Podstawy techniki cyfrowej zima 2020

Wykład 3

dr inż. Rafał Walkowiak

20.10.2020

Metoda Quine’a-McCluskeya

1. Utwórz grupy kombinacji zmiennych (z wartością funkcji =1) posiadające jednakowa liczbę 1 w ich reprezentacji binarnej. Jest to tworzenie grup początkowych implikantów.
2. Utwórz wszystkie implikanty przez połączenie implikantów jednej grupy z implikantami kolejnej grupy – jest to możliwe jeżeli reprezentacje binarne kombinacji zmiennych różnią się wartością jednej zmiennej, zaznacz wykorzystane do łączenia implikanty (nie będą już dalej wykorzystywane).
3. Powtarzaj krok 2 bazując na implikantach uzyskanych w poprzedniej iteracji 2 kroku.
4. Niewykorzystane w połączeniach implikanty tworzą zbiór implikantów prostych. Wybierz minimalny zbiór implikantów prostych za pomocą tablicy pokrycia lub funkcji Petrica.

Metoda Quine'a-McCluskeya

generacja implikantów prostych

wygodna dla funkcji wielu zmiennych

Funkcja f(a,b,c,d)= $\cup$(0,1,2,3,8,9,10)+d(5,13)

1	2	3
0 0000 √	0,1 000- √	0,1,2,3 00—
	0,2 00-0 √	0,2,8,10 -0-0
1 0001 √	0,8 -000 √	0,1,8,9 -00-
2 0010 √		
8 1000 √	1,3 00-1 √	1,5,9,13 –01
	1,5 0-01 √	
3 0011 √	1,9 -001 √	Implikanty proste
5 0101 √	2,3 001- √	00– d'c'
9 1001 √	2,10 -010 √	-0-0 c'a'
10 1010 √	8,9 100- √	-00- c'b'
	8,10 10-0 √	--01 b'a
13 1101 √		
	5,13 -101 √	
	9,13 1-01 √	

Metoda Quine'a-McCluskeya
tablica pokrycia mintermów

	0	1	2	3	8	9	10
0,1,2,3 00—	√	√	√	√			
0,2,8,10 -0-0	√		√		√		√
0,1,8,9 -00-	√	√			√	√	
1,5,9,13 –01		√				√	

W kolumnach tablicy uwzględniamy tylko mintermy z określonymi dla funkcji wartościami

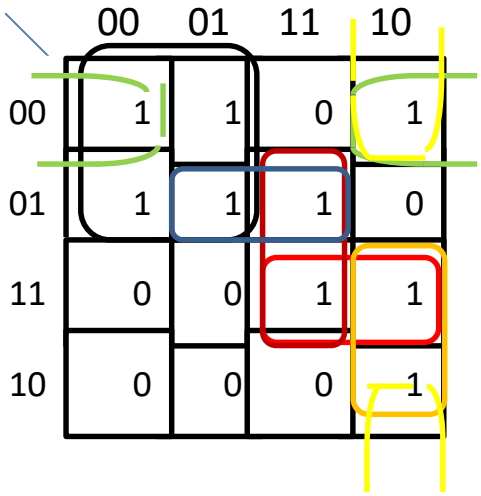
Pomarańczowo - Implikanty istotne

Zielono - Mintermy pokryte przez implikanty istotne

Możliwe warianty funkcji o minimalnej liczbie implikantów:

F=d'c'+c'a' + c'b'

F=d'c'+c'a' + b'a

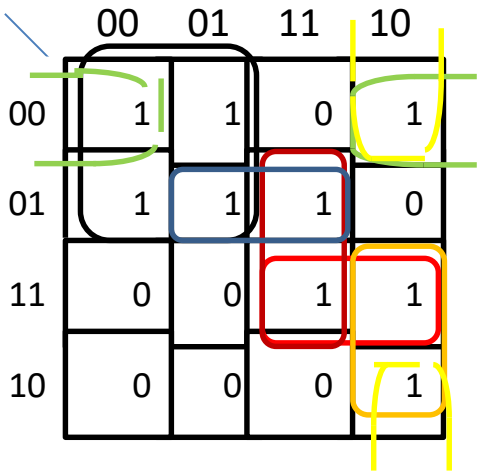


Metoda Petricka – przykład 2

Pozwala na wyznaczenie minimalnego zbioru implikantów prostych (nie dotyczy implikantów istotnych). Metoda jest stosowany jeśli tablica pokrycia nie wystarcza do łatwego określenia implikantów pokrywających wszystkie mintermy.

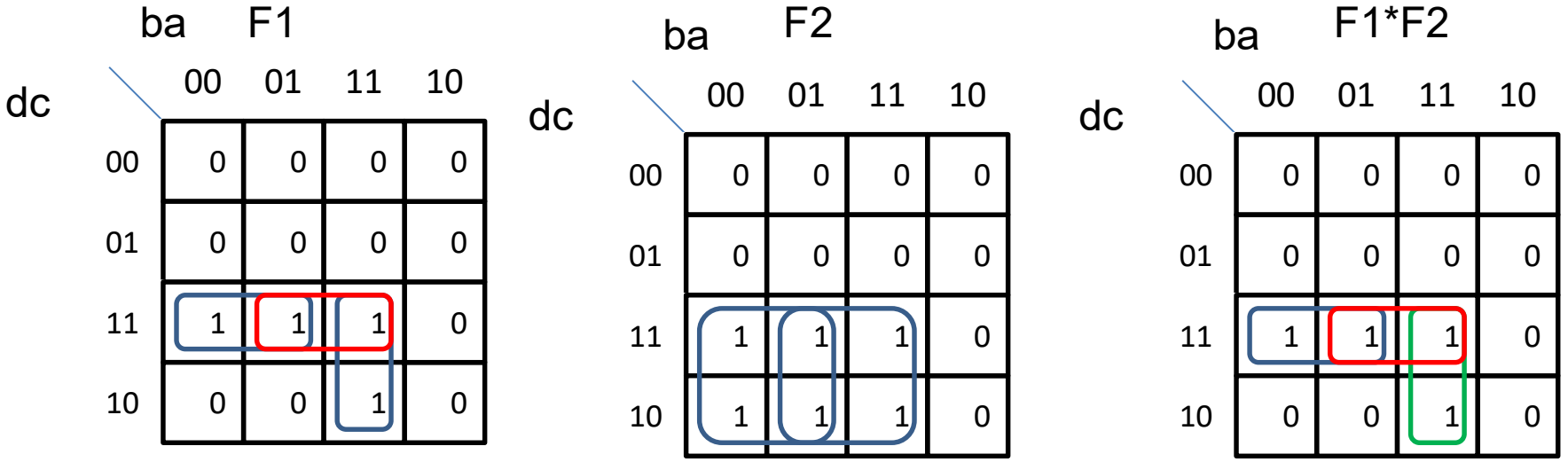
Przykład, wynik sklejania mintermów:

- jeden implikant istotny, poczwórny
- 6 implikantów prostych można wykorzystać do pokrycia 5 mintermów.
- Wyznaczamy funkcję Petrica. Pokrycie pełnego zbioru mintermów wystąpi, gdy zastosujemy implikanty, dla których funkcja Petricka przyjmuje wartość jeden.



- Px w równaniu na FP oznacza wykorzystanie implikanta x.
- Kolejne czynniki równania zapewniają pokrycie jednego mintermu. – FP=(P0 +P1)(P1 + P2)(P2 + P3)(P3 + P4)(P4 + P5) = P1P3P5+ P1P2P4+P0P2P4
- Powyższy zapis funkcji FP czytamy: pokrycie wystąpi gdy użyjemy implikant (0 lub 1) i (1 lub 2) i (2 lub 3) i (3 lub 4) …
- Po przekształceniach uzyskujemy alternatywne zbiory implikantów zapewniające pokrycie.
- Zbiór zapewniający pokrycie stanowią przykładowo oznaczone przez 1, 2 i 3 implikanty: niebieski (1), czerwony(3) i żółty (5). Do pełnego pokrycia wszystkich mintermów dołączmy implikant oznaczony na czarno.

Minimalizacja funkcji wielowyjściowych



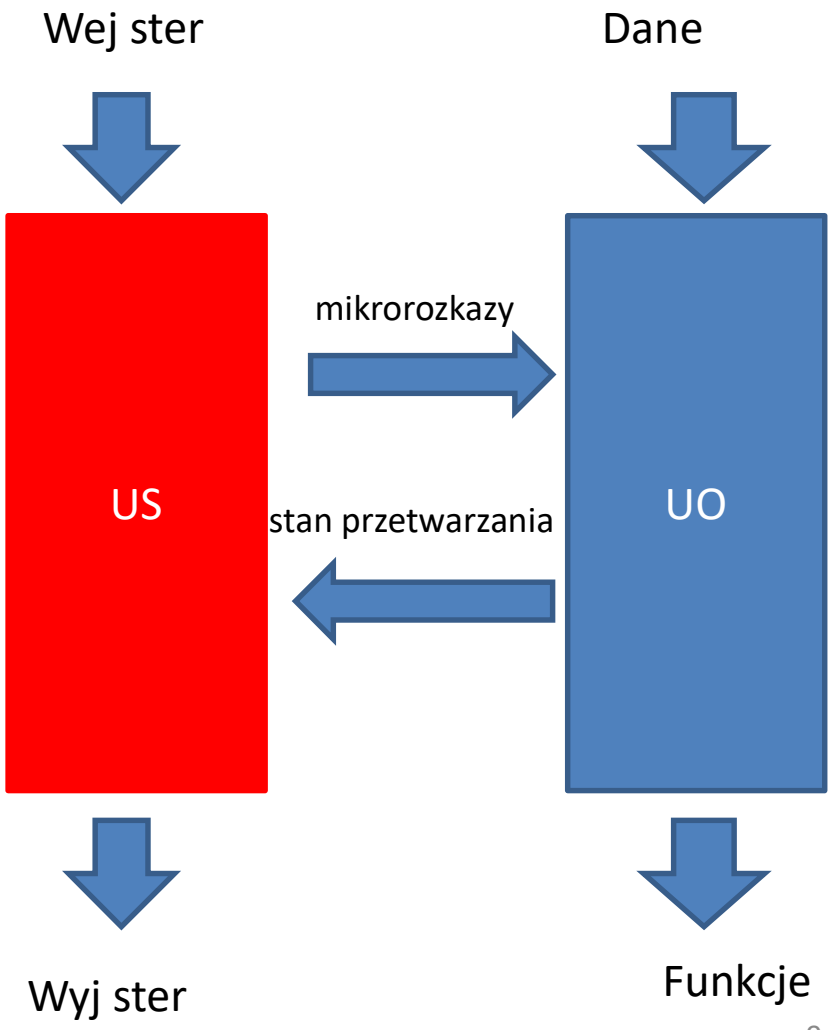
- Wyznaczenie implikantów prostych dla: funkcji optymalizowanych i wszystkich iloczynów funkcji - (powyżej 6 implikantów prostych w 3 grupach).
- Znajdowanie pokrycia minimalną liczbą spośród wszystkich implikantów (tablica pokrycia): implikant iloczynu dwóch funkcji (zielony) pokrywa mintermy obu funkcji

Komputerowo wspomaganie minimalizacji funkcji logicznych

- Znalezienie pokrycia minimalnego jest problemem NP-trudnym.
- Ze względu na trudność problemu dla dużych instancji stosowane są metody przybliżone.
  - brak generacji wszystkich implikantów
  - zapewnienie pokrycia funkcji przez wybrany zbiór implikantów

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

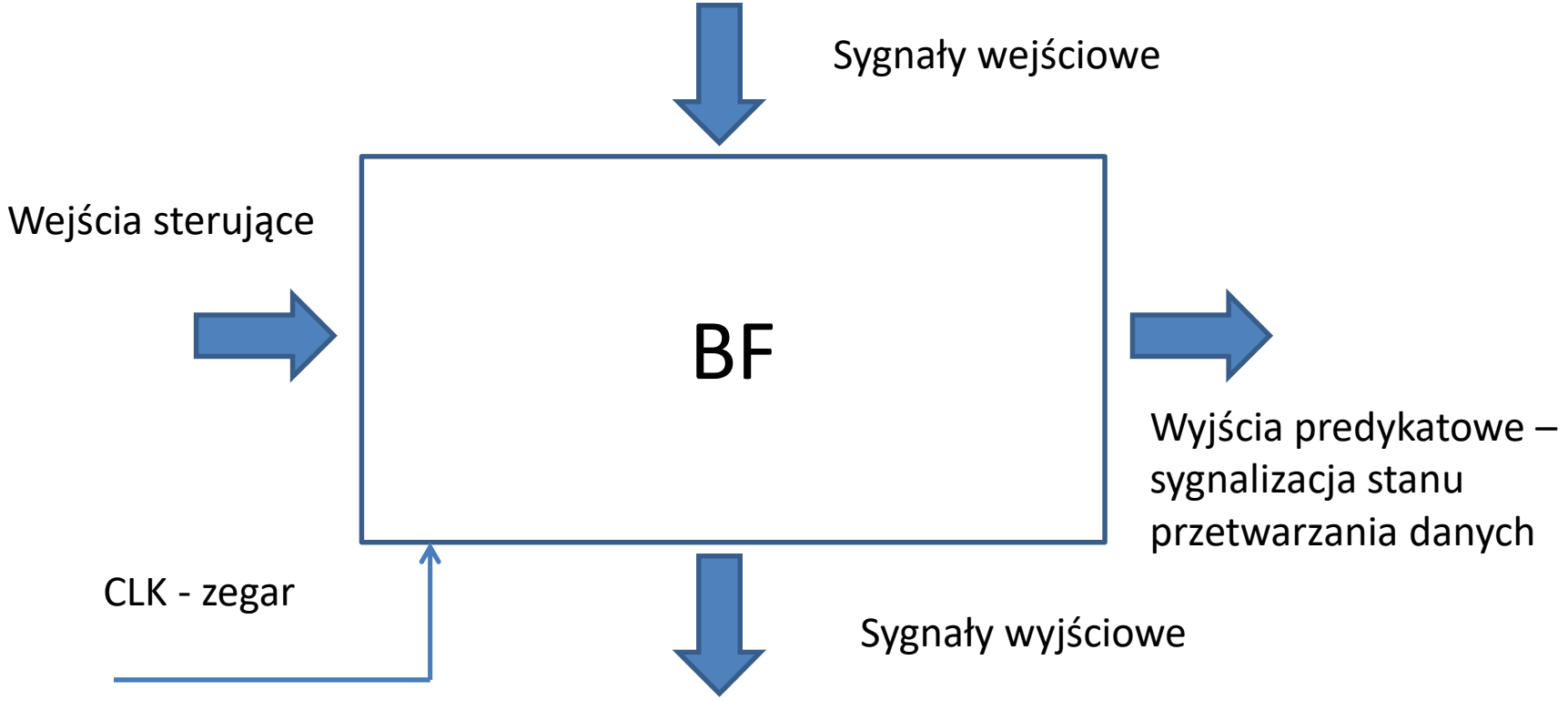
1. Określenie:
   - struktury bloków funkcjonalnych: liczniki, rejestry, komparatory, sumatory … i
   - połączeń między blokami funkcjonalnymi
2. Określenie sposobu pracy układu sterowania realizującego algorytm przetwarzania zapisanego za pomocą sieci działań – diagram ASM (ang. Algorithmic State Machine)

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 funktory układów cyfrowych
układy kombinacyjne

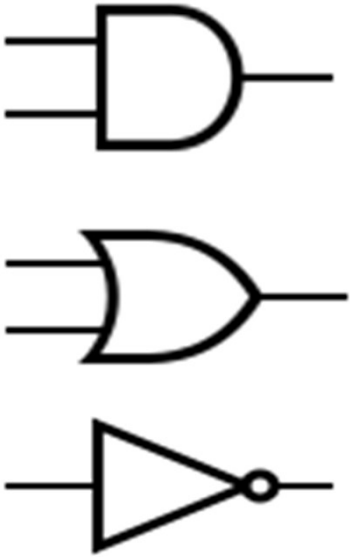
Por. Układy cyfrowe Wilkinson 2.4-6,3.5

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

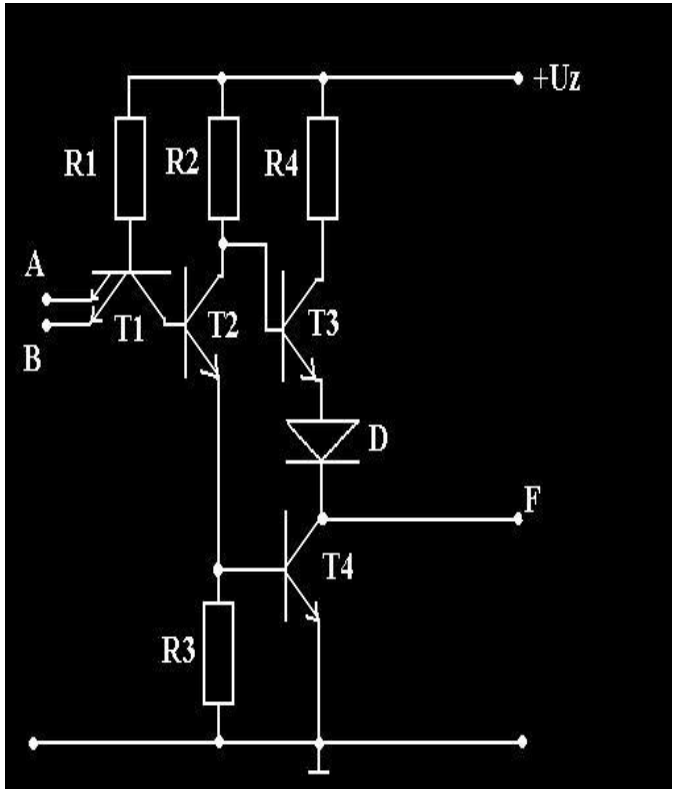
XNOR

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 ÷ 0, 8V sygnał TTL jest niski - logiczne „0” – logika dodatnia.
- Poziom sygnału jest między 2V ÷ 5V jest określany jako stan wysoki - logiczna „1”.
- Gdy wartość napięcia jest z przedziału 0, 8V ÷ 2V - sygnał jest nieokreślony.

Bramki TTL



Bramka NAND

Na wejściach poziom wysoki napięcia powoduje nasycenie T2 i T4 oraz zatkanie T3 (baza i emiter na tym samym potencjale), na wyjściu stan niski napięcia.

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

Dowolna liczba wejść z niskim napięciem daje wysokie napięcie na wyjściu.

Układy CMOS

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

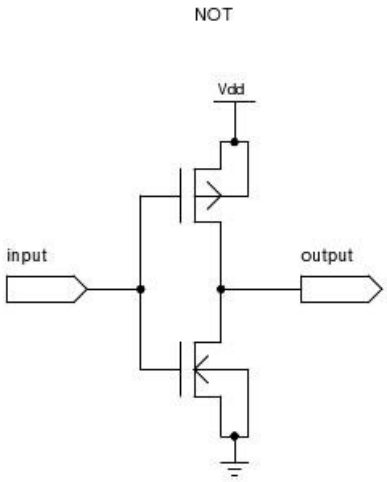
Bramki zbudowana są z par tranzystorów o przeciwnym typie przewodnictwa i połączonych w taki sposób, że w ustalonym stanie logicznym przewodzi tylko jeden z nich.

- Układy CMOS są relatywnie proste i tanie w produkcji, umożliwiając uzyskanie bardzo dużych gęstości upakowania.
- Układy cyfrowe wykonane w technologii CMOS mogą być zasilanie 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



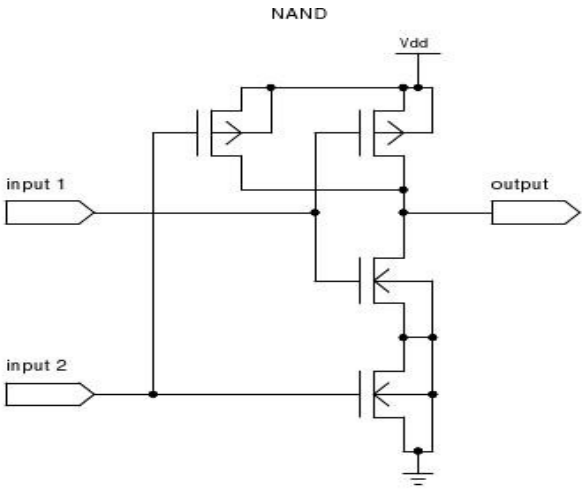
- CMOS układy z komplementarnymi (typu p i typu n) tranzystorami MOS
- Strzałka określa wymagany kierunek wzrostu napięcia pomiędzy bramką a podłożem otwierającego klucz (w kierunku źródło –dren) wykonany w oparciu o tranzystor

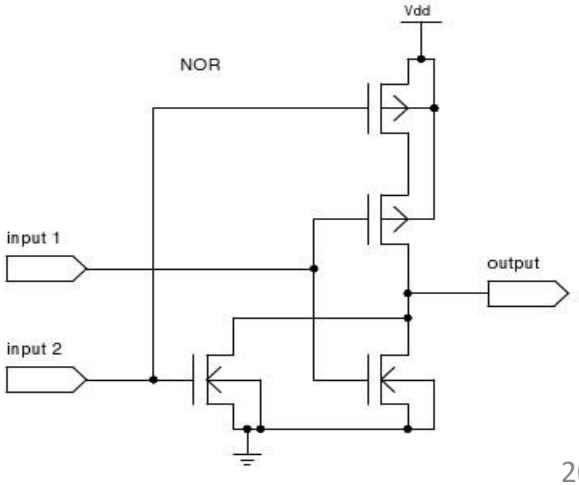
Bramka NOT

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

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

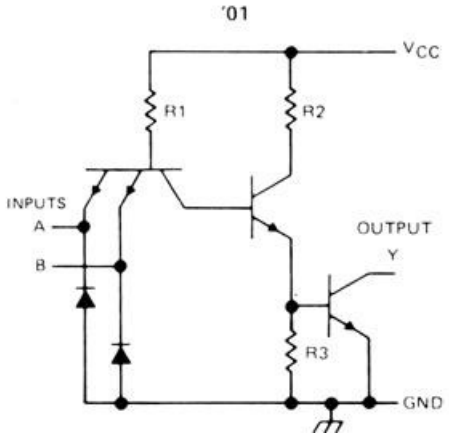
Bramka NOR – jedna 1 na wejściu wystarczy aby wyjście wysterować do poziomu wysokiego

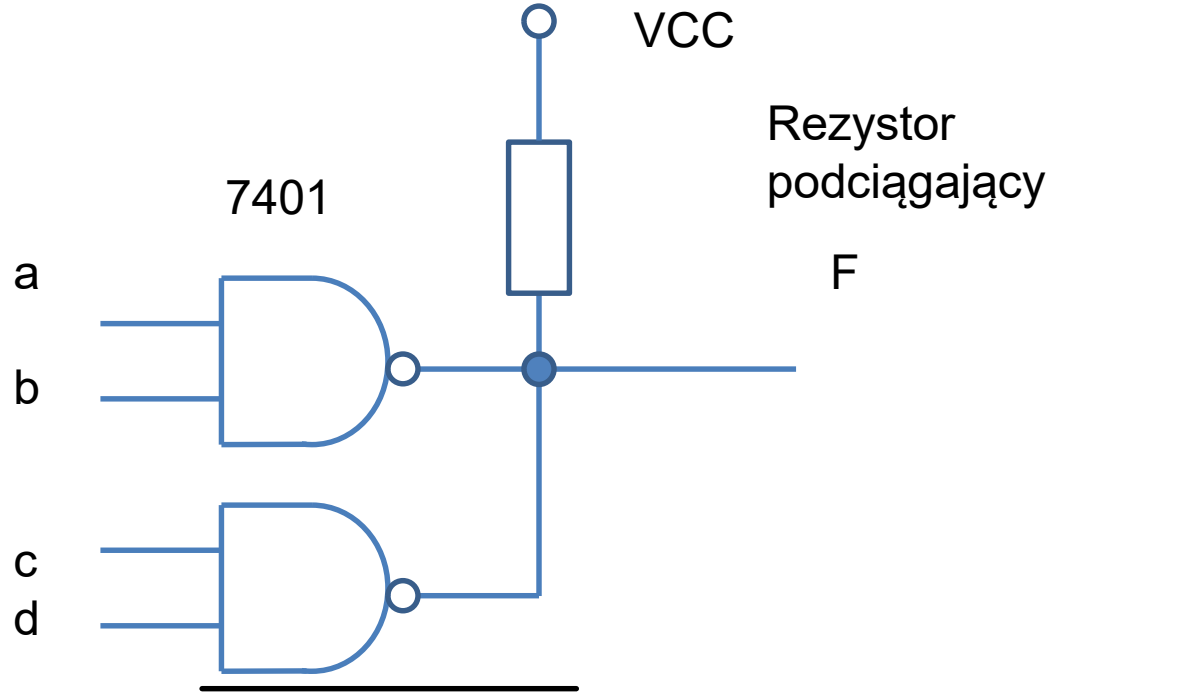




Bramki typu „open collector”

„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

$$F = \overline{ab + cd} = (ab)'(cd)'$$

W obwodzie powyżej połączenie wyjść typu open colector (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:

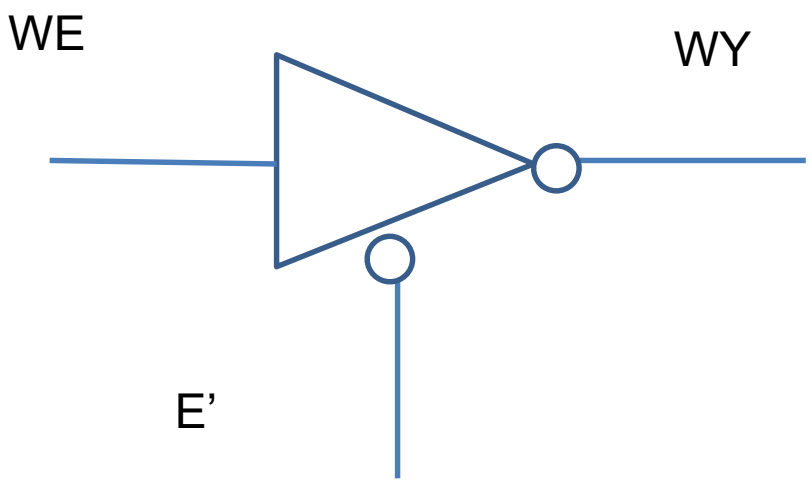
- wysokiej impedancji – być nieaktywne lub
- zwarte z masą – być aktywne ,

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

Zastosowanie:

- Zastosowanie odpowiedniego poziomu napięcia zasilnia 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

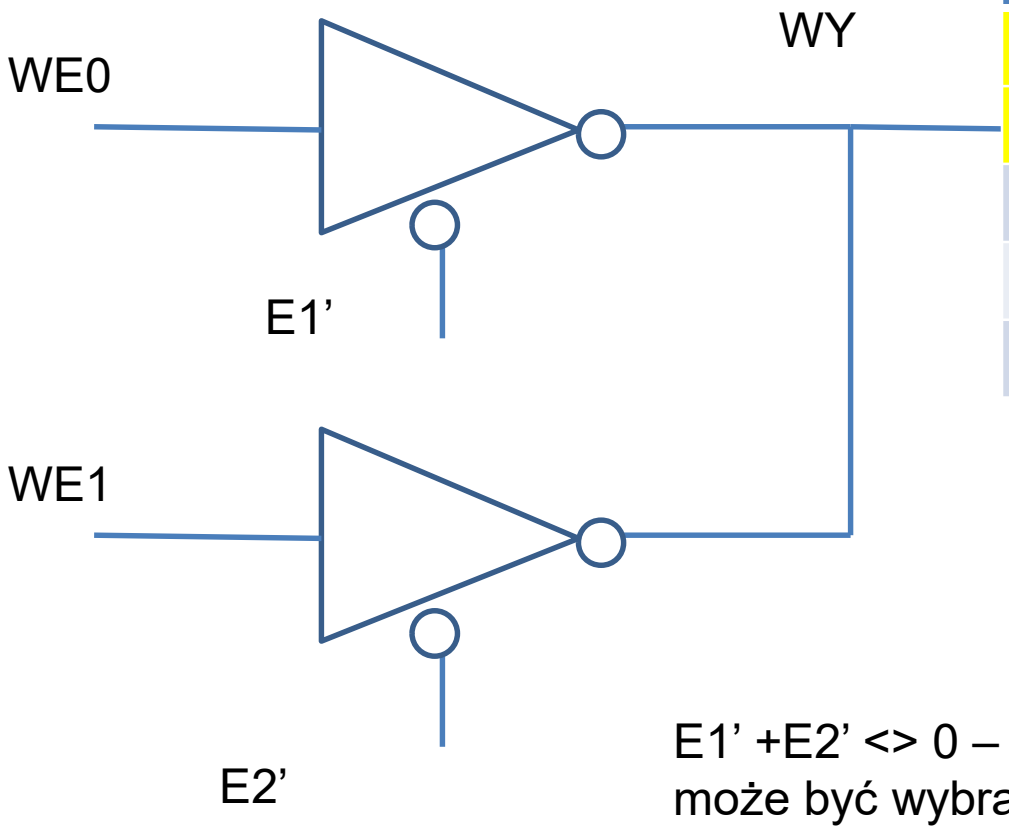
stan wysokiej impedancji (ang. high impedance) Z.

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

Zastosowanie:

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

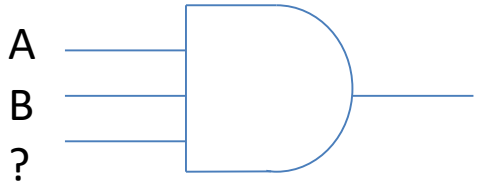
Bramki trójstanowe zastosowanie



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’ <> 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

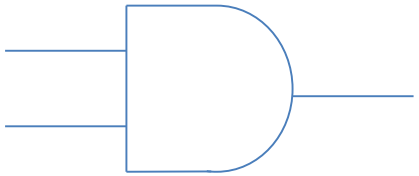
Możliwe zatem rozwiązania to podłączenie:

- 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 połaczeń 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 a funkcje bramek

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 Napięć 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

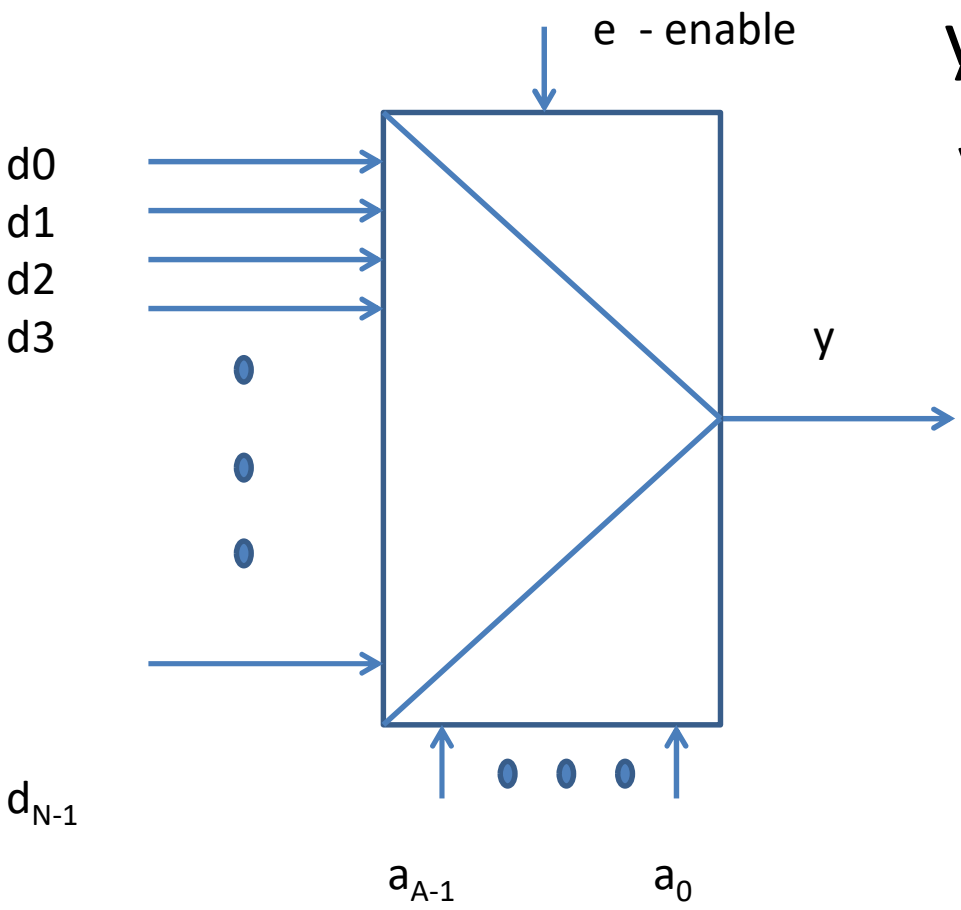
Multipleksery

- 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 A zależnością $N=2^A$
- FUNKCJA REALIZOWANA PRZEZ MULTIPLEKSER:

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

- Gdzie $P_k(Adr)$ oznacza pełny iloczyn zmiennych adresowych $a_{n-1}, \ldots, a_0$ prostych lub zanegowanych = 1 dla k-tej kombinacji zmiennych – dla A=2 i k=1 $P_1(Adr) = a_1'a_0$
- D_k oznacza sygnał wejściowy k-tego wejścia informacyjnego

Multiplekser



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

$$y=e\,(a_1'a_0'\,d_0 +$$

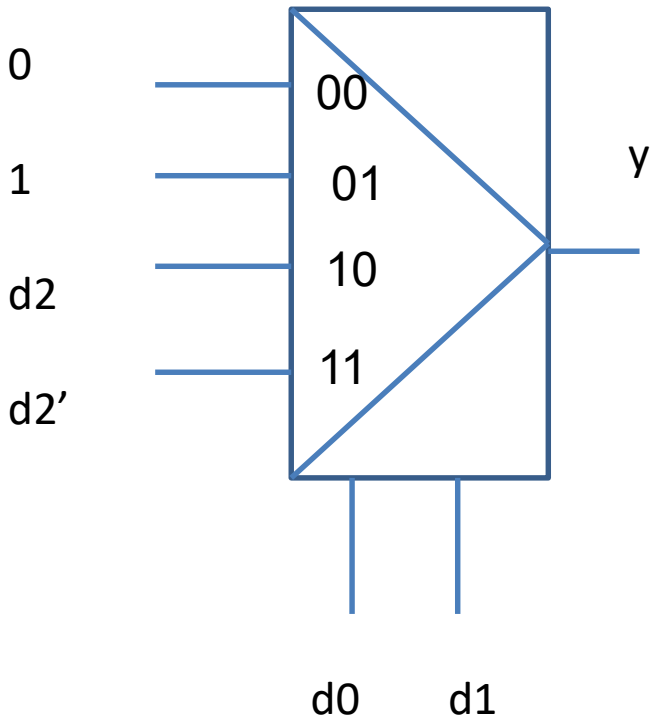
$$a_1'a_0\,d_1 +$$

$$a_1a_0'\,d_2 +$$

$$a_1a_0\;\;d_3)$$

Multiplekser w syntezie funkcji logicznych

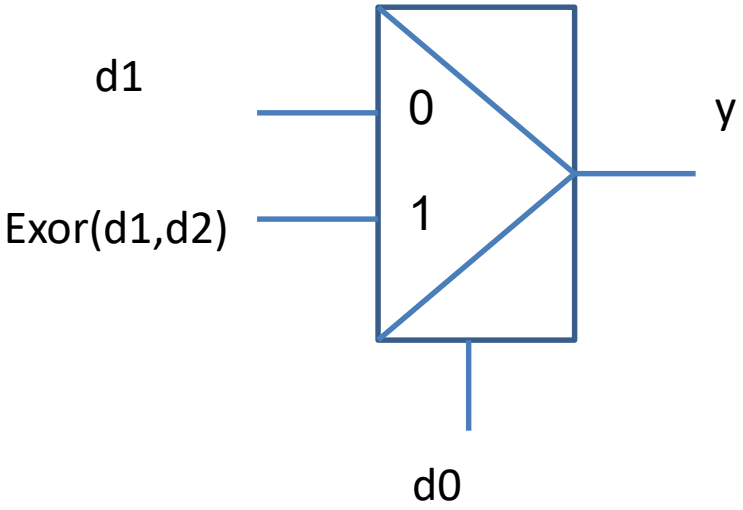
d0	d1	d2	y
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0



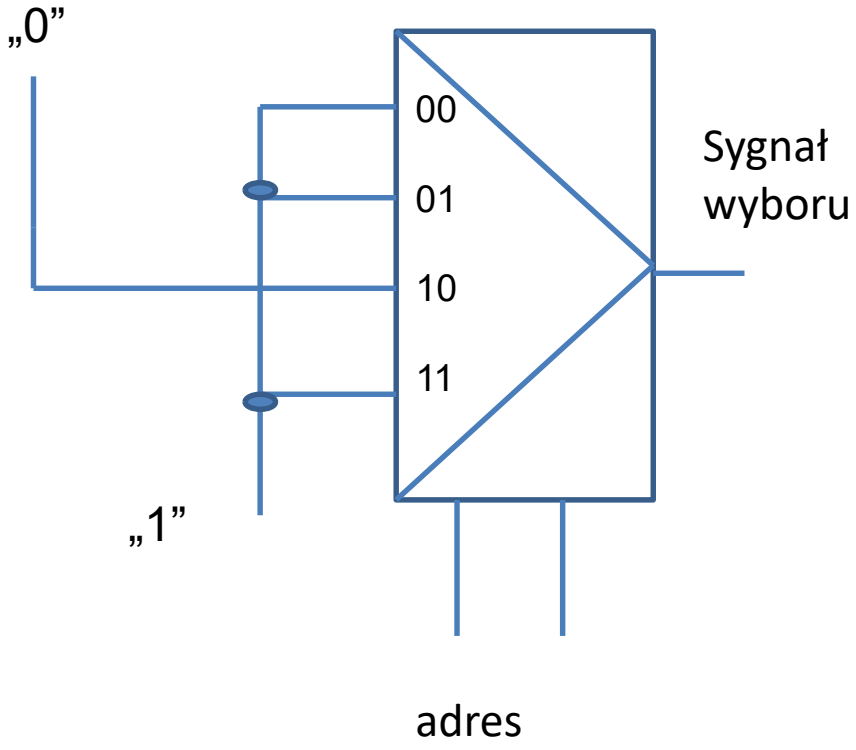
Multiplekser z A wejściami adresowymi,
Realizacja funkcji 3 wejść za pomocą multipleksera 2 wejściowego, podanie na wejście: jednego sygnału: 0, 1, trzeciego wejścia prostego lub zanegowanego.

Multiplekser 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



Multiplekser jako dekoder adresu



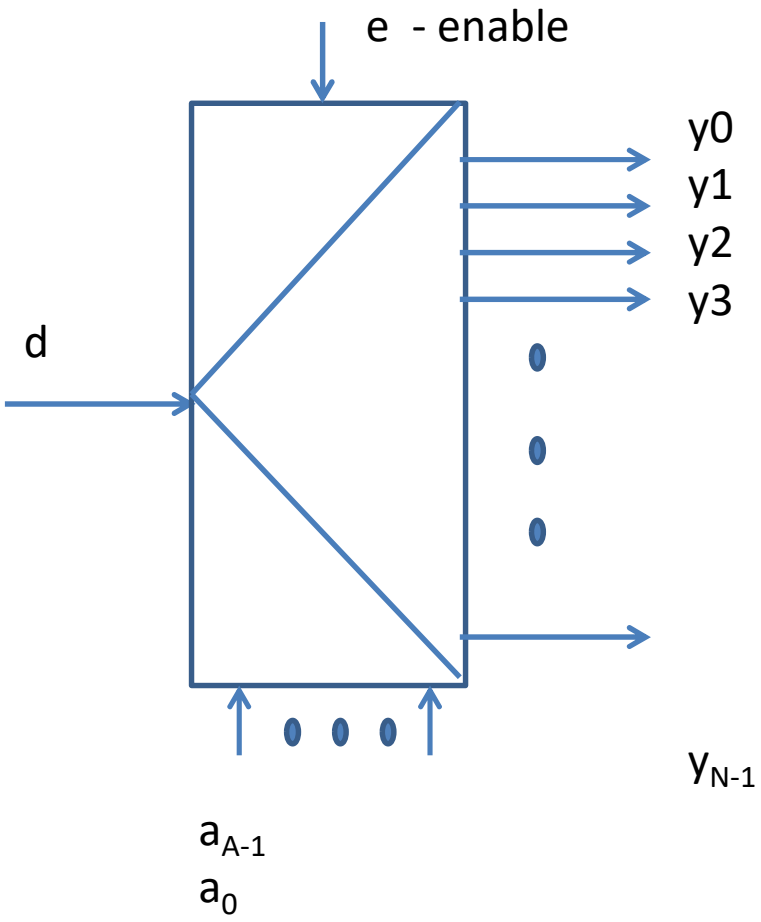
Określenie adresu urządzenia poprzez podłączenie poziomu aktywnego na wejście multipleksera odpowiadające adresowi urządzenia, pozostałe wejścia nieaktywne (1).

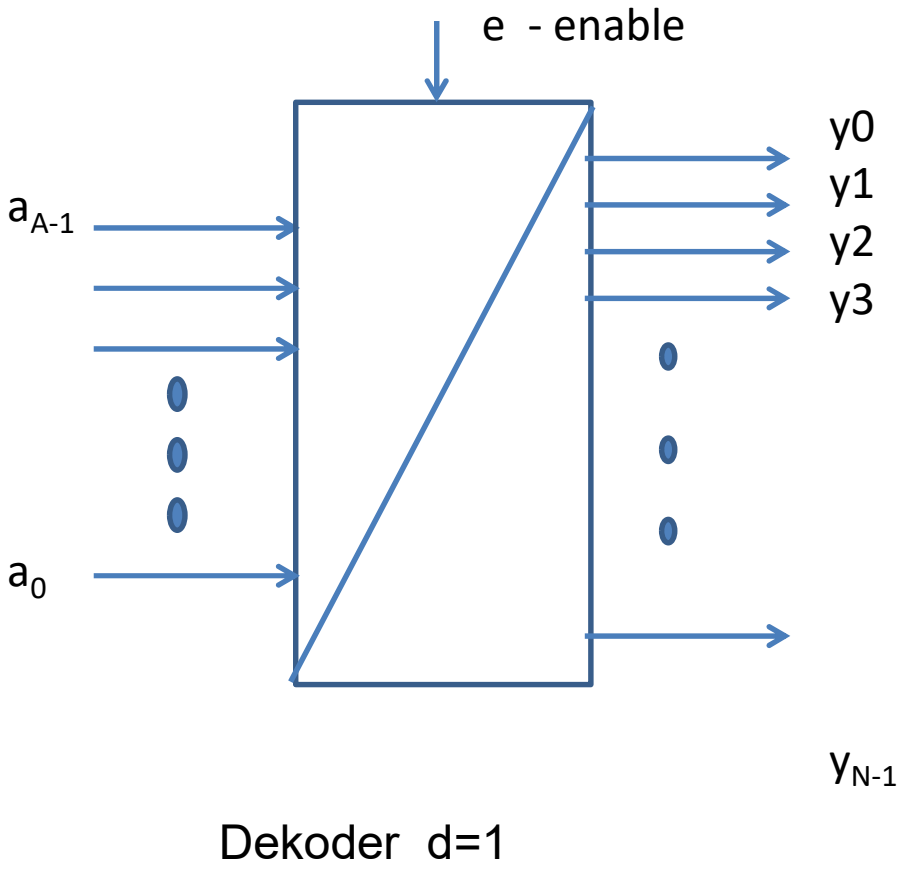
Jeśli aktywny poziom wyboru układu jest niski to dla adresu urządzenie równego 2 należy na wejścia podać sygnały jak na rysunku. Adres =2 powoduje pojawienie się na wyjściu 0.

Demultipleksery

- Pozwalają na wybór wyjścia i przesłanie 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”

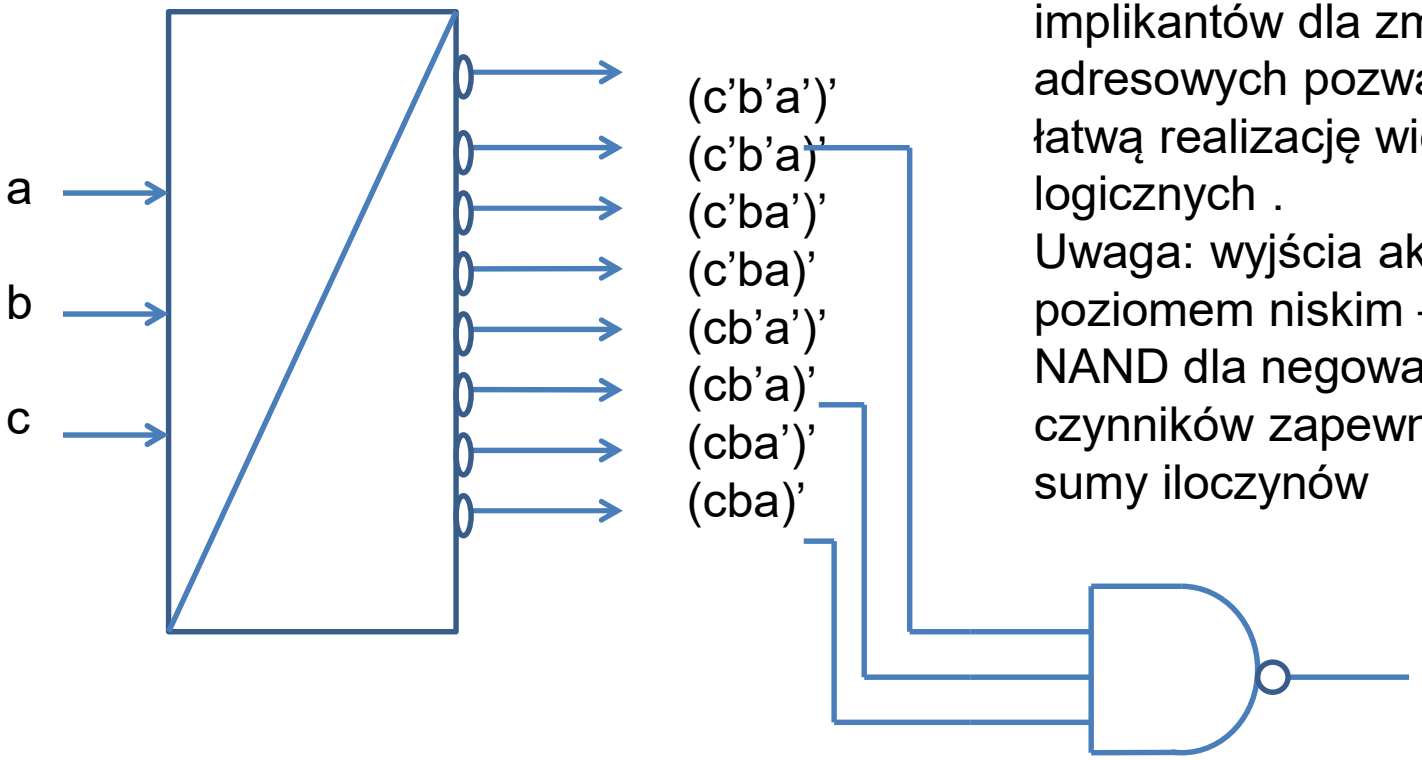
Demultiplekser , dekoder



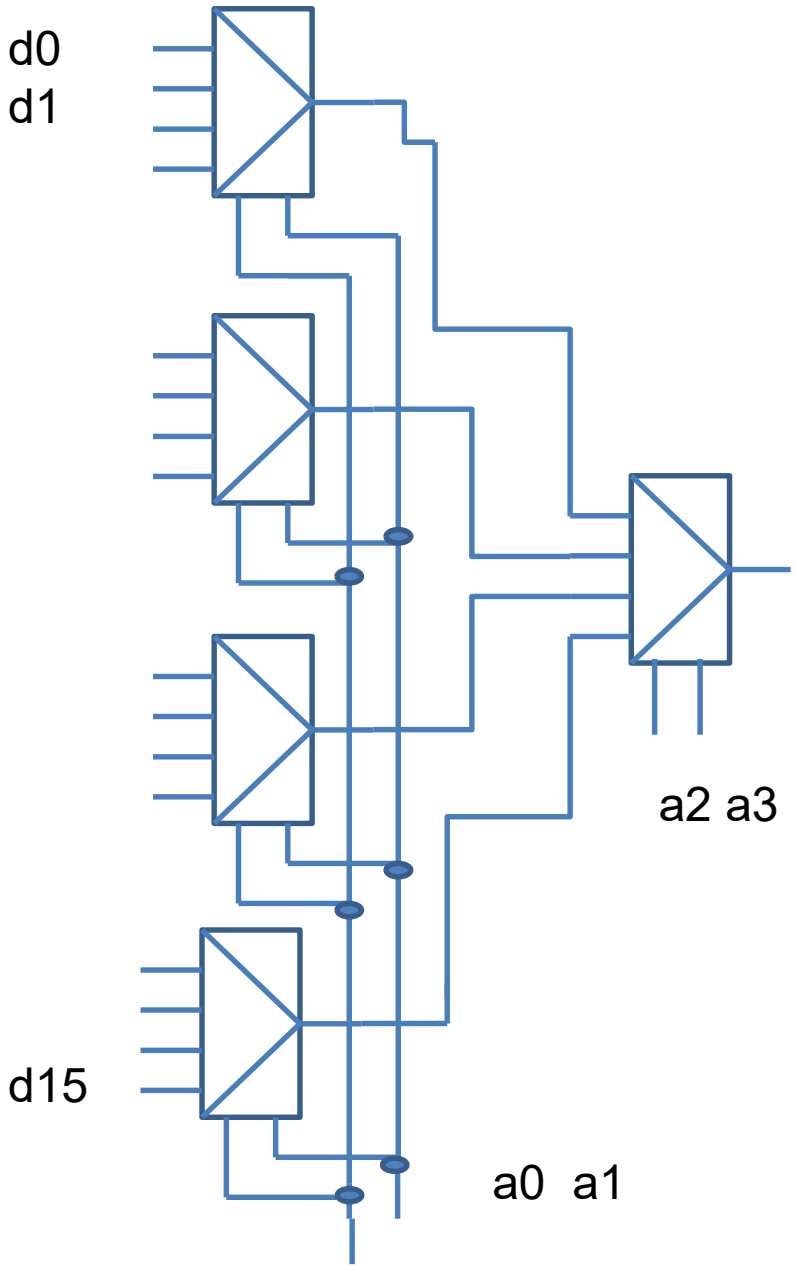


Dekoder d=1

Synteza funkcji logicznych w oparciu o dekoder



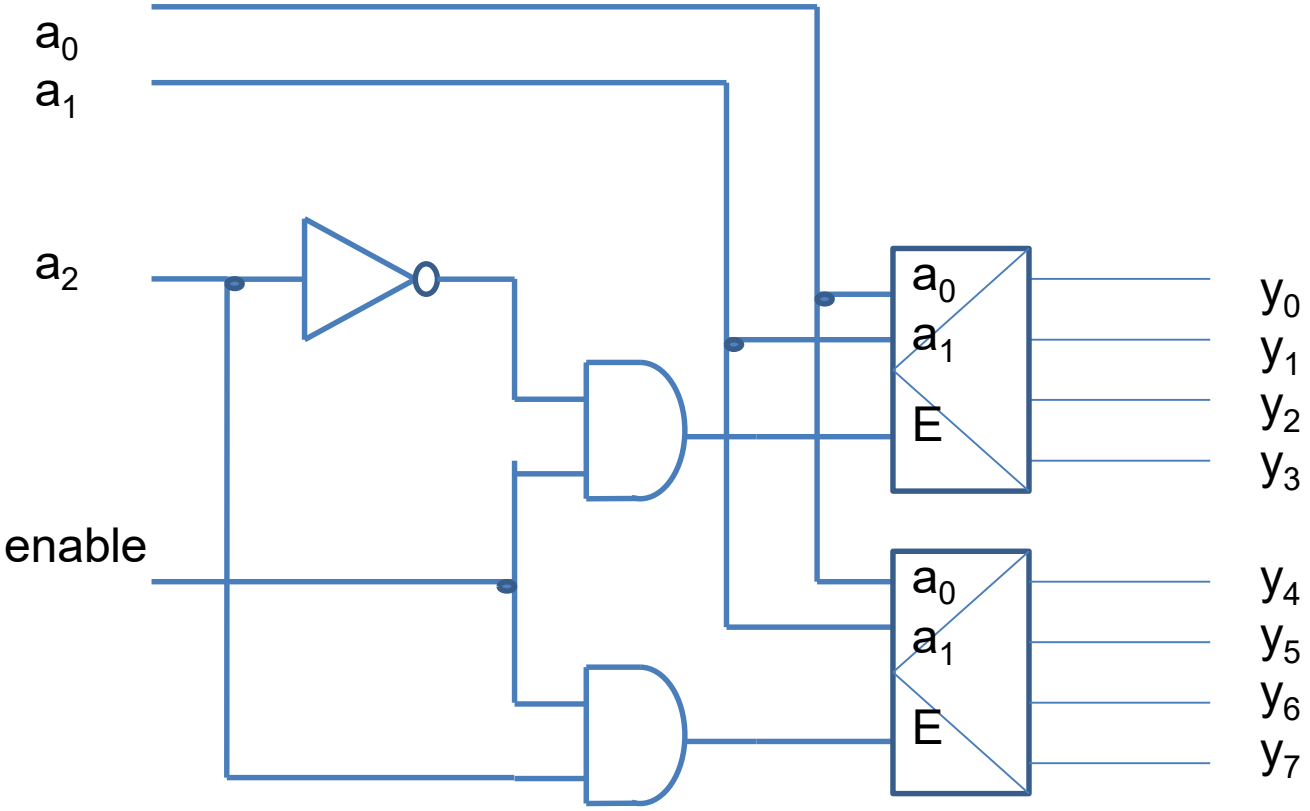
Dostępność wszystkich implikantów dla zmiennych adresowych pozwala na łatwą realizację wielu funkcji logicznych .
Uwaga: wyjścia aktywne poziomem niskim – bramka NAND dla negowanych czynników zapewni realizację sumy iloczynów



Łączenie multiplekserów

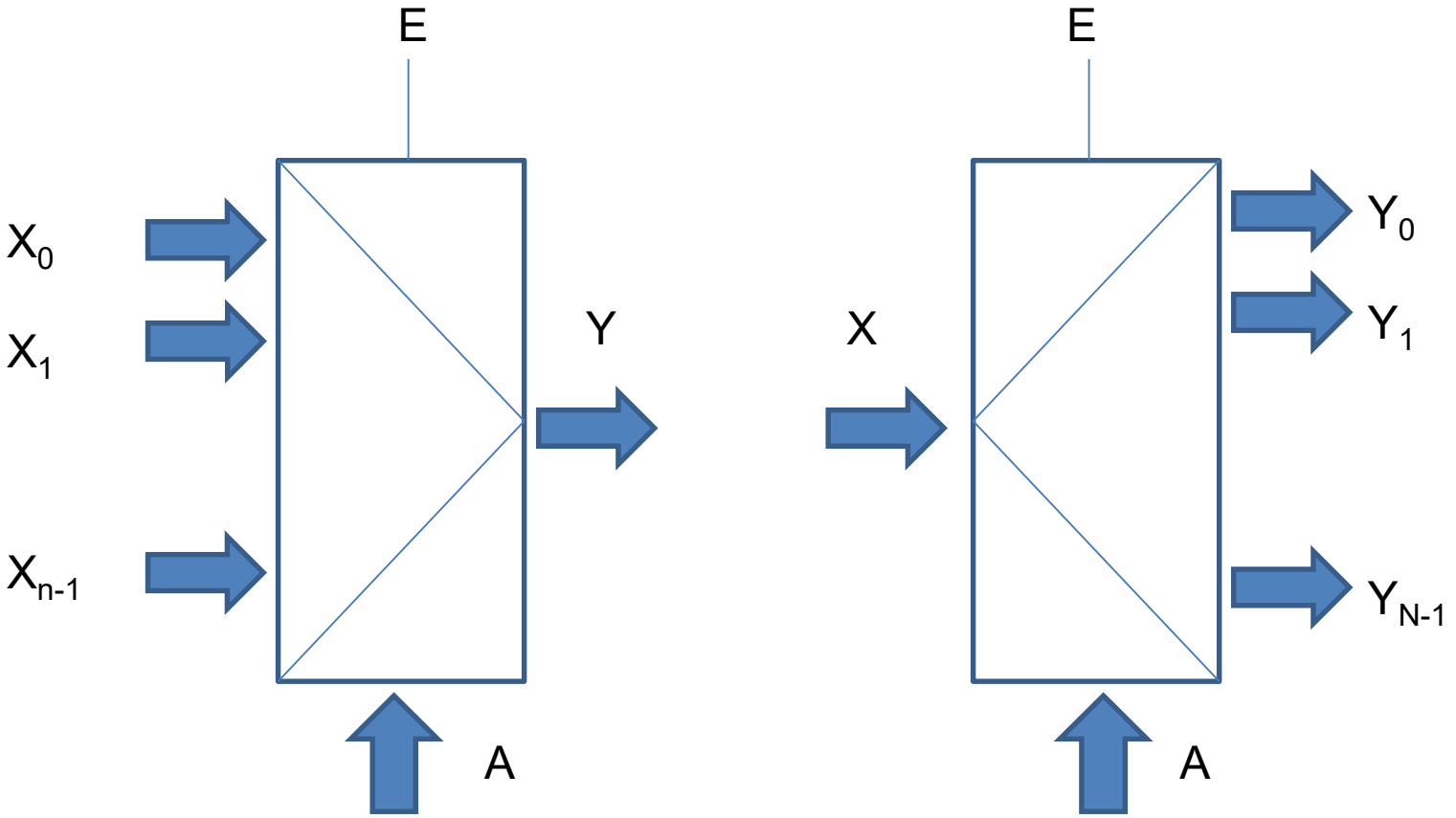
- Multiplekser 1 z 16 zbudowany jako dwu-poziomowa struktura 5 multiplekserów 1 z 4

Łączenie dekoderów



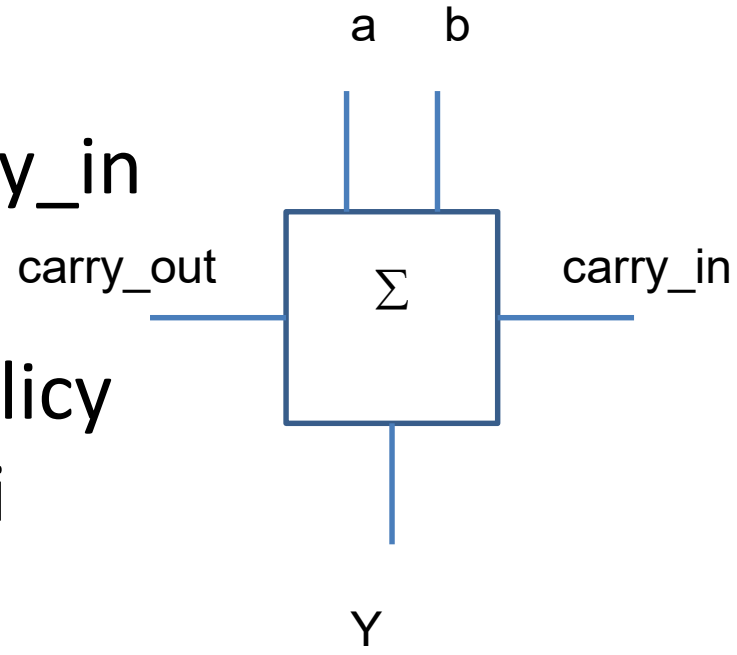
- Dekoder 3/8zbudowany z dwóch dekoderów 2/4,
- E wejście zezwalające na wybór wyjścia

Multiplekser i demultiplekser grupowy



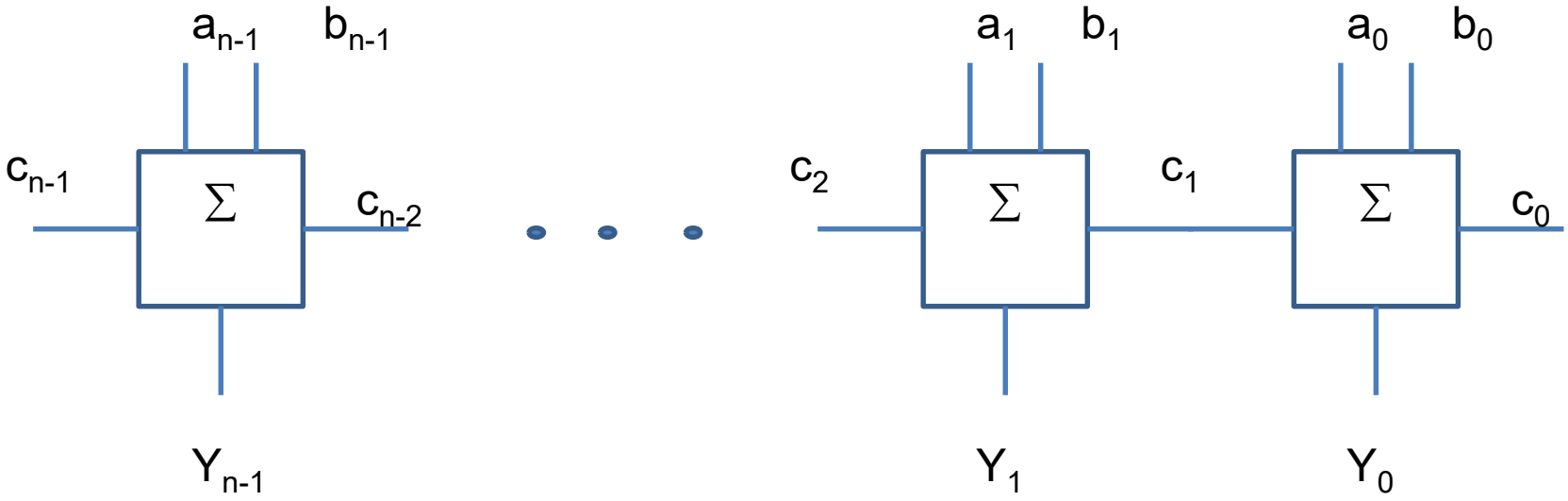
Sumator jednobitowy

- Sumator jednobitowy pełny
- Y=a ⊕ b ⊕ carry_in
- carry_out =ab+ (a+b)carry_in
  = ab+(a⊕b) carry_in
- Zależności wynikają z tablicy prawdy dla funkcji sumy i funkcji przeniesienia



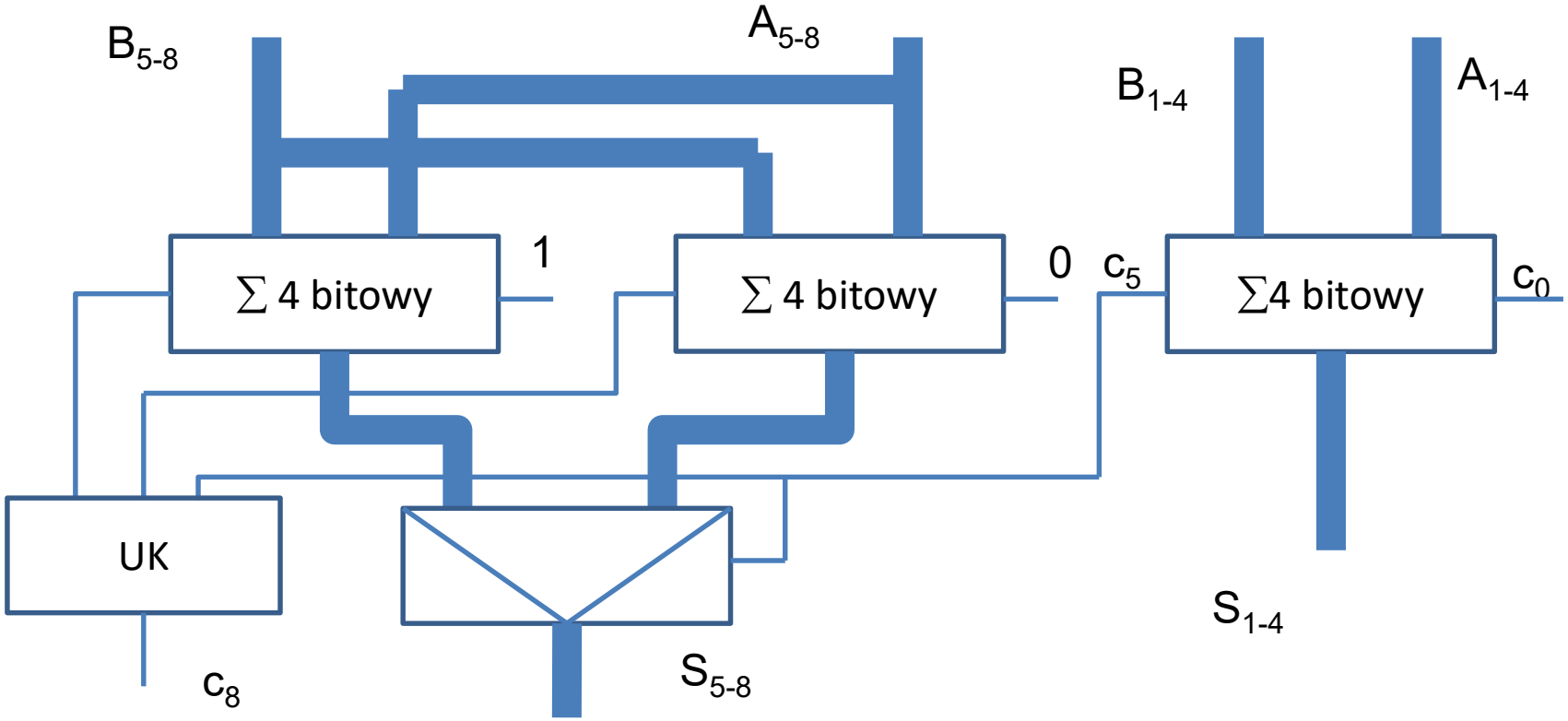
Sumator

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 ze sterowanym przeniesieniem wyborem wyniku



UK w zależności od c_5 określa skąd pochodzi przeniesienie

Podział sumatorów

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

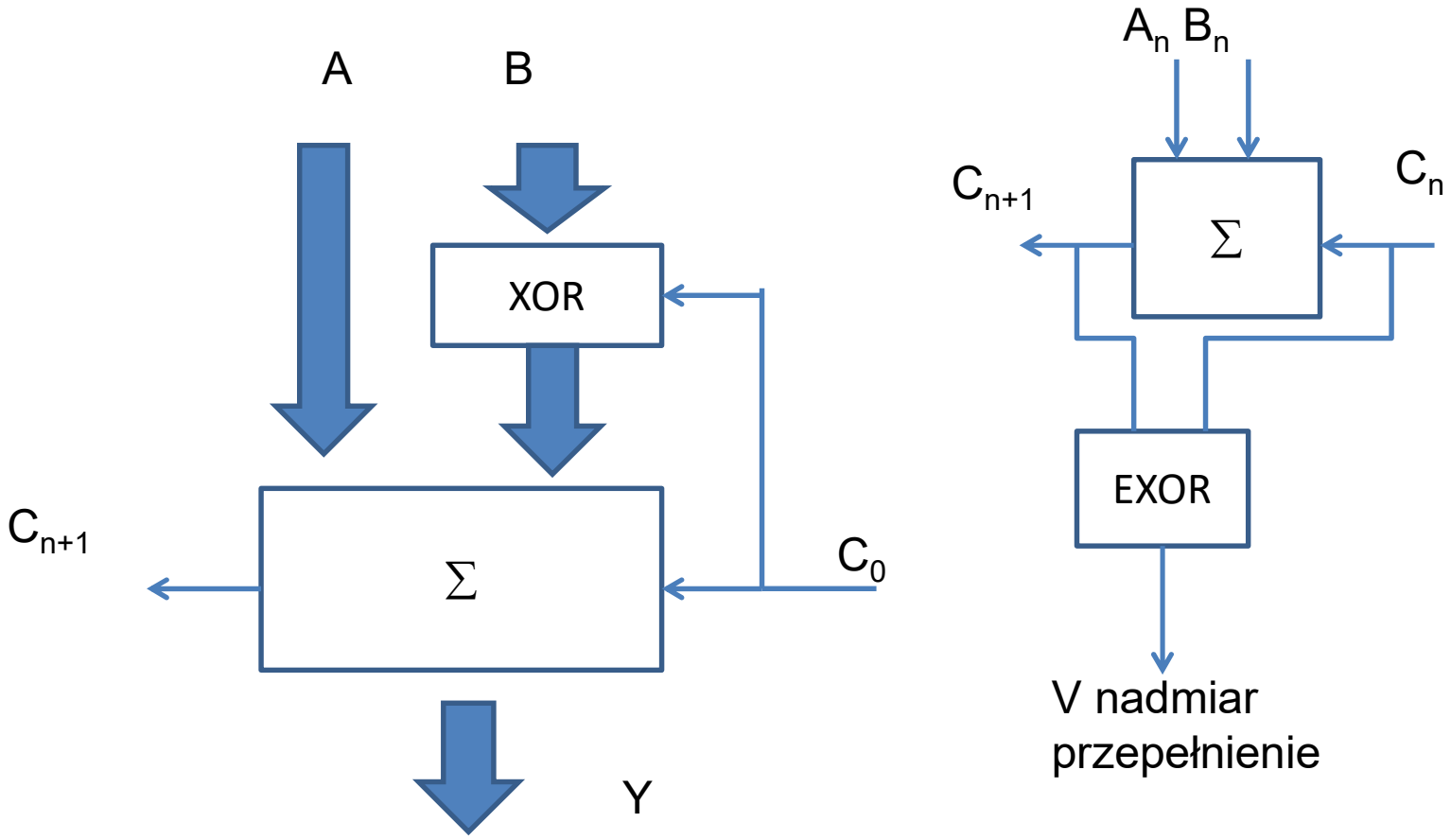
Odejmowanie

- Równanie poniższe przedstawia działanie układu sumatora lub substraktera w zależności od wartości zmiennej c. Działania realizowane są na wartościach w kodzie U2.
- Y= A+ (B⊕c)+c

gdzie ⊕oznacza sumę modulo wektora A i skalara c

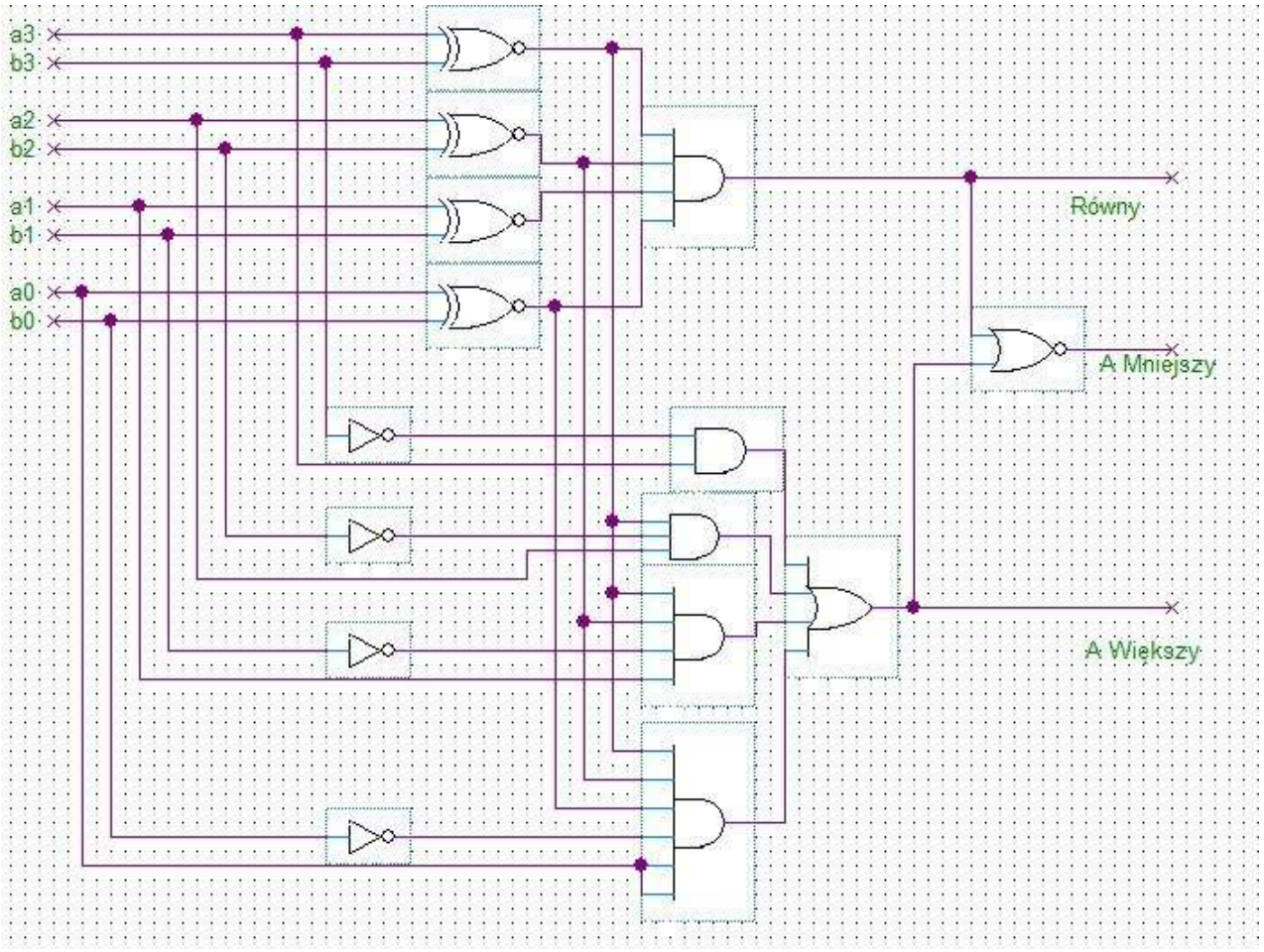
- Dla c=0 realizowane jest dodawanie.
- Dla c=1 Y= A +B’ + 1 gdzie B’ + 1 oznacza liczbę „-B” w kodzie U2.
- Zasada działania: układ dodający, w przypadku odejmowania dodaje liczbę przeciwną do odjemnika.

Układ sumatora/substraktera



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

Komparator



Porównywanie liczb binarnych dodatnich $a_3a_2a_1a_0$ i $b_3b_2b_1b_0$

Bramka exnor – równoważności

Znajdź błąd na schemacie