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).

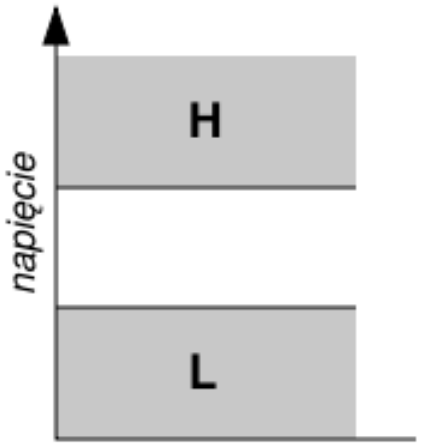
Pracę takich układów cyfrowych (układów logicznych) opisuje się za pomocą dwuwartościowej algebry Boole`a

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**

Układy cyfrowe



Logika dodatnia:
1 – 2.0 V – 5.0 V
0 - 0.0 V – 0.8 V

Logika ujemna:
1 – -50 mV – 0.0 mV
0 - -1.0 V – -0.4 V

Ze względu na obecność zakłóceń, wahania napięcia zasilającego 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.

Układy logiczne

Algebra Boole'a:

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

Podstawowe operacje:

$A * B \equiv A \; \boldsymbol{AND} \; B$

$A + B \equiv A \; \boldsymbol{OR} \; B$

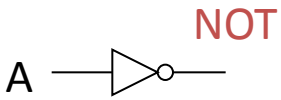
$\overline{A} \equiv \boldsymbol{NOT} \; A$

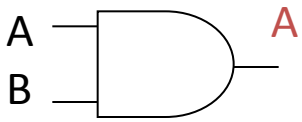
Podstawowe tożsamości algebry Boole'a

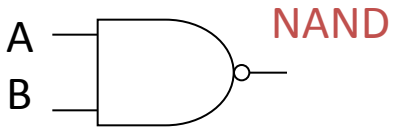
A* B = B*A	A+B = B+A	prawo przemienności
A*(B+C) = A*B + A*C	A+(B*C) = (A+B)*(A+C)	prawo rozdzielności
1 * A = A	0 + A = A	prawo tożsamości
$A * \overline{A} = 0$	$A + \overline{A} = 1$	prawo odwrotności
0 * A = 0	1 + A = 1	
A * A = A	A + A = A	
$\overline{A * B} = \overline{A} + \overline{B}$	$\overline{A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

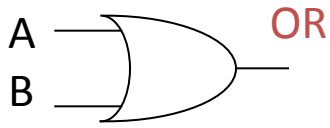
Bramki logiczne

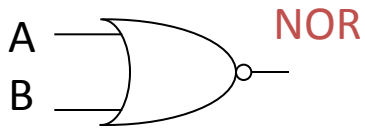
Podstawowe składniki wszystkich układów logicznych

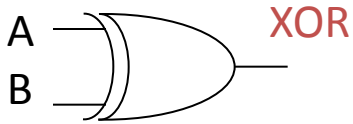


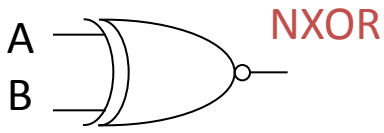












A	NOT
0	1
1	0

A	B	AND	NAND
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

A	B	OR	NOR
0	0	0	1
0	1	1	0
1	0	1	0
1	1	1	0

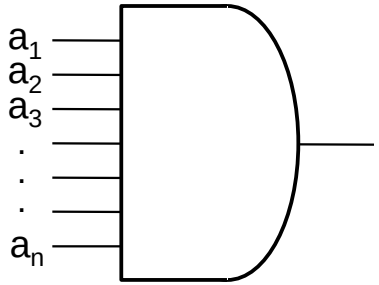
A	B	XOR	NXOR
0	0	0	1
0	1	1	0
1	0	1	0
1	1	0	1

Bramki logiczne

Oprócz bramek dwuwejściowych stosowane są również bramki wielowejściowe.

Przykład:

Wielowejściowa bramka AND

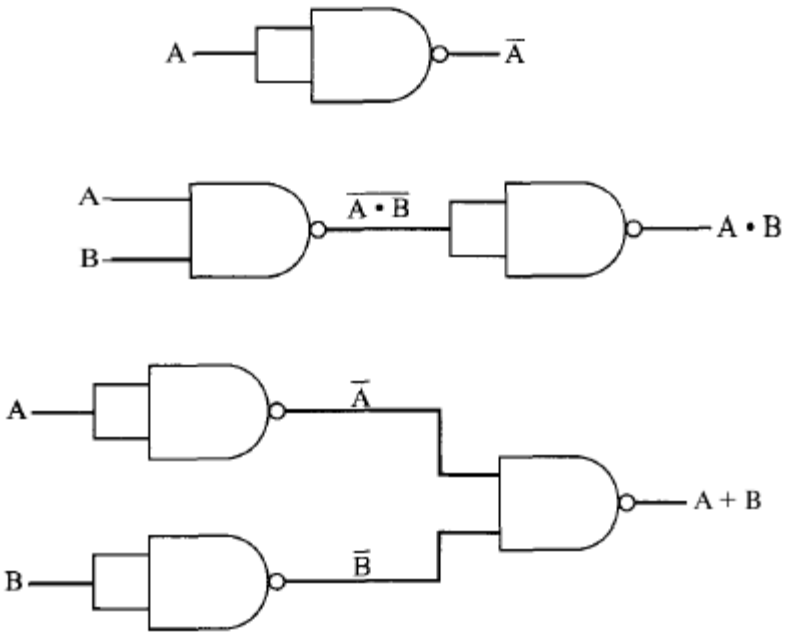


$f = a_1 * \cdots .* a_n$

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.

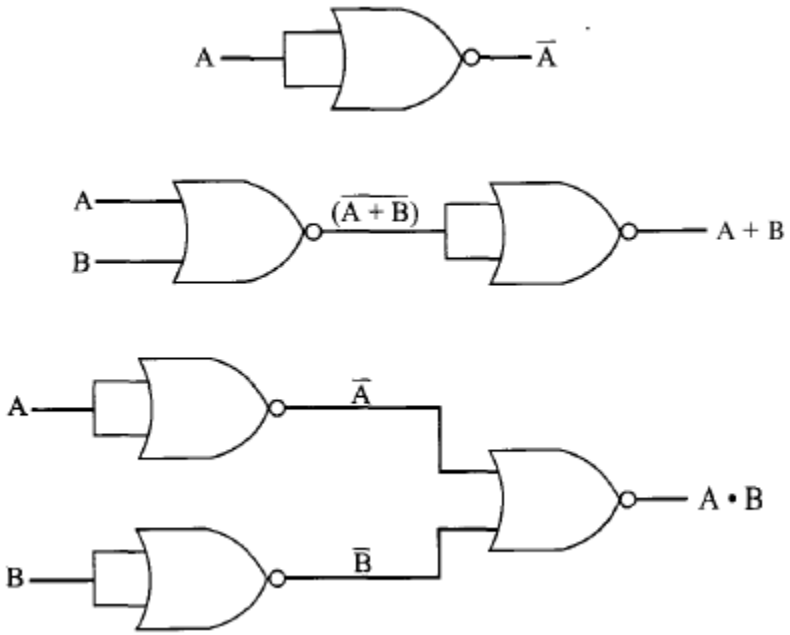
Przykłady realizacji funkcji logicznych

NAND

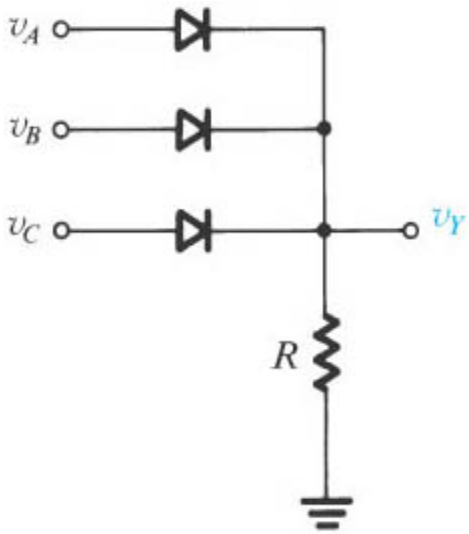


Przykłady realizacji funkcji logicznych

NOR

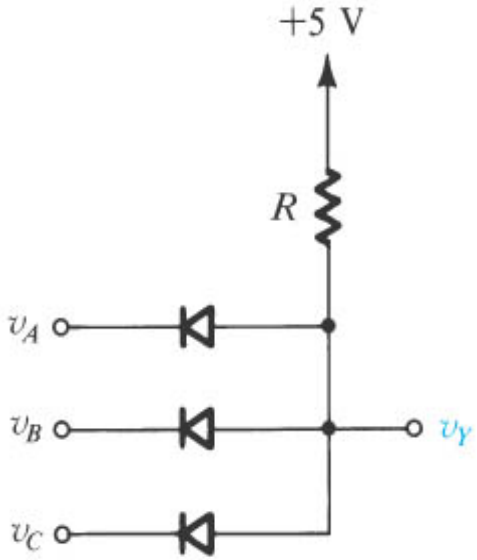


Praktyczna realizacja układów logicznych



(a)

OR



(b)

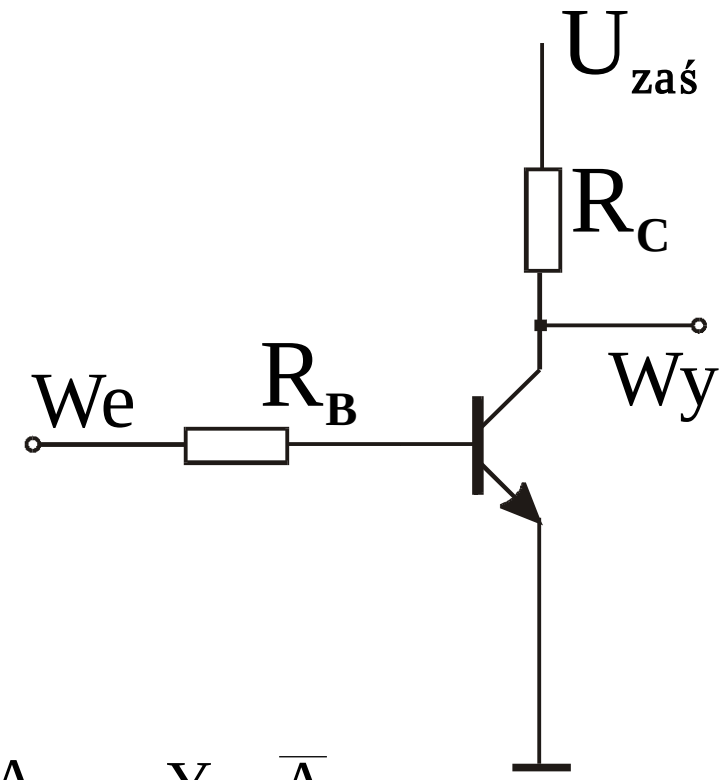
AND

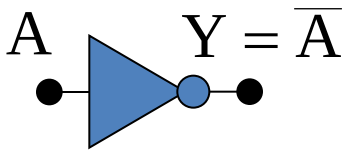
Układ **AND**

Podanie na jedno z wejść napięcia 0 V polaryzuje diodę w kierunku przewodzenia.

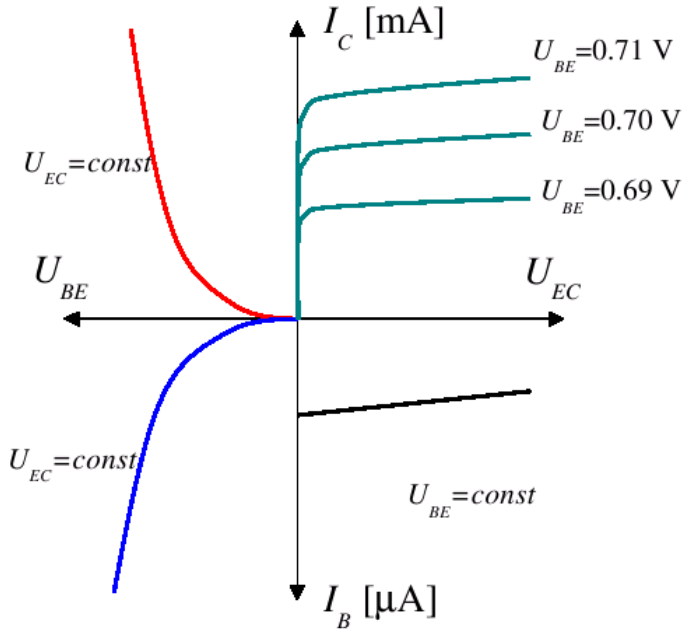
Na wyjściu mamy niewielkie napięcie równe spadkowi napięcia na diodzie spolaryzowanej w kierunku przewodzenia

Praktyczna realizacja układów logicznych

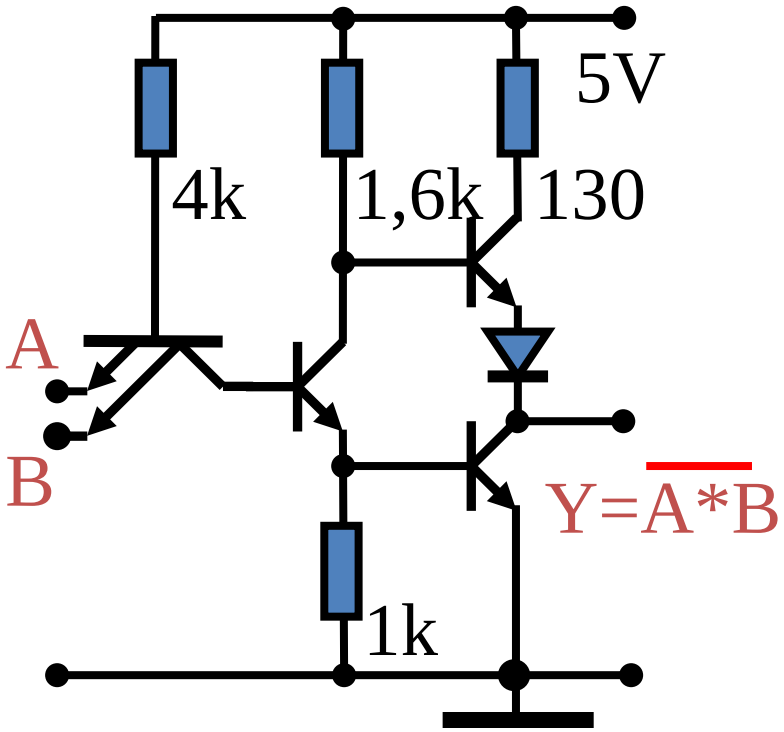




NOT



Praktyczna realizacja układów logicznych



NAND

Układ TTL 7400

Zakres napięć na wejściach:

L – (0 - 0.8) V

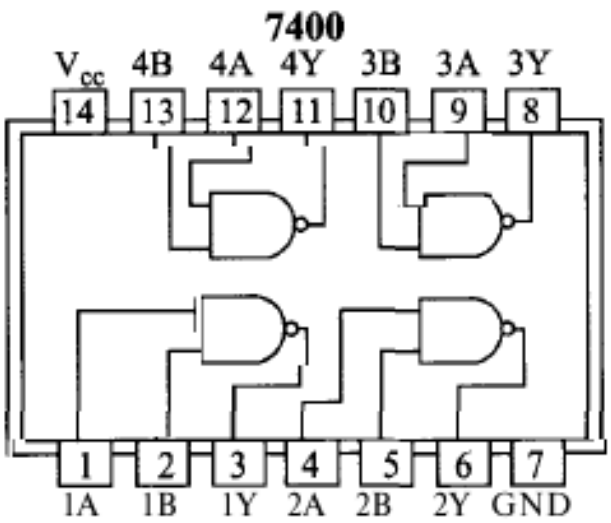
H - (2 - 5.5) V

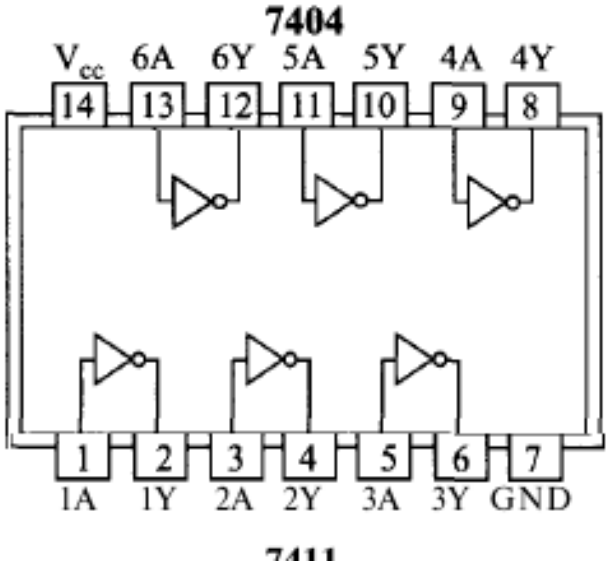
Zakres napięć na wyjściu

L – (0 - 0.4) V

H - (2.4 - 5.5) V

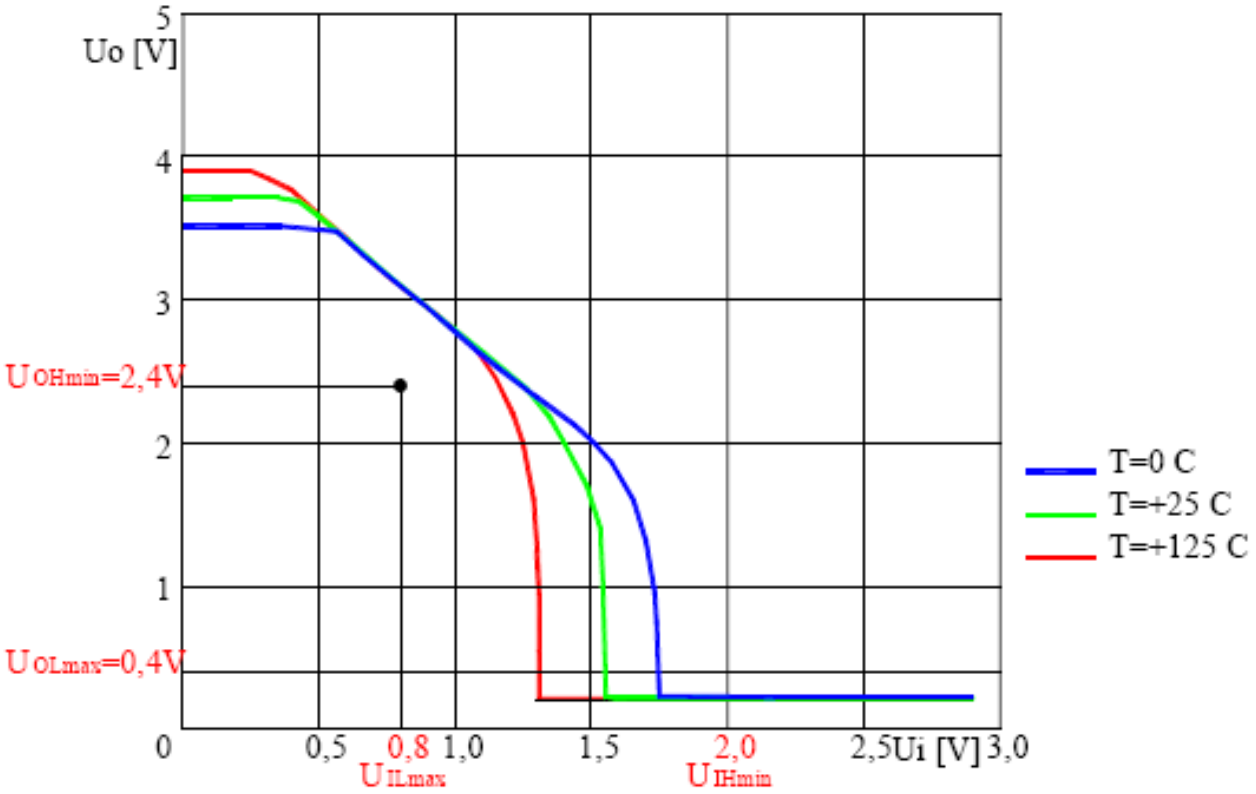
Układy małej skali integracji (SSI)





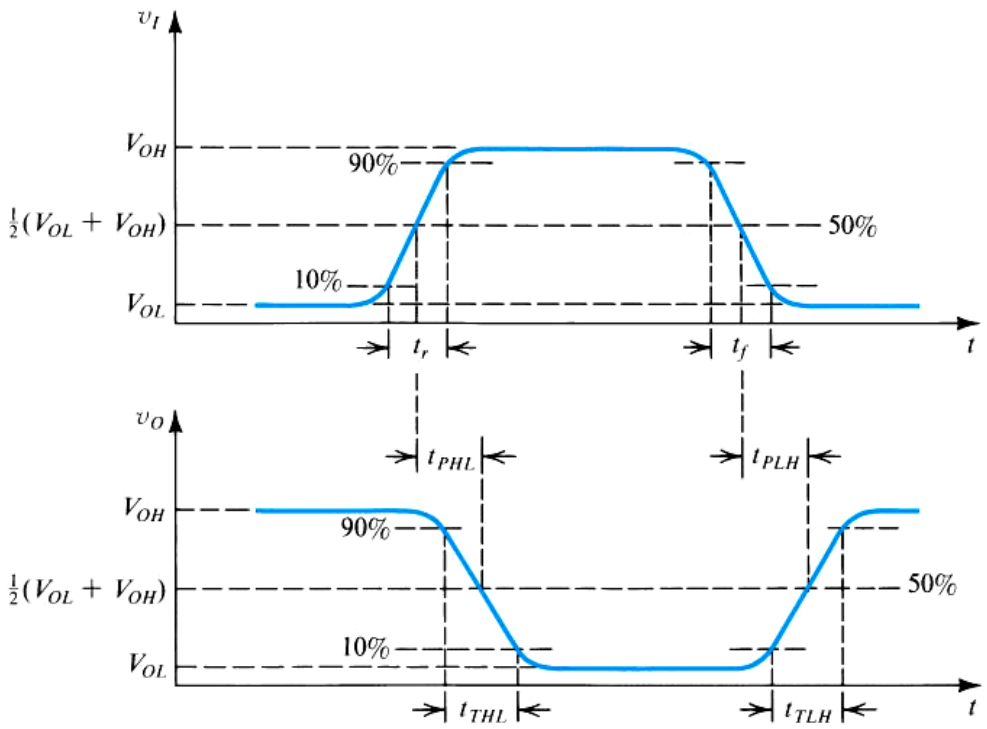
Aby zrealizować funkcję logiczną należy użyć pewną liczbe tych układów

Charakterystyka przejściowa bramki NAND TTL



Charakterystyka przejściowa podstawowej bramki NAND TTL serii standardowej, zależność charakterystyki przejściowej od temperatury

Czas propagacji



$$t_P \equiv \frac{t_{PLH} + t_{PHL}}{2}$$

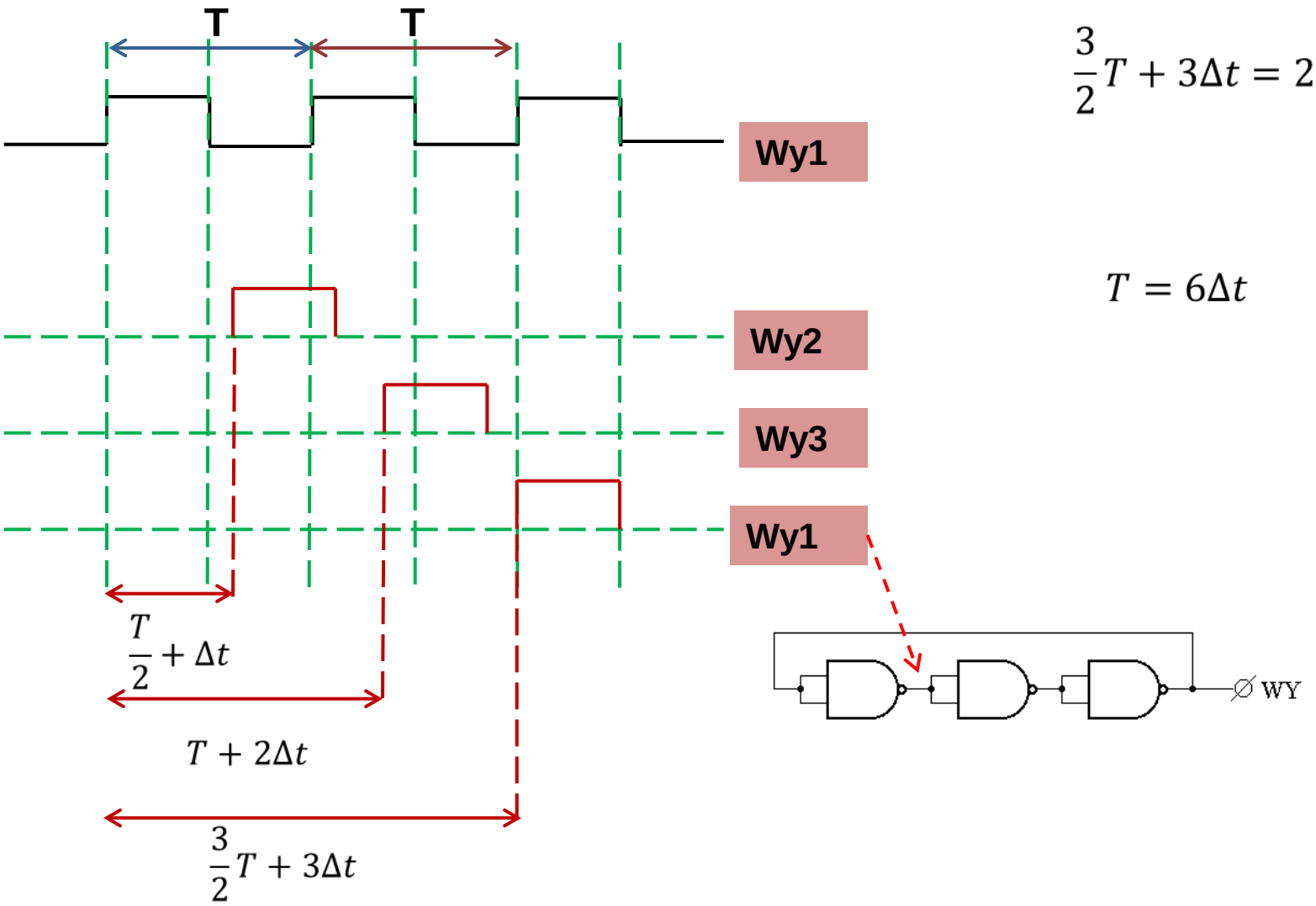
dla 7400:

t_{PHL} = 15 nsek

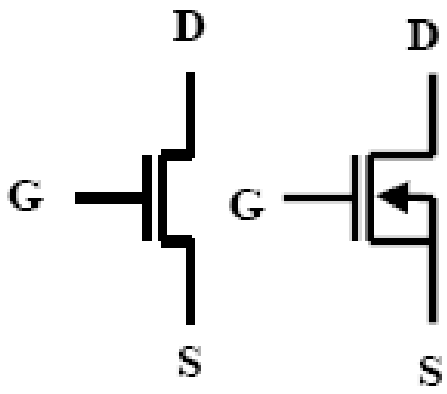
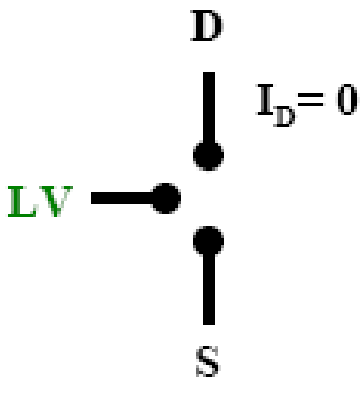
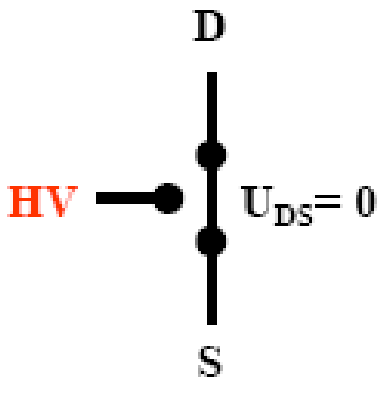
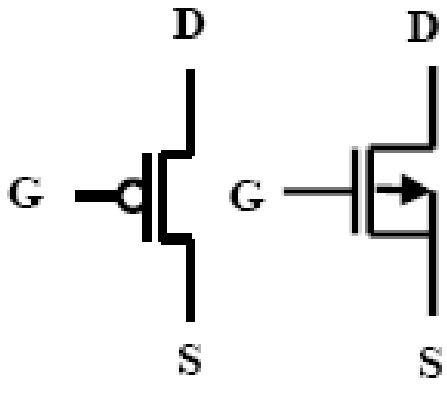
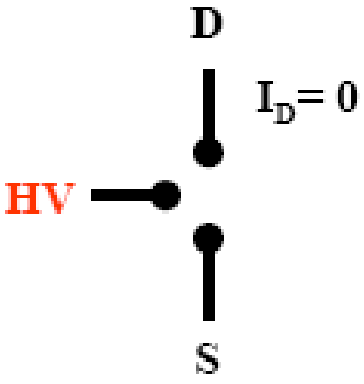
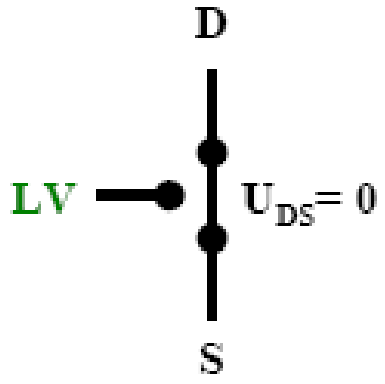
t_{PLH} = 22 nsek

W układach z diodami Shottky'ego (dioda ze złączem metal-półprzewodnik) uzyskuje się czasy przełaczenia rzędu 3 nsek

Generator zbudowany na NAND'ach



Tranzystor MOS jako łącznik

	symbol	OFF	ON
N-MOS przewodzi LV			
P-MOS przewodzi HV			

Klasy układów cyfrowych

TTL (Transistor – Transistor - Logic) – układy TTL,

ECL (Emiter – Coupled Logic) – układy o sprzężeniu emiterowym,

MOS (Metal – Oxide - Semiconductor) – układy MOS,

CMOS (Complementary MOS) – układy komplementarne MOS,

BiCMOS (Bipolar CMOS) – układy ,,mieszane", bipolarne CMOS,

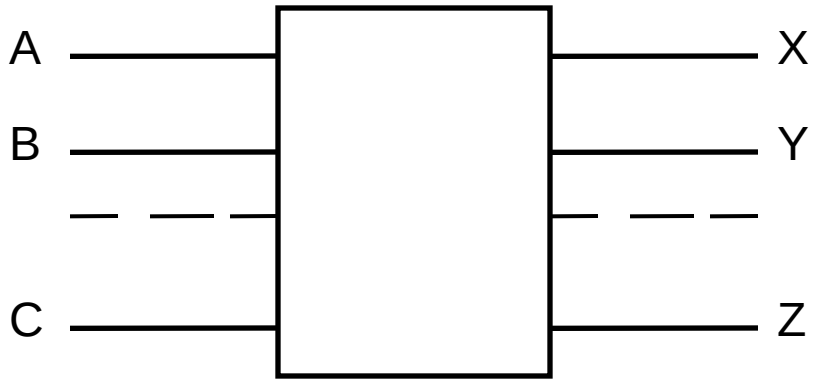
I^2L (Integrated Injection Logic) – układy iniekcyjne,

CTD (Charge Transfer Device) – układy o sprzężeniu ładunkowym,

GaAs MESFET – układy GaAs.

Bloki funkcjonalne

- elementarny blok mający jedno lub więcej wejść i jedno lub więcej wyjść. Jest on zwykle projektowany jako standardowa jednostka funkcjonalna. Zadaniem układu logicznego jest przyjmowanie standardowych sygnałów logicznych na swoich wejściach i produkowanie na wyjściach innych, również standardowych sygnałów logicznych



Ogólne oznaczenie układu logicznego

Struktura wewnętrzna układu logicznego może zawierać różne rodzaje układów przełączających.Zmienne logiczne (mające wartości 0 lub 1) są oznaczone przez *A, B, C..., X, Y, Z.*

Bloki funkcjonale

Układy kombinacyjne

Stan wyjść jest jednoznacznie określony przez stan wejść układu:

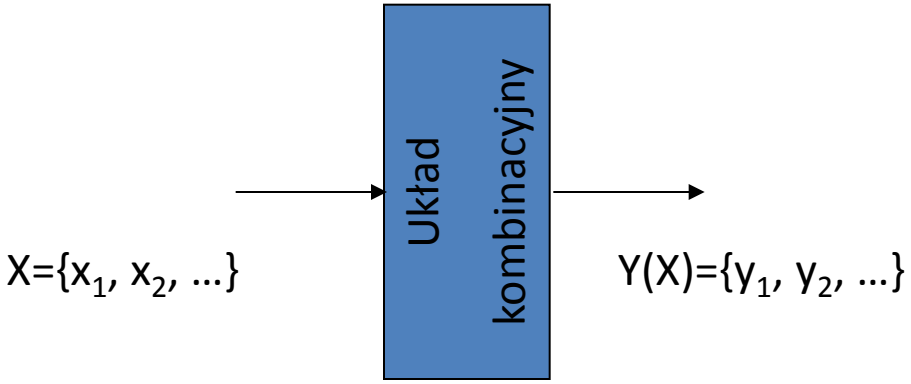
Układy sekwencyjne

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

$$Y(t_n) = f(X(t_0), X(t_1), \dots, X(t_{n-1}))$$

Układy kombinacyjne

- Stan wyjść zależy tylko od stanu wejść
- Układ taki można definiować za pomocą:
  - Tablicy prawdy
  - Symbolu graficznego
  - Równania Boole'a



Układy kombinacyjne

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

Układy kombinacyjne

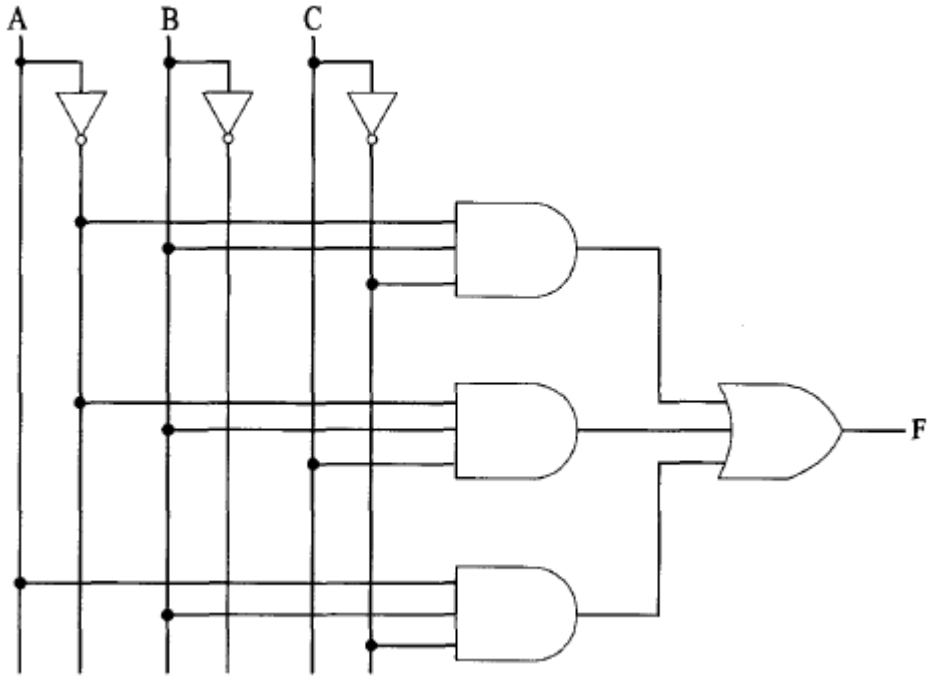
Równanie Boole’a:

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

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

Układy kombinacyjne

Realizacja układu za pomocą bramek AND, OR i NOT:



Metody upraszczanie układów kombinacyjnych

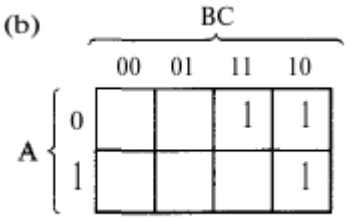
Mapa Karnaugha:

Kod Graya

(a)

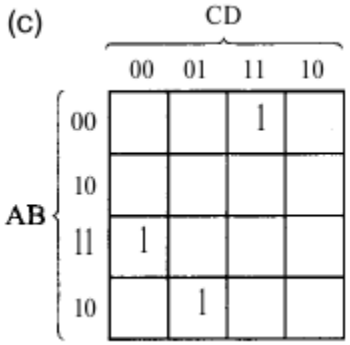
$F=\overline{A}*B+A*\overline{B}$

(b)



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

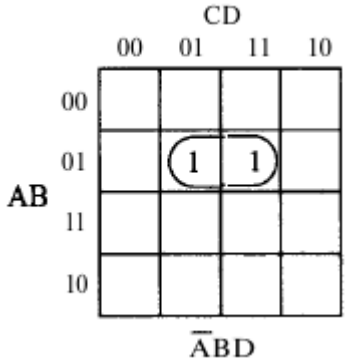
(c)



$F=\overline{A}*\overline{B}*C*D+A*B*\overline{C}*\overline{D}+A*\overline{B}*\overline{C}*D$

Metody upraszczanie układów kombinacyjnych

Mapa Karnaugha:

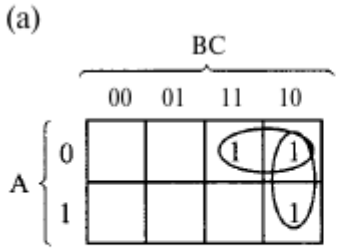


$$F = \overline{A} * B * \overline{C} * D + \overline{A} * B * C * D = \overline{A} * B * D$$

Jeśli sąsiadujące kwadraty zawierają 1, to odpowiednie iloczyny różnią się tylko jedną zmienną. W takim przypadku te iloczyny mogą być połączone przez wyeliminowanie tej zmiennej

Metody upraszczanie układów kombinacyjnych

Mapa Karnaugha:

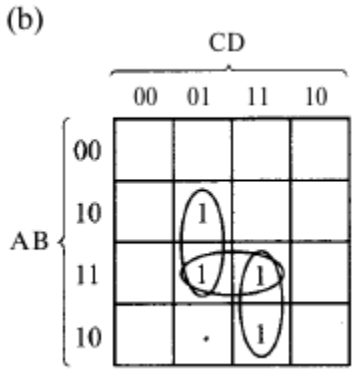


$F=\overline{A} * B + B * \overline{C}$

Gdy zakreślamy grupy, dozwolone jest użycie tej samej jedynki więcej niż jeden raz.

Metody upraszczanie układów kombinacyjnych

Mapa Karnaugha:



$$F=A*\overline{C}*D+A*C*D$$

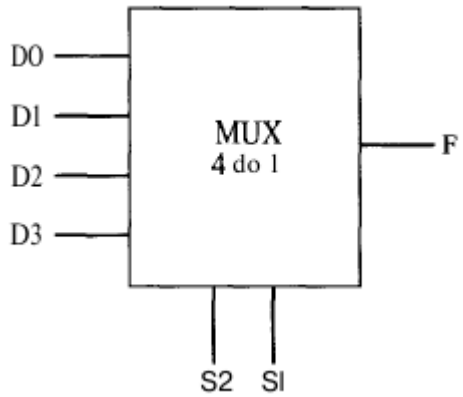
Możemy wyeliminować dowolną grupę jedynek, która w całości nakłada się z innymi grupami

Układy kombinacyjne

Przykładowe układy:

- Multiplekser, demultiplekser
- Koder, dekoder
- Sumator
- Komparator

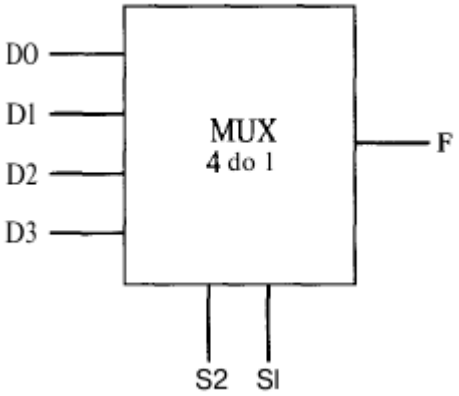
Multiplekser

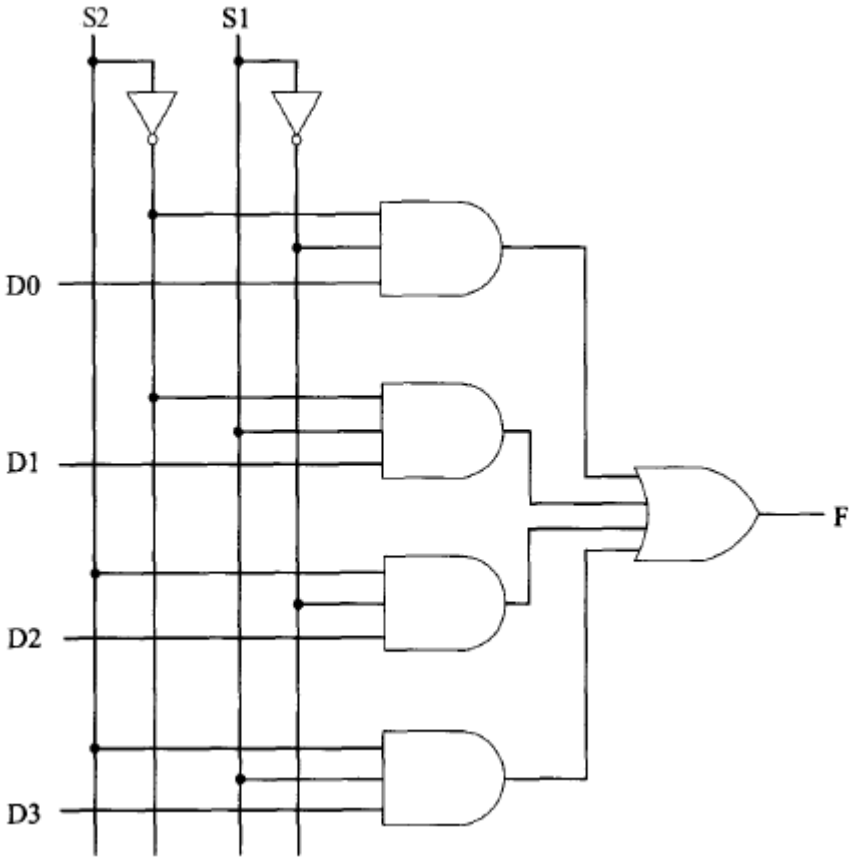


S2	S1	F
0	0	D0
0	1	D1
1	0	D2
1	1	D3

Tablica prawdy

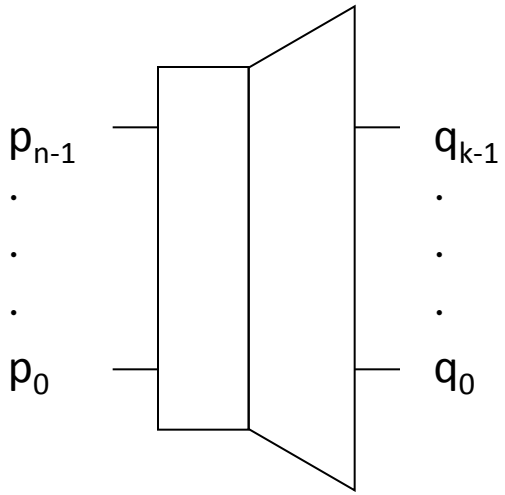
Multiplekser

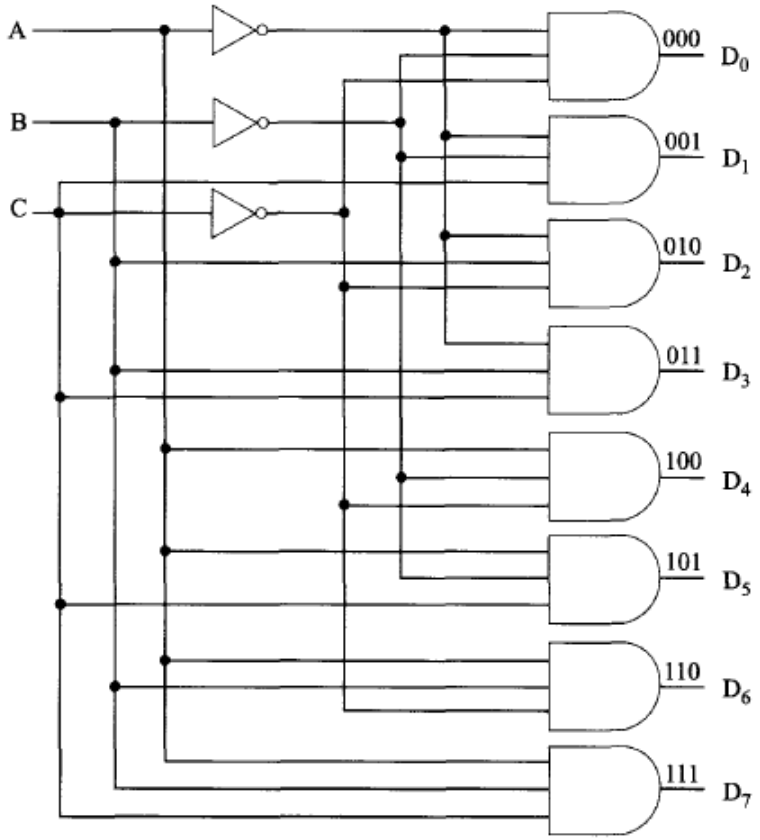




S2	S1	F
0	0	D0
0	1	D1
1	0	D2
1	1	D3

Dekodery





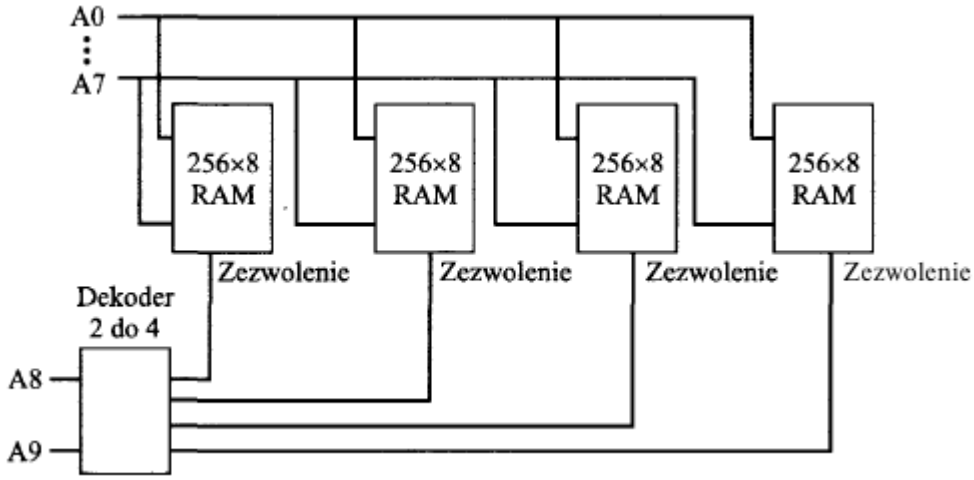
$p_{n-1} \dots p_0$ – wejścia dekodera

$q_{k-1} \dots q_0$ – wyjścia dekodera

$k=2^n$

Dekodery znajdują zastosowanie np. do dekodowania adresu

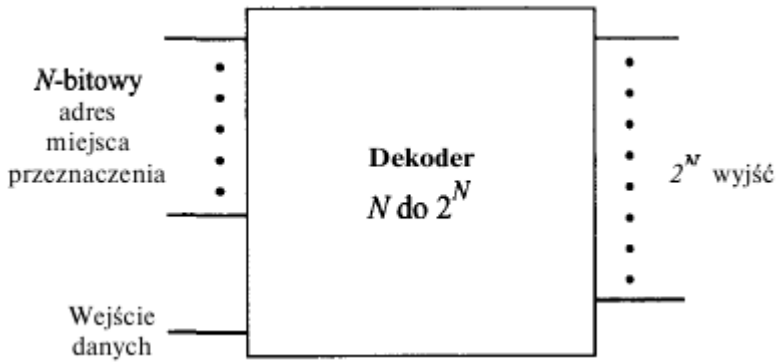
Dekodery



Chcemy zbudować 1 kilobajtowa z czterech układów RAM o pojemności 256 bajtów. Przestrzeń adresową możemy podzielić następująco:

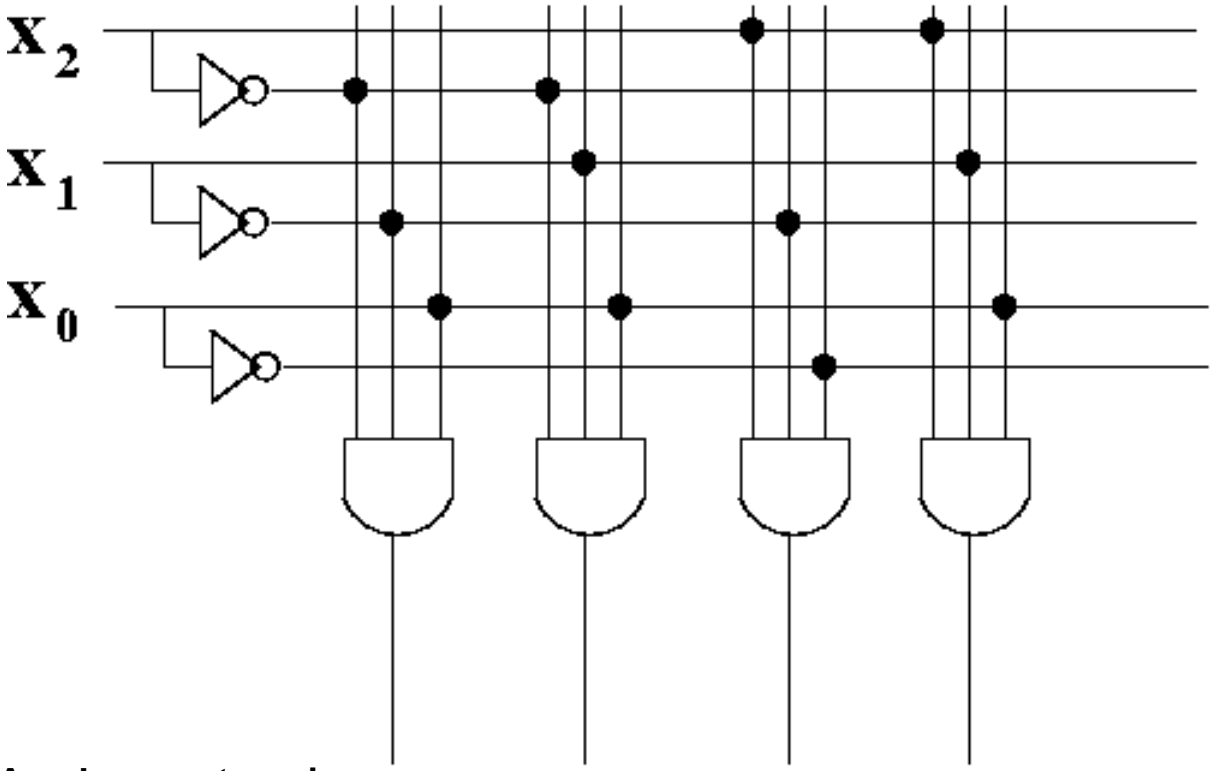
adres	układ
0000 – 00FF	0
0100 – 01FF	1
0200 – 02FF	2
0300 – 03FF	3

Demultiplekser



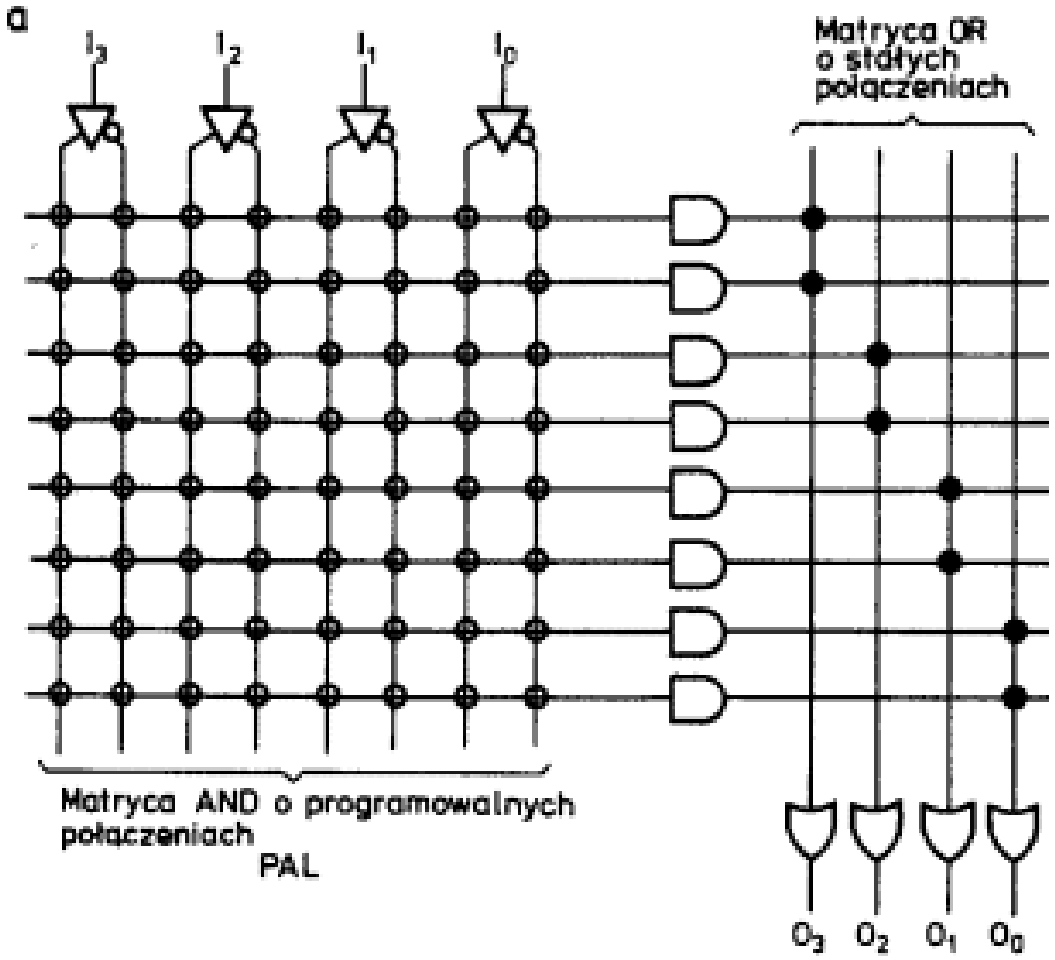
Po dodaniu jednej linii wejściowej dekoder może służyć jako demultiplekser

Programowalne tablice logiczne (PLA)

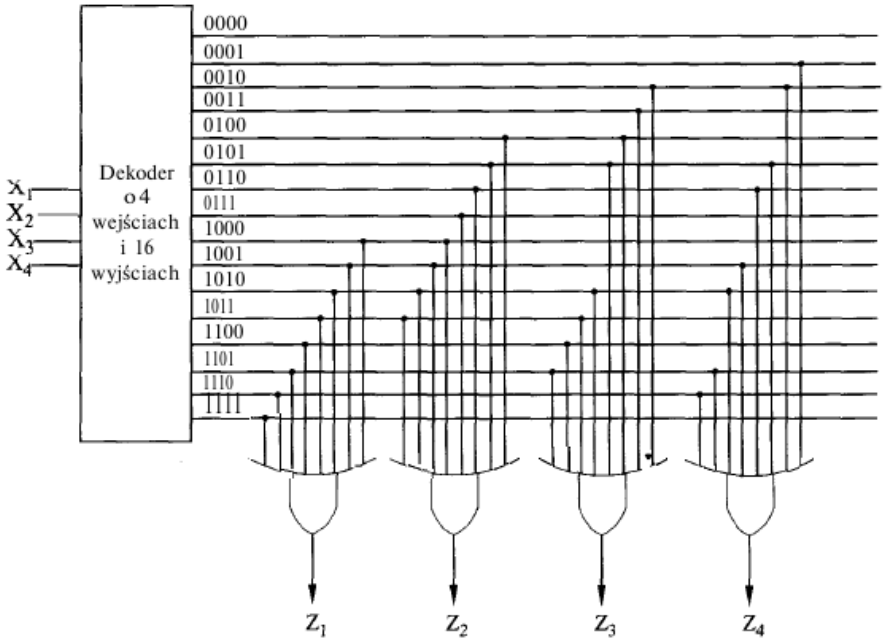


Koncepcja PLA polega na tym, że dowolna funkcja Boole'a może być wyrażona na podstawie sumy iloczynów. Programowanie polega na przepalaniu zbędnych połaczeń.

Programowalne tablice logiczne (PLA)



Pamięć stała (ROM –read only memory)



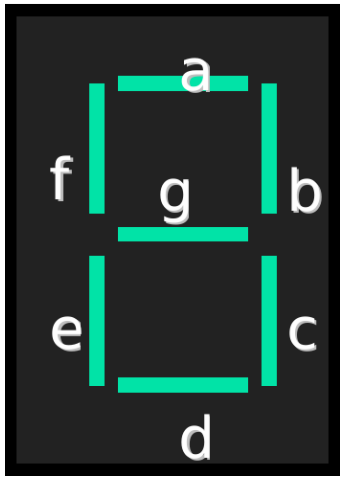
Wejścia/ adresy				Wyjścia /zawartość			
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	0

Informacja zawarta w pamięci ROM jest trwała. Jest ona zapisana w procesie tworzenia układu.

Kody, konwertery kodów

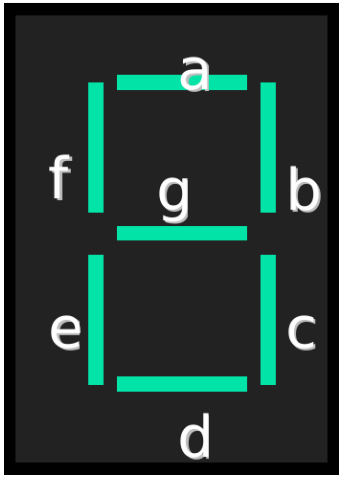
Każda informacja może być przedstawiona jako określona kombinacja bitów. Kombinacja bitów przypisana danej informacji jest nazywana **kodem**. Kodowanie umożliwia na przykład przedstawienie symboli cyfrowych, liter lub znaków w postaci binarnych słów logicznych.

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

Układy logiczne realizujące kod siedmiosegmentowego wyświetlacza



Dla segmentu c mamy:

A \ BC	0 0	01	11	1 0
0	1	1	1	0
1	1	1	1	1

$$F = \bar{B} * \bar{C} + \bar{B} * C + B * C + A * B = \bar{B}(\bar{C} + C) + BC + AB$$
$$= \bar{B} + B(A + C)$$
$$= (\bar{B} + B)\bar{B} + A + C$$
$$= \bar{B} + A + C = \overline{\bar{A}B} + C = \overline{\bar{A}B\bar{C}}$$

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