UKŁADY KOMBINACYJNE

KOMBINACYJNE UKŁADY CYFROWE

Układy kombinacyjne są to układy cyfrowe, których stany wyjść są zawsze jednoznacznie określone przez stany wejść.

Oznacza to, że doprowadzając na wejście układu kombinacyjnego określoną kombinację sygnałów binarnych, otrzymujemy na wyjściu odpowiedź specyficzną jedynie dla funkcji logicznej jaką układ wykonuje, niezależne od tego co działo się z tymi układem wcześniej.

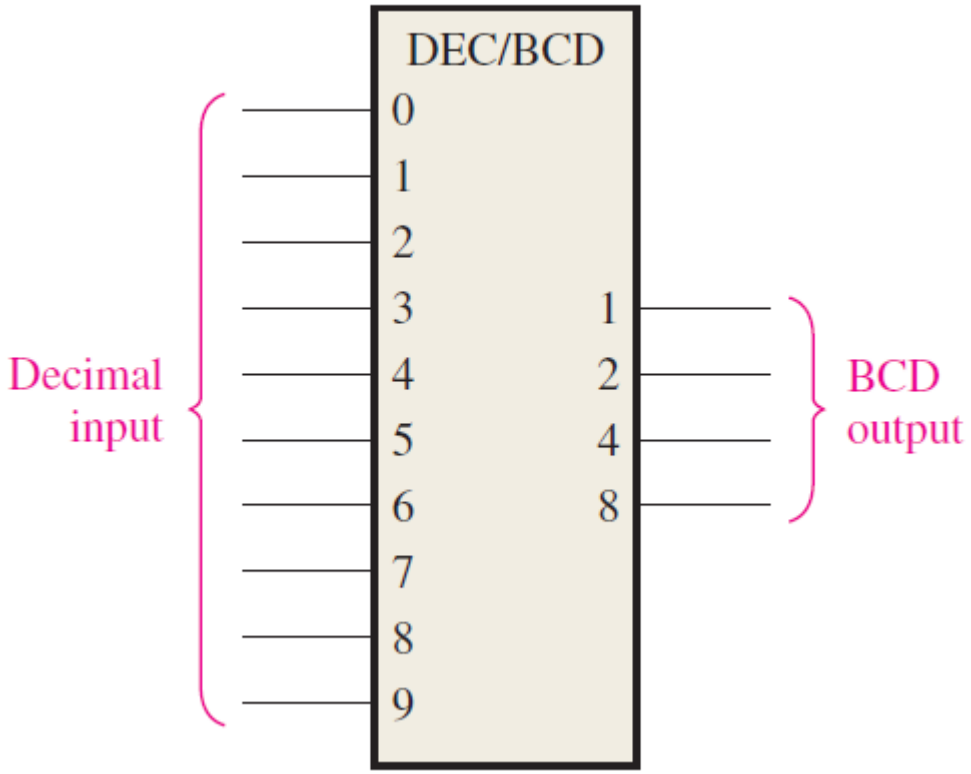
Można zatem stwierdzić, że układy kombinacyjne są układami bez pamięci.

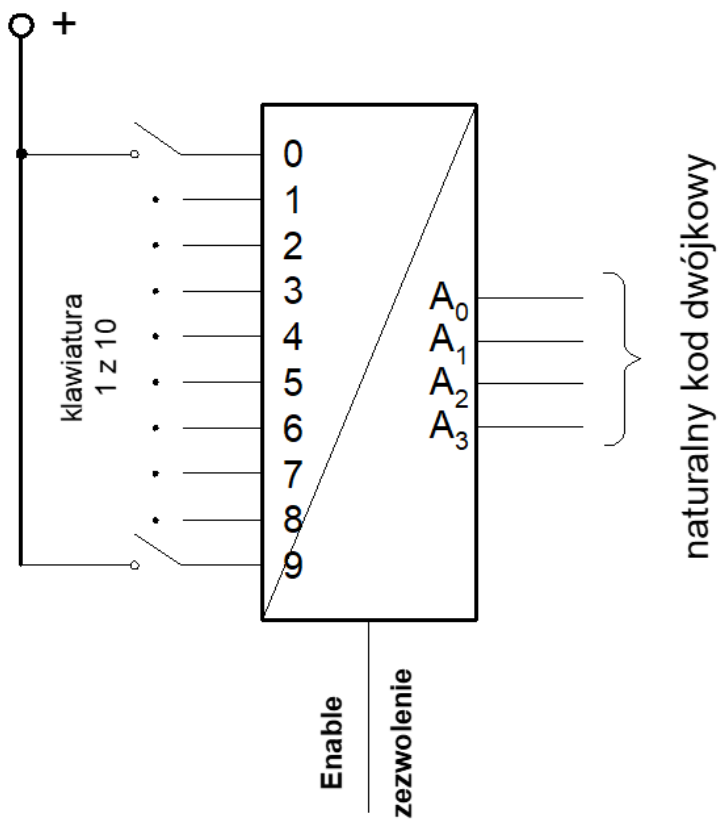
Podstawowymi przedstawicielami tych układów są bramki.

KODERY

Koderem nazywamy układ cyfrowy który zmienia słowa kodu 1 z N na odpowiadające im słowa w dowolnym innym kodzie.

koder 1 z 10 na n.k.d.





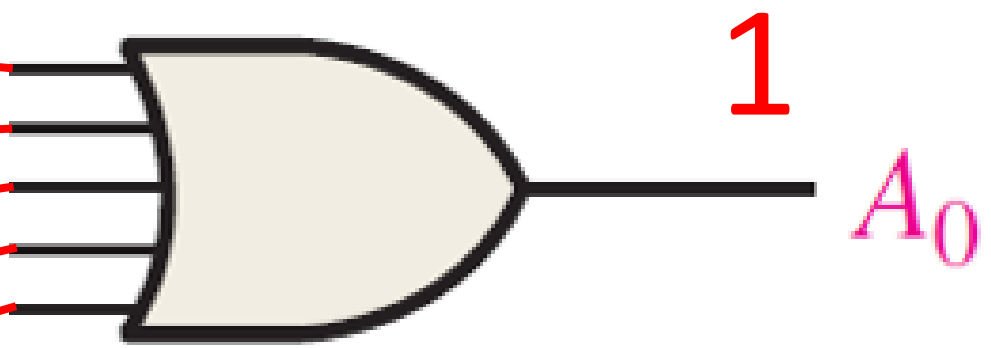
1 z 10										naturalny kod dwójkowy			
9	8	7	6	5	4	3	2	1	0	A_3	A_2	A_1	A_0
0	0	0	0	0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	0	1	0	0	0	0	1
0	0	0	0	0	0	0	1	0	0	0	0	1	0
0	0	0	0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	0	1	0	0	0	0	0	1	0	0
0	0	0	0	1	0	0	0	0	0	0	1	0	1
0	0	0	1	0	0	0	0	0	0	0	1	1	0
0	0	1	0	0	0	0	0	0	0	0	1	1	1
0	1	0	0	0	0	0	0	0	0	1	0	0	0
1	0	0	0	0	0	0	0	0	0	1	0	0	1

koder 1 z 10 na n.k.d.

w układzie mamy dziesięć wejść (we0, we2, we3, we,4, we5, we6, we7, we8, we9 i cztery wyjścia (A0, A1, A2, A3)

1 z 10										naturalny kod dwójkowy			
9	8	7	6	5	4	3	2	1	0	A_3	A_2	A_1	A_0
0	0	0	0	0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	0	**1**	0	0	0	0	1
0	0	0	0	0	0	0	1	0	0	0	0	1	0
0	0	0	0	0	0	**1**	0	0	0	0	0	1	1
0	0	0	0	0	1	0	0	0	0	0	1	0	0
0	0	0	0	**1**	0	0	0	0	0	0	1	0	1
0	0	0	1	0	0	0	0	0	0	0	1	1	0
0	0	**1**	0	0	0	0	0	0	0	0	1	1	1
0	1	0	0	0	0	0	0	0	0	1	0	0	0
1	0	0	0	0	0	0	0	0	0	1	0	0	1

na wyjściu A0 jest stan wysoki (1) gdy:
we1=1 lub we3=1 lub we5=1
lub we7=1 lub we9=1



realizuje to bramka OR pobierająca
sygnały wejściowe z we1, we3, we5, we7, we9

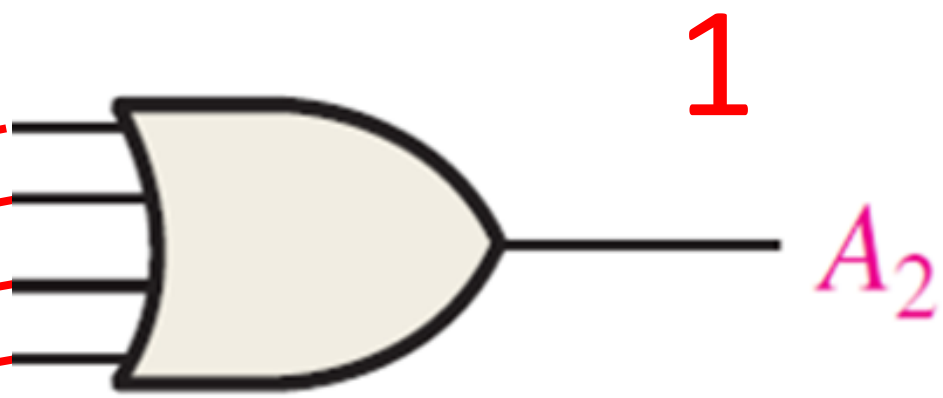
1 z 10										naturalny kod dwójkowy			
9	8	7	6	5	4	3	2	1	0	A_3	A_2	A_1	A_0
0	0	0	0	0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	0	1	0	0	0	0	1
0	0	0	0	0	0	0	1	0	0	0	0	1	0
0	0	0	0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	0	1	0	0	0	0	0	1	0	0
0	0	0	0	1	0	0	0	0	0	0	1	0	1
0	0	0	1	0	0	0	0	0	0	0	1	1	0
0	0	1	0	0	0	0	0	0	0	0	1	1	1
0	1	0	0	0	0	0	0	0	0	1	0	0	0
1	0	0	0	0	0	0	0	0	0	1	0	0	1

na wyjściu A1 jest stan wysoki (1) gdy:
we2=1 lub we3=1 lub we6=1 lub we7=1

realizuje to bramka OR pobierająca sygnały wejściowe z we2, we3, we6, we7

1 z 10										naturalny kod dwójkowy			
9	8	7	6	5	4	3	2	1	0	A_3	A_2	A_1	A_0
0	0	0	0	0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	0	1	0	0	0	0	1
0	0	0	0	0	0	0	1	0	0	0	0	1	0
0	0	0	0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	0	1	0	0	0	0	0	1	0	0
0	0	0	0	1	0	0	0	0	0	0	1	0	1
0	0	0	1	0	0	0	0	0	0	0	1	1	0
0	0	1	0	0	0	0	0	0	0	0	1	1	1
0	1	0	0	0	0	0	0	0	0	1	0	0	0
1	0	0	0	0	0	0	0	0	0	1	0	0	1

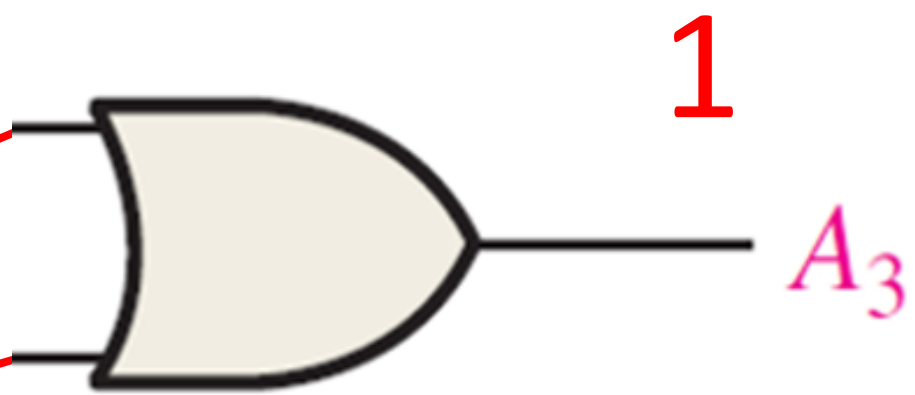
na wyjściu A2 jest stan wysoki (1) gdy:
we4=1 lub we5=1 lub we6=1 lub we7=1



realizuje to bramka OR pobierająca sygnały wejściowe z we4, we5, we6, we7

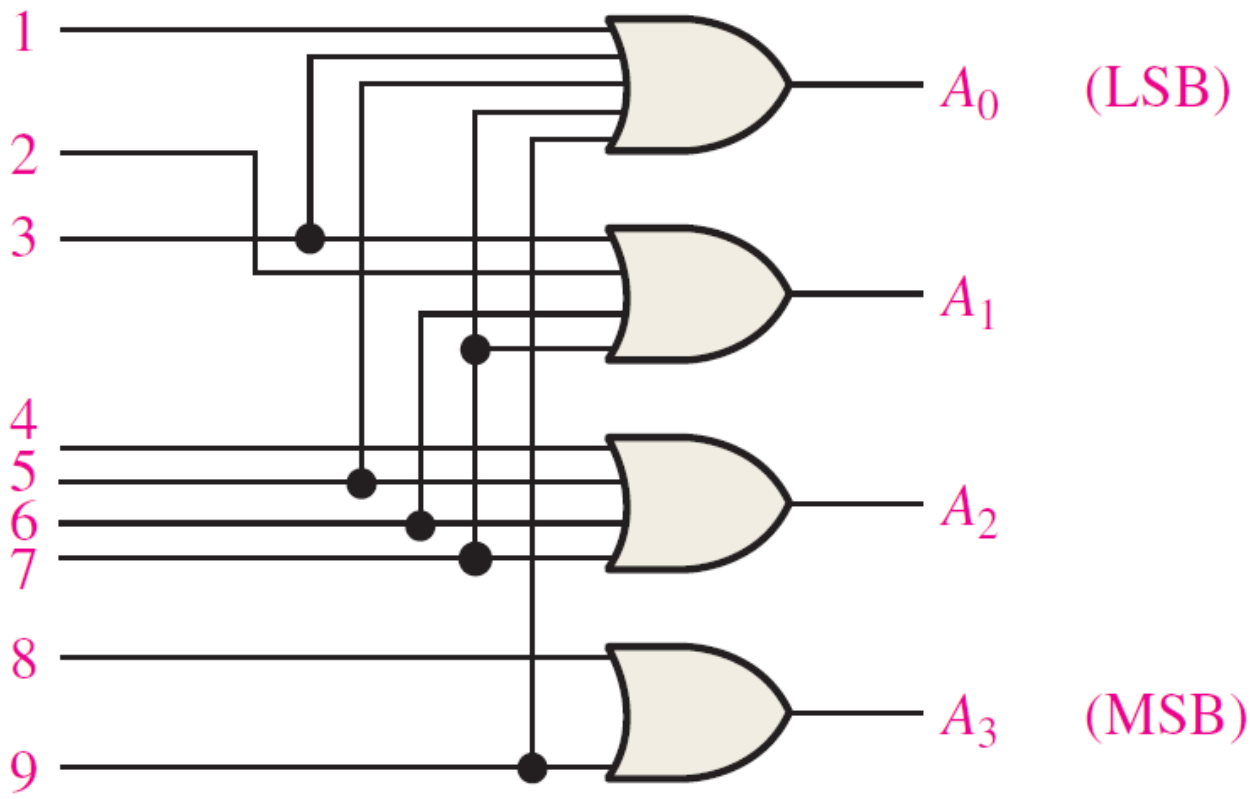
1 z 10										naturalny kod dwójkowy			
9	8	7	6	5	4	3	2	1	0	A_3	A_2	A_1	A_0
0	0	0	0	0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	0	1	0	0	0	0	1
0	0	0	0	0	0	0	1	0	0	0	0	1	0
0	0	0	0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	0	1	0	0	0	0	0	1	0	0
0	0	0	0	1	0	0	0	0	0	0	1	0	1
0	0	0	1	0	0	0	0	0	0	0	1	1	0
0	0	1	0	0	0	0	0	0	0	0	1	1	1
0	1	0	0	0	0	0	0	0	0	1	0	0	0
1	0	0	0	0	0	0	0	0	0	1	0	0	1

na wyjściu A2 jest stan wysoki (1) gdy:
we8=1 lub we9=1



realizuje to bramka OR pobierająca sygnały wejściowe z we8, we9

kompletny układ kodera (*złożenie czterech poprzednich bramek w jeden układ*)



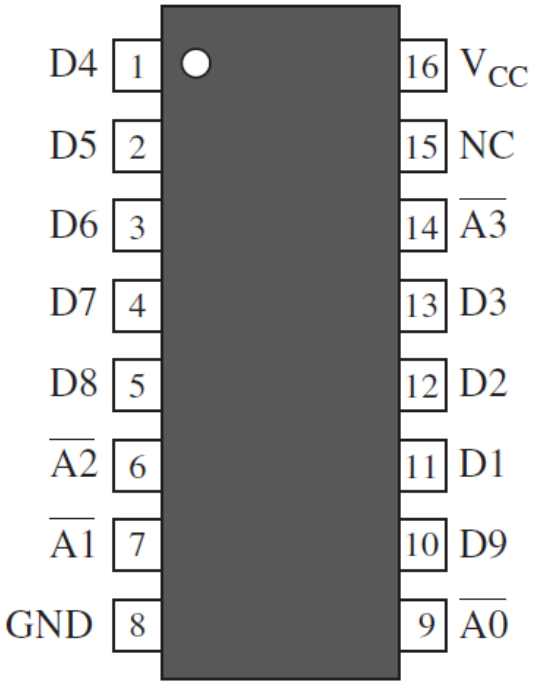
Schemat logiczny kodera 1 z 10 na n.k.d. [*]

Uwaga: wejścia ze stanem niskim (0) nie uwzględnia się ponieważ wyjścia BCD są w stanie niskim (0), gdy nie ma na wejściach wymuszenia stanem wysokim (1)

realizacja w postaci scalonej

koder 1 z 10 na n.k.d.

74HC147



układ w obudowie
z oznaczeniem kontaktów (pinów)

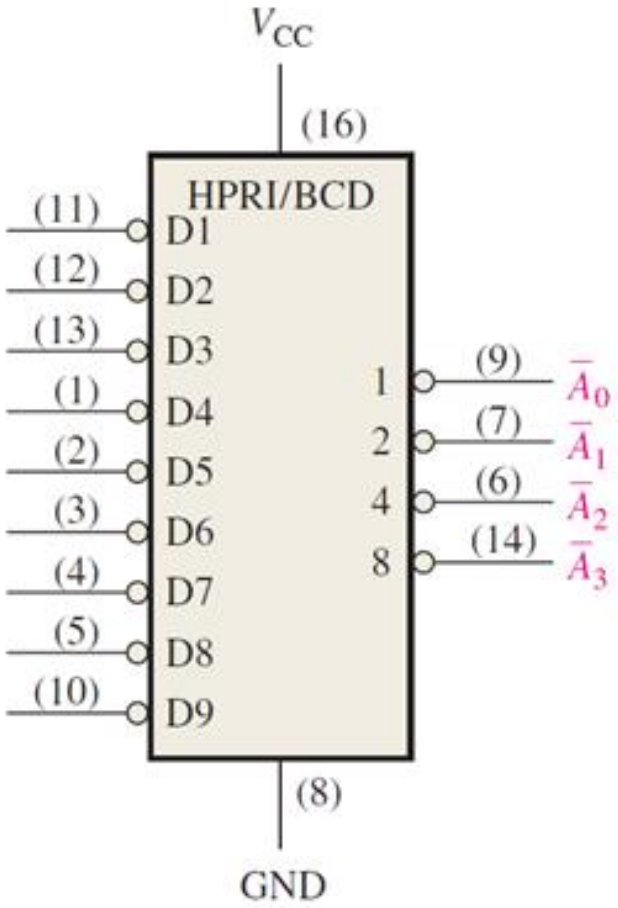


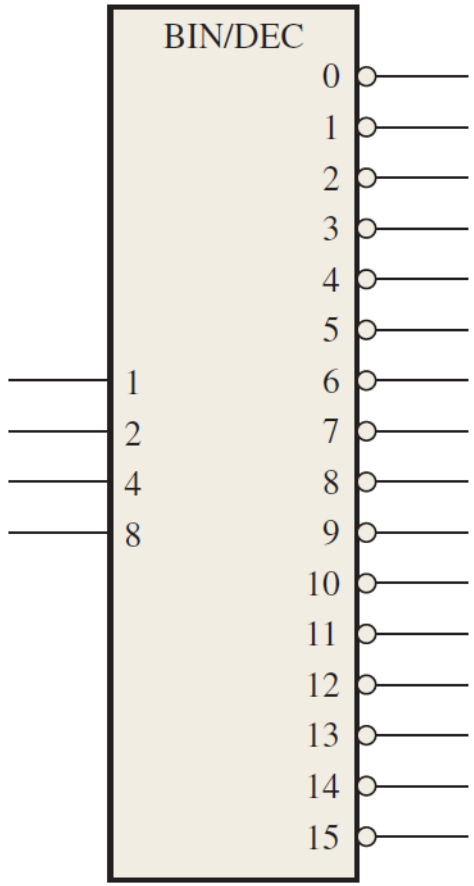
diagram logiczny układu
numery kontaktów (pinów) w nawiasach

wejścia i wyjścia aktywne w stanie niskim (0) - symbol negacji
dziewięć wejść dlatego, zero na wyjściu (w n.k.d.) pojawia się ,
gdy żadne z wejść nie jest aktywne - w ten sposób zaoszczędzono jeden pin

DEKODERY

Dekoderem nazywamy układ cyfrowy, który zmienia słowo określone w dowolnym kodzie na słowo w kodzie 1 z N

dekoder n.k.d. na 1 z 16
(*4-Bit decoder, 4-line-to-16-line decoder*)

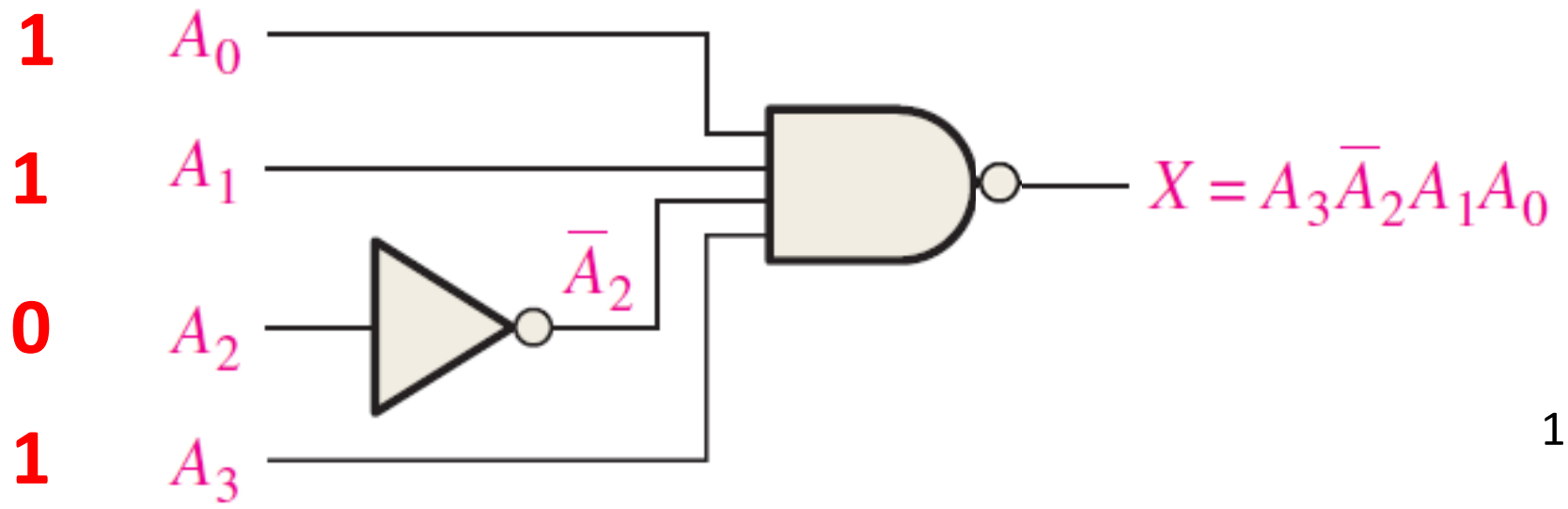


dekoder n.k.d. na 1 z 16

Binary Inputs				Decoding	Outputs																Decimal
A_3	A_2	A_1	A_0	Function	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	Digit
0	0	0	0	$\overline{A}_3\overline{A}_2\overline{A}_1\overline{A}_0$	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
0	0	0	1	$\overline{A}_3\overline{A}_2\overline{A}_1A_0$	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0	0	1	0	$\overline{A}_3\overline{A}_2A_1\overline{A}_0$	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	2
0	0	1	1	$\overline{A}_3\overline{A}_2A_1A_0$	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	3
0	1	0	0	$\overline{A}_3A_2\overline{A}_1\overline{A}_0$	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	4
0	1	0	1	$\overline{A}_3A_2\overline{A}_1A_0$	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	5
0	1	1	0	$\overline{A}_3A_2A_1\overline{A}_0$	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	6
0	1	1	1	$\overline{A}_3A_2A_1A_0$	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	7
1	0	0	0	$A_3\overline{A}_2\overline{A}_1\overline{A}_0$	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	8
1	0	0	1	$A_3\overline{A}_2\overline{A}_1A_0$	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	9
1	0	1	0	$A_3\overline{A}_2A_1\overline{A}_0$	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	10
1	0	1	1	$A_3\overline{A}_2A_1A_0$	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	11
1	1	0	0	$A_3A_2\overline{A}_1\overline{A}_0$	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	12
1	1	0	1	$A_3A_2\overline{A}_1A_0$	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	13
1	1	1	0	$A_3A_2A_1\overline{A}_0$	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	14
1	1	1	1	$A_3A_2A_1A_0$	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	15

rozbudowana tabela prawdy dekodera

przykładowe powstawanie sygnału na wyjściu jedenastym dekodera

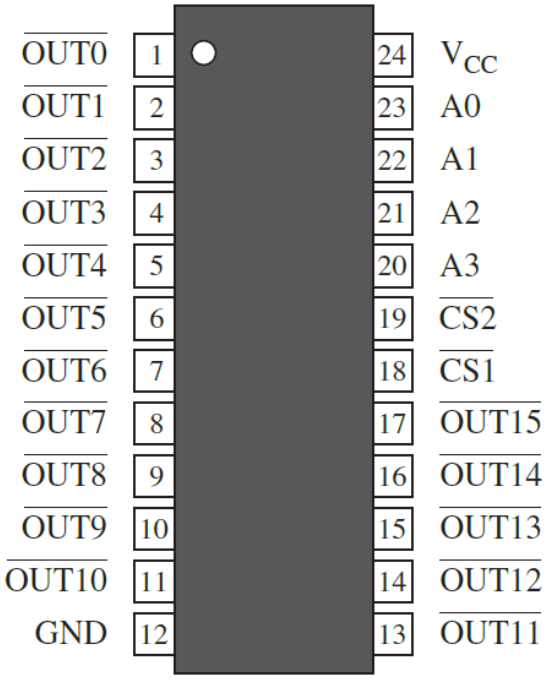


stan
na wyjściu
11-tym dekodera

dekodowanie stanu 1011_2 czyli 11_{10}

dekoder n.k.d. na 1 z 16

74HC154



układ w obudowie
z oznaczeniem kontaktów (pinów)

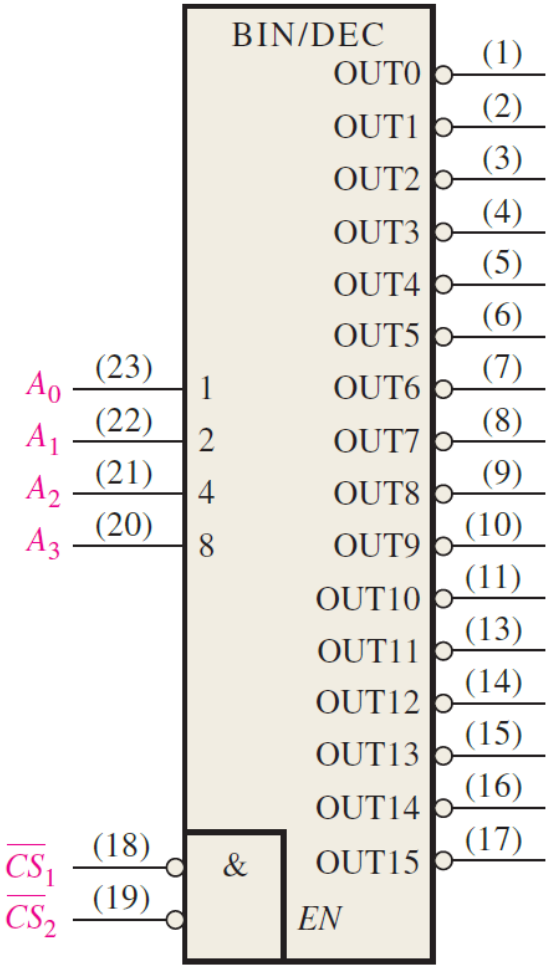
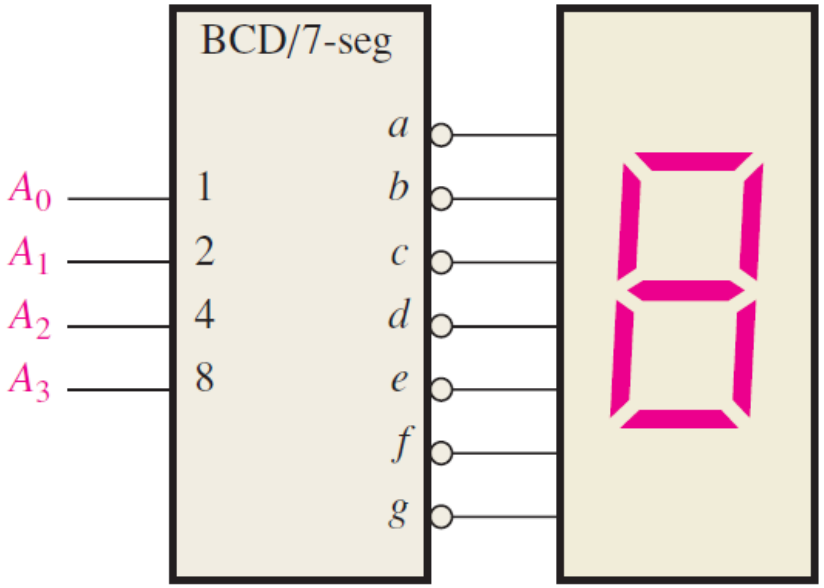


diagram logiczny układu

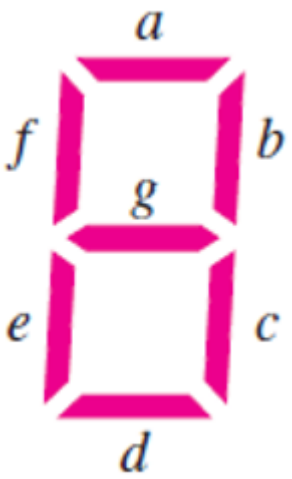
dekoder n.k.d. na kod siedmiosegmentowy (*BCD-to-7-segment decoder*)



wskaźnik siedmiosegmentowy

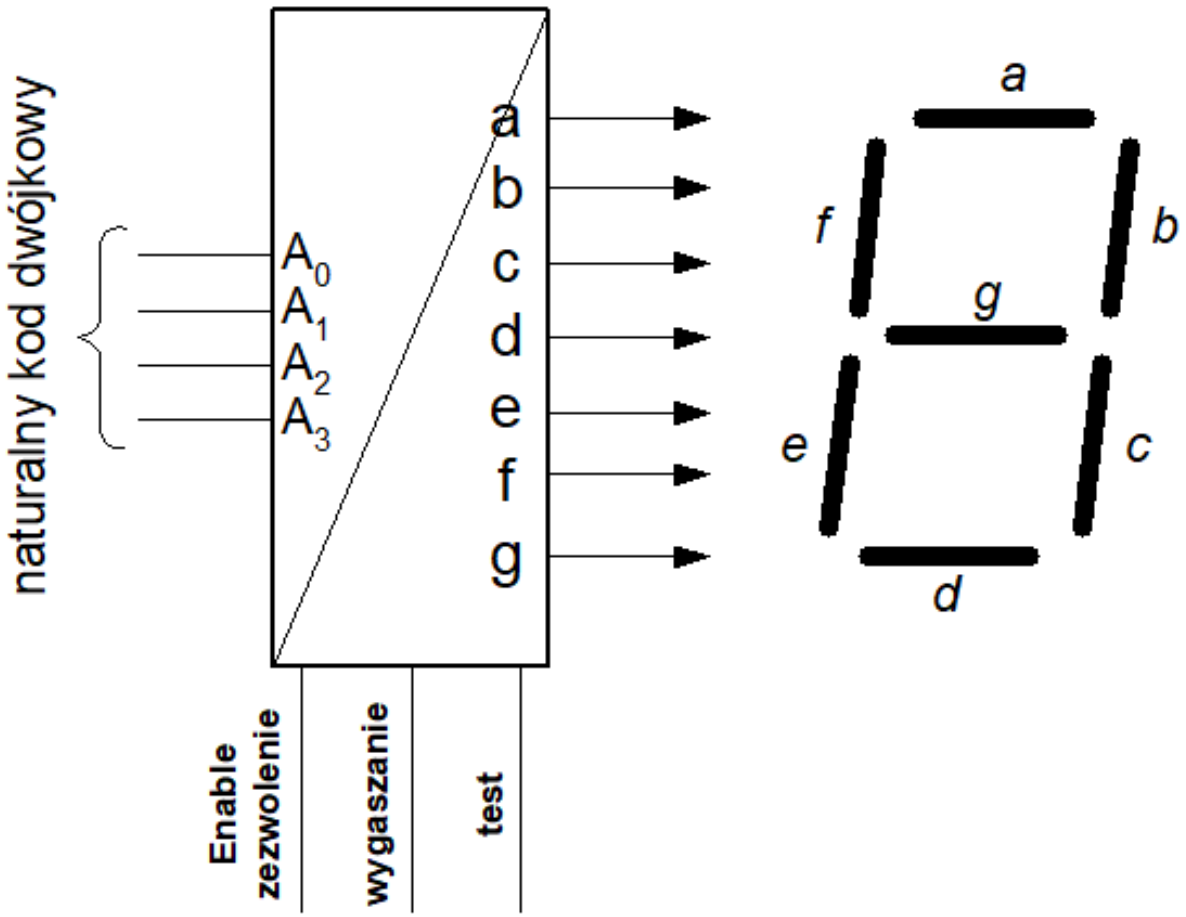
segmenty zbudowane z:

- LED (diody elektroluminescencyjne)
  - wskaźnik świeci samodzielnym światłem
- LCD (ciekły kryształ) - wskaźnik musi być oświetlany



wyświetla cyfry

może też wyświetlać niektóre litery



dekoder n.k.d. na kod siedmiosegmentowy

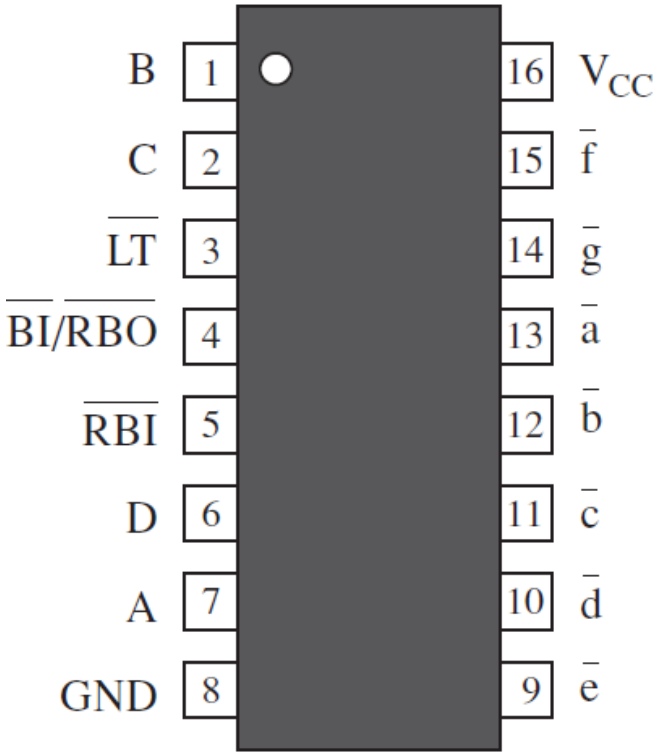
	naturalny kod dwójkowy				kod 7-mio segmentowy						
	A_3	A_2	A_1	A_0	a	b	c	d	e	f	g
0	0	0	0	0	1	1	1	1	1	1	0
1	0	0	0	1	0	1	1	0	0	0	0
2	0	0	1	0	1	1	0	1	1	0	1
3	0	0	1	1	1	1	1	1	0	0	1
4	0	1	0	0	0	1	1	0	0	1	1
5	0	1	0	1	1	0	1	1	0	1	1
6	0	1	1	0	1	0	1	1	1	1	1
7	0	1	1	1	1	1	1	0	0	0	0
8	1	0	0	0	1	1	1	1	1	1	1
9	1	0	0	1	1	1	1	1	0	1	1
	wygaszanie				0	0	0	0	0	0	0
	test				1	1	1	1	1	1	1

tabela prawdy

	naturalny kod dwójkowy				kod 7-mio segmentowy						
	A_3	A_2	A_1	A_0	a	b	c	d	e	f	g
0	0	0	0	0	1	1	1	1	1	1	0
1	0	0	0	1	0	1	1	0	0	0	0
2	0	0	1	0	1	1	0	1	1	0	1
3	0	0	1	1	1	1	1	1	0	0	1
4	0	1	0	0	0	1	1	0	0	1	1
5	0	1	0	1	1	0	1	1	0	1	1
6	0	1	1	0	1	0	1	1	1	1	1
7	0	1	1	1	1	1	1	0	0	0	0
8	1	0	0	0	1	1	1	1	1	1	1
9	1	0	0	1	1	1	1	1	0	1	1
	wygaszanie				0	0	0	0	0	0	0
	test				1	1	1	1	1	1	1

dekoder n.k.d. na kod siedmiosegmentowy

74HC47



układ w obudowie
z oznaczeniem kontaktów (pinów)

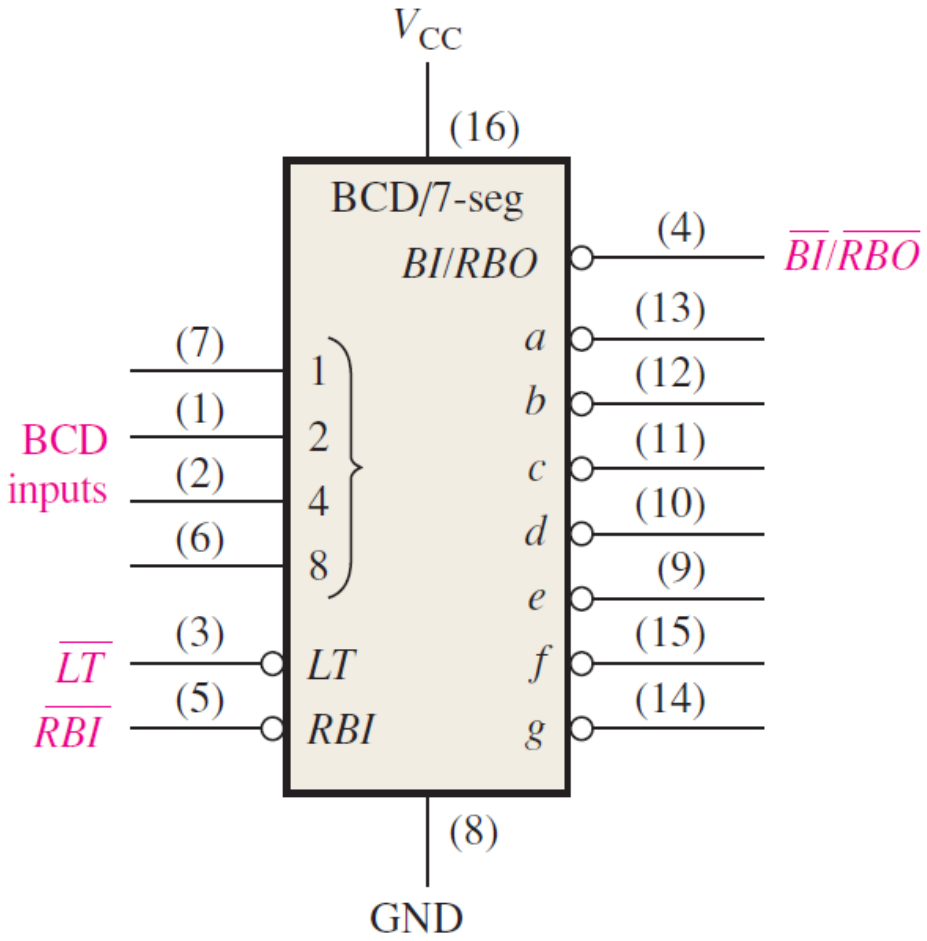


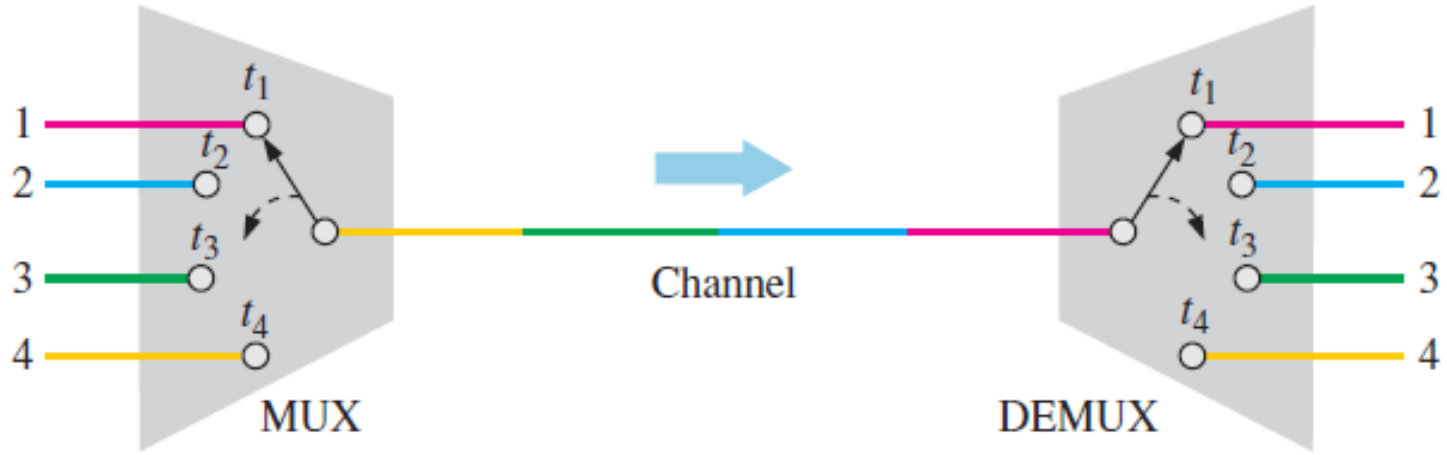
diagram logiczny układu

MULTIPLEKSER I DEMULTIPLEKSER

Multipleksowanie służy do przesyłania danych cyfrowych z wielu źródeł jednym kanałem komunikacyjnym.

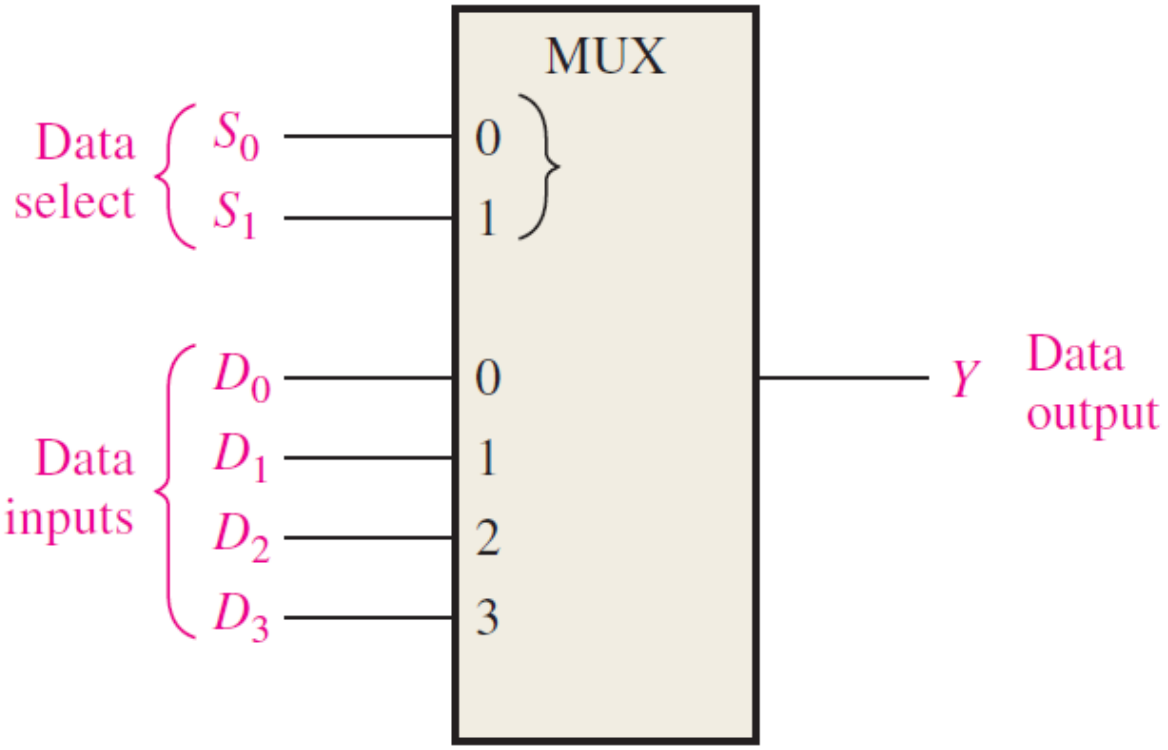
Demultipleksowanie to proces rozdzielania danych z jednego kanału na wiele kanałów.

Multipleksowanie i demultipleksowanie jest szeroko stosowane w telekomunikacji i sieciach komputerowych.



Idea multipleksowania i demultipleksowania [*]

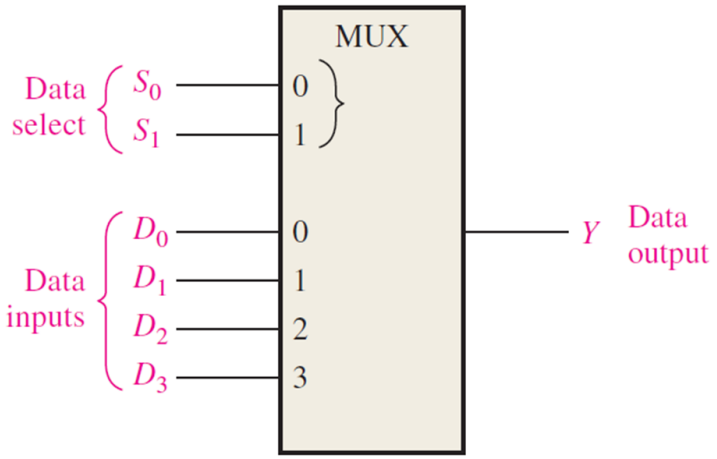
MULTIPLEKSER



multiplekser czterowejściowy [*]

Data-Select Inputs		
S_1	S_0	Input Selected
0	0	D_0
0	1	D_1
1	0	D_2
1	1	D_3

tabela prawdy multipleksera czterowejściowego



Data-Select Inputs		
S_1	S_0	Input Selected
0	0	D_0
0	1	D_1
1	0	D_2
1	1	D_3

z wejścia D_0 do wyjścia Y przekazywany sygnał wtedy, gdy S_1=0 i S_0 =0 zatem $Y = D_0 \bar{S}_1 \bar{S}_0$

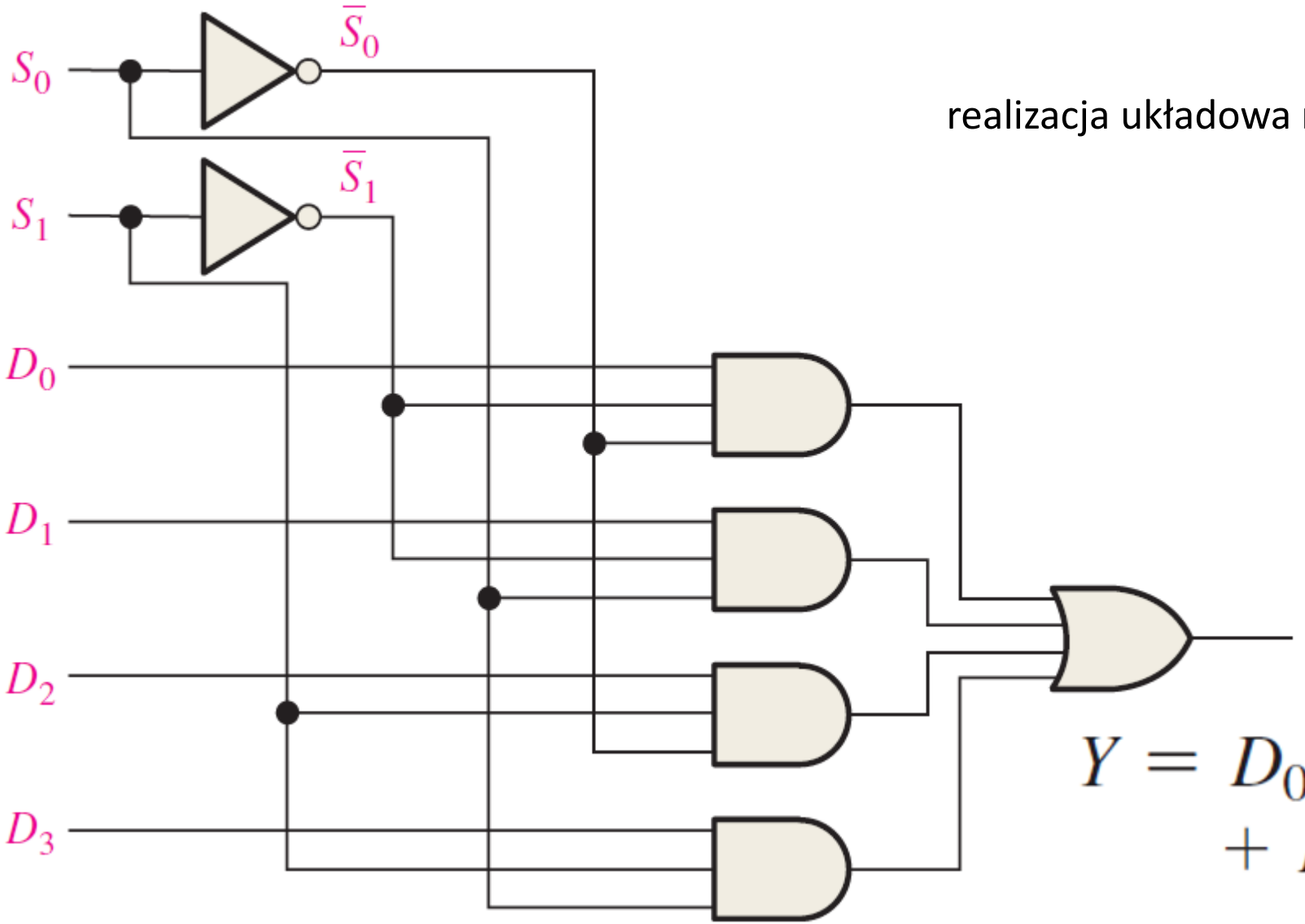
z wejścia D_1 do wyjścia Y przekazywany sygnał wtedy, gdy S_1=0 i S_0 =1 zatem $Y = D_1 \bar{S}_1 S_0$

z wejścia D_2 do wyjścia Y przekazywany sygnał wtedy, gdy S_1=1 i S_0 =0 zatem $Y = D_2 S_1 \bar{S}_0$

z wejścia D_3 do wyjścia Y przekazywany sygnał wtedy, gdy S_1=1 i S_0 =1 zatem $Y = D_3 S_1 S_0$

$$Y = D_0\overline{S}_1\overline{S}_0 + D_1\overline{S}_1 S_0 + D_2 S_1\overline{S}_0 + D_3 S_1 S_0$$

realizacja układowa multipleksera [*]

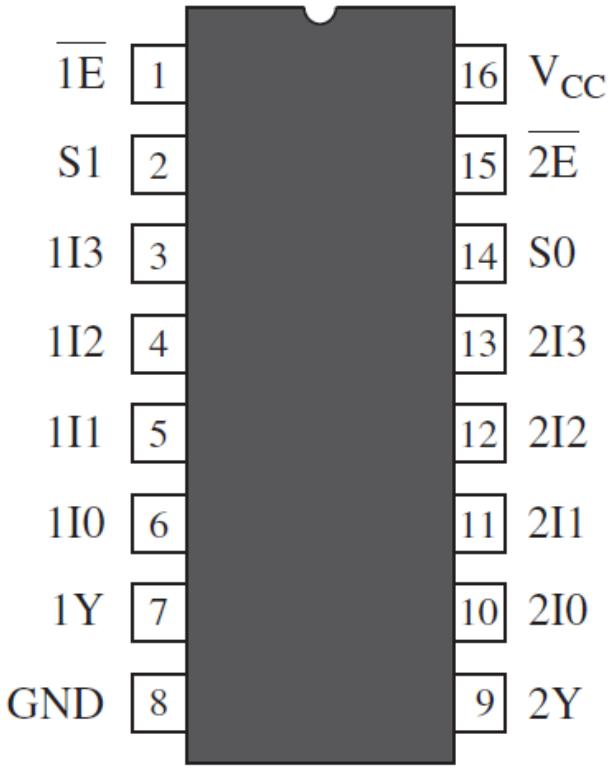


$$Y = D_0\overline{S}_1\overline{S}_0 + D_1\overline{S}_1S_0 + D_2S_1\overline{S}_0 + D_3S_1S_0$$

realizacja w postaci scalonej

podwójny czterowejściowy multiplekser

4HC153



układ w obudowie
z oznaczeniem kontaktów (pinów)

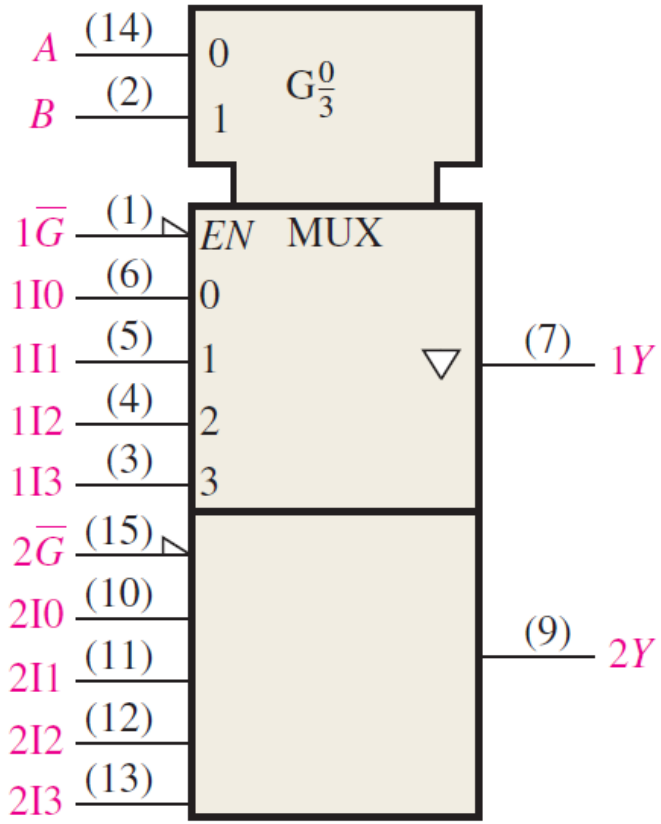


diagram logiczny układu

DEMULTIPLEKSER

Demultiplekser odwraca funkcję multipleksowania.

Pobiera informacje cyfrowe z jednej linii i dystrybuuje je do określonej liczby linii wyjściowych.

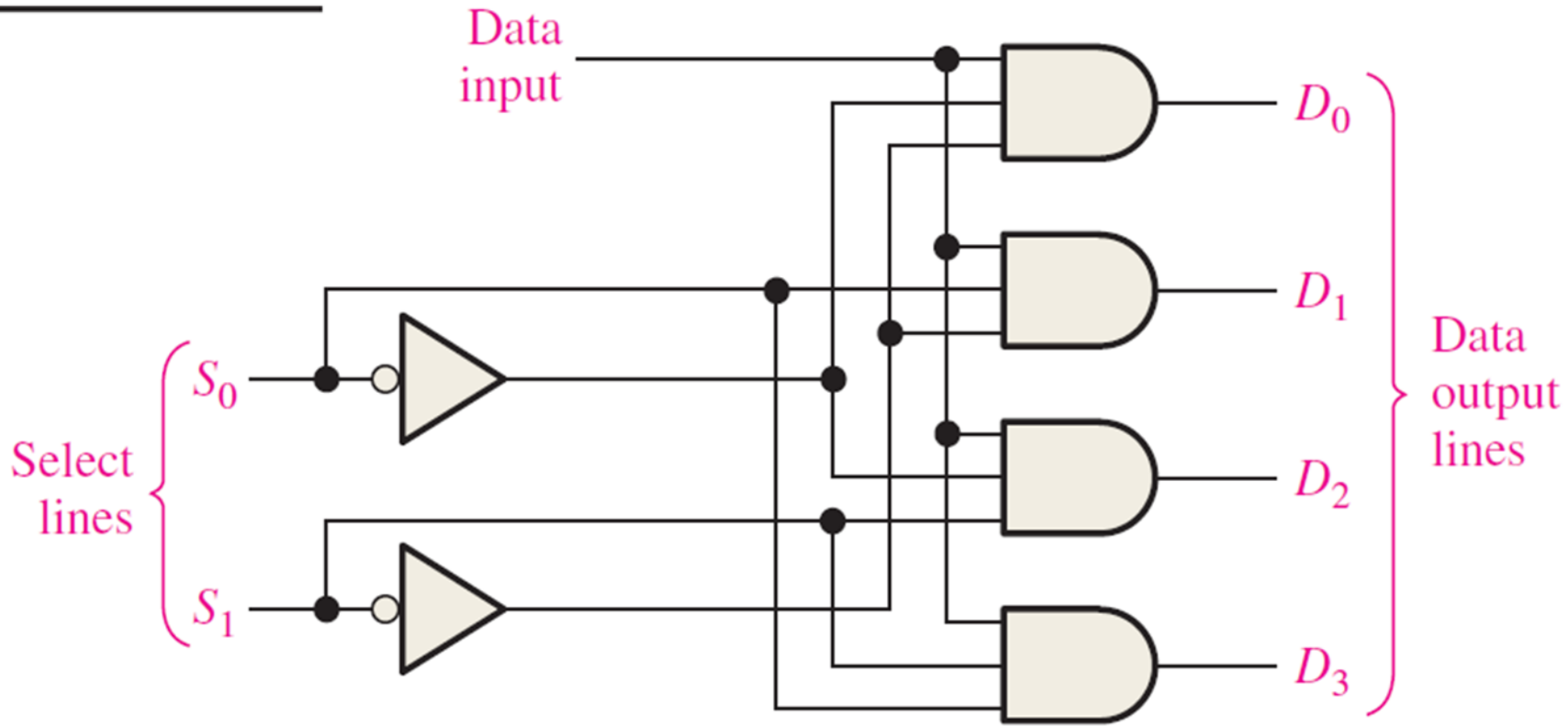
Z tego powodu demultiplekser jest zwany dystrybutorem danych.

Jako demultipleksery mogą być używane dekodery.

tabela prawdy

S_1	S_0	Data output lines
0	0	D_0
0	1	D_1
1	0	D_2
1	1	D_3

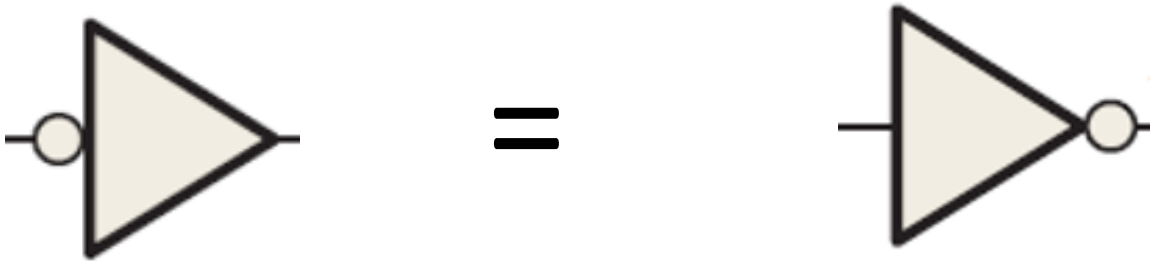
realizacja układowa demultipleksera 4-liniowego [*]



! podstawowa rola bramek AND jako elementów sterujących dystrybucją sygnału wyjściowego

wyjaśnienie użytej symboliki

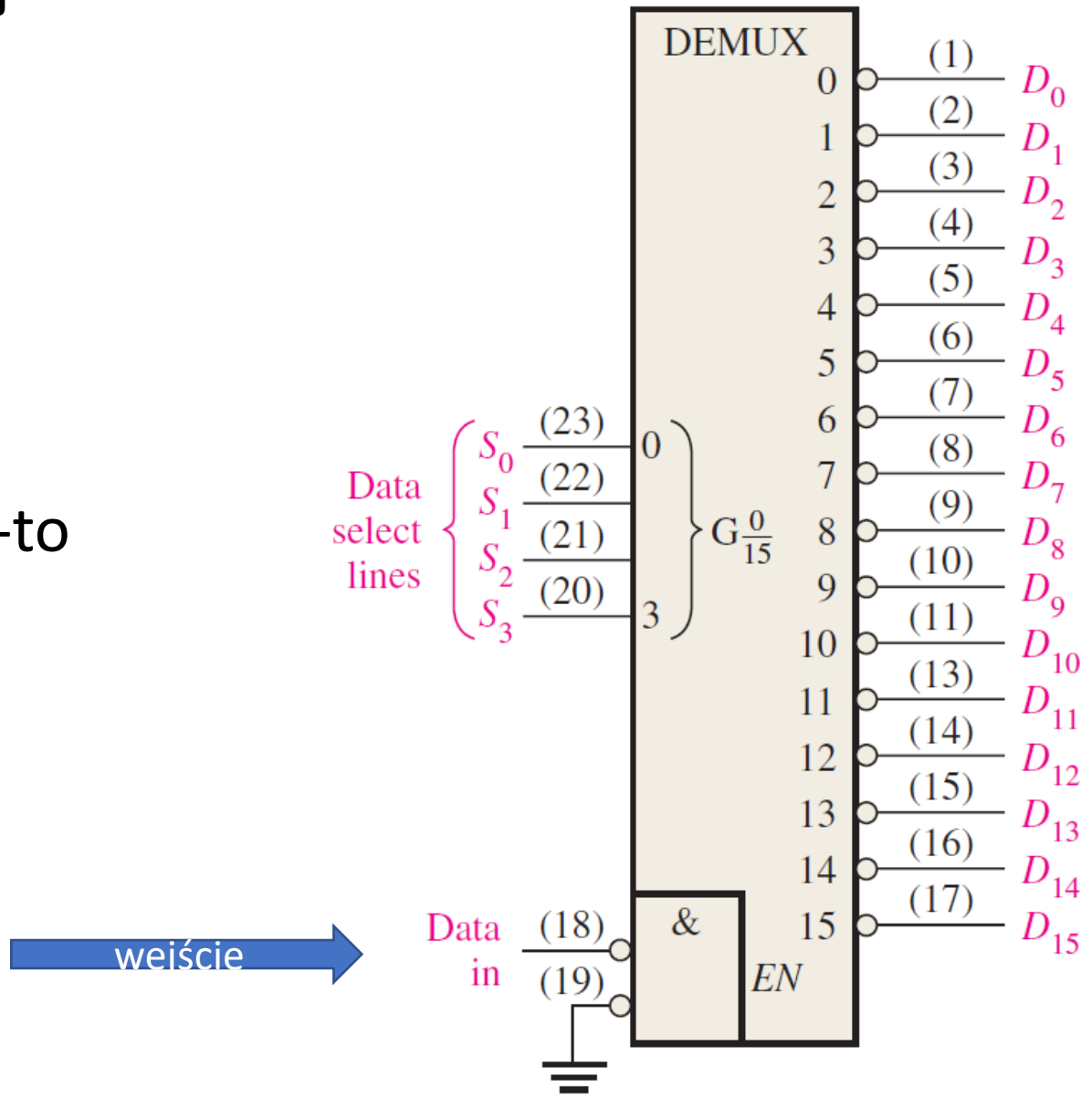
istnieją symbole równoważne bramek w sensie praw de Morgana, ale dla NOT jest jak poniżej



realizacja w postaci scalonej

dekoder
n.k.d. na 1 z 16
jako
demultiplekser 16-to
liniowy

74HC154



wejście

linie wejściowe są używane jako linie wyboru danych

wyjścia na poszczególne linie

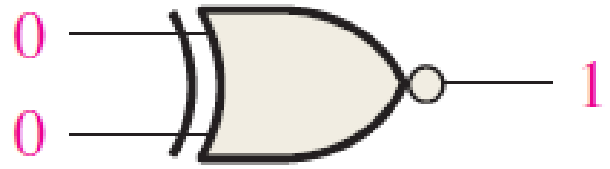
Dekoder zmienia słowo określone
w dowolnym kodzie na słowo w kodzie 1 z N

KOMPARATORY

Komparatory są to cyfrowe układy kombinacyjne, które porównują ze sobą dwie liczby

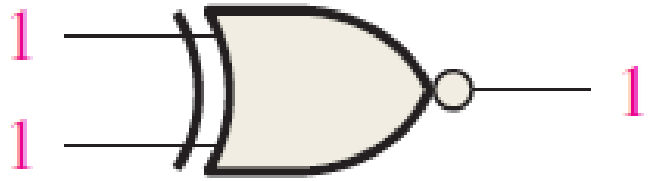
Najważniejsze kryteria porównania dwóch liczb A i B to: A>B, A=B i A<B

Bramka EXNOR może być używana jako podstawowy komparator, ponieważ jej wyjście ma wartość 0, jeśli dwa bity wejściowe nie są równe, oraz 1, jeśli bity wejściowe są równe.

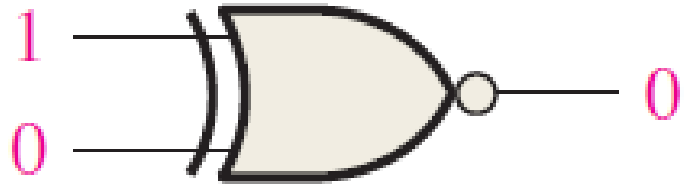


bity wejściowe są równe

bity wejściowe nie są równe



bity wejściowe są równe



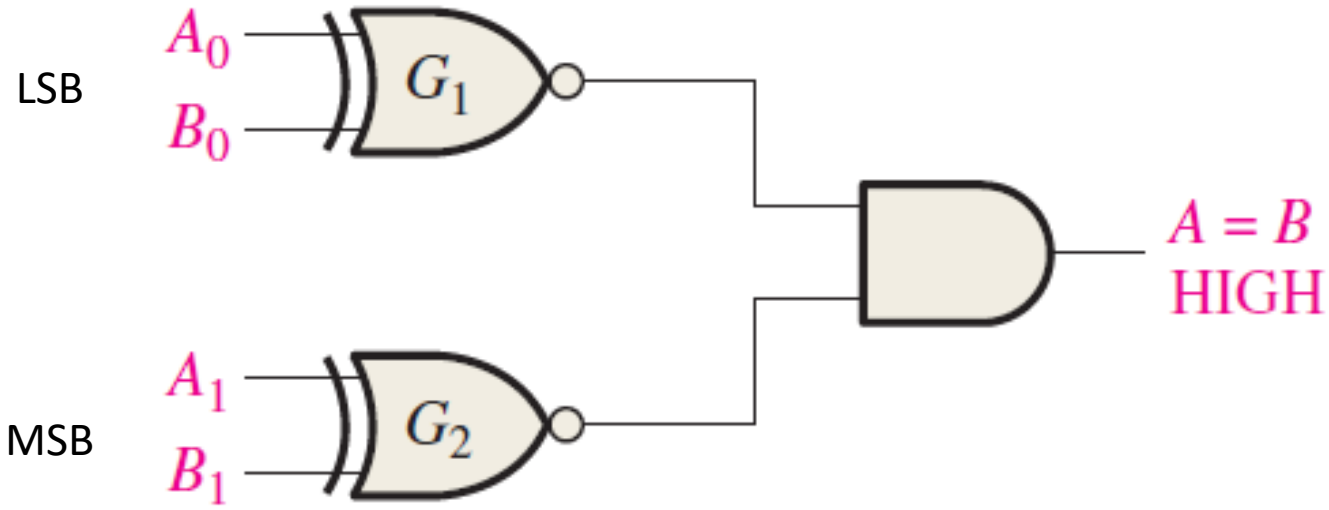
bity wejściowe nie są równe

Bramka EXNOR jako komparator

komparator identyczności dwóch liczb dwubitowych

Dwa najmniej znaczące bity (LSB) dwóch liczb są porównywane przez bramkę G1
dwa najbardziej znaczące bity (MSB) są porównywane przez bramkę G2.
Jeśli te dwie liczby są równe, odpowiadające im bity są takie same, a na wyjściu każdej bramki EXNOR jest 1.
Jeśli odpowiednie zestawy bitów nie są równe, na wyjściu bramki EXNOR występuje 0.
Iloczyn (bramka AND) daje odpowiedz odpowiednio 1 lub 0.

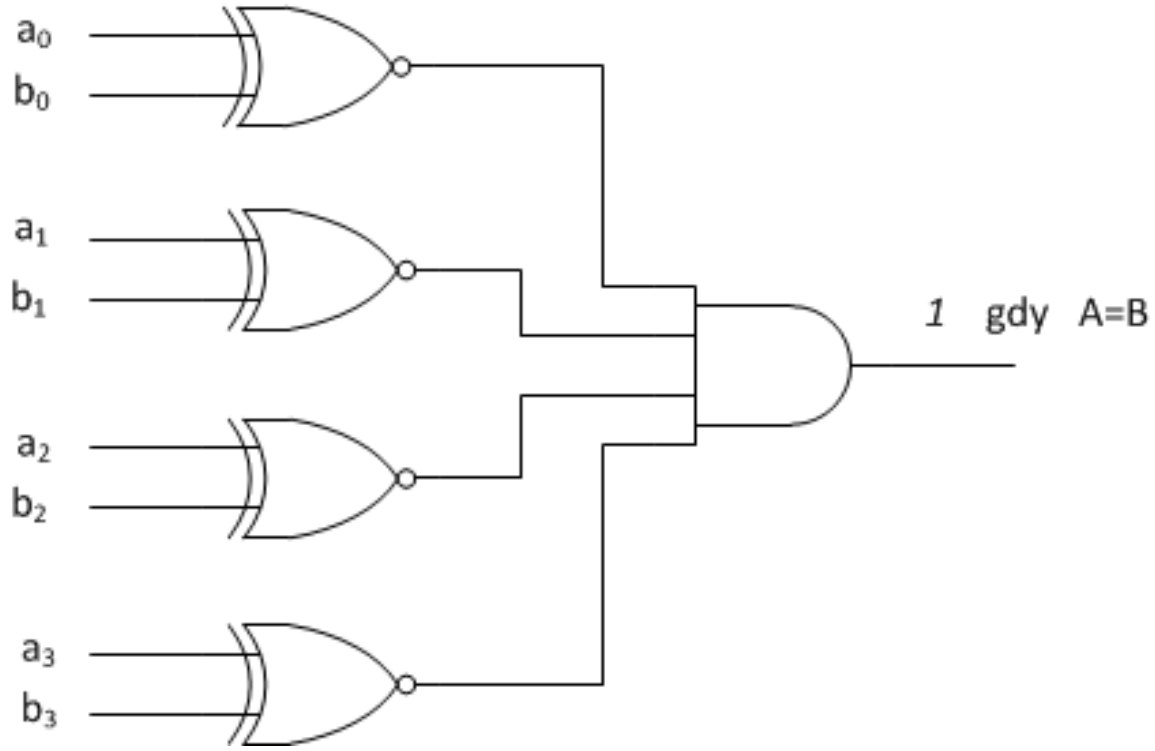
liczba binarna $A=A_1A_0$
liczba binarna $B=B_1B_0$



schemat komparatora identyczności liczb dwubitowych

komparator identyczności liczb czterobitowych

liczba A = $a_3a_2a_1a_0$
liczba B = $b_3b_2b_1b_0$



schemat komparatora identyczności liczb czterobitowych

$$\prod_{i=0}^{i=3} \overline{a_i \oplus b_i} = \begin{cases} 1 & gdy\ liczby\ są\ identyczne \\ 0 & gdy\ liczby\ są\ różne \end{cases}$$

przykładowe zastosowanie komparatora identyczności

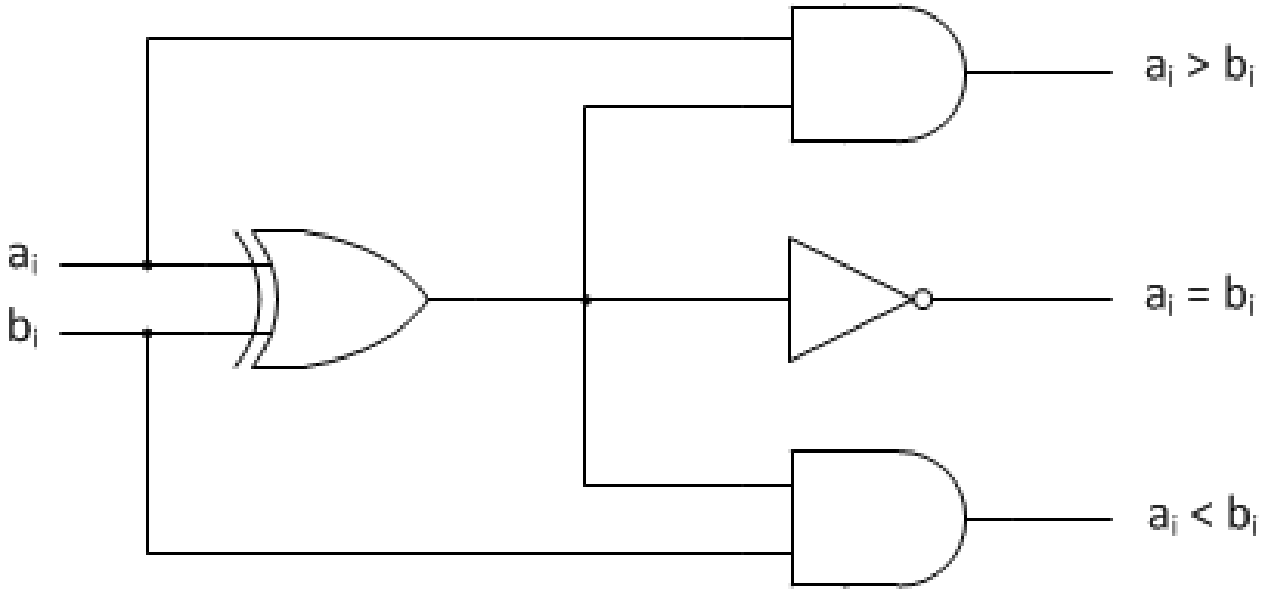
W komputerze pamięć podręczna to bardzo szybka pamięć pośrednicząca między jednostką centralną (CPU) a wolniejszą pamięcią główną.

CPU żąda danych, wysyłając swój adres (unikalną lokalizację) w pamięci.

Część tego adresu nazywana jest tagiem (znacznikiem). Komparator adresu porównuje tag z procesora ze tagiem z katalogu pamięci podręcznej. Jeśli oba są zgodne, zaadresowane dane są już w pamięci podręcznej i są bardzo szybko pobierane.

Jeśli tagi nie zgadzają się, dane muszą być pobierane z pamięci głównej w znacznie wolniejszym tempie.

jednobitowy komparator wielkości (pokazuje: A=B, A<B, A>B)



a_i	b_i	$a_i = b_i$	$a_i > b_i$	$a_i < b_i$
0	0	1	0	0
0	1	0	0	1
1	0	0	1	0
1	1	1	0	0

tabela prawdy komparatora

Aby porównać dwie liczby czterobitowe trzeba mieć cztery takie komparatory

$A = a_3a_2a_1a_0$
$B = b_3b_2b_1b_0$

Działanie komparatora wielkości w procesie porównywania kolejnych bitów dwóch liczb

Niech liczby A = $a_3a_2a_1a_0$ i B = $b_3b_2b_1b_0$ będą zapisane w naturalnym kodzie dwójkowym.

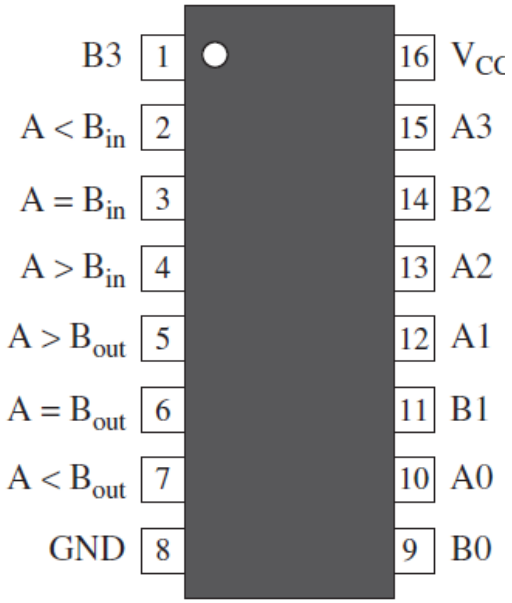
Porównuje się w pierwszej kolejności bity najbardziej znaczące (najstarsze, o największej wadze) i wtedy to, gdy:

$a_3 > b_3$ to już wiadomo, że A>B i następuje koniec procesu porównywania
$a_3 < b_3$ to już wiadomo, że A<B i następuje koniec procesu porównywania
$a_3 = b_3$ przechodzi się do porównywania kolejnych młodszych bitów
i wtedy to, gdy:
$a_2 > b_2$ to już wiadomo, że A>B i następuje koniec procesu porównywania
$a_2 < b_2$ to już wiadomo, że A<B i następuje koniec procesu porównywania
$a_2 = b_2$ przechodzi się do porównywania kolejnych młodszych bitów
i wtedy to, gdy:
$a_1 > b_1$ to już wiadomo, że A>B i następuje koniec procesu porównywania
$a_1 < b_1$ to już wiadomo, że A<B i następuje koniec procesu porównywania
$a_1 = b_1$ przechodzi się do porównania ostatnich najmłodszych bitów
i wtedy, gdy:
$a_0 > b_0$ to A>B
$a_0 < b_0$ to A<B
$a_0 = b_0$ to A=B **ostateczny koniec porównania**

realizacja w postaci scalonej

74HC85

czterobitowy komparator wielkości



układ w obudowie
z oznaczeniem kontaktów (pinów)

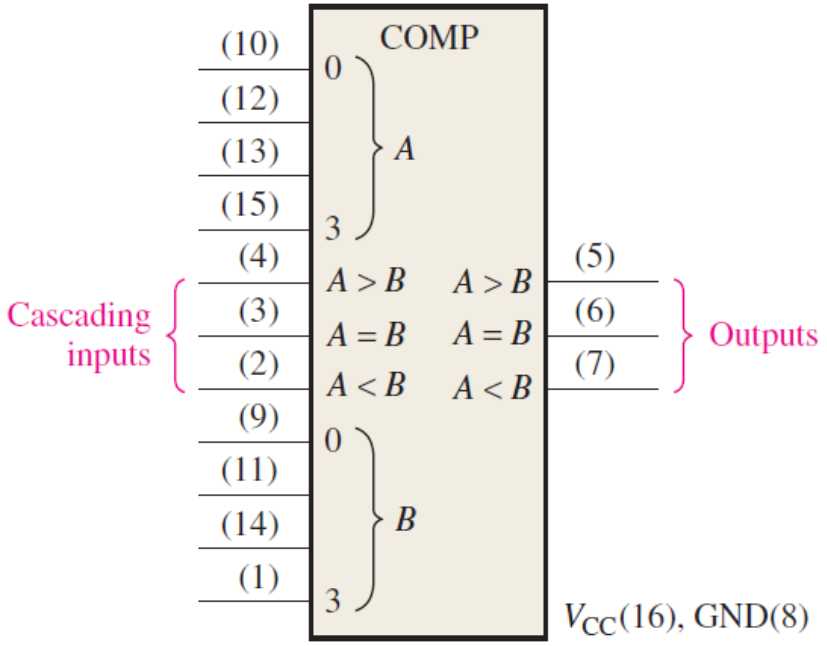


diagram logiczny układu

SUMATORY

Układy cyfrowe wykonujące operację sumowania logicznego licz binarnych

podstawowe zasady dodawania binarnego

$$0 + 0 = 0$$
$$0 + 1 = 1$$
$$1 + 0 = 1$$
$$1 + 1 = 10$$

przeniesienie *Carry*

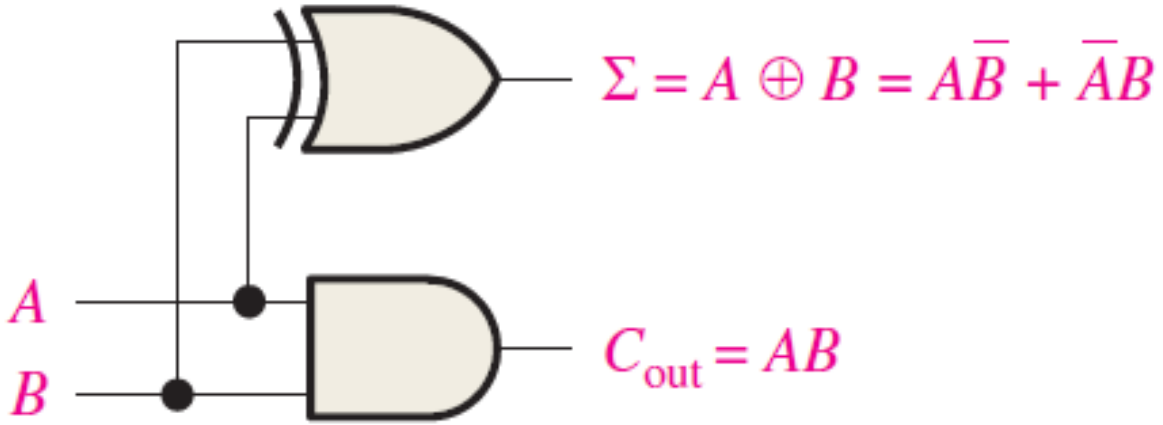
PÓŁSUMATOR

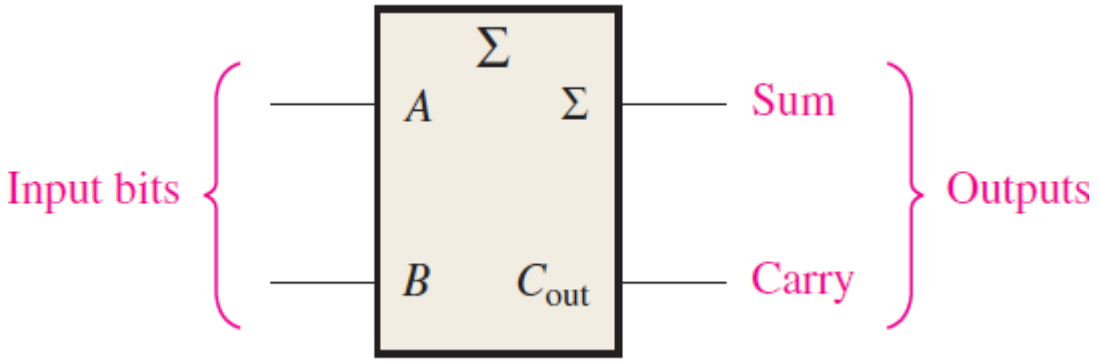
Układ, który dodaje dwa bity wejściowe

i wytwarza przeniesienie wyjściowe

$$0 + 0 = 0$$
$$0 + 1 = 1$$
$$1 + 0 = 1$$
$$1 + 1 = 10$$

PÓŁSUMATOR





logiczny symbol półsumatora

A	B	C_{out}	Σ
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

tabela prawdy półsumatora

[*]

SUMATOR

Układ, który dodaje dwa bity wejściowe i przeniesienie wejściowe i wytwarza przeniesienie wyjściowe

dodawanie dwóch bitów wejściowych i przeniesienia wejściowego

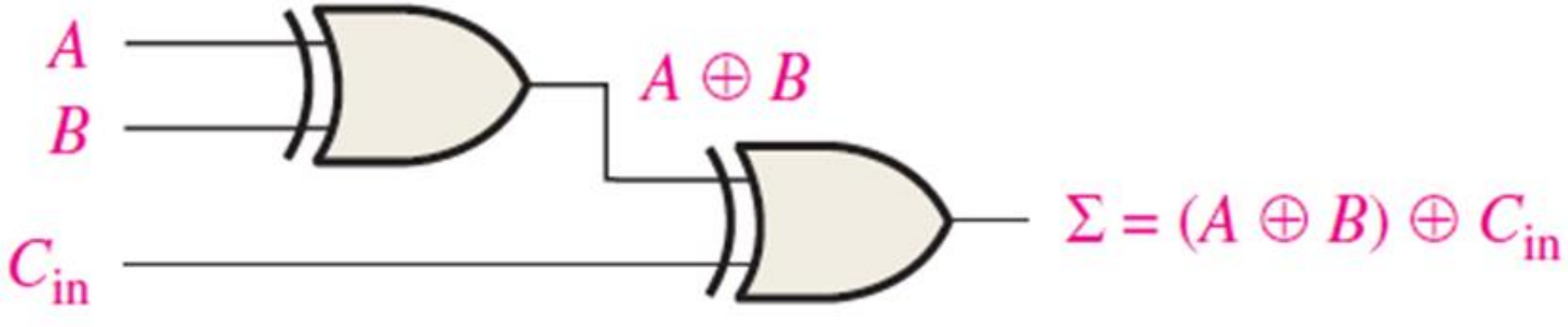


tabela prawdy

A	B	C_{in}	Σ
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

[*]

wytworzenie przeniesienia własnego

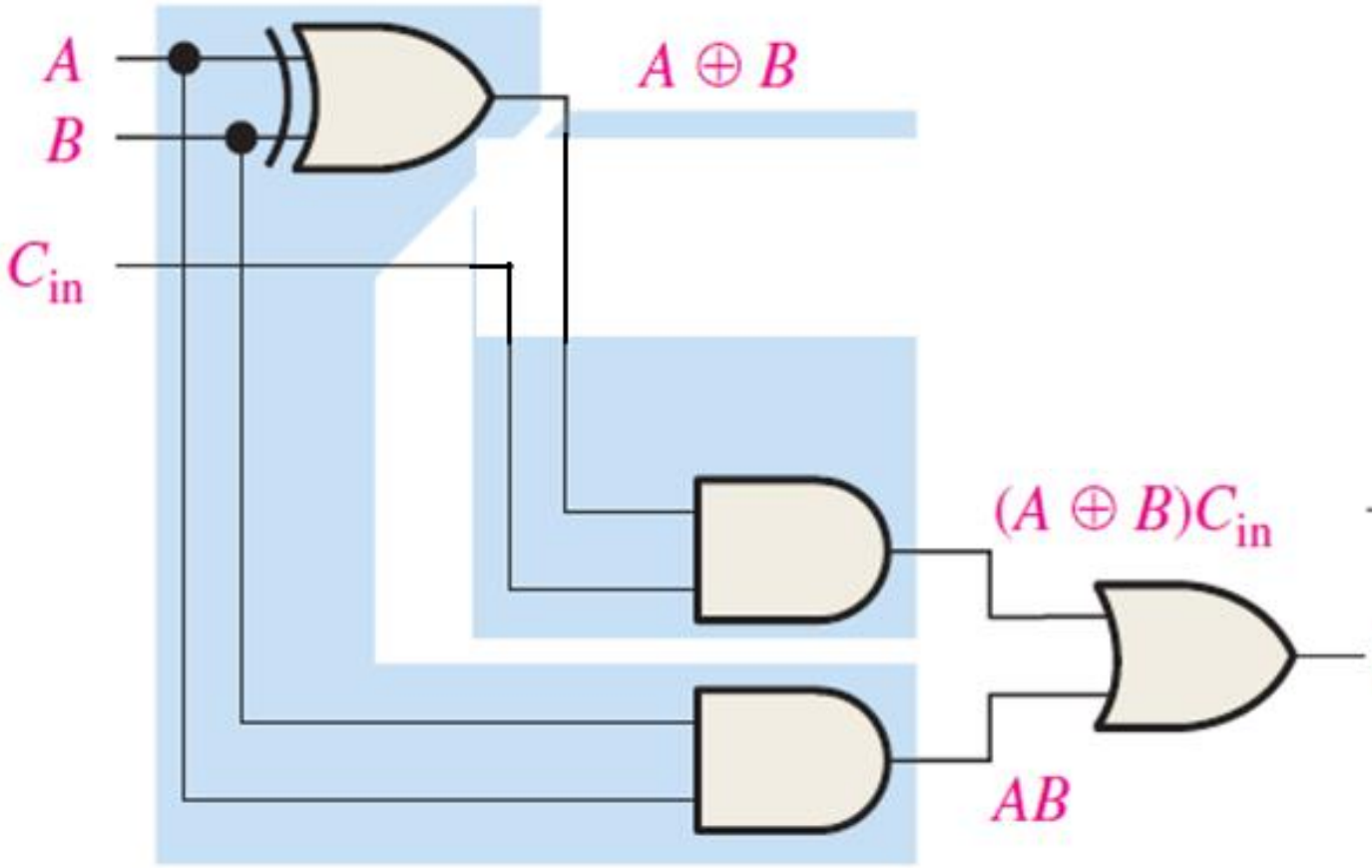


tabela prawdy

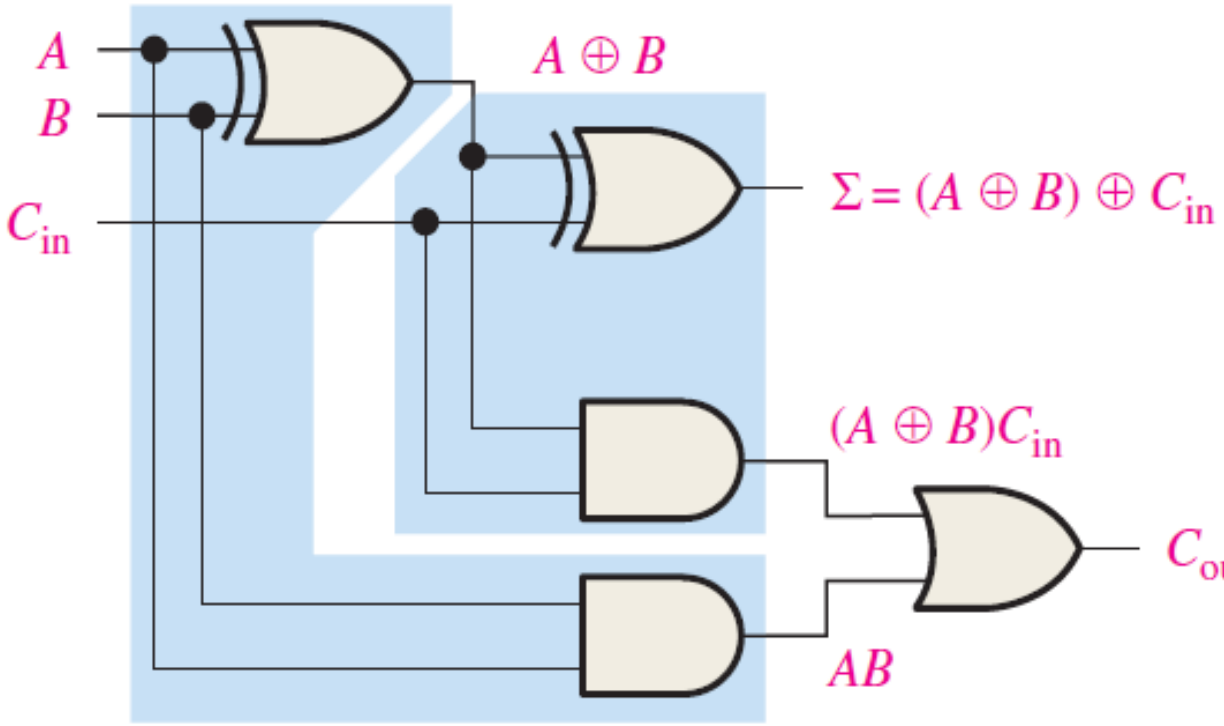
A	B	C_{in}	C_{out}
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

$$C_{out} = AB + (A \oplus B)C_{in}$$

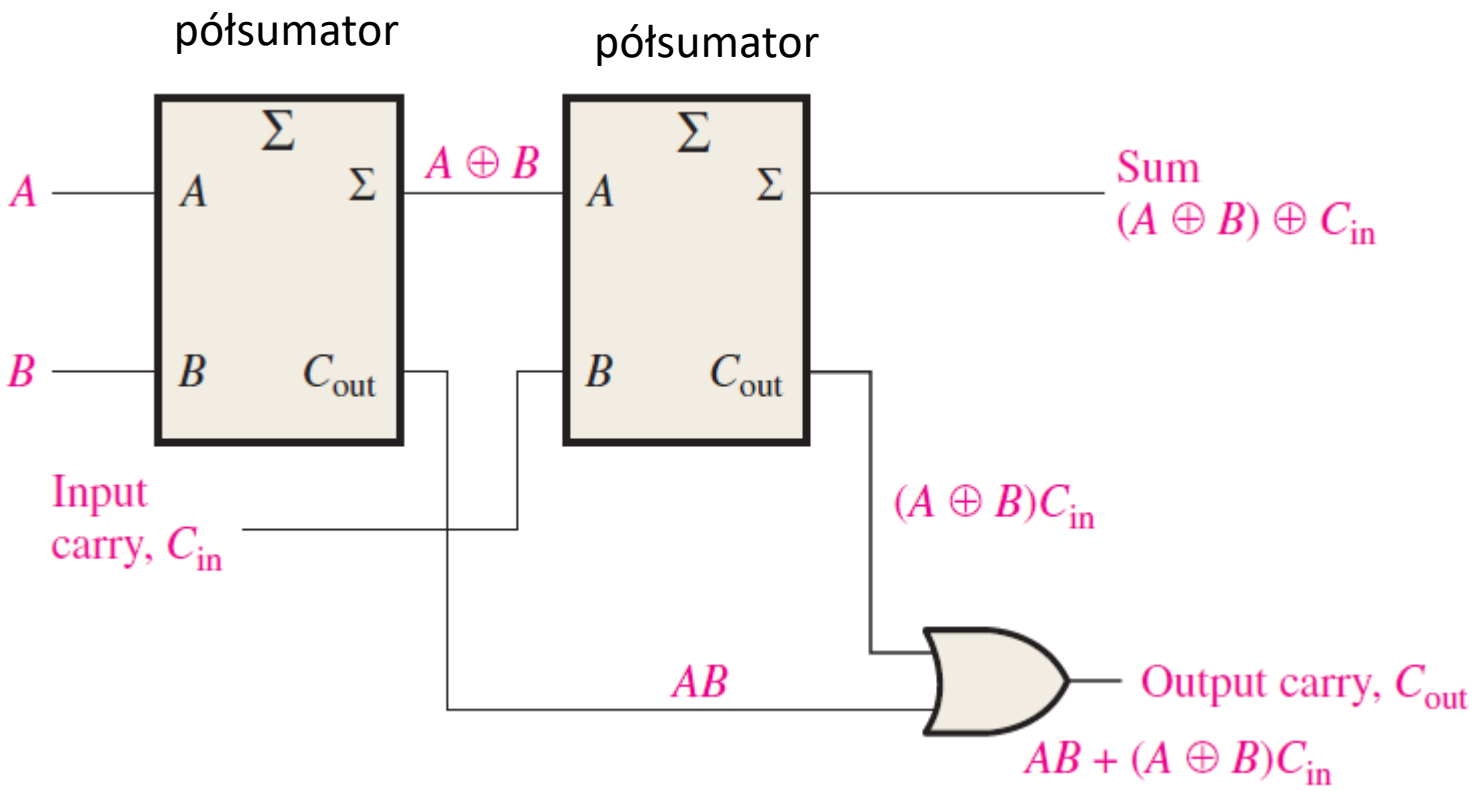
kompletny schemat sumatora

tabela prawdy

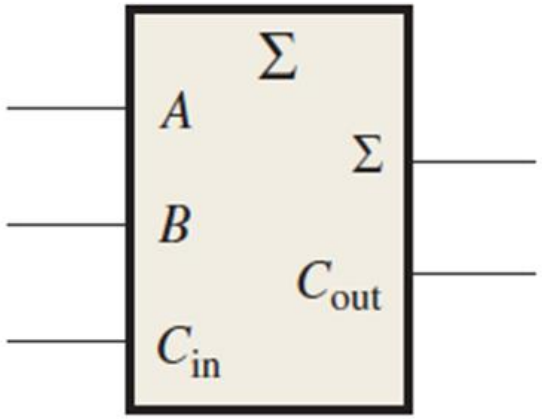
A	B	C_{in}	C_{out}	Σ
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1



[*]



dwa półsumatory tworzące sumator [*]



logiczny symbol sumatora

Równoległe sumatory binarne

Jeden pełny sumator jest w stanie dodać dwie liczby 1-bitowe i przeniesienie wejściowe. Aby dodać liczby binarne z więcej niż jednym bitem trzeba użyć dodatkowe sumatory.

Kiedy jedna liczba binarna jest dodawana do drugiej, każda kolumna generuje bit sumy i bit przenoszenia 1 lub 0 do następnej kolumny po lewej stronie.

Aby dodać dwie liczby binarne 2-bitowe potrzebne są dwa sumatory;
w przypadku liczb 4-bitowych używane są cztery sumatory; i tak dalej.

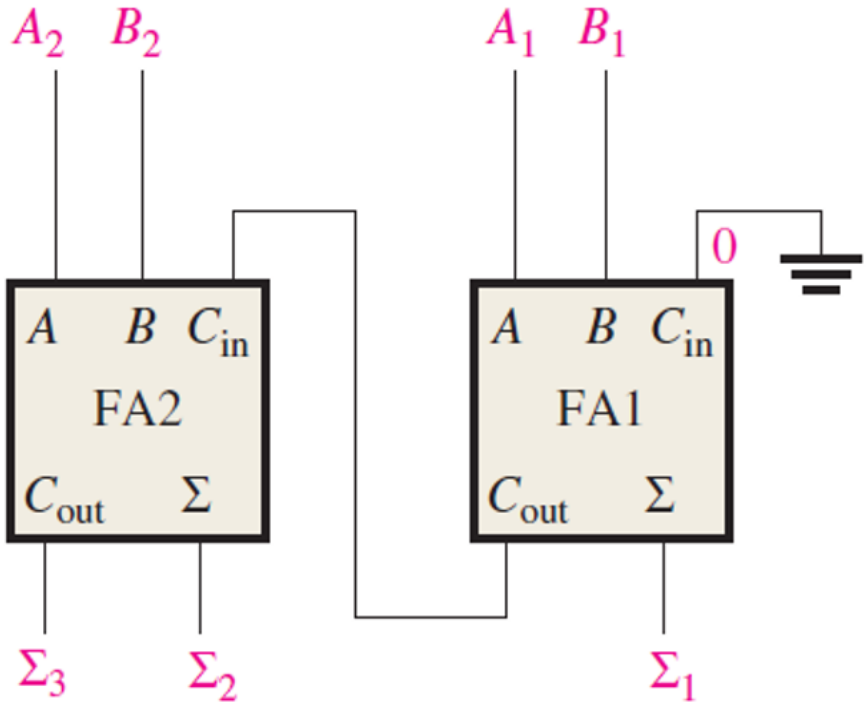
Wyjście przenoszenia każdego sumatora jest podłączone do wejścia przenoszenia następnego sumatora wyższego rzędu.

$$\begin{array}{r} A_2A_1 \\ +\,B_2B_1 \\ \hline \Sigma_3\Sigma_2\Sigma_1 \end{array}$$

przykładowe sumowanie:

$$\begin{array}{r} 1 \\ 11 \\ +\,01 \\ \hline 100 \end{array}$$

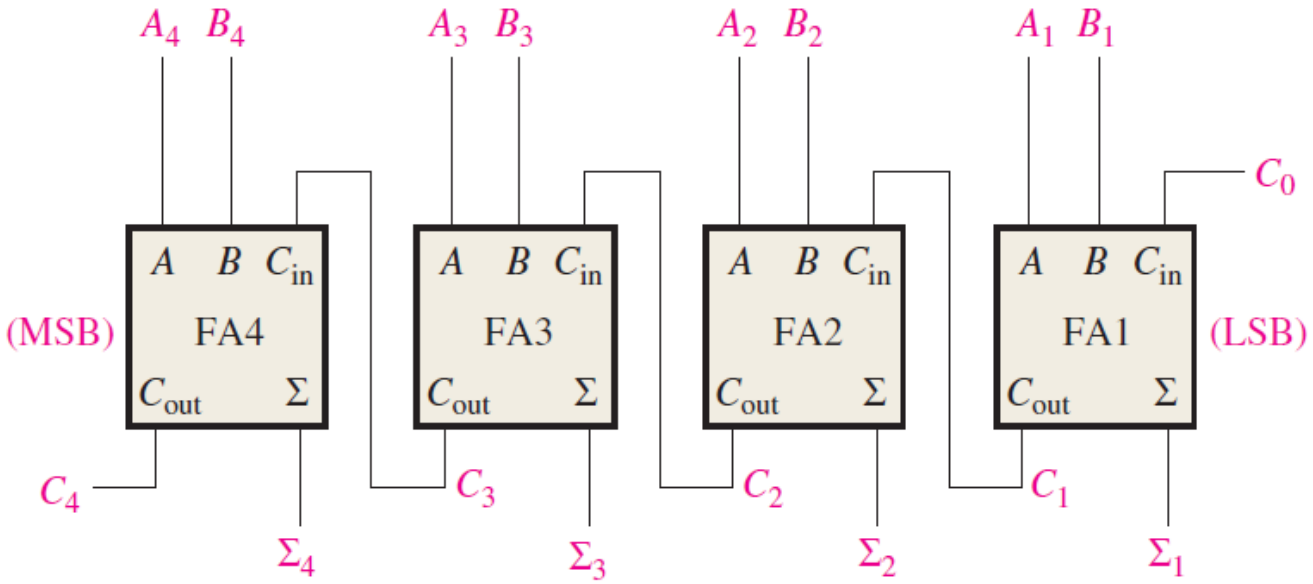
przeniesienie



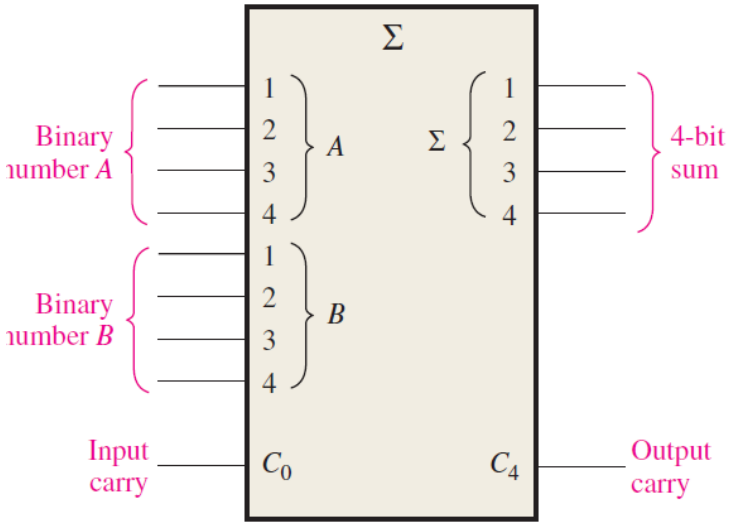
dwubitowy sumator równoległy

Wejście przeniesienia sumatora na najmniej znaczącej pozycji może być ustawione na 0 (uziemione), ponieważ nie ma przeniesienia do najmniej znaczącej pozycji bitu.

[*]



czterobitowy sumator równoległy



logiczny symbol sumatora

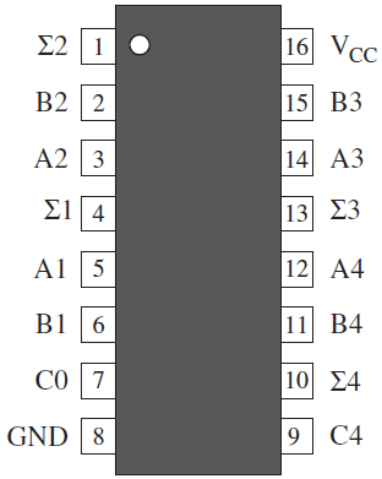
C_{n-1}	A_n	B_n	Σ_n	C_n
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

tabela prawdy

[*]

czterobitowy sumator równoległy

74HC283



układ w obudowie
z oznaczeniem kontaktów (pinów)

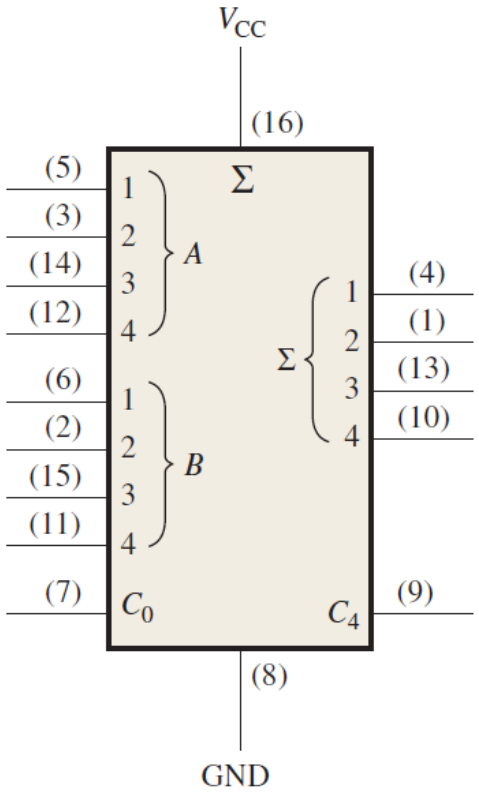
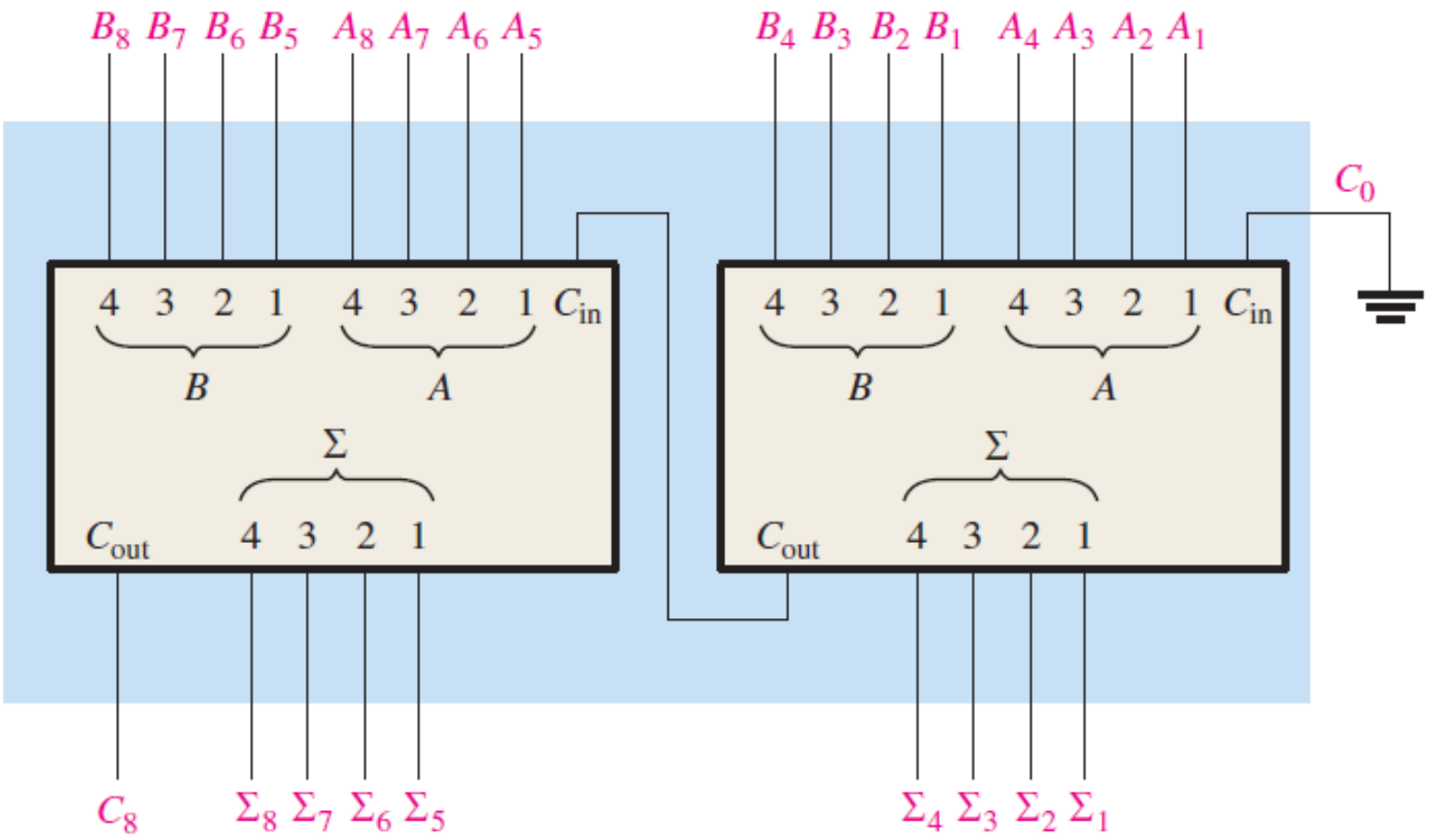


diagram logiczny układu

Rozszerzenia sumatorów

4-bitowy sumator równoległy można rozszerzyć w celu obsługi dodawania dwóch 8-bitowych liczb za pomocą dwóch 4-bitowych sumatorów.
Wejście przenoszenia sumatora niskiego rzędu (C0) jest połączone z masą, ponieważ nie ma przeniesienia do najmniej znaczącej pozycji bitu, a wyjście przenoszenia sumatora niskiego rzędu jest podłączone do wejścia przenoszenia sumatora wysokiego rzędu.
To połączenie nazywamy kaskadowym.
Przeniesienie wyjściowe jest oznaczone jako C8, ponieważ jest generowane z pozycji ósmego bitu.

Podobnie, cztery 4-bitowe sumatory można połączyć kaskadowo, aby dodać dwie 16-bitowe liczby itd.



Sumator 8-bitowy otrzymany poprzez kaskadowe połączenie dwóch 4-bitowych sumatorów [*]

UKŁADY KOMBINACYJNE

K O N I E C

[*] T.L.Floyd: Digital Fundamentals, PEARSON