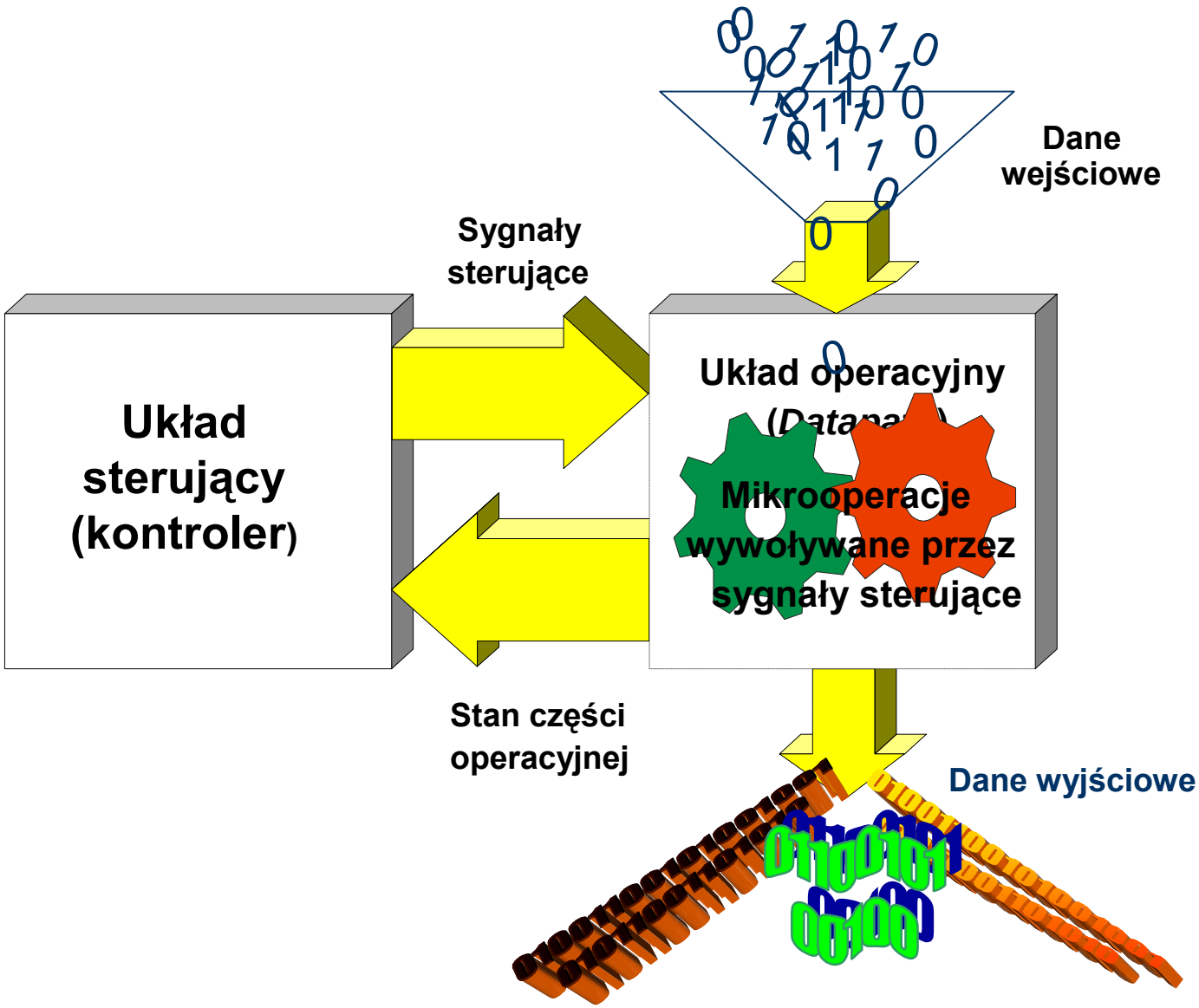
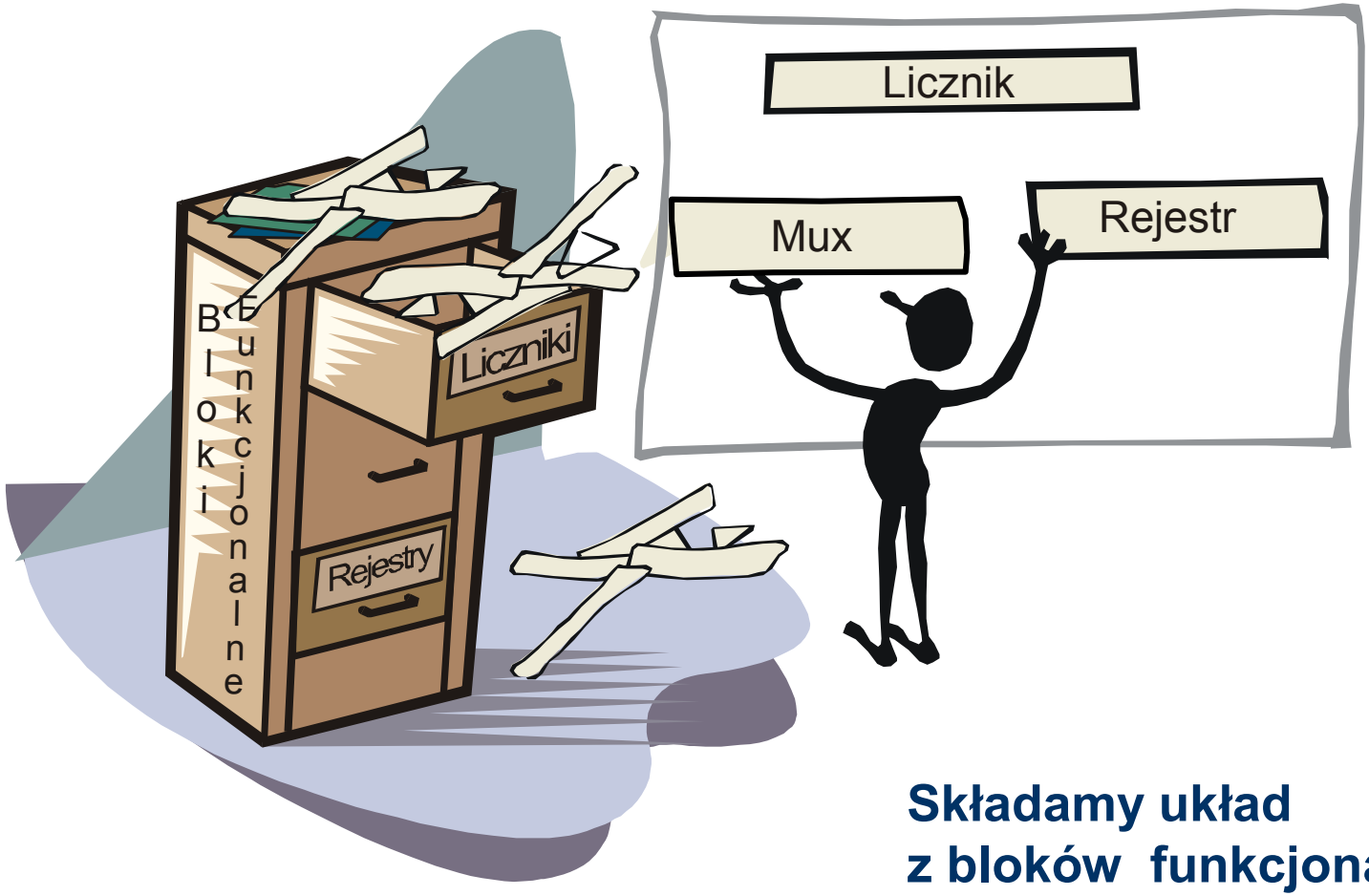
Układ cyfrowy

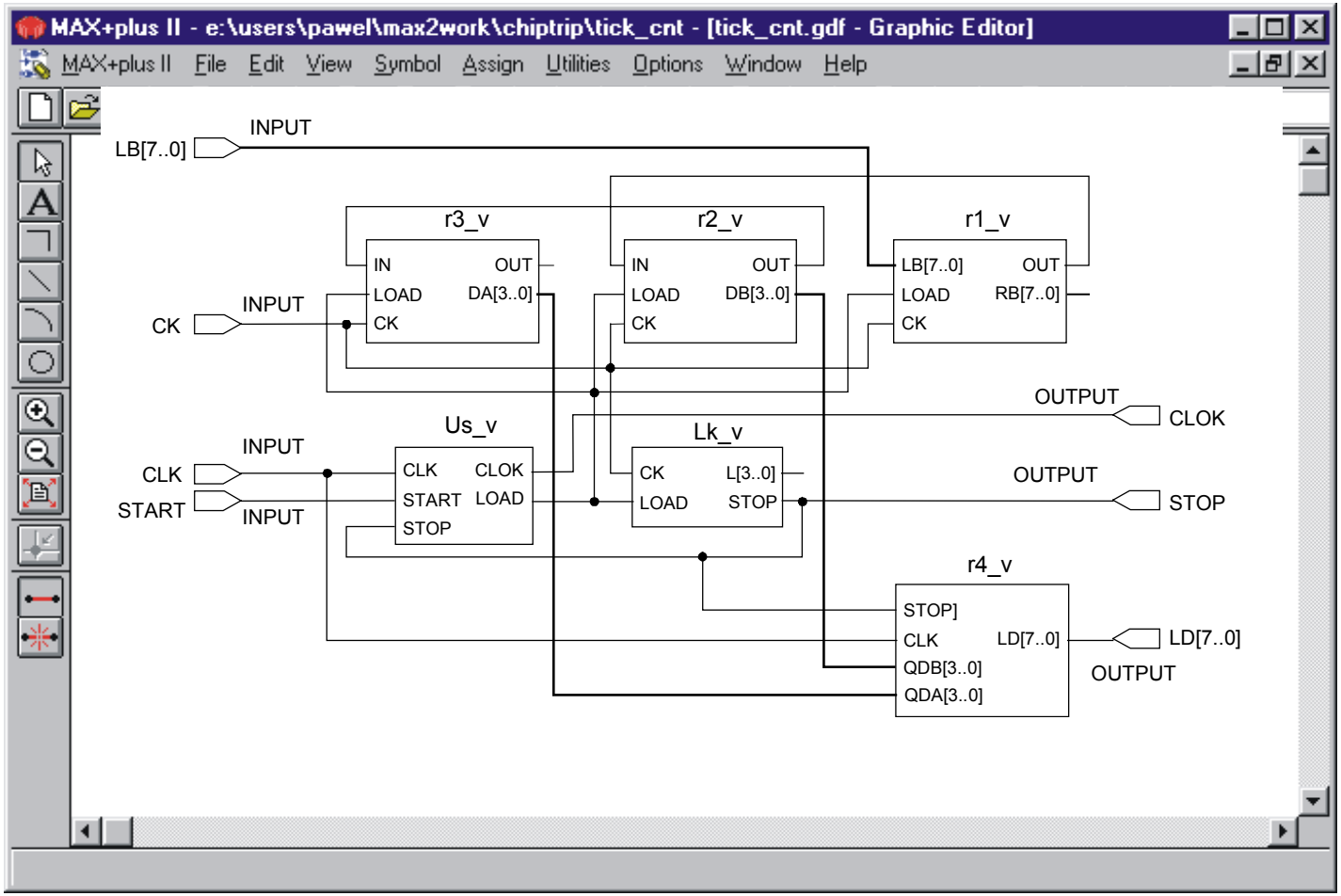


Synteza strukturalna układów cyfrowych

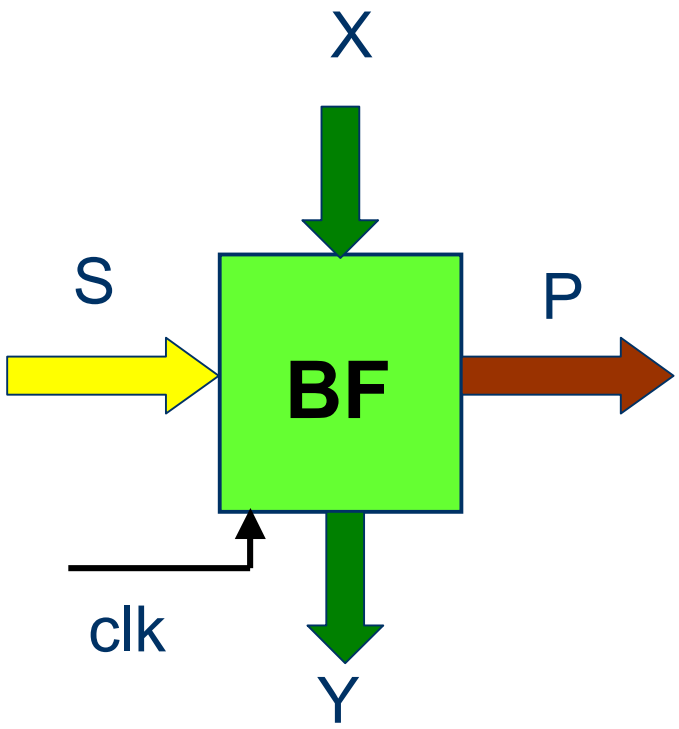


Składamy układ
z bloków funkcjonalnych

Edytor graficzny



Bloki funkcjonalne



***X*, (*Y*)** – wejścia (wyjścia) sygnałów reprezentujących dane wejściowe i wyjściowe

S – wejścia sterujące,

P – wyjścia predykatowe,

clk – wejście zegarowe

Bloki funkcjonalne stanowią wyposażenie bibliotek komputerowych systemów projektowania

I T P W ZPT

System MAX+PLUSII…

…jest wyposażony w dwie biblioteki komponentów:

a) bibliotekę tzw. makrofunkcji

b) bibliotekę megafunkcji (moduły LPM)

Library of Parameterized Modules (LPM)

I T P W

ZPT

Makrofunkcje

Macrofunctions:

- Adders
- Latches
- Arithmetic Logic Units
- Buffers
- Multiplexers
- Comparators
- Converters
- Counters
- Registers
- Shift Registers
- Multipliers

} Były kiedyś produkowane jako bloki funkcjonalne serii 74xx

I T P W ZPT

Macrofunctions

Multiplexers

21mux	2-Line-to-1-Line Multiplexer
161mux	16-Line-to-1-Line Multiplexer
2X8mux	2-Line-to-1-Line Multiplexer for 8-Bit Buses
74151b	8-Line-to-1-Line Multiplexer
74153	Dual 4-Line-to-1-Line Multiplexer
74157	Quad 2-Line-to-1-Line Multiplexer
74258	Quad 2-Line-to-1-Line Multiplexers with Inverting Tri-State Outputs
74352	Dual 4-Line-to-1-Line Data Selector/Multiplexer with Inverting Outputs
74354	8-Line-to-1-Line Data Selector/Multiplexer/Register with Tri-State Outputs

I T P W ZPT

Macrofunctions

Registers

7491	Serial-In Serial-Out Shift Register
7494	4-Bit Shift Register with Asynchronous Preset and Clear
7495	4-Bit Parallel-Access Shift Register
7496	5-Bit Shift Register
7499	4-Bit Shift Register with /JK Serial Inputs and Parallel Outputs
74164	Serial-In Parallel-Out Shift Register
74194	4-Bit Bidirectional Shift Register with Parallel Load
74295	4-Bit Right-Shift Left-Shift Register with Tri-State Outputs
74299	8-Bit Universal Shift/Storage Register
74674	16-Bit Shift Register

I T P W

ZPT

Macrofunctions

Counters

7490	Decade or Binary Counter with Clear and Set-to-9
7492	Divide-by-12 Counter
7493	4-Bit Binary Counter
74160	4-Bit Decade Counter with Synchronous Load and Asynchronous Clear
74161	4-Bit Binary Up Counter with Synchronous Load and Asynchronous Clear
74162	4-Bit Decade Up Counter with Synchronous Load and Synchronous Clear
74163	4-Bit Binary Up Counter with Synchronous Load and Synchronous Clear
74190	4-Bit Decade Up/Down Counter with Asynchronous Load
74192	4-Bit Decade Up/Down Counter with Asynchronous Clear
74193	4-Bit Binary Up/Down Counter with Asynchronous Clear
74294	Programmable Frequency Divider/Digital Timer

I T P W

ZPT

Konsekwencje wprowadzenia makrofunkcji

Struktury makrofunkcji nie są odpowiednie do technologii układów programowalnych a ich odwzorowanie technologiczne na komórki aktualnie produkowanych układów FPGA nie prowadzi do optymalnego wykorzystania zasobów sprzętowych

I T P W

ZPT

Megafunkcje

System MAX + plus II jest wyposażony w moduły LPM

(Library of Parameterized Modules)

Moduły LPM są parametryzowane: użytkownik może ustalić np. wielkość MUX, liczbę bitów argumentów sumatora lub niektóre mikrooperacje.

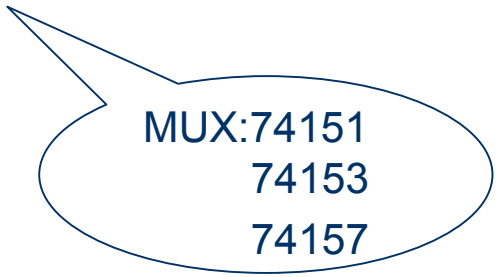
I T P W

ZPT

Megafunctions/LPM

Gates

lpm_and
lpm_inv
lpm_bustri
lpm_clshift
lpm_constant
lpm_decode
lpm_mux
lpm_or
lpm_xor



Arithmetic Components

lpm_compare
lpm_counter
lpm_add_sub
lpm_mult

Storage Components

lpm_latch
lpm_shiftreg
lpm_ram_dp
lpm_ram_io
pm_ff
lpm_rom
lpm_fifo

I T P W

ZPT

Konfiguracja modułu LPM

Konfiguracja i wyposażenie jest definiowana parametrami

Moduł sumator/układ odejmujący: *lpm_add_sub*

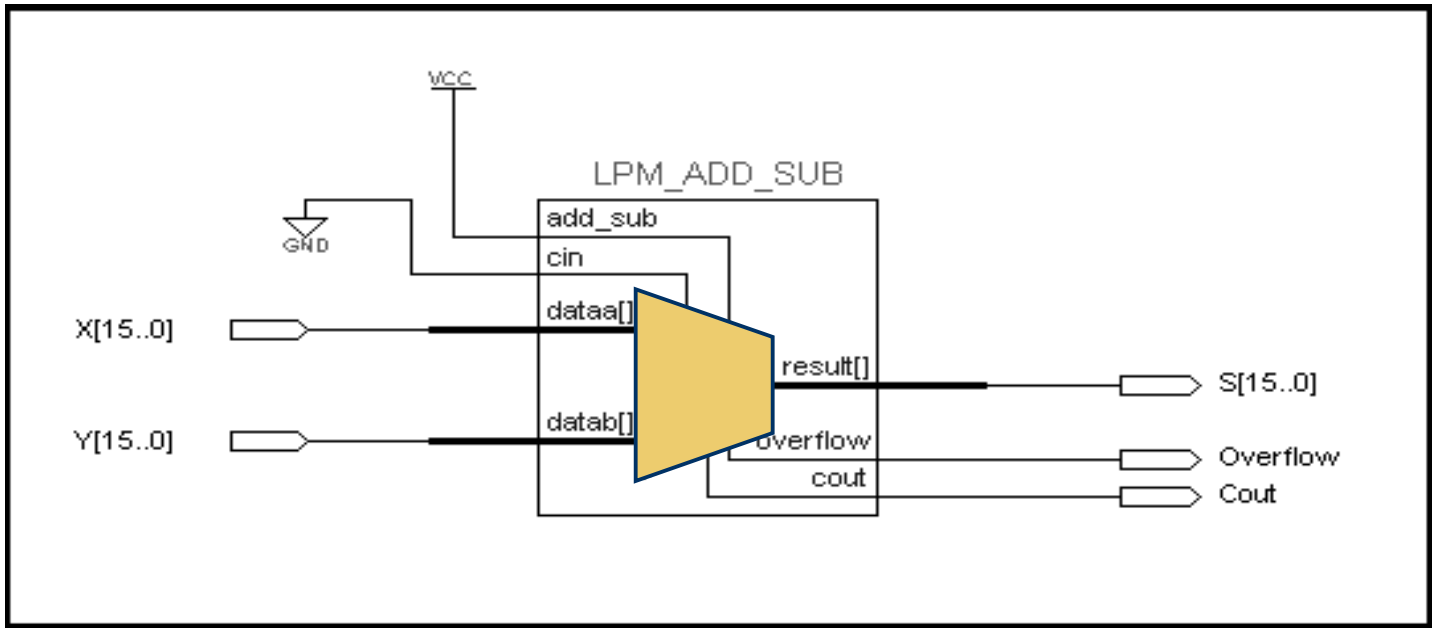
a) LPM_WIDTH
b) LPM_CONFIGURATION

a) określa liczbę bitów sumatora
b) określa operację: liczby ze znakiem (signed) lub
liczby bez znaku (unsigned)

I T P W

ZPT

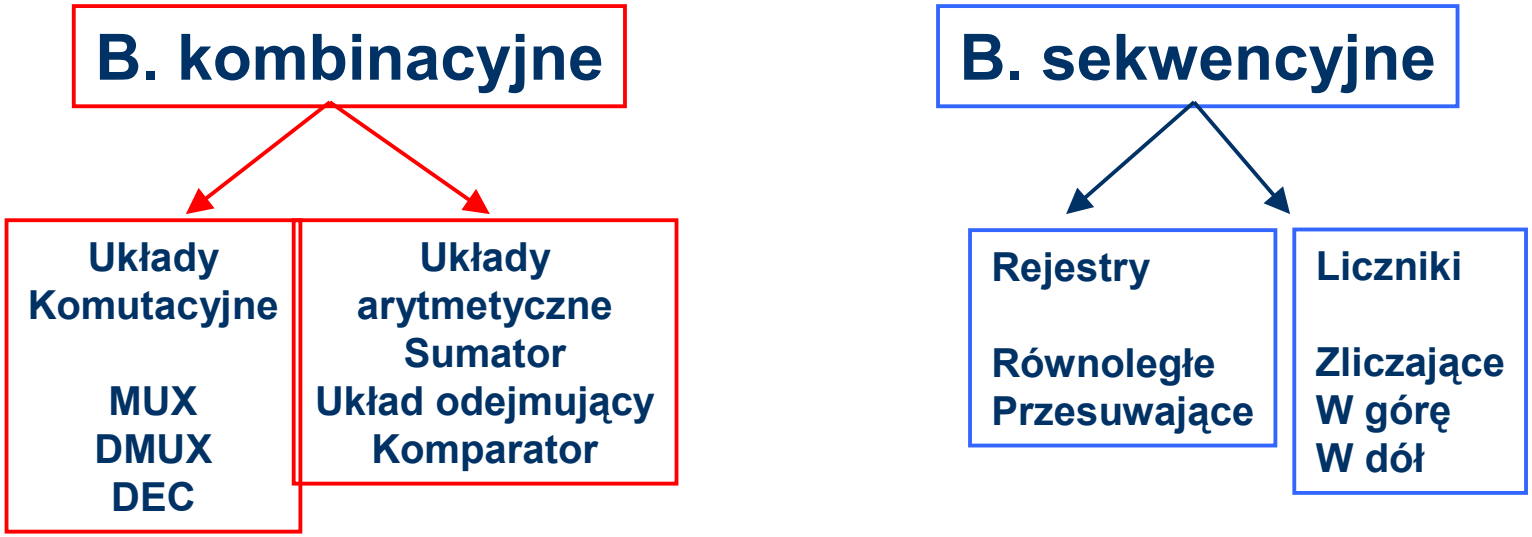
Sumator w strukturze LPM



I T P W

ZPT

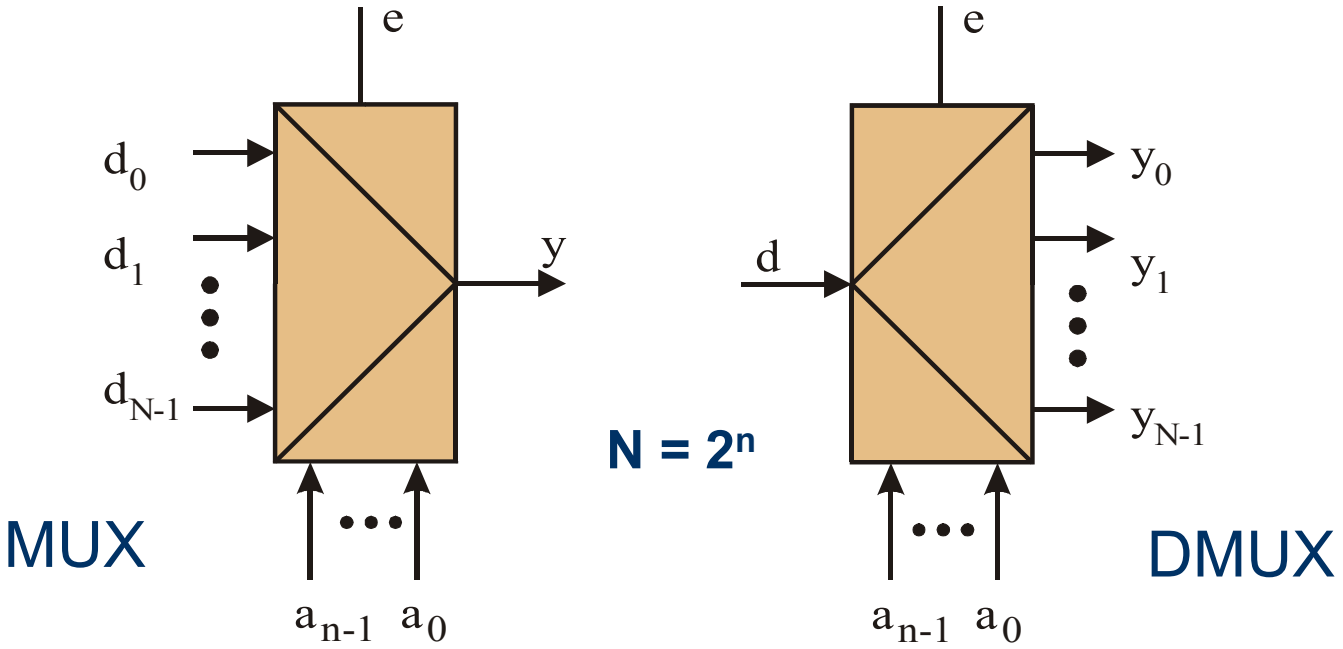
Najważniejsze bloki funkcjonalne



I T P W

ZPT

Multipleksery, demultipleksery



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

k = L(A), P_k – pełny iloczyn

I T P W ZPT

Multipleksery

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

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

Dla n = 1 (MUX 2 : 1):

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

dla n = 2 (MUX 4 : 1):

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

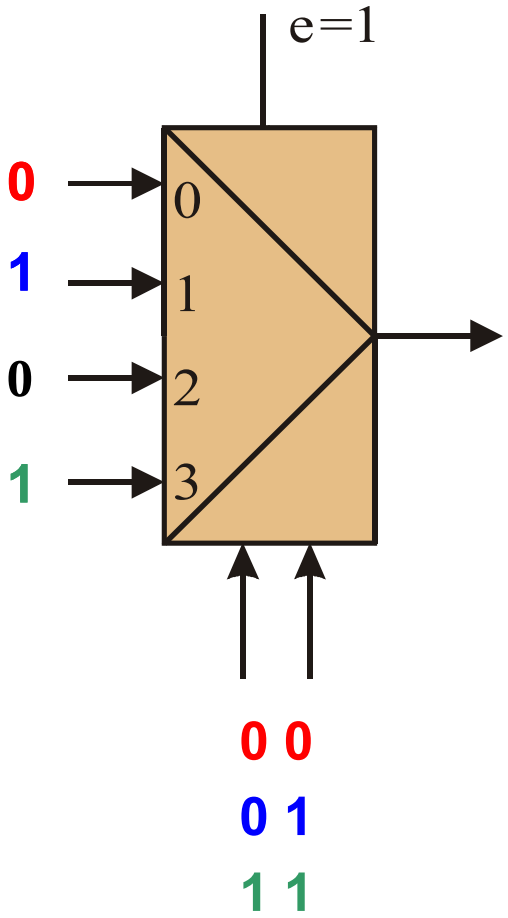
dla n = 3 (MUX 8 : 1):

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

I T P W

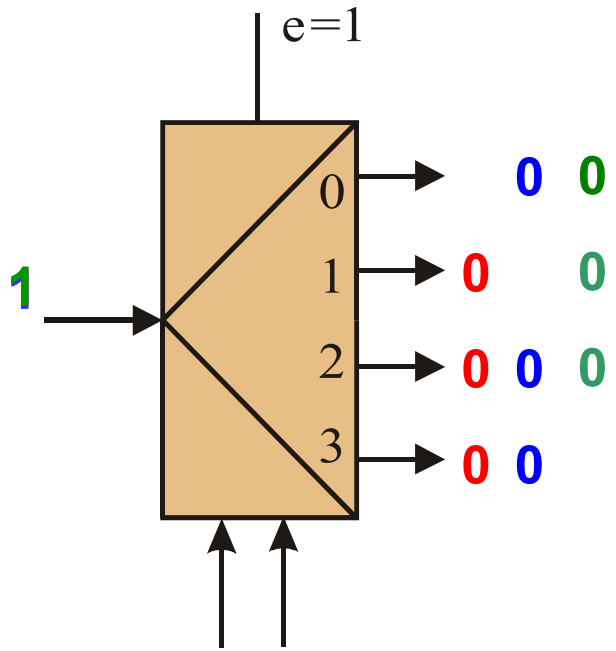
ZPT

Multiplekser



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

Demultiplekser



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

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

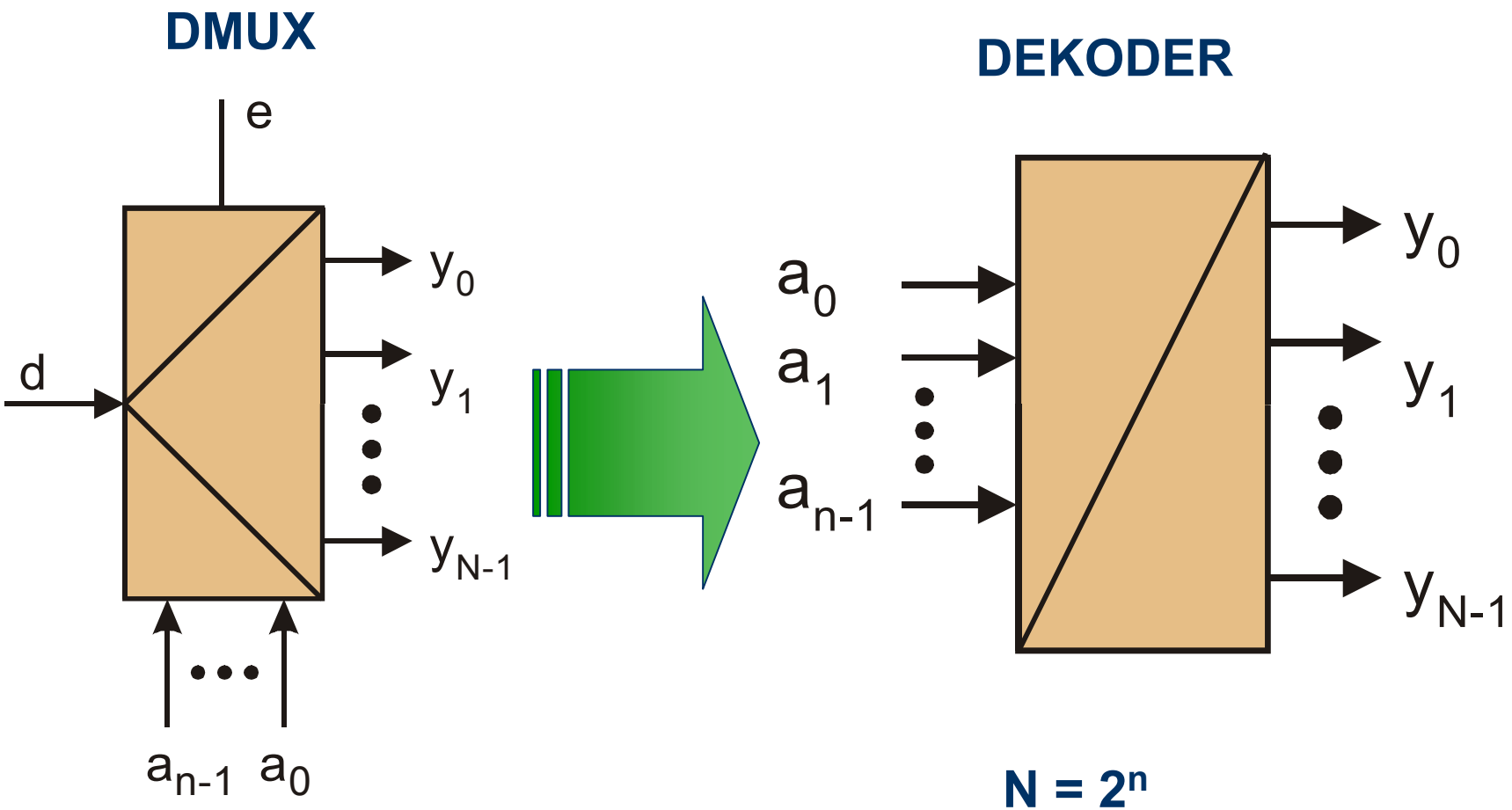
$$y_2 = a_1 \overline{a}_0 d$$

$$y_3 = a_1 a_0 d$$

I T P W

ZPT

Dekoder



$N = 2^n$

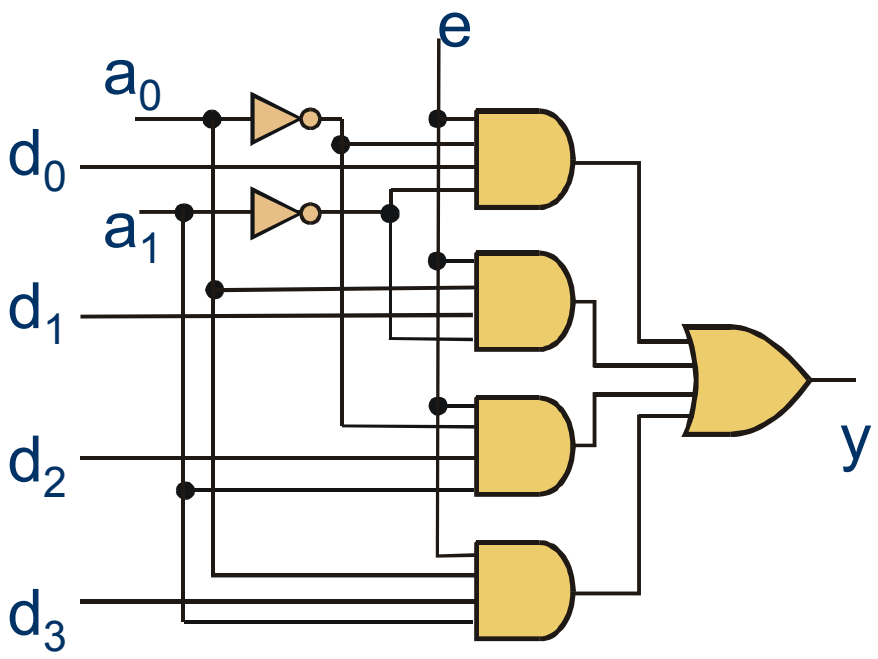
I T P W

ZPT

Multipleksery, demultipleksery

Multiplekser

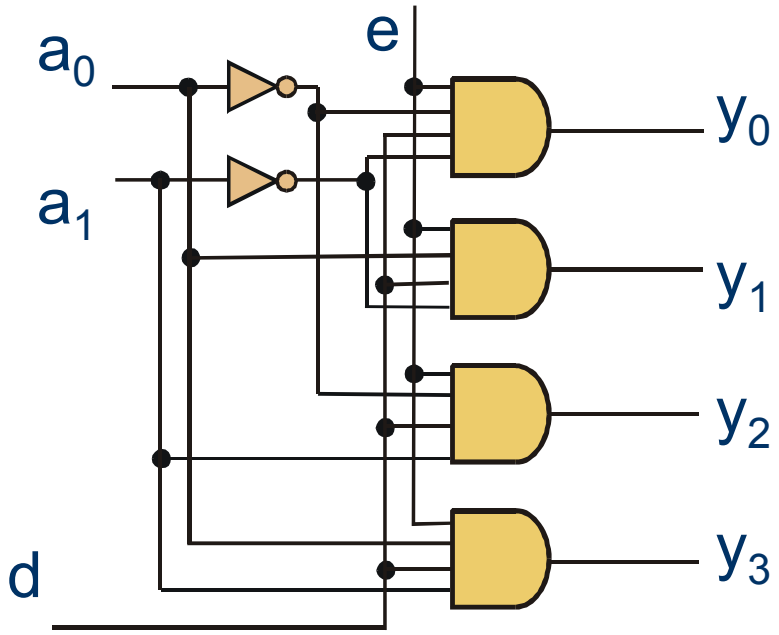
$$y = \bar{a}_1\bar{a}_0d_0 + \bar{a}_1a_0d_1 + a_1\bar{a}_0d_2 + a_1a_0d_3$$



Demultiplekser

$$y_0 = \bar{a}_1\bar{a}_0d \quad y_1 = \bar{a}_1a_0d$$

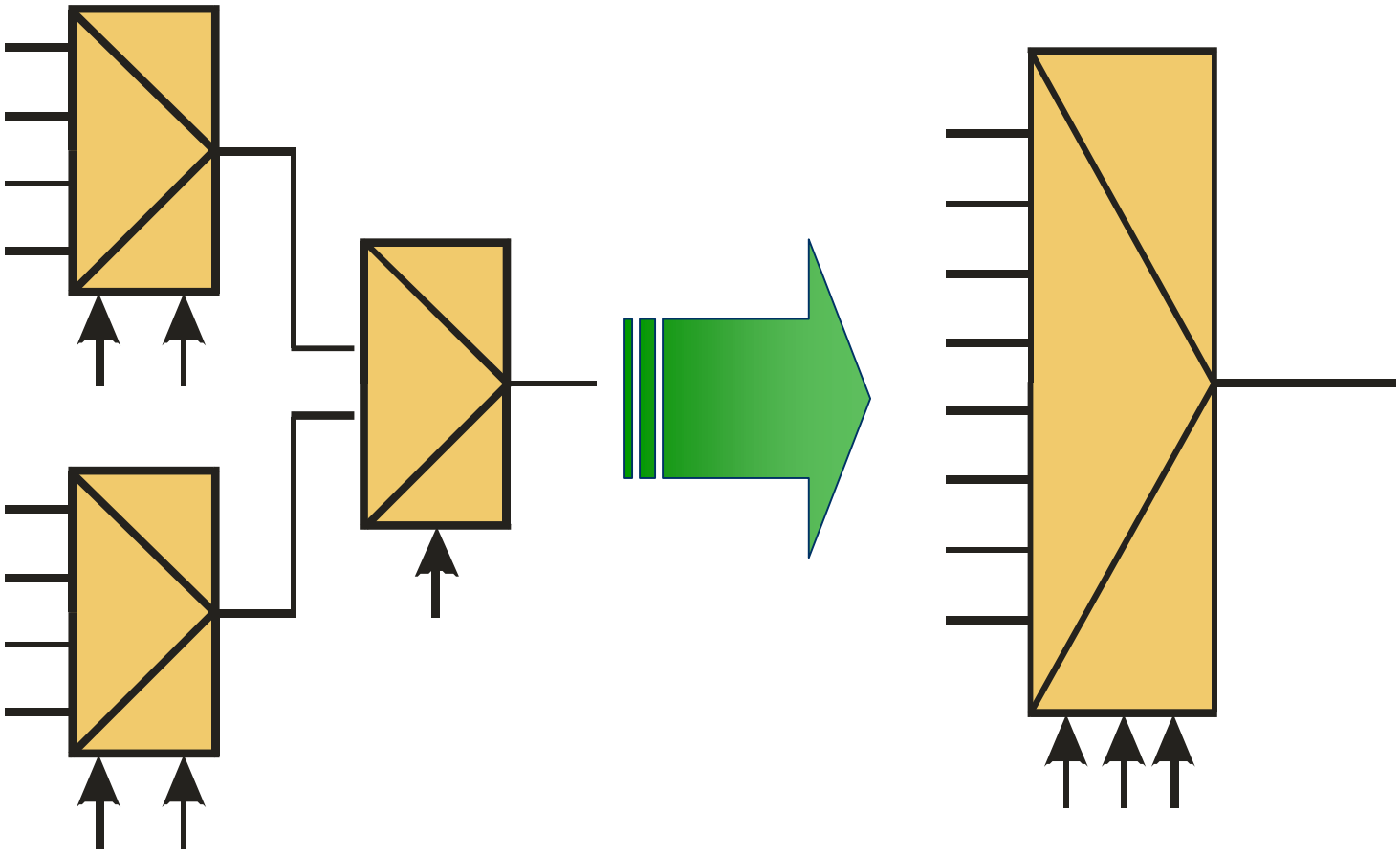
$$y_2 = a_1\bar{a}_0d \quad y_3 = a_1a_0d$$



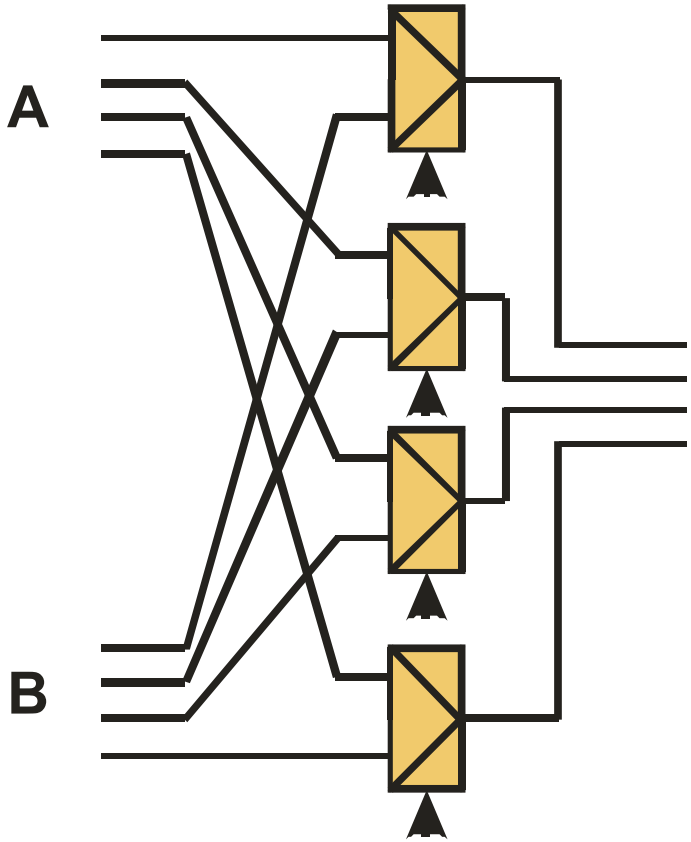
I T P W

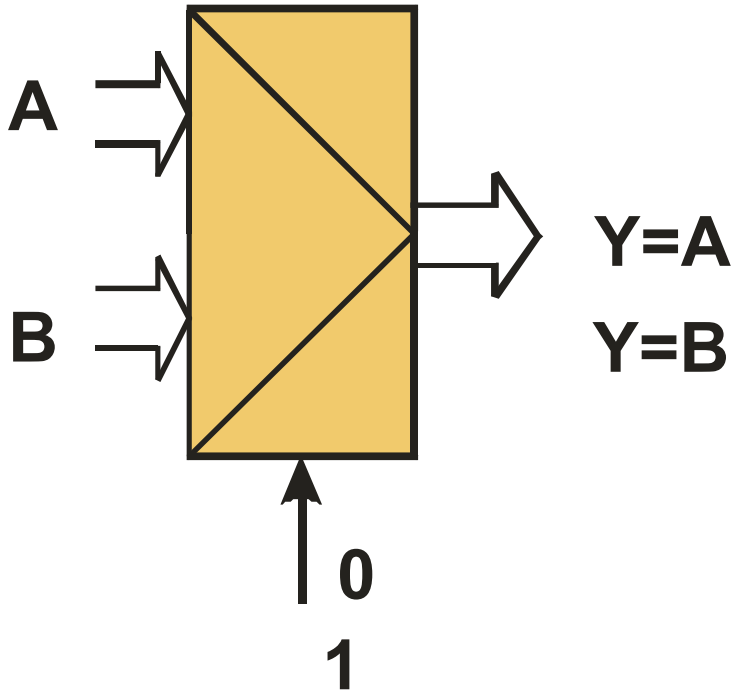
ZPT

Multipleksery kaskadowe



Multipleksery grupowe

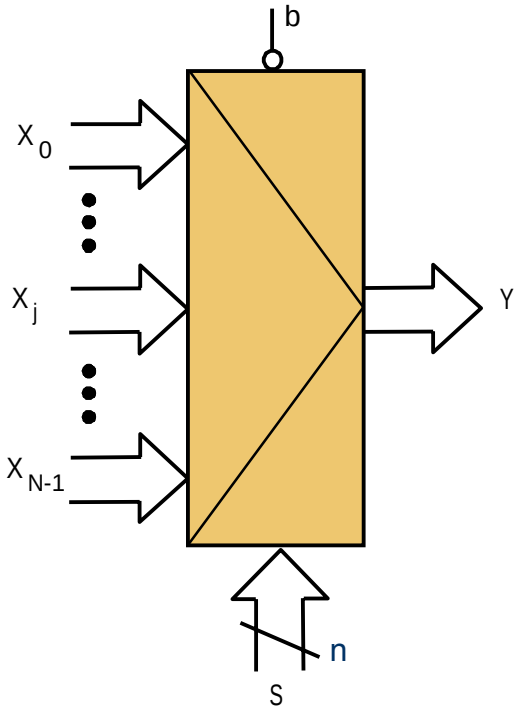




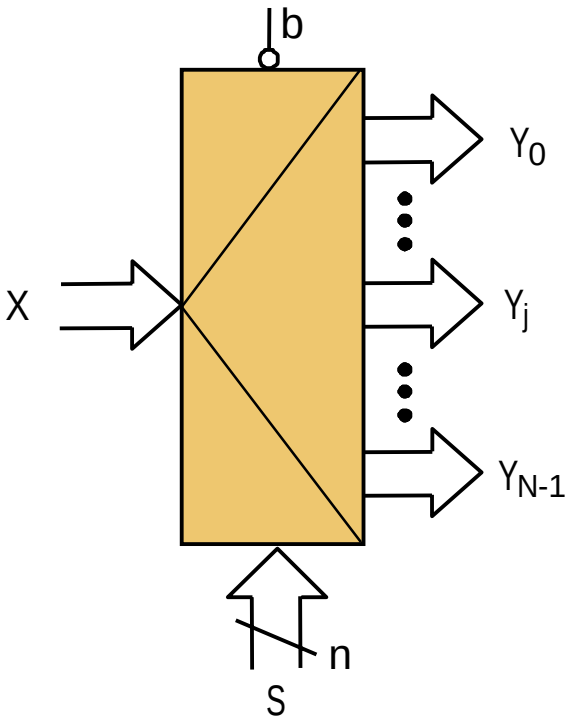
I T P W

ZPT

Bloki komutacyjne



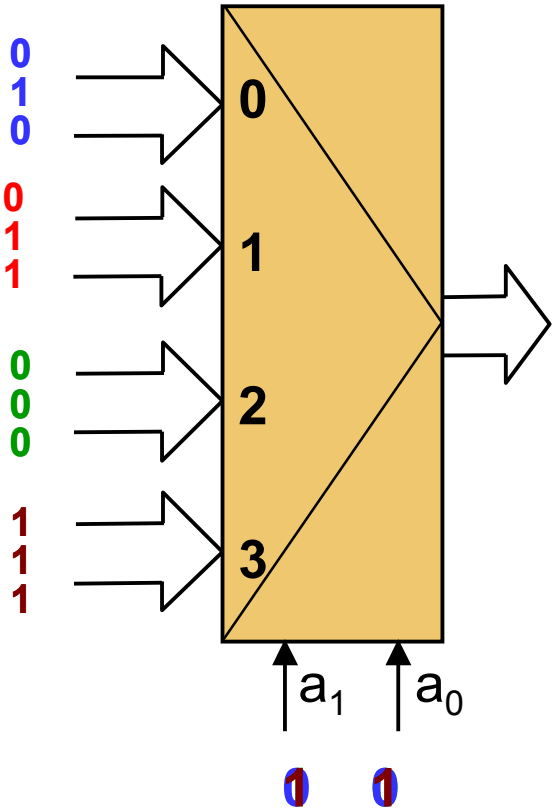
Multiplekser służy do wybierania jednego z wielu słów wejściowych i przesyłania go na wyjście. Na wyjściu Y pojawia się słowo wejściowe wskazane adresem A (wg naturalnego kodu binarnego).

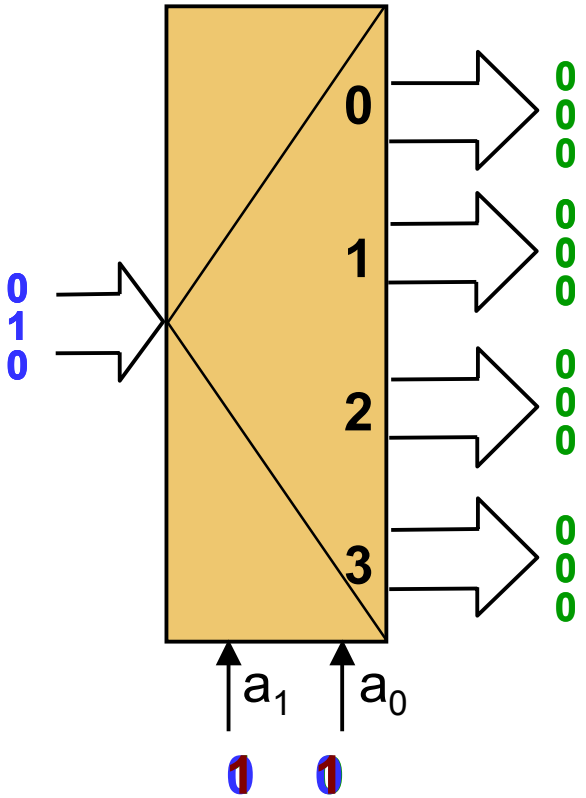


Demultiplekser służy do przesyłania słowa X wejściowego na jedno z wielu wyjść; numer tego wyjścia jest równy aktualnej wartości adresu.

I T P W ZPT

Bloki komutacyjne

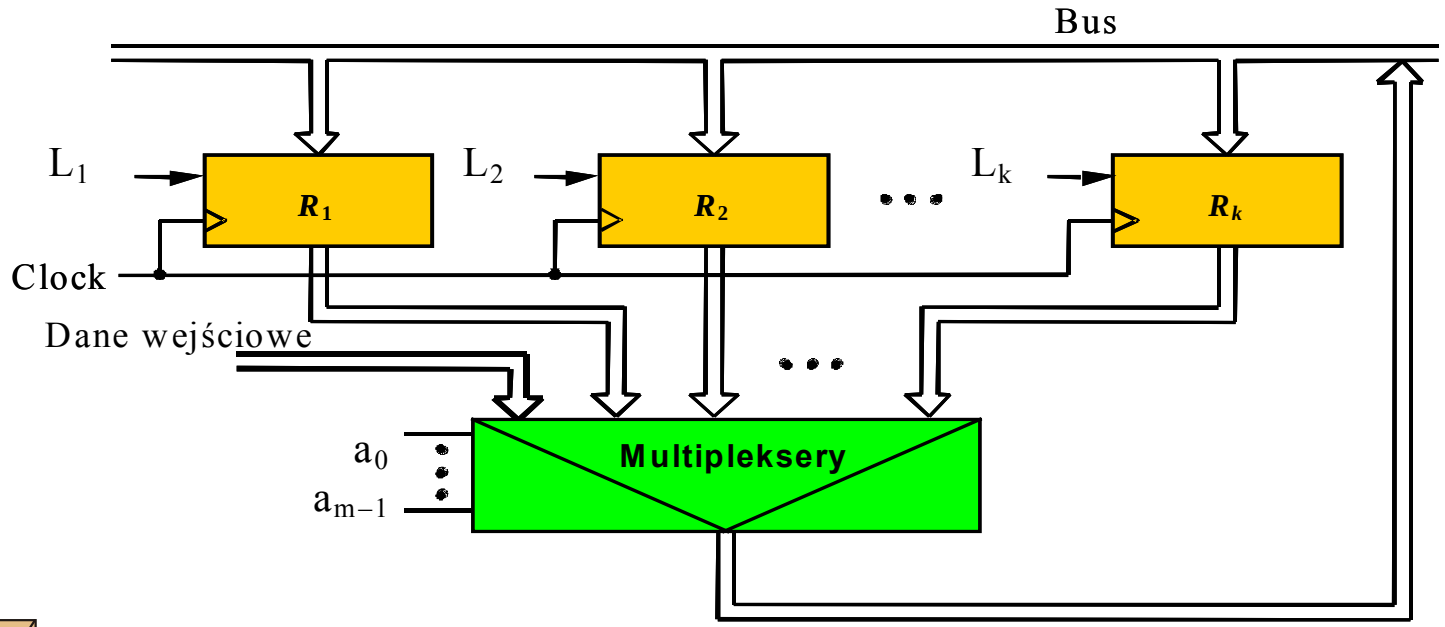


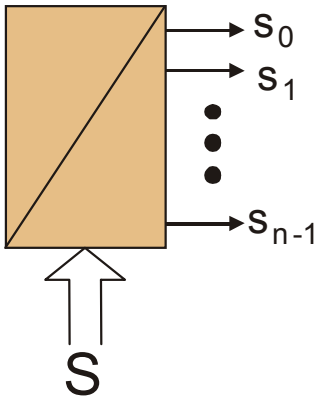


I
T
P
W

ZPT

Magistrala (realizacja z multiplekserami)



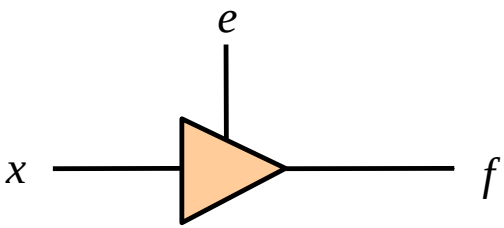


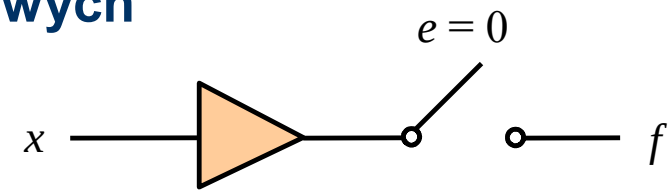
I T P W

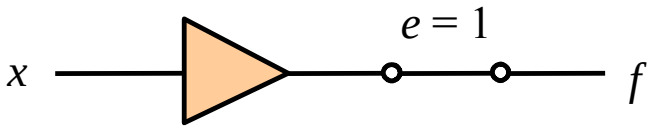
ZPT

Magistrale (szyny)

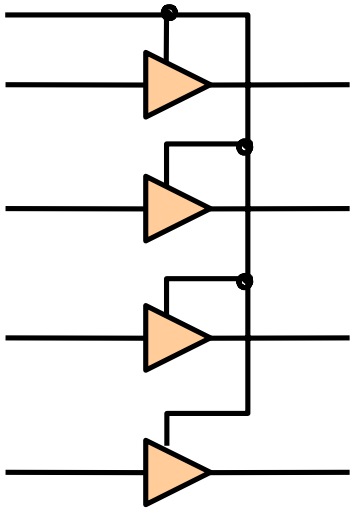
Budowane z elementów trójstanowych



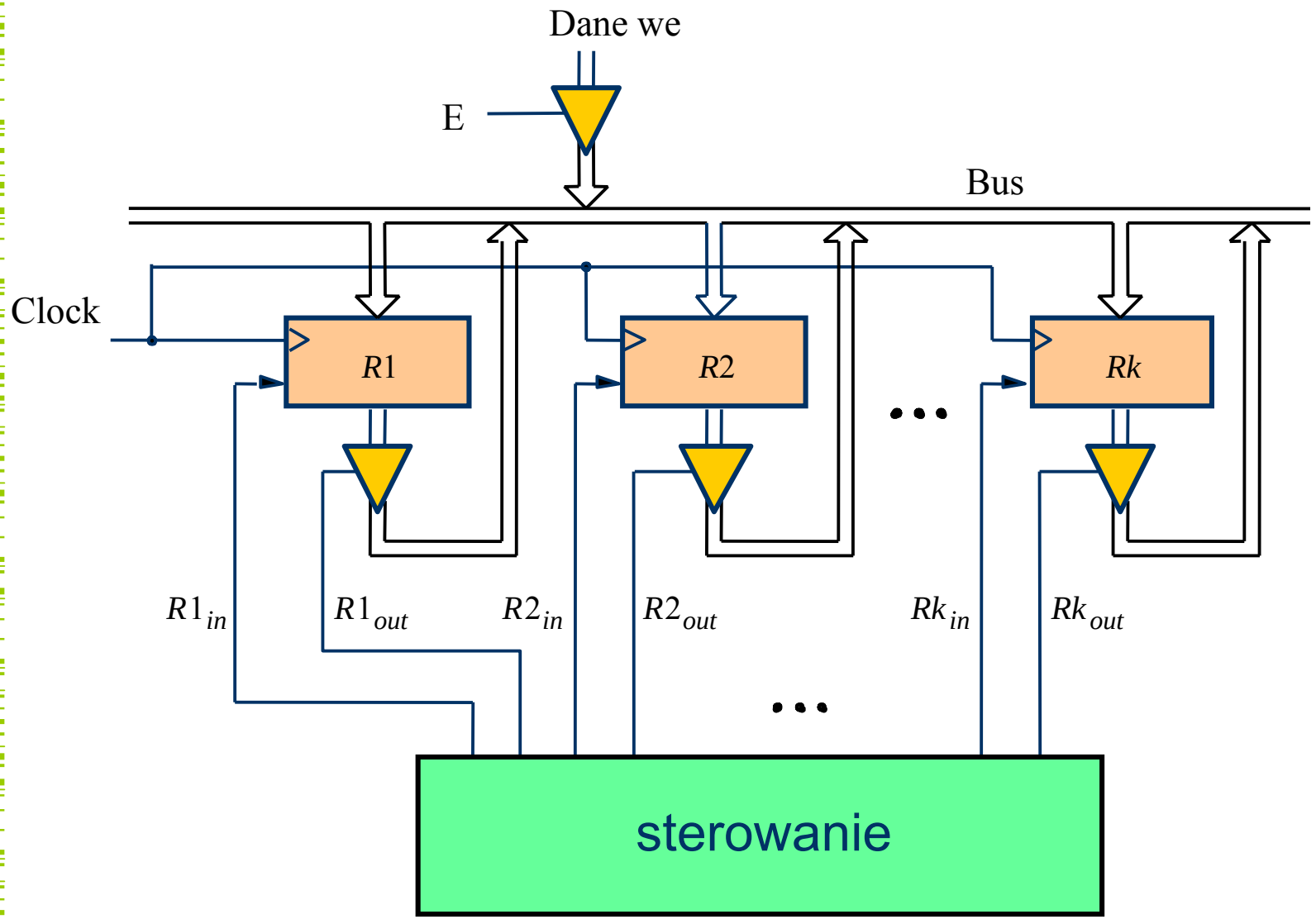




e	x	f
0	0	Z
0	1	Z
1	0	0
1	1	1

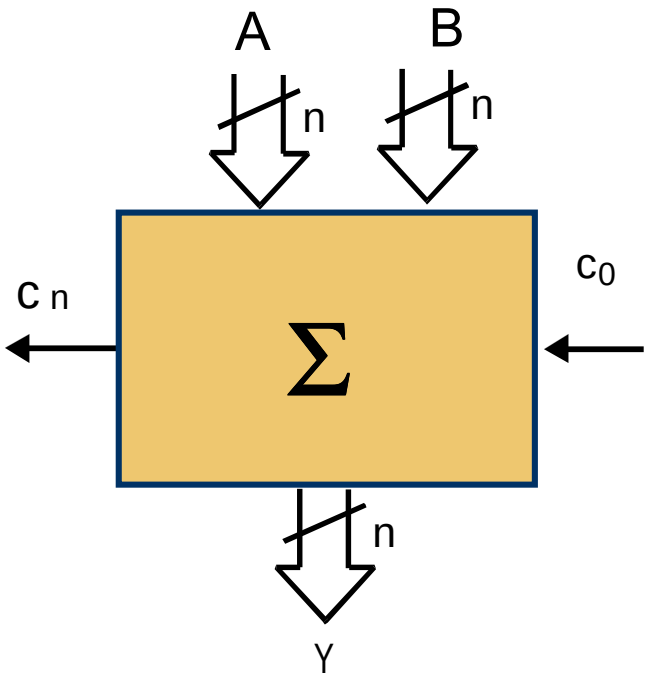


Magistrala (realizacja z buforami)



Sumatory

Sumator – podstawowy BF powszechnie stosowany w technice DSP



Inne układy arytmetyczne:
układy odejmowania
układy mnożące
układy dzielenia

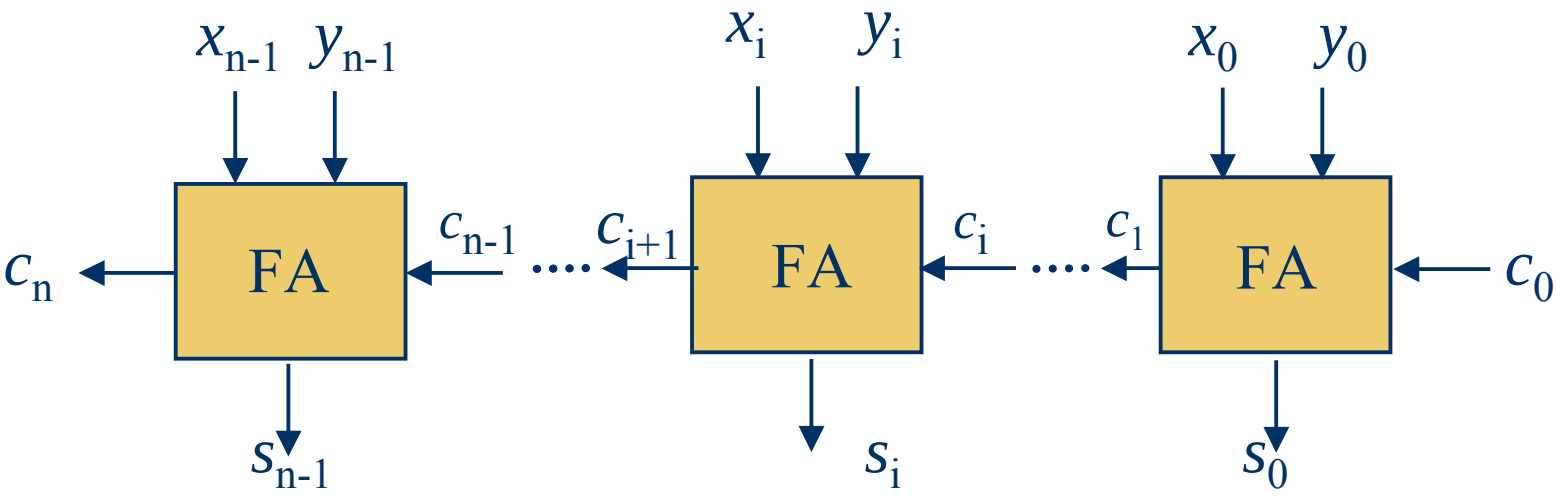
...są budowane z sumatorów

I T P W

ZPT

Sumator kaskadowy

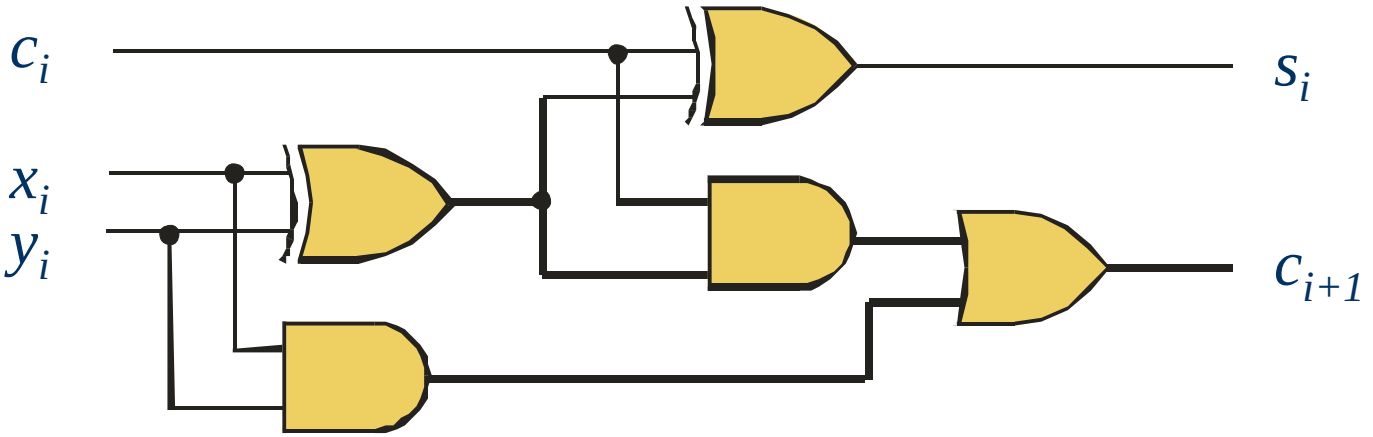
Ripple carry adder



$$s_i = x_i \oplus y_i \oplus c_i$$

$$c_{i+1} = x_i y_i \vee c_i (x_i \oplus y_i)$$

Sumator (Full adder)



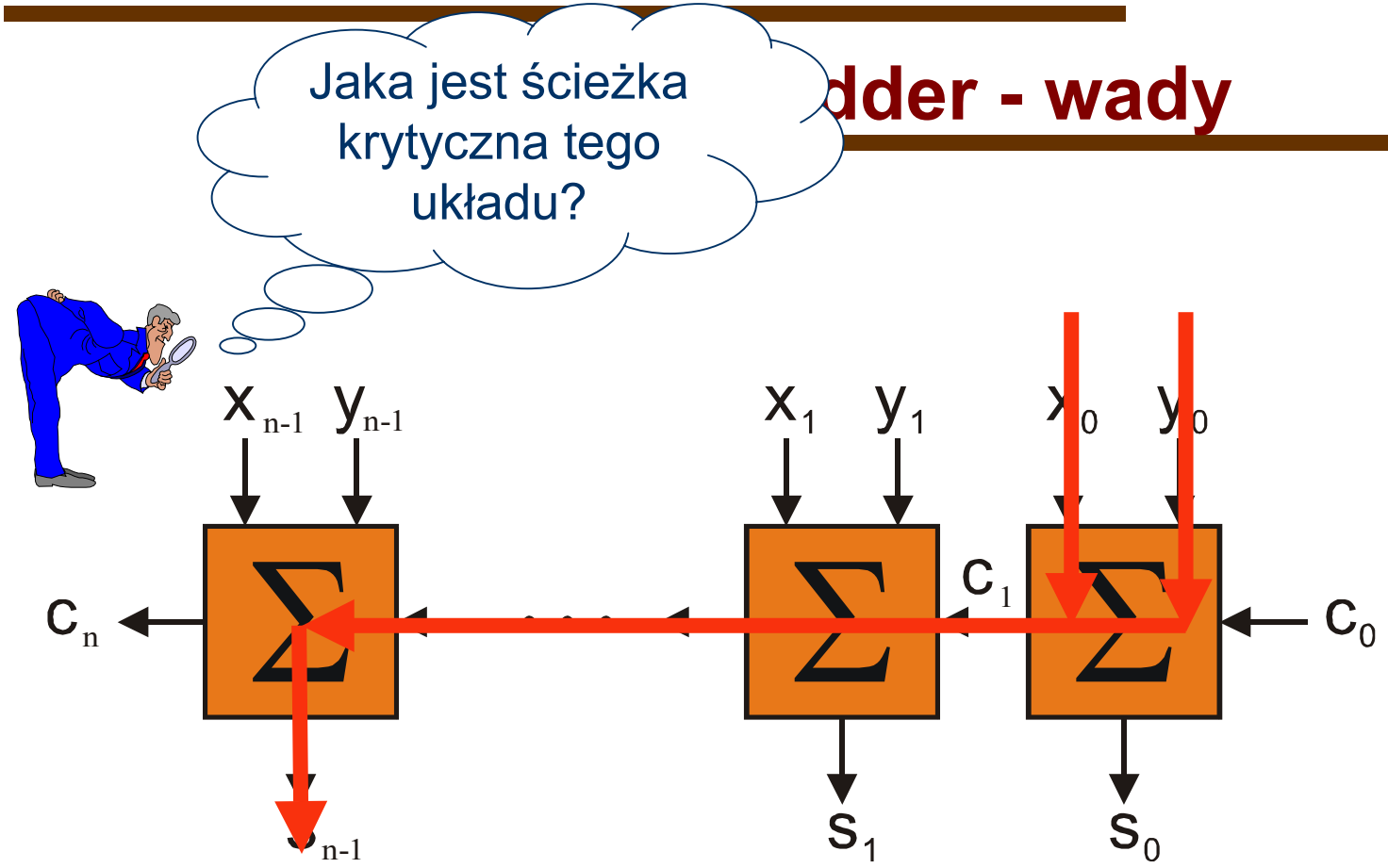
$$s_i = x_i \oplus y_i \oplus c_i$$

$$c_{i+1} = x_i y_i \vee c_i (x_i \oplus y_i)$$

I T P W

ZPT

dder - wady

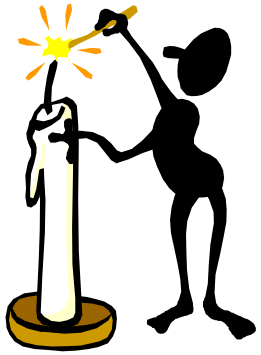


Bardzo długa - liniowo zależna od wielkości sumatora

Dla większości zastosowań sumator kaskadowy jest zbyt wolny

I T P W

ZPT



Znacznie lepszy jest sumator z antycypacją przeniesień,

w którym wszystkie przeniesienia są wytwarzane jednocześnie na podstawie bitów sumowanych składników.

I T P W

ZPT

Sumator z antycypacją przeniesień

$$c_{i+1} = x_i y_i \vee c_i(x_i \vee y_i)$$
$$g_i = x_i y_i$$
$$p_i = x_i \vee y_i$$

Wtedy:

$$c_{i+1} = g_i \vee p_i c_i$$

$$s_i = c_i \oplus (p_i \bar{g}_i)$$
$$s_i = x_i \oplus y_i \oplus c_i$$

I T P W

ZPT

Sumator z antycypacją przeniesień

$c_{i+1} = g_i \vee p_i c_i$

c_0

$c_1 = g_0 \vee p_0 c_0$

$c_1 = g_0 \vee p_0 c_0$

$c_2 = g_1 \vee p_1 c_1 = g_1 \vee p_1 (g_0 \vee p_0 c_0)$

$c_2 = g_1 \vee p_1 g_0 \vee p_1 p_0 c_0$ (funkcja 5 arg.)

$c_3 = g_2 \vee p_2 g_1 \vee p_2 p_1 g_0 \vee p_2 p_1 p_0 c_0$

$c_4 = g_3 \vee p_3 g_2 \vee p_3 p_2 g_1 \vee p_3 p_2 p_1 g_0 \vee p_3 p_2 p_1 p_0 c_0$

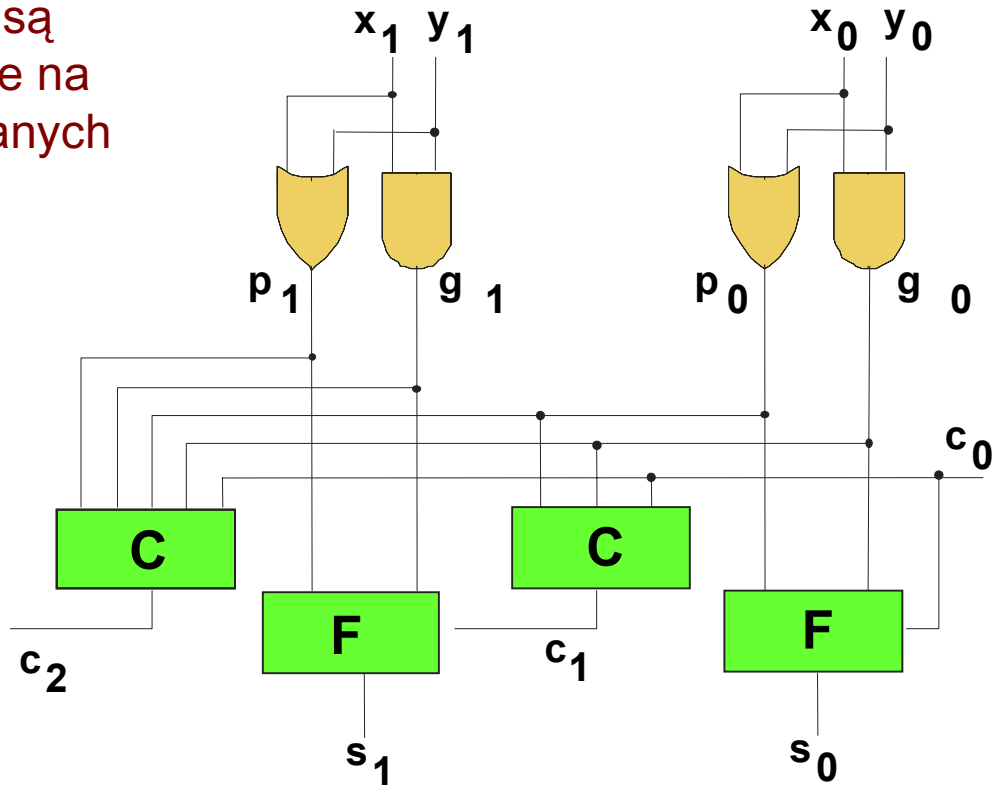
I T P W

ZPT

Sumator z antycypacją przeniesień

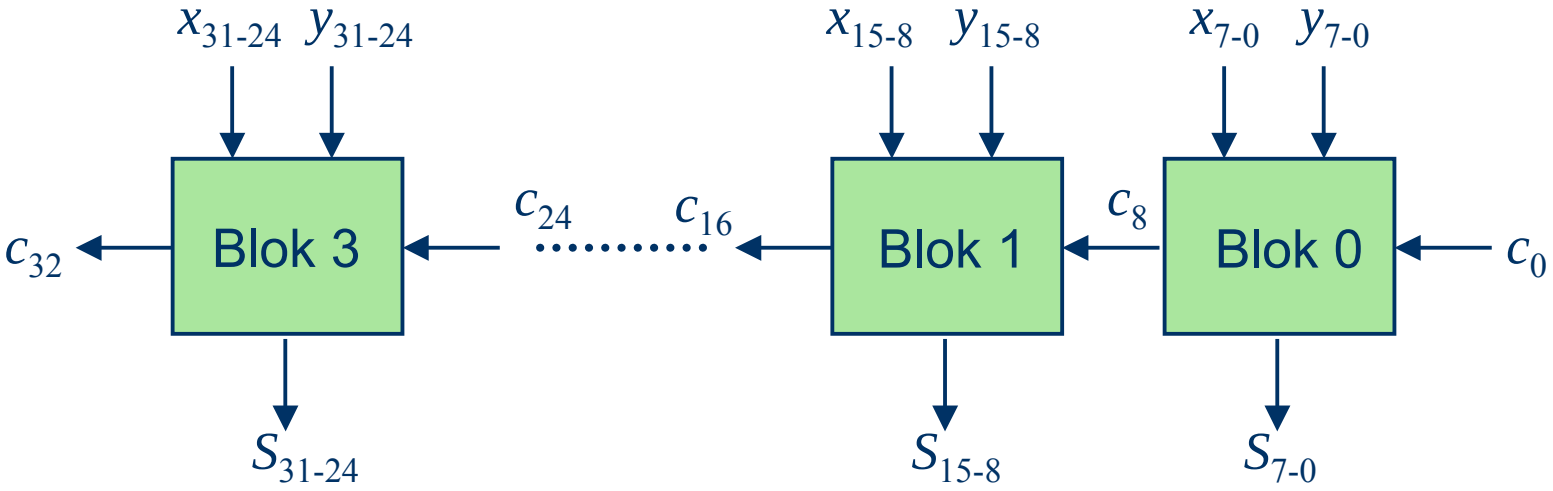
$$c_2 = g_1 \vee p_1 g_0 \vee p_1 p_0 c_0$$

Wszystkie przeniesienia są wytwarzane jednocześnie na podstawie bitów sumowanych składników!

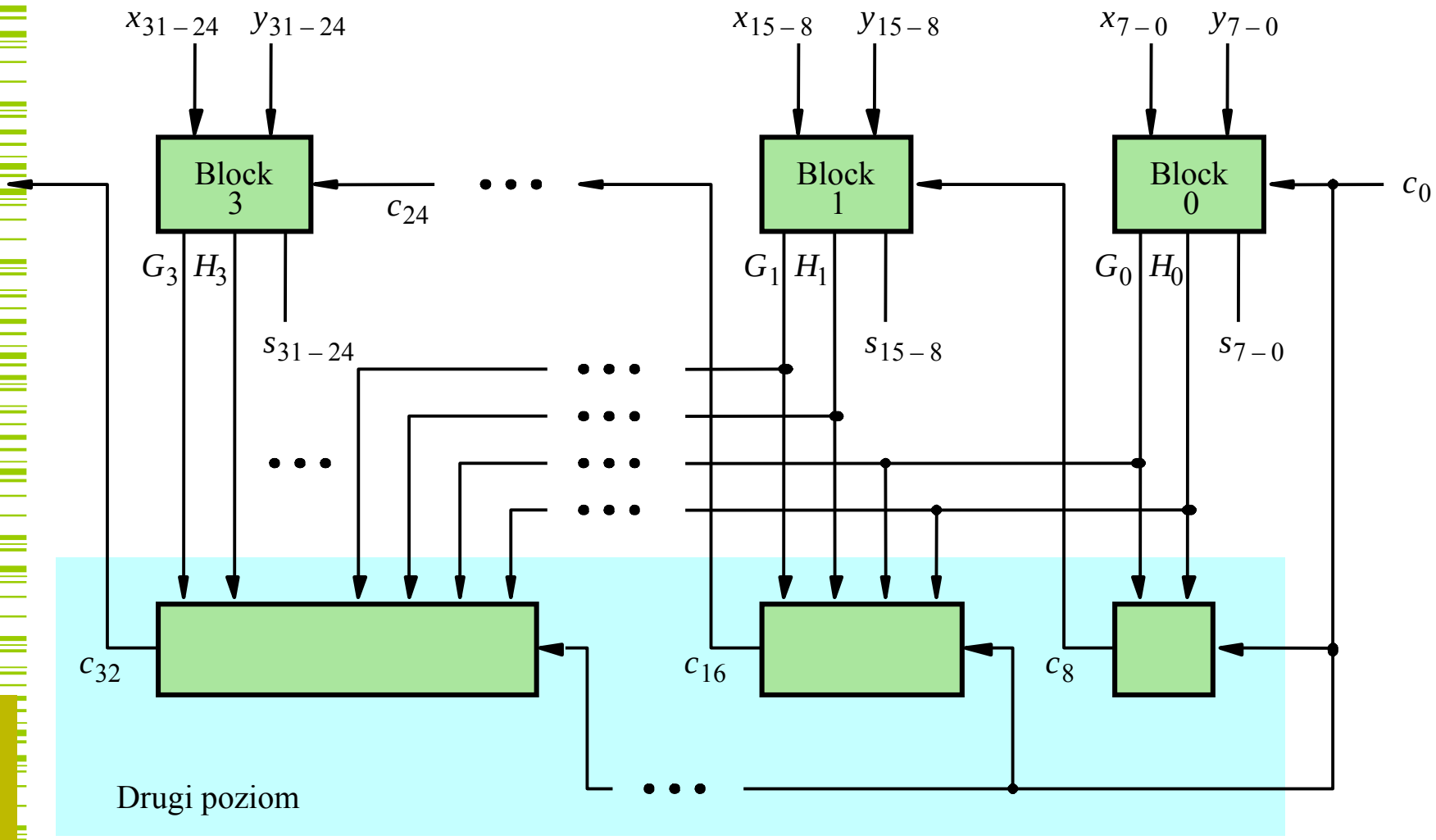


Sumatory z antycypacją przeniesień…

…można łączyć szeregowo



Hierarchiczny sumator z antycypacją przeniesień

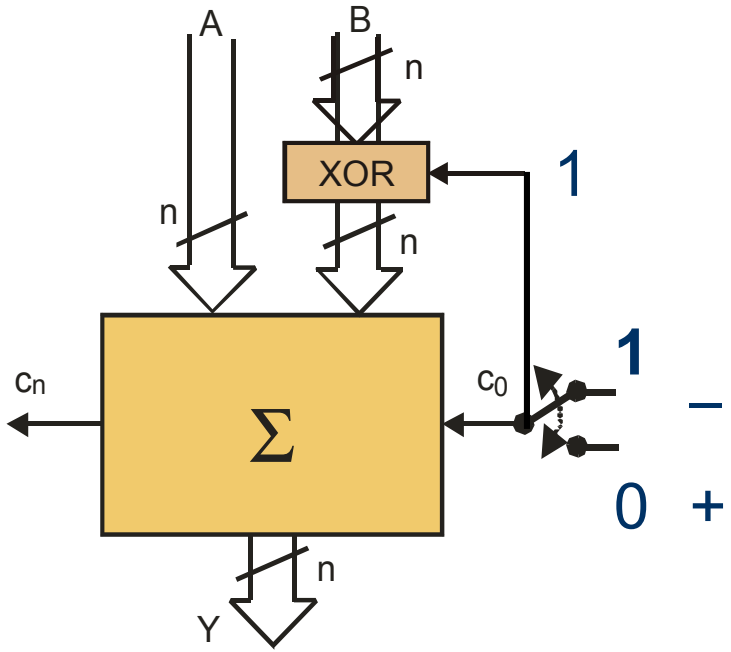


Sumator/układ odejmujący

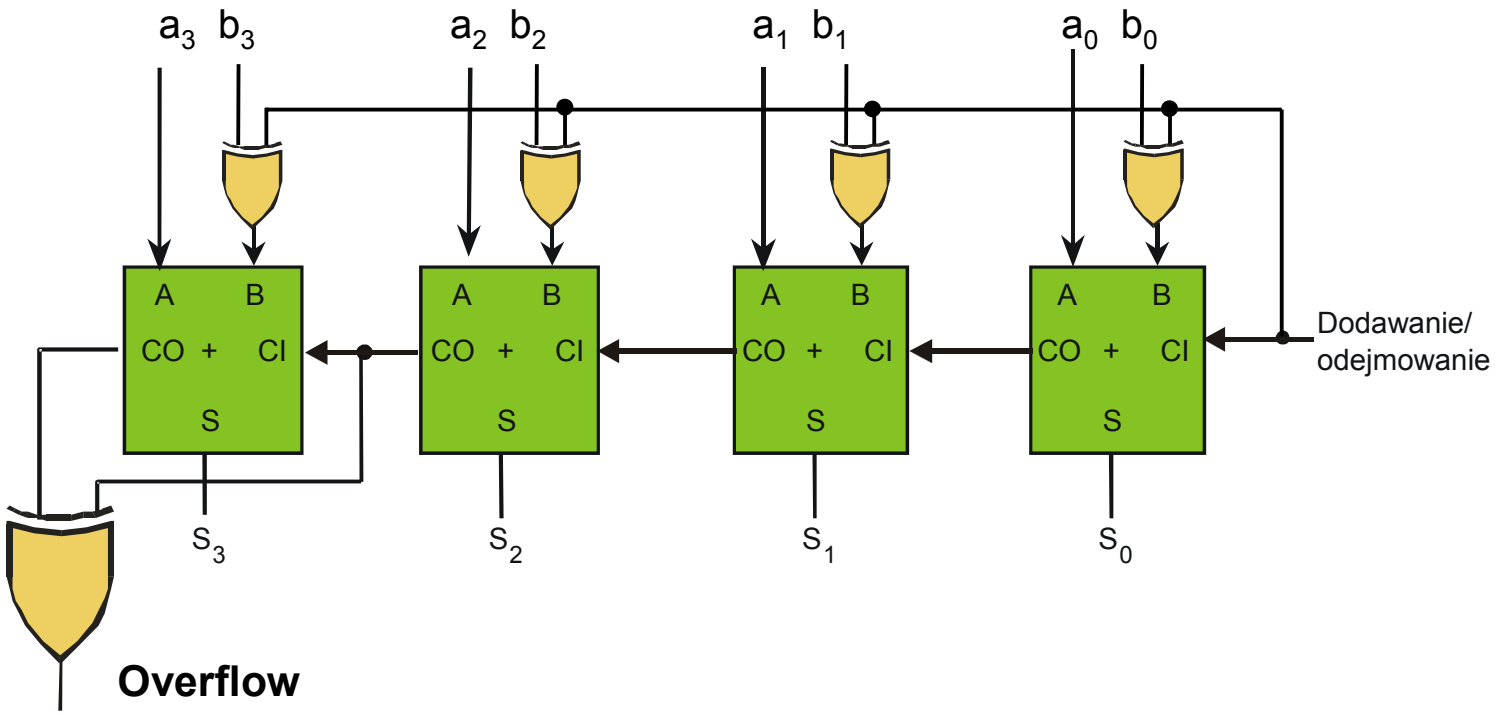
$$\mathbf{A}_{U2} = (a_{n-1}, \ldots, a_j, \ldots, a_0)$$

Kod U2

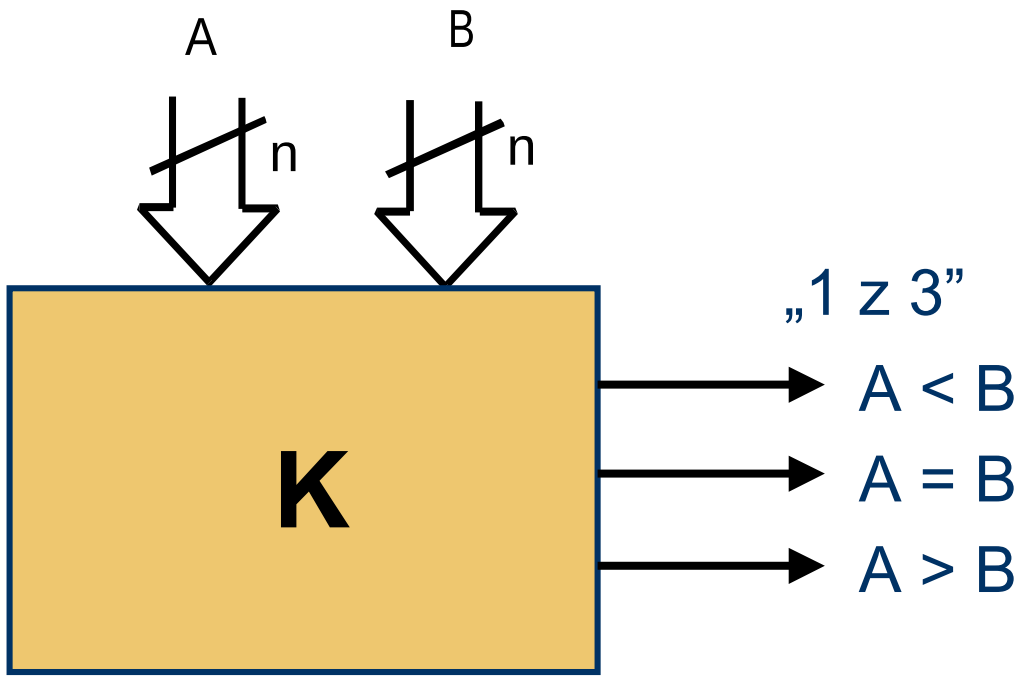
$$A_D = L(A_{U2}) = -a_{n-1} \cdot 2^{n-1} + \sum_{j=0}^{n-2} a_j 2^j$$



Sumator/układ odejmujący



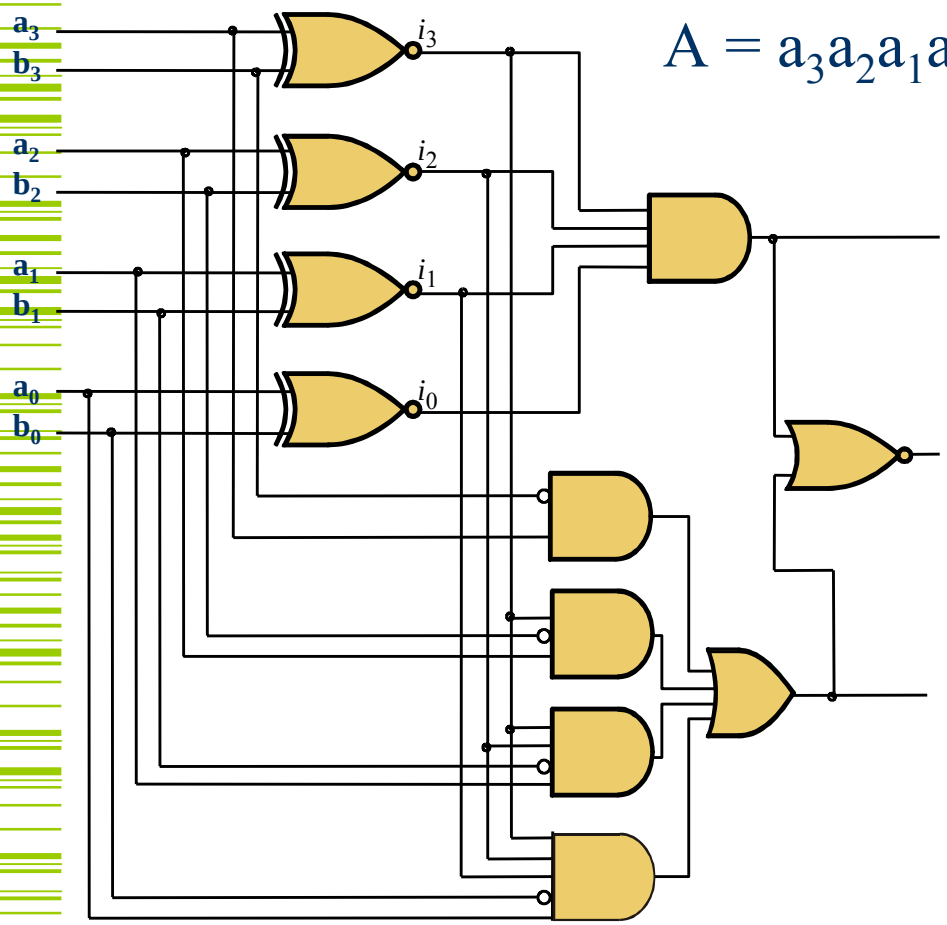
Komparator



I
T
P
W

ZPT

Komparator



$$A = a_3a_2a_1a_0 \quad B = b_3b_2b_1b_0 \quad i_k = \overline{a_k \oplus b_k}$$

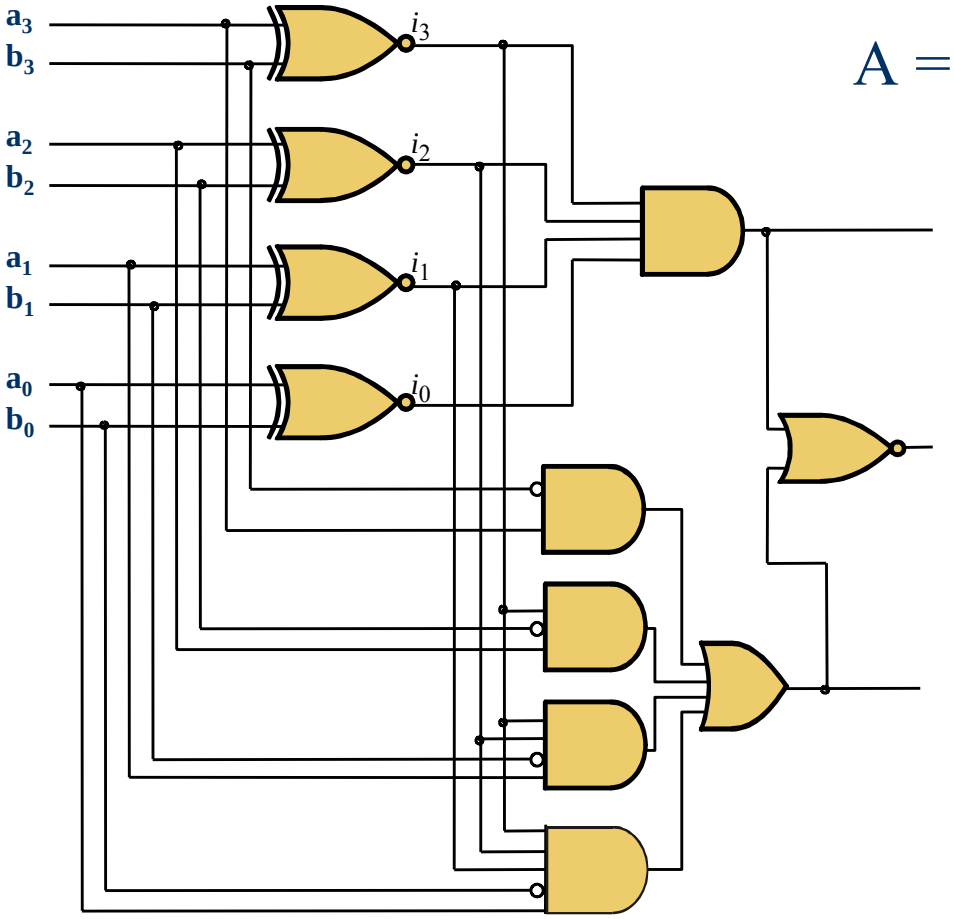
$$A\ eq\ B = i_3i_2i_1i_0$$

$$A < B = \overline{A\ eq\ B + A\ gt\ B}$$

$$A > B = a_3\bar{b}_3 + i_3a_2\bar{b}_2 + i_3i_2a_1\bar{b}_1 + i_3i_2i_1a_0\bar{b}_0$$

$$a_k \neq b_k \Rightarrow \quad A < B,\ \text{gdy}\ a_k = 0,\ b_k = 1$$
$$A > B,\ \text{gdy}\ a_k = 1,\ b_k = 0$$

Komparator



$A = a_3a_2a_1a_0 \quad B = b_3b_2b_1b_0$

A = B

A < B

A > B

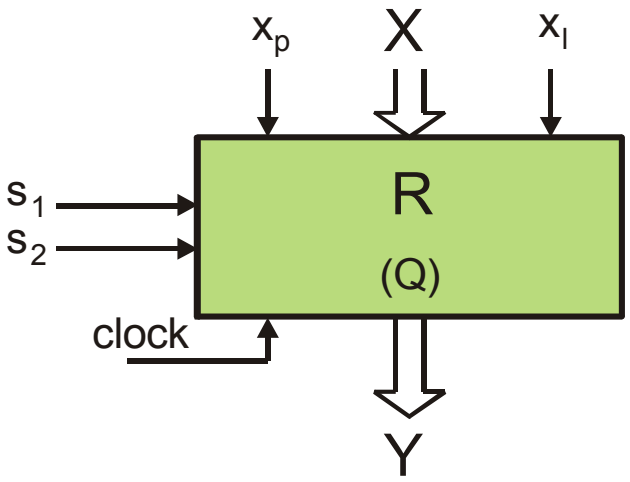
Sekwencyjne bloki funkcjonalne

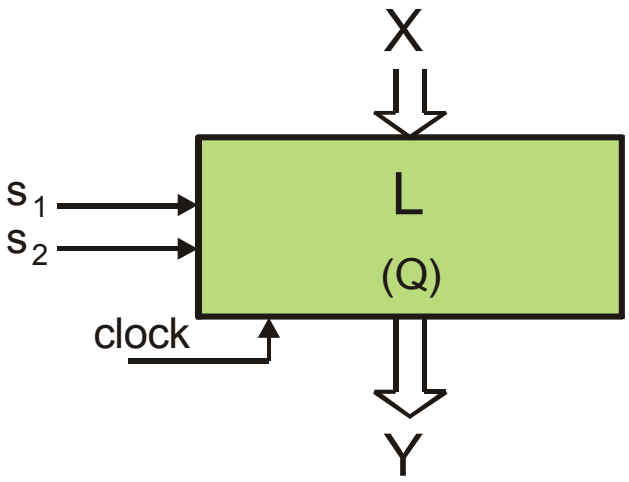
Y := X LOAD
Y := Y HOLD

Rejestry

Liczniki

Y := <0...0> RESET (CLEAR)





Y := SHR(x_p, Y)
Y := SHL(Y, x_l)

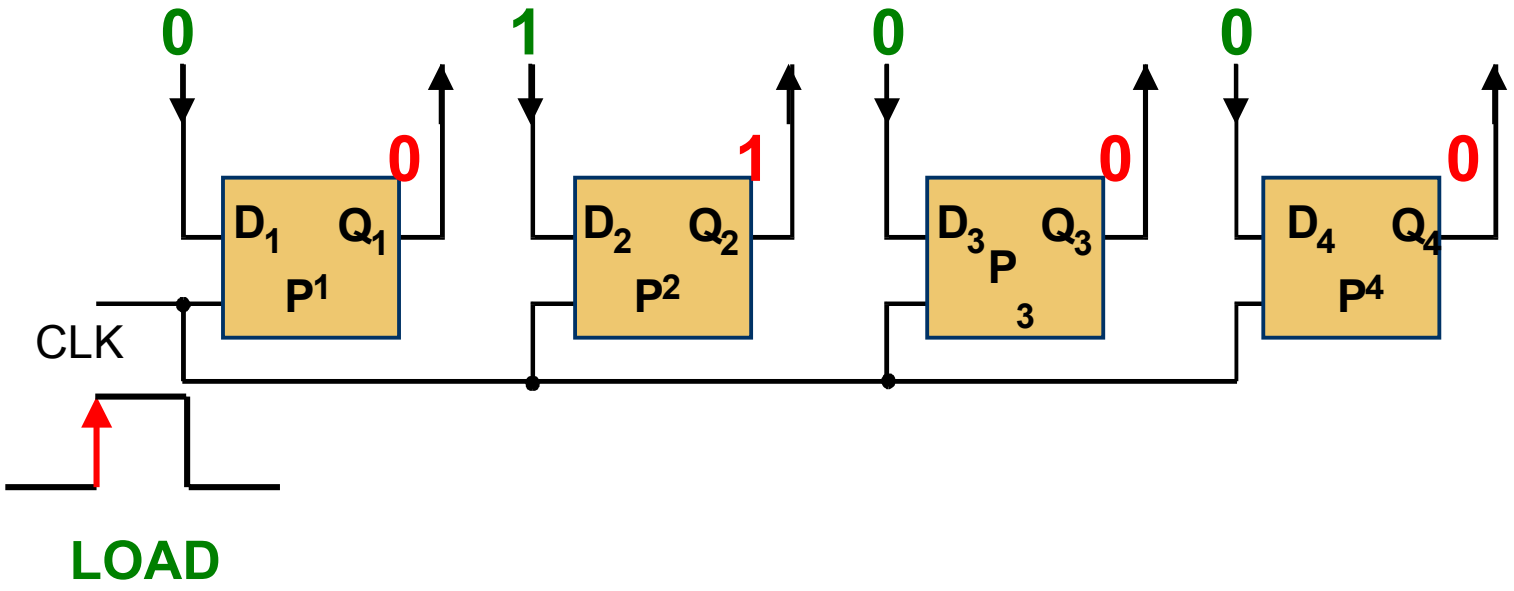
Y := Y + 1 = INC(Y)
Y := Y – 1 = DEC(Y)

I T P W

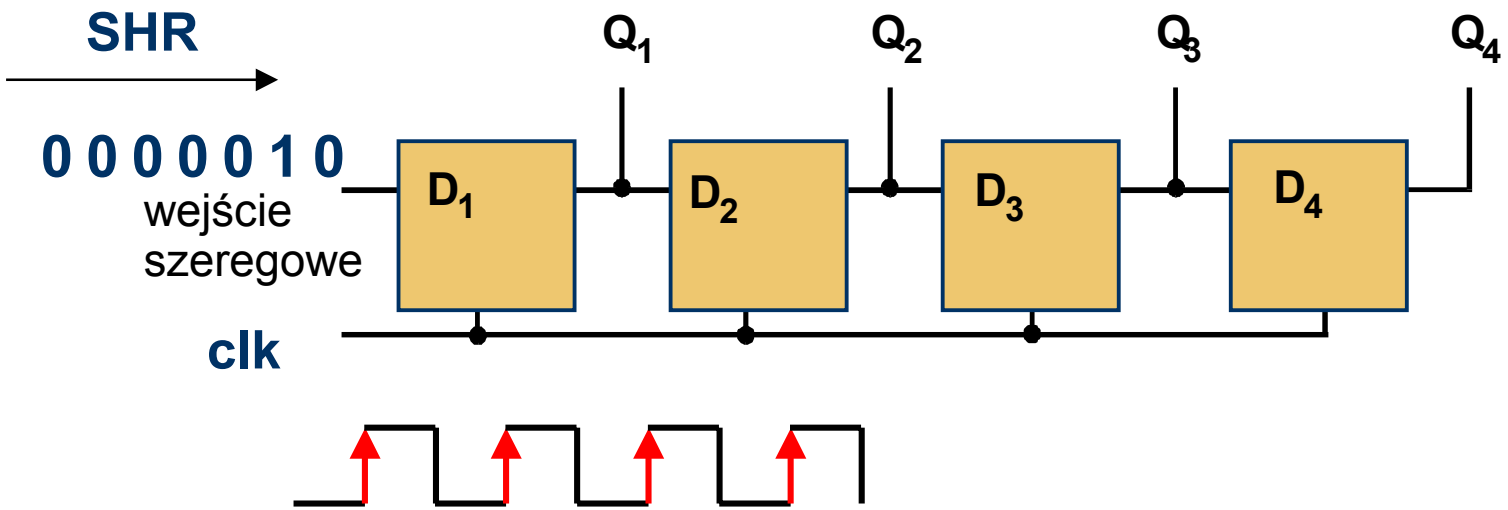
ZPT

Prosty rejestr

Rejestr zbudowany z przerzutników
– ładowanie (load) i pamiętanie

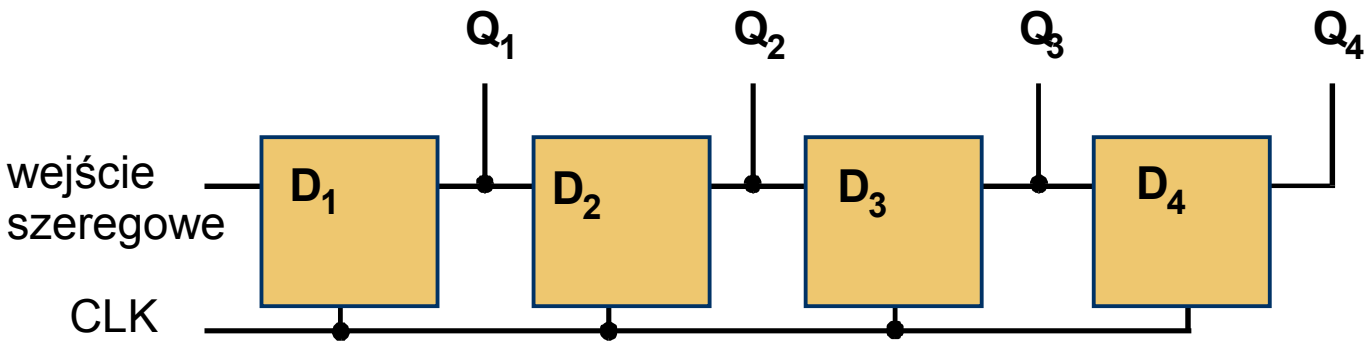


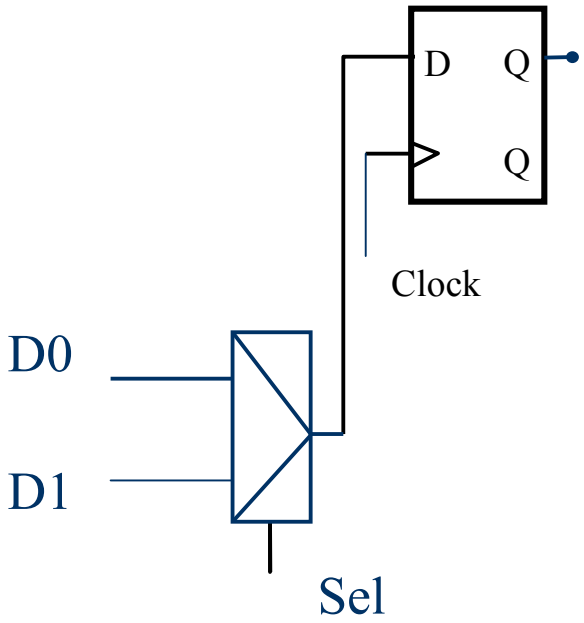
Rejestr przesuwający



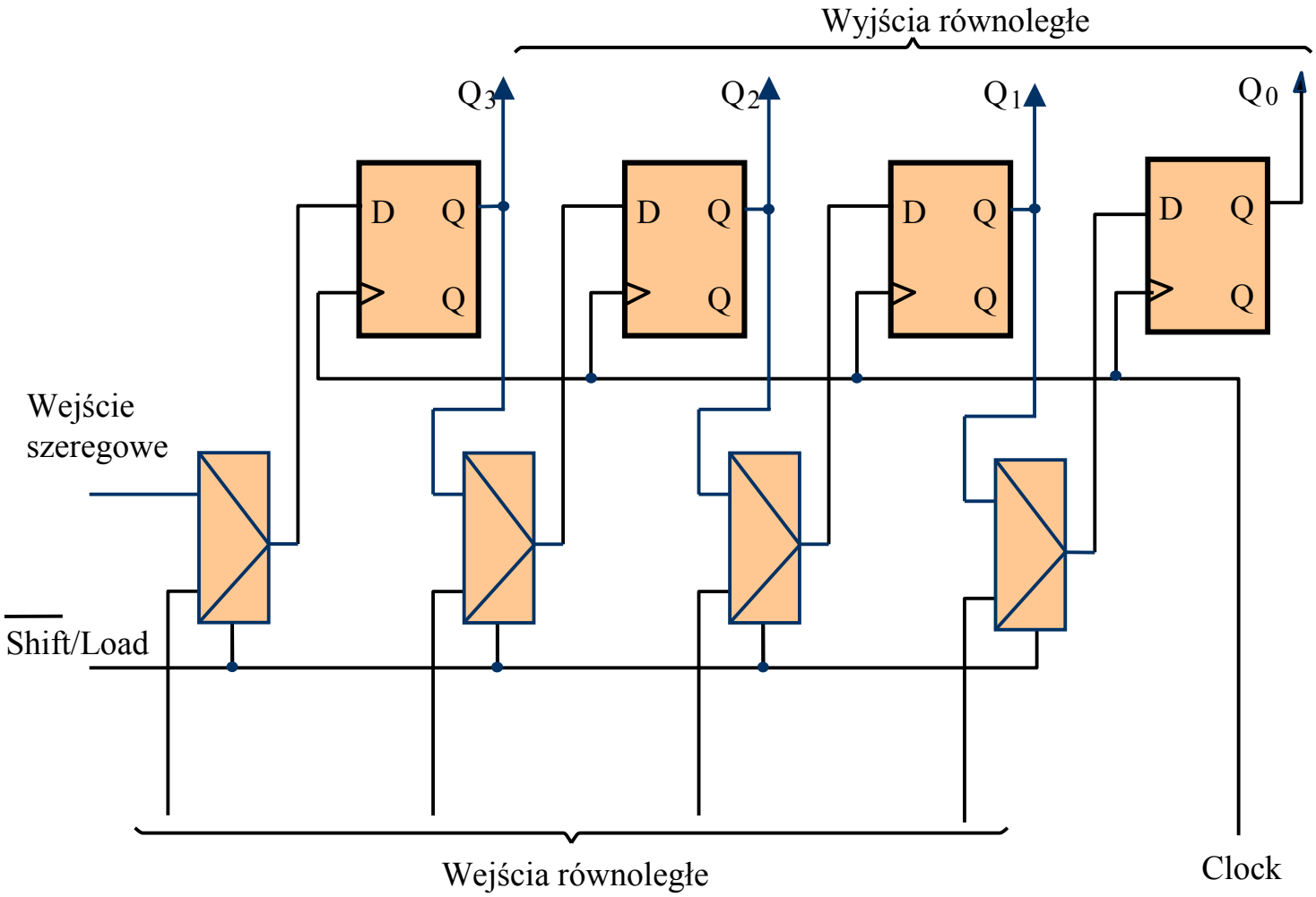
WE	Q_1	Q_2	Q_3	Q_4
0	0	0	0	0
1	1	0	0	0
0	0	1	0	0
0	0	0	1	0
0	0	0	0	1
0	0	0	0	0

Rejestr przesuwający

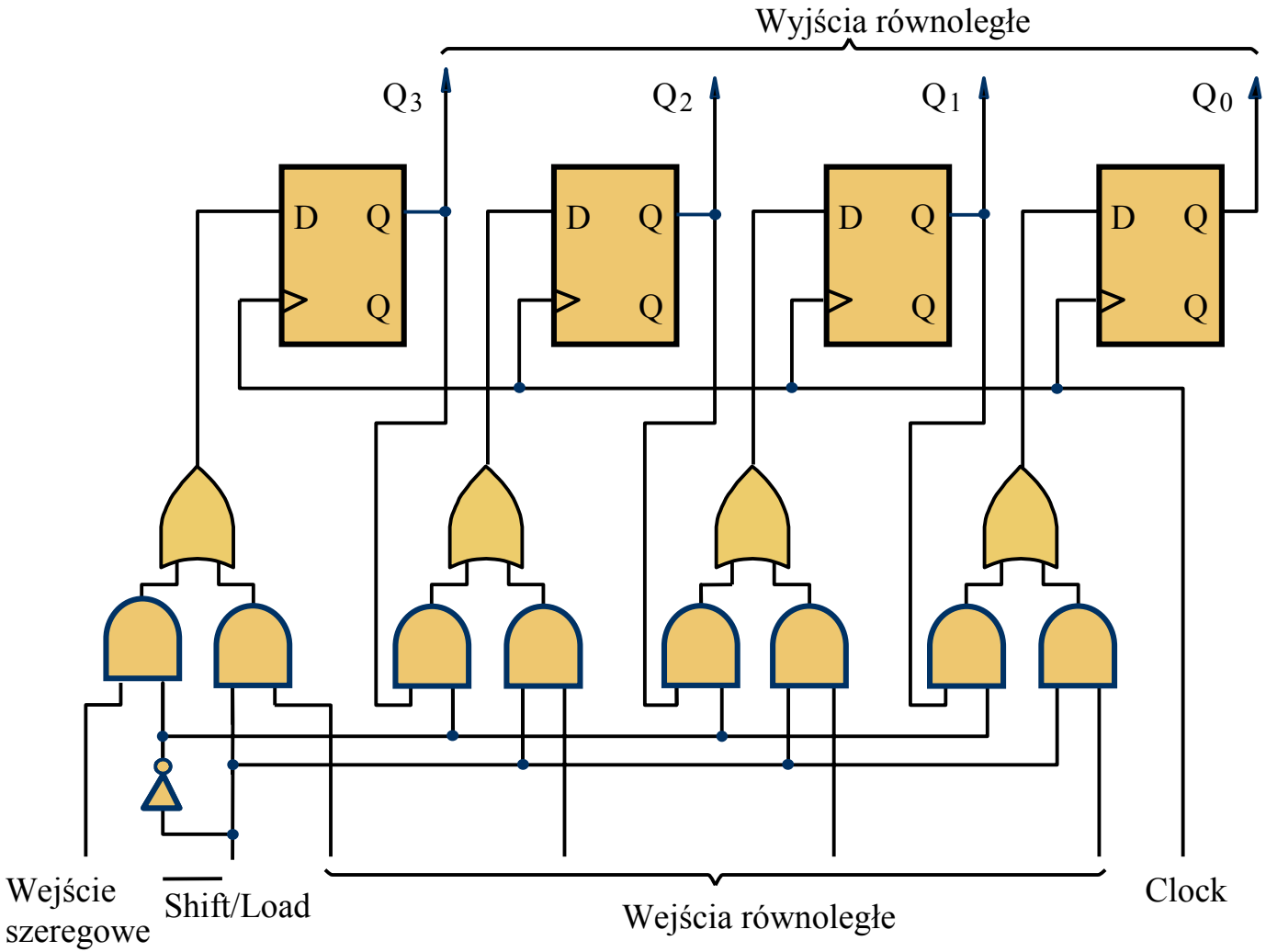




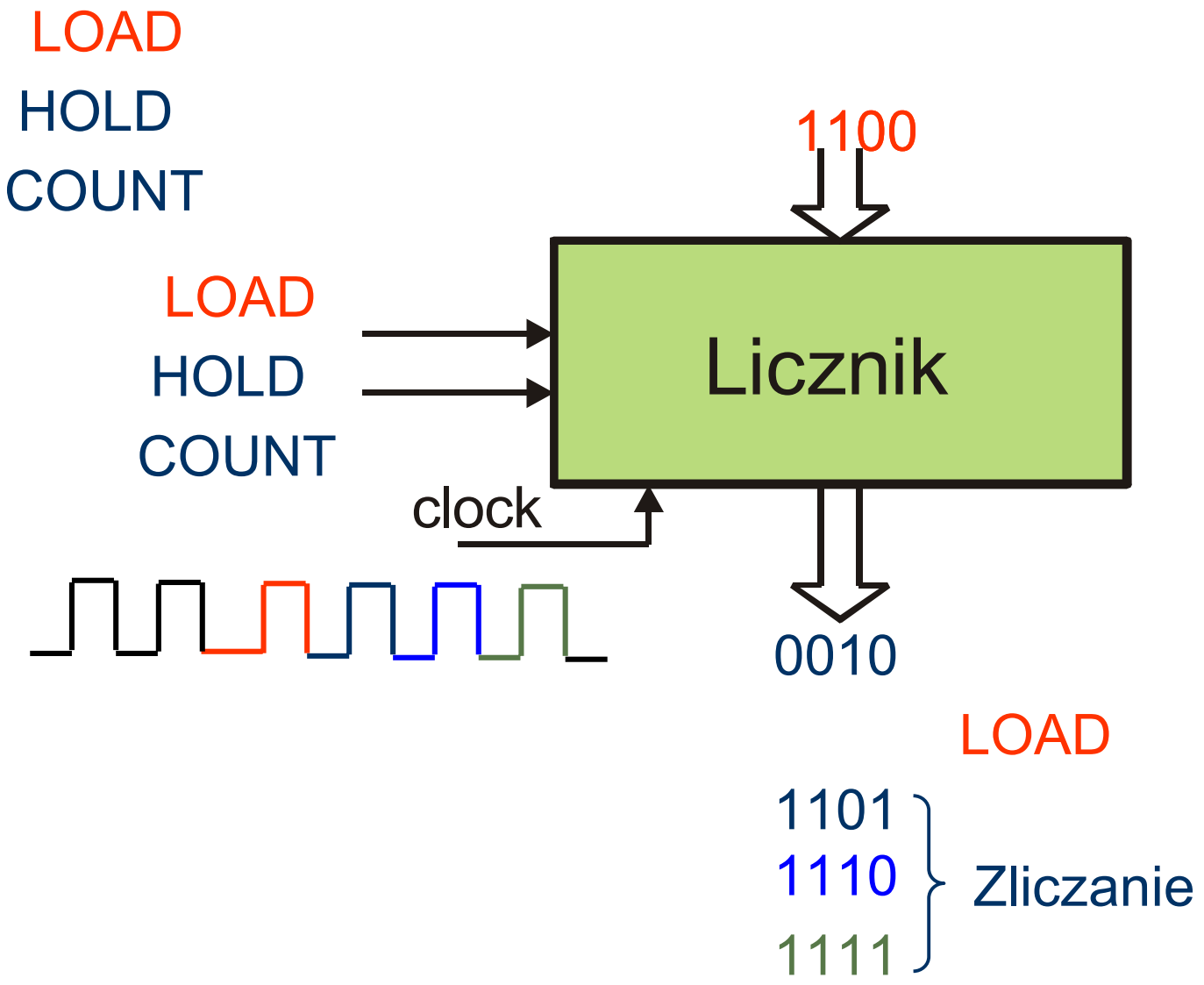
Rejestr przesuwający z wpisem równoległym



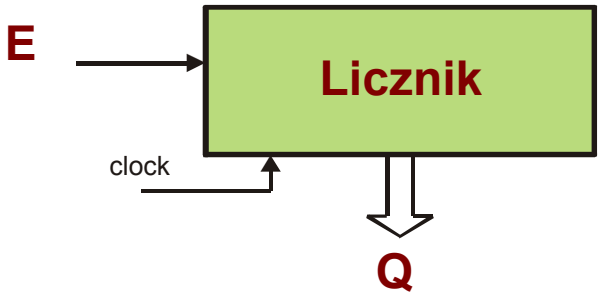
Rejestr przesuwający z wpisem równoległym



Mikrooperacje licznika



Przykład licznika z wejściem Enable



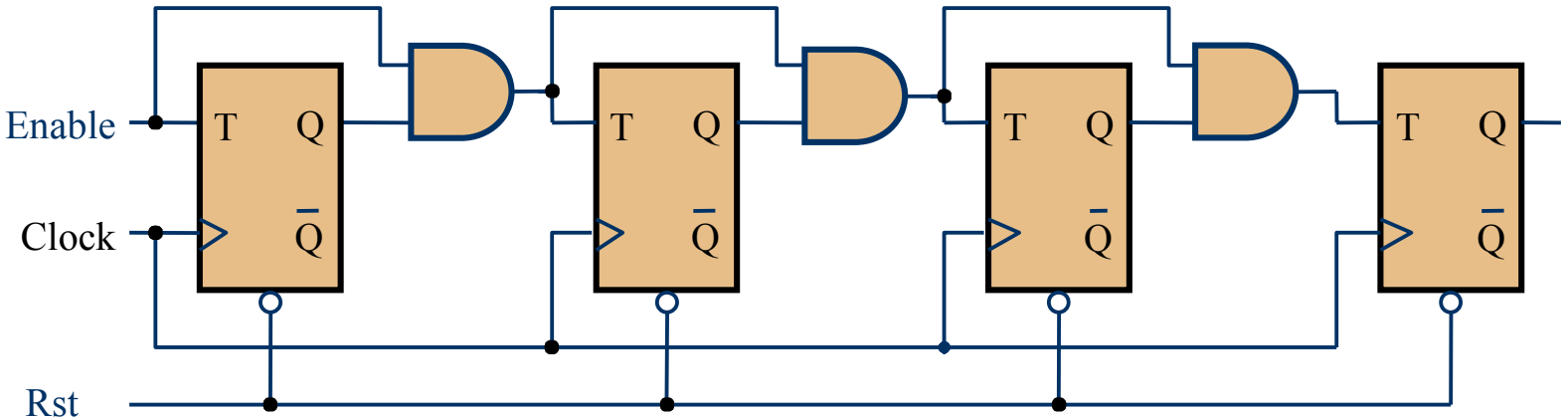
A \ E	0	1
A_0	A_0	A_1
A_1	A_1	A_2
A_2	A_2	A_3
A_3	A_3	A_4
A_4	A	A_5
⋮		
A_{14}	A_{14}	A_{15}
A_{15}	A_{15}	A_0

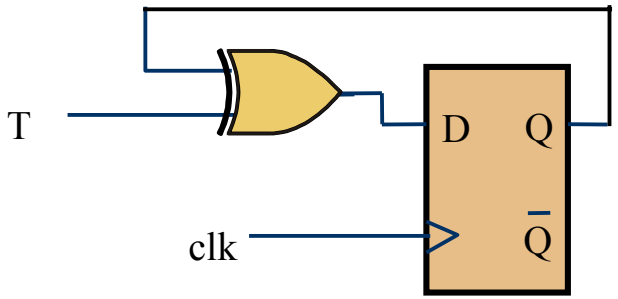
$$T_0 = E$$
$$T_1 = EQ_0$$
$$T_2 = EQ_0Q_1 = T_1Q_1$$
$$T_3 = EQ_0Q_1Q_2 = T_2Q_2$$

I T P W

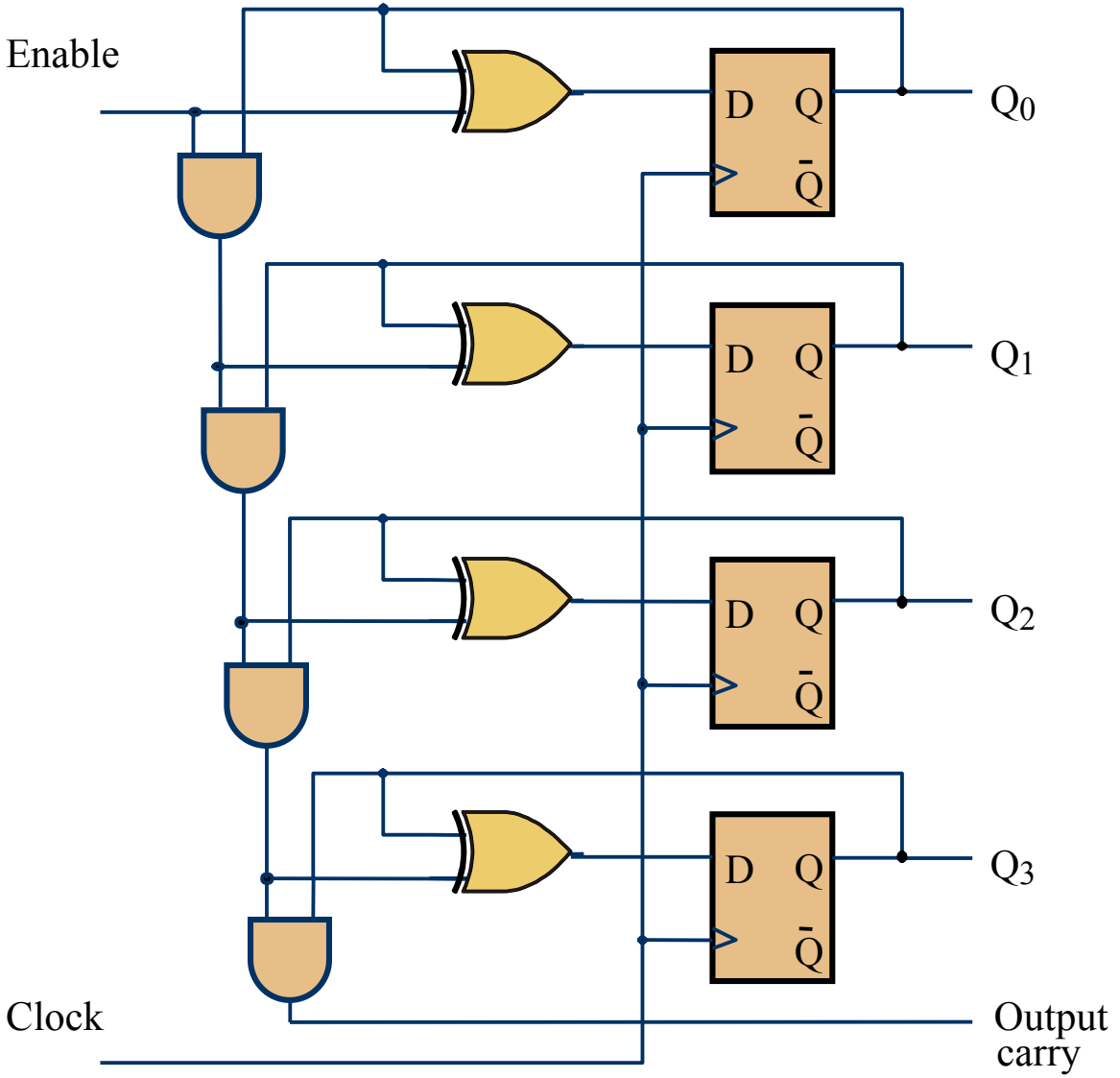
ZPT

Licznik w górę

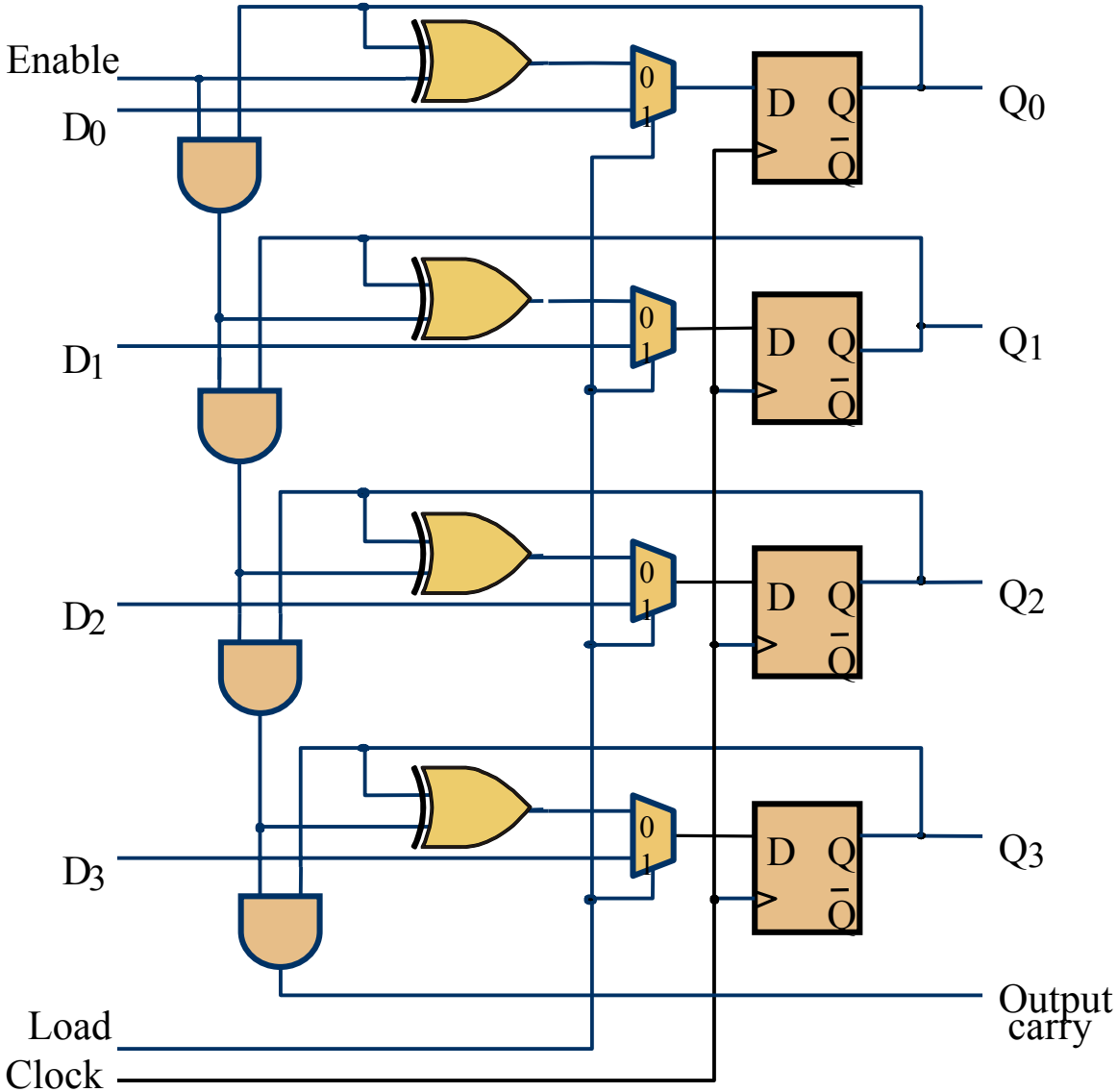




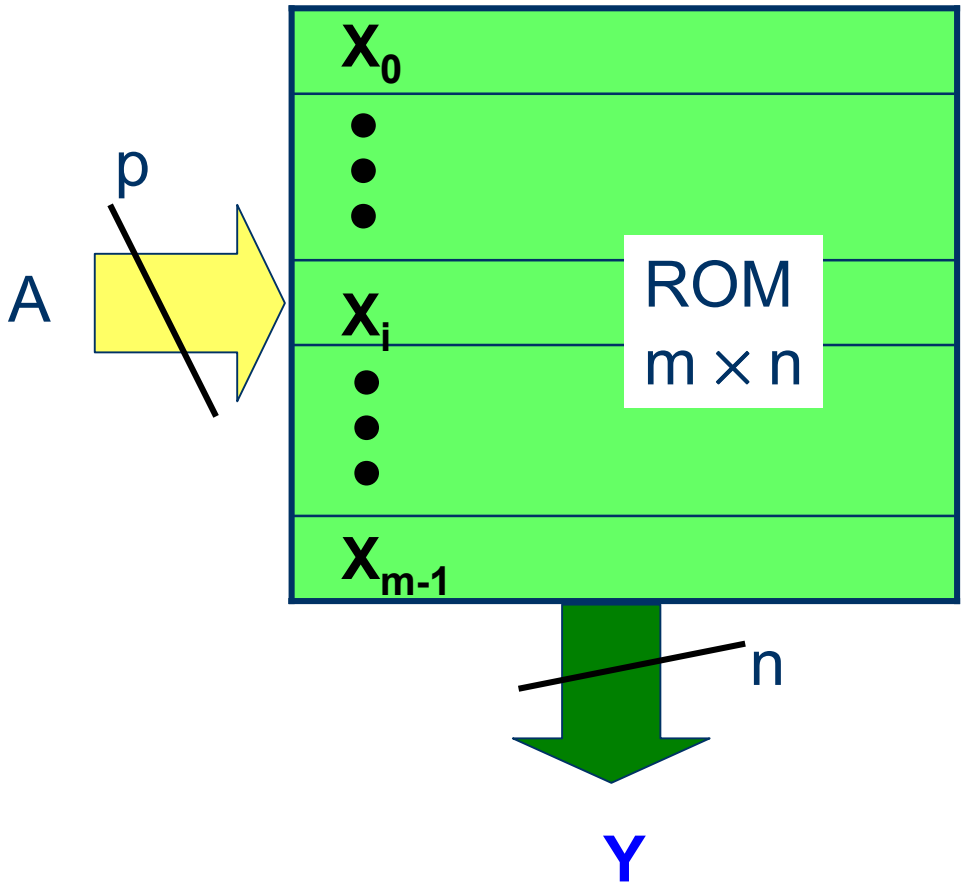
Licznik z przerzutnikami D



Licznik z wpisywaniem równoległym

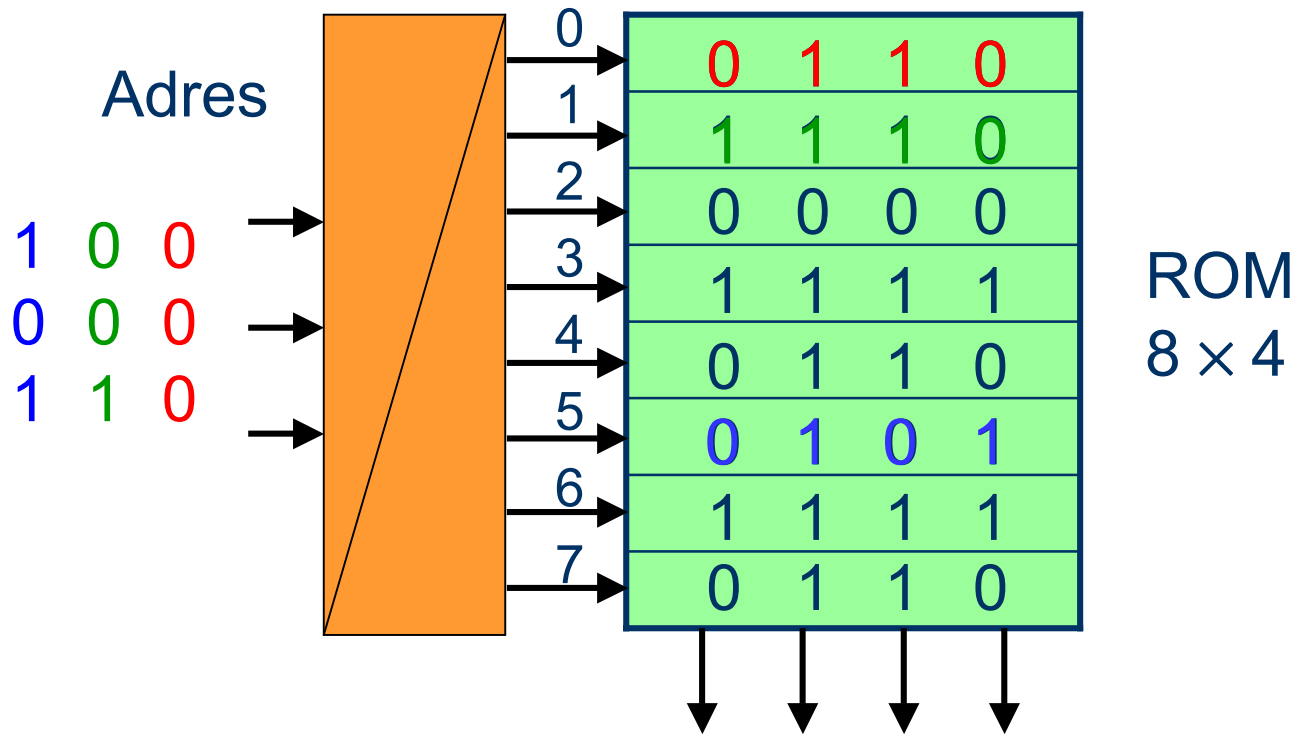


Pamięci typu ROM



ROM – uniwersalny układ kombinacyjny

Pamięci typu ROM



Pamięci typu ROM

(struktura)