

Układy cyfrowe - elementarz

Tomasz Słupiński

27.04.2015

Pracownia Fizyczna i Elektroniczna,
dla Inżynierii Nanostruktur oraz
Energetyki i Chemii Jądrowej

Pliki symulacji do wykładu dla programu **Logisim** (licencja GPL, dr Carl Burch) są
dostępne tutaj:

http://www.fuw.edu.pl/~tomslu/PFiE_2014L/

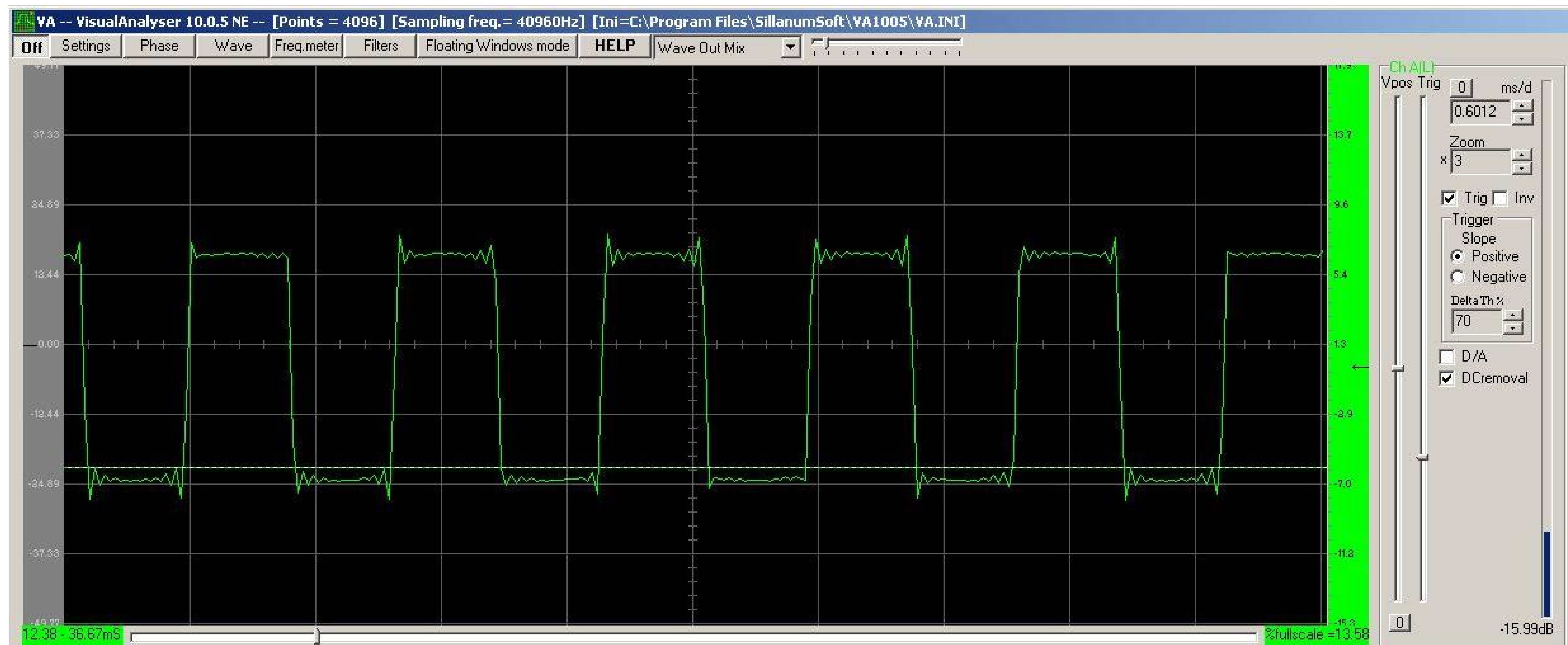
Plan

- **Elektronika cyfrowa vs analogowa**
 - bity, układy liczenia: dwójkowy, oktalny, heksadecymalny,
 - kod BCD
 - transmisja danych równoległa i szeregowo
 - algebra Boole'a - rachunek zdań, tożsamości logiczne, funkcje logiczne
- **Bramki logiczne – układy kombinacyjne**
 - reprezentacja układów logicznych, tabelki prawdy, zapis funkcyjny, układy bramek logicznych,
 - bramki NOT, OR, NOR, AND, NAND, XOR
 - rodzina bramek TTL i HC
- **Symulator układów logicznych *Logisim* i przykłady symulacji działania układów**
 - przykłady działania bramek
 - dekodery liczby dziesiętnej kodowanej binarnie w kodzie BCD
 - wyświetlacz numeryczny 7-segmentowy
 - półsumator, sumator

- Przykłady układów kombinacyjnych i zasady ich projektowania, symulator *Logisim*
- Układy sekwencyjne – przerzutniki
 - przerzutnik jako elementarny układ pamięciowy
 - przerzutniki typu T, D, J-K, S-R
 - wejścia asynchroniczne przerzutników
- Przykłady układów z przerzutnikami
 - licznik 4-bitowy
- Inne ważne układy cyfrowe (tylko wspomniane, bez szczegółów):
 - przetworniki analogowo-cyfrowy A/D i cyfrowo-analogowy D/A
 - rejestry
 - pamięć
 - ALU – jednostka arytmetyczno-logiczna mikroprocesora
 - układy wejścia – wyjścia mikroprocesora lub mikrokontrolera
- Omówienie makiety do ćwiczeń

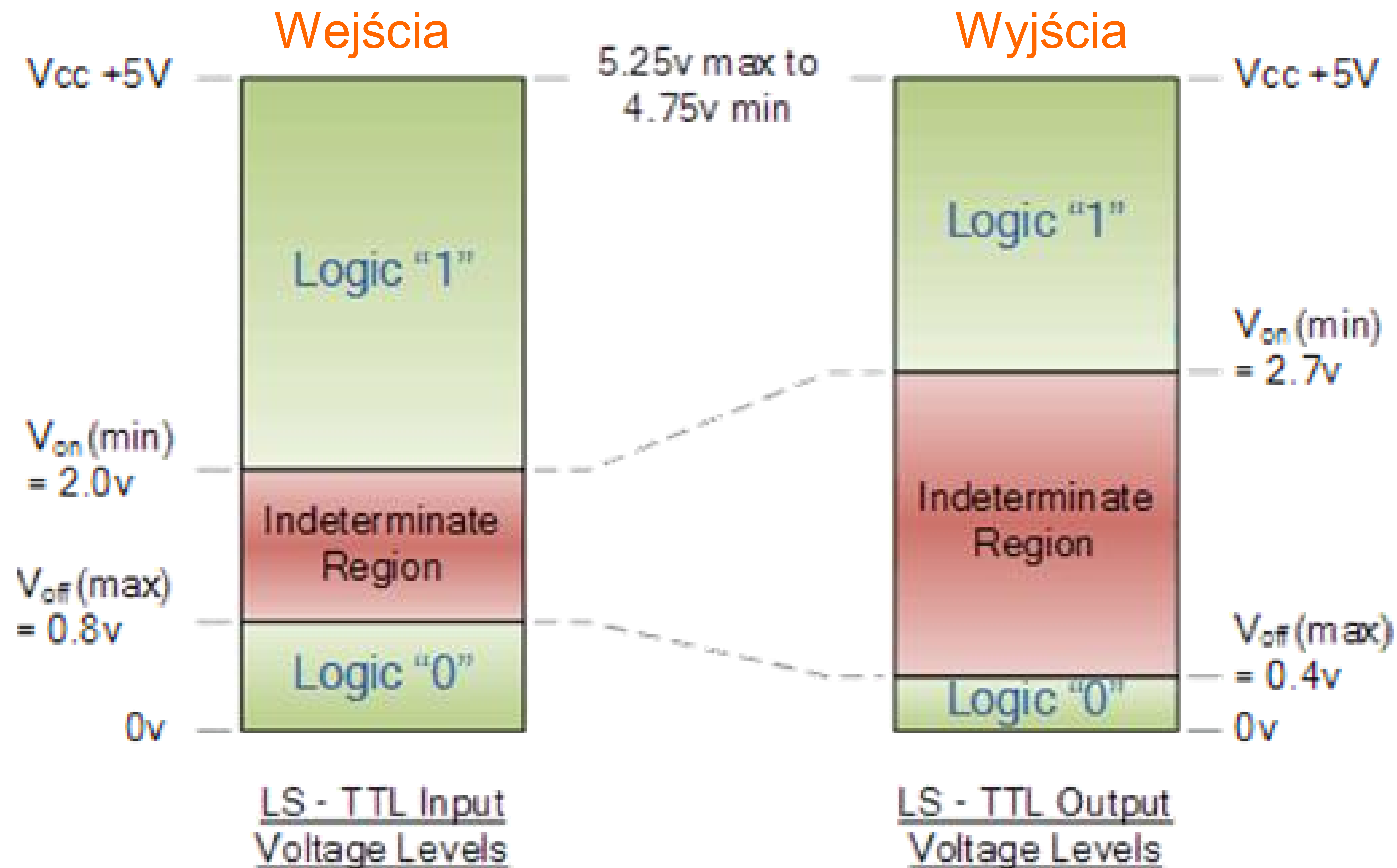
Układy analogowe – napięcie przetwarzane przez układ analogowy może przyjmować dowolne wartości z pewnego przedziału
np. wzmacniacz akustyczny, wzmacniacz pomiarowy, itp

Układy cyfrowe – układ interpretuje napięcia jako jeden z dwu stanów
0 (niski L = Low) albo 1 (wysoki H = High)
i przetwarza takie sygnały dwustanowe zgodnie z zasadami logiki matematycznej (algebry Boole'a) oraz dwójkowego układu liczenia



0 1 0 1 0 1 0 1 0 1
- przebieg prostokątny o stałej częstotliwości

Poziomy napięć odpowiadających stanom logicznym 0 (L=low) i 1 (H=high) dla rodziny układów **TTL / HC** (ozn. 74nnn)

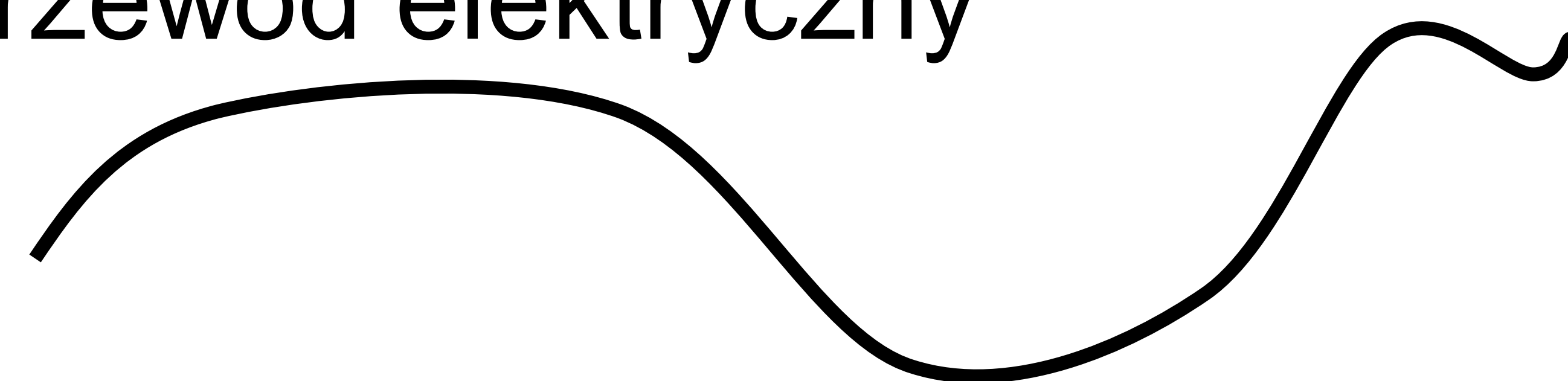


stany TTL	„0”	U wejścia: 0 – 0.8 V	U wyjścia: 0 – 0.4 V
	„1”	U wejścia: 2 – 5 V	U wyjścia: 2.7 – 5 V

Wyjścia mają większy odstęp napięć dla stanów 0 i 1 niż wejścia aby zmniejszyć szanse błędu przy podłączaniu wyjść do wejść kolejnych układów

Sygnały analogowe vs. cyfrowe; dwójkowy układ liczenia, bity, bajty

przewód elektryczny



$U(t)$ - sygnał analogowy, np. 155 woltów

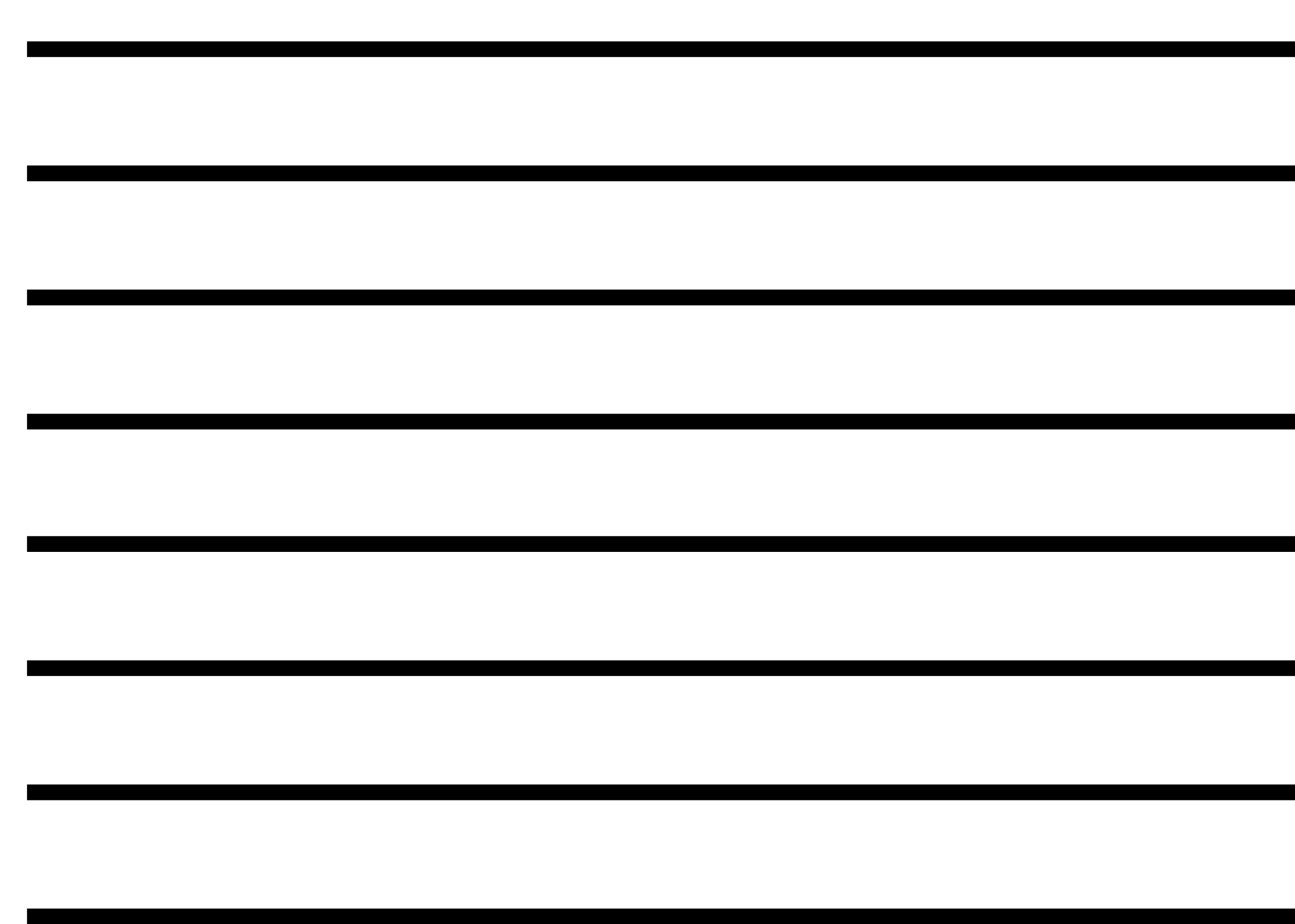
Sygnały cyfrowe:

Kod dwójkowy: $10011011_{(2)} = 1 \cdot 2^7 + 0 \cdot 2^6 + 0 \cdot 2^5 + 1 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = 155_{(10)}$

Przesyłanie wartości liczbowej 8-bitowej (transmisja **równoległa** 8-bitowa)

MSB – bit najbardziej
znaczący,
najstarszy

LSB – bit najmniej
znaczący,
najmłodszy



8 przewodów

2^7
 2^6
 2^5
 2^4
 2^3
 2^2
 2^1
 2^0

np.: 1
0
0
1
1
0
1
1

słowo 8-bitowe = **bajt**

Bity liczby dwójkowej można też przesyłać kolejno pojedynczym przewodem wg ustalonego standardu - transmisja **szeregowa** (np. USB – Universal Serial Bus)

Układ liczenia ósemkowy (oktalny)

$$10011011_{(2)} = 1 \cdot 2^7 + 0 \cdot 2^6 + 0 \cdot 2^5 + 1 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = 155_{(10)} = \\ = 233_{(8)} = (1 \cdot 2^7 + 0 \cdot 2^6) + (0 \cdot 2^5 + 1 \cdot 2^4 + 1 \cdot 2^3) + (0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0)$$

Oktalnie: cyfry 0, 1, 2, 3, 4, 5, 6, 7

kolejne pozycje liczby oktalnej: $8^0 = 1$, $8^1 = 8$, $8^2 = 64$, $8^3 = 512$ itd.

Łatwe tłumaczenie liczb w systemie dwójkowym na system oktalny – trójki cyfr dwójkowych od prawej strony dają kolejne cyfry liczby oktalnej.

CYFRA	kod dwójkowy
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111

Układ liczenia szesnastkowy (heksadecymalny)

cyfry 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

kolejne pozycje liczby heksadecymalnej: $16^0=1$, $16^1=16$, $16^2=256$,

np. $10Bh = 1 \cdot 16^2 + 0 \cdot 16^1 + 11 \cdot 16^0 = 267_{(10)}$ inny zapis: 0x10B

- takim kodem numerowane są np. [komórki pamięci komputerów](#)

LICZBA	CYFRA W KODZIE HEKSADECYMALNYM	ZAPIS DWÓJKOWY
A		2^3 2^2 2^1 2^0
0	0	0 0 0 0
1	1	0 0 0 1
2	2	0 0 1 0
3	3	0 0 1 1
4	4	0 1 0 0
5	5	0 1 0 1
6	6	0 1 1 0
7	7	0 1 1 1
8	8	1 0 0 0
9	9	1 0 0 1
10	A	1 0 1 0
11	B	1 0 1 1
12	C	1 1 0 0
13	D	1 1 0 1
14	E	1 1 1 0
15	F	1 1 1 1

Łatwe tłumaczenie zapisu dwójkowego na heksadecymalny $10011011_{(2)} = 9Bh$

- czwórki cyfr binarnych dają cyfry heksadecymalne.

Kod BCD – Binary Coded Decimal

Kod BCD – Binary Coded Decimal
(cyfry dziesiętne kodowane dwójkowo)

Kodowanie BCD umożliwia przesyłanie
4 przewodami kolejnych cyfr
liczby dziesiętnej.

CYFRA	BCD
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

Np. $155_{(10)} = (0001\ 0101\ 0101)_{\text{BCD}}$

Sygnały analogowe vs. cyfrowe

Sygnały cyfrowe:

- znacznie bardziej odporne na zakłócenia, znacznie mniej błędów
- można je łatwo przetwarzać wg różnych funkcji
- można w znaczny sposób zautomatyzować wykonywanie pomiarów, przetwarzanie wyników i wykorzystanie wyników do sterowania i kontroli
- dane w postaci cyfrowej można łatwo przechowywać (pamięć)

Układy cyfrowe są względnie tanie dzięki masowej produkcji standardowych uniwersalnych części składowych – **cyfrowych układów scalonych**



Pomiary fizyczne dotyczą zwykle sygnałów analogowych, sterowanie też zwykle wymaga sygnałów analogowych.

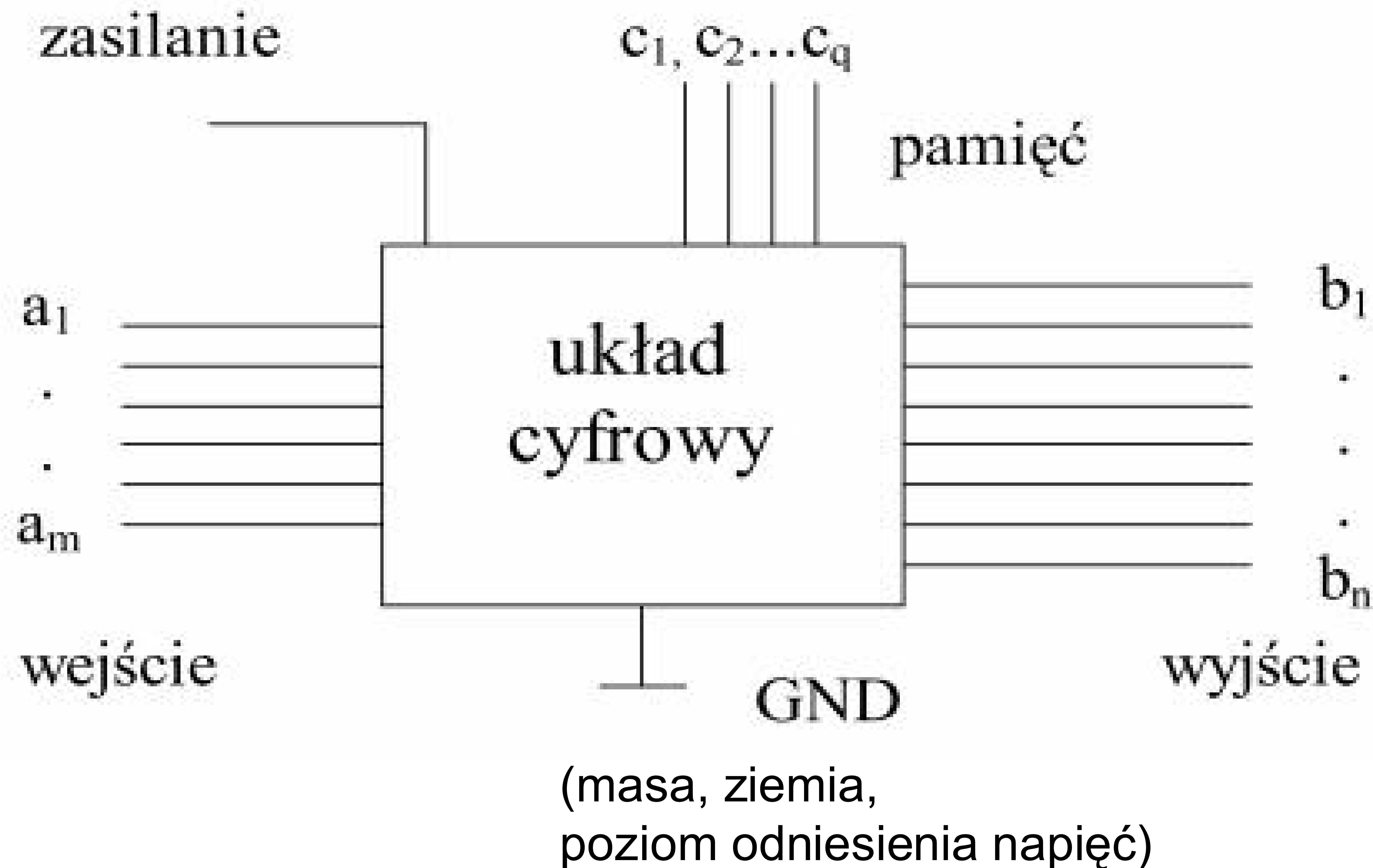
Do konwersji analogowo-cyfrowej wykorzystuje się specjalizowane układy scalone:

- przetworniki analogowo - cyfrowe (A/D - analog/digital)
- przetworniki cyfrowo – analogowe (D/A - digital/analog)

Przykłady urządzeń zawierających przetworników A/D i D/A:

- karta dźwiękowa w komputerze PC (A/D – wejścia karty, D/A – wyjścia karty)
- oscyloskop cyfrowy, miernik uniwersalny cyfrowy (A/D),
- generator funkcyjny cyfrowy (D/A)

Układy cyfrowe – do przetwarzania sygnałów cyfrowych



Układy cyfrowe – układ interpretuje napięcia jako jeden z dwu stanów

0 (niski L = Low) albo 1 (wysoki H = High)

i przetwarza takie sygnały dwustanowe zgodnie z zasadami **logiki matematycznej (algebry Boole'a)** oraz dwójkowego układu liczenia

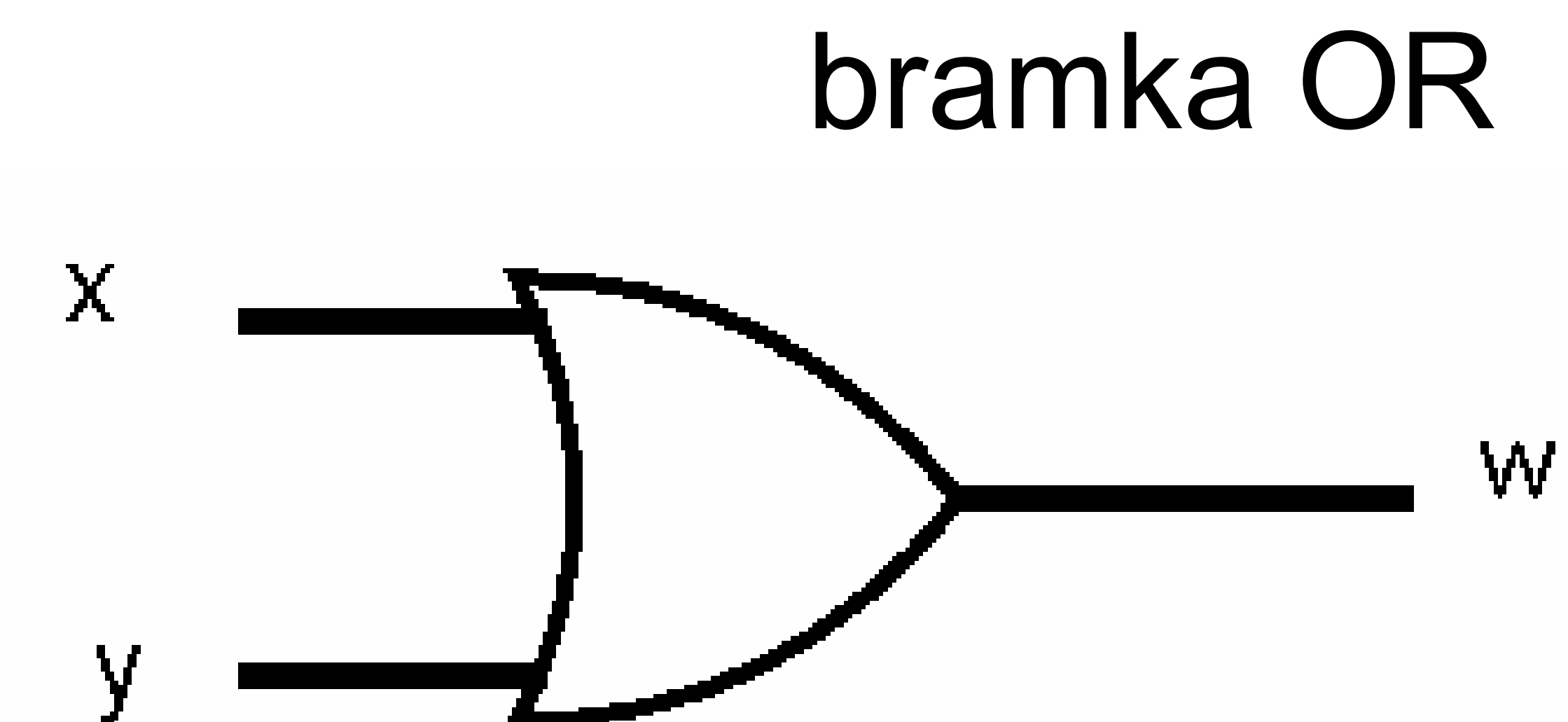
Podstawowe operacje logiczne, algebra Boole'a (= rachunek zdań)

- Aby wykonywać operacje matematyczne na liczbach dwójkowych wykorzystuje się, jako podstawowy element, operacje logiczne na pojedynczych bitach – przykłady będą dalej,
- Przyporządkowanie: 0 = stan L = False (fałsz)
 1 = stan H = True (prawda)
- Podstawowe działania (funkcje) logiczne:
suma logiczna (alternatywa): $x + y$ OR
iloczyn logiczny (koniunkcja): $x \cdot y$ AND
negacja: $\sim x = \bar{x}$ NOT
- Każdą operację logiczną można złożyć z tych podstawowych działań logicznych
- Układy cyfrowe są układami elektronicznymi wykonującymi funkcje logiczne i matematyczne w układzie dwójkowym

Podstawowe operacje logiczne – realizacje przez bramki cyfrowe

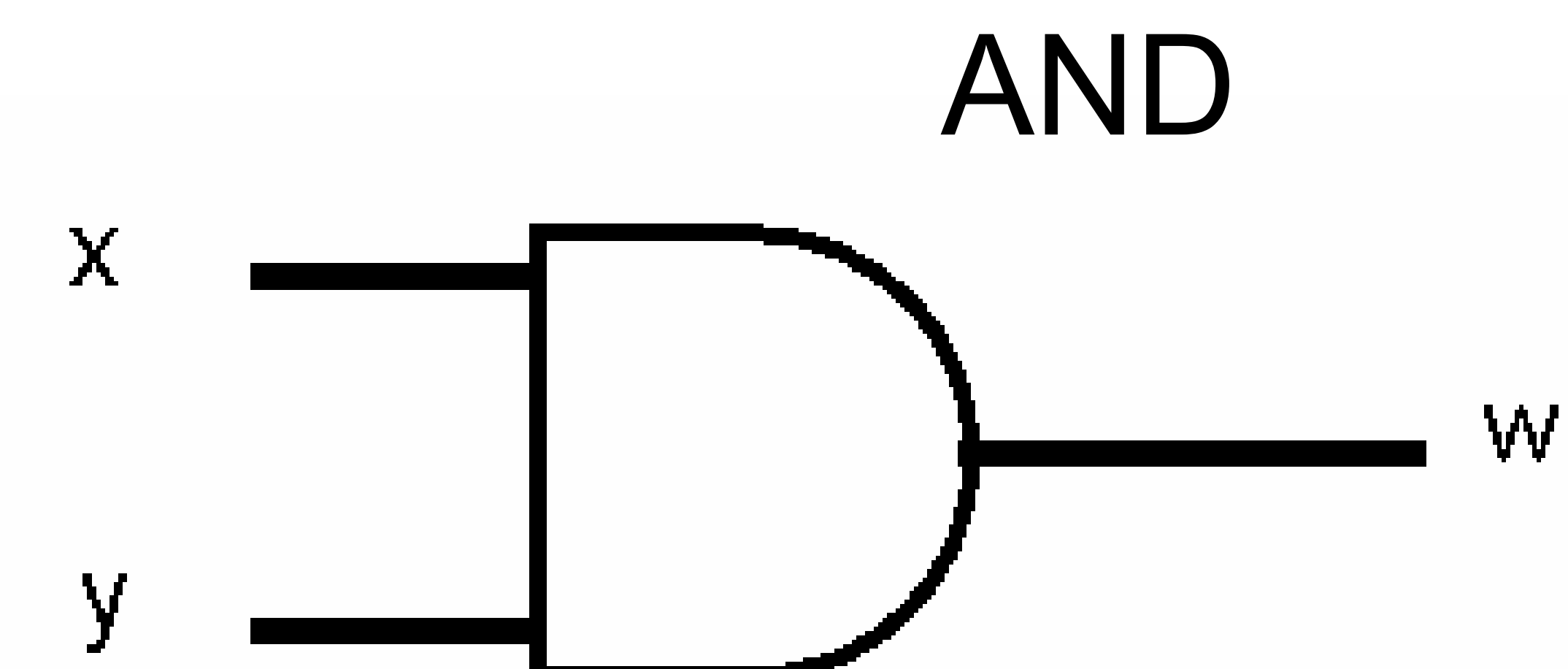
Suma logiczna
 $x + y$
alternatywa

x	y	$w = x + y$
0	0	0
1	0	1
0	1	1
1	1	1



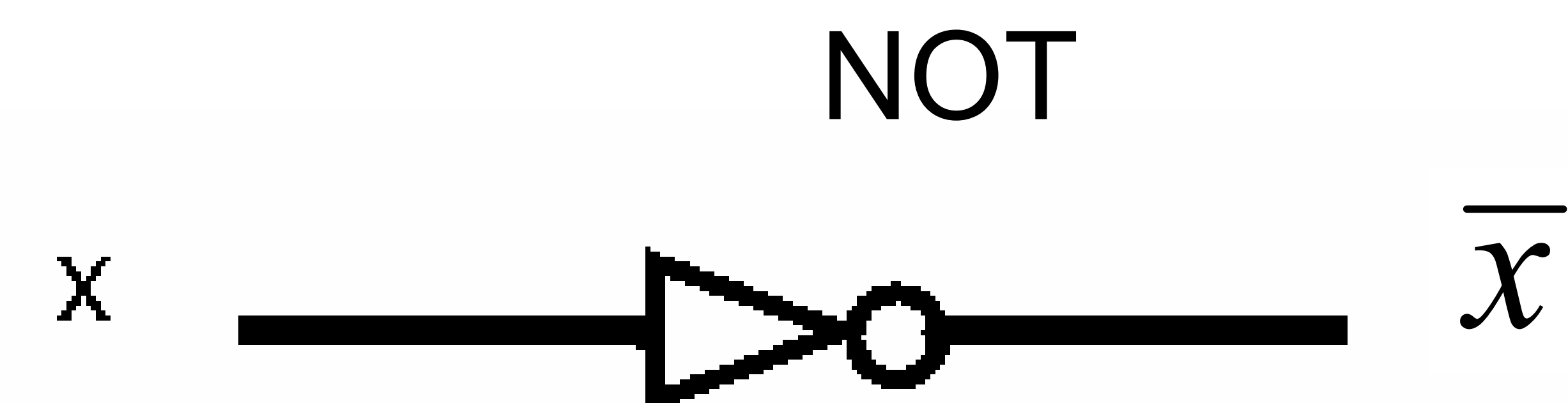
Iloczyn logiczny
 $x y$
koniunkcja

x	y	$w = xy$
0	0	0
1	0	0
0	1	0
1	1	1



Negacja $\bar{x}$

x	$\bar{x}$
0	1
1	0



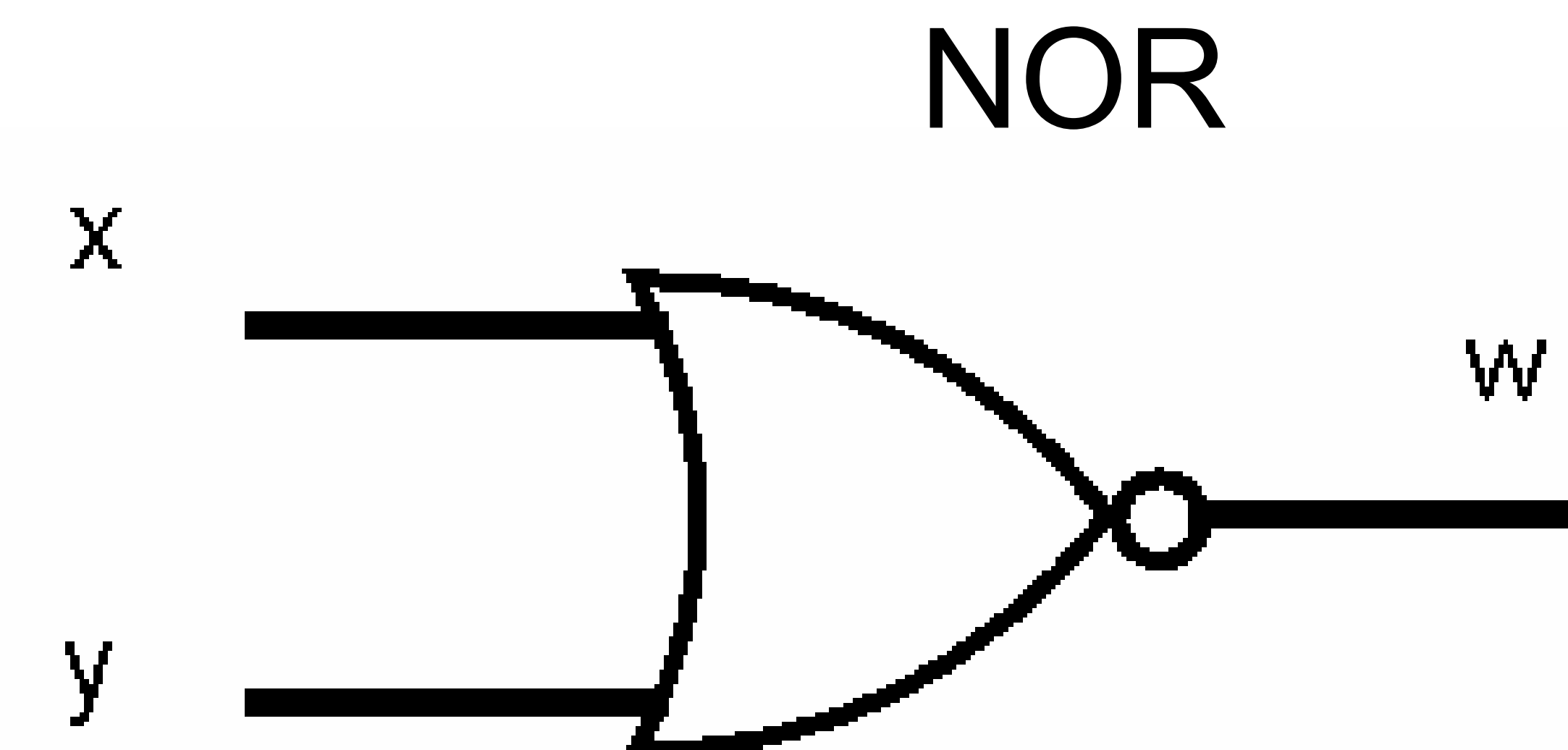
Inne użyteczne funkcje logiczne i ich bramki cyfrowe

Funkcja NOR

$$w = \overline{x + y} = \bar{x} \cdot \bar{y}$$

(zaprzeczona
alternatywa)

x	y	NOR
0	0	1
1	0	0
0	1	0
1	1	0

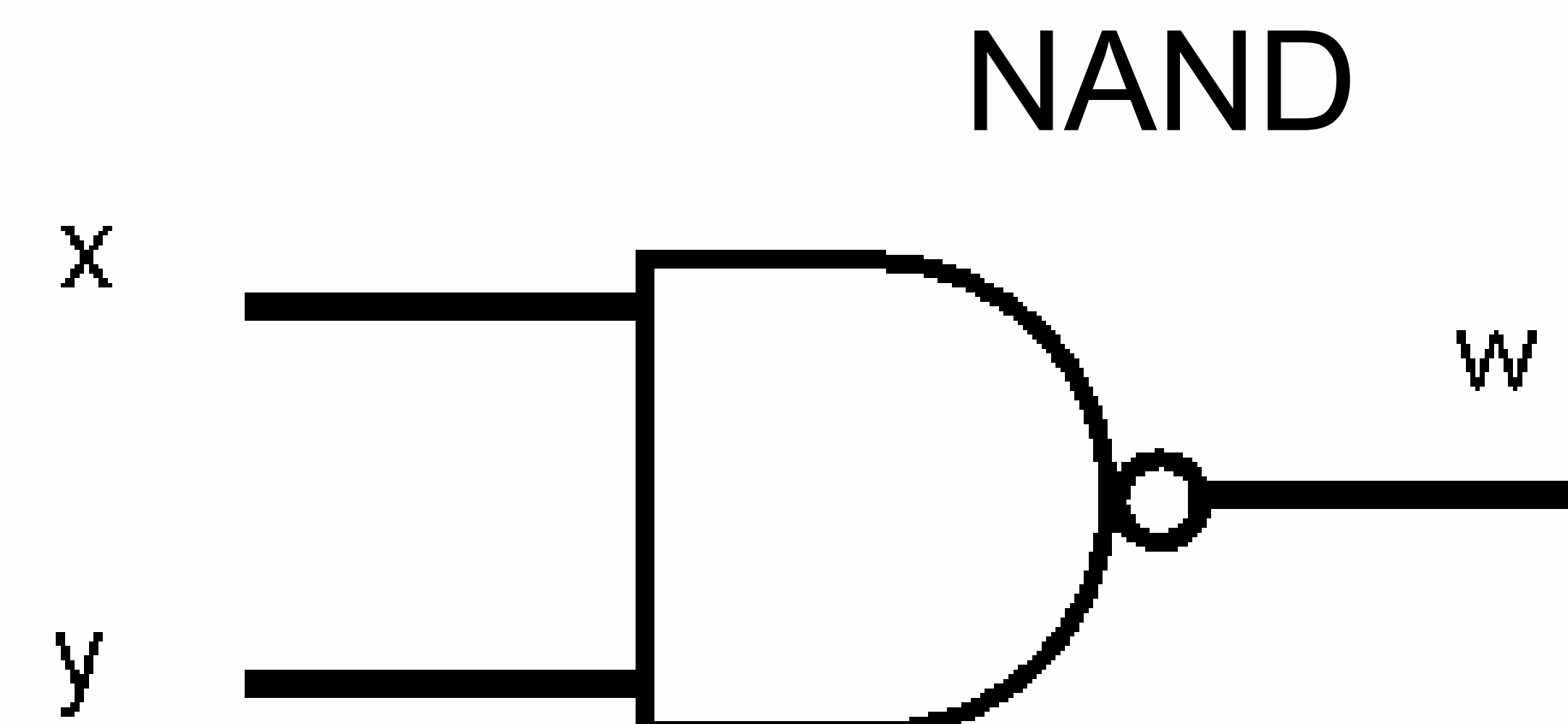


Funkcja NAND

$$w = \overline{xy} = \bar{x} + \bar{y}$$

(zaprzeczona
koniunkcja)

x	y	NAND
0	0	1
1	0	1
0	1	1
1	1	0

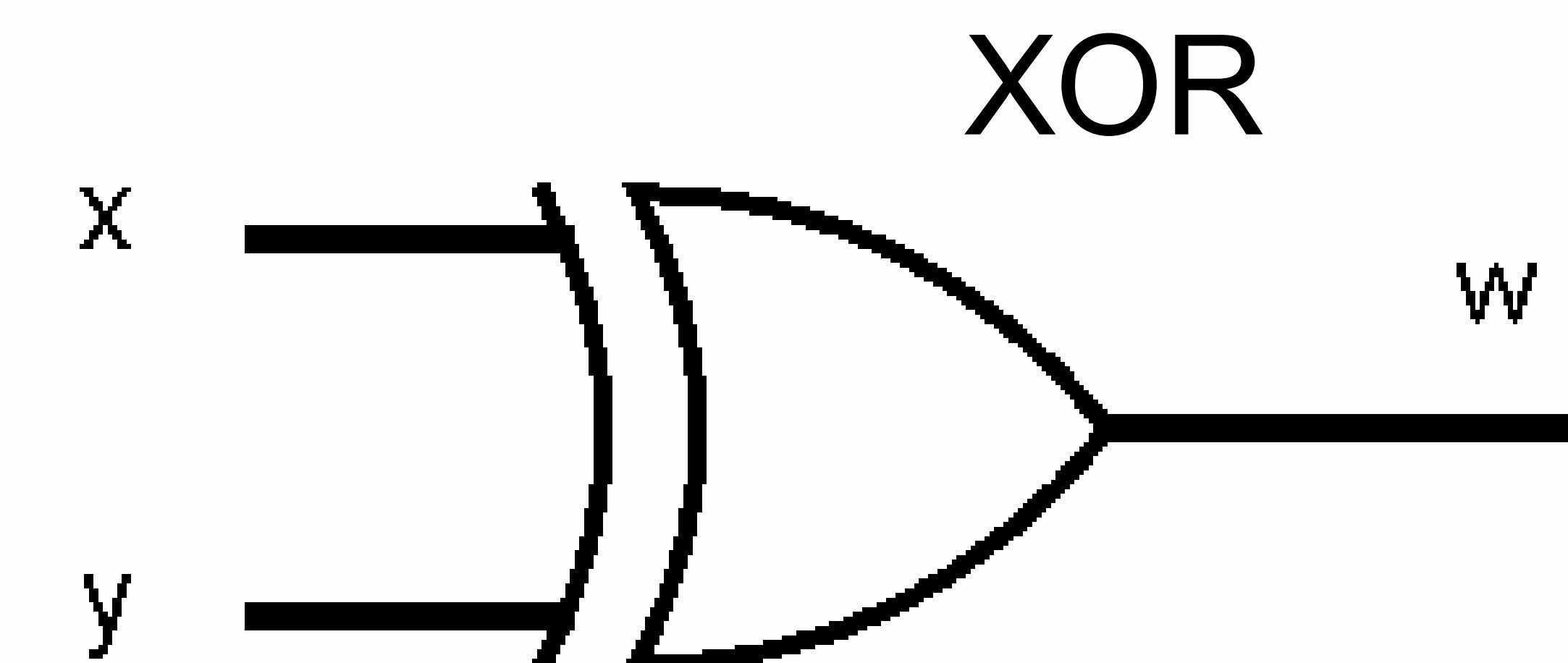


Exclusive OR = XOR

$$w = \bar{x}y + x\bar{y}$$

(alternatywa
wykluczająca)

x	y	XOR
0	0	0
1	0	1
0	1	1
1	1	0



Tożsamości rachunku zdań = algebry Boole'a

$$x + y = y + x$$

- Suma logiczna, alternatywa

$$xy = yx$$

- Iloczyn logiczny, koniunkcja

$$x(y + z) = xy + xz$$

- Rozdzielność koniunkcji

$$x + yz = (x + y)(x + z)$$

- Rozdzielność alternatywy !!!

$$x + \bar{x} = 1$$

$$x \cdot \bar{x} = 0$$

=

$$\bar{\bar{x}} = x$$

- Podwójna negacja

$$\overline{x + y} = \bar{x} \cdot \bar{y}$$

- Prawa de Morgana

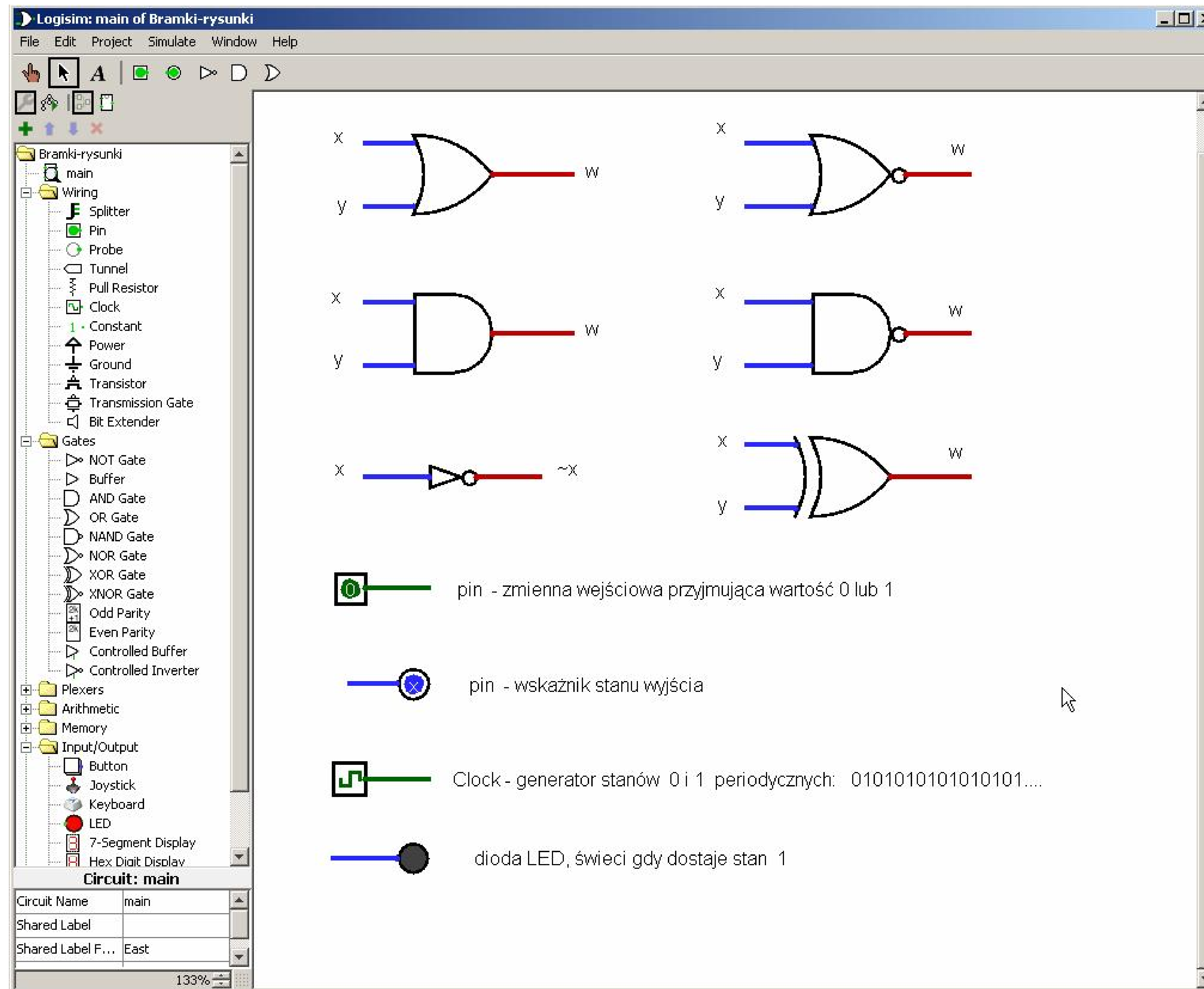
$$\overline{x \cdot y} = \bar{x} + \bar{y}$$

Własność funkcji logicznych

- Wszystkie funkcje logiczne, nawet najbardziej skomplikowane, daje się zrealizować z podstawowych funkcji logicznych, np:
 - 1) tylko z AND oraz NOT
 - 2) tylko z NAND
 - 3) tylko z NOR
 - 4) tylko z XOR oraz AND

Aby to udowodnić należy wykazać, że z danego zestawu da się złożyć pozostałe podstawowe funkcje logiczne.

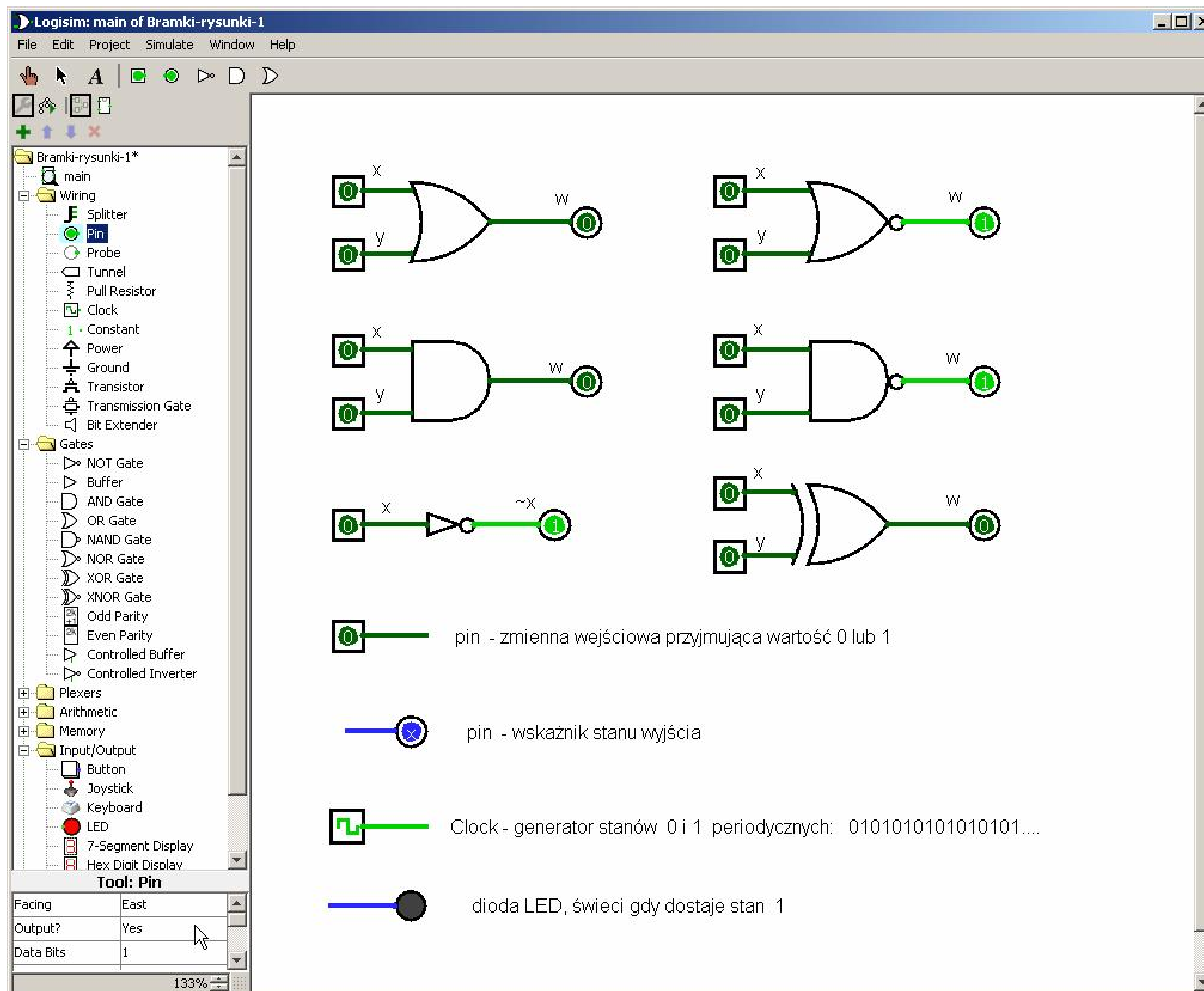
Jak to pracuje – symulator układów logicznych *Logisim*



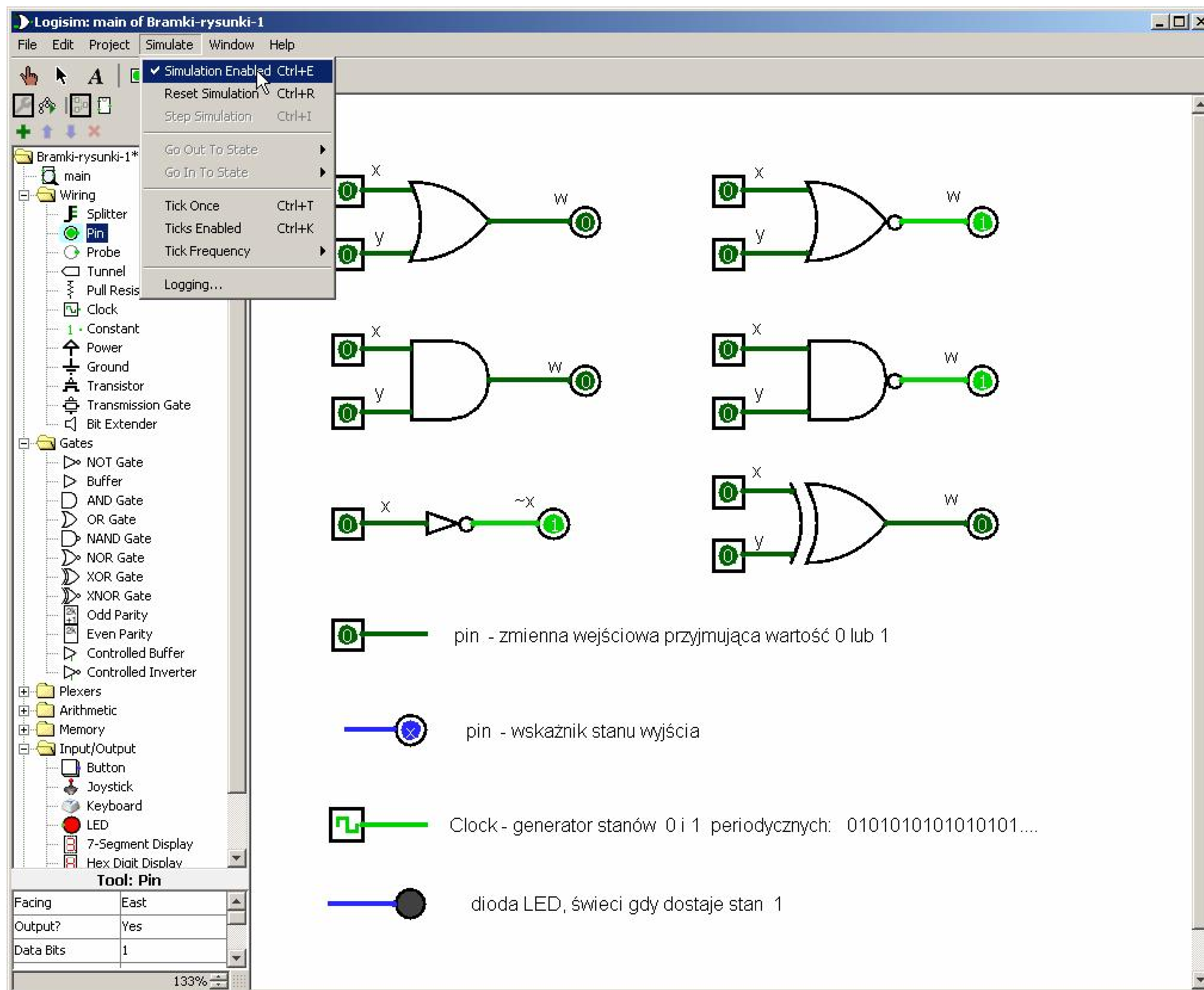
<http://www.cburch.com/logisim/> - program na licencji GPL (free !!!)

wymaga zainstalowania Java RE ver. 5 lub późniejszej (np. dla WinXP plik: jre-7u55-windows-i586.exe)

Java RE do pobrania np. stąd: <https://www.java.com/pl/download/>



Plik przykładu w programie Logisim: [p0_Bramki-rysunki-1.circ](#)



- uruchomienie modu symulacji z menu Simulation → Simulation Enabled,
- stany zmiennych wejściowych (Pin → Output? No) zmienia się wskaźnikiem „rączka”

Przykłady

1. Sprawdzić tabelki prawdy dla podstawowych bramek

pliki symulacji Logisim: [p0_Bramki-rysunki-1.circ](#) , [p1_OR_AND_NAND_NOR.circ](#)

2. Zbudować układ cyfrowy realizujący sumę logiczną $x + y$ używając wyłącznie bramek NAND, sprawdzić rezultat tabelką prawdy i przez porównanie wyniku z bramką OR

- należy skorzystać z prawa de Morgana

plik symulacji: [p1c_Suma_NAND-1.circ](#)

$$x + y = \overline{\overline{x + y}} = \overline{\overline{x} \cdot \overline{y}}$$

3. Zbudować dekodery („wykrywacz”) liczby 7, (potem także 7 i 9) spośród liczb 4-bitowych

pliki symulacji: [p2_Dekoder_liczby-7.circ](#) , [p2b_Dekoder_7-9_BCD.circ](#)

4. Zbudować układ cyfrowy obsługujący 7-segmentowy wyświetlacz jednej cyfry dziesiętnej kodowanej BCD, czyli wyświetlający cyfrę dziesiętną z jej kodu BCD - nieco trudne zadanie !

pliki symulacji: [p3_7-segment-display_1.circ](#) , [p3a_7-segment-display_2_caly-uklad-2.circ](#)

5. Zbudować układ sumujący dwie liczby 3 lub 4 bitowe - także nieco trudne zadanie !

pliki symulacji: [p4_Sumator-3-bitowy.circ](#) , [p4a_Sumator-4-bitowy.circ](#)

ad Przykład 3:

CYFRA	BCD
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

Funkcja ma wykrywać liczby 7 i 9, czyli ma dawać w wyniku wartość 1 wtedy i tylko wtedy, gdy 4 bity na wejściu tej funkcji, czyli 4 bity DCBA kodu BCD, kodują cyfrę 7 lub cyfrę 9.

Fragment **tabelki prawdy** (wypisać całość !)

	wejścia D	C	B	A	wyście w
7	0	1	1	1	1
9	1	0	0	1	1

Szukana funkcja:

$$w = \overline{D}CBA + D\overline{C}\overline{B}A = (\overline{D}CB + D\overline{C}\overline{B})A$$

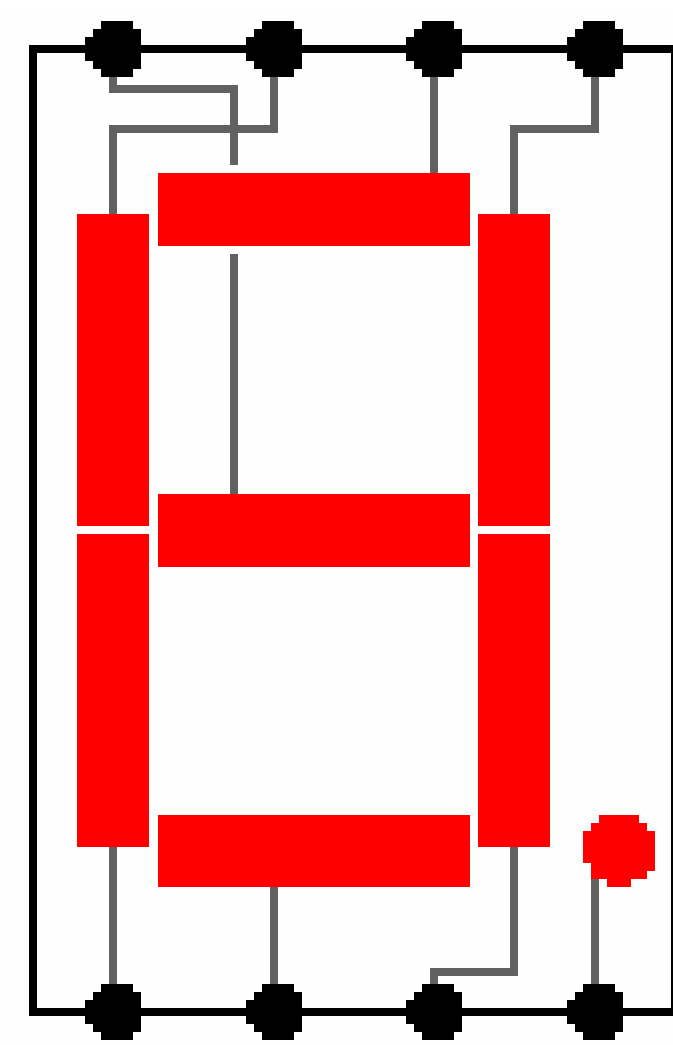
Funkcję tę wypisuje się jako sumę składników będących iloczynem wejść lub zaprzeczeń wejść w taki sposób, aby każdy iloczyn dawał wartość $w=1$ dla pożądaných wartości wejść. Z własności koniunkcji wynika, że dla innych wartości wejść będzie $w=0$. Funkcję następnie należy **zminimalizować (uproszczyć)** korzystając z tożsamości rachunku zdań – algebry Boola – lub z ich graficznego przedstawienia – tablic (map) Karnaugh'a.

- Najłatwiej zrealizować tę funkcję korzystając m.in. z 3-wejściowych bramek AND

Zadanie domowe: jak ją zrealizować korzystając wyłącznie z bramek 2-wejściowych?

ad Przykład 4:

w5w8



w1 w4

- wyświetlacz 7-segmentowy (ósme pole to kropka dziesiętna)

Chcemy sterować nim, tzn. wyświetlać cyfry dziesiętne kodowane jako 4 bity kodu BCD (przykład zrealizowano dla cyfr szesnastkowych).

Wyjścia: w1, w2, ...w8 - kolejne pola świecące wyświetlacza, czyli poszukujemy 8 tych funkcji

(pole wyświetlacza świeci, gdy dostaje stan 1, gaśnie gdy stan 0)

Wejścia: D C B A - 4 bity kodu BCD

Tabela prawdy 8 funkcji

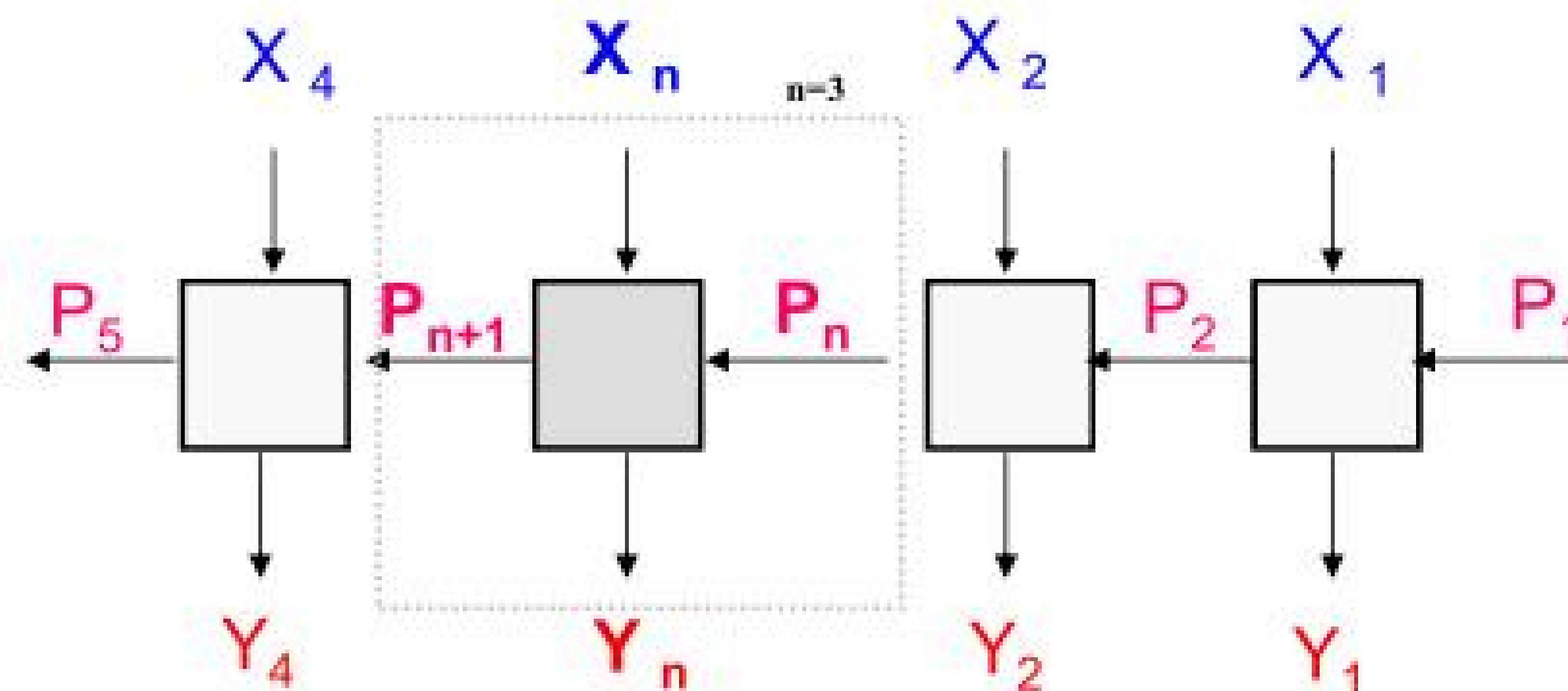
w1..w8 wypełniona na podstawie rozpoznania, które pola wyświetlacza świecą dla kolejnych cyfr szesnastkowych 0, 1, ...9,A,...,F

Logisim pozwala na podstawie tabelki prawdy wyznaczyć funkcje i zaproponować układ bramek !!!

cyfra	D	C	B	A	w1	w2	w3	w4	w5	w6	w7	w8
0	0	0	0	0	1	1	1	0	0	1	1	1
1	0	0	0	1	0	0	1	0	0	0	0	1
2	0	0	1	0	1	1	0	0	1	0	1	1
3	0	0	1	1	0	1	1	0	1	0	1	1
4	0	1	0	0	0	0	1	0	1	1	0	1
5	0	1	0	1	0	1	1	0	1	1	1	0
6	0	1	1	0	1	1	1	0	1	1	1	0
7	0	1	1	1	0	0	1	0	0	0	1	1
8	1	0	0	0	1	1	1	0	1	1	1	1
9	1	0	0	1	0	1	1	0	1	1	1	1
A	1	0	0	1	1	0	1	0	1	1	1	1
B	1	0	0	1	1	1	1	0	1	1	0	0
C	1	0	0	1	1	1	0	0	0	1	1	0
D	1	0	0	1	1	1	1	0	1	0	0	1
E	1	0	0	1	1	1	0	0	1	1	1	0
F	1	0	0	1	1	0	0	0	1	1	1	0

ad. Przykład 5 Półsumator, sumator (sumowanie liczb dwójkowych)

Układy arytmetyczne (układy iteracyjne)



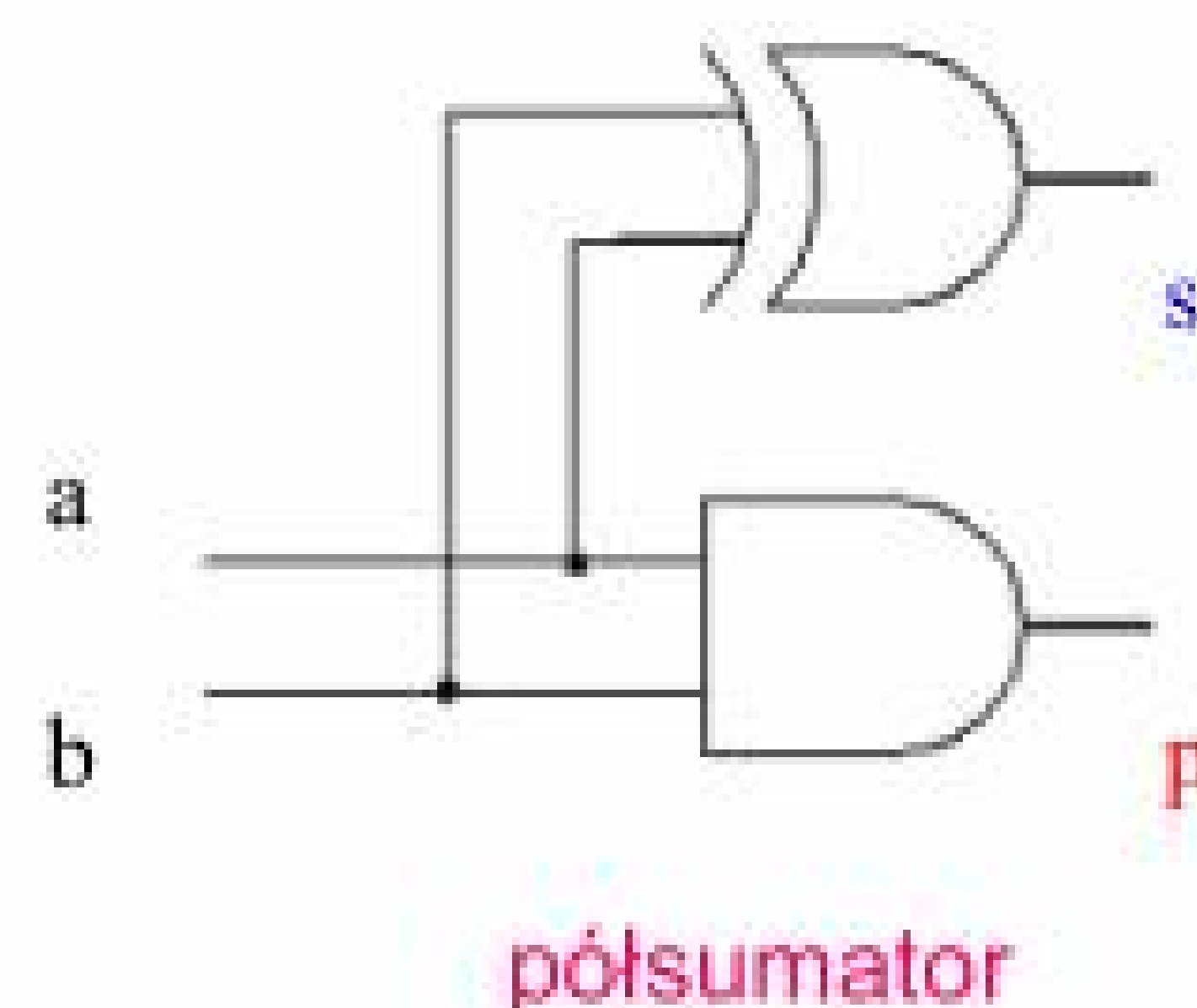
Słowo logiczne: liczba zapisana w danym **kodzie binarnym**.

Na przykład: słowo **(1011)** = liczba **11** = $1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0$

Układy cyfrowe → operacje arytmetyczne na liczbach (słowach logicznych)

Półsumator - układ dodający dwie liczby jednobitowe **a** i **b**

Wynik: liczba dwubitowa - suma **s** i przeniesienie **p**



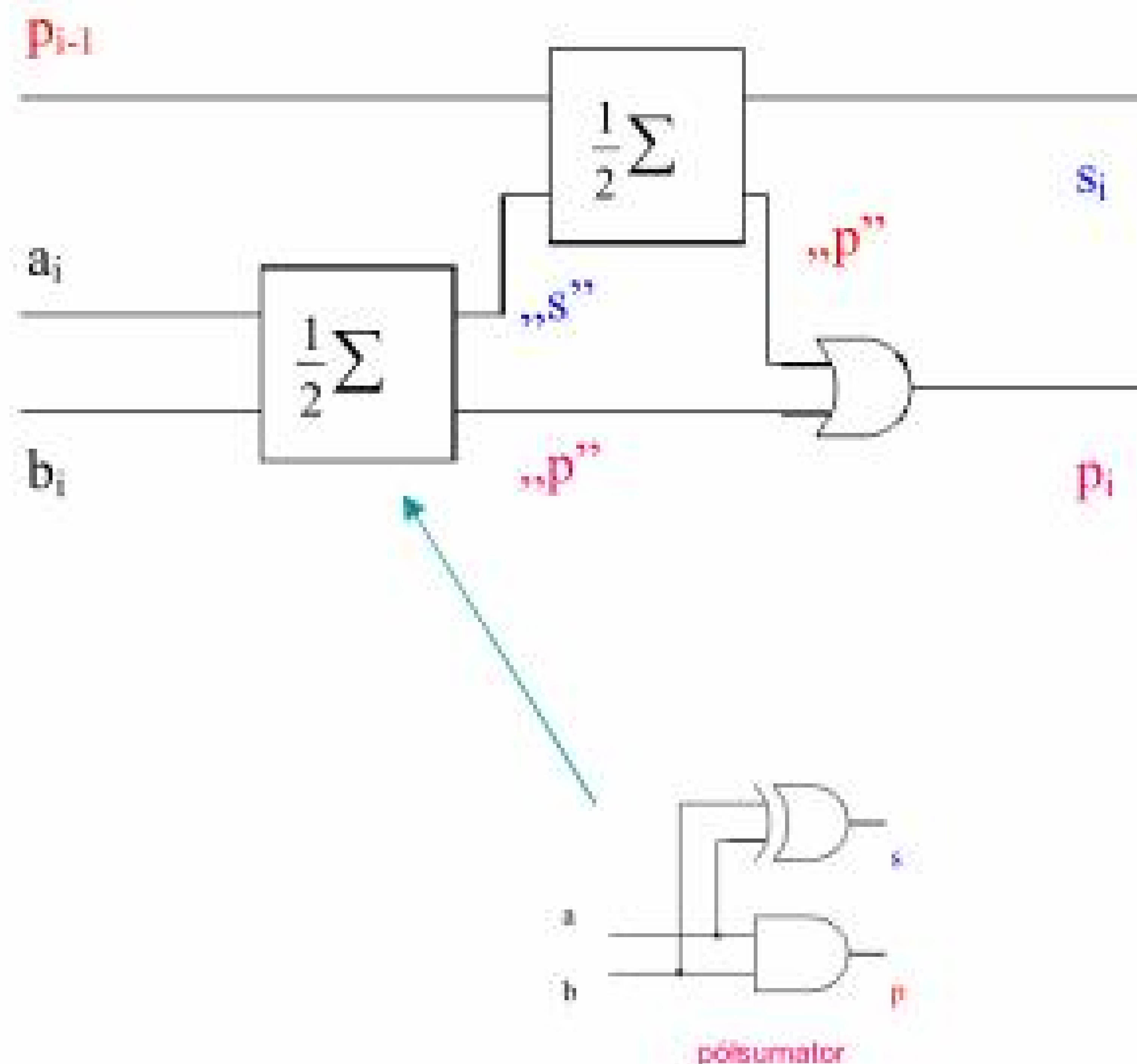
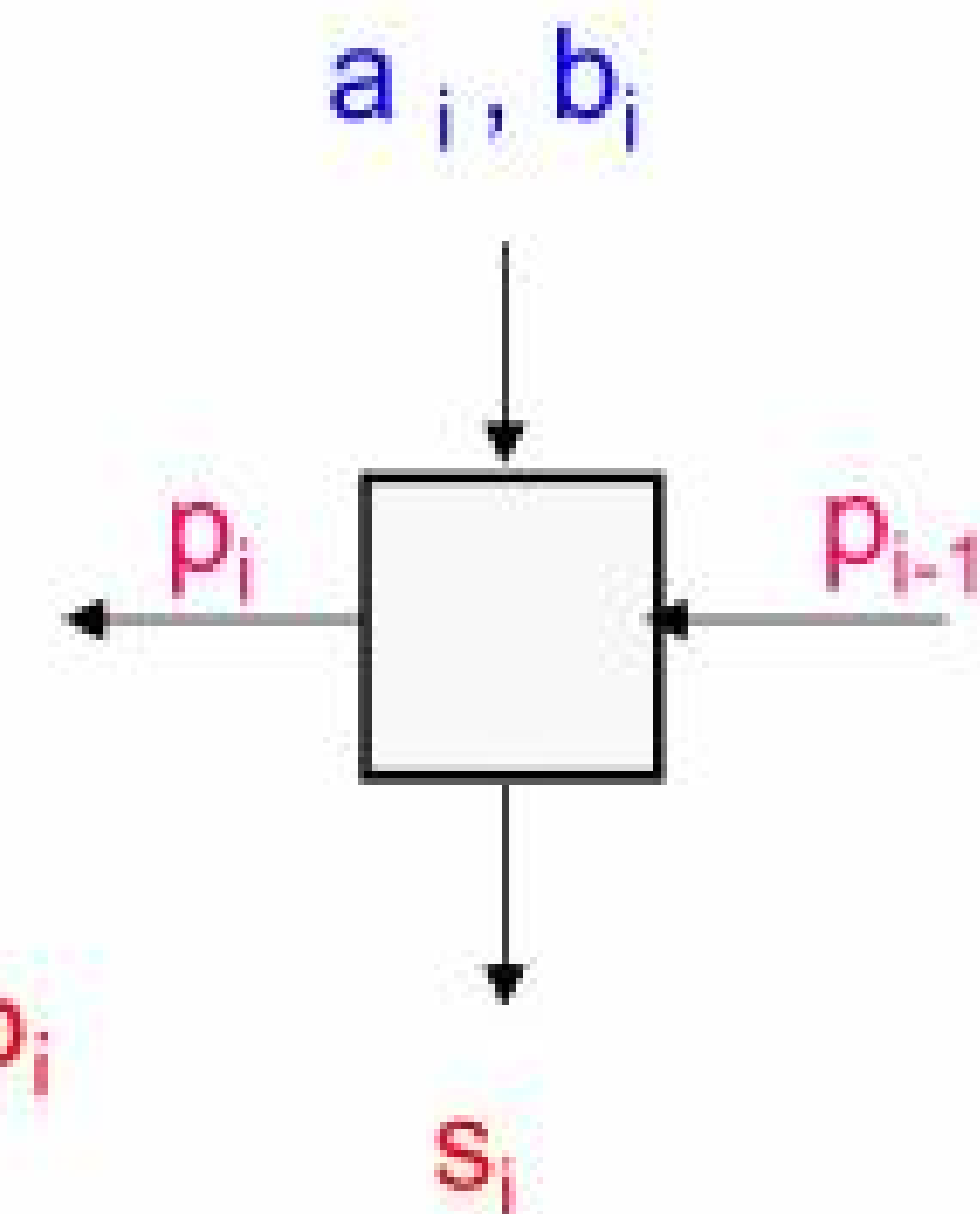
s - funkcja EXOR
p - funkcja AND

a	b	s	p
0	0	0	0
1	0	1	0
0	1	1	0
1	1	0	1

Sumator jednobitowy

Układ iteracyjny:

- sumowanie a_i i b_i na i -tej pozycji
- uwzględnia przeniesienie z pozycji p_{i-1}
- generuje sumę s_i i przeniesienie na pozycję następną p_i



a_i	b_i	p_{i-1}	s_i	p_i
0	0	0	0	0
1	0	0	1	0
0	1	0	1	0
0	0	1	1	0
1	1	0	0	1
0	1	1	0	1
1	0	1	0	1
1	1	1	1	1

Realizacje w układach scalonych serii TTL lub HC

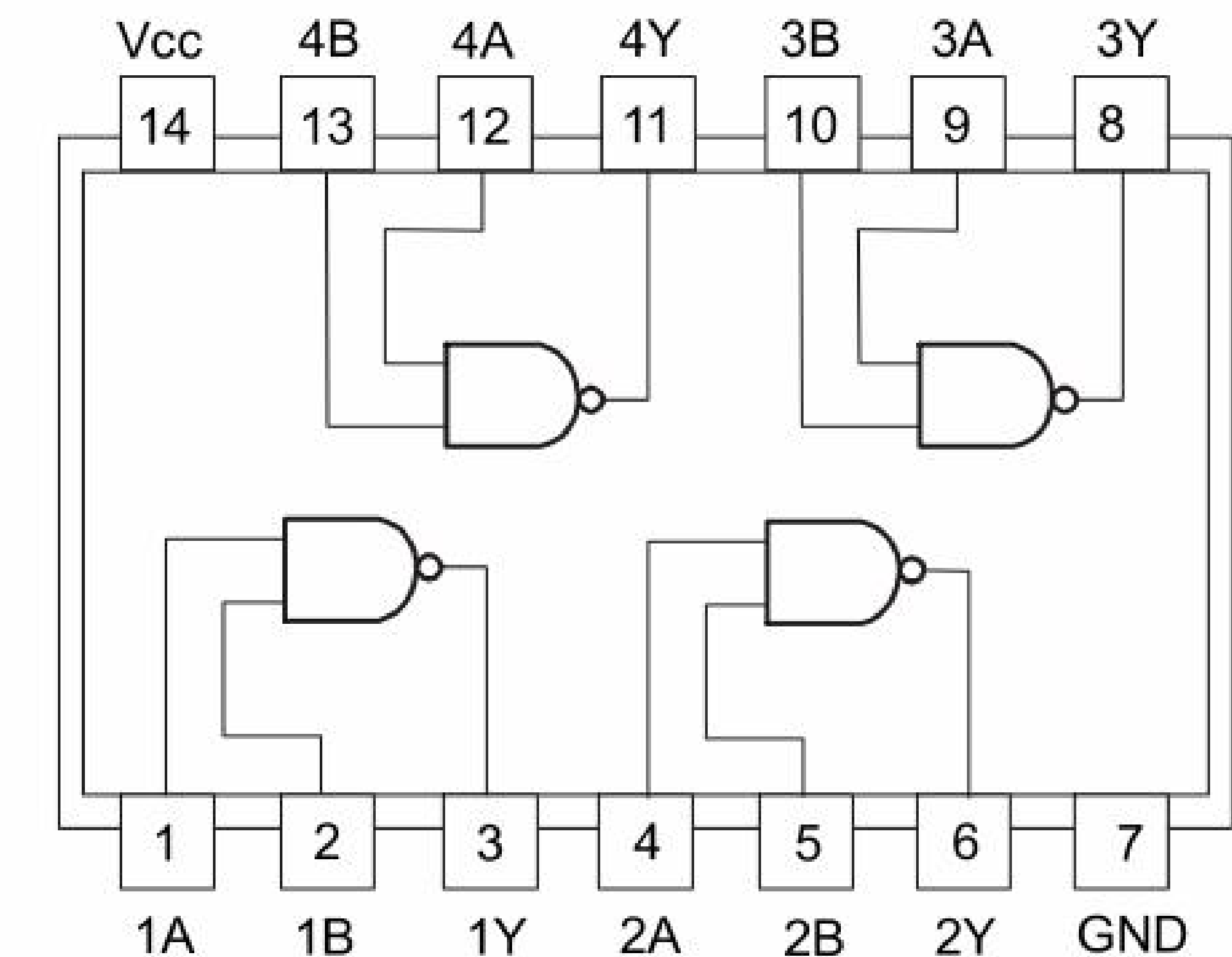


QUADRUPLE 2-INPUT POSITIVE-NAND GATES

7400

positive logic

$$Y = \overline{AB}$$

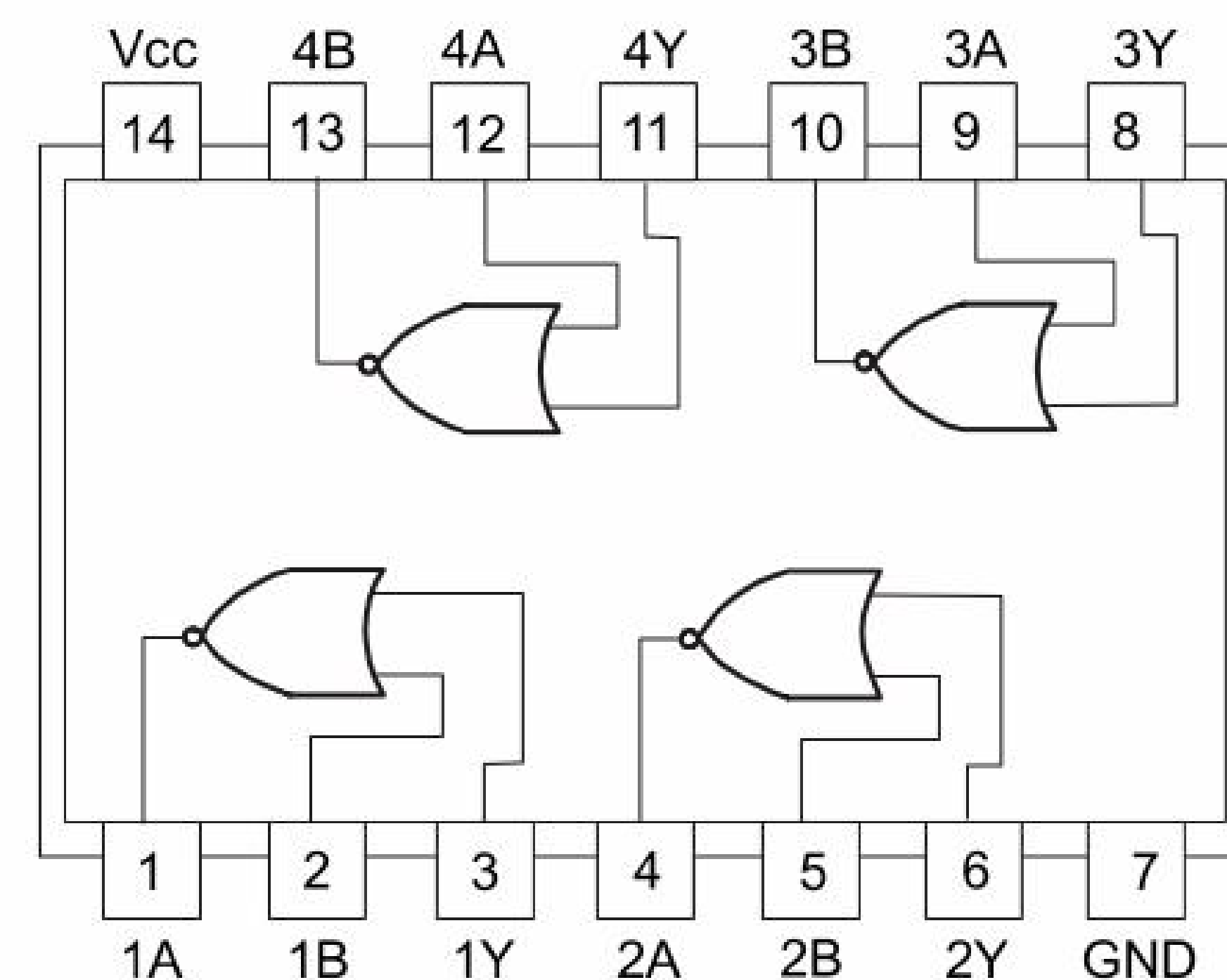


QUADRUPLE 2-INPUT POSITIVE-NOR GATES

7402

positive logic

$$Y = \overline{A+B}$$

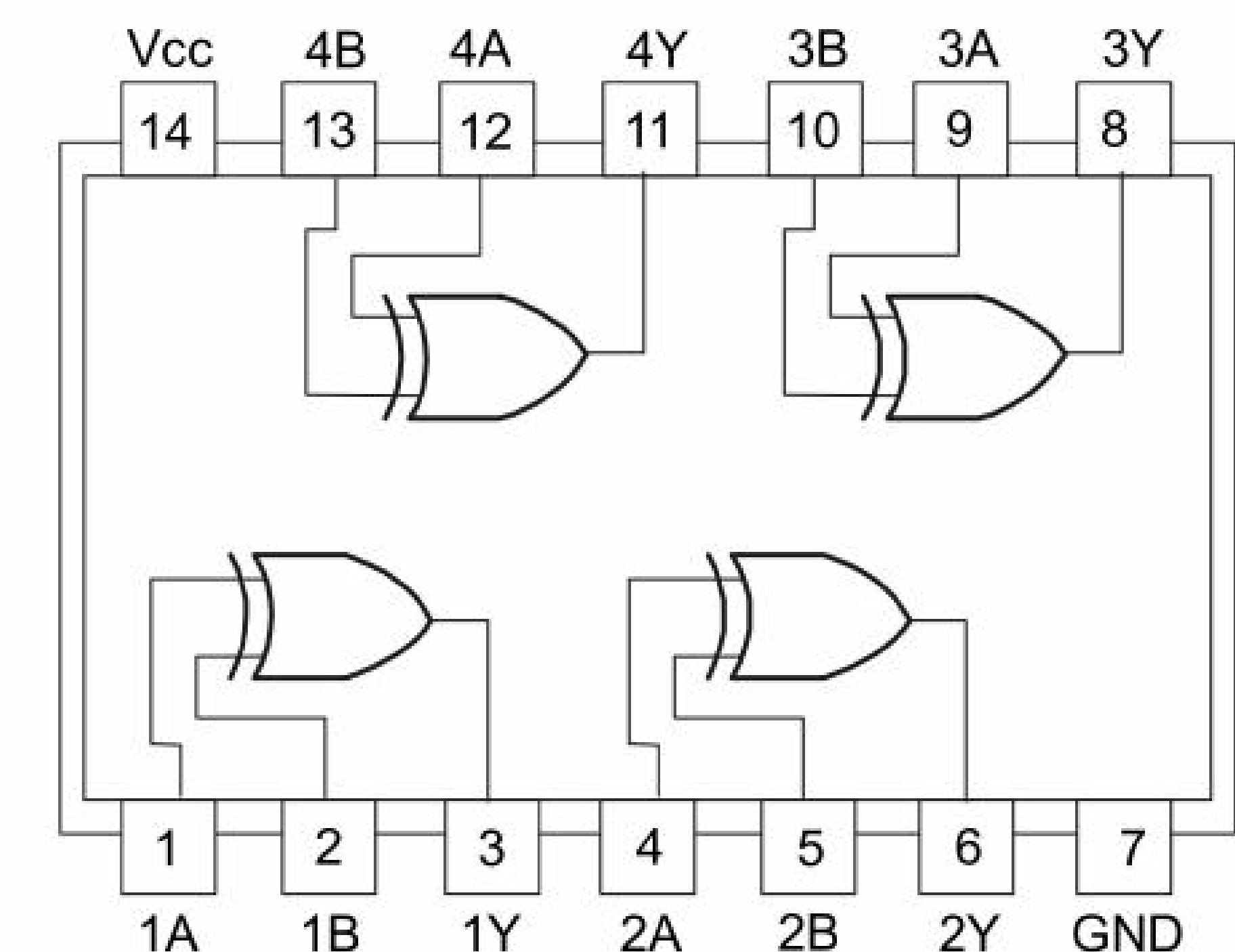


QUADRUPLE 2-INPUT EXCLUSIVE-OR GATES

7486 $Y = A \oplus B = \overline{A}B + A\overline{B}$

FUNCTION TABLE

INPUTS		OUTPUT
A	B	Y
L	L	L
L	H	H
H	L	H
H	H	L



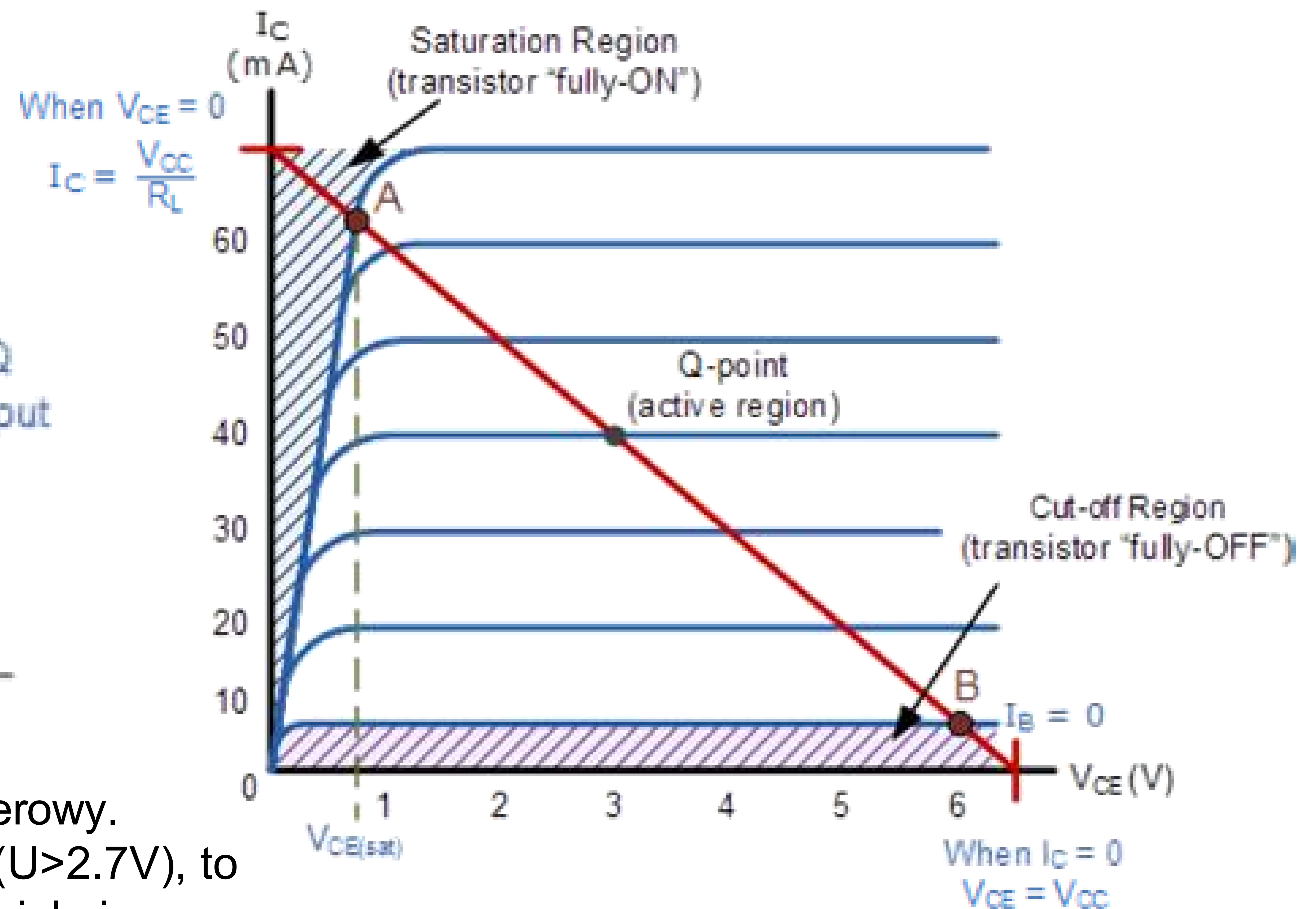
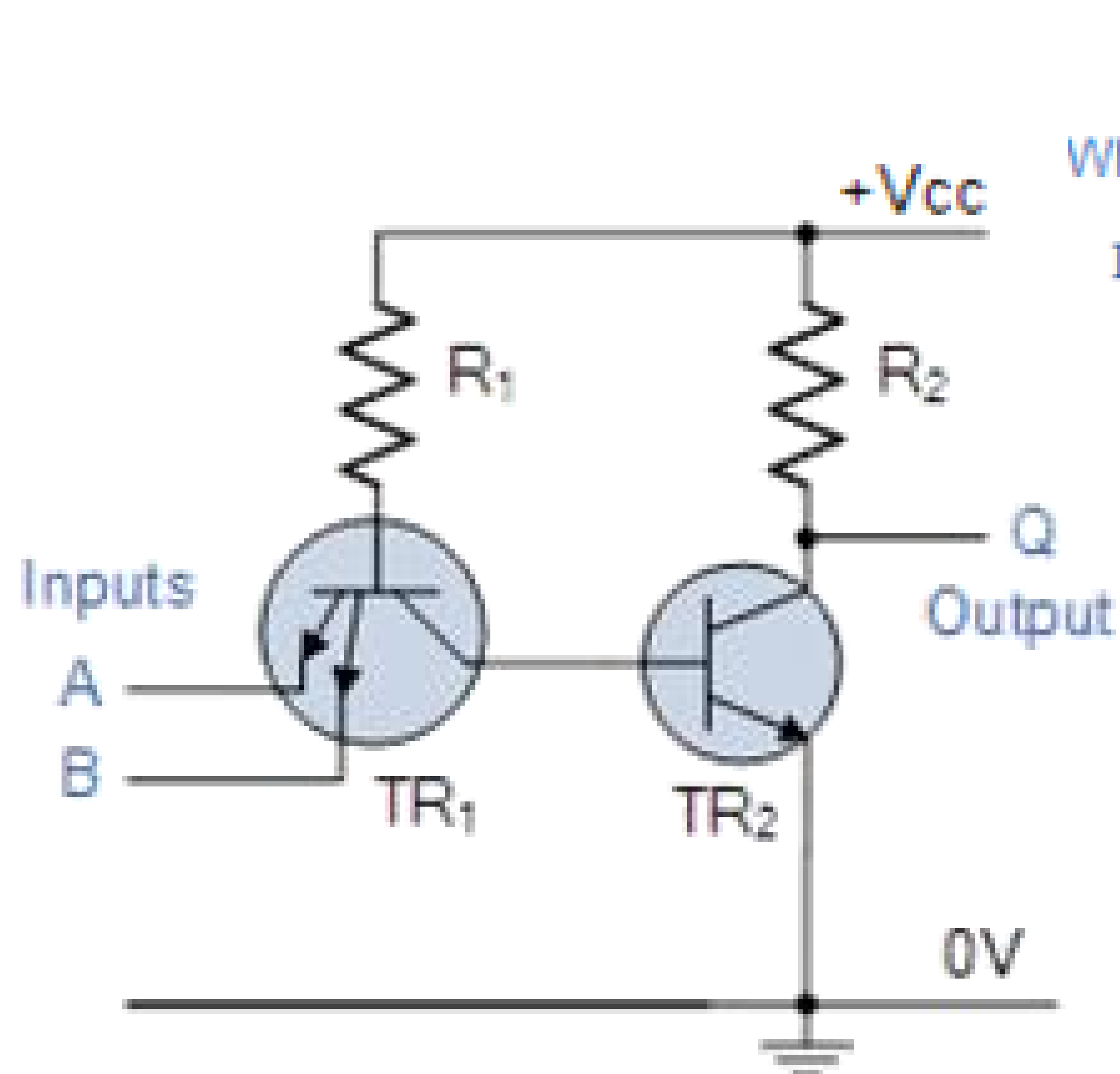
Zasilanie układów scalonych TTL / HC: $V_{cc} = +5V$, $GND = 0V$

Zasady budowania układów cyfrowych z bramek TTL lub HC

TTL = Transistor – Transistor Logic

- Układy scalone TTL i HC są zasilane napięciem 5V +/- 0.25V
- Logicznemu zeru odpowiada napięcie 0 - 0.4V (0 – 0.8V dla wejść)
- Logicznej jedynce odpowiada napięcie 2.7 – 5V (2 – 5V dla wejść)
- Niepodłączone wejście bramki przyjmuje samoistnie stan 1 (tylko dla TTL)
- W układach serii HC niepodłączone wejścia mogą przyjmować stany nieokreślone – WNIOSEK: **nie można** zostawiać wejść z niezdefiniowanymi stanami (stany wszystkich wejść należy definiować przez odpowiednie połączenie do masy, napięcia zasilania 5V lub odpowiednich wyjść)
- Nie wolno łączyć ze sobą wyjść bramek – możliwe uszkodzenie układu !!!
- Średni czas przenoszenia sygnału przez bramkę TTL wynosi 1-30 ns
- Przed przyłączeniem napięć (zasilacz, generator) do układu należy upewnić się, że wartości tych napięć są zgodne ze standardem TTL / HC (max 5V) !!!

Jak są zbudowane układy TTL – na przykładzie 2-wejściowej bramki NAND (schemat uproszczony)



Na wejściu jest tranzystor 2-emiterowy. Jeśli wejścia A i B w stanach "1" ($U > 2.7V$), to Baza TR1 jest na wysokim potencjale i złącze BC TR1 przewodzi prąd otwierając TR2, czyli wyjście $Q = "0"$. Jeśli choć jedno z wejść A lub B jest w stanie "0", to Baza TR1 jest na niskim potencjale (złącze EB w stanie przewodzenia), więc przez Colector TR1 nie płynie prąd i TR2 EC nie przewodzi, czyli $Q = "1"$.

Tranzystory bipolarne wykorzystane jako przyrządy przełączające, pracujące albo w stanie odcięcia (cut-off), albo w stanie nasycenia = otwarcia (fully-ON)

Produkowane obecnie rodziny układów cyfrowych

dominują tranzystory CMOS (unipolarne): niższe napięcia zasilania, mniejsze moce, większe częstotliwości pracy, prostsza produkcja

PRODUCT LIFE CYCLE



z firmy Texas Instruments,

<http://logic.ti.com>

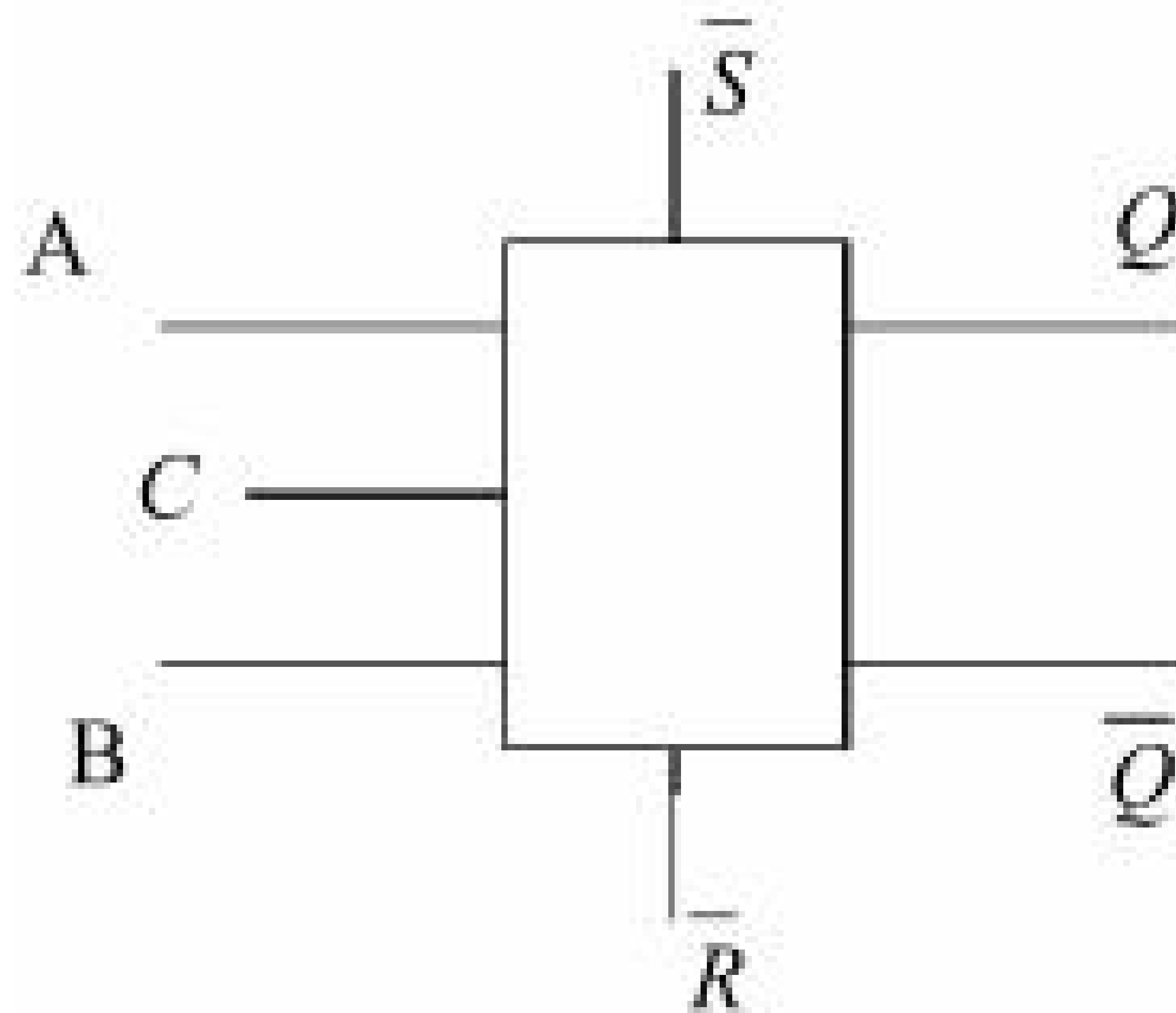
Układy sekwencyjne - przerzutniki

- Bramki logiczne – przykłady **układów kombinacyjnych**, dla których stan wyjść w danej chwili zależy tylko od stanu wejść **w tej samej chwili**
- **Układy sekwencyjne** – stan wyjść zależy od stanu wejść w chwili **taktów zegara**, także może zależeć od stanu wyjść w poprzednich chwilach.

Ustalenie stanu wyjść następuje w chwili taktu zegara.

Takt zegara to zmiana stanu wejścia zegarowego CLK z 0 na 1

Przerzutniki (ang. flip-flop) – najprostsze układy **pamiętające** swój stan



A, B – wejścia danych (0 lub 1)

C= CLK clock - wejście synchronizacyjne (zegarowe)

Q – wyjście,

$\sim Q$ – zanegowany stan wyjścia Q

R, S – wejścia asynchroniczne (wymuszają **natychmiastową** zmianę stanu Q)

Reset (wymusza $Q=0$), Set (wymusza $Q=1$)

Przy nieaktywnych wejściach asynchronicznych stan wyjścia Q jest ustalany w chwili taktu zegara CLK (takt zegara to zmiana stanu CLK z 0 na 1).

Miedzy taktami zegara przerzutnik **pamięta** swój stan – wyjście Q nie zmienia się niezależnie od stanów wejść A i B.

Przerzutniki T, D, J-K, S-R

n - kolejne takty zegara, tzn. zmiany stanu wejścia CLK z 0 na 1

T	Q_{n+1}
0	Q_n
1	$\sim Q_n$

- Przy każdym takcie zegara przerzutnik T zmienia wartość Q na przeciwną jeśli T=1, lub pozostawia niezmienną jeśli T=0.

D	Q_{n+1}
0	0
1	1

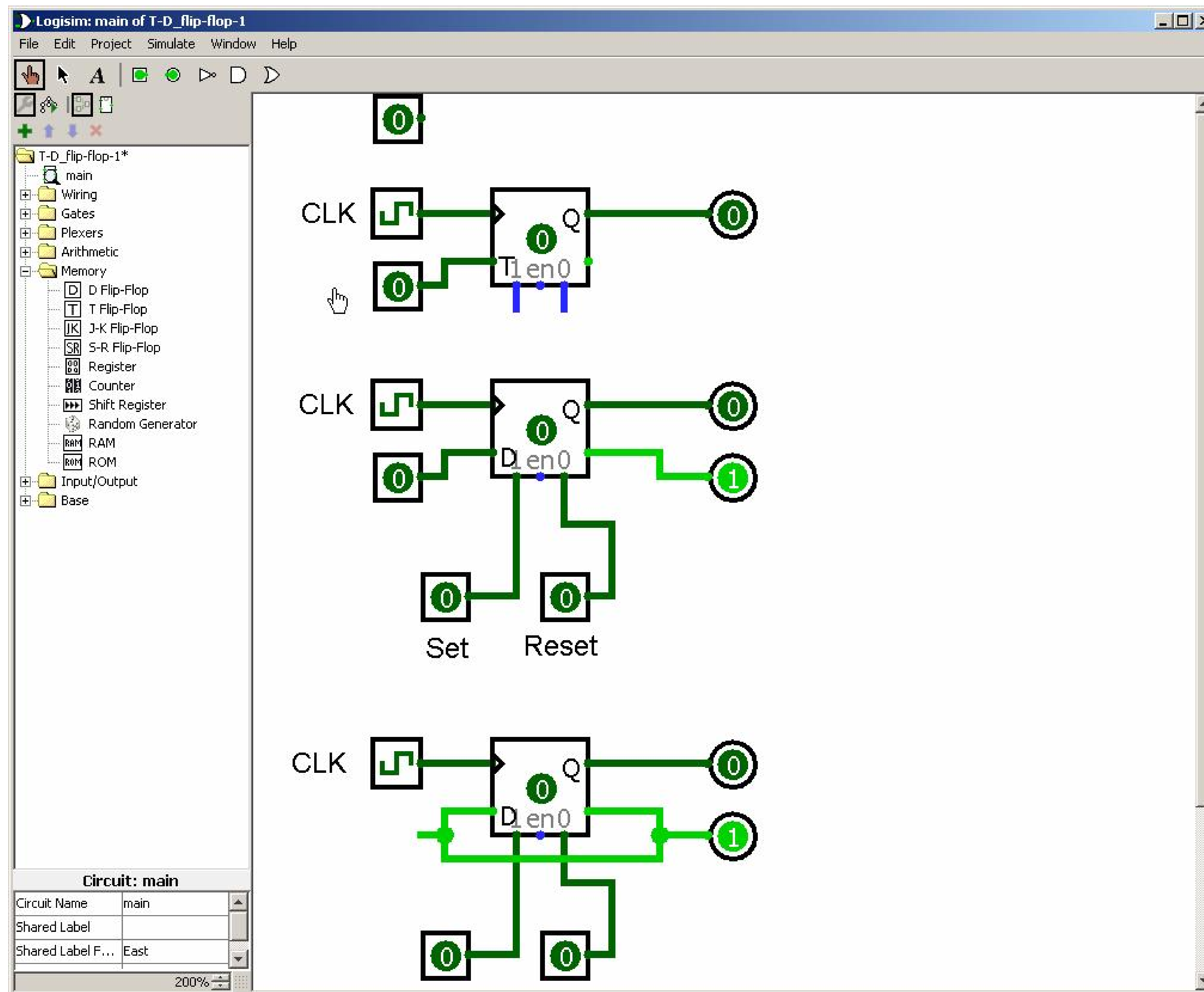
- Przy każdym takcie zegara przerzutnik D ustala wartość Q równą wartości wejścia D

J	K	Q_{n+1}
0	0	Q_n
0	1	0
1	0	1
1	1	$\sim Q_n$

S	R	Q_{n+1}
0	0	Q_n
0	1	0
1	0	1
1	1	???

Pliki symulacji: [p5_przerzutniki-T-D.circ](#), [p5a_T-D_flip-flop-1.circ](#)

Jak to pracuje – symulator *Logisim*



Plik symulacji: [p5a_T-D_flip-flop-1.circ](#)

Realizacje w układach scalonych serii TTL

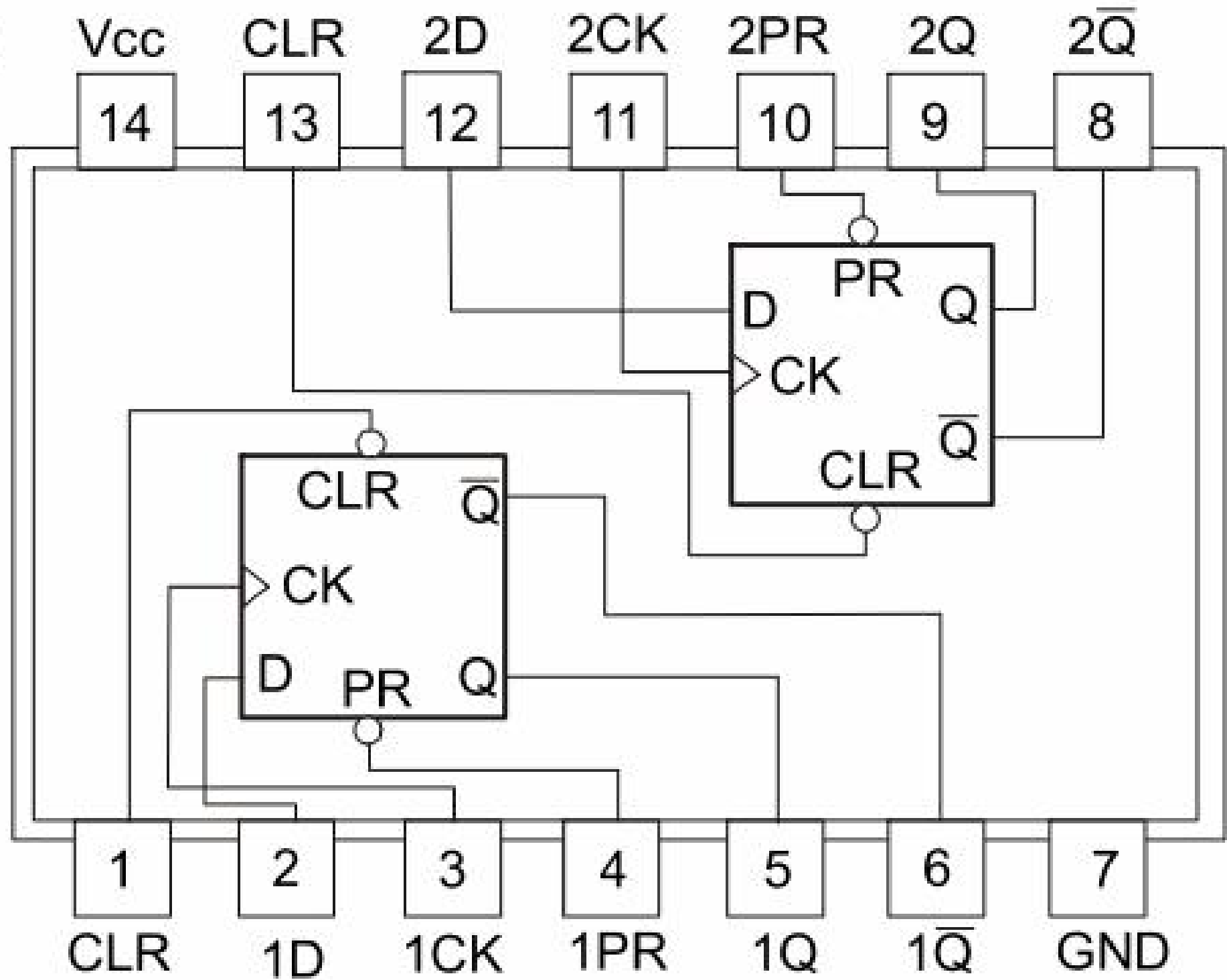
DUAL D-TYPE POSITIVE-EDGE-TRIGGERED FLIP-FLOPS WITH PRESET AND CLEAR

7474

FUNCTION TABLE

INPUTS				OUTPUTS	
PRESET	CLEAR	CLOCK	D	Q	$\sim Q$
L	H	X	X	H	L
H	L	X	X	L	H
L	L	X	X	H*	H*
H	H	$\uparrow$	H	H	L
H	H	$\uparrow$	L	L	H
H	H	L	X	Q ₀	$\sim Q_0$

* This configuration is nonstable

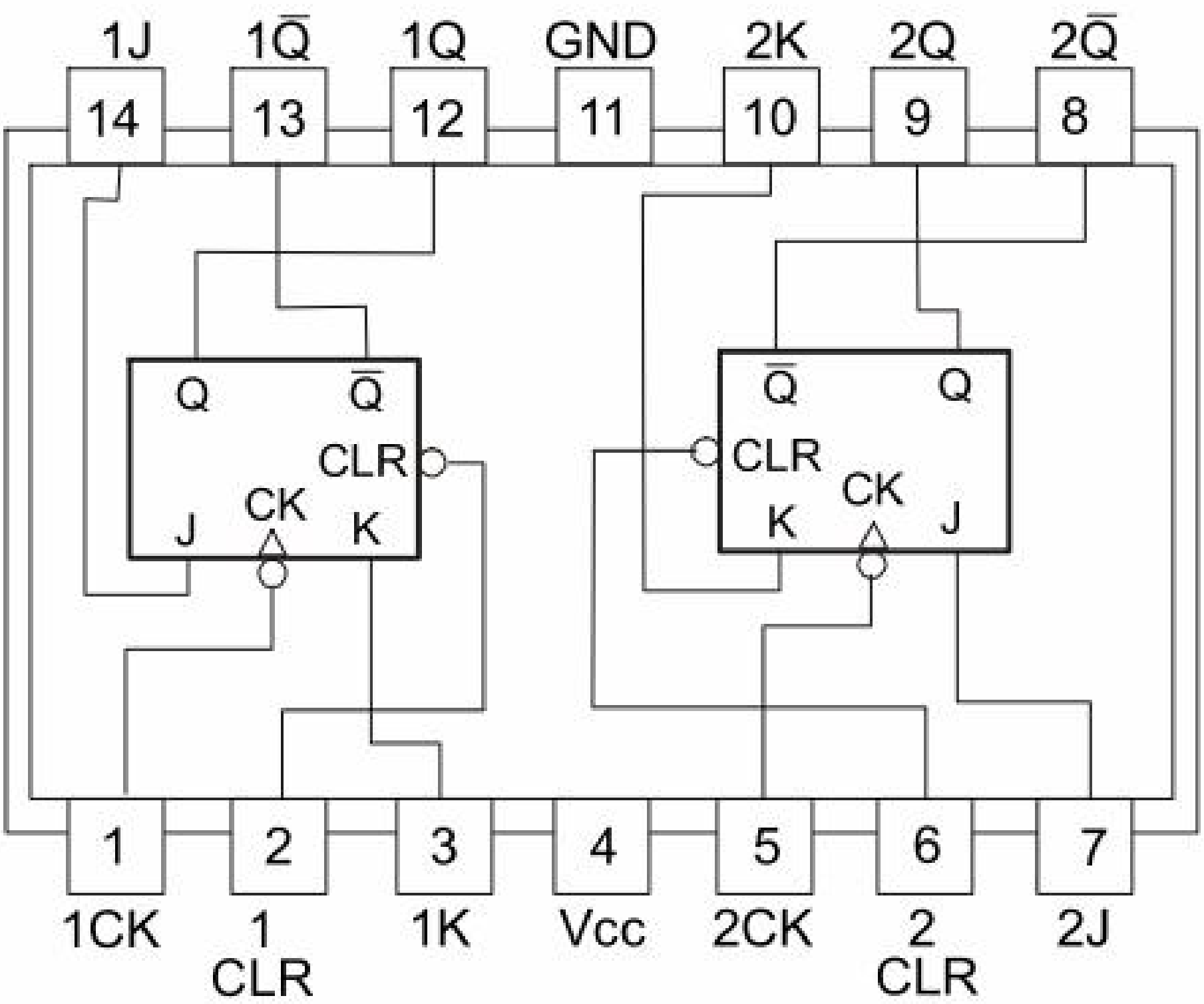


DUAL J-K FLIP-FLOPS WITH CLEAR

7473

FUNCTION TABLE

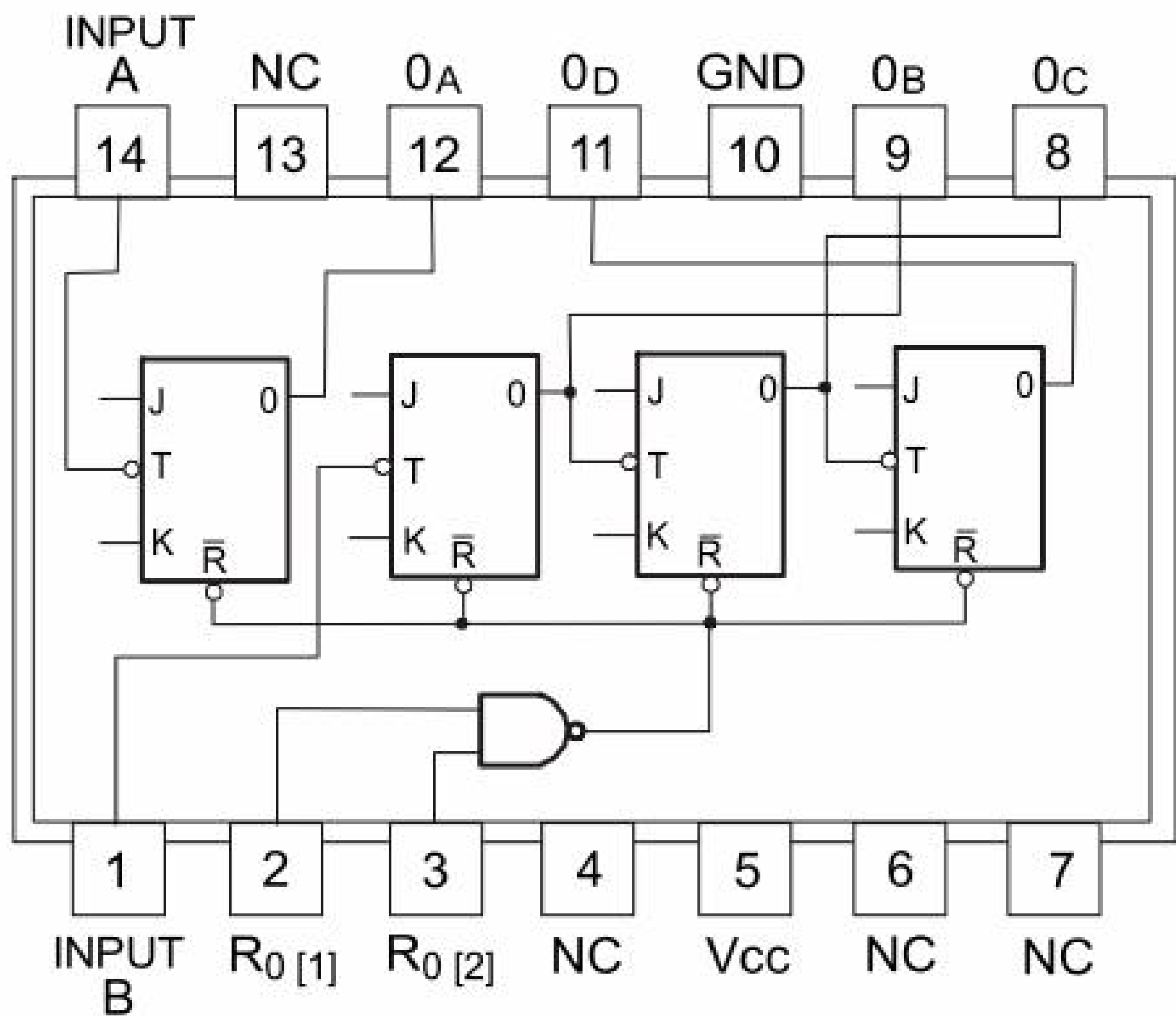
INPUTS				OUTPUTS	
CLEAR	CLOCK	J	K	Q	$\sim Q$
L	X	X	X	L	H
H		L	L	Q ₀	$\sim Q_0$
H		H	L	H	L
H		L	H	L	H
H		H	H	TOGGLE	TOGGLE



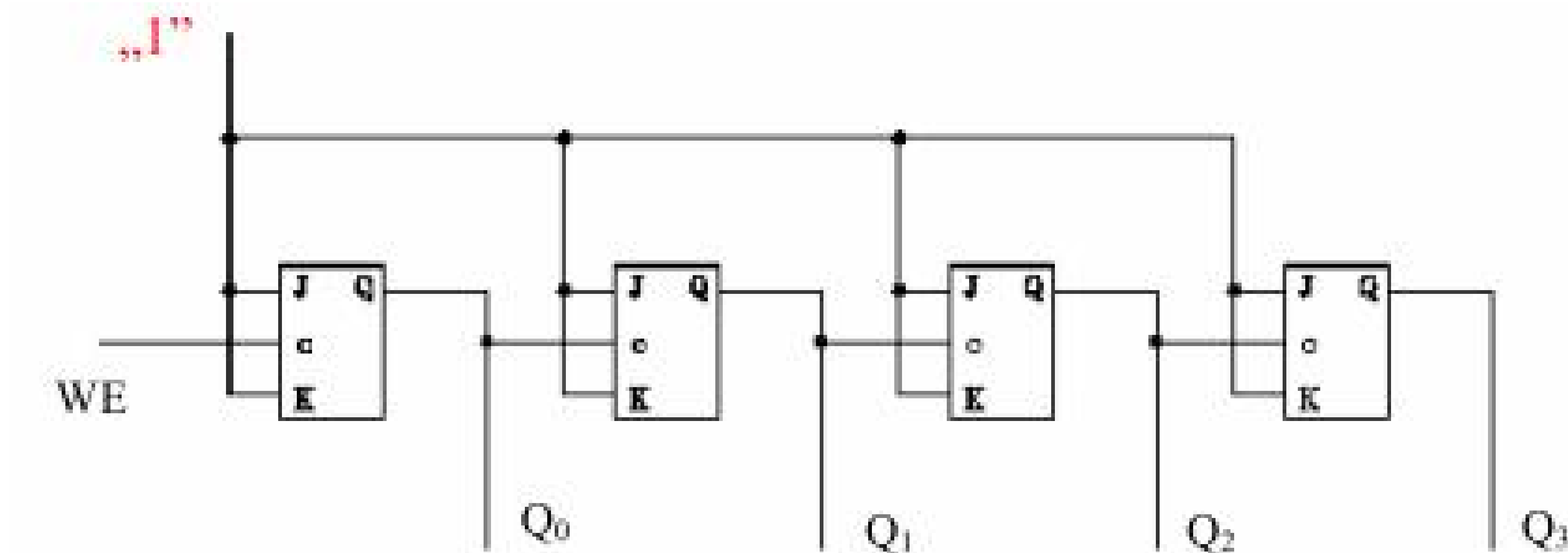
4-BIT BINARY COUNTERS

7493

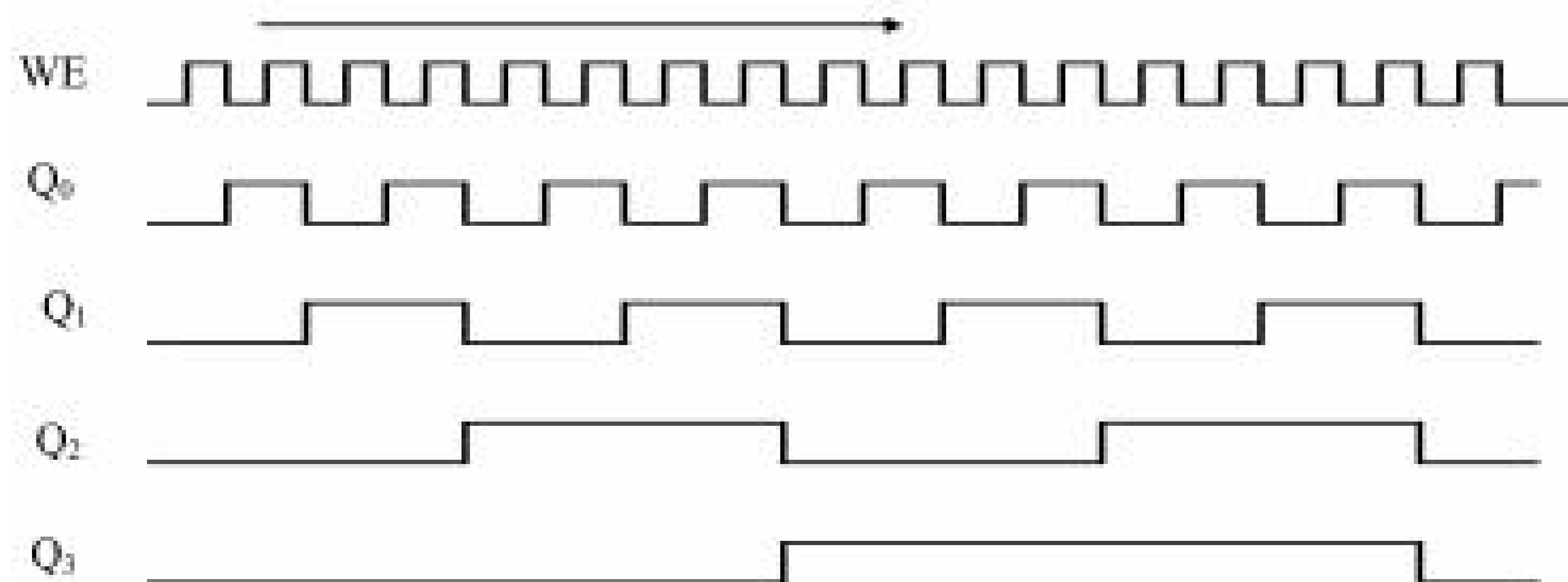
DIVIDE-BY-TWO AND DIVIDE-BY-EIGHT



Licznik 4-bitowy („przelicznik” – daje kolejne liczby 4-bitowe, zlicza impulsy zegara)



Przerzutniki JK zastosowane jako przerzutniki T (zwarłe wejścia J i K – porównaj tabelki prawdy przerzutników JK i T)

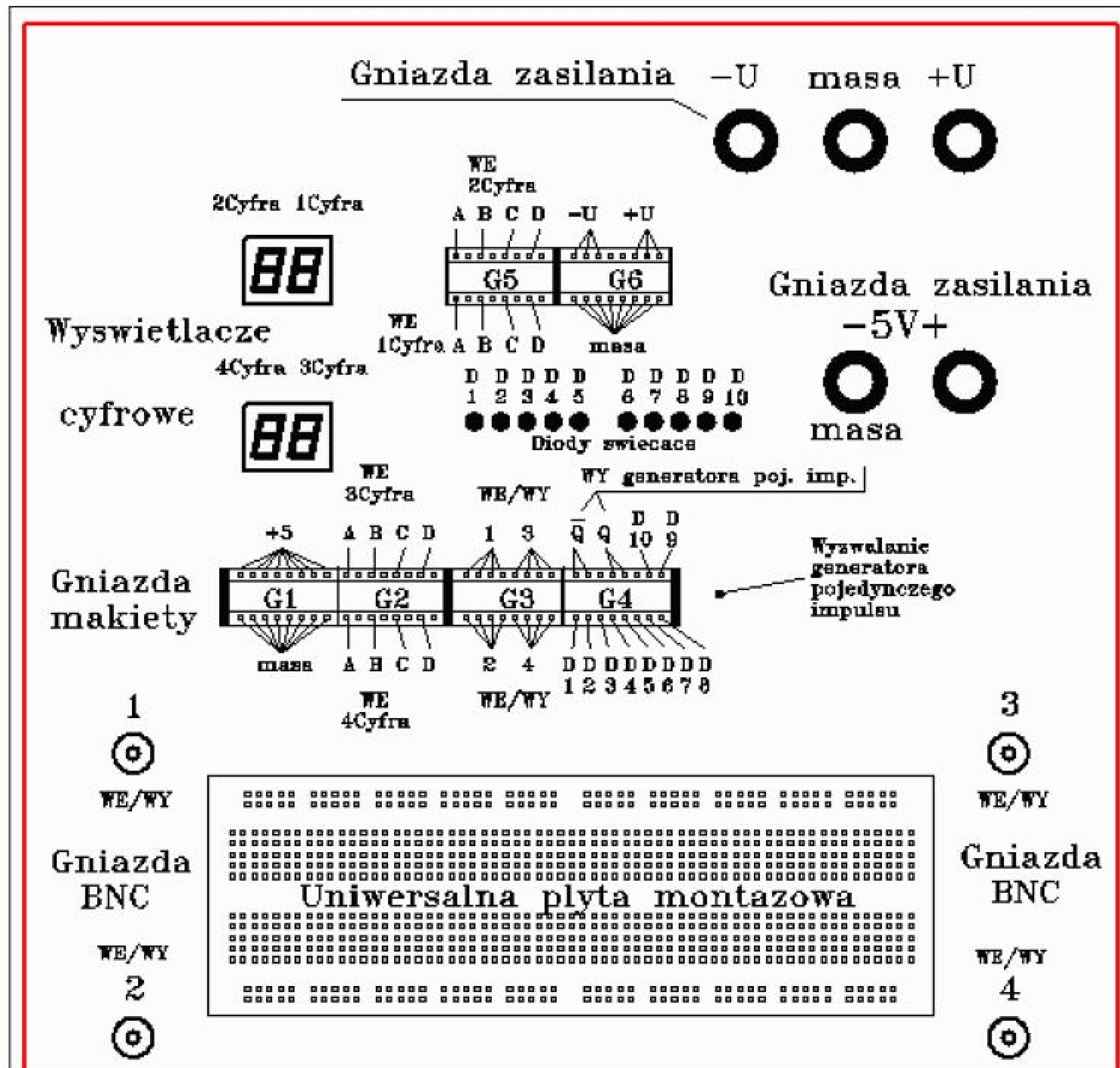


Pliki symulacji: [p6_Licznik-4-przerzutniki-T.circ](#) , [p6a_licznik_4-bit_4xD-flip-flop_v2.circ](#)

... w kierunku mikroprocesorów

- **ALU** – arithmetic-logic unit (jednostka arytmetyczno-logiczna),
sumator to najprostszy element ALU,
ALU może wykonywać różne operacje na dwu argumentach: dodawanie, odejmowanie, negowanie, przesuwanie bitów, mnożenie, porównywanie, przesyłanie do pamięci, itp.
Operacje wykonywane przez ALU są sterowane komendami języka maszynowego zakodowanymi jako słowa np. 8-bitowe, 16-bitowe,
Łatwiejsze niż język maszynowy do zapamiętania przez programistę komendy noszą nazwę języka Assembler.
- **Rejestr** – np. 8-bitowy analog **przerzutnika D**,
służy do przechowywania słów bitowych. W mikroprocesorach są rejestry danych dla ALU i rejestry komend do wykonania przez ALU.
- **Pamięć** – można o niej myśleć jako o macierzy przerzutników mających swoje adresy numerujące komórki pamięci
- **Układy wejścia – wyjścia** - transmisja (przeważnie równoległa) danych zewnętrznych (do lub z mikroprocesora)

Makieta do ćwiczenia dot. układów cyfrowych - praca bez lutowania



Zasada efektywnej pracy:
złożone układy należy
montować i uruchamiać
etapami, czyli sprawdzać
poprawność działania
kolejnych niewielkich części.

Jako próbnik stanów
logicznych można
wykorzystać jedną z diod
D1 ... D10.

Rys. 1 Płyta czołowa uniwersalnej makiety