

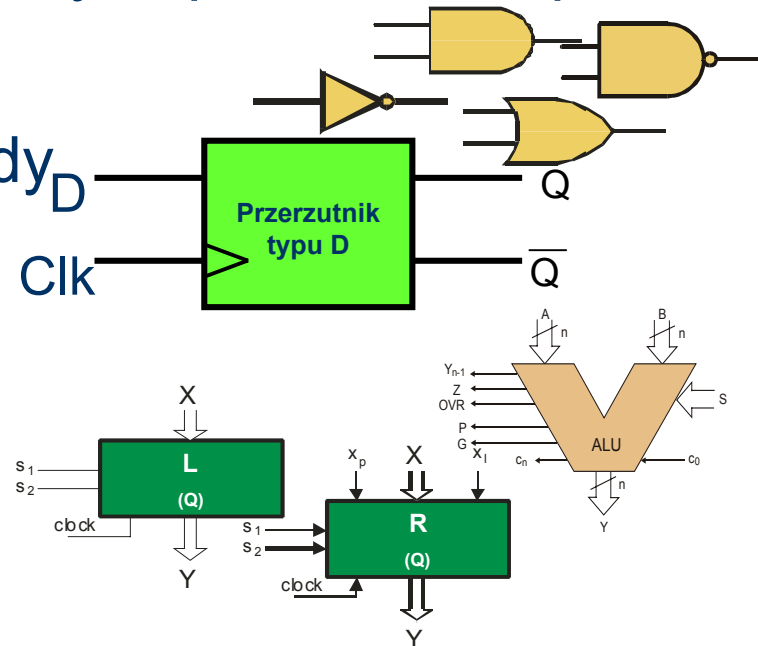
Układy cyfrowe

Układy logiczne (cyfrowe) konstruowane są w różnych technologiach i na różnych poziomach opisu.

Poziomy opis:

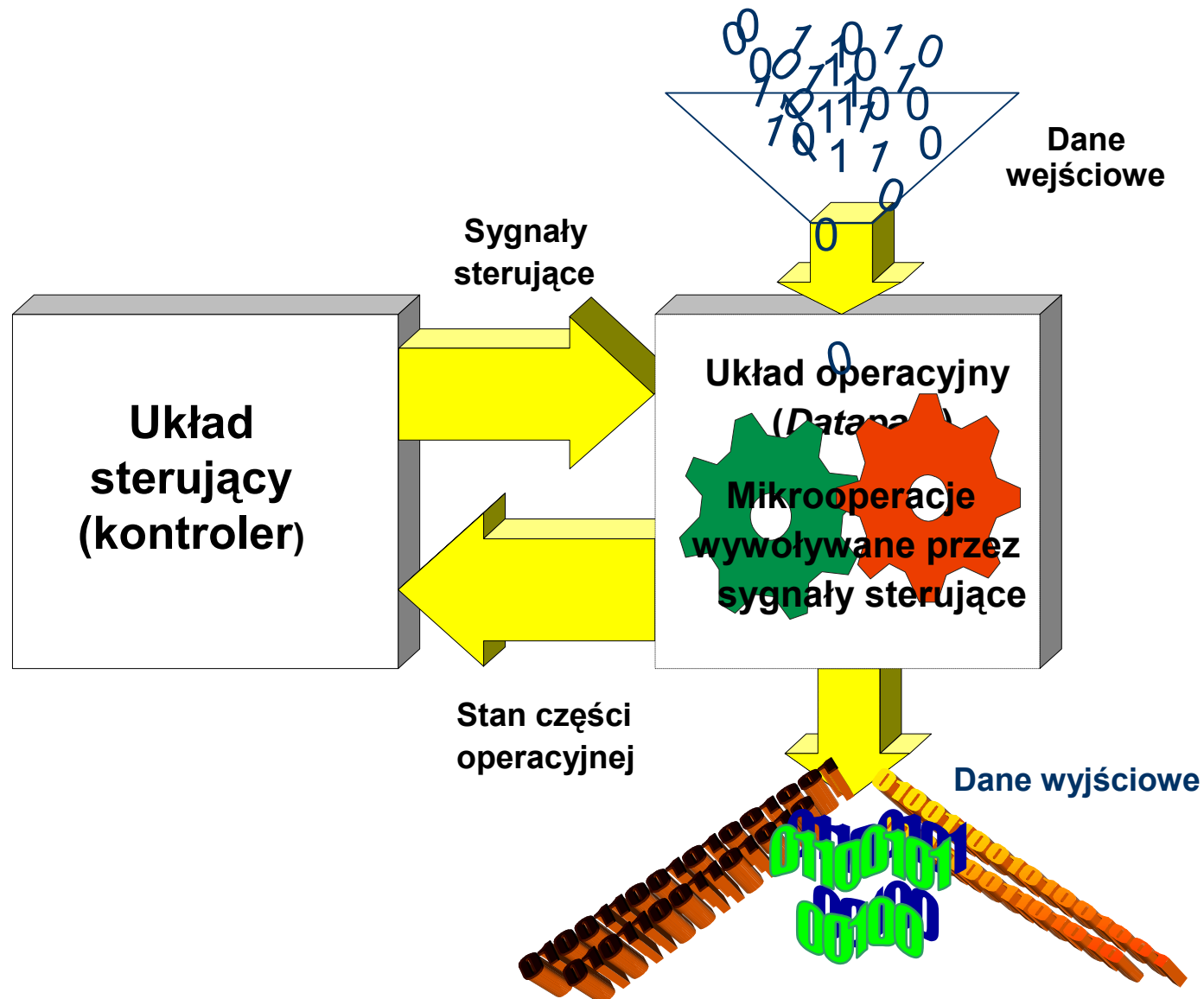
1) Bramki i elementarne układy pamięciowe (przerzutniki)

2) Bloki funkcjonalne: układy arytmetyczne (sumatory), liczniki, rejestry.

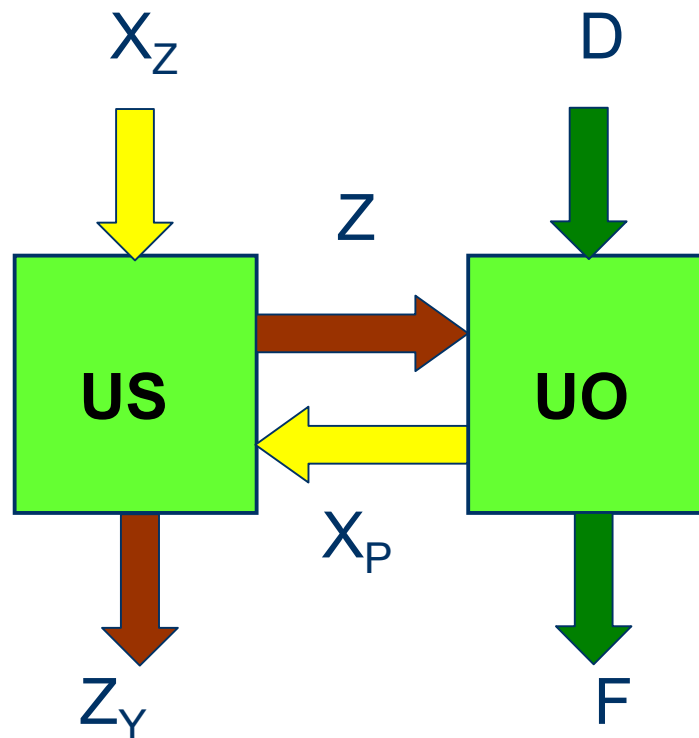


Tworzą one nowe elementy konstrukcyjne, z których buduje się złożone układy cyfrowe o różnorodnych zastosowaniach: układy przetwarzania sygnałów, układy sterowania, specjalizowane procesory, układy kryptograficzne

System cyfrowy



System cyfrowy...



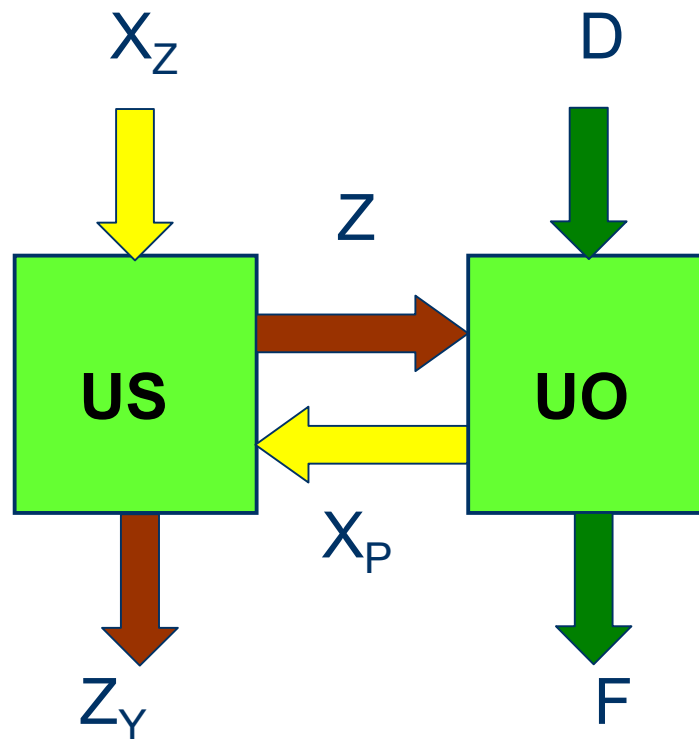
D, F - przetwarzana informacja
(wektory binarne),

X - sygnały warunków,

Z - sygnały sterujące
(mikrorozkazy)

US - układ sterujący UO - układ operacyjny

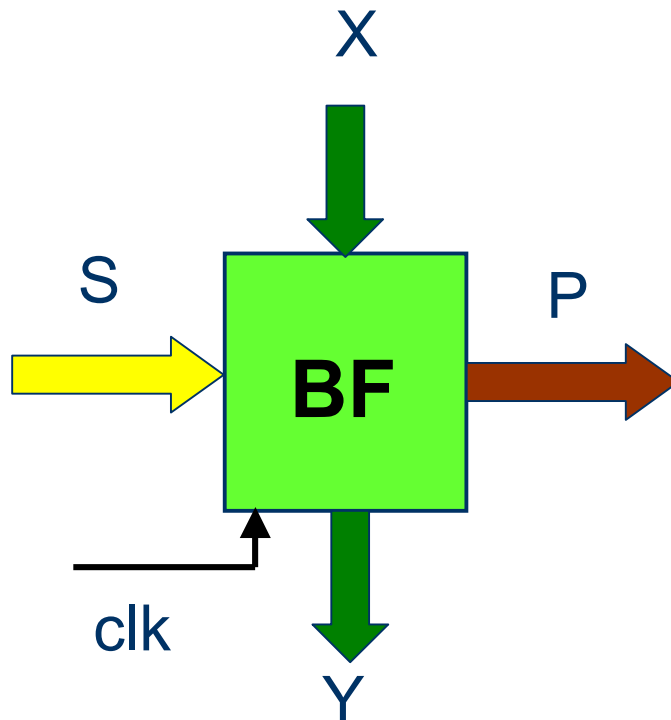
System cyfrowy - realizacja



UO – z bloków funkcjonalnych

**US – automat
lub układ mikroprogramowany**

Bloki funkcjonalne



X – wejścia sygnałów
reprezentujących dane wejściowe

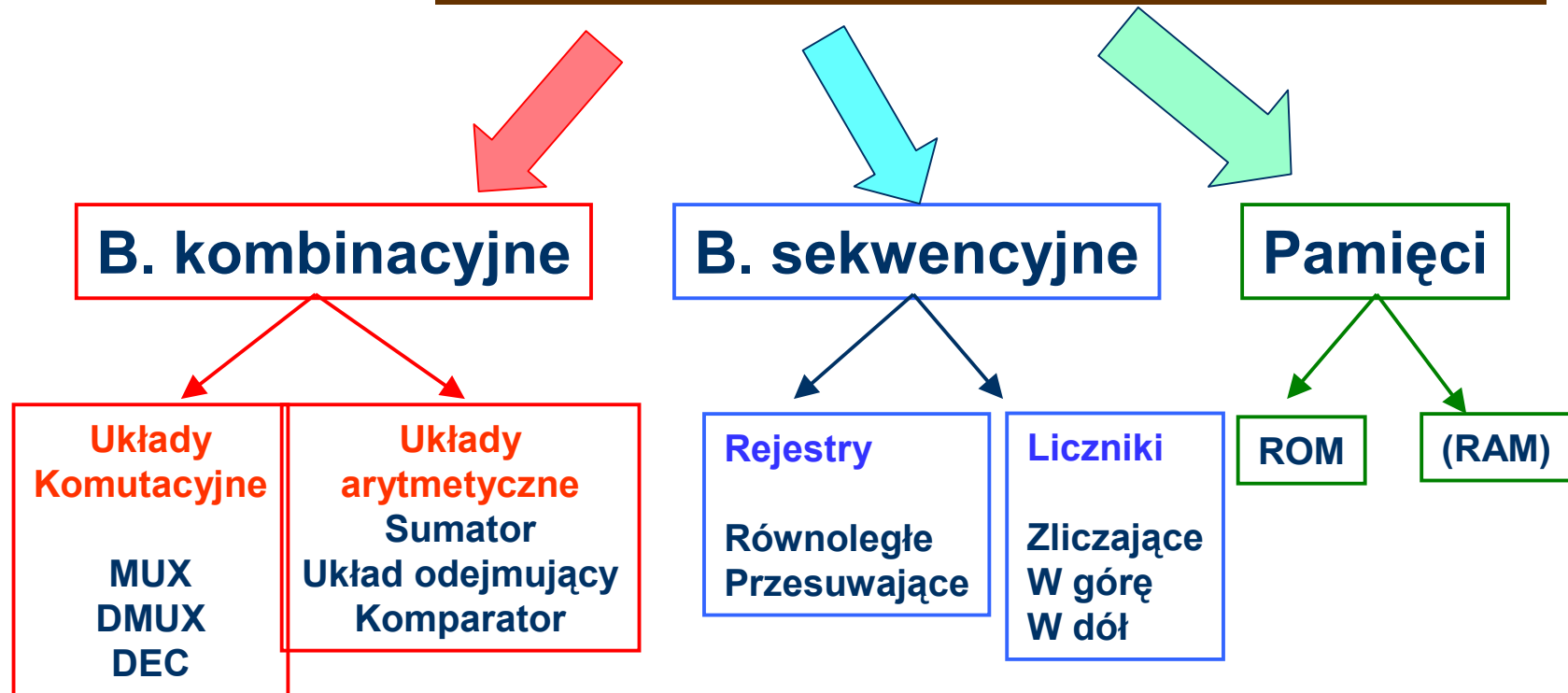
Y – wyjścia sygnałów
reprezentujących dane wyjściowe,

S – wejścia sterujące,

P – wyjścia predykatowe,
sygnalizujące pewne szczególne
stany przetwarzania danych,

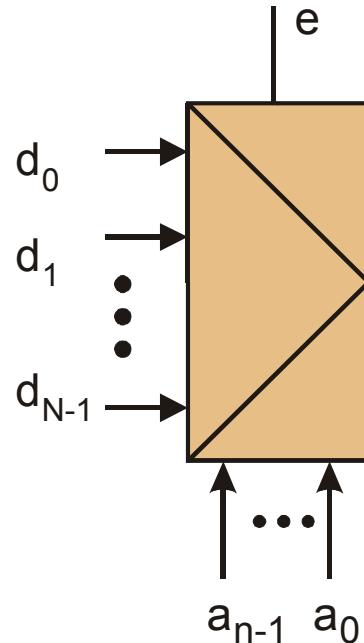
clk – wejście zegarowe

Bloki funkcjonalne



Multiplexer (MUX)

$$N = 2^n$$



$$y = e \sum_{k=0}^{N-1} P_k(A) d_k$$

$$A = (a_{n-1}, \dots, a_j, \dots, a_0)$$

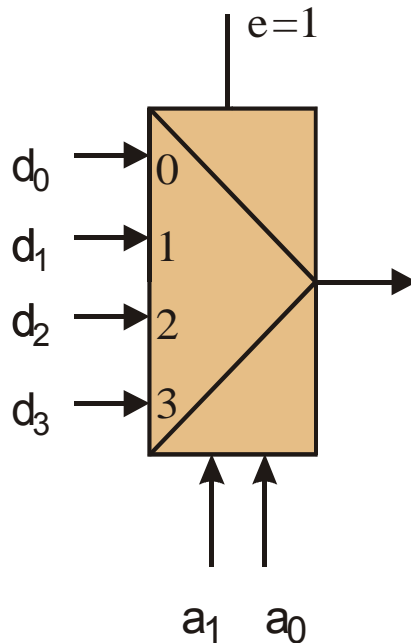
$P_k(A)$ oznacza pełny iloczyn zmiennych a_{n-1}, \dots, a_0 , prostych lub zanegowanych, zgodnie z reprezentacją binarną liczby $k = L(A)$.

Multiplexery

$$y = e \sum_{k=0}^{N-1} P_k(A) d_k$$

Dla $n = 1$ (MUX 2 : 1):

$$y = \bar{a}d_0 + ad_1$$



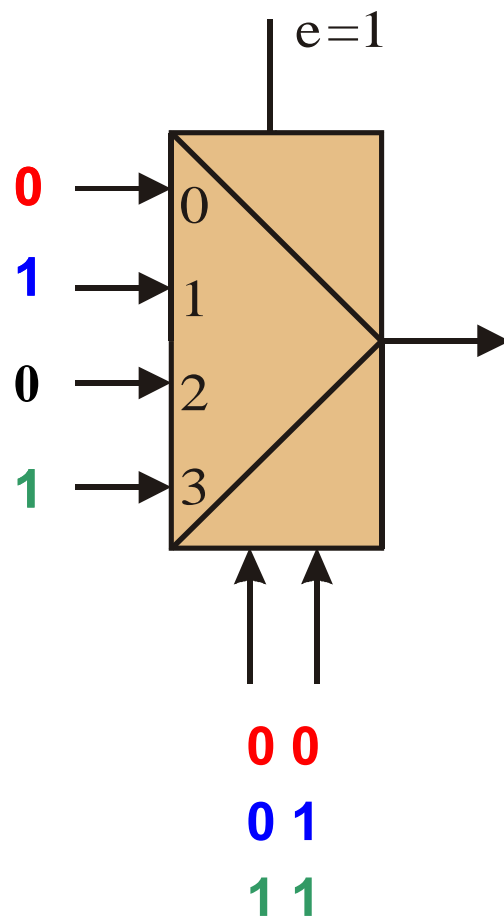
dla $n = 2$ (MUX 4 : 1):

$$y = \bar{a}_1\bar{a}_0d_0 + \bar{a}_1a_0d_1 + a_1\bar{a}_0d_2 + a_1a_0d_3$$

dla $n = 3$ (MUX 8 : 1):

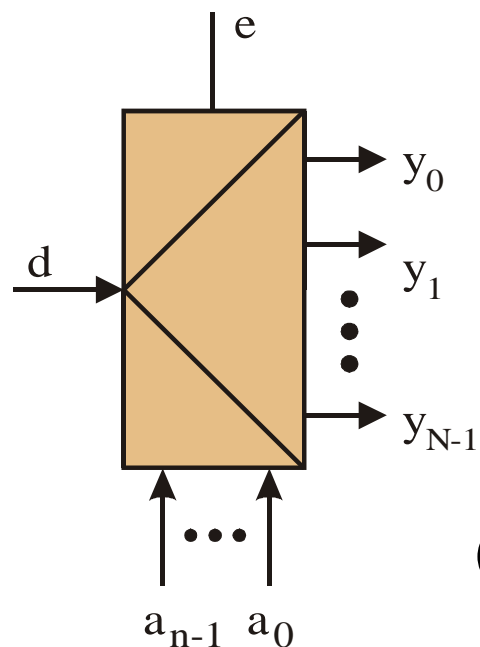
$$y = \bar{a}_2\bar{a}_1\bar{a}_0d_0 + \bar{a}_2\bar{a}_1a_0d_1 + \bar{a}_2a_1\bar{a}_0d_2 + \bar{a}_2a_1a_0d_3 + \\ + a_2\bar{a}_1\bar{a}_0d_4 + a_2\bar{a}_1a_0d_5 + a_2a_1\bar{a}_0d_6 + a_2a_1a_0d_7$$

Multiplexer jako przełącznik



$$y = \bar{a}_1\bar{a}_0d_0 + \bar{a}_1a_0d_1 + a_1\bar{a}_0d_2 + a_1a_0d_3$$

Demultiplekser

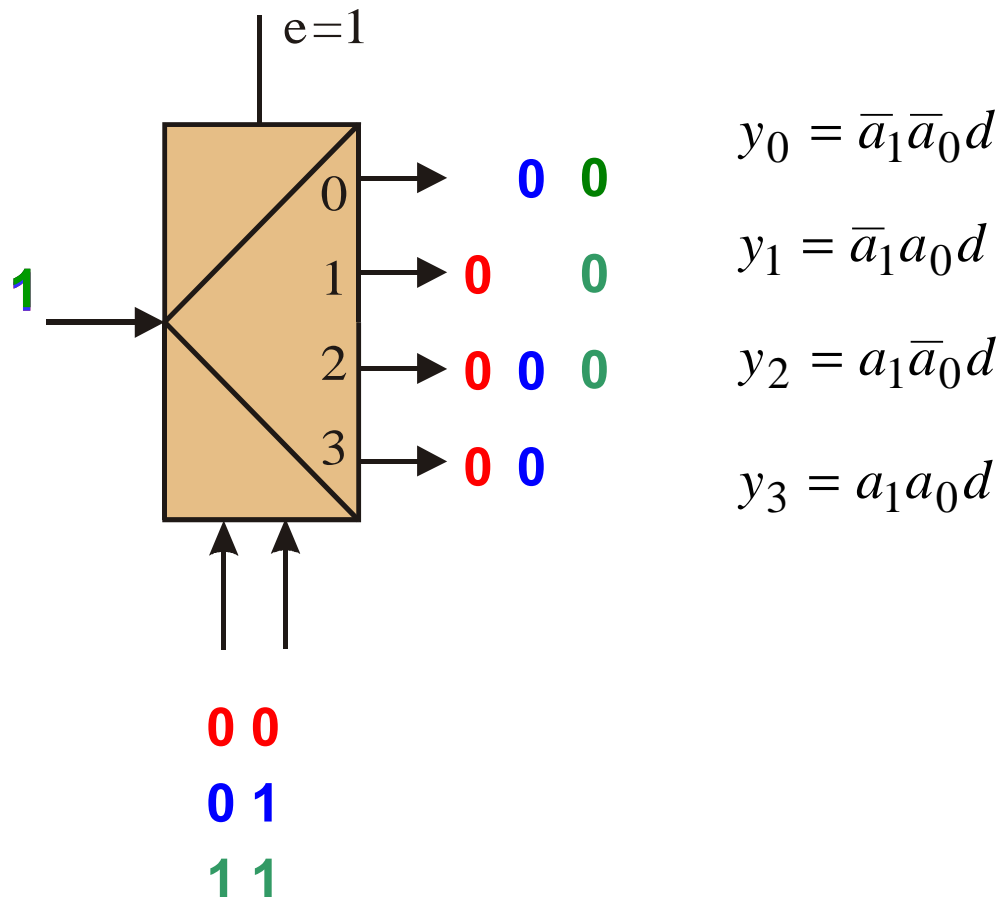


$$y_k = eP_k(A)d$$

$$(N = 2^n)$$

$P_k(A)$ oznacza pełny iloczyn zmiennych a_{n-1}, \dots, a_0 , prostych lub zanegowanych, zgodnie z reprezentacją binarną liczby $k = L(A)$.

Demultiplekser jako przełącznik



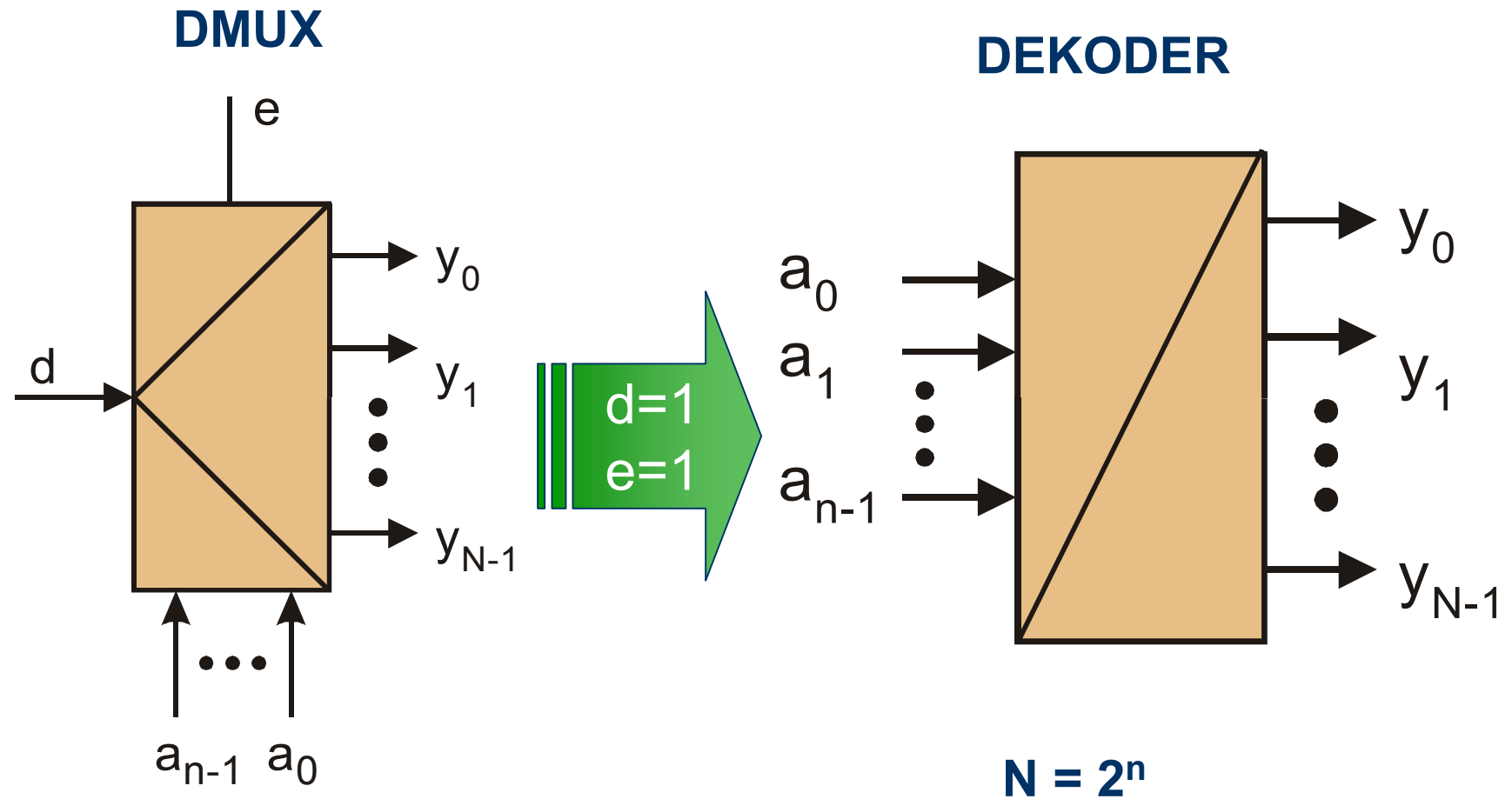
$$y_0 = \bar{a}_1 \bar{a}_0 d$$

$$y_1 = \bar{a}_1 a_0 d$$

$$y_2 = a_1 \bar{a}_0 d$$

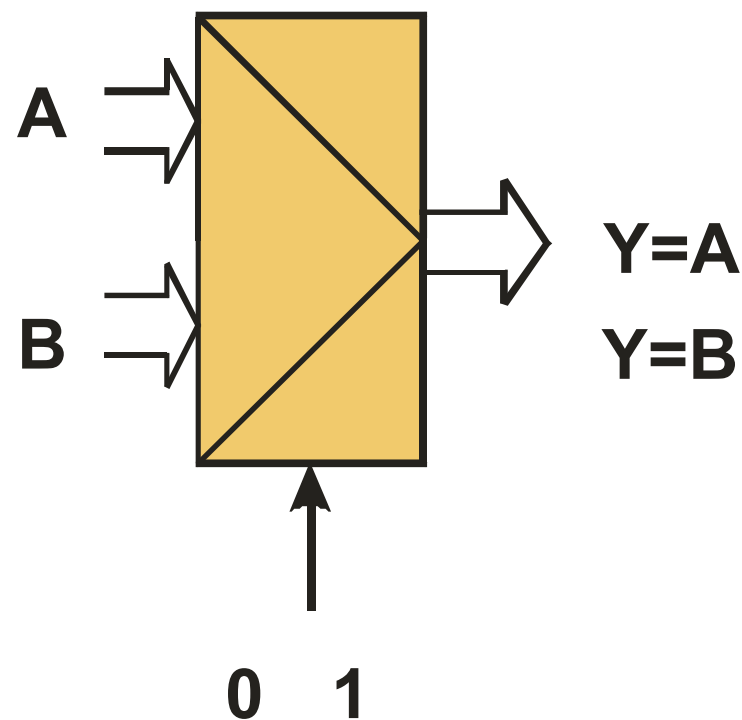
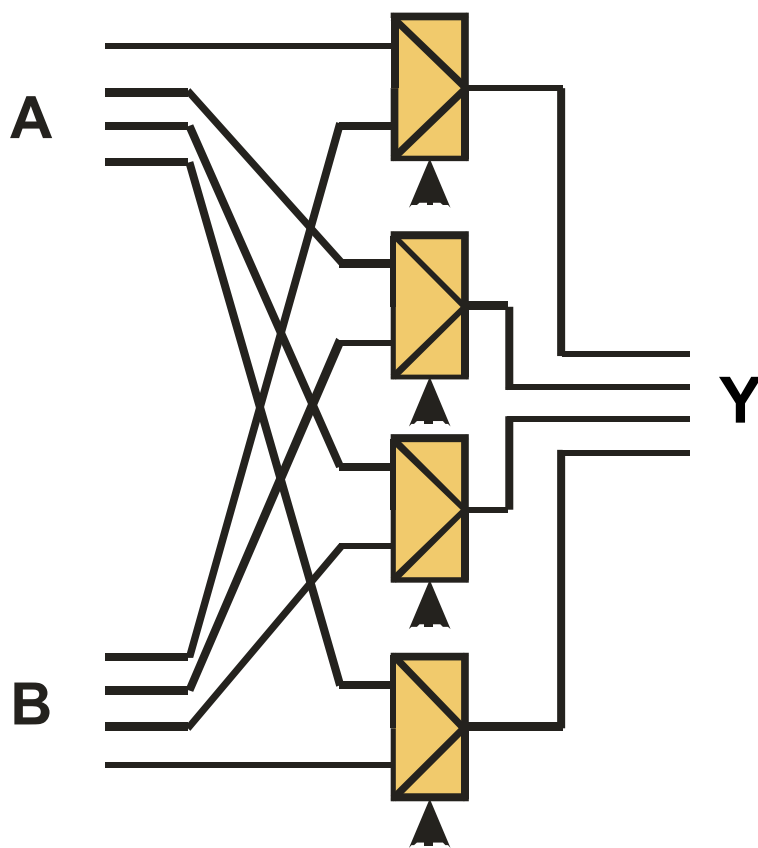
$$y_3 = a_1 a_0 d$$

Dekoder

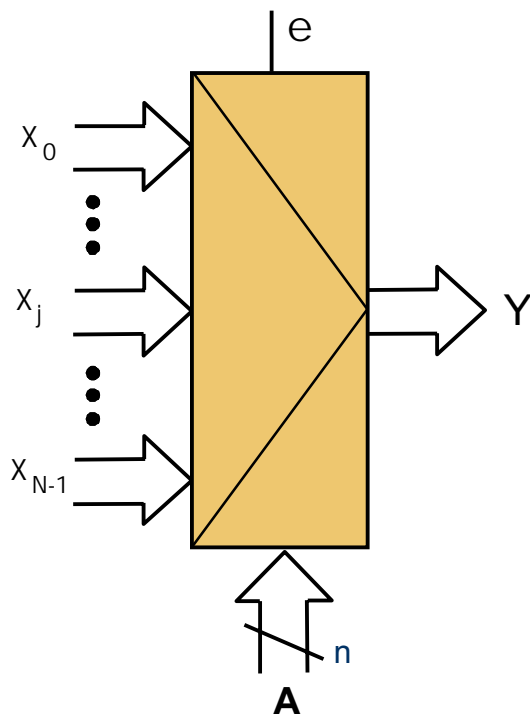


Multipleksery grupowe

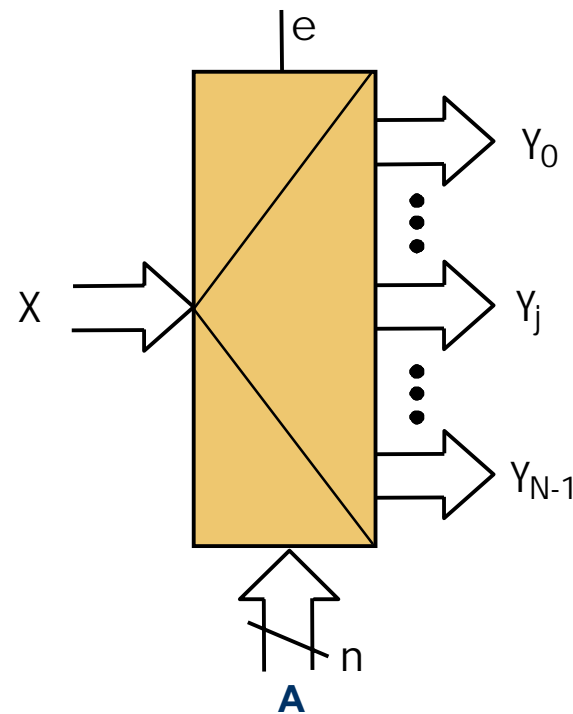
MUX-y i DMUX-y można przystosować do przełączania (komutacji) sygnałów wielobitowych (grupowych)



Bloki komutacyjne



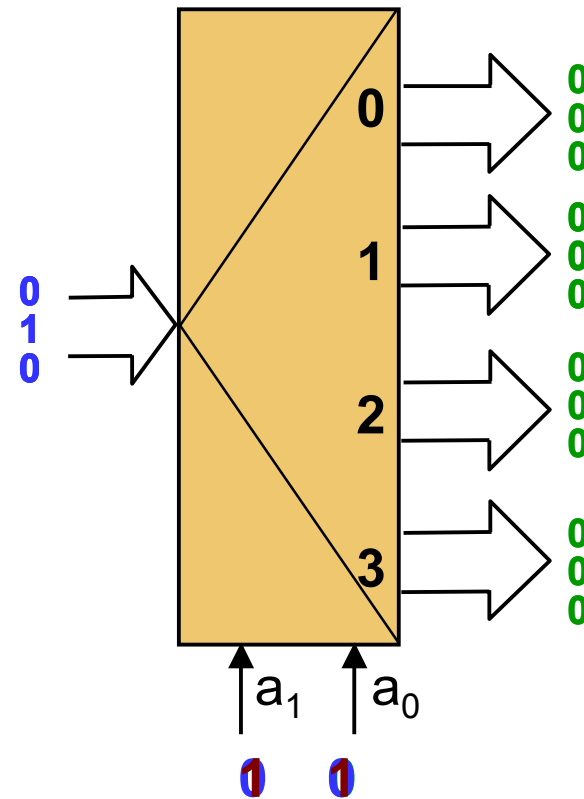
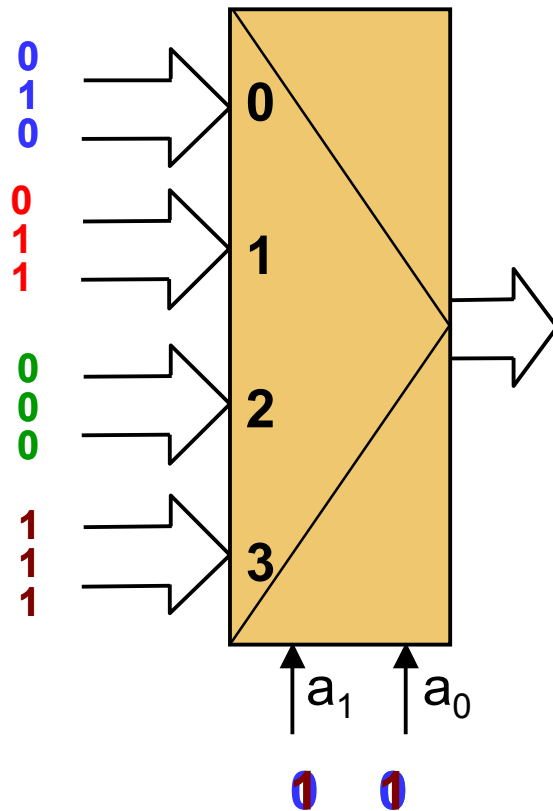
Multiplexer służy do wybierania jednego z wielu słów wejściowych i przesyłania go na wyjście. Na wyjściu Y pojawia się słowo wejściowe wskazane adresem A (wg naturalnego kodu binarnego).



Demultiplexer służy do przesyłania słowa X wejściowego na jedno z wielu wyjść; numer tego wyjścia jest równy aktualnej wartości adresu.

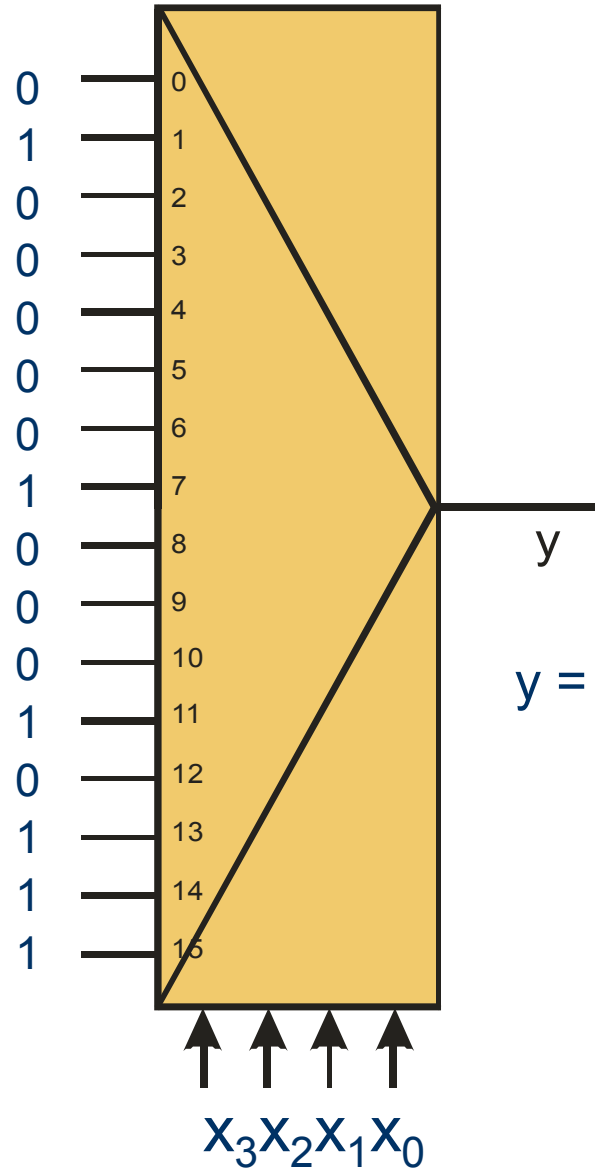
Bloki komutacyjne

Najważniejsze zastosowanie:



Inne zastosowania...

**Zastosowanie MUX do realizacji
funkcji boolowskich**



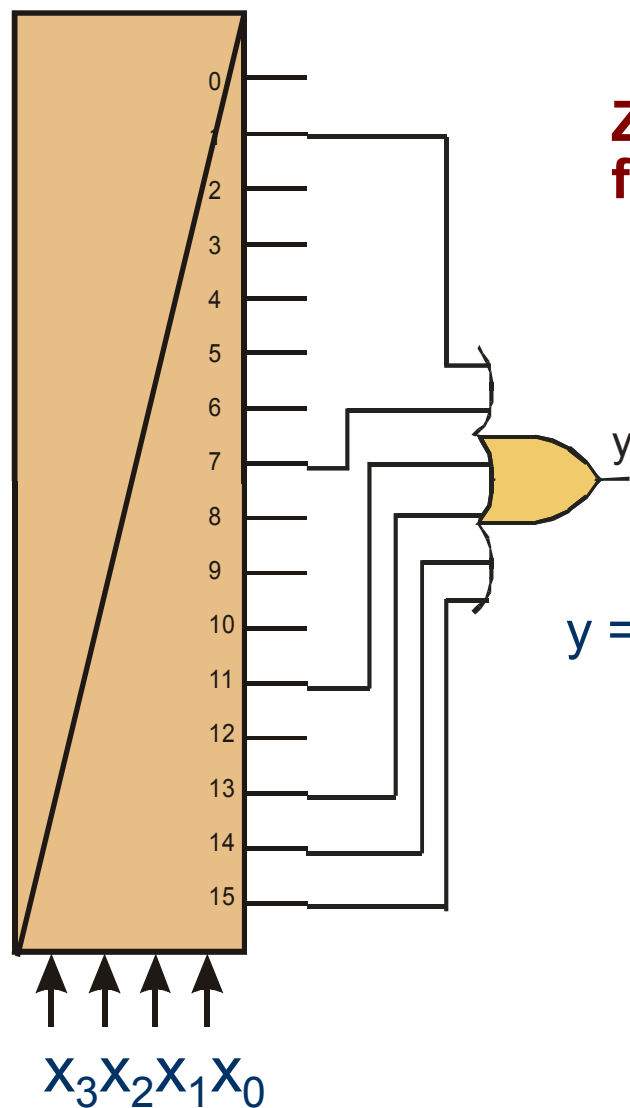
$$y = \Sigma(1, 7, 11, 13, 14, 15)$$

I
T
P
W

ZPT

Inne zastosowania...

Zastosowanie dekodera do realizacji
funkcji boolowskich



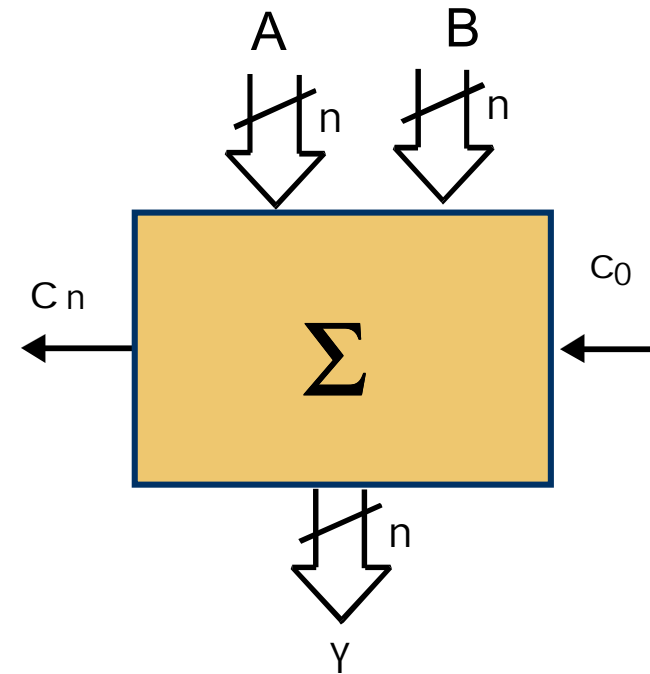
$$y = \Sigma(1,7,11,13,14,15)$$

... należy odłożyć do kosza!

Sumatory

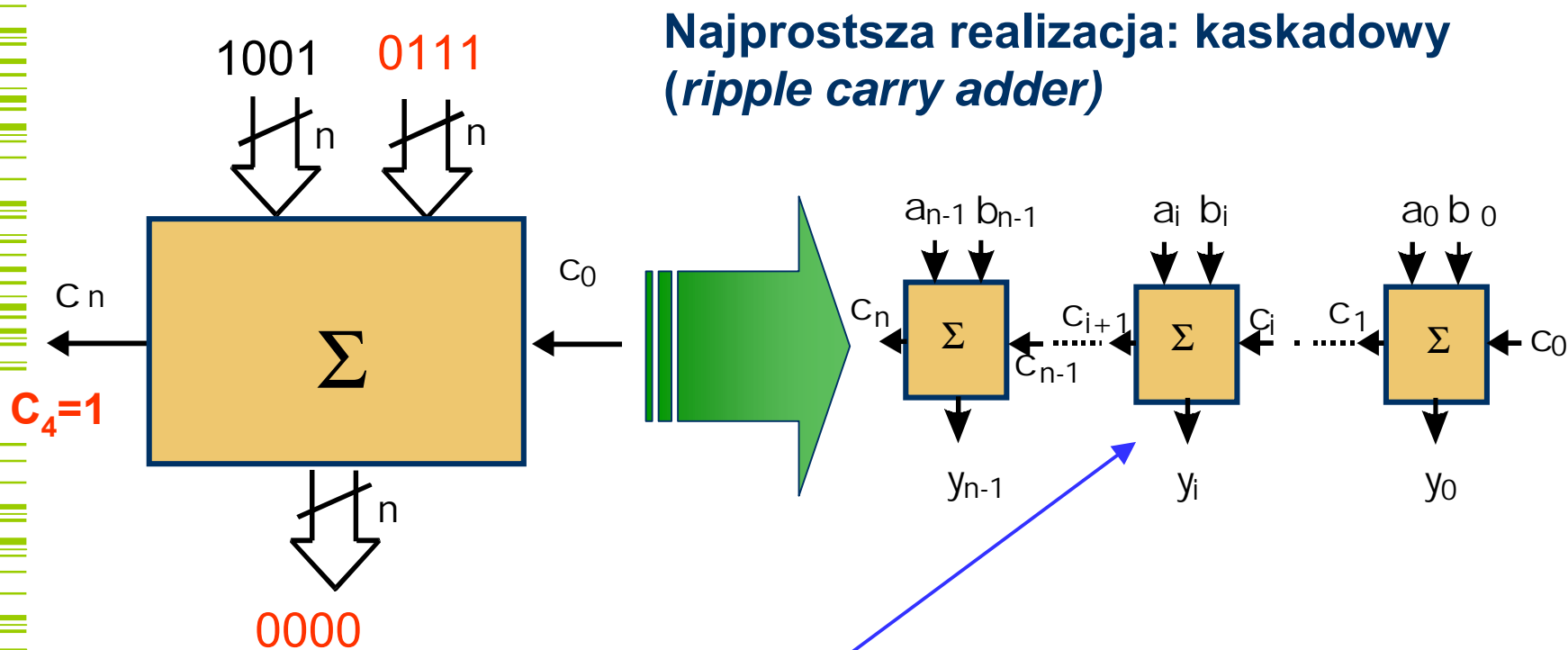
**Sumator – podstawowy
BF powszechnie
stosowany w technice
cyfrowej**

Inne układy
arytmetyczne:
układy odejmowania
układy mnożące
układy dzielenia



...są budowane z sumatorów

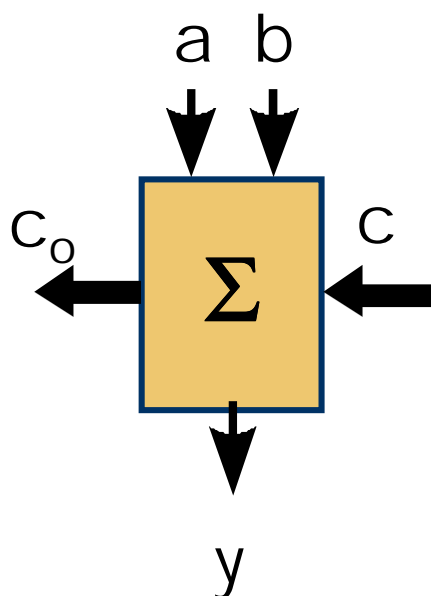
Najprostszy sumator



Jak jest zbudowane pojedyncze ogniwo?



Funkcje logiczne sumatora



a	b	c	c _o	y
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

$$y_i = a_i \oplus b_i \oplus c_i$$

$$c_{i+1} = a_i b_i \vee c_i (a_i \vee b_i)$$

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

$$y = cab \vee c\bar{a}\bar{b} \vee \bar{c}ab \vee \bar{c}\bar{a}b$$

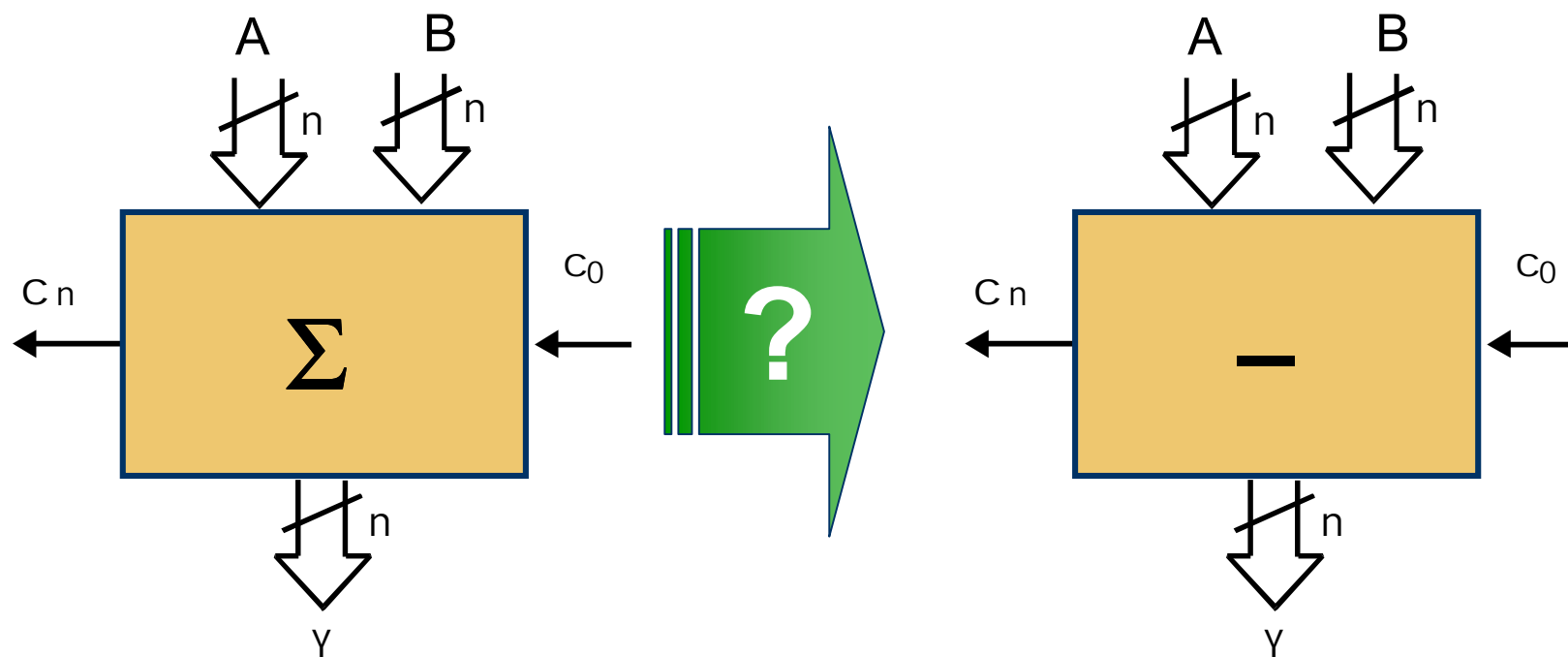
$$= c(a \oplus b) \vee \bar{c}(a \oplus b)$$

$$y = c \oplus a \oplus b$$

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

$$c_o = ab \vee c(a \vee b) = ab \vee c(a \oplus b)$$

Co z odejmowaniem?



Reprezentacje liczb – NKB/U2

$A = \langle a_{n-1}, \dots, a_j, \dots, a_0 \rangle$ gdzie $a_j \in \{0,1\}$

NKB:

$$A_D = L(A_{\text{NKB}}) = \sum_{j=0}^{n-1} a_j 2^j$$

U2:

$$A_D = L(A_{\text{U2}}) = -a_{n-1} \cdot 2^{n-1} + \sum_{j=0}^{n-2} a_j 2^j$$

Kod U2

$A_{U2} = \langle a_{n-1}, \dots, a_j, \dots, a_0 \rangle$, gdzie $a_j \in \{0, 1\}$

$$A_D = L(A_{U2}) = -a_{n-1} \cdot 2^{n-1} + \sum_{j=0}^{n-2} a_j 2^j$$

Bit a_{n-1} można interpretować jako bit znaku.

Jeśli $a_{n-1} = 0$, to liczba jest dodatnia;

jeśli $a_{n-1} = 1$ to liczba jest ujemna; pozostałe bity stanowią uzupełnienie (różnicę) wartości liczby do najwyższej potęgi liczby 2

$$\langle 0101 \rangle_{U2} = +5_D; \langle 1011 \rangle_{U2} = -5_D$$

$$\text{Zakres: } -2^{n-1} \leq A_D \leq 2^{n-1} - 1$$

Sumator/układ odejmujący

Jak z sumatora zbudować układ odejmujący?

$$Y = A - B = A + (-B|_{U_2})$$

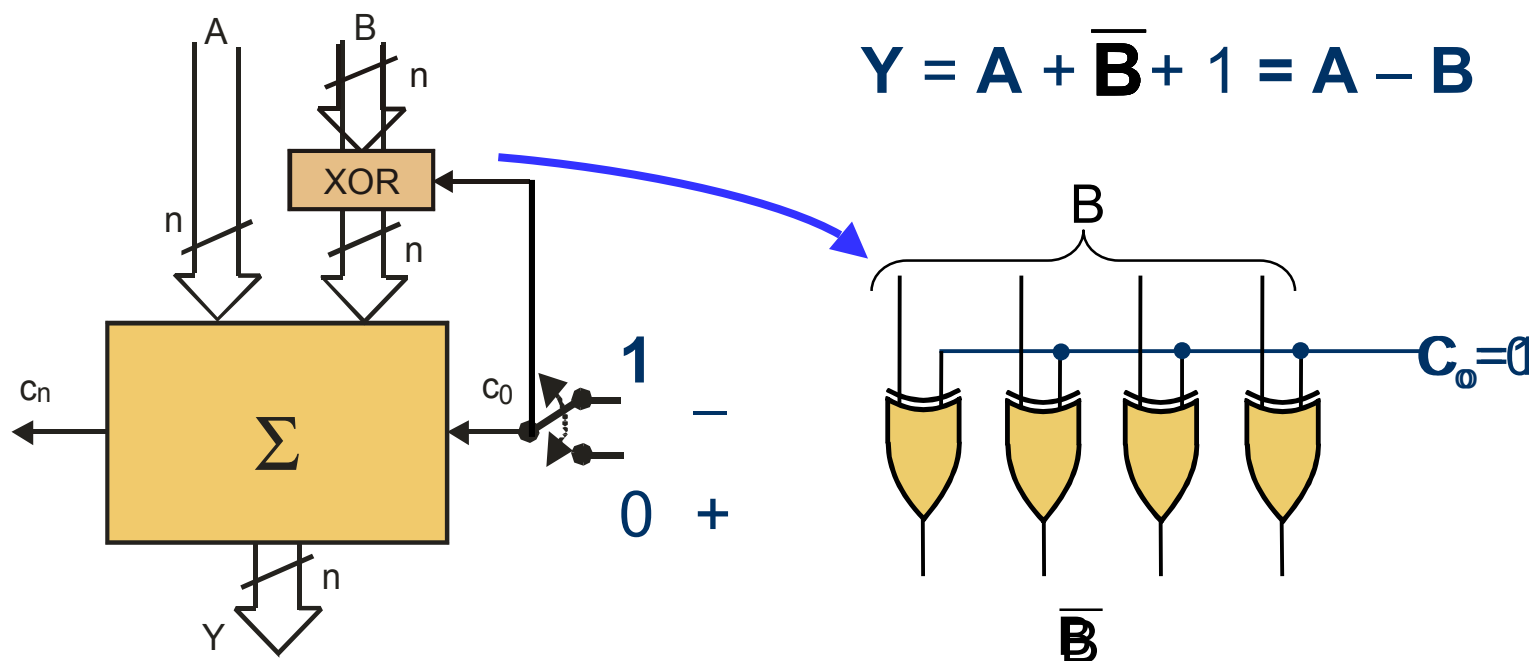
$$-B|_{U_2} = \overline{B} + 1 = B \oplus 1 + 1$$

Dla $c_0 = 0$

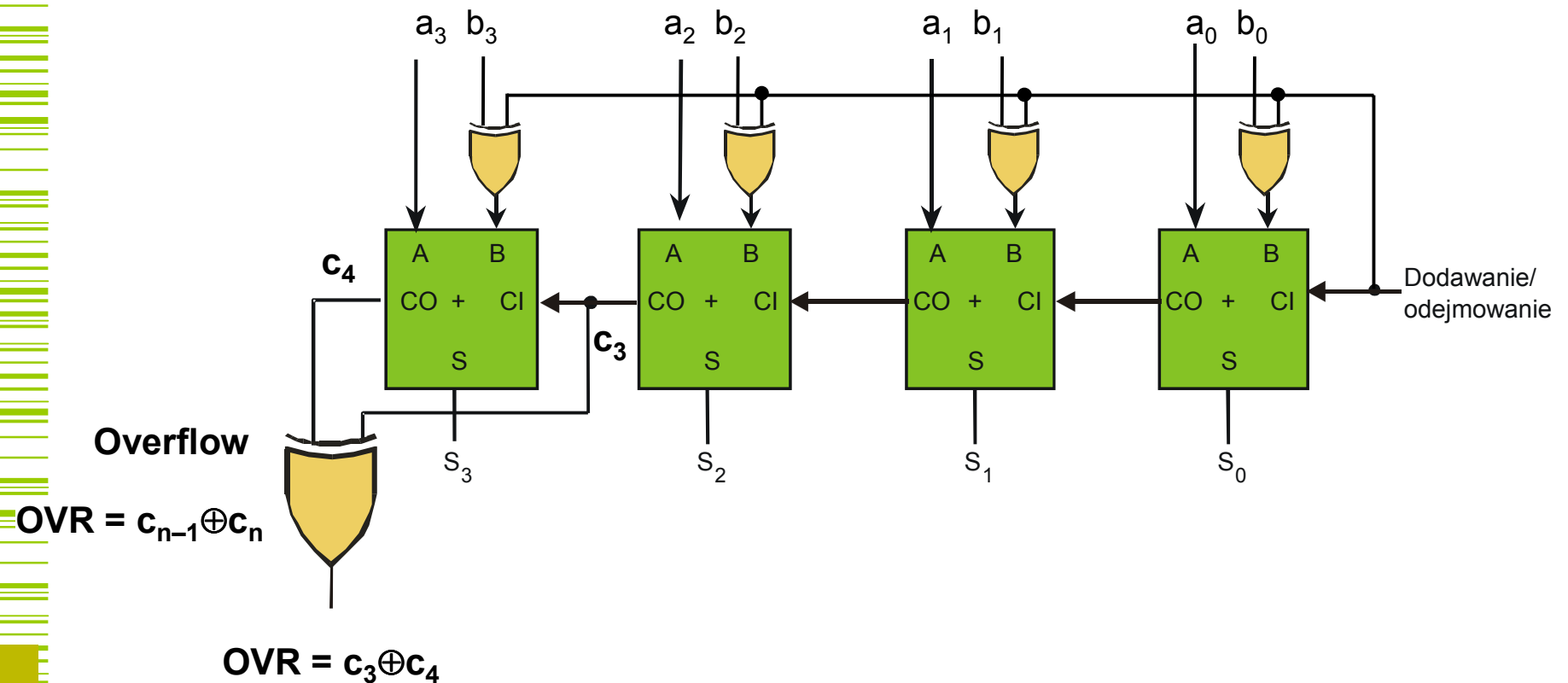
$$Y = A + B \oplus 0 + 0 = A + B$$

Dla $c_0 = 1$

$$Y = A + \overline{B} + 1 = A - B$$



Sumator/układ odejmujący



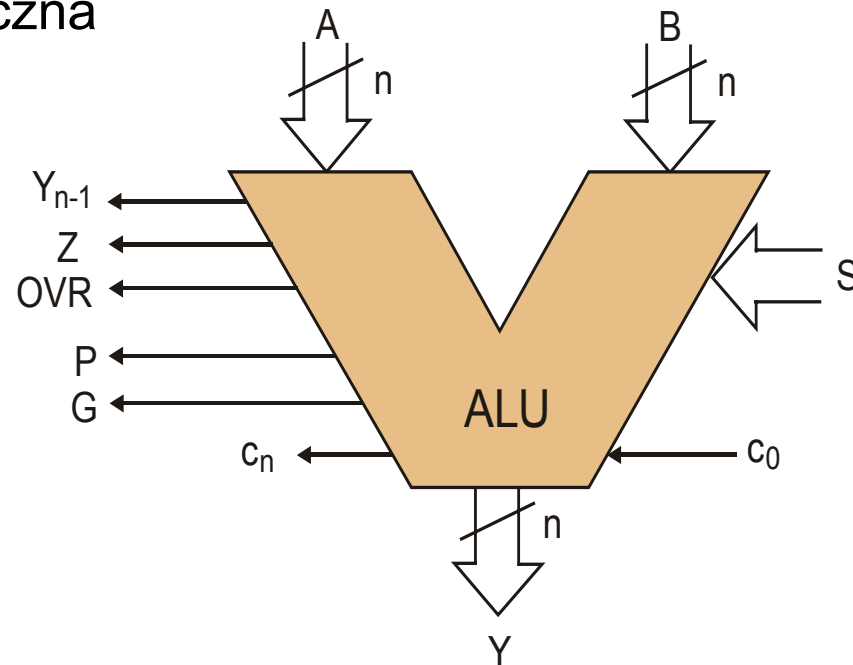
Zastosowania

Jednostka arytmetyczno-logiczna

Arytmometr (układ wykonawczy: mikrokontrolera, procesora sygnałowego)

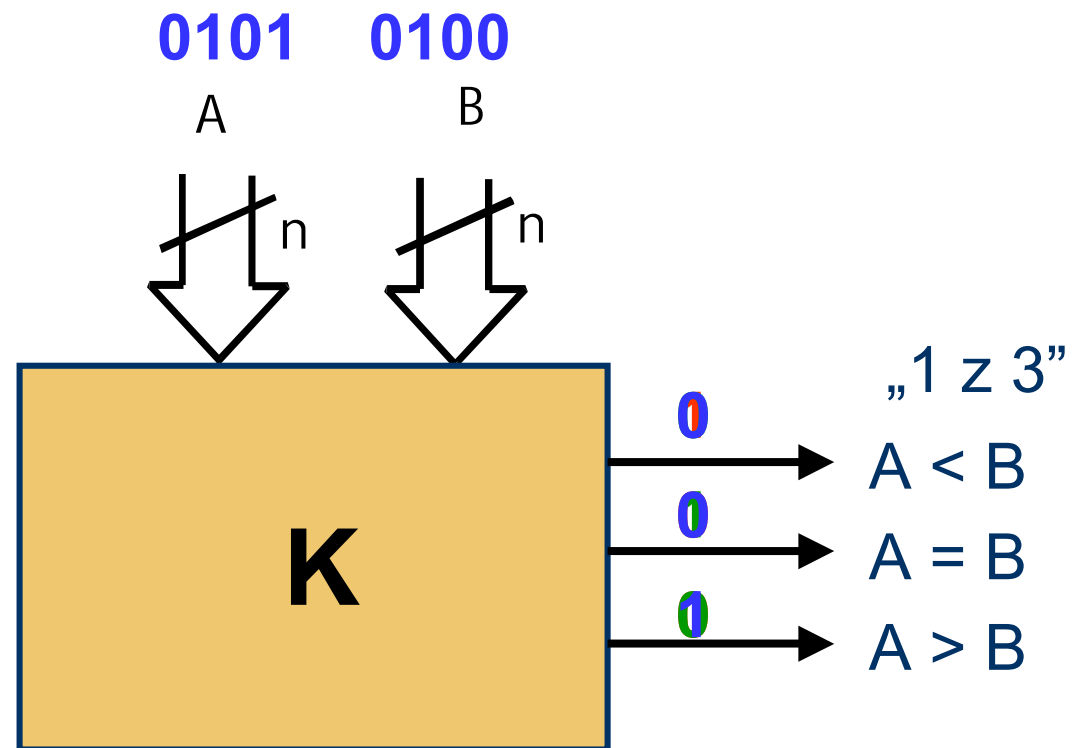
Inne układy arytmetyczne:

układy mnożące
układy
kryptograficzne



...są budowane z sumatorów

Komparator

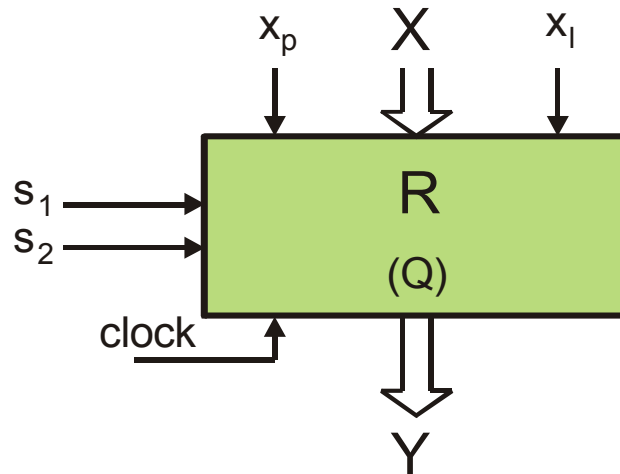


Sekwencyjne bloki funkcjonalne

$Y := X$
 $Y := Y$

LOAD
HOLD

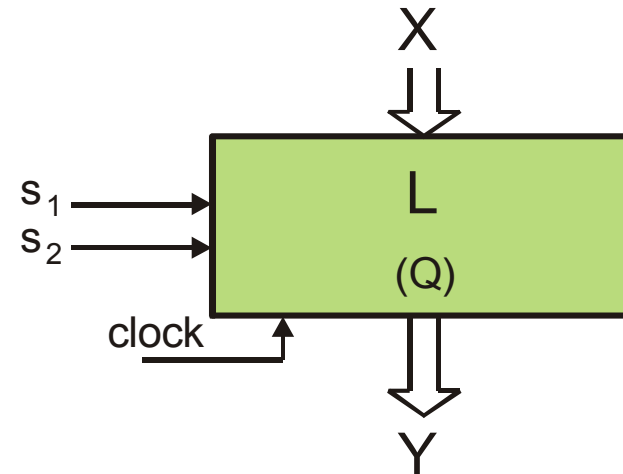
Rejestry



$Y := \text{SHR}(x_p, Y)$
 $Y := \text{SHL}(Y, x_l)$

Liczniki

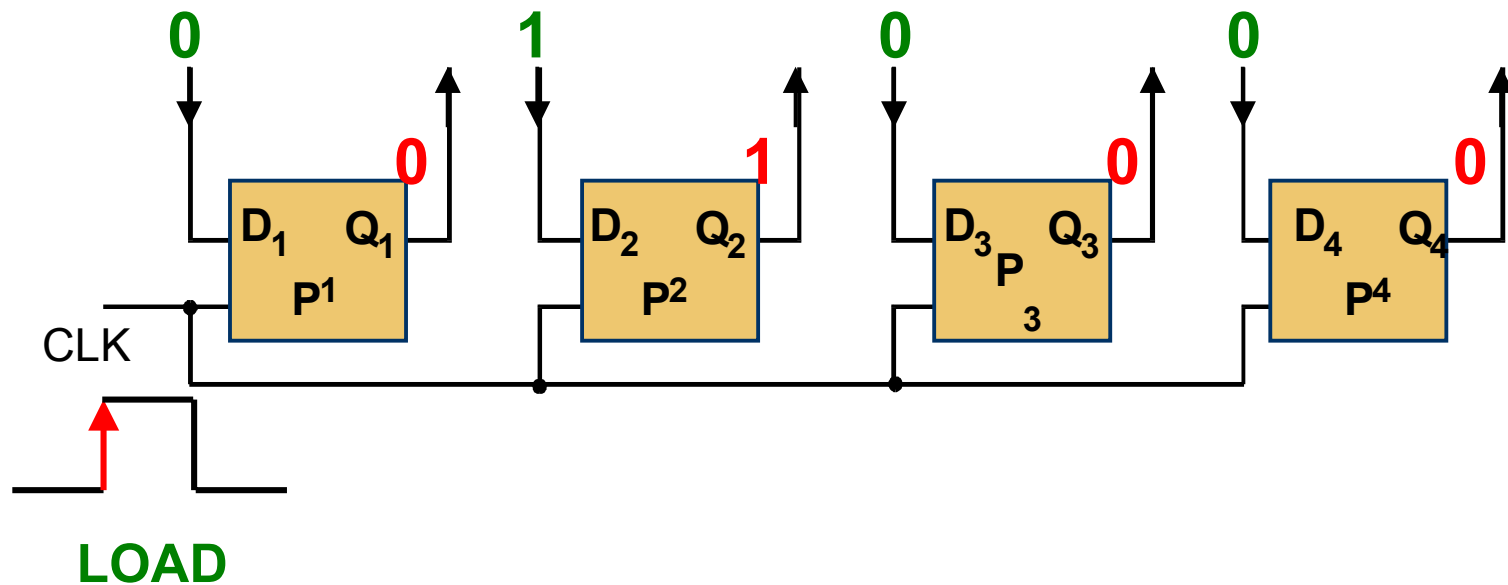
$Y := \langle 0 \dots 0 \rangle$ RESET
(CLEAR)



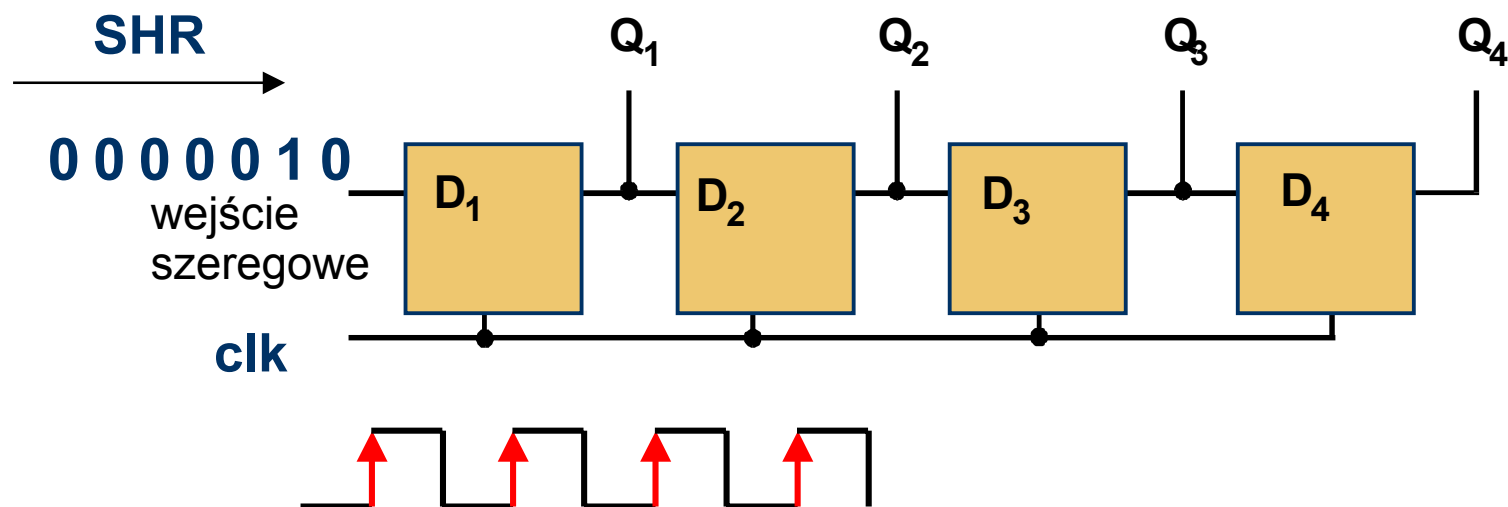
$Y := Y + 1 = \text{INC}(Y)$
 $Y := Y - 1 = \text{DEC}(Y)$

Prosty rejestr

Rejestr zbudowany z przerzutników
– ładowanie (load) i pamiętanie

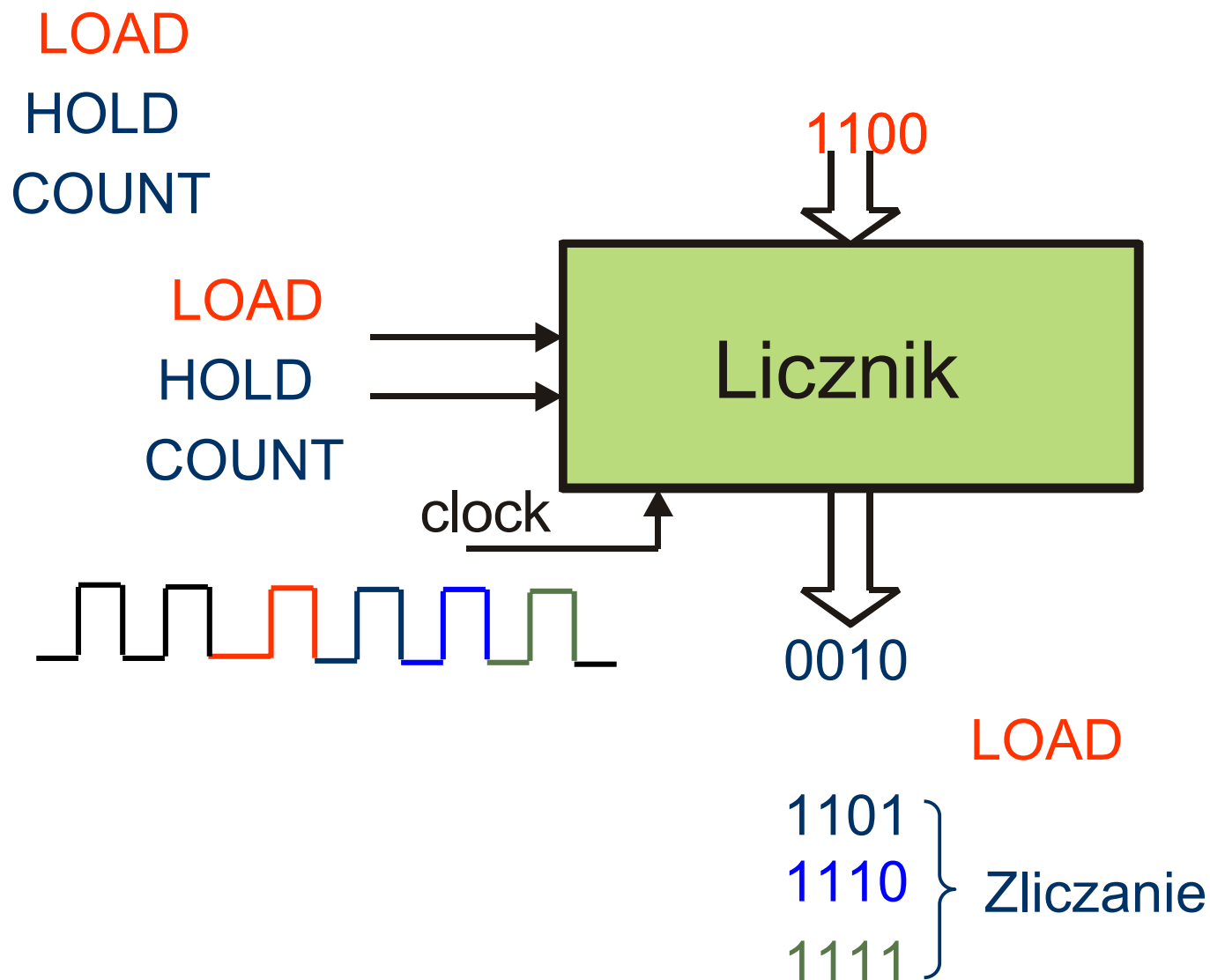


Rejestr przesuwający

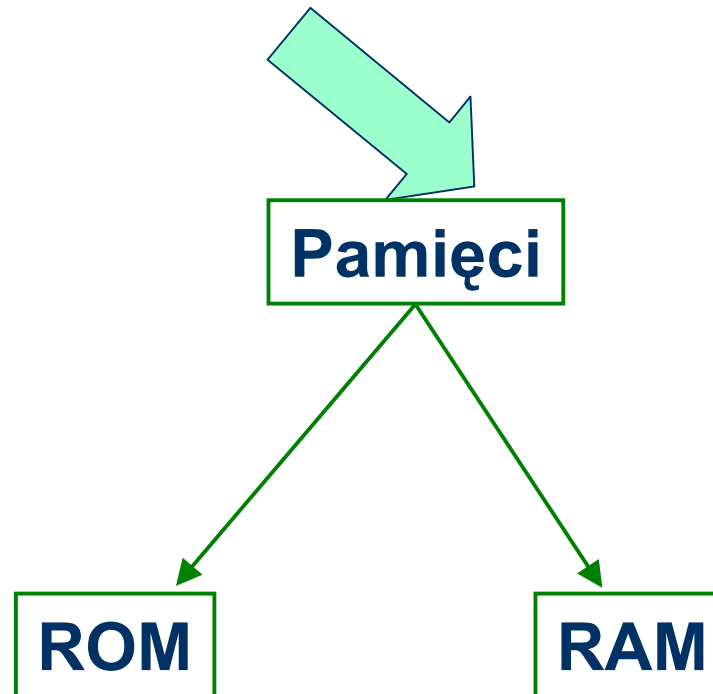


WE	Q ₁	Q ₂	Q ₃	Q ₄
0	0	0	0	0
1	1	0	0	0
0	0	1	0	0
0	0	0	1	0
0	0	0	0	1
0	0	0	0	0

Mikrooperacje licznika



Bloki funkcjonalne



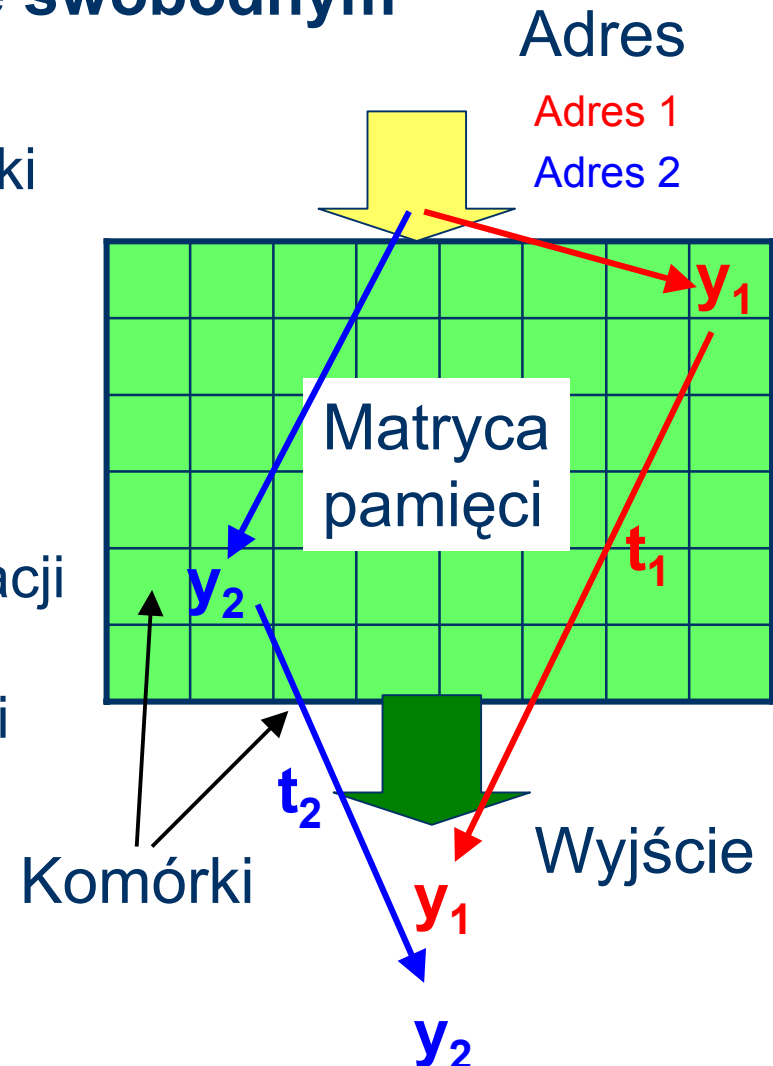
Pamięci

Pamięci o dostępie swobodnym

Czas dostępu t - czas, jaki upływa od momentu podania adresu komórki na wejścia adresowe pamięci do momentu ustalenia informacji na wyjściu pamięci.

Dostęp swobodny (*Random Access*) - **czas dostępu** do informacji zapisanej w komórce pamięci jest niezależny od położenia tej komórki w matrycy pamięci.

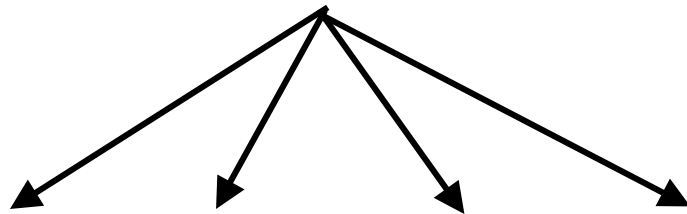
$$t_1 = t_2$$



Pamięci

ROM

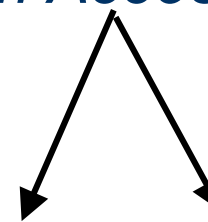
(Read Only Memory)



ROM, PROM, EPROM, E²PROM

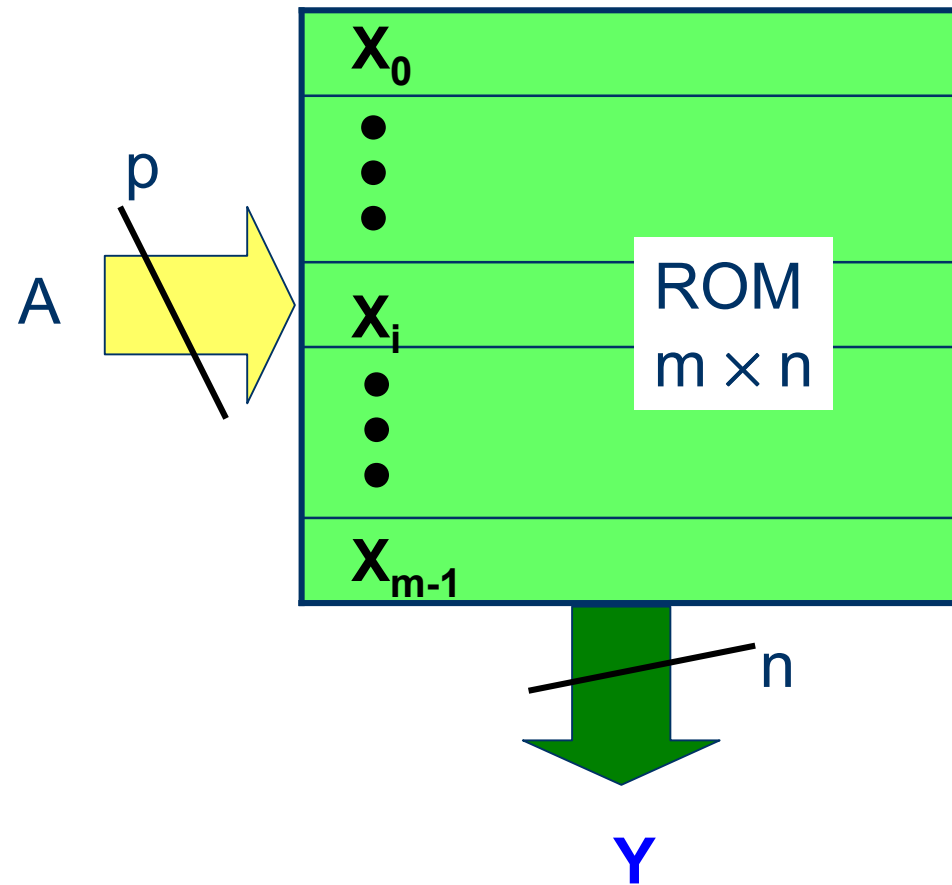
RAM

(Random Access memory)



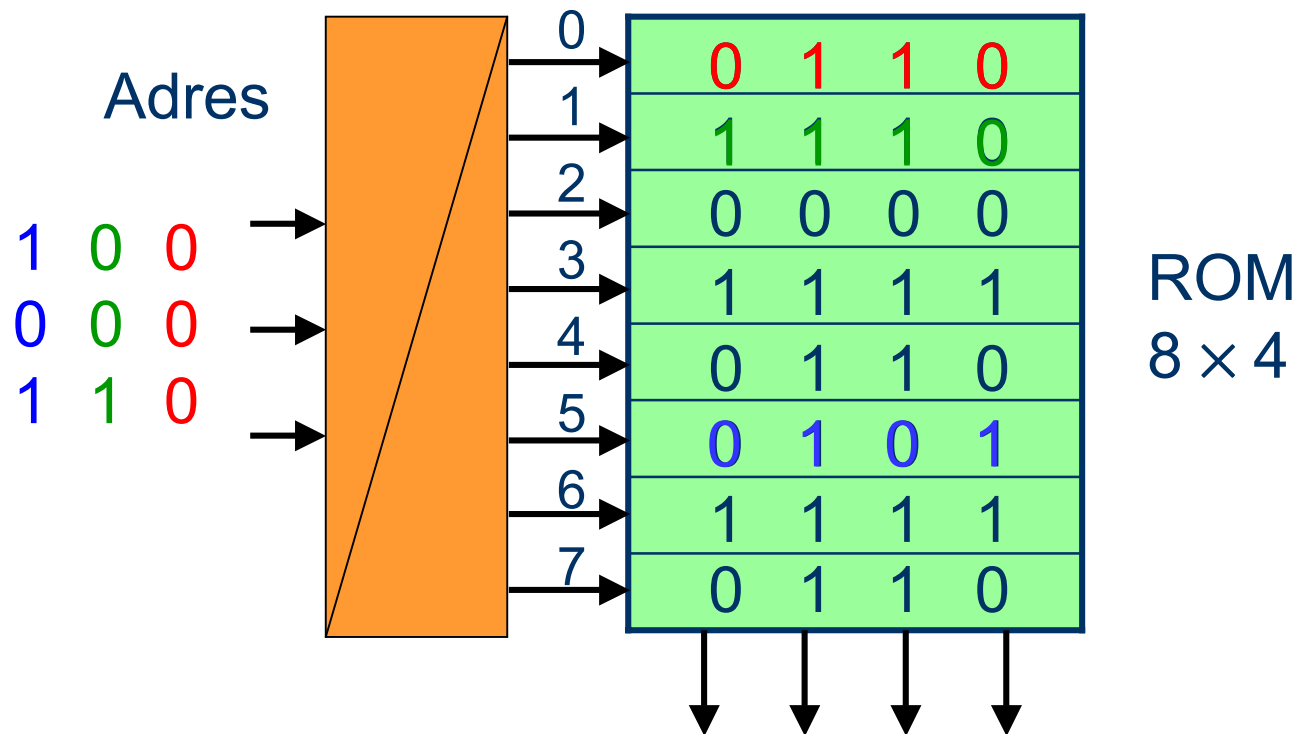
DRAM, SRAM
(non-volatile)

Pamięci typu ROM

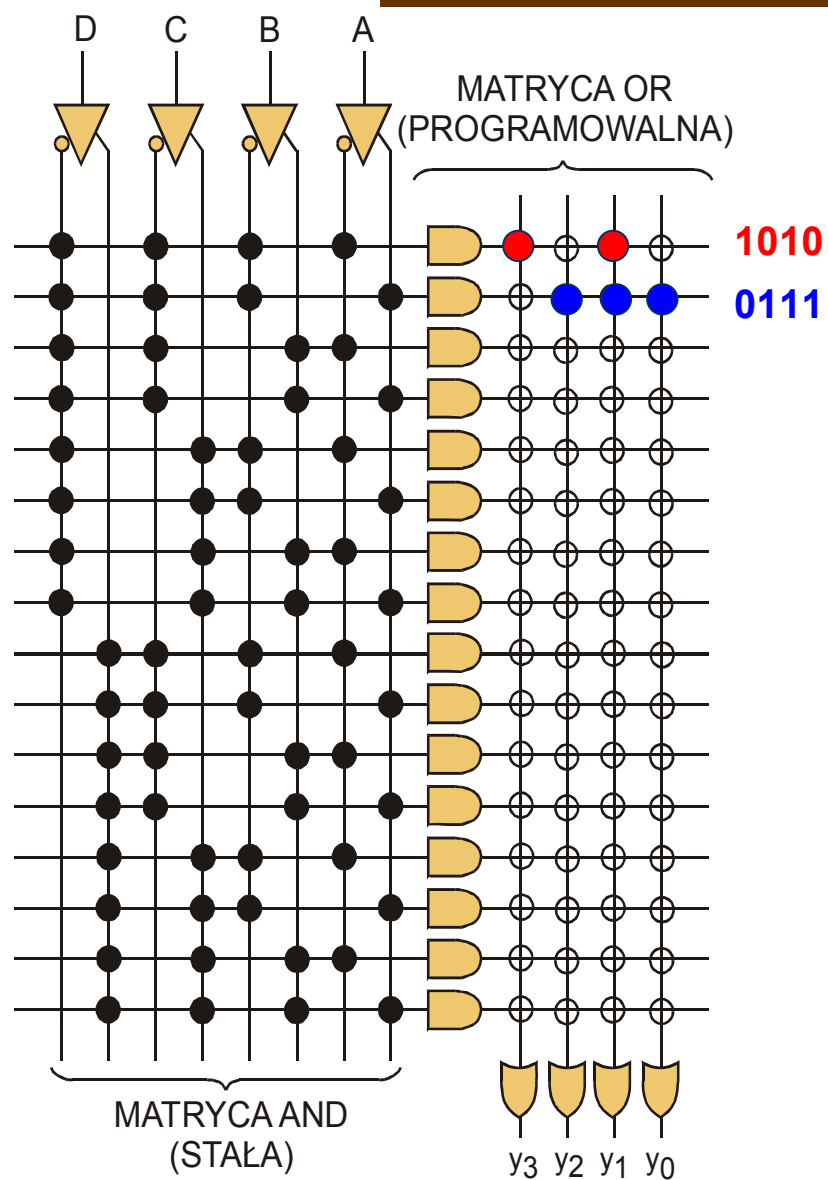


ROM – uniwersalny układ kombinacyjny

Pamięci typu ROM

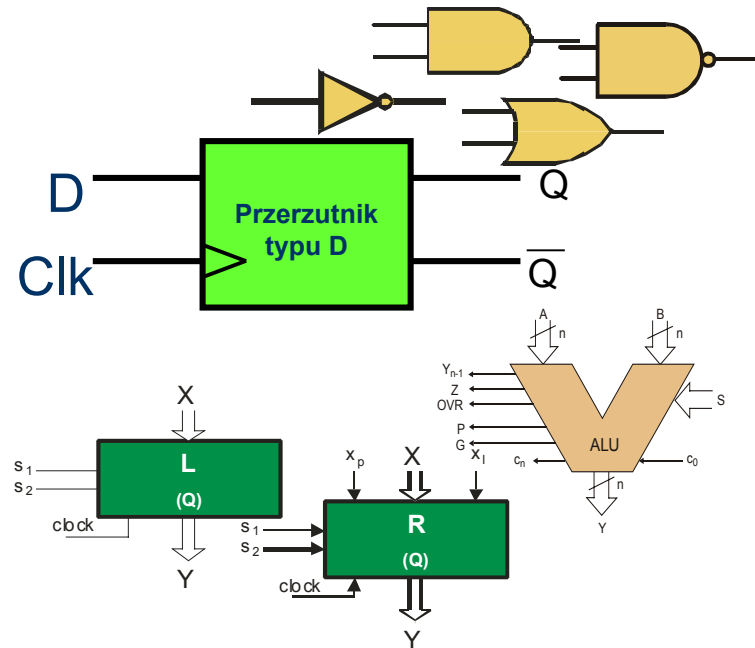


Pamięci typu ROM



(struktura)

Synteza strukturalna



Bloki funkcjonalne są stosowane w syntezie:

- 👉 **układów wykonawczych (operacyjnych),**
- 👉 **układów sterujących.**

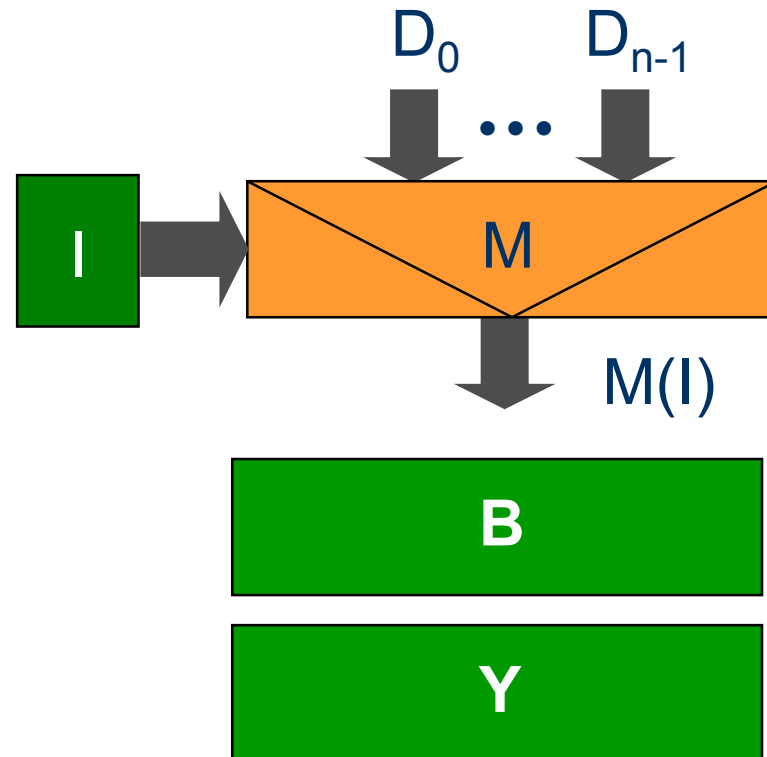
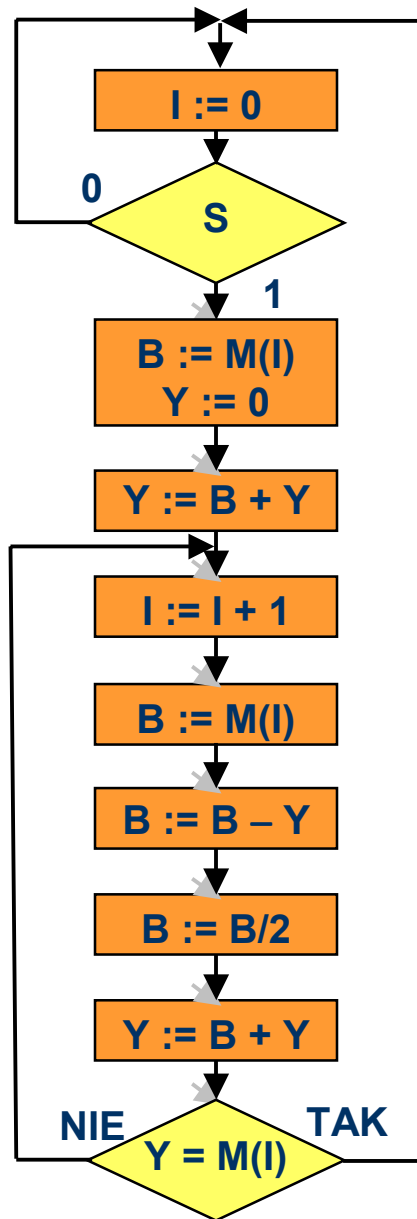
Synteza strukturalna

Bloki funkcjonalne są stosowane w syntezie układów wykonawczych (operacyjnych).

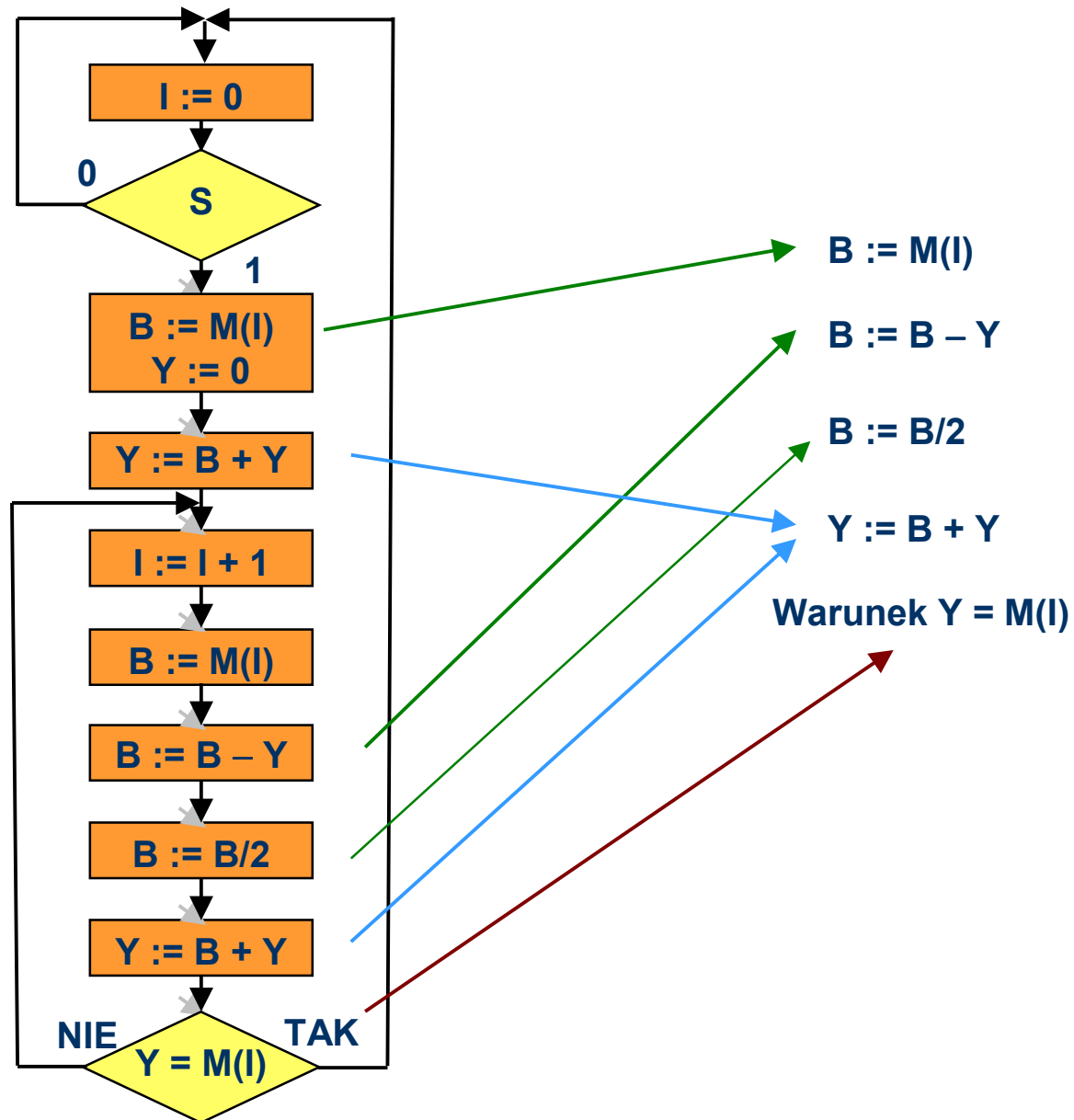
1. Analiza algorytmu pracy układu (sieci działań)
– zmienne, operacje
2. Dobór bloków funkcjonalnych do przechowywania zmiennych i wykonywania operacji
3. Dobór bloków komutacyjnych

Przykład

Rozważmy przykład syntezy układu cyfrowego pobierającego dane z wejść D_0, \dots, D_{n-1} w celu ich przetworzenia w rejestrach B i Y . Układ operacyjny ma przetwarzać informację $M(I)$ pobieraną z n źródeł D_0, \dots, D_{n-1} za pośrednictwem multipleksera M adresowanego licznikiem I .

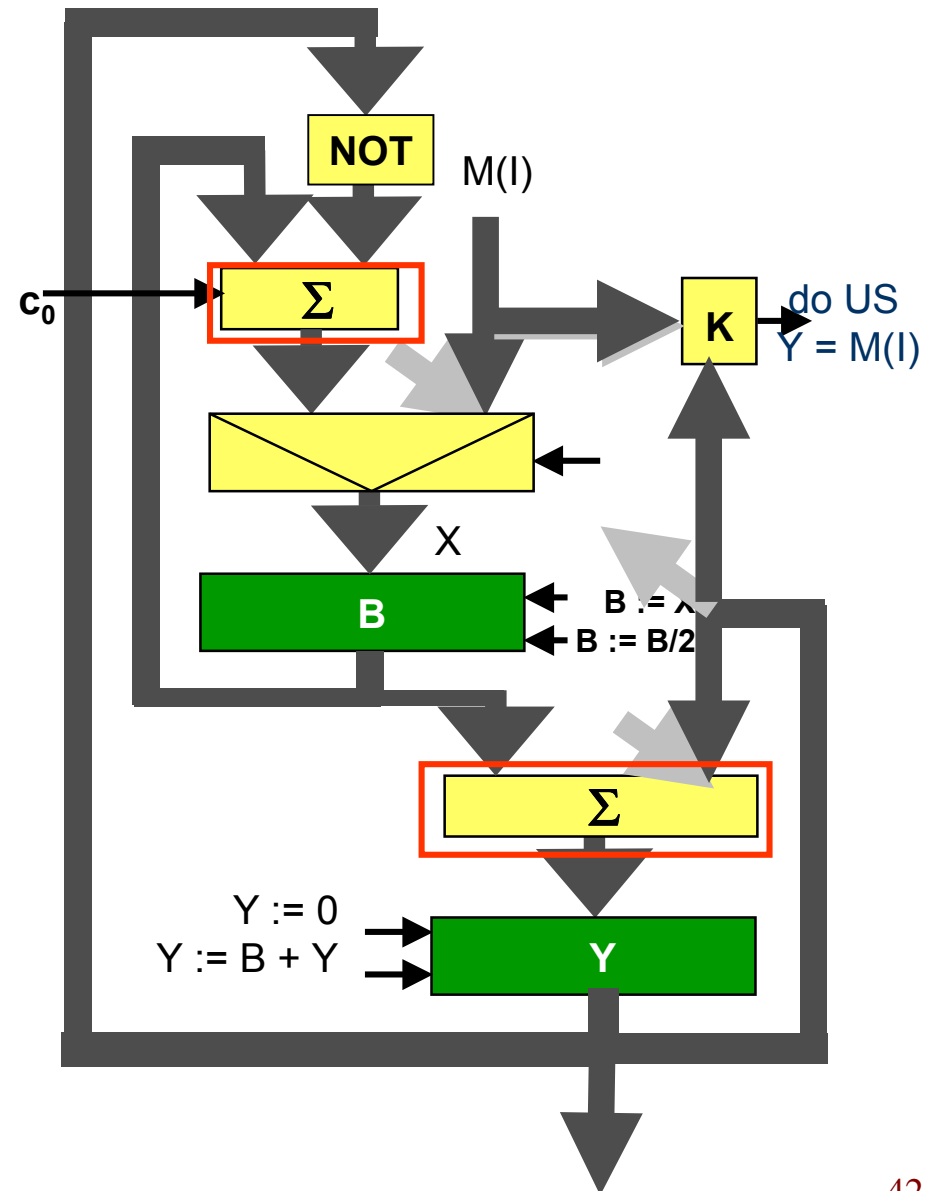


Przykład



Przykład c.d.

$B := M(I)$
 $B := B - Y$
 $B := B/2$ (SHR(B))
 $Y := B + Y$
 Warunek $Y = M(I)$
 $B := B - Y$
 $Y := B + Y$

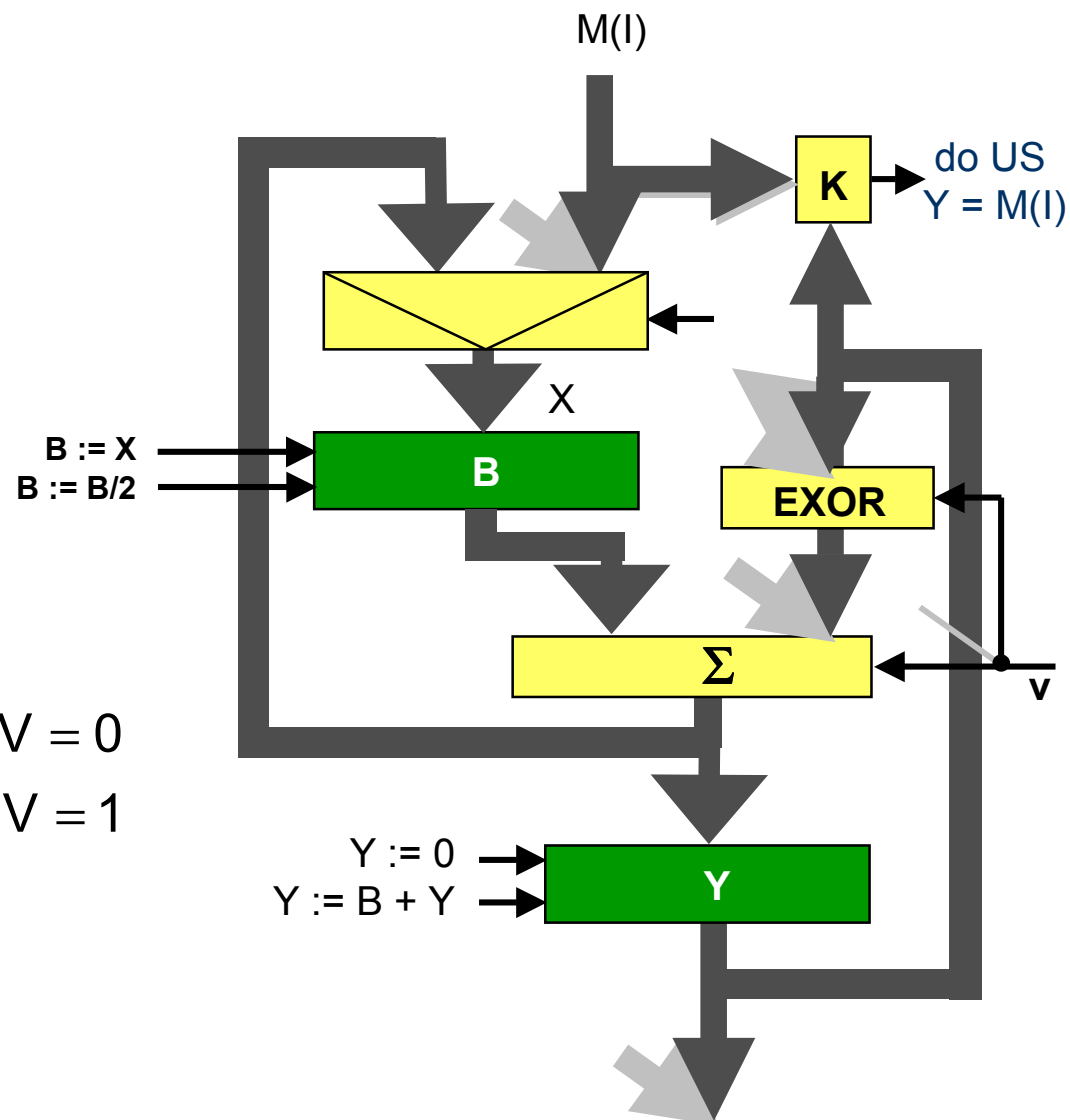


Przykład c.d.

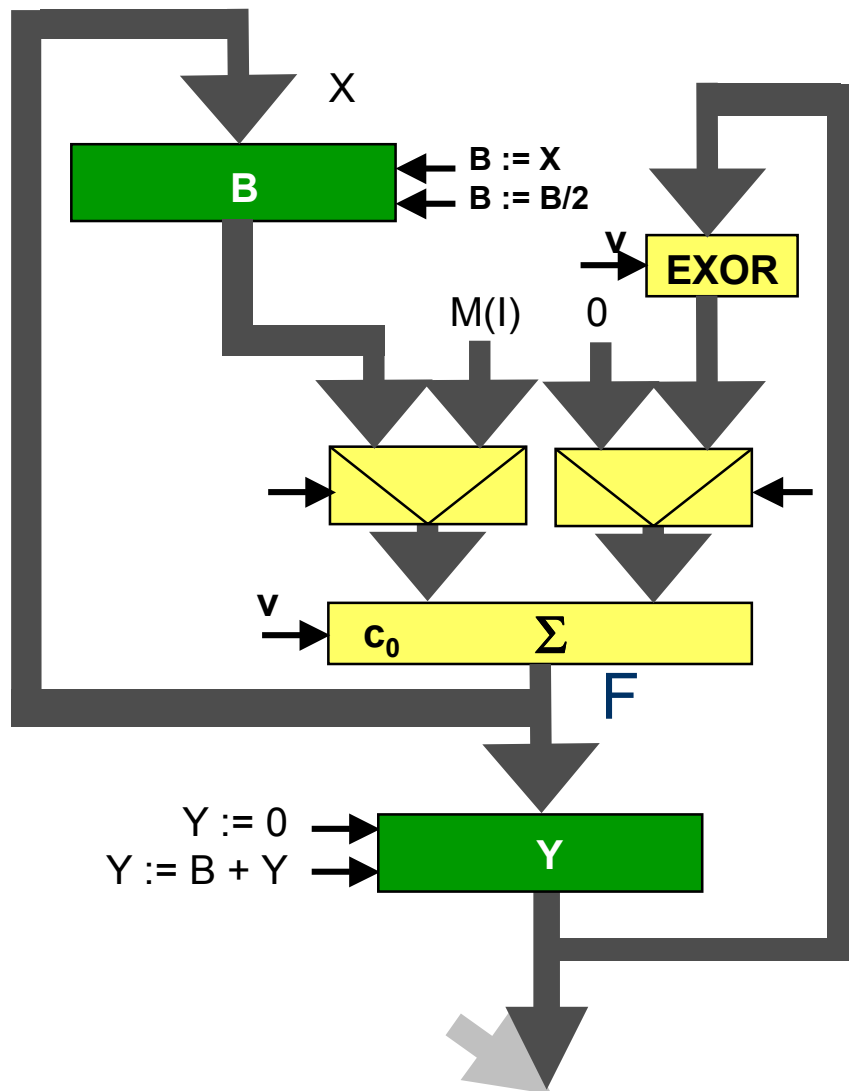
$Y := B + Y$

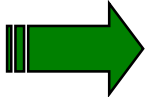
$B := B - Y$

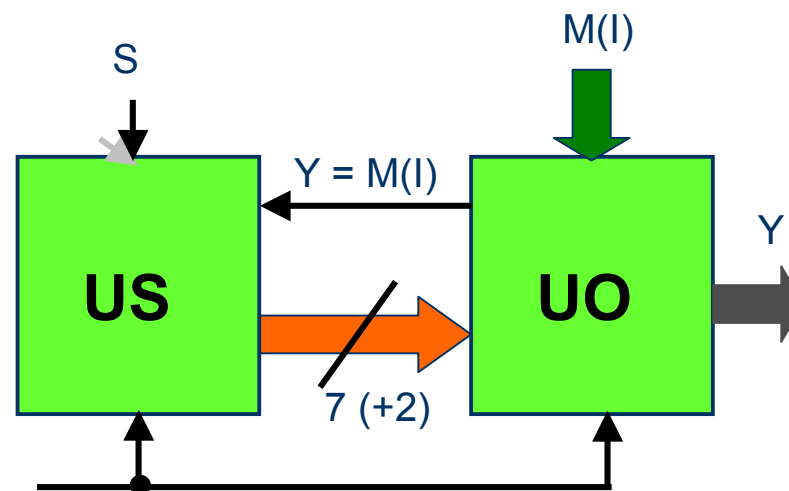
$$V \oplus Y = \begin{cases} Y, & \text{gdy } V = 0 \\ \bar{Y}, & \text{gdy } V = 1 \end{cases}$$



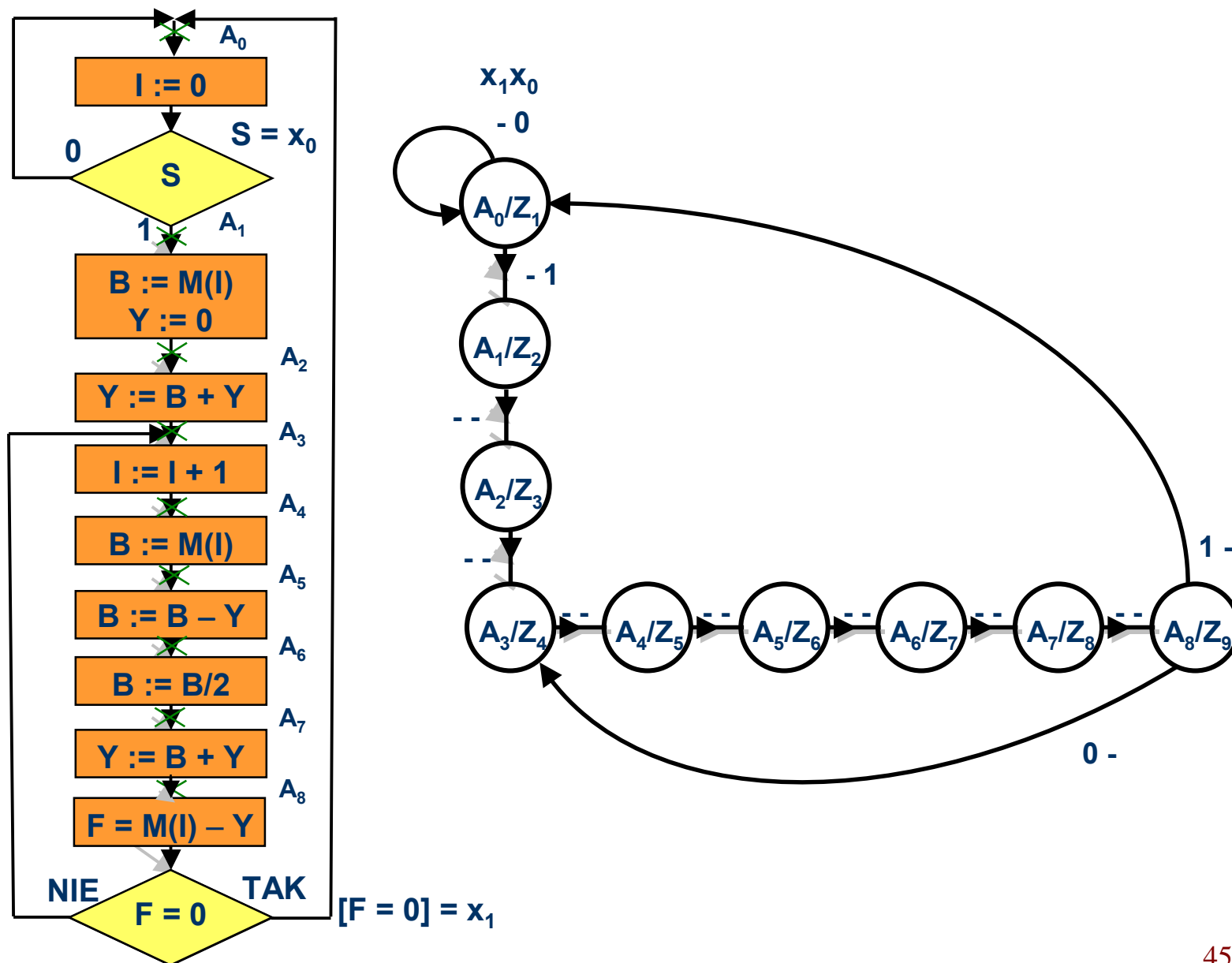
Przykład c.d.



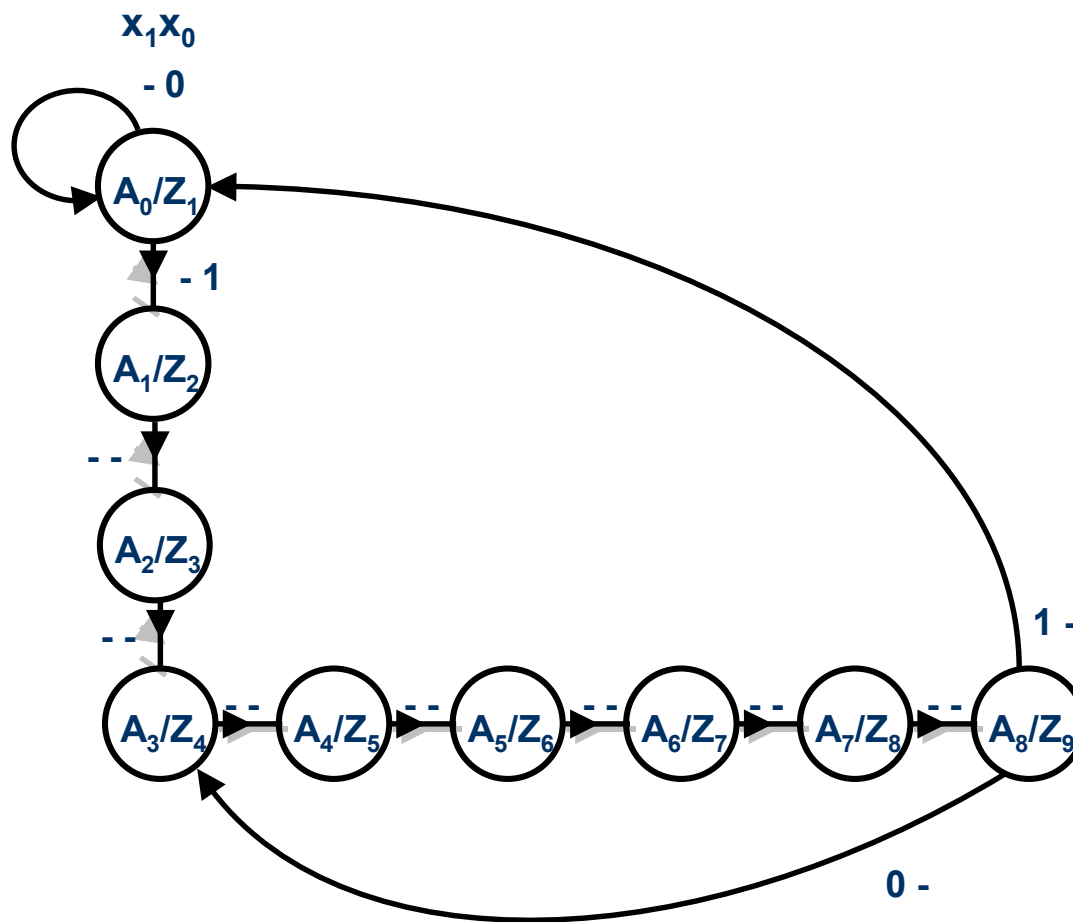
$B + Y$
 $B - Y$
 Warunek $Y = M(I)$  Warunek $F = M(I) - Y = 0$



Przykład c.d.



Przykład c.d.



x_1x_0 A	00	01	11	10	Z
A_0	A_0	A_1	A_1	A_0	Z_1
A_1	A_2	A_2	A_2	A_2	Z_2
A_2	A_3	A_3	A_3	A_3	Z_3
A_3	A_4	A_4	A_4	A_4	Z_4
A_4	A_5	A_5	A_5	A_5	Z_5
A_5	A_6	A_6	A_6	A_6	Z_6
A_6	A_7	A_7	A_7	A_7	Z_7
A_7	A_8	A_8	A_8	A_8	Z_8
A_8	A_3	A_3	A_0	A_0	Z_9