

Podstawy techniki cyfrowej cz.2

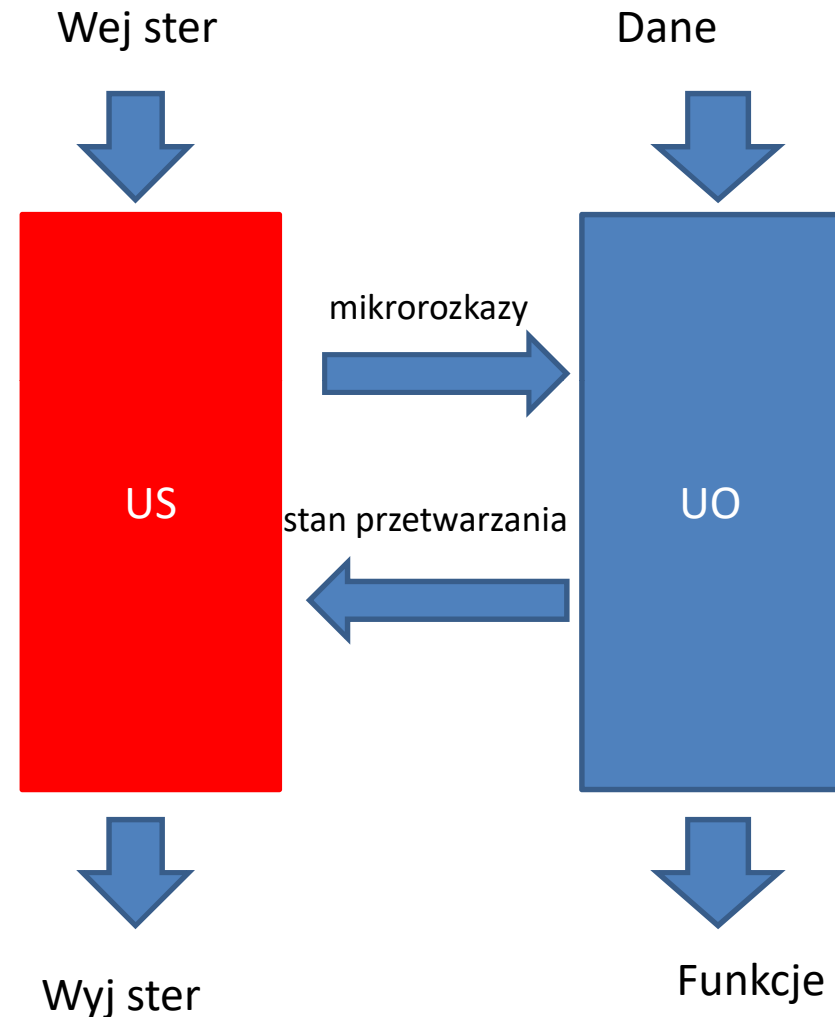
zima 2017

Rafał Walkowiak

16.12.2017

Układy cyfrowe

- Ogólna struktura logiczna:
 - układ sterowania
 - bloki funkcjonalne dla realizacji określonych funkcji przetwarzania danych czyli **układ operacyjny** lub ścieżka danych (ang. data path)



Synteza strukturalna

1. Określenie:

- struktury bloków funkcjonalnych: liczniki, rejestry, komparatory, sumatory ... i
- połączeń między blokami funkcjonalnymi

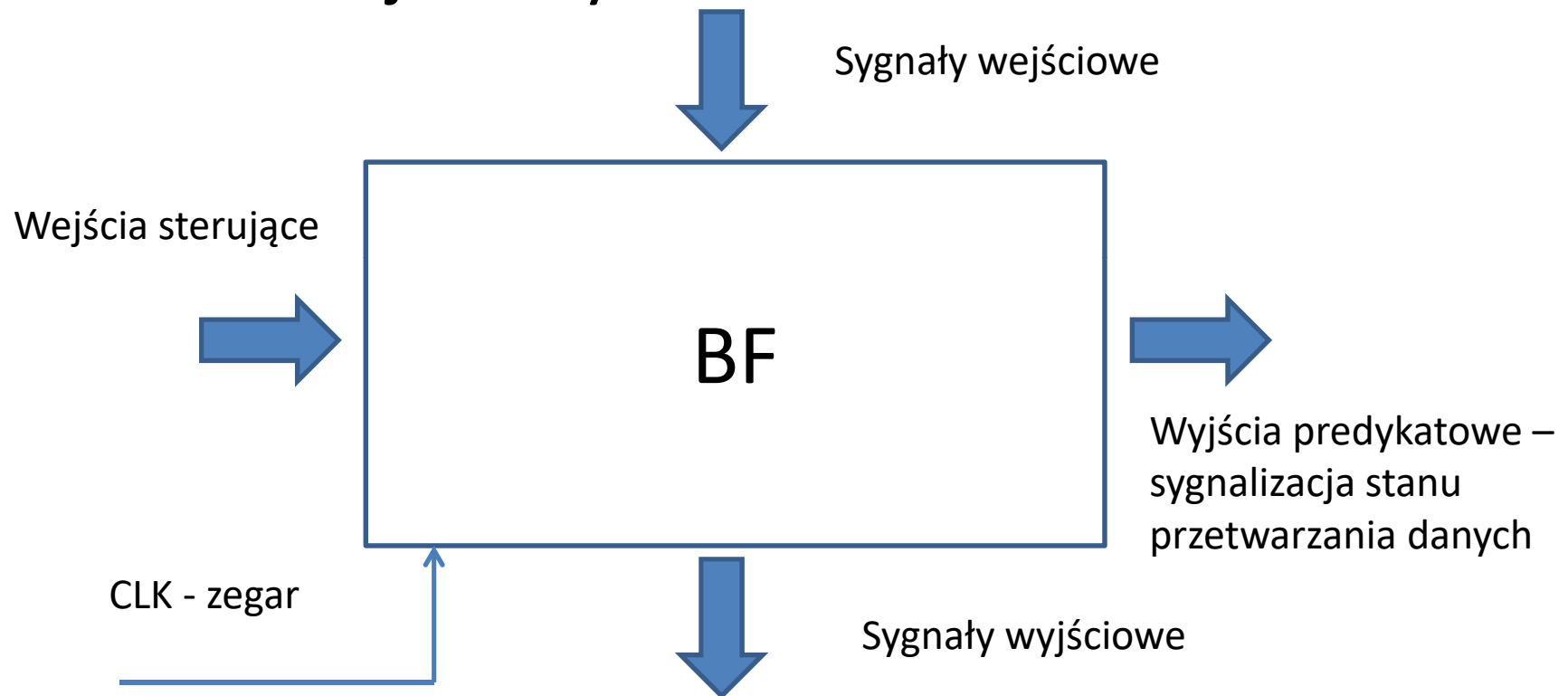
2. Określenie sposobu pracy układu sterowania realizującego algorytm przetwarzania zapisanego za pomocą sieci działań – diagram ASM (ang. Algorithmic State Machine)

Synteza behawioralna

- Opisanie działania układu za pomocą języka opisu sprzętu HDL (ang. hardware description language)
- **Określenie i połączenie** odpowiednich bloków funkcjonalnych **realizowane** poprzez kompilator języka HDL w ramach komputerowego systemu projektowania.

Synteza strukturalna

- Blok funkcjonalny



Podział układów cyfrowych

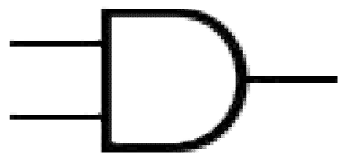
- **Układy kombinacyjne** – pozbawione właściwości pamiętania stanów, realizujące funkcje logiczne w oparciu o bramki i inne proste układy cyfrowe.
- **Układy sekwencyjne** - posiadają cechę pamiętania stanów logicznych, zbudowane dodatkowo z przerzutników.

Podstawowe funkcje układów cyfrowych układy kombinacyjne

Bramki

- **Bramkami** nazywamy kombinacyjne układy cyfrowe realizujące proste funkcje logiczne jednej lub wielu zmiennych. Zmienną logiczną jest sygnał elektryczny występujący na wejściach i wyjściach tych układów.
- Działanie bramek jest opisane za pomocą tablicy prawdy zawierającej stany logiczne 0/1 lub poziom wielkości fizycznej (np. napięcia) niski (L) lub wysoki (H).

Bramki logiczne – symbole graficzne



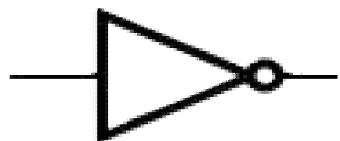
Iloczyn

AND



Suma

OR



Negacja

NOT



Negacja iloczynu
NAND



Negacja sumy
NOR



Suma mod 2
XOR



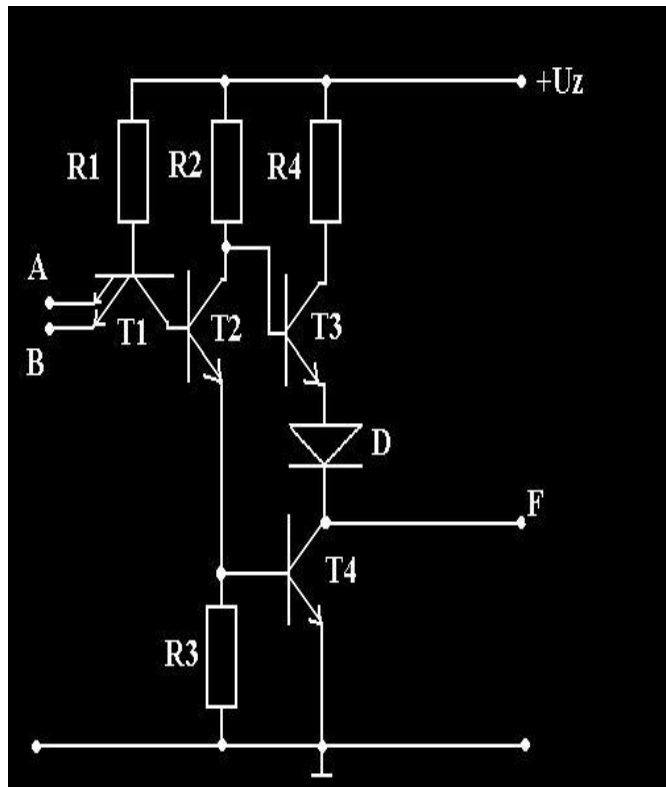
Równoważność
XNOR

Rysunki za: http://en.wikipedia.org/wiki/Logic_gate

Sygnał cyfrowy – technologia TTL

- Układy TTL zbudowane są z tranzystorów bipolarnych i zasila się je napięciem stałym 5 V.
- Gdy poziom sygnału ma wartość $0V \div 0,8V$ sygnał TTL jest niski - logiczne „0” – logika dodatnia.
- Poziom sygnału jest między $2V \div 5V$ jest określany jako stan wysoki - logiczna „1”.
- Gdy wartość napięcia jest z przedziału $0,8V \div 2V$ - sygnał jest nieokreślony.

Bramki TTL



Bramka NAND

Na wejściach jedynki powodują nasycenie T2 i T4 oraz zatkanie T3 (baza i emiter na tym samym potencjale), na wyjściu stan niski.

Na wejściu 0 powoduje nasycenie T1, obniżenie napięcia na bazie T2 - zatkanie T2 i T4, T3 przewodzi, na wyjściu stan wysoki.

Układy CMOS

Bramki w technologii CMOS są zbudowane są z tranzystorów MOSFET (ang. metal–oxide–semiconductor field-effect transistor).

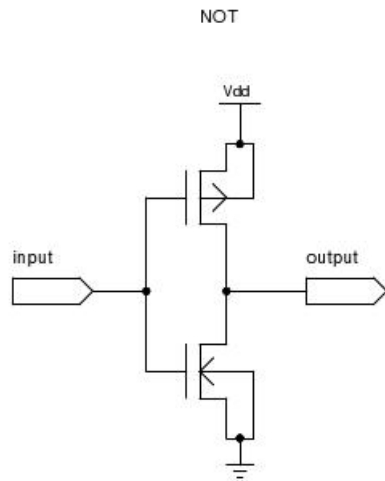
Bramki zbudowana są z par tranzystorów o przeciwnym typie przewodnictwa i połączonych w taki sposób, że w ustalonym stanie logicznym przewodzi tylko jeden z nich.

- Układy CMOS są relatywnie proste i tanie w produkcji, umożliwiając uzyskanie bardzo dużych gęstości upakowania.
- Układy cyfrowe wykonane w technologii CMOS mogą być zasilane napięciem 3 ÷ 18V, praktycznie nie pobierają mocy statycznie.
- Poziomy logiczne sygnałów są zbliżone do napięć zasilających (masa - logiczne "0", zasilanie – logiczna "1").

Układy niskonapięciowe (Low Voltage)

- Obniżanie napięcia zasilania prowadzi do spadku zużycia mocy
- Produkowane są serie układów cyfrowych CMOS przystosowane do zasilania napięciem 3, 3V , 2, 5V czy nawet 1, 8V

Bramki w technologii CMOS



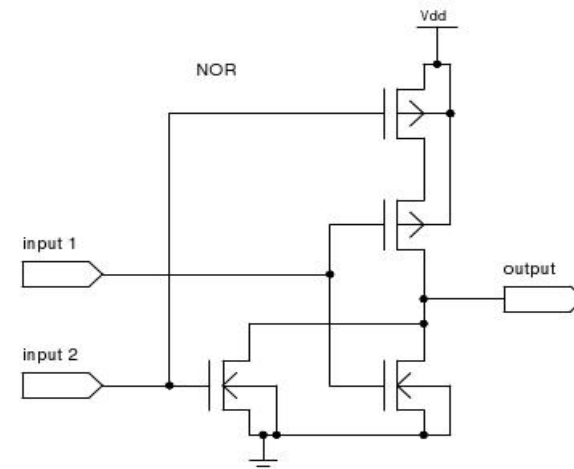
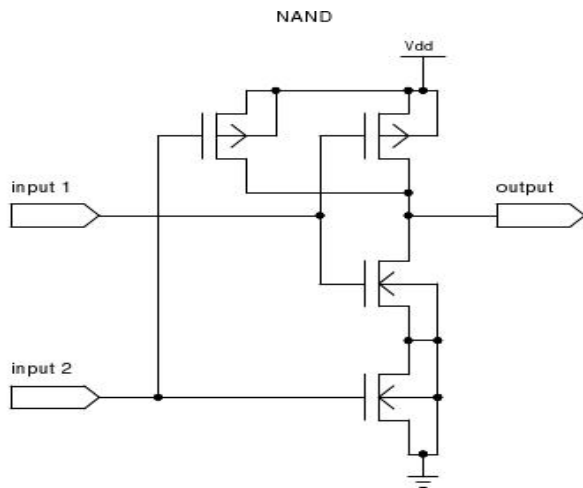
- CMOS układy z komplementarnymi (typu p i typu n) tranzystorami MOS
- Strzałka określa wymagany kierunek wzrostu napięcia pomiędzy bramką a podłożem otwierającego klucz (w kierunku źródło – dren) wykonany w oparciu o tranzystor

Bramka NOT

- input=0 otwarcie tranzystora górnego, wyjścieysterowane do poziomu wysokiego
- input=1 otwarcie tranzystora dolnego, wyjścieysterowane do poziomu niskiego

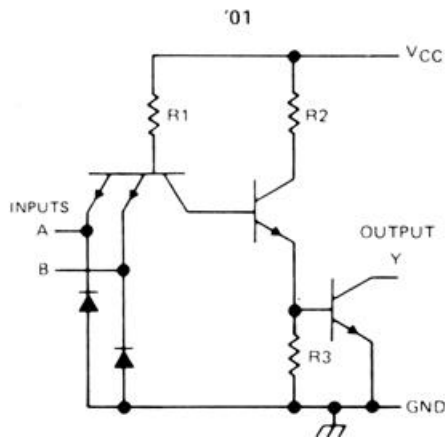
Bramka NAND – jedno 0 na wejściu powoduje przewodzenie górnego tranzystora iysterowanie wyjścia do poziomu wysokiego

Bramka NOR – jedna 1 na wejściu wystarczy aby wyjścieysterować do poziomu wysokiego

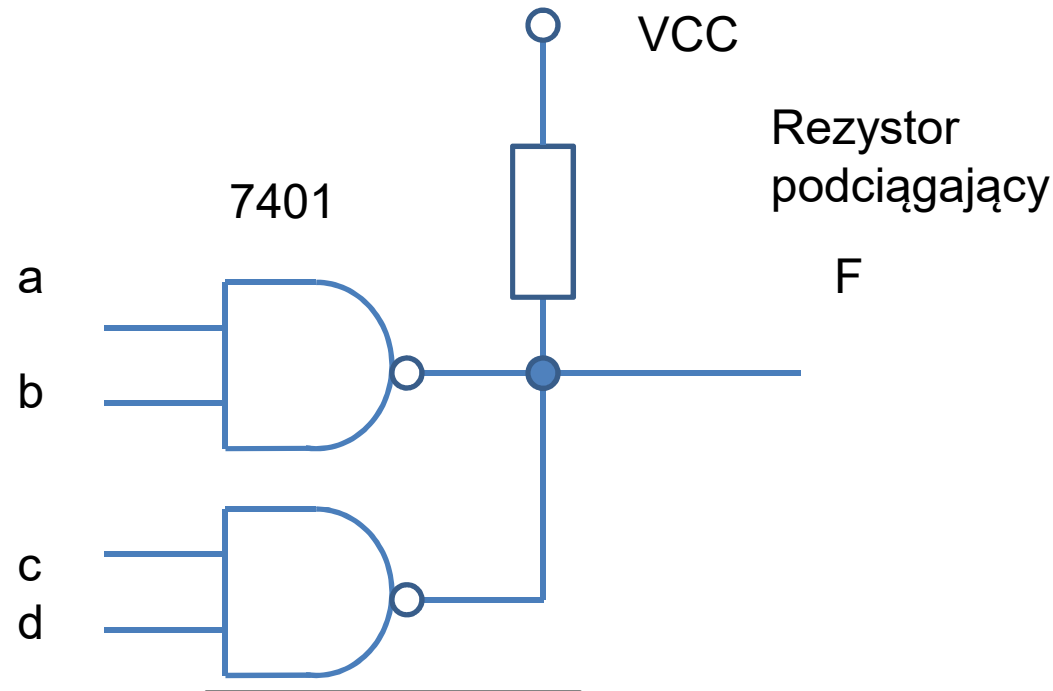


Bramki typu „open collector”

„iloczyn na drucie”



a	b	c	d	F
0	x	0	x	1
x	0	x	0	1
1	1	0	0	0
0	0	1	1	0
1	1	1	1	0



$$F = ab + cd = (ab)' (cd)'$$

W obwodzie powyżej połączenie wyjść typu open collector (tzw. „iloczyn na drucie”) bramek NAND zastępuje bramkę sumy

Przy zwartych wyjściach OC dowolna liczba wyjść bramek NAND równych 0 daje stan F=0.

Bramki typu "open collector,, - zastosowanie

Bramki logiczne typu „open collector” (TTL) lub otwarty dren (MOSFET), których wyjście pozostawać może w stanie:

- wysokiej impedancji – być nieaktywne lub
- zwarte z masą – być aktywne ,

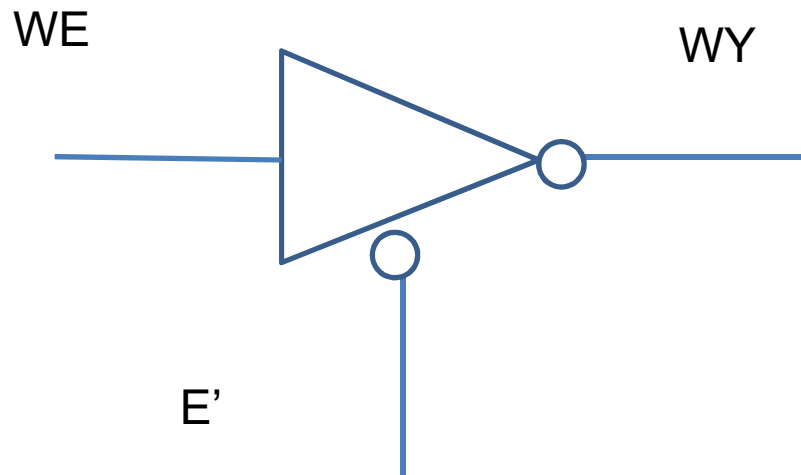
Aby wyjście bramek, aby mogło być traktowane jako logiczne "0" albo "1" należy poprzez rezystor połączyć je do zasilania (rezystor podciągający (ang. pull up resistor)).

Zastosowanie:

- Zastosowanie odpowiedniego poziomu napięcia zasilania na rezystorze podciągającym pozwala na sterowanie poprzez wyjście układami pracującymi przy innych poziomach napięć sygnałów.
- Kilka wyjść OC można połączyć do jednej linii sygnałowej umożliwiając sterowanie jednej linii z wielu źródeł (np. magistrala).

Bramki trójstanowe

Trzeci stan logiczny



WE	E'	WY
0	0	1
0	1	Z
1	0	0
1	1	Z

Koncepcja:

W niektórych implementacjach bramek logicznych oprócz logicznego "0" i logicznej "1" istnieje trzeci stan logiczny

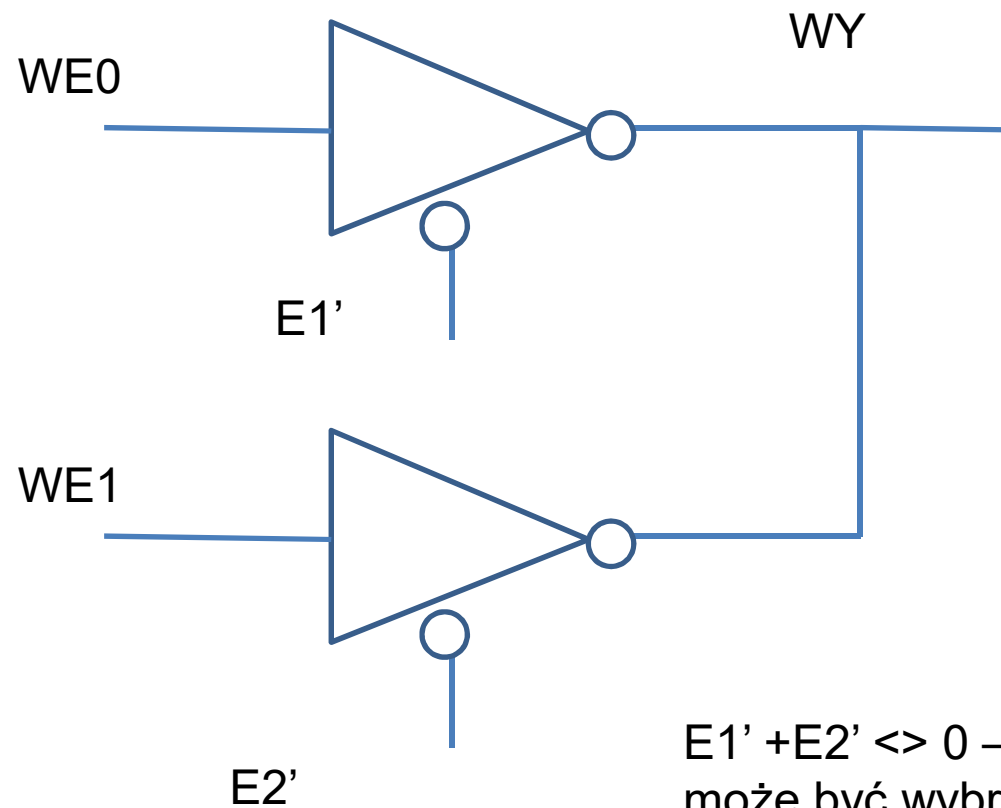
stan wysokiej impedancji (ang. high impedance) Z.

Posiadają możliwość brakuysterowania wyjść. Gdy **wyjście układu** nie jest połączone galwanicznie z układem cyfrowym znajduje się ono w stanie wysokiej impedancji – nie jestysterowane (ani do poziomu wysokiego ani niskiego).

Zastosowanie:

podłączenie układu do magistrali jako jednego z możliwych źródeł jejysterowania. Gdy konieczne jest zapewnienie wykluczającego się wyboru źródła.

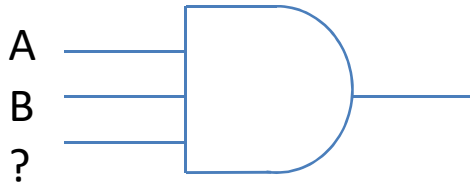
Bramki trójstanowe zastosowanie



WE0	WE1	E1'	E2'	WY
0	X	0	1	1
1	X	0	1	0
X	0	1	0	1
X	1	1	0	0
X	X	1	1	Z

$E1' + E2' \neq 0$ – TYLKO jedno wejście ENABLE może być wybrane

Niewykorzystane wejścia bramek



Dla bramki AND zachowanie realizowanej funkcji jest możliwe gdy $? = 1$ gdyż $AB1 = AB$
lub $? = B$ gdyż $ABB = AB$

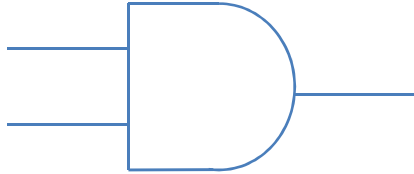
Możliwe zatem rozwiązania to podłączenie:

- wejścia do wartości stałej „1” lub
- zwarcie wejść np. $? = B$ (lub $? = A$).
- Drugie rozwiązanie nie jest korzystne ze względu na dodatkowe obciążenie układu generującego sygnał B (A). W technologii realizacji bramek istnieje parametr „obciążalności wyjściowej (ang. fan-out) określający maksymalną liczbę możliwych do podłączenia wejść, która umożliwi poprawną pracę układu.

Analogiczna sytuacja z niewykorzystanymi wejściami innych bramek:

- OR $A+B+B = A+B+0$
- XOR $A \oplus B \oplus 0 = A \oplus B$

Uszkodzenia bramek



- Uszkodzenia bramek (lub połączeń z wejściami bramek) mogą powodować błędne działanie bramki i układu.

Możliwe błędy związane z wejściem B bramki to:

- Stała wartość 1 na jednym wejściu – $B=1$,
- Stała wartość 0 na jednym wejściu – $B=0$,
- Połączenie wejść $B=A$.

W przypadku bramki AND efektem powyższego będą odpowiednio:

- Na wyjściu Y wartość równa wartości drugiego wejścia, $Y=A$,
- Na wyjściu wartość 0 - stała na wyjściu niezależnie od wartości wejść, $Y=0$,
- Na wyjściu wartość równa wartości drugiego wejścia, $Y=A$.

Obserwacja określonego niespodziewanego zachowania wyjścia bramki, na podstawie znajomości realizowanej funkcji może ukierunkować na przyczynę błędu np. brak połączenia, przypadkowe zwarcie sygnałów itp.

Konwencje logiki-

odwzorowanie poziomów napięć sygnałów układu na wartości logiczne

Konwencja logiki dodatniej:

- Wartość logiczna 0 reprezentowana przez niskie napięcie (np. 0),
- Wartość logiczna 1 reprezentowana przez wysokie napięcie (np. +5V).

Konwencja logiki ujemnej (odwrotne przyporządkowanie):

- Wartość logiczna 1 reprezentowana przez niskie napięcie (np. 0),
- Wartość logiczna 0 reprezentowana przez wysokie napięcie (np. +5V).

- **Bramka AND** analizowana w konwencji **logiki dodatniej** (L=0, H=1) realizuje **funkcję AND**.
- **Bramka AND** analizowana w konwencji **logiki ujemnej** (L=1, H=0) realizuje **funkcję OR**.
- Bramka OR analizowana w konwencji logiki dodatniej (L=0, H=1) realizuje funkcję OR.
- Bramka OR analizowana w konwencji logiki ujemnej (L=1, H=0) realizuje funkcję AND.
- Analogicznie zgodnie zasadą dualności bramki NOR i NAND

Poziomy Napięcie bramki AND	Wartości Logiczne LOGIKA DODATNIA	Wartości Logiczne LOGIKA UJEMNA
A B Y	A B Y	A B Y
H H H	1 1 1	0 0 0
H L L	1 0 0	0 1 1
L H L	0 1 0	1 0 1
L L L	0 0 0	1 1 1
	Logicznie BRAMKA AND	Logicznie BRAMKA OR

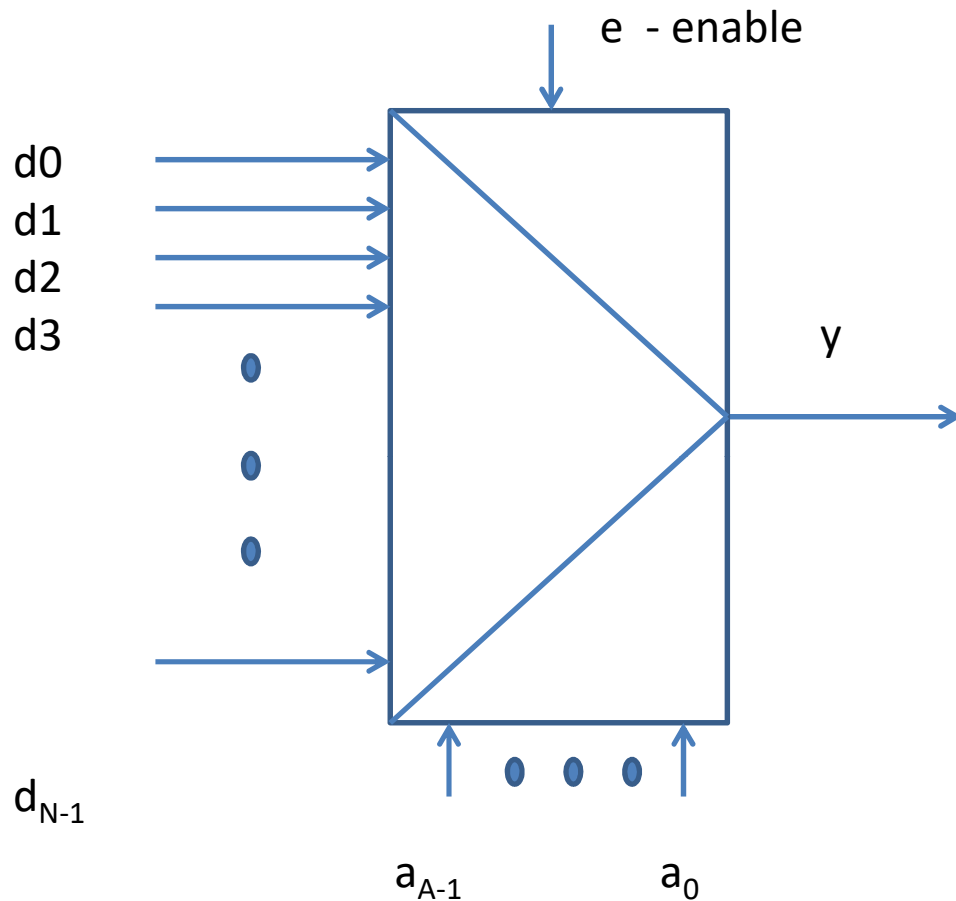
Multipleksery

- Służą do wyboru i przesłania na wyjście sygnału z jednego z N wejść informacyjnych.
- Wejścia adresowe określają przekazywany na wyjście sygnał (A - liczba wejść adresowych).
- Wejście „enable” służy do uaktywnienia/zablokowania wyjścia multipleksera.
- Liczba wejść informacyjnych N związana jest z liczbą wejść adresowych zależnością $N=2^A$
- FUNKCJA REALIZOWANA PRZEZ MULTIPLEKSER:

$$y=e \sum_{k=0..N-1} P_k(\text{Adr}) * d_k$$

- Gdzie $P_k(\text{Adr})$ oznacza pełny iloczyn zmiennych adresowych $a_{n-1}, \dots, a_0$ prostych lub zanegowanych będący = 1 dla k-tej kombinacji zmiennych – dla $A=2$ i $k=1$ $P_1(\text{Adr}) = a_1'a_0$

Multiplexer

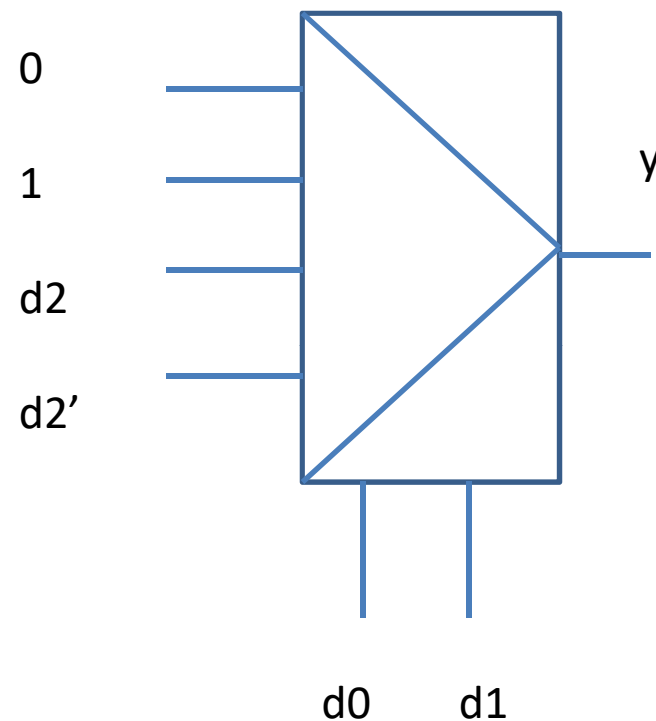


$$y = e \sum_{k=0..3} P_k(\text{Adr}) * d_k$$

$$y = e (a_1' a_0' d_0 + a_1' a_0 d_1 + a_1 a_0' d_2 + a_1 a_0 d_3)$$

Multiplexer w syntezie funkcji logicznych

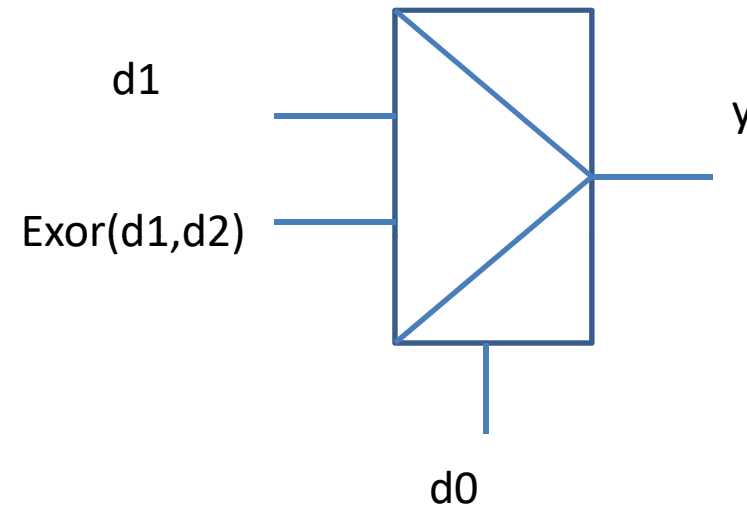
d0	d1	d2	y
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0



Multiplexer z A wejściami adresowymi,
Realizacja funkcji 3 wejść za pomocą multiplexera 2 wejściowego, podanie na wejście:
jednego sygnału: 0, 1, trzeciego wejścia prostego lub zanegowanego.

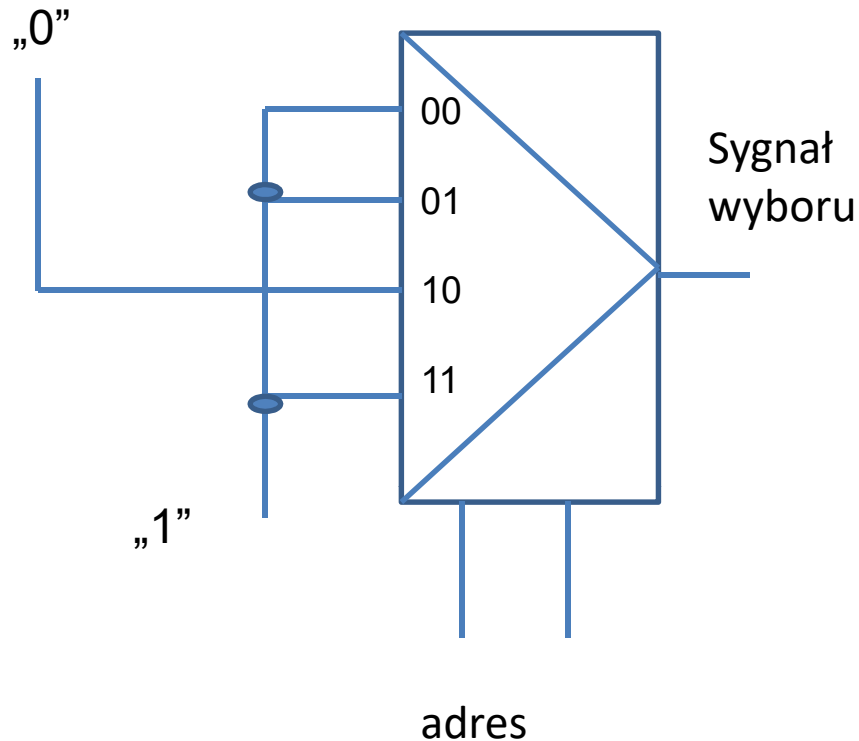
Multiplexer w syntezie funkcji logicznych

d0	d1	d2	y
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	1
1	1	1	0



Realizacja możliwa - liczba mintermów = 4 jest wielokrotnością $2^{N-A-1} = 2^{3-1-1} = 2$

Multiplexer jako dekodery adresu



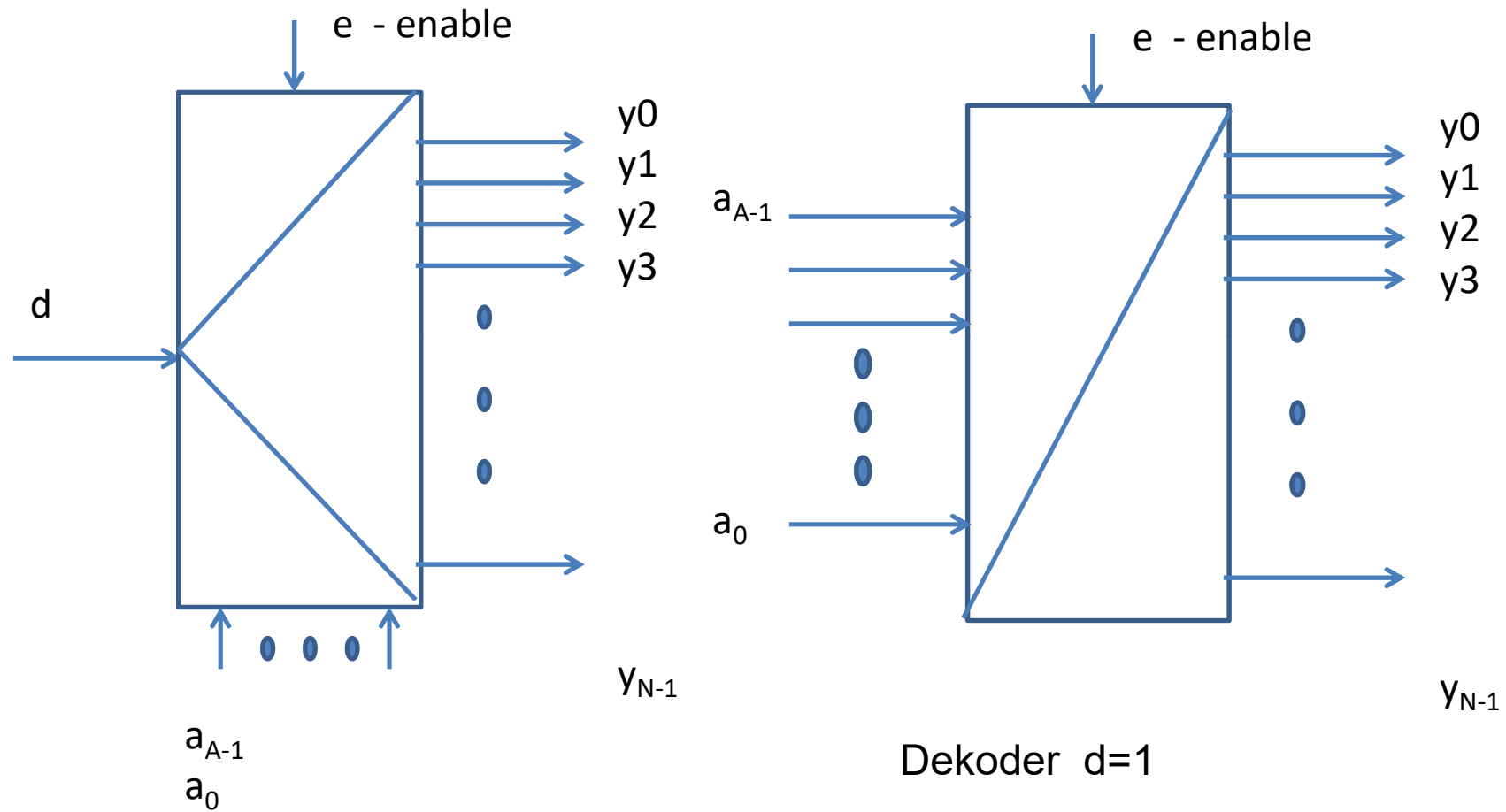
Określenie adresu urządzenia poprzez podłączenie poziomu aktywnego na wejście multiplexera odpowiadające adresowi urządzenia, pozostałe wejścia nieaktywne (1).

Jeśli aktywny poziom wyboru układu jest niski to dla adresu urządzenie równego 2 należy na wejścia podać sygnały jak na rysunku. Adres =2 powoduje pojawienie się na wyjściu 0.

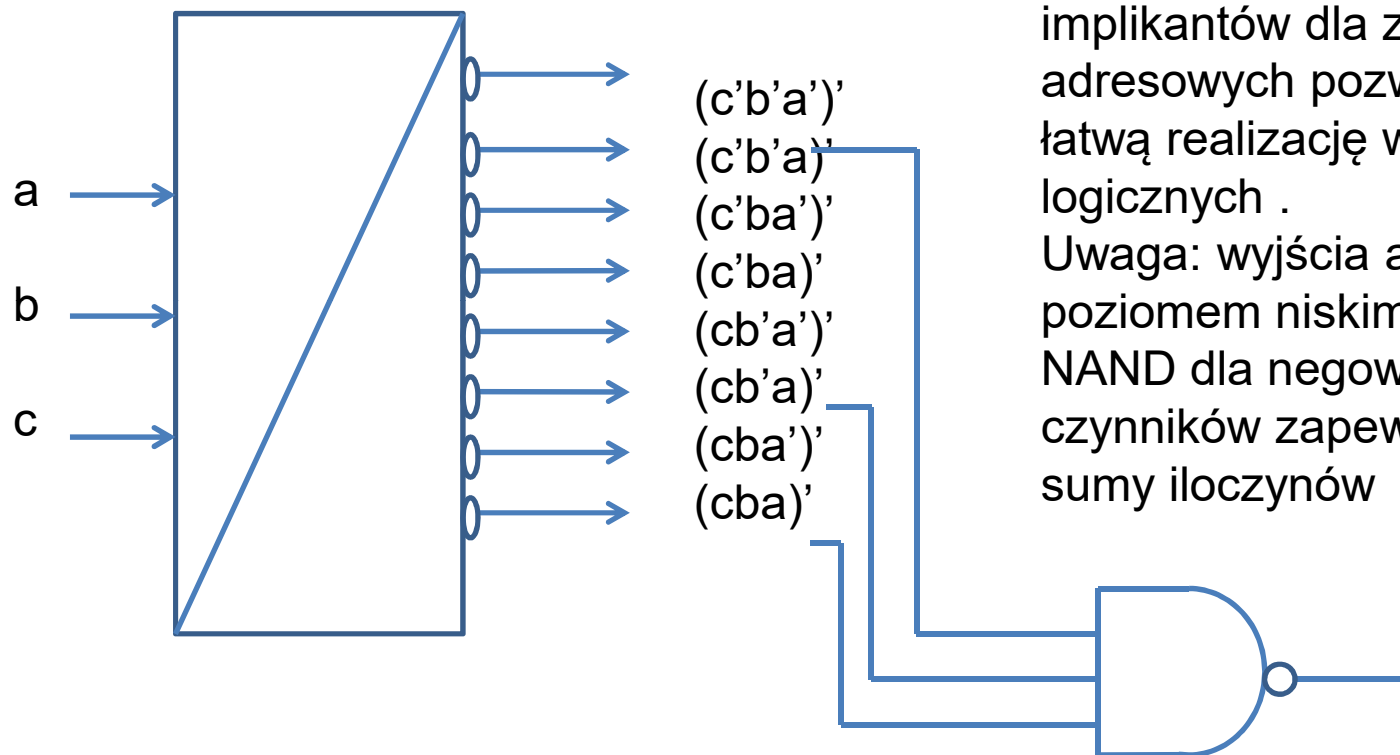
Demultipleksery

- Pozwalają na wybór wyjścia i przestanie na nie sygnału z wejścia
- Numer wyjścia jest określany przez stan wejść adresowych
- Możliwe zastosowanie jako dekodery z naturalnego kodu binarnego na kod „1 z N”

Demultiplexer , dekoder



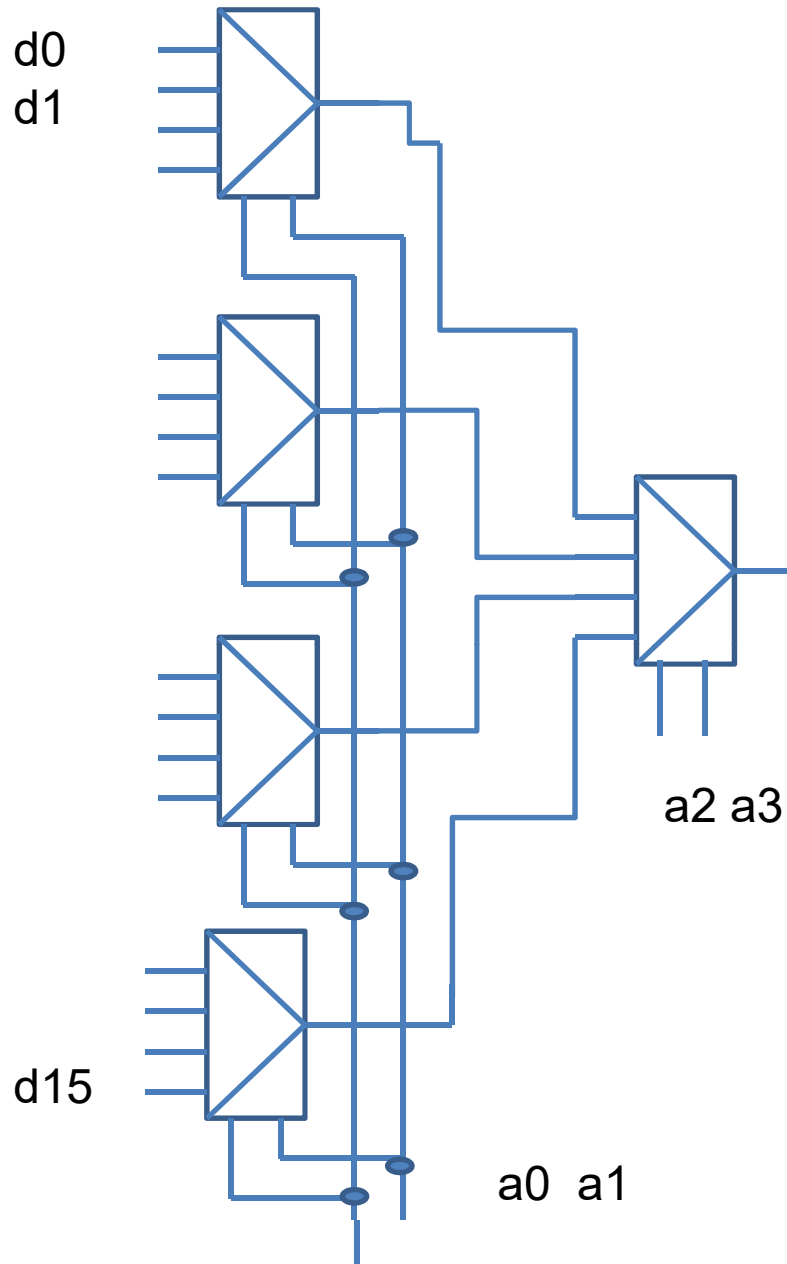
Synteza funkcji logicznych w oparciu o dekodery



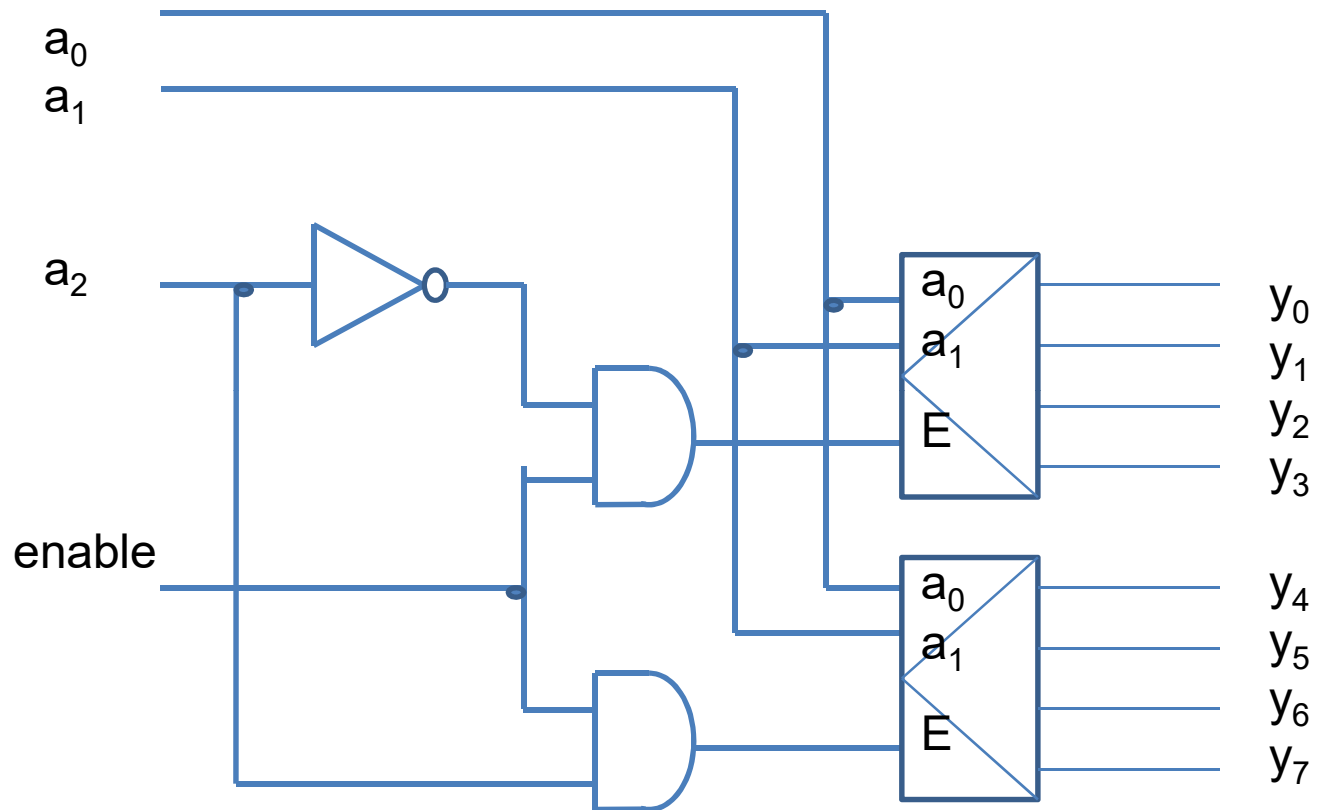
Dostępność wszystkich implikantów dla zmiennych adresowych pozwala na łatwą realizację wielu funkcji logicznych .
Uwaga: wyjścia aktywne poziomem niskim – bramka NAND dla negowanych czynników zapewni realizację sumy iloczynów

Łączenie multiplexerów

- Multiplexer 1 z 16
zbudowany jako dwu-
poziomowa struktura
5 multiplexerów 1 z 4

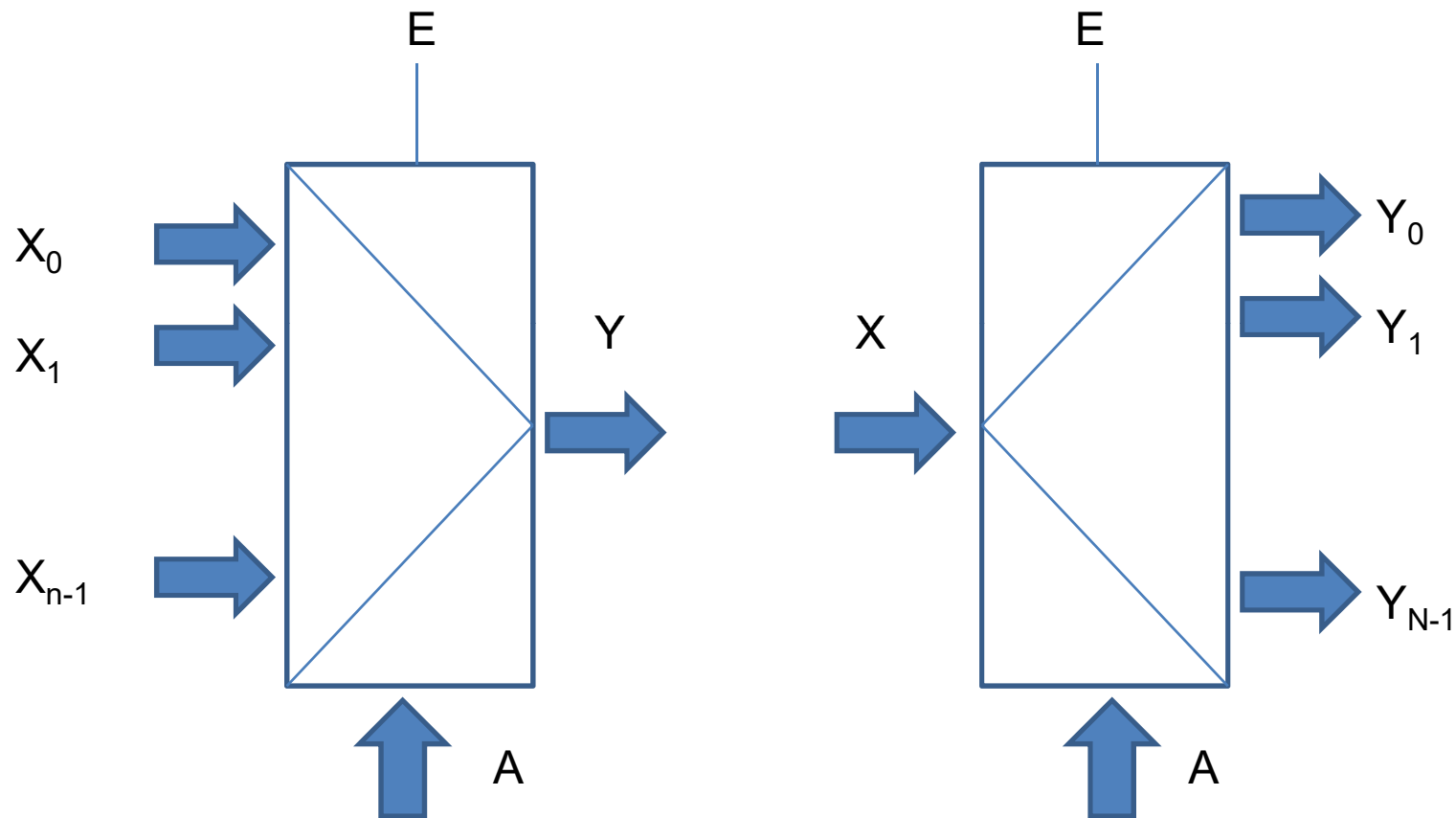


Łączenie dekoderek



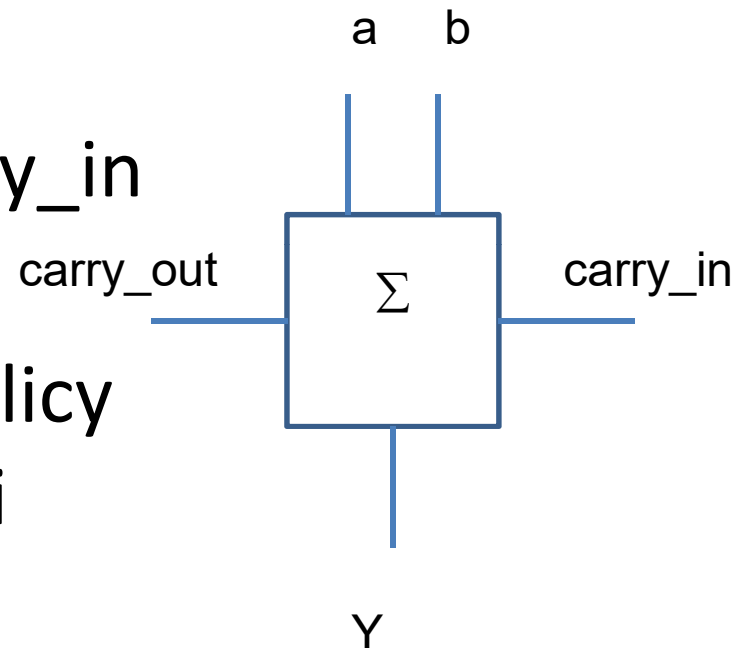
- Dekoder 3/8zbudowany z dwóch dekoderek 2/4,
- E wejście zezwalające na wybór wyjścia

Multiplexer i demultiplexer grupowy



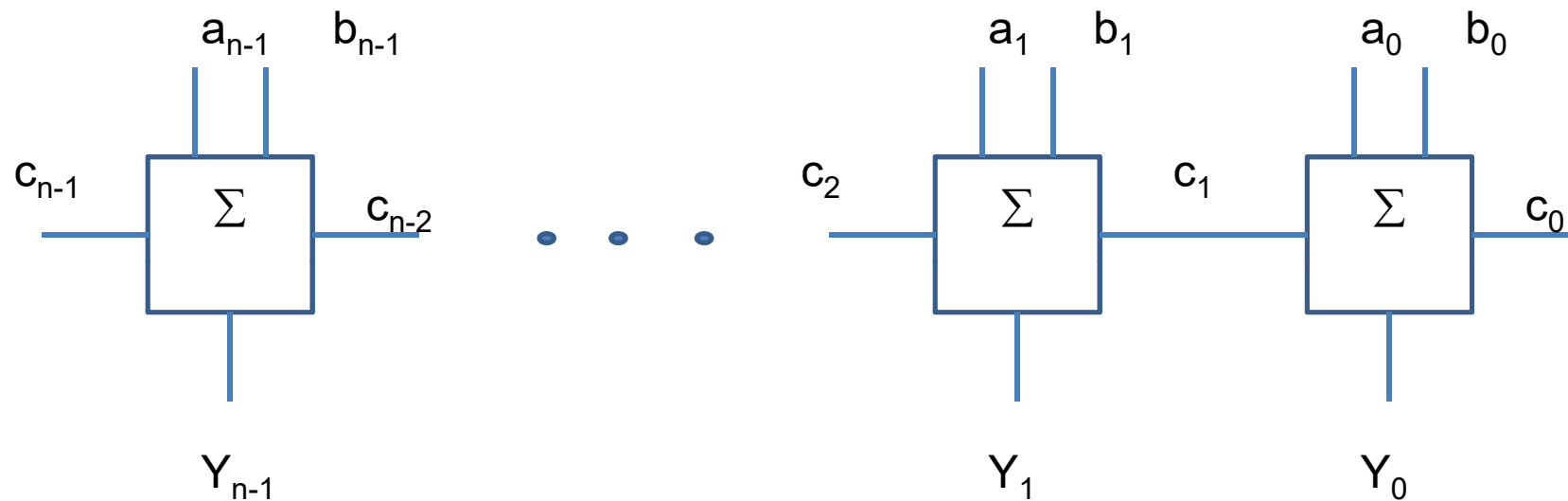
Sumator jednobitowy

- Sumator jednobitowy pełny
- $Y = a \oplus b \oplus \text{carry_in}$
- $\text{carry_out} = ab + (a+b)\text{carry_in}$
 $= ab + (a \oplus b) \text{carry_in}$
- Zależności wynikają z tablicy prawdy dla funkcji sumy i funkcji przeniesienia



Sumator

z przeniesieniem szeregowym



Czas działania układu o takiej strukturze jest sumą czasów działania układów składowych ze względu na sekwencyjną propagację przeniesienia - 2 poziomy bramkowania na jeden bit.

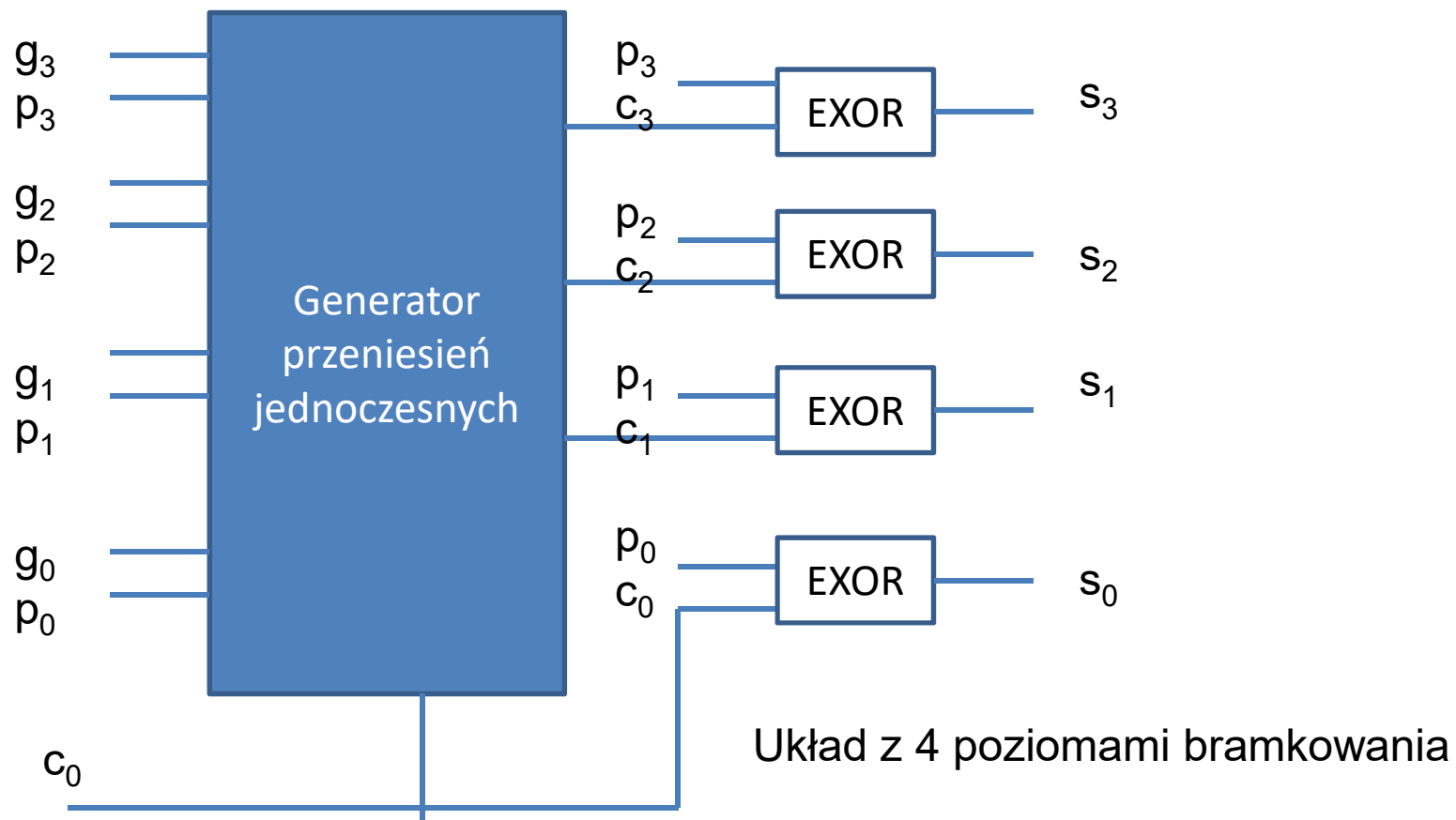
Sumator z przeniesieniami równoległymi

- G_i - Warunek generacji przeniesienia – gdy 2 sumowane bity są „1”
- P_i - Warunek propagacji przeniesienia – gdy jeden sumowany bit jest „1”
- Jeżeli $G_i = 1$ to $P_i = 0$
- $G_i = A_i B_i$ $P_i = A_i \oplus B_i$
- $c_{i+1} = G_i + P_i c_i$
- $Y_i = c_i \oplus P_i$
- $c_1 = g_1 + p_1 c_0$
- $c_2 = g_2 + p_2 c_1 = g_2 + p_2(g_1 + p_1 c_0) = g_2 + p_2 g_1 + p_2 p_1 c_0$
- $c_3 = g_3 + p_3 c_2 = g_3 + p_3(g_2 + p_2 g_1 + p_2 p_1 c_0) = g_3 + p_3 g_2 + p_3 p_2 g_1 + p_3 p_2 p_1 c_0$

Dla wyznaczenia przeniesienia c_i brak konieczności znajomości wyniku z pozycji wcześniejszej c_{i-1} – możliwość równoległej dla wszystkich pozycji sumatora generacji przeniesień jako sumy iloczynów sygnałów dostępnych na starcie przetwarzania.

Wykorzystanie powyższych równań pozwala na uzyskanie wyniku – sumy na 4 bitach po przejściu 4 poziomów bramkowania – wyznaczenie G, P 1 poziom, wyznaczenie C - 2 poziomy, wyznaczenie wyniku jeden poziom bramkowania.

Sumator z przeniesieniami równoległymi



Sumator z przeniesieniami równoległymi

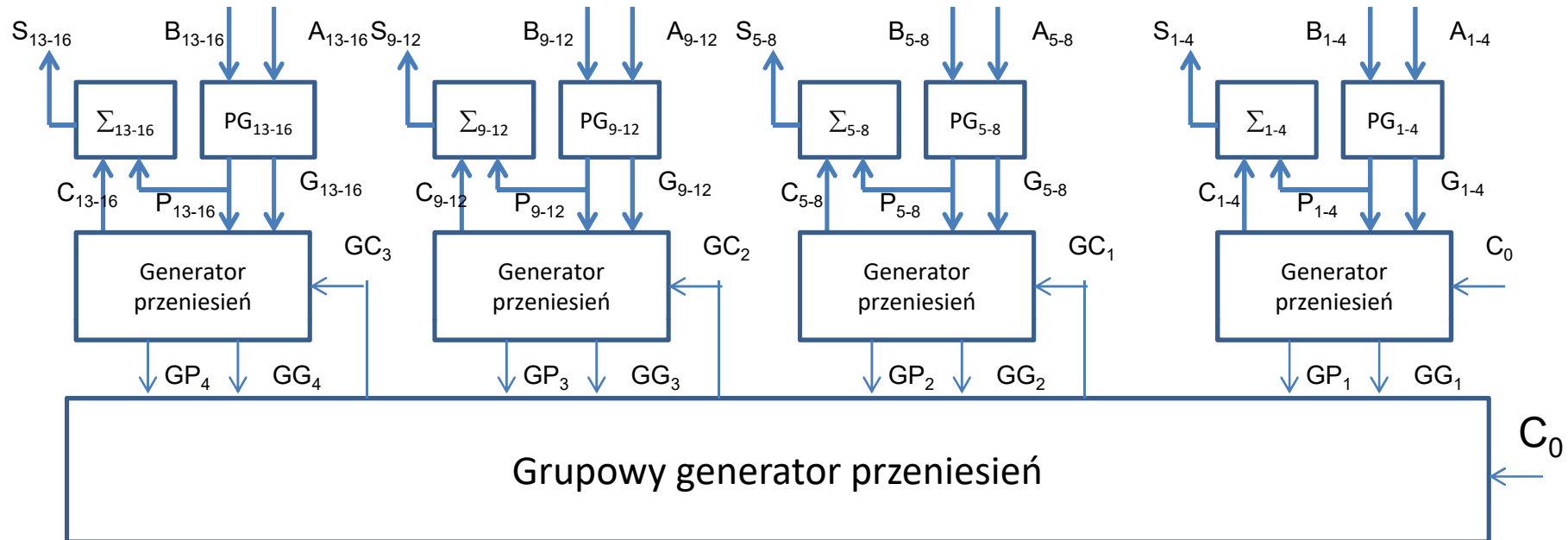
Problemem jest: duża liczba wejść bloku wyznaczającego przeniesienie wejściowe dla starszych bitów.

Rozwiązaniem problemu jest: przetwarzanie grup bitów i wykorzystanie do otrzymania przeniesienia z bieżącej grupy sygnału przeniesienia wyznaczonego dla młodszej grupy.

Rozważmy sumator 16 bitowy - 4 grupy po 4 bity:

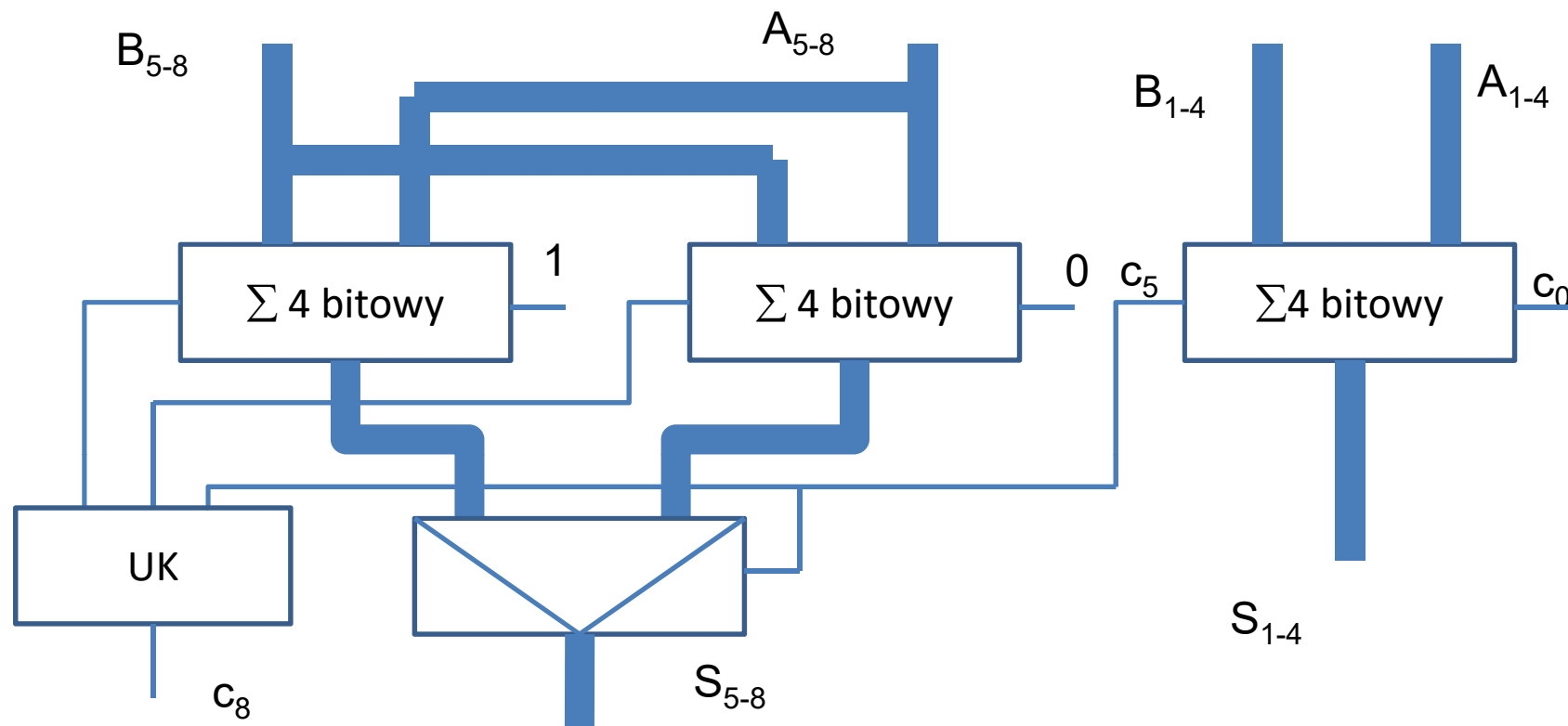
- Warunek generacji grupy 4 bitów $G = G_4 + G_3P_4 + G_2P_4P_3 + G_1P_4P_3P_2$
 - Warunek propagacji grupy $P = P_4P_3P_2P_1$
 - Przeniesienie z grupy $C_g = G + P C_{g-1}$ (grupowy generator przeniesienia)
 - Czasy: t_p – przejścia jednego poziomu bramkowania
1. Generacja sygnałów G i P (dla grup) jest realizowana równolegle dla każdej z grup - $3 t_p$
 2. Wyznaczenie sygnału przeniesienia grupy $+2 t_p$ (na grupę) (grupowy generator przeniesienia)
 3. Powtórzenie kroku 2 dla kolejnych 3 grup
- Czas propagacji: dla 16 bitów mamy zatem $3t_p + 4 \cdot 2t_p = 11 t_p$ zamiast 16 kroków (po 2 poziomy bram.) - $32 t_p$ przetwarzania sumatora z przeniesieniem szeregowym.

Grupowy sumator z przeniesieniami równoległymi



- Blok Σ sprowadza się do bramek EXOR, a generatory przeniesień pracują wg podanych wcześniej wzorów funkcji.

Sumator ze sterowanym przeniesieniem wybozem wyniku



UK w zależności od c_5 określa skąd pochodzi przeniesienie

Podział sumatorów

- Równoległe:
 - Z przeniesieniem szeregowym
 - Z przeniesieniem równoległym
- Szeregowo (układy sekwencyjne)
 - Zwykłe
 - Akumulujące
- Szeregowo-równoległe

Odejmowanie

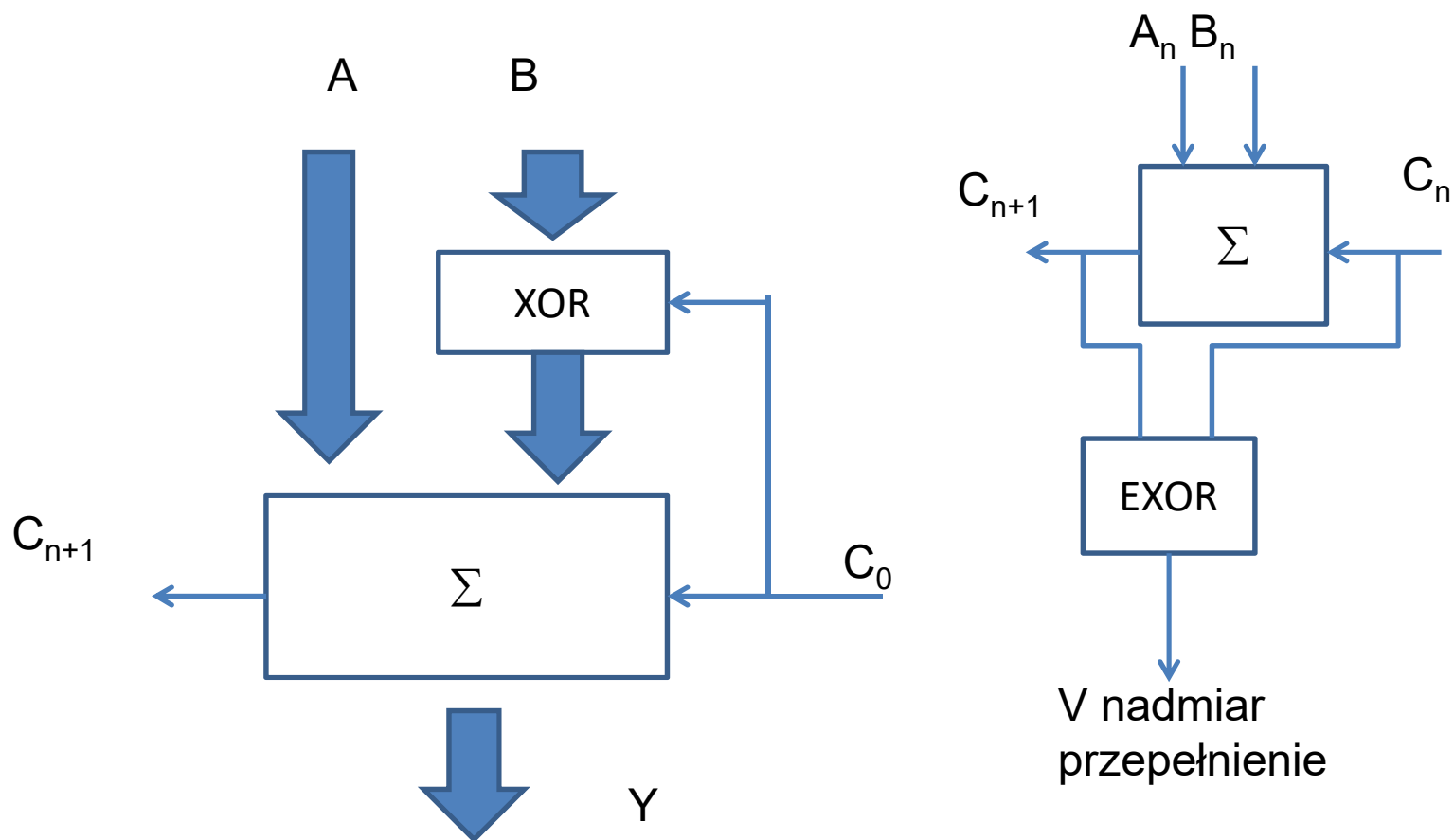
- Równanie poniższe przedstawia działanie układu sumatora lub substraktera w zależności od wartości zmiennej c . Działania realizowane na wartościach w kodzie U2.

- $Y = A + (B \oplus c) + c$

gdzie $\oplus$ oznacza sumę modulo wektora A i skalar c

- Dla $c=0$ realizowane jest dodawanie.
- Dla $c=1$ $Y = A + B' + 1$ gdzie $B' + 1$ oznacza liczbę „-B” w kodzie U2.
- Zasada działania: układ dodający, w przypadku odejmowania dodaje liczbę przeciwną do odjemnika.

Układ sumatora/substraktera



Nadmiar pojawia się gdy wejście i wyjście przeniesienia na najstarszym bicie są różne.

Algorytm mnożenia

				a_3	a_2	a_1	a_0
			X	b_3	b_2	b_1	b_0
				$a_3 b_0$	$a_2 b_0$	$a_1 b_0$	$a_0 b_0$
		+	$a_3 b_1$	$a_2 b_1$	$a_1 b_1$	$a_0 b_1$	
			$a_3 b_1$	s_{31}	s_{21}	s_{11}	
			c_{31}	c_{21}	c_{11}		
	+	$a_3 b_2$	$a_2 b_2$	$a_1 b_2$	$a_0 b_2$		
		$a_3 b_2$	s_{42}	s_{32}	s_{22}		
		c_{42}	c_{32}	c_{22}			
+	$a_3 b_3$	$a_2 b_3$	$a_1 b_3$	$a_0 b_3$			
	$a_3 b_3$	s_{53}	s_{43}	s_{33}			
+	c_{53}	c_{43}	c_{33}				
s_7	s_6	s_5	s_4	s_3	s_2	s_1	s_0

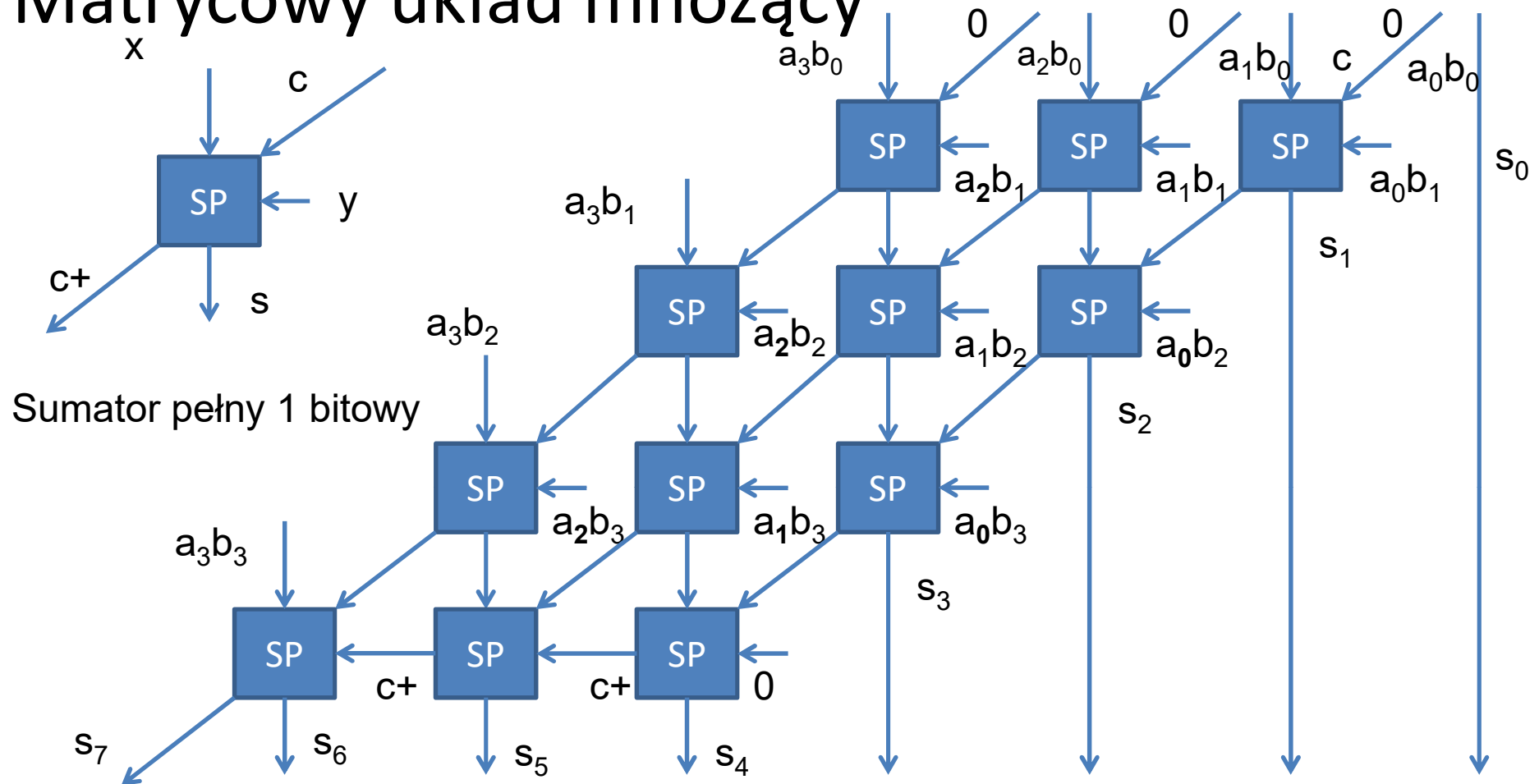
Algorytm mnożenia

				a_3	a_2	a_1	a_0
			X	b_3	b_2	b_1	b_0
				$a_3 b_0$	$a_2 b_0$	$a_1 b_0$	$a_0 b_0$
		+	$a_3 b_1$	$a_2 b_1$	$a_1 b_1$	$a_0 b_1$	
			$a_3 b_1$	s_{31}	s_{21}	s_{11}	
			c_{31}	c_{21}	c_{11}		
	+	$a_3 b_2$	$a_2 b_2$	$a_1 b_2$	$a_0 b_2$		
		$a_3 b_2$	s_{42}	s_{32}	s_{22}		
		c_{42}	c_{32}	c_{22}			
+	$a_3 b_3$	$a_2 b_3$	$a_1 b_3$	$a_0 b_3$			
	$a_3 b_3$	s_{53}	s_{43}	s_{33}			
+	c_{53}	c_{43}	c_{33}				
s_7	s_6	s_5	s_4	s_3	s_2	s_1	s_0

Iloczyn czynnika i najmłodszego bitu T_{PAND}
Iloczyn czynnika i 2 bitu
Wynik częściowy (bez przeniesienia) $+T_{PSUM}$
Przeniesienie
Iloczyn czynnika i 3 bitu
Wynik częściowy (bez przeniesienia) $+T_{PSUM}$
Przeniesienie
Iloczyn czynnika i 4 bitu
Wynik częściowy (bez przeniesienia) $+T_{PSUM}$
SUMAWANIE Przeniesienia Z PROPAGACJĄ
Wynik ostateczny $+3T_{PSUM}$

Dla liczby 4 bitowej 4 etapy sumowania, 3 etapy sumowania na 3 bitach bez propagacji przeniesienia, przeniesienie dodawane jako składnik sumy kolejnego etapu, 4 etap – sumowanie przeniesienia z jego propagacją. Czas wyznaczania iloczynu $T_{PAND} + 6 * T_{PSUM}$

Matrycowy układ mnożący

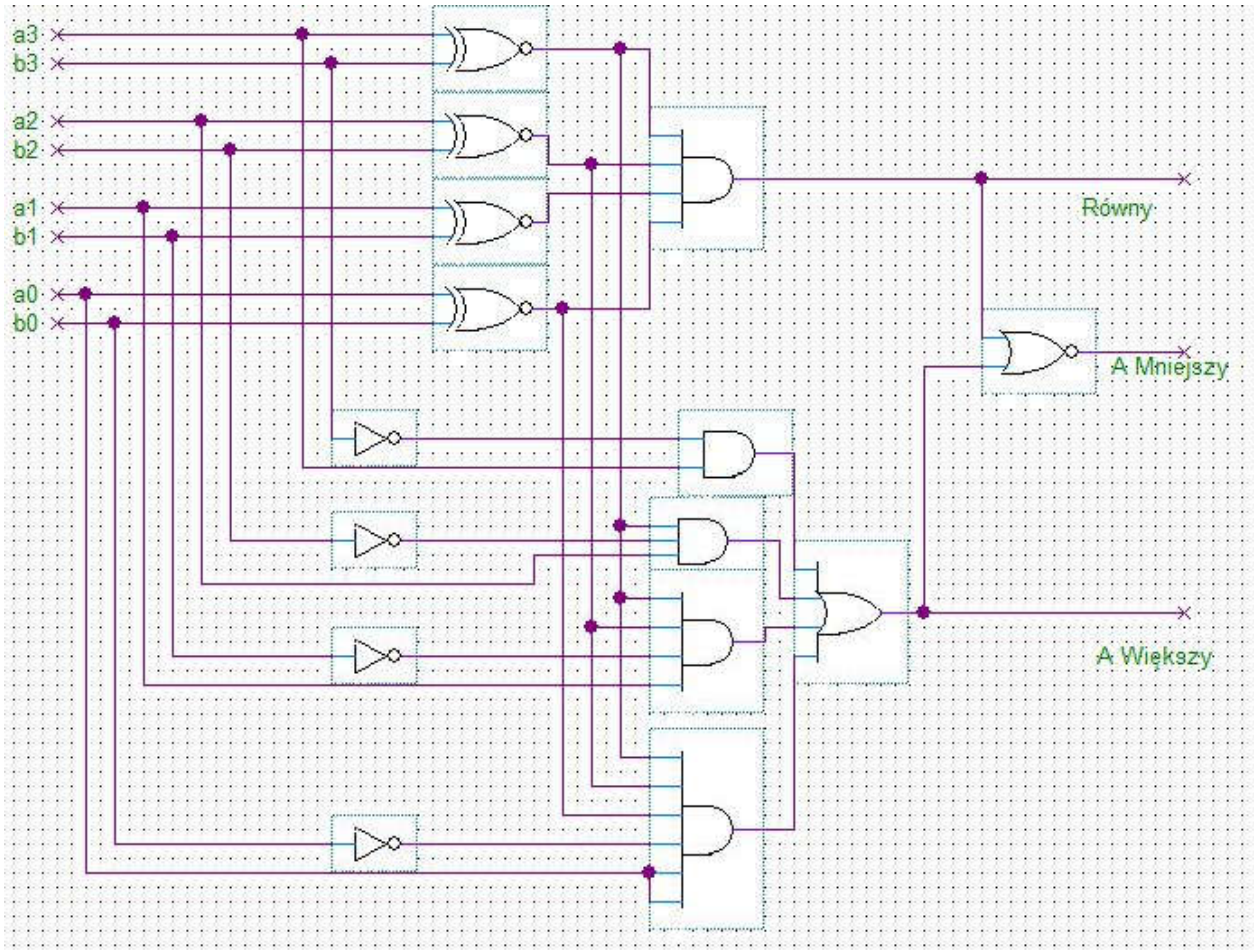


T_{p_sp} – propagacja sumatora pełnego

T_{p_b} – propagacja bramki

Czas propagacji układu jest równy $T_{p_b} + 6 * T_{p_sp}$

Komparator



Porównywanie
liczb binarych
dodatnich
 $a_3a_2a_1a_0$ i
 $b_3b_2b_1b_0$

Bramka exnor –
równoważności

Znajdź błąd na
schemacie