączanie lamp odbywa się wg następujących zasad:

- załączanie – jeżeli lampy są wyłączone (dzień) i w chwili testowania stanu czujników co najmniej jeden z nich wskazuje porę nocną, to lampy zostają załączone;
- wyłączanie – jeżeli lampy są załączone (noc) i w kolejnych chwilach testowania stanu czujników obydwa czujniki wskazują zgodnie nastanie pory dziennej, to w momencie stwierdzenia trzeciej z rzędu zgodności wskazań następuje wyłączenie lamp.

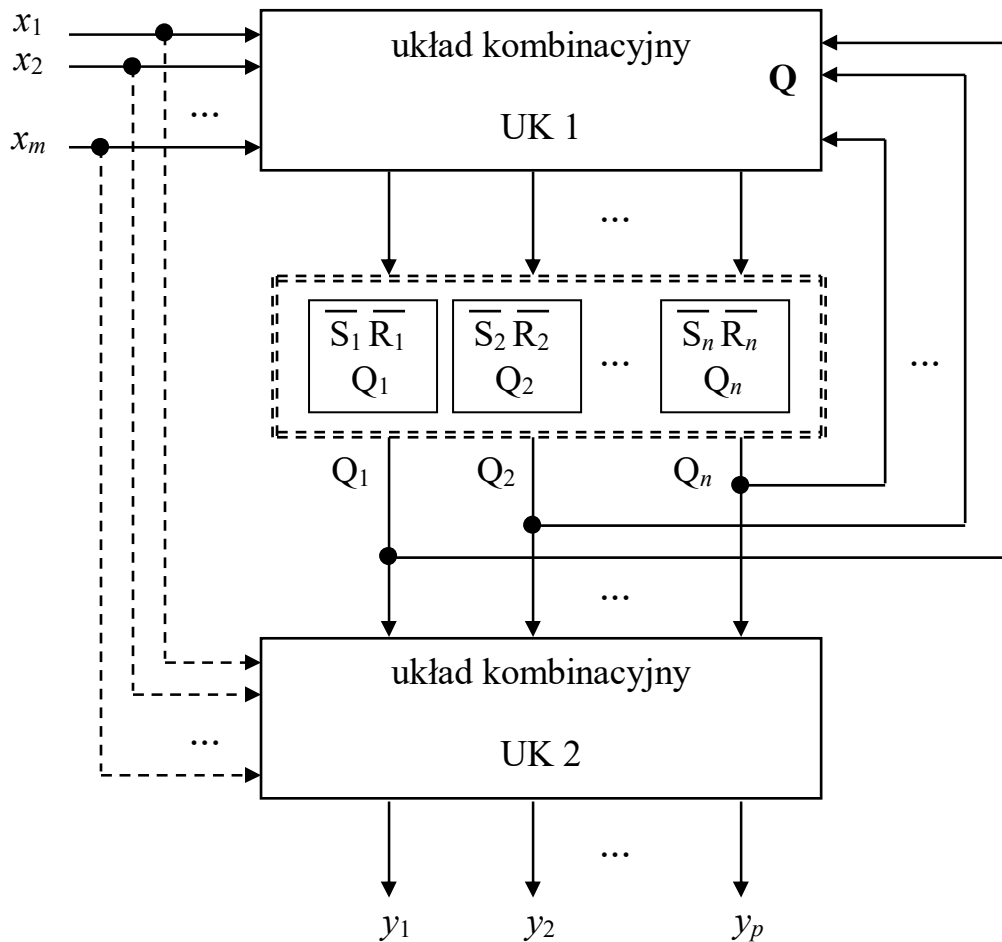
Podać schemat realizacyjny układu na przerzutnikach JK i bramkach NAND.



Wskazówka: działanie układu opisuje poniższy graf.



Układy sekwencyjne asynchroniczne – schemat blokowy



Oznaczenia:

wejscie: $\mathbf{x} = x_1, x_2, \dots, x_m$

stan: $\mathbf{Q} = Q_1, Q_2, \dots, Q_n$

wyjście: $\mathbf{y} = y_1, y_2, \dots, y_p$

Modele układów sekwencyjnych:

• układ Moore'a:

- stan następny (\mathbf{Q}^{k+1}) zależy od stanu bieżącego (\mathbf{Q}^k) i wejścia bieżącego (\mathbf{x}^k),
- wyjście (\mathbf{y}^{k+1}) zależy tylko od stanu (\mathbf{Q}^{k+1}),

• układ Mealy'ego:

- stan następny (\mathbf{Q}^{k+1}) zależy od stanu bieżącego (\mathbf{Q}^k) i wejścia bieżącego (\mathbf{x}^k),
- wyjście (\mathbf{y}^{k+1}) zależy od stanu (\mathbf{Q}^{k+1}) i wejścia (\mathbf{x}^{k+1}).

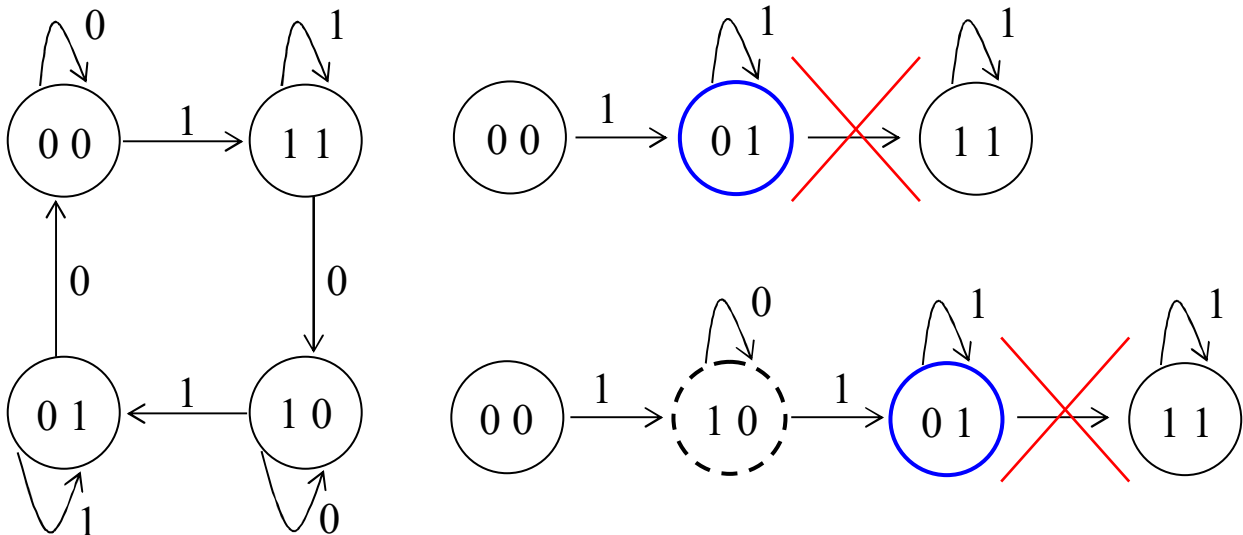
układ Moore'a	układ Mealy'ego
$\mathbf{Q}^{k+1} = f(\mathbf{Q}^k, \mathbf{x}^k)$ $\mathbf{y}^{k+1} = g(\mathbf{Q}^{k+1})$	$\mathbf{Q}^{k+1} = f(\mathbf{Q}^k, \mathbf{x}^k)$ $\mathbf{y}^{k+1} = g(\mathbf{Q}^{k+1}, \mathbf{x}^{k+1})$

Projektowanie układów asynchronicznych – zasady:

- kodowanie stanów **bez wyścigów** (stany połączone różnią się w kodzie na jednym bicie),
- minimalizacja funkcji stanu **bez hazardów** (grupy zer / jedynek tworzą w tablicy "łańcuch"),
- możliwe realizacje układu:
 - kanoniczna sieć sprzężeniowa (NAND lub NOR),
 - kanoniczna sieć sprzężeniowa i przerzutniki S R.

Zjawisko wyścigu – przykład:

Analiza zmiany bitów stanu: $00 \rightarrow 11$

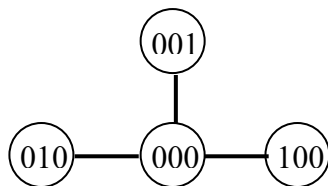


Uwagi:

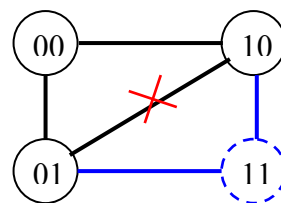
- jednoczesna zmiana bitów stanu (np. $00 \rightarrow 11$) jest niemożliwa (różne czasy propagacji),
- jeśli młodszy bit stanu zmieni się najpierw, to układ zatrzyma się w stanie **01**,
- jeśli starszy bit stanu zmieni się najpierw, to układ minie stan **10** i zatrzyma się w stanie **01**.
- aby zlikwidować wyścig należy zamienić kody stanów **10** i **11**.

Kodowanie stanów – bez wyścigów:

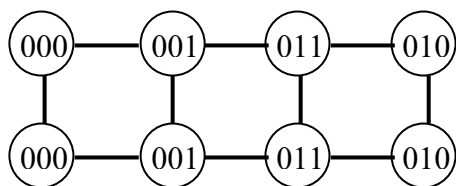
graf typu "gwiazda" (więcej bitów kodu)



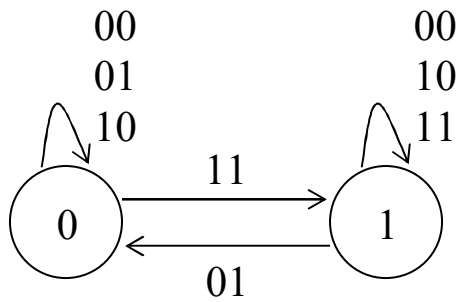
graf typu "trójkąt" (dodać pomocniczy stan)



graf typu "kratownica" (kody Gray'a)



Zjawisko hazardu – przykład:

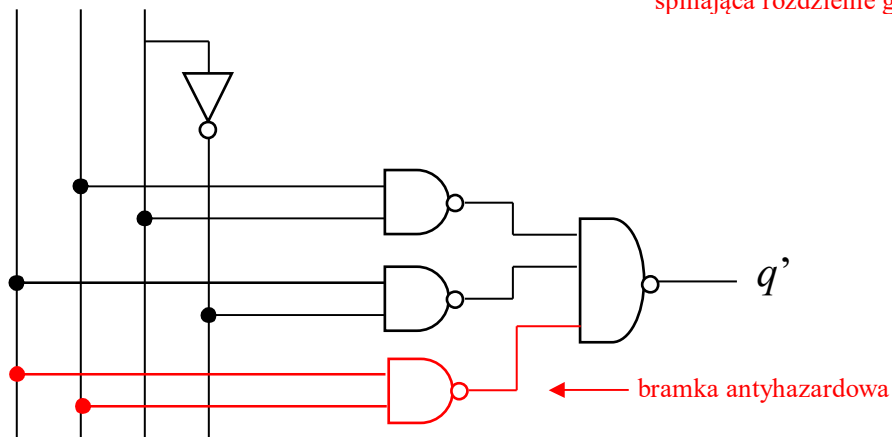


$q \backslash a \ b$	00	01	11	10
0	0	0	1	0
1	1	0	1	1

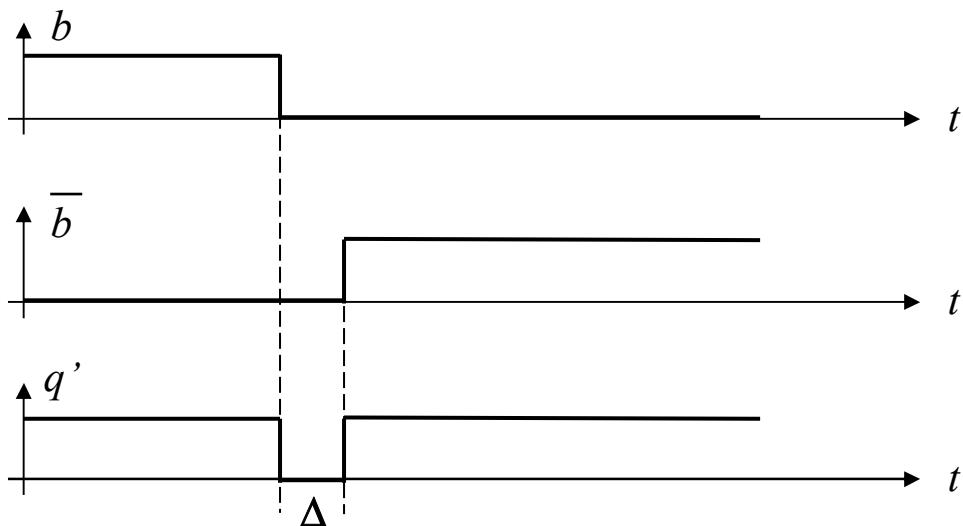
$$q' = a \cdot b + q \cdot \bar{b} + \textcolor{red}{q} \cdot \textcolor{red}{a}$$

*dodatkowa grupa antyhazardowa
spinająca rozdzielne grupy*

$q \quad a \quad b$



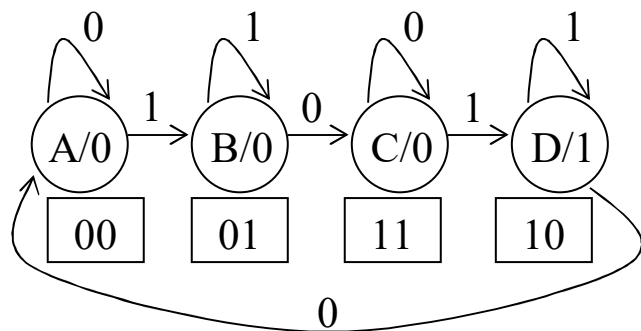
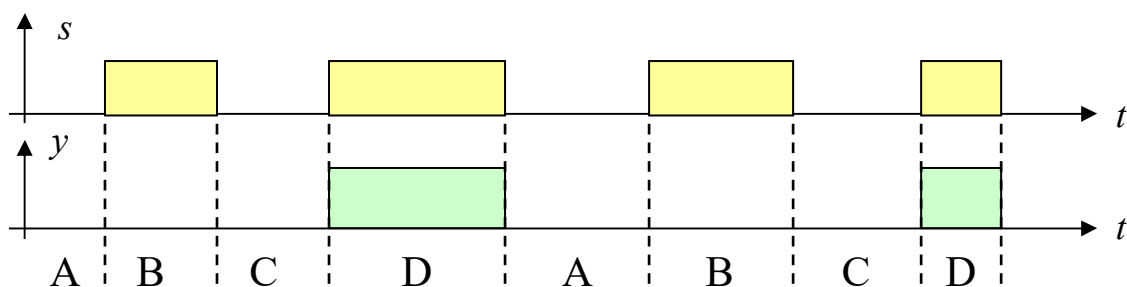
Analiza czasowa: ($q = 1$, $a = 1$, $q' = a \cdot b + q \cdot \bar{b}$, Δ – czas propagacji bramki NOT)



Uwagi:

- zgodnie z tabelą zmiana wejścia b z 1 na 0 (gdy $q = 1$ i $a = 1$) nie powinna zmienić wartości q ,
- z uwagi na niezerowy czas propagacji bramki NOT nowy stan będzie chwilowo $q' = 0$,
- aby zlikwidować hazard należy uwzględnić grupę bitów "spinającą" rozdzielne grupy,
- w układzie asynchronicznym hazard może być szkodliwy, gdyż q' jest połączone z q .

Zadanie: zbudować układ asynchroniczny przepuszczający co drugi impuls z wejścia na wyjście.



	s	0	1	y
A	00	00	01	
B	01			
C	11	11	01	
D	10	11	10	

Rozwiązanie 1: sieć sprzężeniowa NAND.

$q_1 \backslash q_0$	0	1
00	0	0
01	1	0
11	1	1
10	0	1

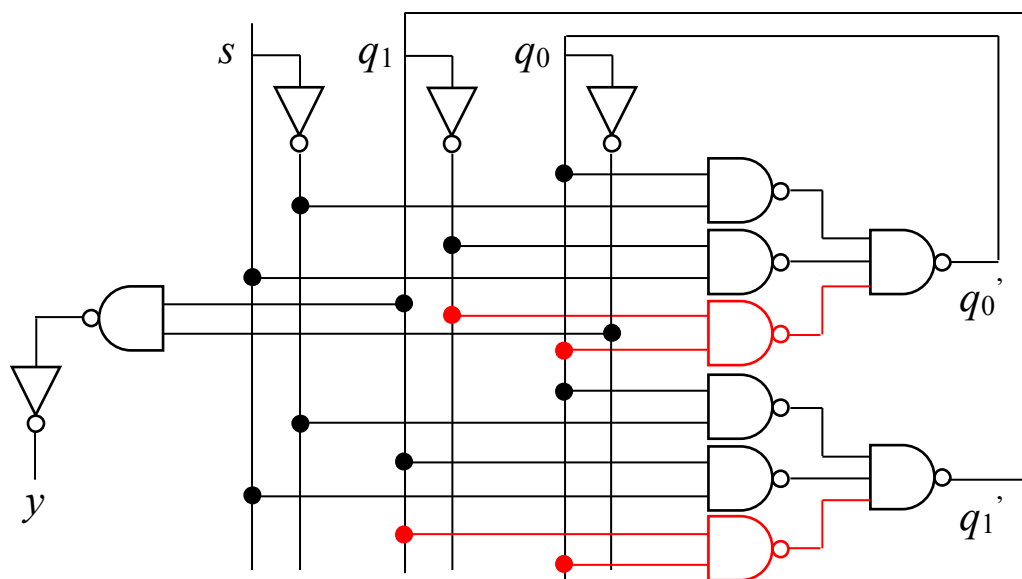
$q_1 \backslash s$	0	1
00	0	1
01	1	1
11	1	0
10	0	0

$q_1 \backslash q_0$	0	1
0	0	0
1	1	0

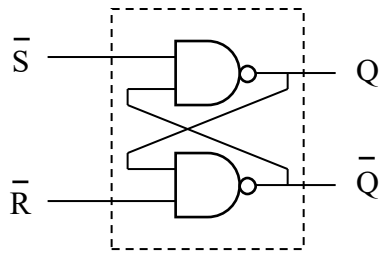
$$q_1' = q_0 \bar{s} + q_1 s + q_1 q_0$$

$$q_0' = q_0 \bar{s} + \bar{q}_1 s + \bar{q}_1 q_0$$

$$y = q_1 \bar{q}_0$$



Rozwiązanie 2: wykorzystanie przerzutników asynchronicznych S R.



Q	Q'	\bar{S}	\bar{R}
0	0	1	x
0	1	0	1
1	0	1	0
1	1	x	1

sterowanie $S_1 R_1$ dla q_1 :

$$\bar{S}_1 = \bar{q}_0 + s$$

$$\bar{R}_1 = q_0 + s$$

s	0	1
$q_1 q_0$		
0 0	1x	1x
0 1	01	1x
1 1	x1	x1
1 0	10	x1

s	0	1
$q_1 q_0$		
0 0	1	1
0 1	0	1
1 1	x	x
1 0	1	x

s	0	1
$q_1 q_0$		
0 0	x	x
0 1	1	x
1 1	1	1
1 0	0	1

sterowanie $S_0 R_0$ dla q_0 :

$$\bar{S}_0 = q_1 + \bar{s}$$

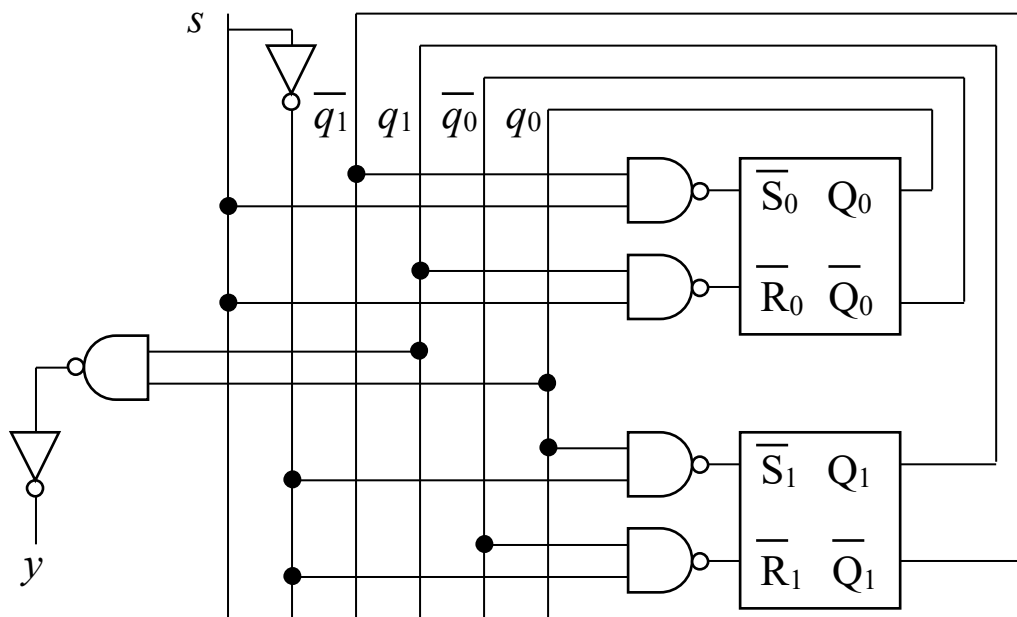
$$\bar{R}_0 = \bar{q}_1 + \bar{s}$$

s	0	1
$q_1 q_0$		
0 0	1x	01
0 1	x1	x1
1 1	x1	10
1 0	1x	1x

s	0	1
$q_1 q_0$		
0 0	1	0
0 1	x	x
1 1	x	1
1 0	1	1

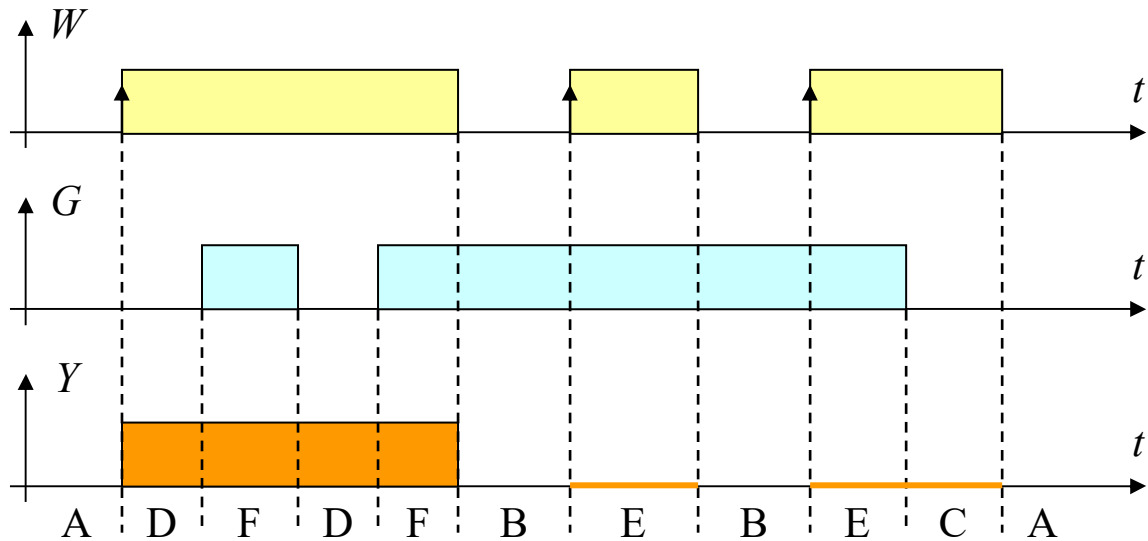
s	0	1
$q_1 q_0$		
0 0	x	1
0 1	1	1
1 1	1	0
1 0	x	x

realizacja wyjścia jak poprzednio: $y = q_1 \bar{q}_0$



Zadanie: zaprojektować układ bramkujący sygnałem G impulsy podawane z wejścia W :

- jeżeli narastające zbocze impulsu W pojawia się przy stanie niskim sygnału bramkującego ($G=0$), to impuls W zostaje przepuszczony na wyjście Y ,
- jeżeli narastające zbocze impulsu W pojawia się przy stanie wysokim sygnału bramkującego ($G=1$), to impuls W nie jest przepuszczony na wyjście Y .

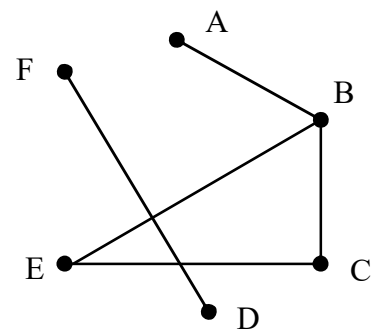


$W \ G$						
Q		00	01	11	10	Y
A		A	B	-	D	0
B		A	B	E	-	0
C		A	-	E	C	0
D		A	-	F	D	1
E		-	B	E	C	0
F		-	B	F	D	1

W	G	Y	stan Q
0	0	0	= A
0	0	1	niemożliwe
0	1	0	= B
0	1	1	niemożliwe
1	0	0	= C
1	0	1	= D
1	1	0	= E
1	1	1	= F

Minimalizacja liczby stanów: pary stanów zgodnych (AB, BC, BE, CE, DF)

B	○				
C	○	○			
D	○	○	○		
E	○	○	○	○	
F	○	○	○	○	○
	A	B	C	D	E



Wykreślanie zbędnych stanów z grup: { AB, BCE, DF }

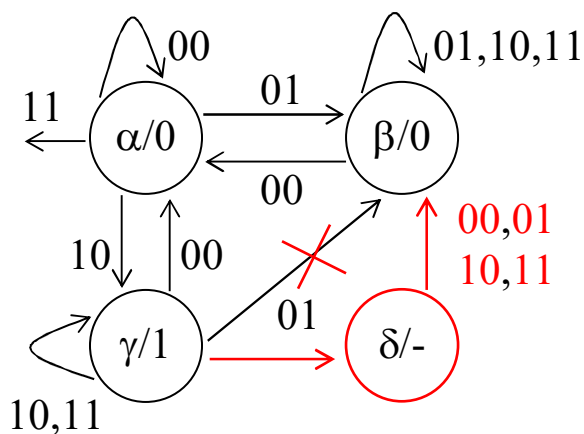
$W G \backslash Q$	A B	B C E	D F
0 0	A A	A A -	A -
0 1	B B	B - B	- B
1 1	- E	E E E	F F
1 0	D -	- C C	D D

Stan **B** powtarza się: albo wykreślamy **B** z grupy AB albo wykreślamy **B** z grupy BCE.

Obydwa rozwiązania są poprawne (spełnione warunki zamkniętości).

Dwie rodziny minimalne: { A , BCE , DF } albo { AB , CE , DF }.

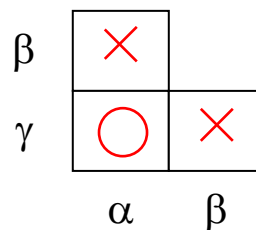
Graf minimalny Moore'a: $\alpha = A$, $\beta = BCE$, $\gamma = DF$ (oraz stan **δ** aby usunąć wyścig)



$W G \backslash Q$	00	01	11	10	Y
α	α	β	-	γ	0
β	α	β	β	β	0
γ	α	β	γ	γ	1

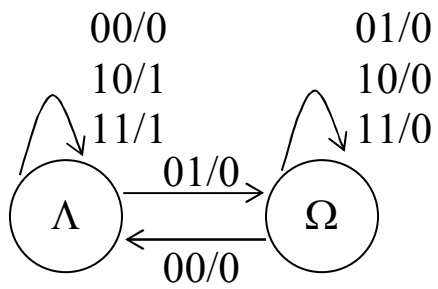
Zamiana układu Moore'a na układ Mealy'ego: (wyjścia Y wpisujemy do tablicy)

$W G \backslash Q$	0 0	0 1	1 1	1 0	Y
α	$\alpha/0$	$\beta/0$	-/-	$\gamma/1$	0
β	$\alpha/0$	$\beta/0$	$\beta/0$	$\beta/0$	0
γ	$\alpha/0$	$\beta/0$	$\gamma/1$	$\gamma/1$	1



Rodzina minimalna: { $\alpha \gamma$, β }.

Graf minimalny Mealy'ego: ($\Lambda = \alpha\gamma$, $\Omega = \beta$)



		WG			
Q		00	01	11	10
	Λ	$\Lambda/0$	$\Omega/0$	$\Lambda/1$	$\Lambda/1$
	Ω	$\Lambda/0$	$\Omega/0$	$\Omega/0$	$\Omega/0$

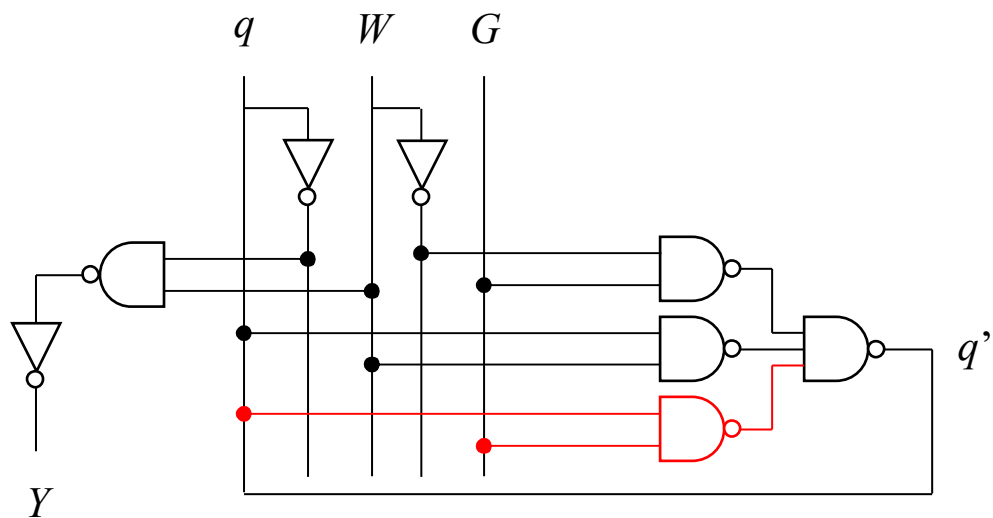
Realizacja: ($\Lambda = 0$, $\Omega = 1$)

stan: $q' = \overline{W} \cdot G + q \cdot W + \textcolor{red}{q} \cdot \textcolor{red}{G}$

wyście: $Y = \overline{q} \cdot W$

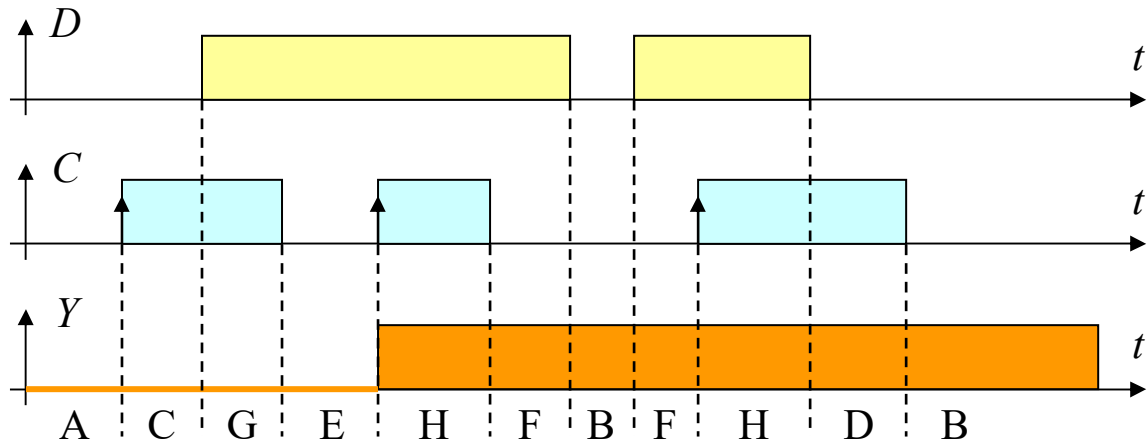
WG	00	01	11	10
q				
0	0	1	0	0
1	0	1	1	1

WG	00	01	11	10
q				
0	0	0	1	1
1	0	0	0	0



Zadanie: zbudować przerzutnik synchroniczny D wyzwalany narastającym zboczem zegara C .

Narastające zbocze zegara C wpisuje daną D na wyjście Y – pokazuje to rysunek.

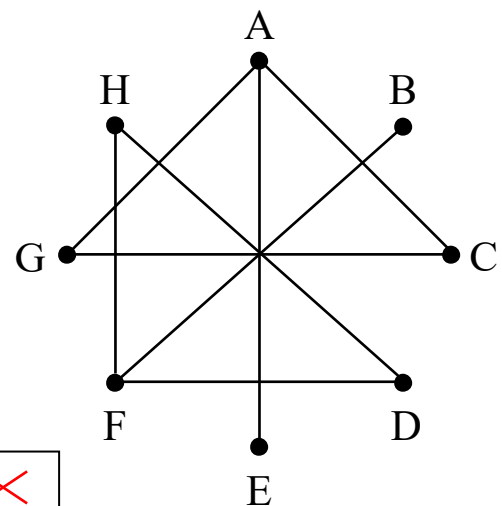


$D \ C$						Y
		00	01	11	10	
Q	A	A	C	-	E	0
	B	B	C	-	F	1
	C	A	C	G	-	0
	D	B	D	H	-	1
	E	A	-	H	E	0
	F	B	-	H	F	1
	G	-	C	G	E	0
	H	-	D	H	F	1

D	C	Y	stan Q
0	0	0	= A
0	0	1	= B
0	1	0	= C
0	1	1	= D
1	0	0	= E
1	0	1	= F
1	1	0	= G
1	1	1	= H

Minimalizacja liczby stanów:

	A	B	C	D	E	F	G
B	×						
C	○	×					
D	×	CD	×				
E	○	×	GH	×			
F	×	○	×	○	×		
G	○	×	○	×	GH	×	
H	×	CD	×	○	×	○	×



Pary stanów zgodnych: (AC, AE, AG, BF, CG, DF, DH, FH)

Wykreślanie zbędnych stanów z grup: { ACG, DFH, AE, BF }

Q	ACG	DFH	AE	BF
DC				
0 0	A A -	B B -	A A	B B
0 1	C C C	D - D	C -	C -
1 1	- G G	H H H	- H	- H
1 0	E - E	- F F	E E	F F

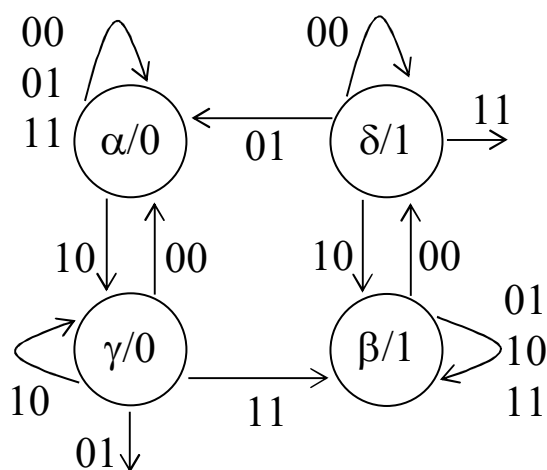
Stany **A** oraz **F** powtarzają się:

- albo wykreślamy **A** z grupy AE oraz wykreślamy **F** z grupy BF,
- albo wykreślamy **A** z grupy ACG oraz wykreślamy **F** z grupy DFH.

Obydwa rozwiązania są poprawne (spełniony warunek zamkniętości).

Dwie rodziny minimalne: { ACG , DFH , E , B } albo { CG , DH , AE , BF }.

Graf minimalny Moore'a: ($\alpha = \text{ACG}$, $\beta = \text{DFH}$, $\gamma = \text{E}$, $\delta = \text{B}$), $\alpha = 00$, $\gamma = 01$, $\beta = 11$, $\delta = 10$.



DC	0 0	0 1	1 1	1 0
$q_1 q_0$				
0 0	0 0	0 0	0 0	0 1
0 1	0 0	x x	1 1	0 1
1 1	1 0	1 1	1 1	1 1
1 0	1 0	0 0	x x	1 1

q_0	0	1
q_1		
0	0	0
1	1	1

$Y = q_1$

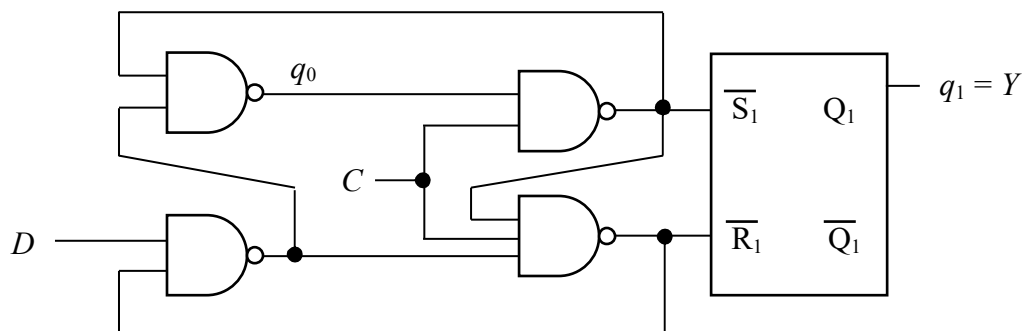
Realizacja sprzężeniowa na bramkach NAND: $q_1' = q_0 C + q_1 \bar{C} + q_1 q_0$, $q_0' = q_0 C + D \bar{C} + q_0 D$.

Realizacja na przerzutnikach SR: $\bar{S}_1 = \bar{q}_0 + \bar{C}$, $\bar{R}_1 = q_0 + \bar{C}$, $\bar{S}_0 = \bar{D} + C$, $\bar{R}_0 = D + C$.

Realizacja mieszana (z wyjściowym przerzutnikiem $S_1 R_1$ i bramkami NAND):

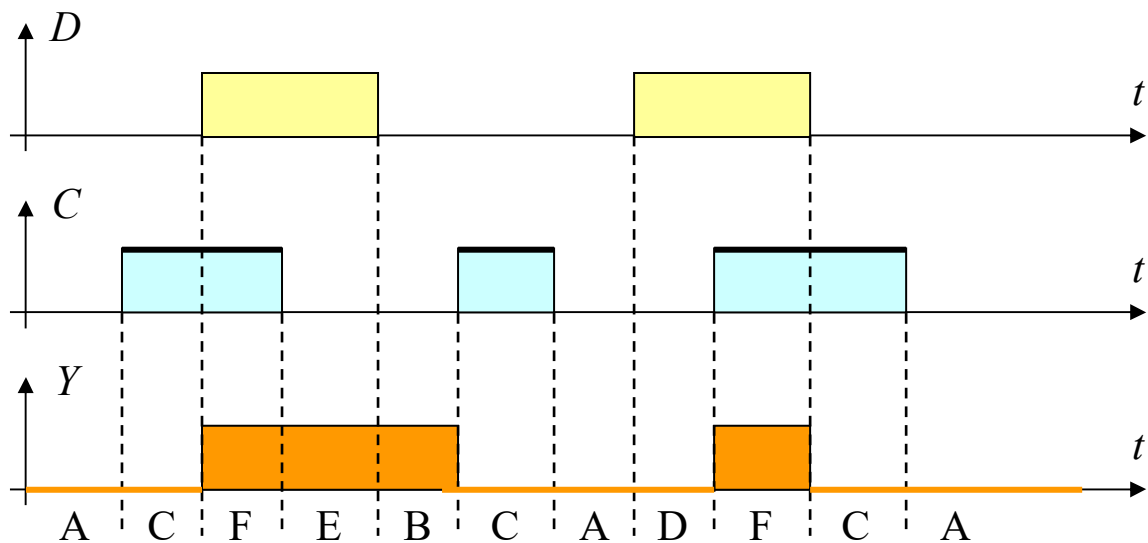
– zachodzi: $\bar{S}_1 = \bar{q}_0 + \bar{C}$, $\bar{R}_1 = q_0 + \bar{C}$, $q_0' = q_0 C + D(q_0 + \bar{C}) = \bar{\bar{S}}_1 + D \bar{R}_1$,

– stąd: $\bar{R}_1 = q_0 + \bar{C} = \bar{\bar{S}}_1 + D \bar{R}_1 + \bar{C}$.



Zadanie: zbudować przerzutnik synchroniczny D wyzwalany poziomym sygnałem C (D -zatrząsk).

Dla $C=1$ dana D przechodzi na wyjście Y , a dla $C=0$ wyjście Y nie zmienia się (zatrząsk).

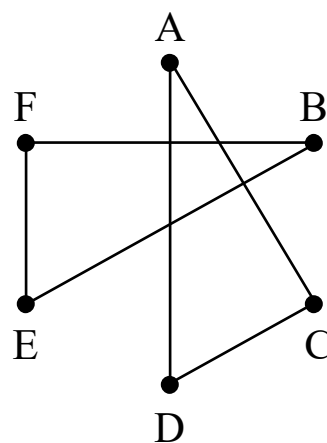


$D \ C$		00	01	11	10	Y
Q	A	A	C	-	D	0
	B	B	C	-	E	1
	C	A	C	F	-	0
	D	A	-	F	D	0
	E	B	-	F	E	1
	F	-	C	F	E	1

D	C	Y	stan Q
0	0	0	= A
0	0	1	= B
0	1	0	= C
0	1	1	niemożliwe
1	0	0	= D
1	0	1	= E
1	1	0	niemożliwe
1	1	1	= F

Minimalizacja liczby stanów:

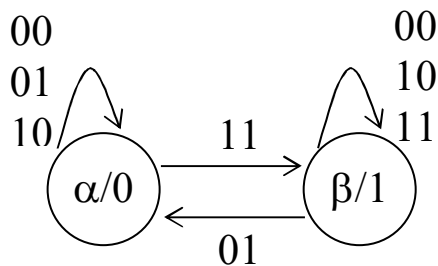
	A	B	C	D	E
B	×				
C	○	×			
D	○	×	○		
E	×	○	×	×	
F	×	○	×	×	○



Pary stanów zgodnych: (AC, AE, AG, BF, CG, DF, DH, FH)

Rodzina minimalna (skreślenia nie zachodzą): $\{ ACD, BEF \}$.

Graf minimalny Moore'a: ($\alpha = ACD, \beta = BEF$), $\alpha = 0, \beta = 1$.



$$q' = D \cdot C + q \cdot \bar{C} + q \cdot D, \quad Y = q$$

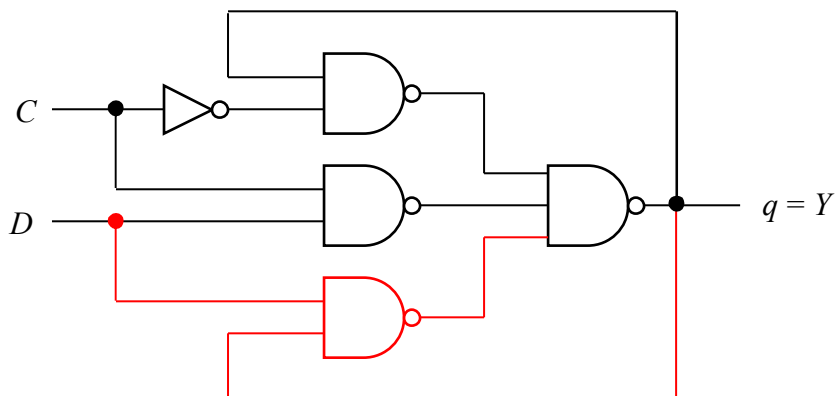
$D C$	0 0	0 1	1 1	1 0
q				
0	0	0	1	0
1	1	0	1	1

sterowanie S R dla q : $\bar{S} = \bar{D} + \bar{C}, \quad \bar{R} = D + \bar{C}$.

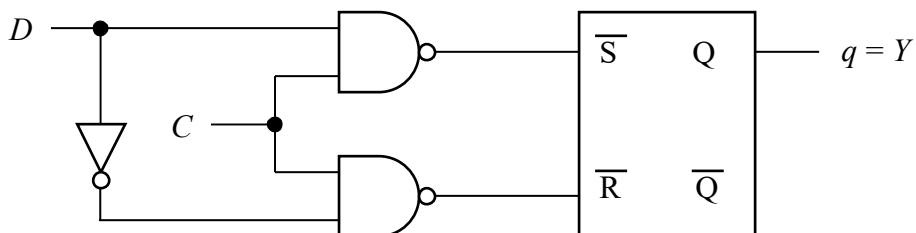
$D C$	0 0	0 1	1 1	1 0
q				
0	1 x	1 x	0 1	1 x
1	x 1	1 0	x 1	x 1

Q	Q'	\bar{S}	\bar{R}
0	0	1	x
0	1	0	1
1	0	1	0
1	1	x	1

Sieć sprzężeniowa NAND:

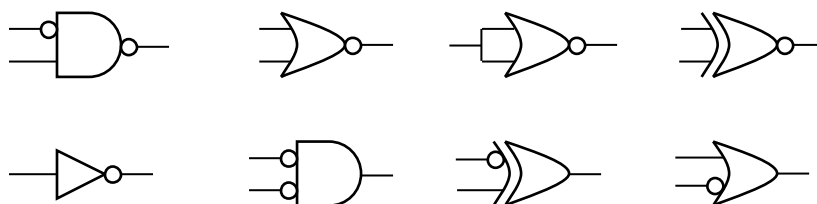


Sieć z przerzutnikiem S R:



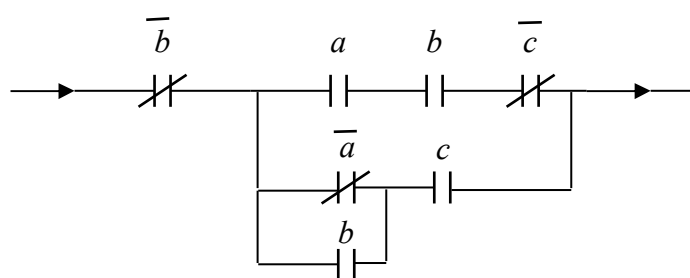
Przykładowe testy egzaminacyjne

1. Podane liczby uporządkować rosnąco: 11000011_{U2} , 10111110_{U1} , 10011011_{Z-M} , -123 .
2. Liczbę dziesiętną $L = 183,17$ zamienić algorytmicznie na liczbę binarną (5 bitów po przecinku).
3. Wskazać pary bramek działających identycznie:



4. Podać tabelki funkcji: $f(a, b) = a \oplus b$; $g(a, b) = a \rightarrow b$. Podać schematy Venna tych funkcji. Funkcje wyrazić w sposób minimalny za pomocą negacji, sumy i iloczynu logicznego.

5. Jaką funkcję $f(a, b, c)$ realizuje podana sieć zestykowa? Funkcję zrealizować w sposób minimalny na bramkach NAND.



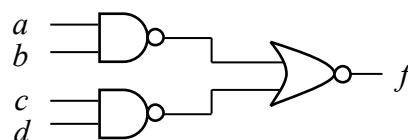
6. Zbadać, czy $g_1 = a \cdot c$ oraz $g_2 = \bar{a} \cdot c$ są implikantami prostymi funkcji $f(a, b, c) = \Sigma(0, 1, 4, 5, 7)$.

7. Funkcję $f(a, b, c) = a \cdot \bar{c} + \bar{b}$ sprowadzić do KPI i zrealizować na bramkach NOR.

8. Funkcję $f(a, b, c) = (a \rightarrow \bar{b}) \rightarrow \bar{c}$ sprowadzić do KPS i zrealizować na Mpx 4/1.

9. Jaką funkcję $f(a, b, c, d)$ realizuje sieć bramek?

Funkcję zminimalizować po "jedynkach".

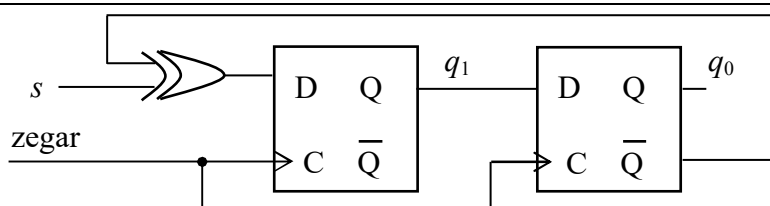


10. Funkcję $f(a, b, c, d) = \Sigma(4, 6, 8, 9, 10, 11, (12, 14))$ zminimalizować po "zerach" bez hazardu.

11. Podać graf układu iteracyjnego odejmującego dwie liczby binarne.

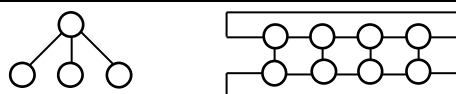
12. Korzystając z dowolnych bramek logicznych i przerzutnika synchronicznego typu T zbudować przerzutnik typu D. Podać grafy obydwu przerzutników.

13. Odtworzyć pełny graf realizowany przez podany układ (stan: $q_1 q_0$).



14. Narysować przebiegi czasowe ilustrujące zasadę działania przerzutnika D wyzwalanego narastającym zboczem sygnału C oraz przerzutnika D wyzwalanego poziomem sygnału C.

15. Zakodować bez wyścigów stany w grafach układów asynchronicznych. Kody stanów muszą zawierać minimalną liczbę bitów.



16. Układ asynchroniczny przepuszcza na wyjście co trzeci impuls z wejścia. Podać przebieg czasowy i zaznaczyć na nim stany układu. Narysować graf układu i poprawnie go zakodować.