WSTĘP DO ELEKTRONIKI

Część VII

Układy cyfrowe

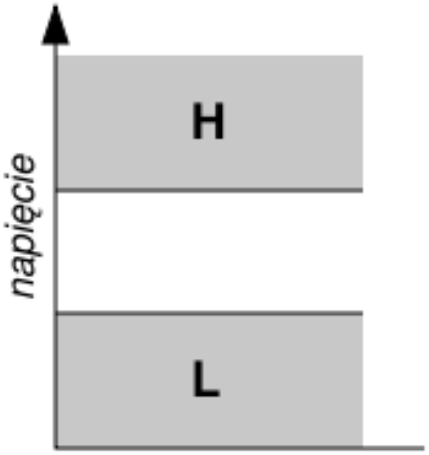
Janusz Brzychczyk IF UJ

Układy cyfrowe

W układach cyfrowych sygnały napięciowe (lub prądowe) przyjmują tylko określoną liczbę poziomów, którym przyporządkowywane są wartości liczbowe.

Najczęściej układy cyfrowe służą do przetwarzania sygnałów o dwóch poziomach napięć:
- wysokiego (**H** – high)
- niskiego (**L** – low).

Pracę takich układów cyfrowych (układów logicznych) opisuje się za pomocą dwuwartościowej algebry Boole`a, nazywanej logiką matematyczną.
Poziomom napięć **H** i **L** przyporządkowuje się wartości logiczne **1** (prawda) oraz **0** (fałsz).
Przyporządkowanie **H → 1** oraz **L → 0** nazywa się logiką dodatnią.
Przyporządkowanie **H → 0** oraz **L → 1** nazywa się logiką ujemną.



Ze względu na obecność zakłóceń, wahania napięcia zasilającego itp. sygnały w układach cyfrowych nie mają ściśle określonych wartości. Z tego powodu liczby przyporządkowuje się nie wartościom napięć, ale przedziałom napięć oddzielonych przerwami.
Jeżeli napięcie przyjmie wartość z zakresu przerwy to stan układu jest nieokreślony.

W standardzie TTL logika dodatnia:
H (1) - 2.0 - 5.0 V
L (0) - 0.0 - 0.8 V

Algebra Boole'a

Zmienne przyjmują dwie wartości:
1 - prawda (true),
0 – fałsz (false)

Podstawowe operacje na zmiennych **a** i **b**:

- Negacja: $\overline{\mathbf{a}} \equiv$ NOT **a**
- Iloczyn logiczny: $\mathbf{a} \cdot \mathbf{b} \equiv$ **a** AND **b**
- Suma logiczna: $\mathbf{a}+\mathbf{b} \equiv$ **a** OR **b**

Tablice prawdy:

$f = \bar{a}$

a	f
0	1
1	0

$f = a \cdot b$

a	b	f
0	0	0
0	1	0
1	0	0
1	1	1

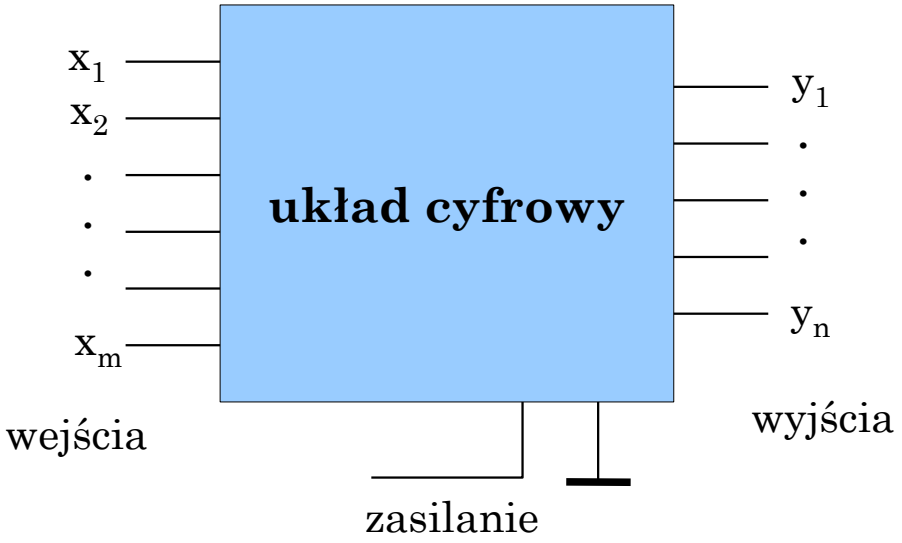
$f = a + b$

a	b	f
0	0	0
0	1	1
1	0	1
1	1	1

Podstawowe tożsamości algebry Boole'a

A* B = B*A	A+B = B+A	prawo przemienności
A*(B+C) = A*B + A*C	A+(B*C) = (A+B)*(A+C)	prawo rozdzielności
1 * A = A	0 + A = A	prawo tożsamości
$A * \overline{A} = 0$	$A + \overline{A} = 1$	prawo odwrotności
0 * A = 0	1 + A = 1	
A * A = A	A + A = A	
$\overline{A * B} = \overline{A} + \overline{B}$	$\overline{A + B} = \overline{A} * \overline{B}$	tw. de Morgana

Układy cyfrowe



Układ cyfrowy o m−wejściach i n−wyjściach.

W układach logicznych na każdym z wejść /wyjść może występować stan **0** lub **1** będący jednostką informacji zwaną **bitem**.
Wektory $\mathbf{x} = (x_1, x_2, \ldots, x_m)$, $\mathbf{y} = (y_1, y_2, \ldots, y_n)$ nazywamy **słowami logicznymi**.
Słowo ośmiobitowe nazywamy **bajtem**.
Każde słowo logiczne może być interpretowane jako pewna liczba zapisana w kodzie binarnym (dwójkowym). Na przykład czterobitowe słowo (1101) w kodzie dziesiętnym jest liczbą:

$$1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 13$$

Klasyfikacja układów cyfrowych

Układy cyfrowe

Układy kombinacyjne

Stan wyjść jest jednoznacznie określony przez stan wejść układu:

$$\mathbf{y} = f(\mathbf{x})$$

Układy sekwencyjne

Stan wyjść zależy od stanu wejść oraz od poprzednich stanów układu:

$$\mathbf{y}(t_n) = f[\mathbf{x}(t_0), \mathbf{x}(t_1), \ldots, \mathbf{x}(t_n)]$$

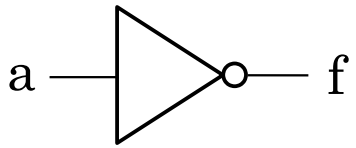
Układy kombinacyjne

- **Bramki logiczne**
- **Bloki funkcjonalne**
  - **komutatory (multipleksery, demultipleksery)**
  - **konwertery kodów (kodery, dekodery, transkodery)**
  - **bloki arytmetyczne (sumatory, komparatory, ...)**

Bramki logiczne (rodzaj, funkcja logiczna, symbol, tablica prawdy)

NOT (negacja):

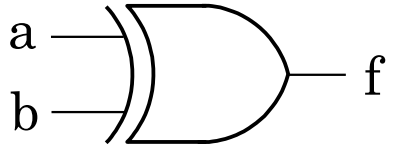
$$f = \bar{a}$$



a	f
0	1
1	0

EXOR (Exclusive OR, wyłączna suma logiczna):

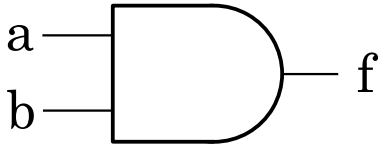
$$f = a \oplus b$$



a	b	f
0	0	0
0	1	1
1	0	1
1	1	0

AND (iloczyn):

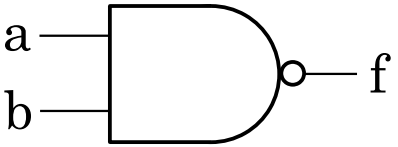
$$f = a \cdot b$$



a	b	f
0	0	0
0	1	0
1	0	0
1	1	1

NAND (NOT-AND, negacja iloczynu):

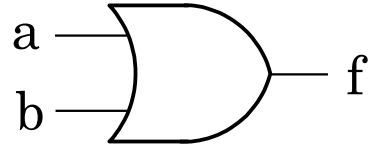
$$f = \overline{a \cdot b}$$



a	b	f
0	0	1
0	1	1
1	0	1
1	1	0

OR (suma):

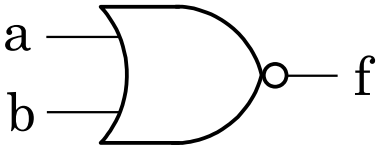
$$f = a + b$$



a	b	f
0	0	0
0	1	1
1	0	1
1	1	1

NOR (NOT-OR, negacja sumy):

$$f = \overline{a + b}$$



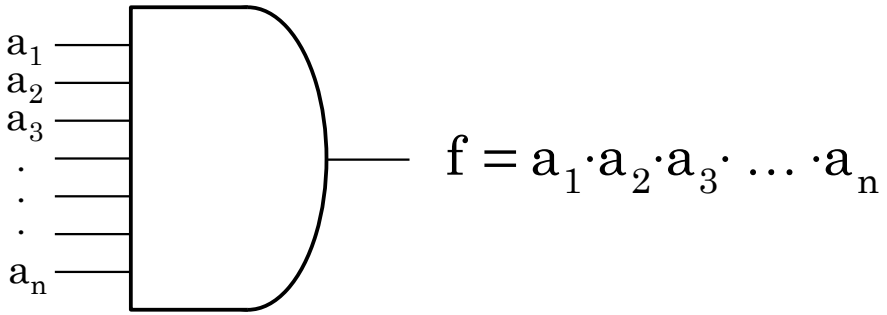
a	b	f
0	0	1
0	1	0
1	0	0
1	1	0

Bramki logiczne

Oprócz bramek dwuwejściowych stosowane są również bramki wielowejściowe.

Przykład:

Wielowejściowa bramka AND



Wartość logiczna **1** pojawia się na wyjściu jedynie wówczas, gdy stan logiczny wszystkich wejść wynosi **1**. W innych przypadkach **f = 0**.
Bramka taka bywa nazywana układem koincydencyjnym.

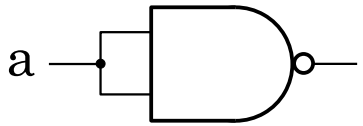
Bramki logiczne

Najbardziej uniwersalnymi bramkami są bramki NAND i NOR.

Używając tyko bramek NAND lub tylko bramek NOR można zbudować układ realizujący dowolną funkcję logiczną.

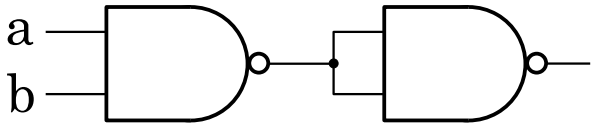
Przykłady realizacji podstawowych funkcji logicznych (NOT, AND, OR) przy użyciu bramek NAND:

NOT



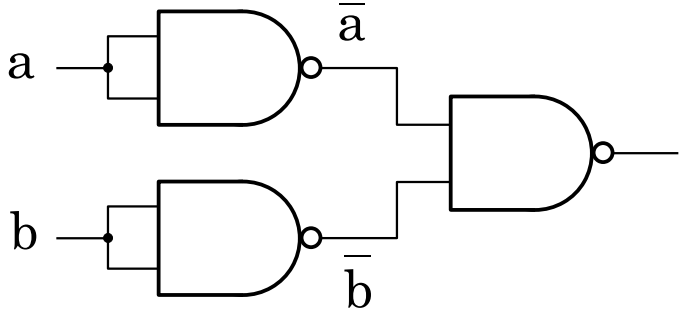
$$f = \overline{a \cdot a} = \overline{a}$$

AND



$$f = \overline{\overline{a \cdot b}} = a \cdot b$$

OR

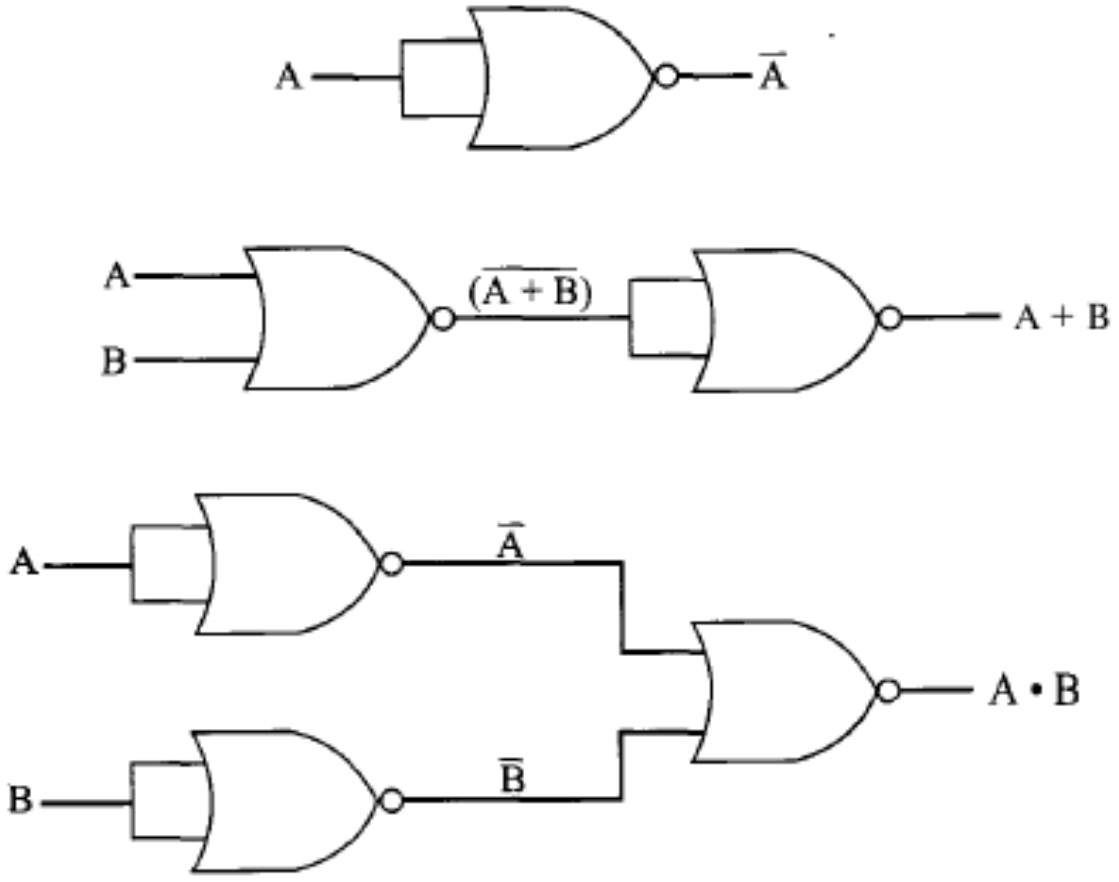


$$f = \overline{\overline{a} \cdot \overline{b}} = a + b$$

Korzystamy z prawa de Morgana:

$$\overline{a+b} = \overline{a} \cdot \overline{b}$$

Przykłady realizacji podstawowych funkcji logicznych (NOT, OR, AND) przy użyciu bramek NOR:



Układy kombinacyjne - przykład

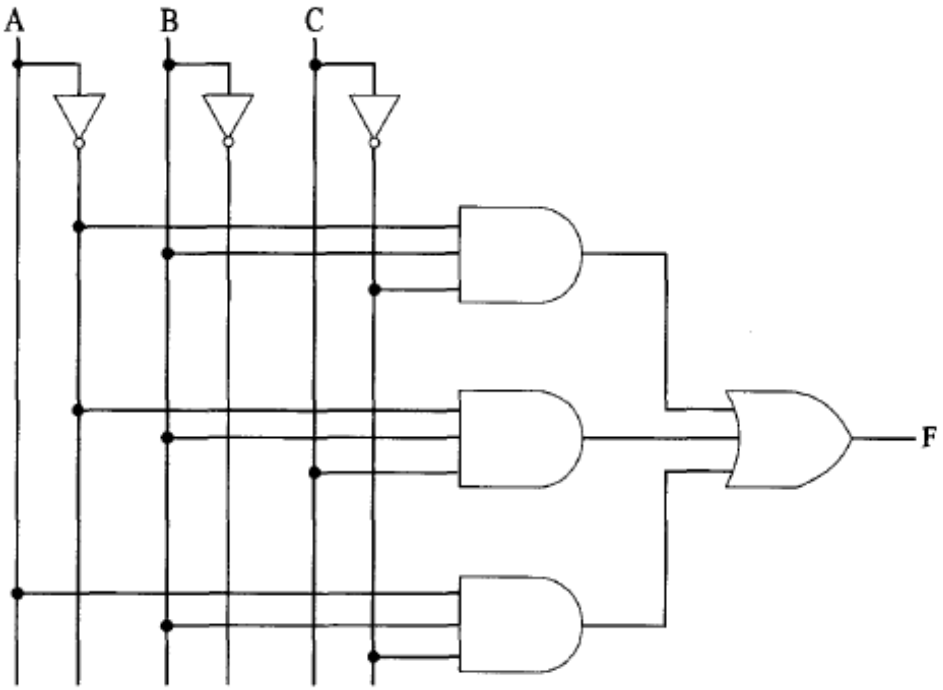
Sygnały wejściowe			Sygnał wyjściowy
A	B	C	F
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

Równanie Boole'a:

Można je wyrazić jako sumę kombinacji wartości zmiennych A, B, C, dla których F =1

$$F = \overline{A} \cdot B \cdot \overline{C} + \overline{A} \cdot B \cdot C + A \cdot B \cdot \overline{C}$$

Realizacja układu za pomocą bramek AND, OR i NOT:

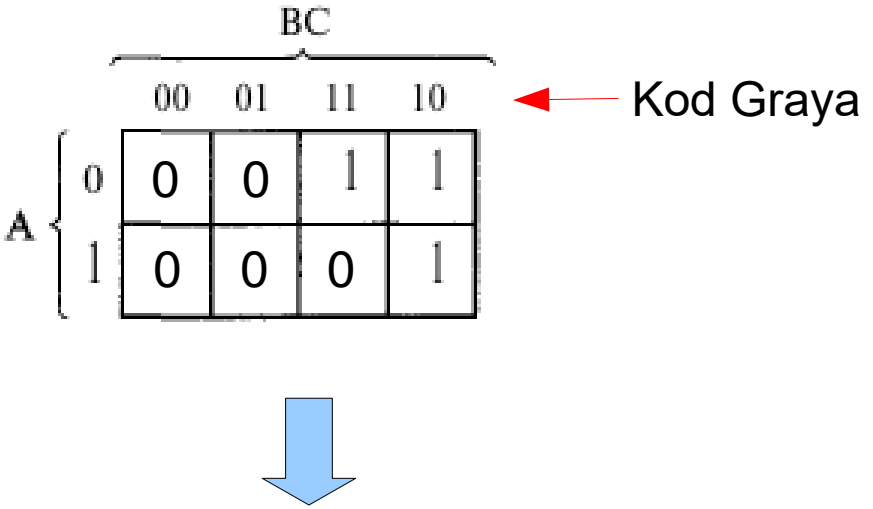


Tablica prawdy

Sygnały wejściowe			Sygnał wyjściowy
A	B	C	F
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

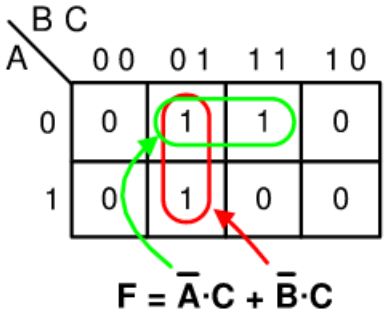
$$F = \overline{A} \cdot B \cdot \overline{C} + \overline{A} \cdot B \cdot C + A \cdot B \cdot \overline{C}$$

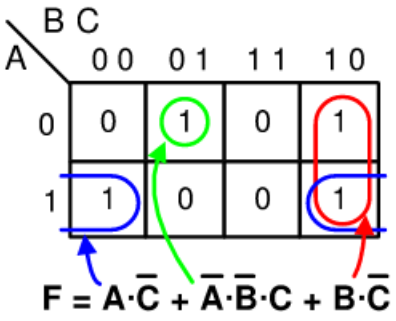
Mapa Karnaugha

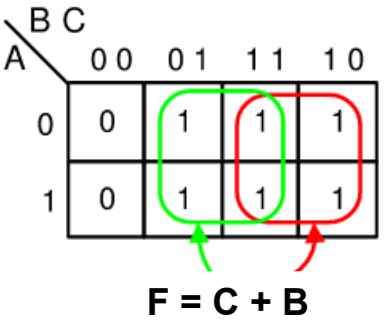


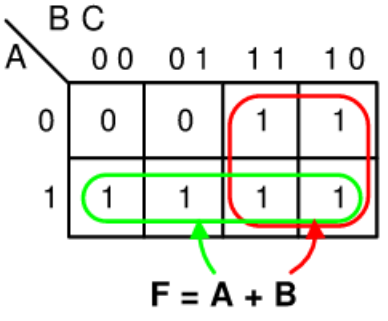
$$F = \overline{A} \cdot B + B \cdot \overline{C}$$

Przykłady minimalizacji formuł Boolowskich dla 3 zmiennych wejściowych (A,B,C) i jednej zmiennej wyjściowej (F) z wykorzystaniem map Karnaugha









Wykorzystywane są reguły sklejania:

$$A \cdot B + A \cdot \overline{B} = A \cdot (B + \overline{B}) = A$$

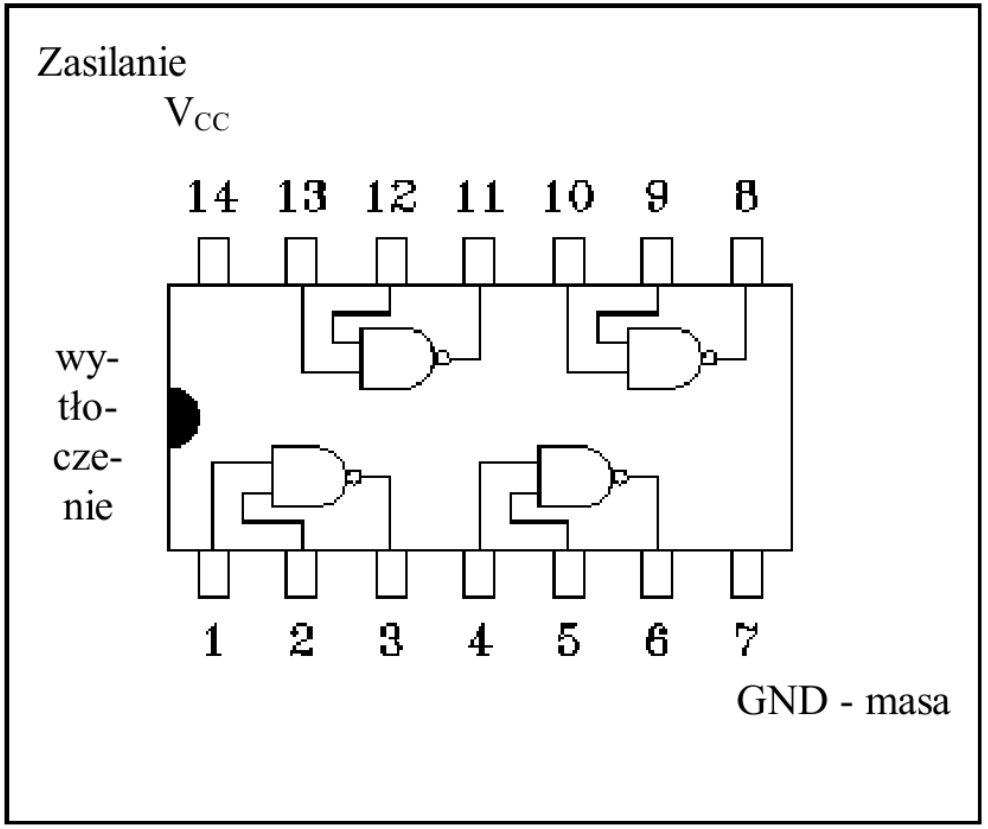
$$(A + B) \cdot (A + \overline{B}) = A$$

Bramki logiczne

Ze względu na stosowane technologie, bramki logiczne tworzą tzw. rodziny (np. TTL, ECL, CMOS). Jedną z najbardziej rozpowszechnionych jest rodzina bramek TTL (Transistor – Transistor Logic).

Charakterystyki układów TTL :

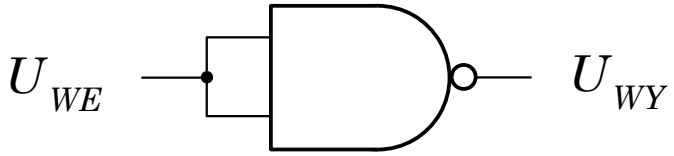
- układy pracują w logice dodatniej
- logicznemu zeru (L – stan niski) odpowiada napięcie z przedziału:
  0 – 0.8 V (sygnały wejściowe)
  0 – 0.4 V (sygnały wyjściowe)
- logicznej jedynce (H – stan wysoki) odpowiada napięcie z zakresu:
- 2.0 – 5 V (sygnały wejściowe)
- 2.7 – 5 V (sygnały wyjściowe)
- wejście bramki niepodłączone do niczego znajduje się w stanie logicznym **1**
- układy zasila się napięciem +5 V.



Schemat układu scalonego UCY7400 (TTL) zawierającego cztery bramki NAND

Bramki logiczne

Odpowiedź na wyjściu bramki nastepuje po pewnym, charakterystycznym dla danego układu czasie od momentu zmiany sygnałów wejściowych (czas propagacji sygnału przez bramkę).



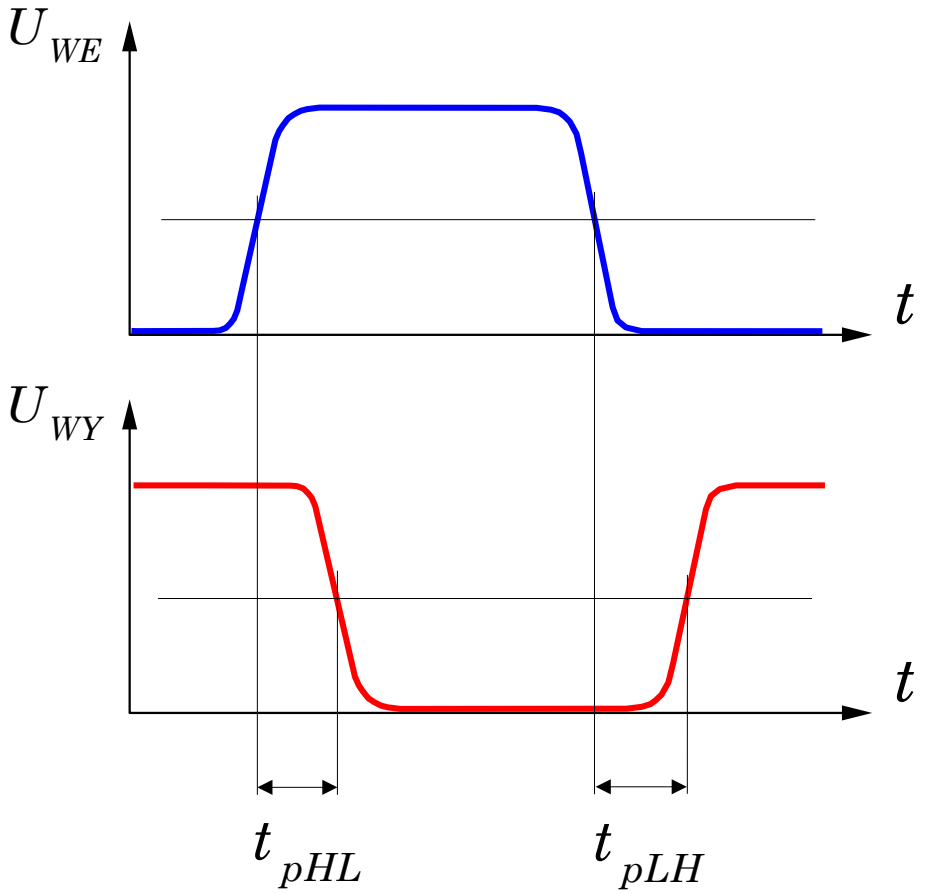
Średni czas propagacji:

$$t_p = \frac{t_{pHL} + t_{pLH}}{2}$$

Czasy przełączania bramki UCY7400:

$$t_{pHL} \simeq 15\ \text{ns}$$

$$t_{pLH} \simeq 22\ \text{ns}$$

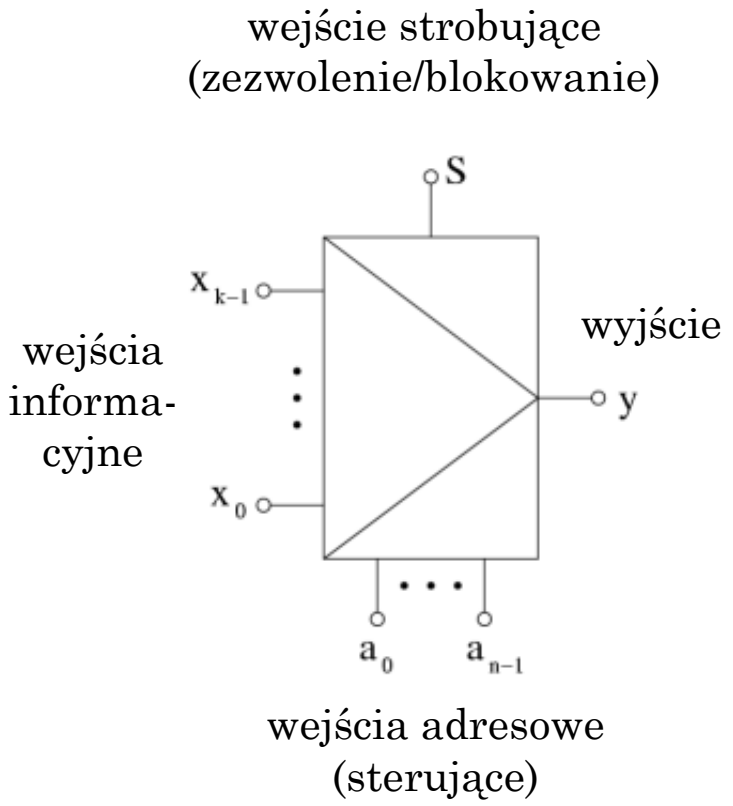


Multiplekser

Często w praktyce pomiarowej lub obliczeniowej informacja napływa równocześnie z wielu źródeł, a nas interesuje tylko przekaz informacji z jednego z nich. Stosujemy wówczas multiplekser.

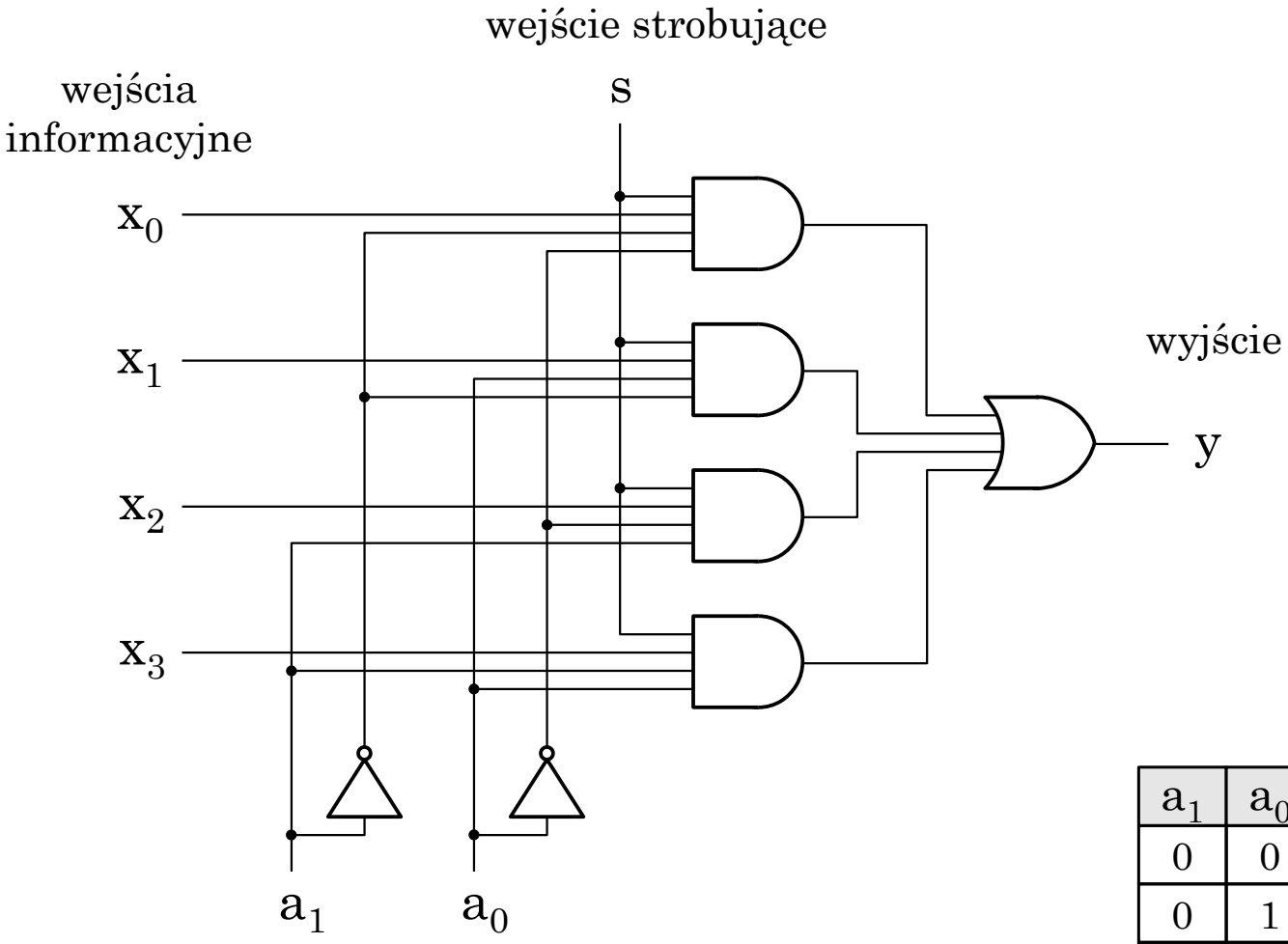
Multiplekser jest układem logicznym realizującym przepływ informacji tylko z jednego wejścia. Wybór wejścia jest określony przez podanie jego adresu (numeru) na wejścia adresowe.

Przepływ informacji z wejścia na wyjście jest możliwy dopiero wówczas, gdy wejście strobujące znajduje się w stanie logicznym **1**.



Symbol graficzny multipleksera.

Przykład multipleksera z czterokanałowym wejściem

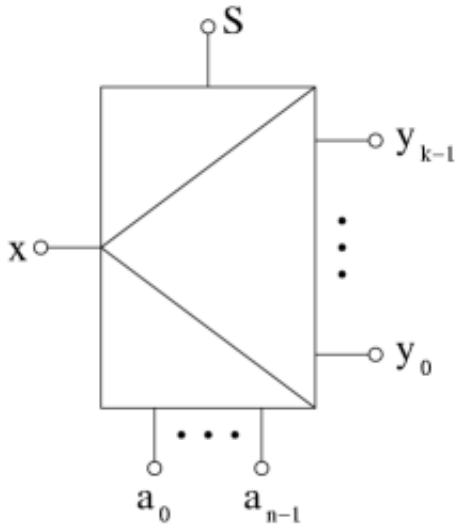


a_1	a_0	y
0	0	x_0
0	1	x_1
1	0	x_2
1	1	x_3

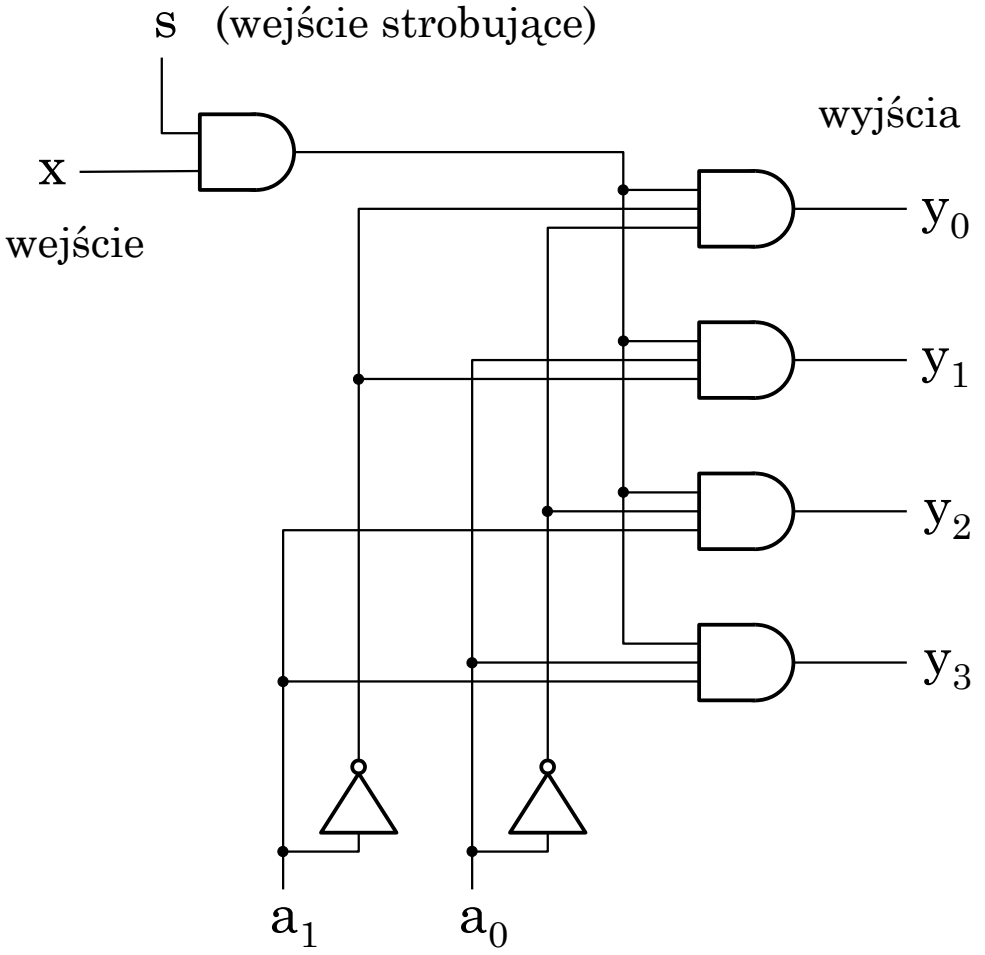
gdy s = 1

Demultiplekser

Demultiplekser przekazuje dane z jednego wejścia na selektywnie wybrane adresem wyjście.



Symbol graficzny demultipleksera



a_1	a_0	y_3	y_2	y_1	y_0
0	0	0	0	0	x
0	1	0	0	x	0
1	0	0	x	0	0
1	1	x	0	0	0

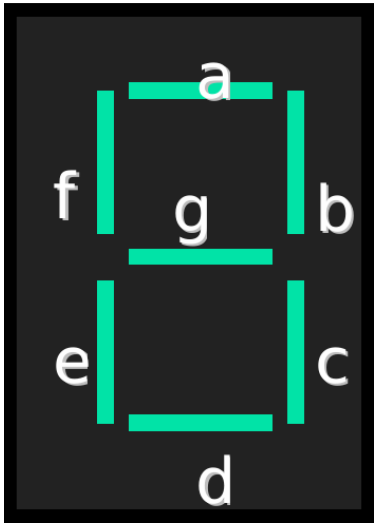
gdy s = 1

Przykład demultipleksera z czterema wyjściami

Kody, konwertery kodów

Każda informacja może być przedstawiona jako określona kombinacja bitów. Kombinacja bitów przypisana danej informacji jest nazywana **kodem**. Kodowanie umożliwia na przykład przedstawienie symboli cyfrowych, liter lub znaków w postaci binarnych słów logicznych.

Komunikacja z człowiekiem wymaga stosowania kodu wyświetlającego. W najprostszym przypadku jest to kod siedmiosegmentowego wyświetlacza cyfr. Cyfry kodujemy tak, aby w siedmiobitowym słowie binarnym każdy bit odpowiadał jednemu z segmentów.



CYFRA	a	b	c	d	e	f	g
0	1	1	1	1	1	1	0
1	0	1	1	0	0	0	0
2	1	1	0	1	1	0	1
3	1	1	1	1	0	0	1
4	0	1	1	0	0	1	1
5	1	0	1	1	0	1	1
6	1	0	1	1	1	1	1
7	1	1	1	0	0	0	0
8	1	1	1	1	1	1	1
9	1	1	1	1	0	1	1

Kody, konwertery kodów

Kod Graya (kod refleksyjny)

Wśród kodów stosowanych w pomiarach można wyróżnić kod Graya. Główną zaletą tego kodu jest to, że przy przejściu do następnej kombinacji zmienia się tylko jeden bit.

Kodem Graya długości **n** (n – bitowym) jest ciąg wszystkich 2^n różnych ciągów n cyfr {0,1}, ustawionych tak, że dwa kolejne ciągi różnią sie tylko na jednej pozycji. Ostatni i pierwszy wyraz tego kodu także spełnia tę zasadę (kod cykliczny).

2 – bitowy
00
01
11
10

3 – bitowy
000
001
011
010
110
111
101
100

Kody, konwertery kodów

Kod „1 z N” (kod pierścieniowy)

W kodzie tym tylko jeden z bitów przyjmuje wartość 1 (pozostałe bity 0).
Umożliwia on na przykład wprowadzanie z klawiatury cyfr
(naciskamy tylko jeden klawisz).

„1 z 4”

0001
0010
0100
1000

„1 z 8”

00000001
00000010
00000100
00001000
00010000
00100000
01000000
10000000

1 – sygnał aktywny

Kody, konwertery kodów

Konwersja pomiędzy kodami:

- liczbowym binarnym (kod naturalny binarny)
- liczbowym dziesiętnym
- binarnym Graya
- binarnym „1 z N”.

(16 elementów)

naturalny binarny	dziesiętny	Graya	Kod „1 z 16”
0000	0	0000	0000000000000001
0001	1	0001	0000000000000010
0010	2	0011	0000000000000100
0011	3	0010	0000000000001000
0100	4	0110	0000000000010000
0101	5	0111	0000000000100000
0110	6	0101	0000000001000000
0111	7	0100	0000000010000000
1000	8	1100	0000000100000000
1001	9	1101	0000001000000000
1010	10	1111	0000010000000000
1011	11	1110	0000100000000000
1100	12	1010	0001000000000000
1101	13	1011	0010000000000000
1110	14	1001	0100000000000000
1111	15	1000	1000000000000000

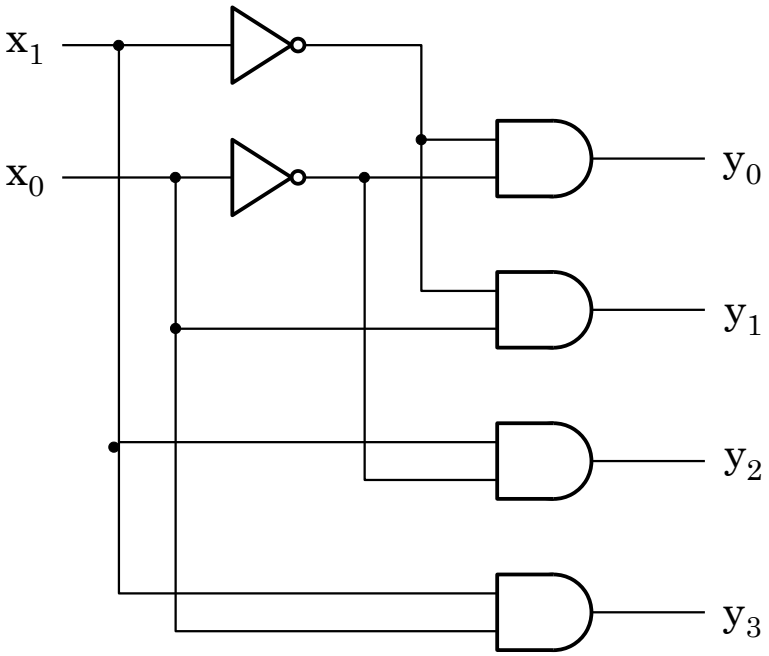
Konwertery kodów

Koderem (enkoderem) nazywamy układ cyfrowy, który przekształca kod „1 z N” na określony kod wyjściowy. Sygnał aktywny (1) pojawiający się na jednym z N wejść zostaje zakodowany w odpowiednie słowo M−bitowe (M wyjść kodera).

Dekoderem nazywamy układ, który przekształca określony kod wejściowy na kod wyjściowy „1 z N”. Dekoder ma więc N wyjść, przy czym każdemu za słów wejściowych jest przyporządkowany sygnał aktywny pojawiający się tylko na jednym z N wyjść.

Transkoderem (translatorem) nazywamy układ realizujący konwersję dwóch dowolnych kodów z których żaden nie jest kodem „1 z N”.

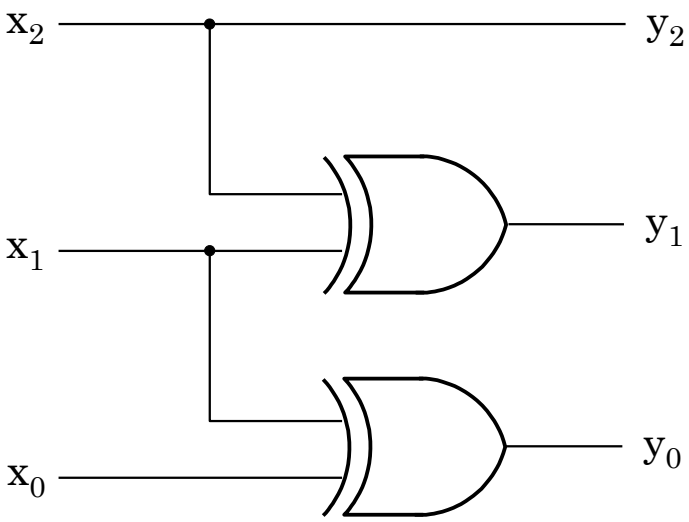
Przykład dekodera



x_1	x_0	y_3	y_2	y_1	y_0
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0

Dekoder realizujący konwersję dwubitowego naturalnego kodu binarnego na kod „1 z 4”.

Przykład transkodera



x_2	x_1	x_0	y_2	y_1	y_0
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	1
0	1	1	0	1	0
1	0	0	1	1	0
1	0	1	1	1	1
1	1	0	1	0	1
1	1	1	1	0	0

Transkoder realizujący konwersję naturalnego kodu binarnego na kod Graya (kody 3 - bitowe).

Układy arytmetyczne

Cyfrowe układy arytmetyczne realizują operacje arytmetyczne na liczbach przedstawionych w zapisie binarnym.

Podstawowym układem arytmetycznym jest układ realizujący dodawanie, nazywany **sumatorem**. Wszystkie inne operacje arytmetyczne (odejmowanie, mnożenie, dzielenie ...) wykonać można za pomocą tylko operacji dodawania stosując odpowiednie algorytmy.

Do układów arytmetycznych zalicza się także układy służące do porównywania dwóch liczb, nazywane **komparatorami** oraz układy wielofunkcyjne, wykonujące różne operacje arytmetyczne i logiczne tzw. **jednostki arytmetyczno-logiczne (ALU)**.

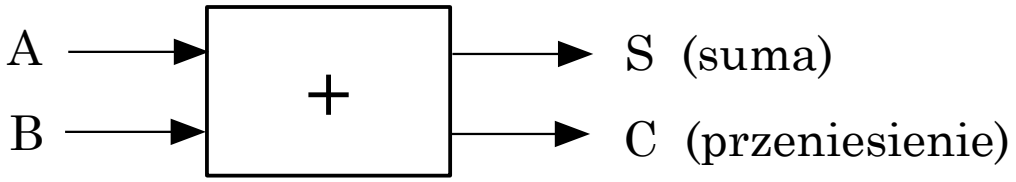
Sumatory

Półsumator

Układ wykonujący dodawanie dwóch jednobitowych liczb binarnych A i B:

0 + 0 = 0 = 00
0 + 1 = 1 = 01
1 + 0 = 1 = 01
1 + 1 = 2 = 10

A B CS



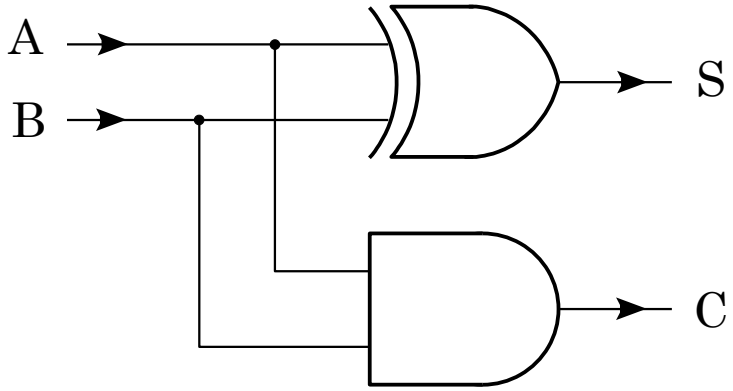
Przedstawienie wyniku wymaga użycia dwóch bitów. Młodszy bit wyniku (bit na mniej znaczącej pozycji) wyprowadza się na wyjście S, starszy bit wyniku na wyjście C.

B	A	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Tabela stanów

$S = A \oplus B$

$C = A \bullet B$

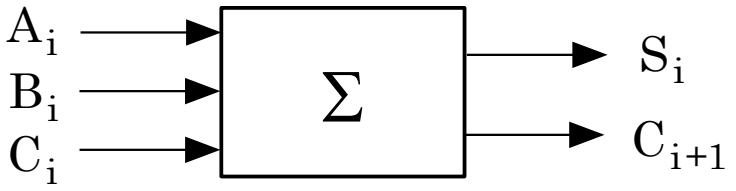


Sposób realizacji półsumatora

Sumatory

Sumator (pełny sumator)

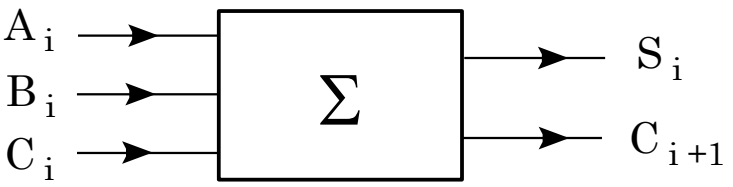
W przypadku dodawania liczb o większej liczbie bitów półsumator można zastosować tylko na najmłodszej pozycji. Na wszystkich pozostałych trzeba dodawać nie dwa, ale trzy bity uwzględniając przeniesienie z poprzedniej pozycji. Potrzebny jest więc układ o trzech wejściach A_i, B_i, C_i oraz dwóch wyjściach S_i, C_{i+1} . Układ taki nazywamy **pełnym sumatorem**.

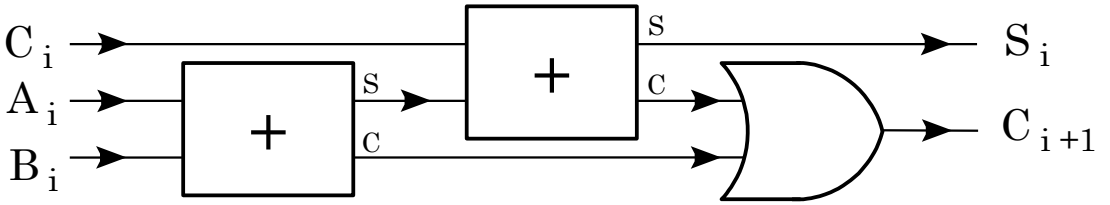


Przykład sumowania dwóch liczb binarnych:

A =	$A_4\ A_3\ A_2\ A_1\ A_0$	⇒	0	1	1	1	1	$(15)_{10}$
B =	$B_4\ B_3\ B_2\ B_1\ B_0$	⇒	0	1	0	1	0	$(10)_{10}$
przeniesienia	$C_4\ C_3\ C_2\ C_1\ C_0$	⇒	1	1	1	0	0	
suma S =	$S_4\ S_3\ S_2\ S_1\ S_0$	⇒	1	1	0	0	1	$(25)_{10}$

Pełny sumator (jednobitowy)





Schemat sumatora zbudowanego z półsumatorów

B_i	A_i	C_i	C_{i+1}	S_i
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

Tablica stanów sumatora

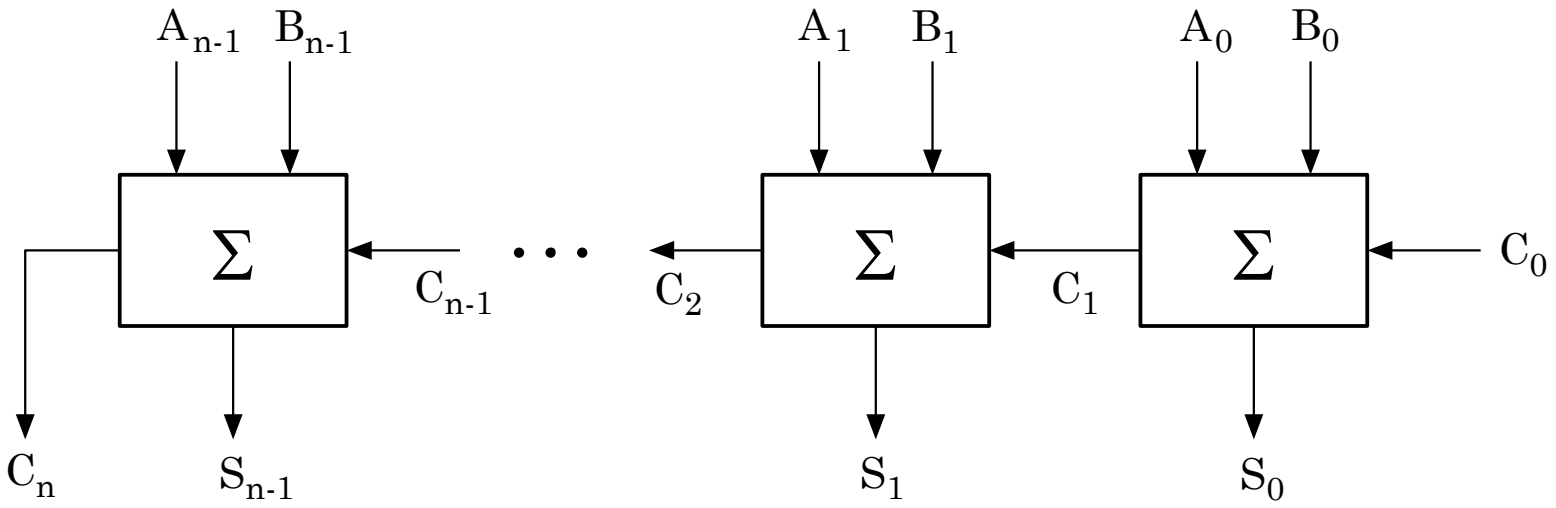
$$S_i = A_i \oplus B_i \oplus C_i$$

$$C_{i+1} = A_i \bullet B_i + A_i \bullet C_i + B_i \bullet C_i$$

Funkcje logiczne sumatora

Sumator wielobitowy

W celu dodawania liczb wielobitowych sumatory jednobitowe łączy się w zespoły. W najprostszym przypadku sumatory łączy sie szeregowo (wyjście przeniesienia łączy się z wejściem przeniesienia bloku następnego).

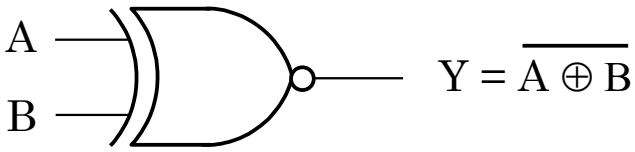


Sumator n–bitowy z przeniesieniami szeregowymi

Komparatory

Komparatorem cyfrowym nazywamy układ służący do porównywania dwu lub więcej liczb binarnych. Najważniejsze kryteria porównawcze to A = B, A > B, A < B. Układ sprawdzający wszystkie trzy relacje nazywa się komparatorem uniwersalnym. Najprostsze komparatory umożliwiają jedynie określenie czy dwie porównywane liczby są sobie równe lub która z liczb jest większa.

Kryterium równości dwóch liczb binarnych jest identyczność wszystkich bitów.
W przypadku dwóch liczb jednobitowych A i B, informację o tym uzyskać można za pomocą funkcji **negacja EXOR**:



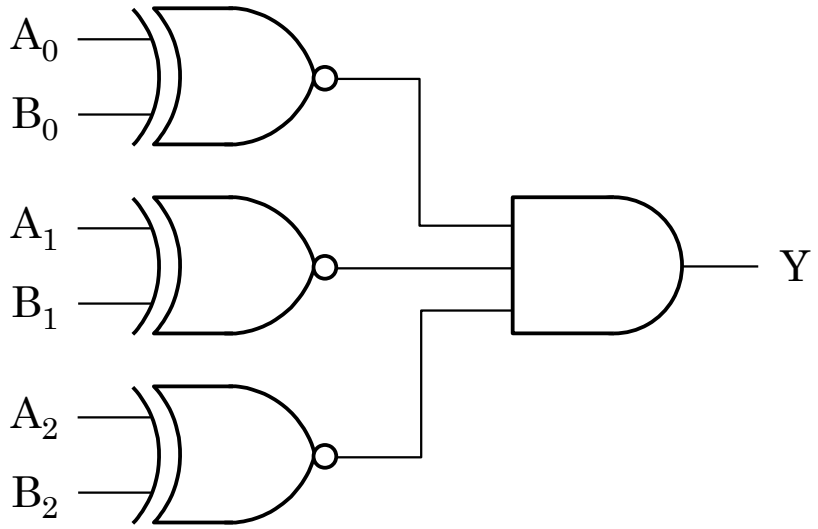
A	B	Y
0	0	1
0	1	0
1	0	0
1	1	1

Wartość 1 na wyjściu sygnalizuje równość A = B.

Komparatory

Przykład komparatora równoległego 3 – bitowego

Komparator równoległy to taki układ, na którego wejścia podawane są jednocześnie wszystkie bity porównywanych liczb.

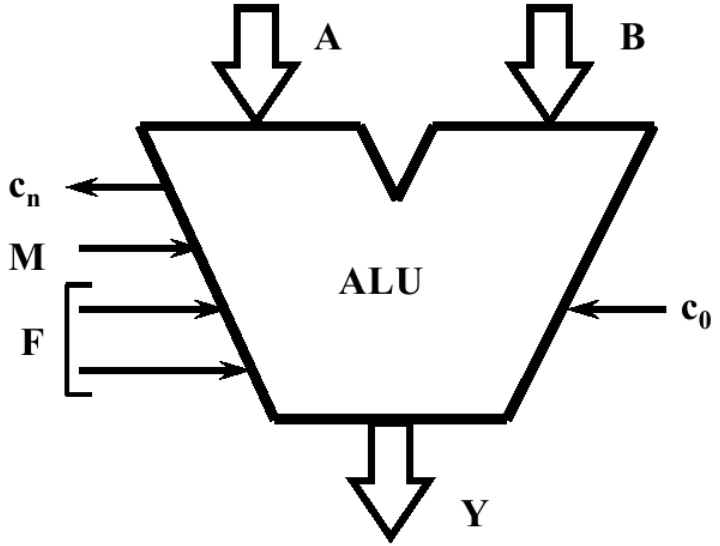


Y = 1 tylko wówczas gdy:
$A_0 = B_0$ i $A_1 = B_1$ i $A_2 = B_2$
czyli A = B.

Jednostki arytmetyczno – logiczne (ALU)

ALU jest uniwersalnym układem cyfrowym przeznaczonym do wykonywania operacji arytmetycznych i logicznych pomiędzy dwoma liczbami binarnymi.

Przykład:



M	F	Y
0	0 0	$A + c_0$
0	0 1	$A + B + c_0$
0	1 0	$A - B + (c_0 - 1)$
0	1 1	$A + (c_0 - 1)$
1	0 0	$A + B$
1	0 1	$A \oplus B$
1	1 0	$A \cdot B$
1	1 1	$\overline{A}$

Układy sekwencyjne

W układzie sekwencyjnym stan wyjść nie tylko zależy od stanu wejść ale także od poprzedniego stanu.
Układy sekwencyjne dzielimy na:

- układy asynchroniczne
- układy synchroniczne.

W układach asynchronicznych sygnały wejściowe bezpośrednio oddziaływują na stan wyjść. W układach synchronicznych zmiana sygnału wyjściowego może nastąpić wyłącznie w określonych chwilach czasu, które wyznacza sygnał zegarowy (*clock*), nazywany też sygnałem taktującym lub wyzwalającym.

Podstawowymi elementami układów sekwencyjnych są przerzutniki.

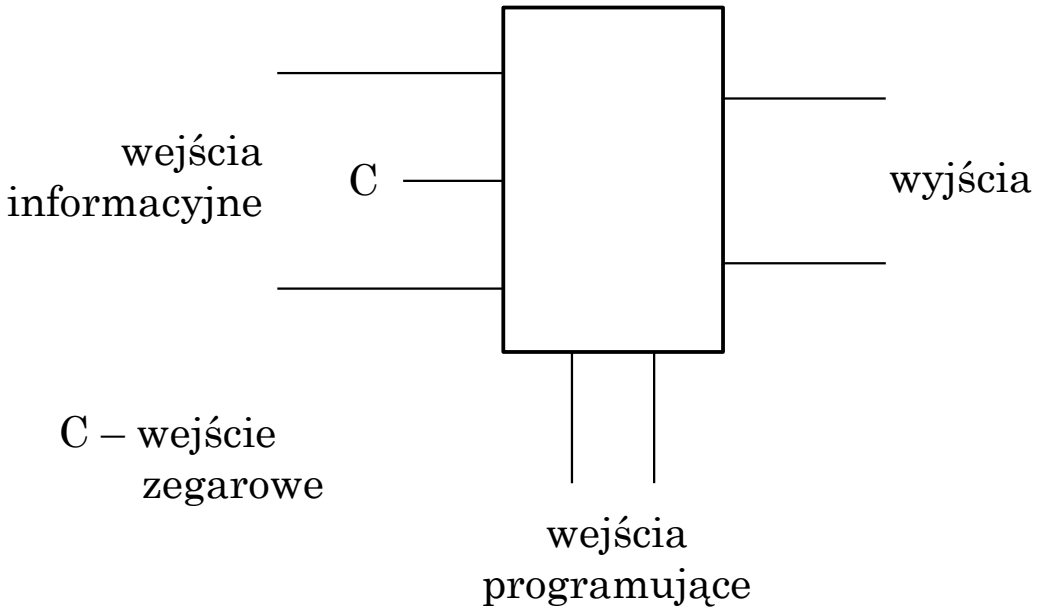
Przerzutniki

Zasadniczym zadaniem przerzutnika jest pamiętanie jednego bitu informacji.
Przerzutnik posiada co najmniej dwa wejścia i zazwyczaj dwa wyjścia.

Rozróżnia się następujące rodzaje wejść przerzutnika:
- informacyjne
- zegarowe
- programujące.

Podstawowe typy przerzutników:

- **RS**
- **JK**
- **D**
- **T**



Asynchroniczny przerzutnik RS

Jest najprostszym przerzutnikiem. Posiada:

- dwa wejścia
  - S (Set) – wejście ustawiające
  - R (Reset) – wejście zerujące
- dwa wyjścia
  - Q – wyjście zwykłe (główne)
  - $\overline{Q}$ – wyjście zanegowane (komplementarne).

Stan wyjść jest zawsze przeciwny.

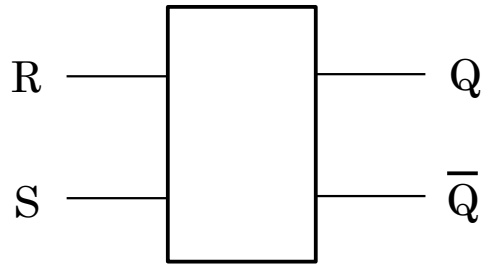
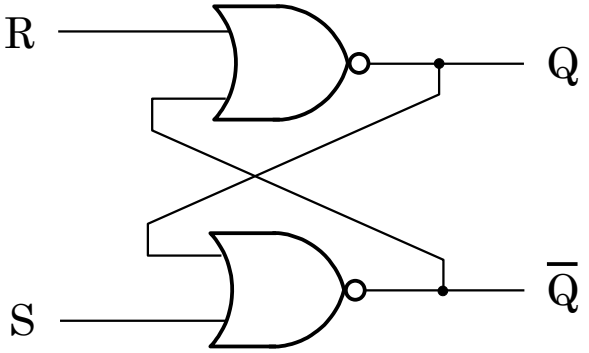


Tabela stanów

R	S	Q
0	0	stan pamiętania
1	0	0
0	1	1
1	1	stan niedozwolony

Przerzutnik RS można zbudować z dwóch bramek NOR lub dwóch bramek NAND stosując dodatnie sprzężenie zwrotne:

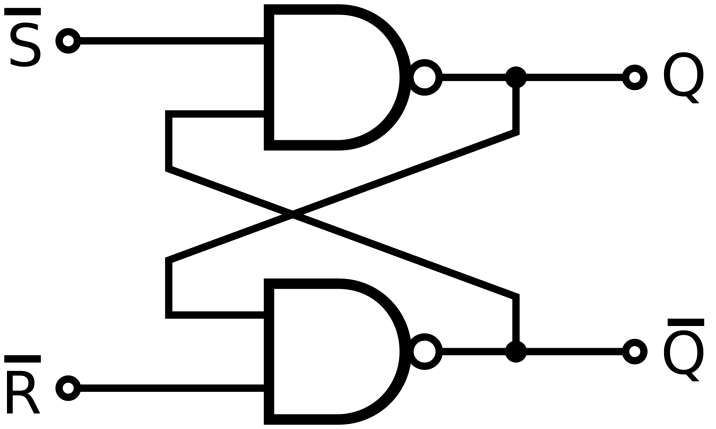


Przerzutnik można ustawić w stan zero (Q =0) przez podanie sygnału 1 na wejście R przy S =0.
Ustawienie przerzutnika w stan jeden (Q =1) realizuje się przez podanie sygnału 1 na wejście S przy R utrzymywanym w stanie 0.
Po przywróceniu stanu R = 0 i S = 0 przerzutnik wprowadzony zostaje w stan pamiętania, przechowując ustawiony stan. W ten sposób w przerzutniku zapamiętuje się elementarną porcję (1 – bit) informacji.

Asynchroniczny przerzutnik RS zbudowany z bramek NAND

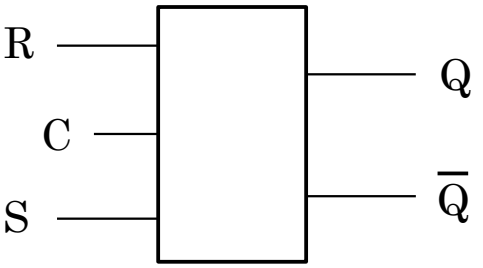
Tabela stanów

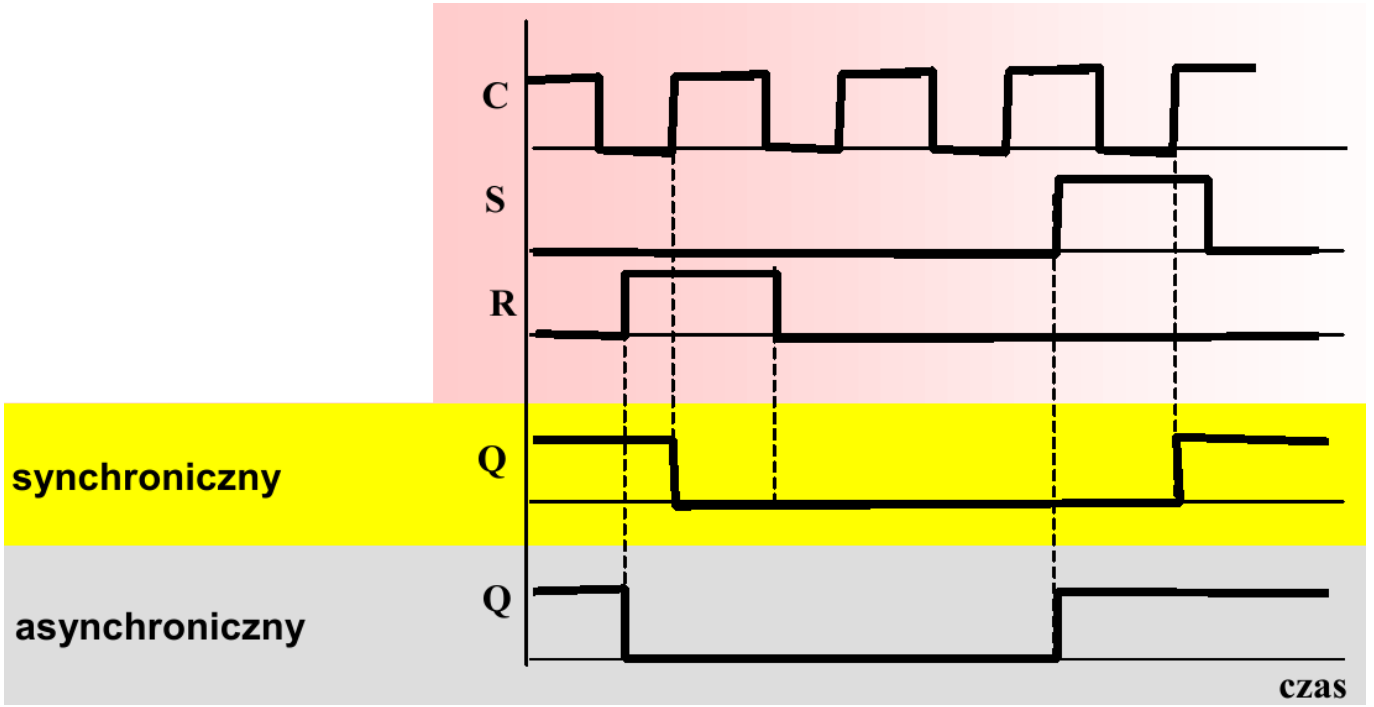
$\overline{S}$	$\overline{R}$	Q
0	0	stan zabroniony
0	1	1
1	0	0
1	1	stan pamiętania



Synchroniczny przerzutnik RS (wyzwalany zboczem)

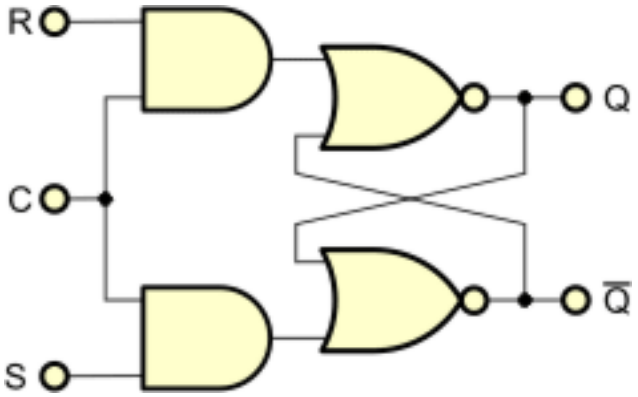
Przerzutnik synchroniczny RS ma dodatkowe wejście C do którego doprowadza się sygnał taktujący (zegarowy, synchronizujący). Zmiana stanu przerzutnika następuje w chwilach wyznaczonych przez sygnał taktujący. Umożliwia to wstępne przygotowanie sygnałów wejściowych i inicjację zmiany stanu przerzutnika po ustaleniu się tych stanów. Wyzwalanie zmiany stanu przerzutnika może następować w chwili gdy np. sygnał taktujący zmienia się ze stanu 0 na 1.



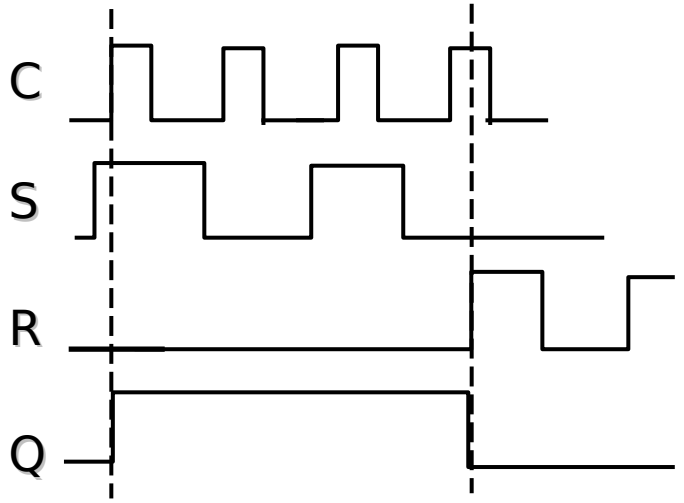


Synchroniczny przerzutnik RS
(wyzwalany poziomem / zatrzaskowy)

Schemat logiczny



Przebiegi czasowe



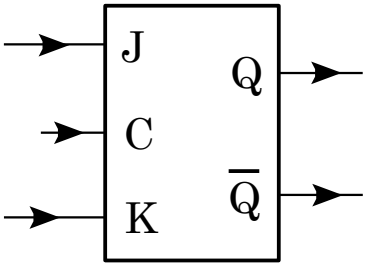
Dla C = 1

R	S	Q
0	0	stan pamiętania
1	0	0
0	1	1
1	1	stan niedozwolony

Dla C = 0
stan pamiętania

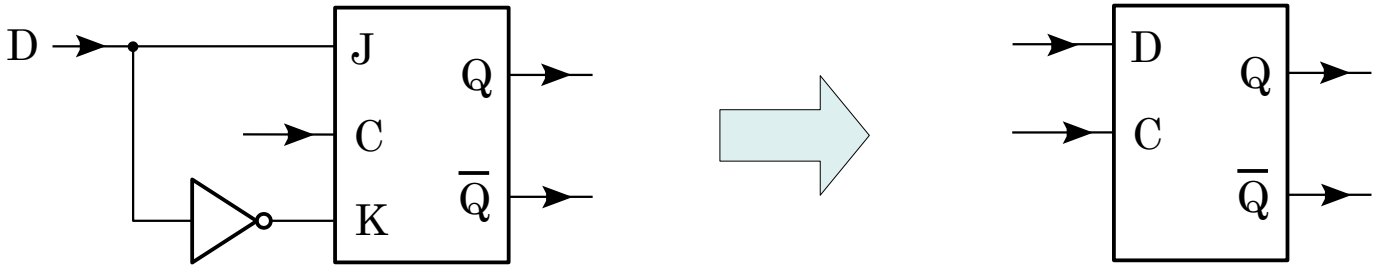
Przerzutnik JK (synchroniczny)

Wejścia informacyjne J i K odpowiadają wejściom S i R przerzutnika RS.
Przerzutnik JK nie ma stanów wejściowych niedozwolonych. W przypadku jednoczesnego podania sygnałów 1 na wejścia J i K, stan przerzutnika zmieni się na przeciwny (w chwili wyzwolenia sygnałem taktującym).



J	K	Q
0	0	stan się nie zmienia
1	0	1
0	1	0
1	1	zmiana stanu na przeciwny

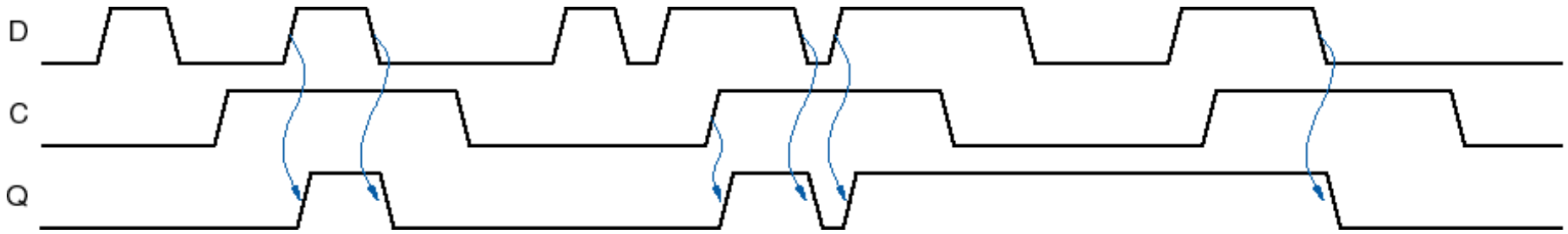
Przerzutnik D



D	J	K	Q
~~0~~	~~0~~	~~0~~	~~stan się nie zmienia~~
1	1	0	1
0	0	1	0
~~1~~	~~1~~	~~1~~	~~zmiana stanu na przeciwny~~

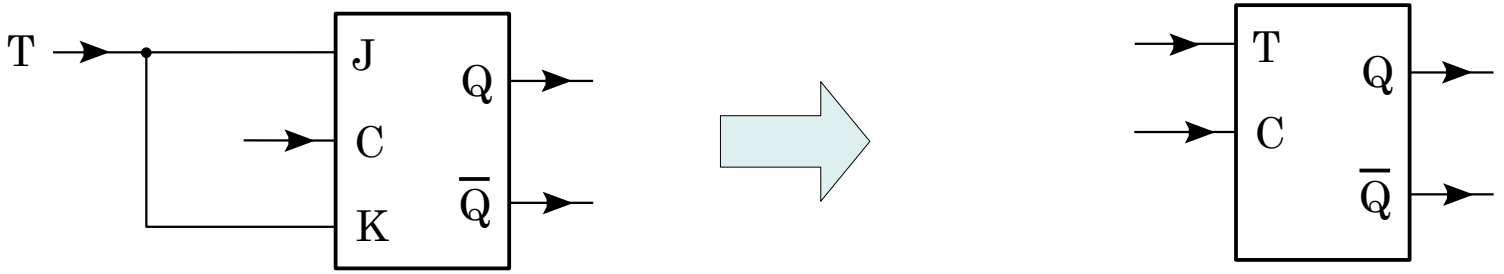
Przerzutnik D zapamiętuje stan wejścia D w chwili impulsu zegara.

Przerzutnik D – latch
(typu „zatrzask”)



Przykład przebiegów czasowych przerzutnika D–latch wyzwalanego stanem wysokim C

Przerzutnik T



T	J	K	Q
0	0	0	stan się nie zmienia
	~~1~~	~~0~~	~~1~~
	~~0~~	~~1~~	~~0~~
1	1	1	zmiana stanu na przeciwny

Przerzutnik T zmienia swój stan na przeciwny w czasie impulsu zegarowego gdy T = 1. Stan przerzutnika pozostaje bez zmiany gdy T = 0.

W przypadku utrzymywania stanu T = 1, każdy kolejny impuls zegarowy zmienia stan przerzutnika na przeciwny. Działanie układu jest więc podobne do pracy włącznika, który przy każdym naciśnięciu na przemian włącza i wyłącza świecącą się lampkę. Układ ten stanowi podstawowy element liczników.

Liczniki

Licznikem nazywamy układ cyfrowy służący do zliczania impulsów.
Na wyjściu licznika pojawia się zakodowana binarnie liczba impulsów podanych na wejście zliczające. Oprócz wejścia impulsów zliczanych, licznik posiada zazwyczaj wejście ustawiające stan początkowy (zerowanie licznika).

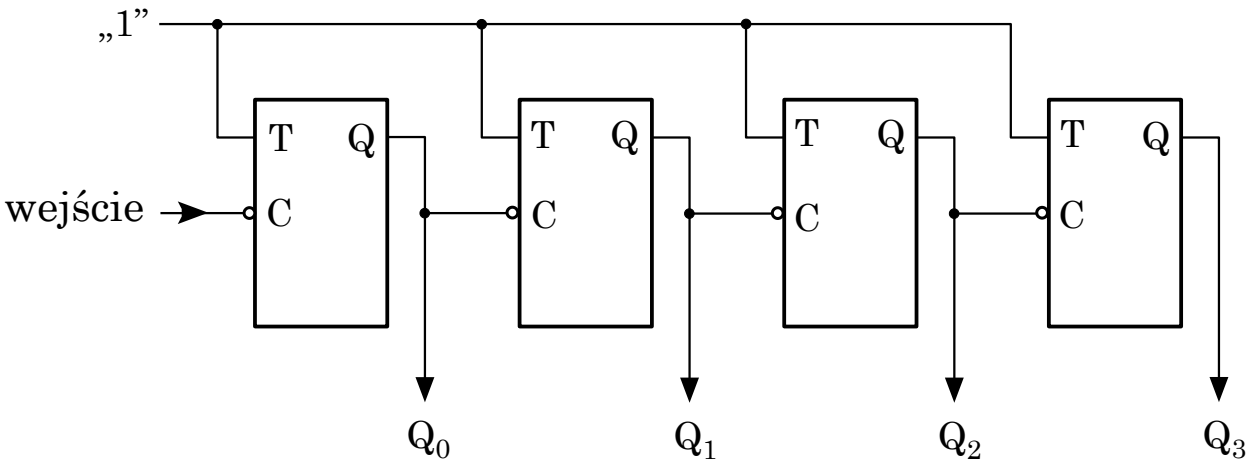
Rodzaje liczników:

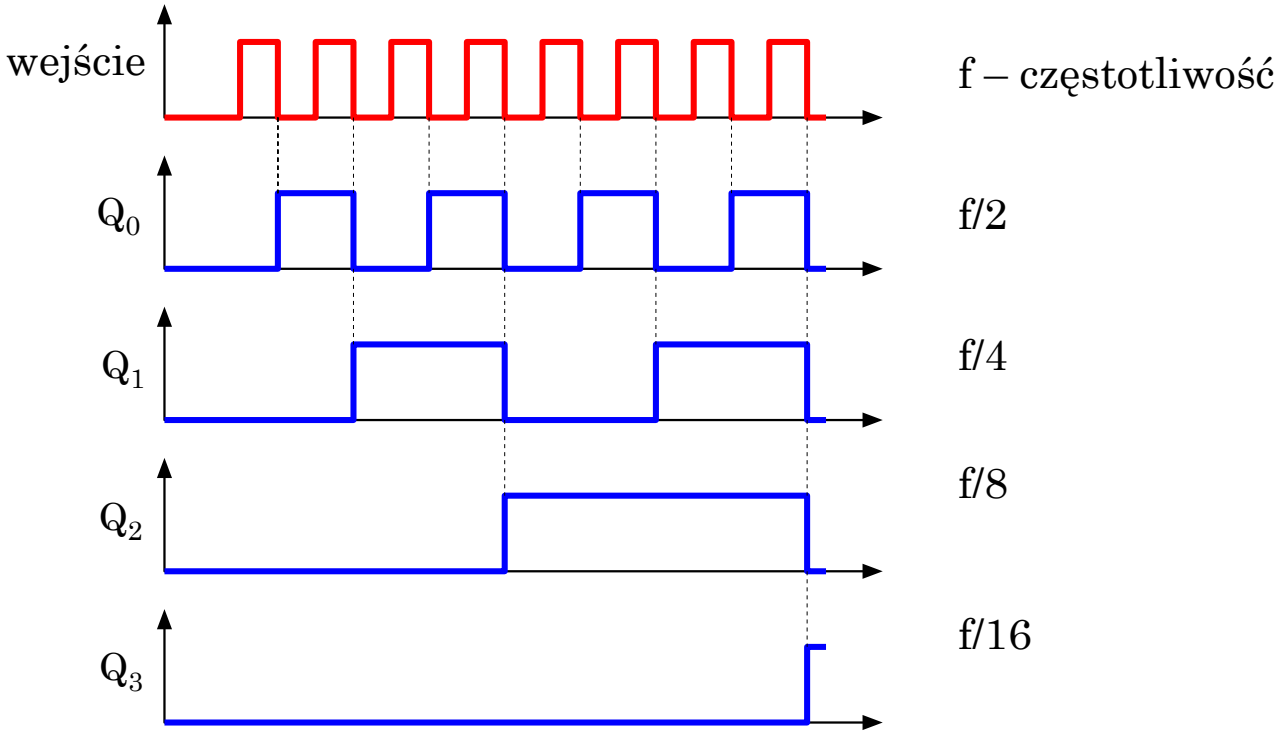
- liczące w przód (następnikowe)
- liczące w tył (poprzednikowe)
- rewersyjne (możliwość zmiany kierunku zliczania)

- szeregowe (asynchroniczne)
- równoległe (synchroniczne).

Podstawowymi elementami liczników są przerzutniki T. Pojedynczy przerzutnik T pozwala na zliczanie 2 impulsów. Najprostszy licznik można zbudować z szeregowo połączonych przerzutników T, z których każdy pod wpływem impulsu zegarowego zmienia swój stan na przeciwny do stanu poprzedniego.

Szeregowy licznik czterobitowy





Numer impulsu	Stan wyjść
n	Q_3 Q_2 Q_1 Q_0
0	0 0 0 0
1	0 0 0 1
2	0 0 1 0
3	0 0 1 1
4	0 1 0 0
5	0 1 0 1
6	0 1 1 0
7	0 1 1 1
8	1 0 0 0
9	1 0 0 1
10	1 0 1 0
11	1 0 1 1
12	1 1 0 0
13	1 1 0 1
14	1 1 1 0
15	1 1 1 1
16	0 0 0 0

Liczba impulsów zapisana w kodzie binarnym (modulo 16)

Przerzutniki wyzwalane zboczem opadającym sygnału zegarowego C

Rejestry

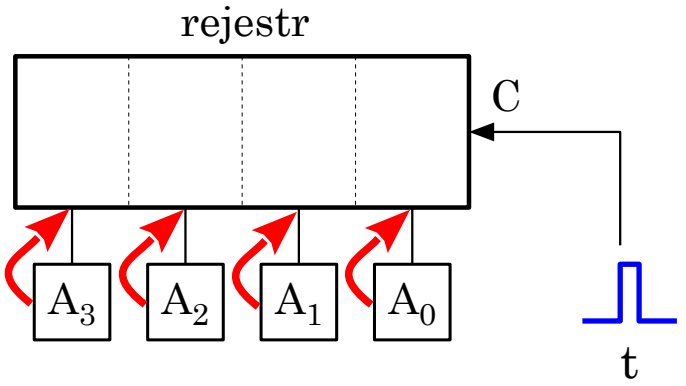
Rejestry służą do przechowywania informacji cyfrowej zapisanej w kodzie binarnym. Wpisana do rejestru informacja przechowywana jest do chwili wprowadzenia kolejnej, nowej informacji. Informacja ta może być również dostępna do odczytu.
Niekiedy odczyt zeruje wpisaną informację.

Ze względu na sposób wprowadzania i wyprowadzania informacji rejestry dzielimy na:

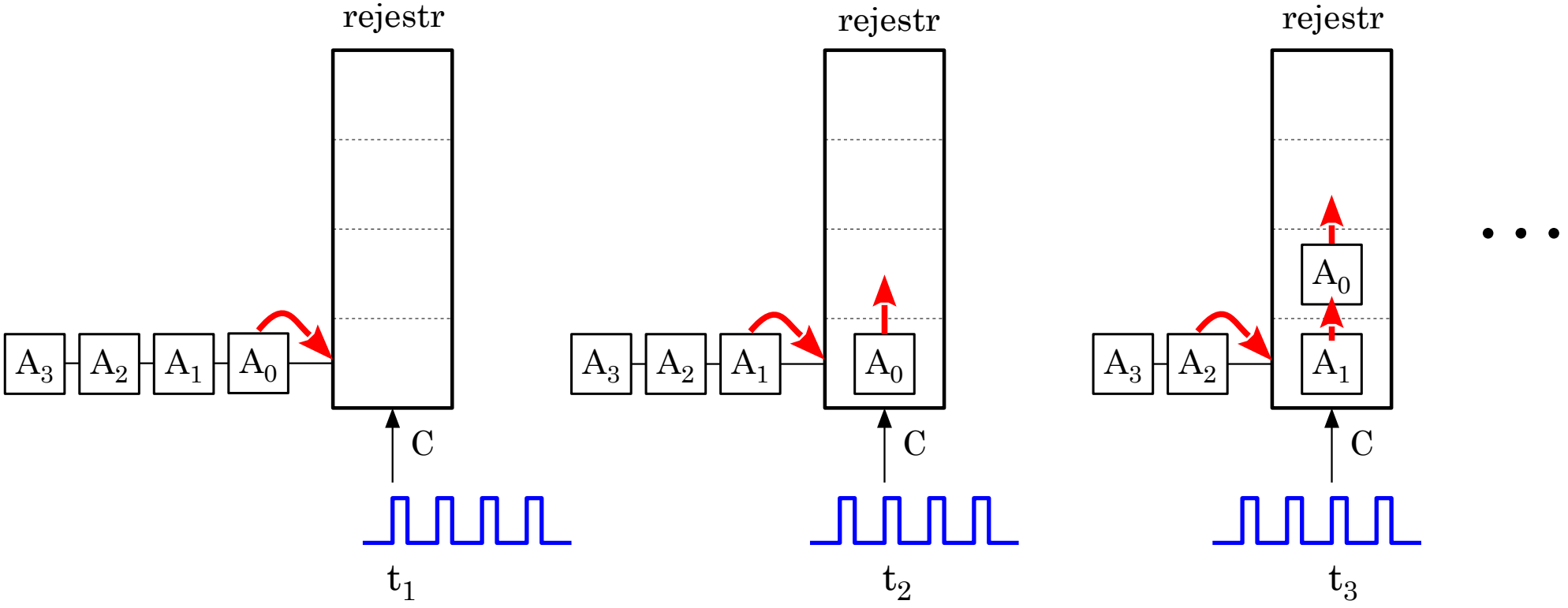
- szeregowe – wejście i wyjście szeregowe (rejestry przesuwające)
- równoległe – wejście i wyjście równoległe (rejestry buforowe)
- szeregowo-równoległe – wejście szeregowe, wyjście równoległe
- równoległo-szeregowe – wejście równoległe, wyjście szeregowe.

Podstawowym elementem rejestru są przerzutniki.
Liczba bitów informacji jaka może być przechowywana w rejestrze jest nazywana długością rejestru i odpowiada liczbie przerzutników z których jest zbudowany rejestr.

Wprowadzanie równoległe – wszystkie bity słowa informacji wprowadzamy jednocześnie, w jednym takcie zegara:

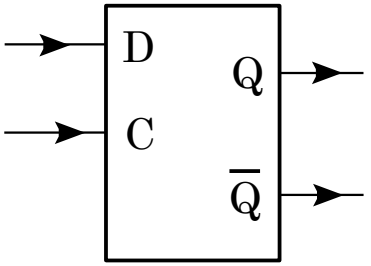


Wprowadzanie szeregowe – słowo wprowadzamy bit po bicie w kolejnych taktach zegara:



Rejestry

Najprostszym rejestrem jest przerzutnik D.



Zestawienie kilku takich przerzutników, np. 8, bez żadnych połączeń pomiędzy nimi utworzy 8 – bitowy rejestr równoległy (sygnał zegarowy wspólny dla wszystkich przerzutników).