Wykład 3

13 kwietnia 2021

Obwody cyfrowe

Krzysztof Korona

Zapis pozycyjny

Zapis pozycyjny dziesiętny:

min:s, s, s,

$3{:}54_{(60)} = \text{CCXXXIV} = 234_{(10)} = \mathbf{2}*100 + \mathbf{3}*10 + \mathbf{4}*1 = \mathbf{2}*10^2 + \mathbf{3}*10^1 + \mathbf{4}*10^0$

W zapisie dwójkowym (binarnym) pierwsza pozycja z prawej strony oznacza jedności, druga dwójki, trzecia czwórki (czyli 2^2), czwarta ósemki (czyli 2^3) itd.

	1	1	1	0	1	0	1	0	$_{(2)}$
	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	
	128	64	32	16	8	4	2	1	
=	128	+ 64	+ 32	+ 0	+ 8	+ 0	+ 2	+ 0	$= 234_{(10)}$

60tkowy
Babilonia 2000 pne

rzymski 1000 pne

Dziesiatkowy z zerem
Brahmagupta 628 r.
Europa:
Fibonacci
Liber Abaci (1202)

Zapis binarny

W zapisie dwójkowym (binarnym) pierwsza pozycja z prawej strony oznacza jedności, druga dwójki, trzecia czwórki (czyli 2^2), czwarta ósemki (czyli 2^3) itd.

	1	1	1	0	1	0	1	0	$_{(2)}$
	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	
	128	64	32	16	8	4	2	1	
=	128	+ 64	+ 32	+ 0	+ 8	+ 0	+ 2	+ 0	= $234_{(10)}$

Inne przykłady:

$6_{(10)} = 110_{(2)} = 4 + 2 + 0,$

$11_{(10)} = 1011_{(2)} = 8 + 0 + 2 + 1,$

$25_{(10)} = 11001_{(2)} = 16 + 8 + 0 + 0 + 1,$

$1\ 0000\ 0000_{(2)} - 1 = 256 - 1 = 255_{(10)} = 1111\ 1111_{(2)}$

Pozycje w zapisie binarnym nazywamy **bitami**. Bity zazwyczaj grupuje się po osiem, czyli w **bajty**. Przy pomocy jednego bajta można zapisać 256 liczb.

Przeliczanie na zapis binarny

Aby zamienić liczbę dziesiętną, na przykład, $\mathbf{25_{(10)}}$, na zapis dwójkowy trzeba posłużyć się następującym algorytmem:

1) Szukamy największej potęgi dwójki mniejszej od danej liczby.
64, 32, 16, 8, 4, 2, 1. W tym wypadku największą potęgą będzie 16, które stoi na piątym miejscu od prawej, zapisujemy:
$1\_\ \_\ \_\ \_\ _{(2)}$.
2) Następnie od danej liczby trzeba odjąć zapisaną liczbę (zostaje 9) i to co zostaje przyrównać do mniejszych potęg dwójki: 1, 2, 4, 8.
W tym wypadku będzie to 8, które stoi na 4 miejscu od prawej, zapisujemy:
$11\_\ \_\ \_\ _{(2)}$.
3) Powtarzamy odejmowanie, zostaje jeden. Powtarzamy porównywanie i zapisujemy 1 na pierwszym miejscu.
$11\_\ \_1_{(2)}$.
4) Zostało zero. Puste miejsca wypełniamy zerami i otrzymujemy wynik:
$11001_{(2)}$.
Otrzymaliśmy $\mathbf{25_{(10)} = 11001_{(2)}}$.

Zapis szesnastkowy

W zapisie szesnastkowym (heksadecymalnym) pierwsza pozycja z prawej strony oznacza jedności, druga szesnastki, trzecia $16^2 = 256$ itd.

Cyfry: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F
= 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15

1	0	F	$F_{(16)}$
16^3	16^2	16^1	16^0
4096	256	16	1

$= 4096 + 0 + 15*16 + 15 = 4351_{(10)}$

Jeden bajt to dwie cyfry szesnastkowe, przykładowo:

$1000\ 0001_{(2)} = 81_{(16)} = 129_{(10)}$
$1111\ 0010_{(2)} = F2_{(16)} = 242_{(10)}$

Zapis kolorów HTML:

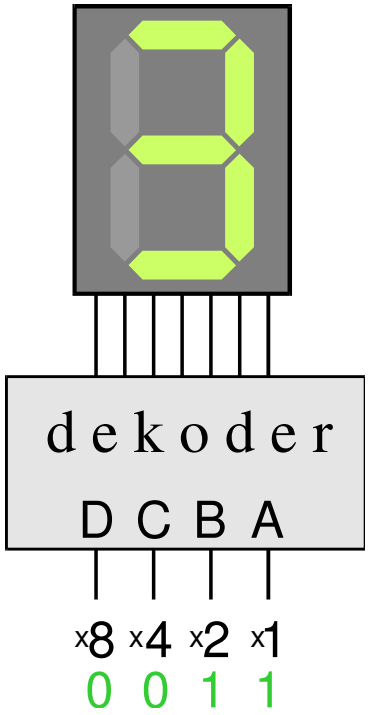
BGCOLOR="#DFDFFF”

BGCOLOR="#FF0000"

#DF = 223, 0xFF=255

Kod BCD (Binary Coded Decimal)

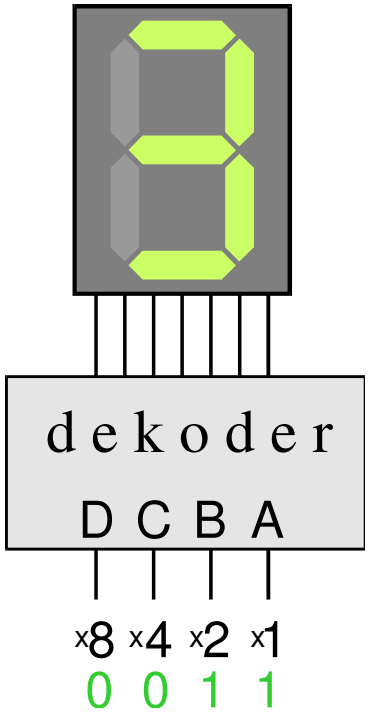
Do pokazywania cyfr dziesiętnych służą wyświetlacze cyfrowe. Składają się one z 7 segmentów, które sterowane są przez dekoder. Dekoder może być zawarty w pojedynczym układzie scalonym, na przykład UCY7447.



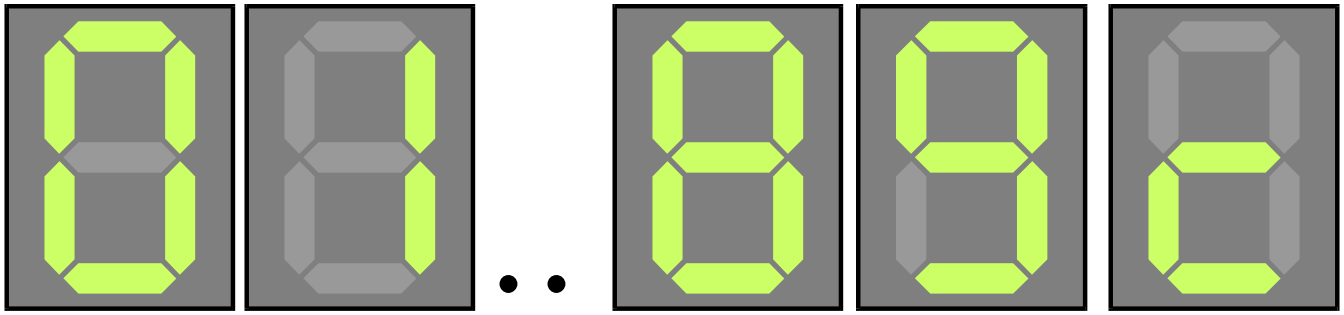
Do dekodera podawana jest informacja w systemie BCD (Binary Coded Decimal). W tym systemie każdą cyfrę dziesiętną zapisuje się przy pomocy 4 bitów.
Na przykład:
$78_{(10)} = 7, 8_{(BCD)} = 0111_{(2)}, 1000_{(2)}$.

Wyświetlacz cyfrowy

Do pokazywania cyfr dziesiętnych służą wyświetlacze cyfrowe. Składają się one z 7 segmentów, które sterowane są przez dekoder. Dekoder może być zawarty w pojedynczym układzie scalonym, na przykład UCY7447.



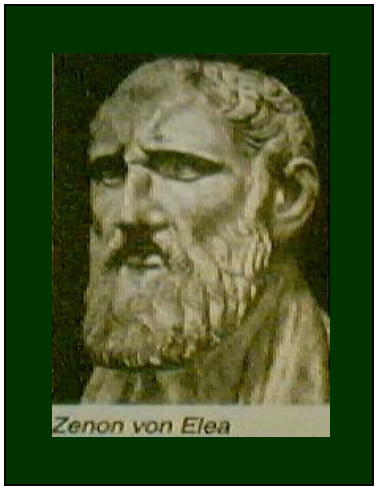
Liczby od 0 do 9 są kodowane standardowo:



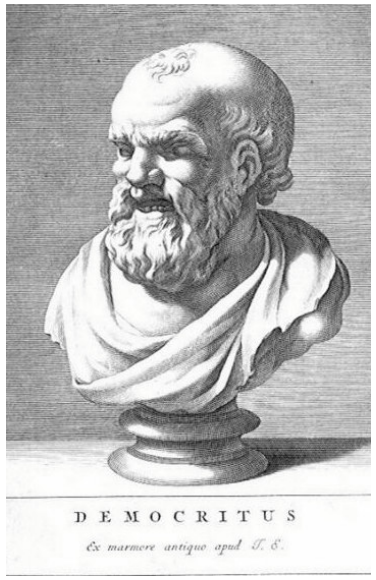
$0000_{(2)} = 0_{(10)}$ $0001_{(2)} = 1_{(10)}$ . . $1000_{(2)} = 8_{(10)}$ $1001_{(2)} = 9_{(10)}$ $1010_{(2)} = 10_{(10)}$

Liczby od $1010_{(2)} = 10_{(10)}$ do $1111_{(2)} = 15_{(10)}$ nie są używane, ale dekodery mogą wyświetlać jakieś układy segmentów. Przykładowo, dla liczby $1010_{(2)} = 10_{(10)}$, dekoder UCY7447 wyświetla kształt litery 'c'

Rachunek zdań



To zdanie jest nieprawdziwe.



Czy zdanie zamieszczone powyżej jest prawdziwe?

Algebra Boole'a

Podstawowym narzędziem matematycznym wykorzystywanym w elektronice cyfrowej jest rachunek zdań oparty na algebrze Boole'a.
Algebra jest działem matematyki. W ramach tego działu definiuje się struktury takie jak: *zbiory*, *przestrzenie*, *grupy, ciała* i *algebry*. *Algebra* to zbiór ze zdefiniowanymi działaniami na parach elementów zbioru.

Najczęściej używana *algebra* Boole'a to dwuelementowa algebra wartości logicznych {prawda, fałsz} lub {0, 1} z działaniami koniunkcji (mnożenia, **i**, AND, ∧), alternatywy (sumy, **lub**, OR, ∨) i zaprzeczenia (negacji, **nie**, NOT, ~).

Tabele działań (mapy Karnaugha)

∧	0	1
0	0	0
1	0	1

mnożenie, **i**,
AND, ∧ ,

∨	0	1
0	0	1
1	1	1

alternatywa, suma, **lub**,
OR, ∨,

	0	1
~	1	0

zaprzeczenie, negacja, **nie**,
NOT, ~.

Algebra Boole'a

Podstawowym narzędziem matematycznym wykorzystywanym w elektronice cyfrowej jest rachunek zdań oparty na algebrze Boole'a.
Algebra jest działem matematyki. W ramach tego działu definiuje się struktury takie jak: *zbiory*, *przestrzenie*, *grupy, ciała* i *algebry*. *Algebra* to zbiór ze zdefiniowanymi działaniami na parach elementów zbioru.

Najczęściej używana *algebra* Boole'a to dwuelementowa algebra wartości logicznych {prawda, fałsz} lub {0, 1} z działaniami koniunkcji (mnożenia, **i**, AND, $\wedge$), alternatywy (sumy, **lub**, OR, $\vee$) i zaprzeczenia (negacji, **nie**, NOT, ~).

Oprócz działań podstawowych używamy wielu innych funkcji logicznych, z których warto wymienić:

NAND – zaprzeczenie koniunkcji (dysjunkcja, **nie..i**, $\uparrow$, /),

$\uparrow$	0	1
0	1	1
1	1	0

XOR – alternatywa rozłączna (ekskluzja, **albo..albo**, $\underline{\vee}$),

$\underline{\vee}$	0	1
0	0	1
1	1	0

NOR – zaprzeczenie alternatywy (binegacja, **ani..ani**, $\downarrow$).

$\downarrow$	0	1
0	1	0
1	0	0

Kolejność działań

Kolejność działań:

~ negacja,

$\wedge$ koniunkcja (iloczyn), $\uparrow$ negacja koniunkcji,

$\vee$ alternatywa (suma), $\downarrow$ negacja alternatywy, $\underline{\vee}$ alternatywa rozłączna.

~A $\wedge$ ~B Oznacza: (nieprawda, że A) i (nieprawda, że B)

C $\vee$ A $\wedge$ B Oznacza: C lub (A i B)

Kolejność tutaj jest analogiczna jak dla zwykłych działań:

-a * -b

c + a*b

Podwójna negacja:

Zaprzeczenie zaprzeczenia przywraca pierwotna wartość:

~ ~A = A

Zdanie "*nie jest nieprawdziwy,*" oznacza "*jest prawdziwy*".

Analogicznie jak -(-a) = a

Właściwości działań

Kolejność działań:

~ negacja,

$\wedge$ koniunkcja (iloczyn), $\uparrow$ negacja koniunkcji,

$\vee$ alternatywa (suma), $\downarrow$ negacja alternatywy, $\underline{\vee}$ alternatywa rozłączna.

Podwójna negacja:

Zaprzeczenie zaprzeczenia przywraca pierwotna wartość:

$\sim \sim A = A$

Rozdzielność działań:

W algebrze liczb całkowitych mamy rozdzielność mnożenia:

$a \cdot (b + c) = a \cdot b + a \cdot c$, ale nie dodawania: $a + b \cdot c \neq (a + b) \cdot (a + c)$.

W algebrze Boole'a mamy zarówno rozdzielność koniunkcji:

$A \wedge (B \vee C) = A \wedge B \vee A \wedge C$, jak i alternatywy: $A \vee B \wedge C = (A \vee B) \wedge (A \vee C)$.

Tożsamości rachunku zdań

Zdanie lub jego zaprzeczenie jest prawdziwe.

$$q \vee \overline{q} = 1$$

Koniunkcja zdania i jego zaprzeczenie daje fałsz.

$$q \wedge \overline{q} = 0$$

Negacja koniunkcji tych samych zdań jest negacją:

$$q \uparrow q = \overline{q \wedge q} \Leftrightarrow \overline{q}$$

$q \uparrow q$ = NAND(q,q) = NOT q

Prawa de Morgana:

1) Zaprzeczenie iloczynu odpowiada sumie zaprzeczeń.

$$\sim(A \wedge B) = \sim A \vee \sim B, \qquad \overline{p \wedge q} \Leftrightarrow \overline{p} \vee \overline{q}$$

$\sim(p \wedge q)$ = NAND(p,q)

2) Zaprzeczenie sumy odpowiada iloczynowi zaprzeczeń.

$$\sim(A \vee B) = \sim A \wedge \sim B, \qquad \overline{p \vee q} \Leftrightarrow \overline{p} \wedge \overline{q}$$

Funkcje podstawowe

Koniunkcji nie da się zapisać przy pomocy samych alternatyw i na odwrót. Dlatego przy pomocy samych sum lub samych koniunkcji nie da się zapisać dowolnej funkcji logicznej. Dzięki prawom de Morgana wiemy jednak, że koniunkcję można zapisać przy pomocy sum i negacji.

Aby zapisać dowolna funkcję logiczną wystarczy użyć:
a) alternatywy i negacji (OR i NOT),
b) koniunkcji i negacji (AND i NOT),

c) zaprzeczeń koniunkcji (NAND), (Funkcję NOT otrzymujemy jako: $A \uparrow A = \sim A$.)

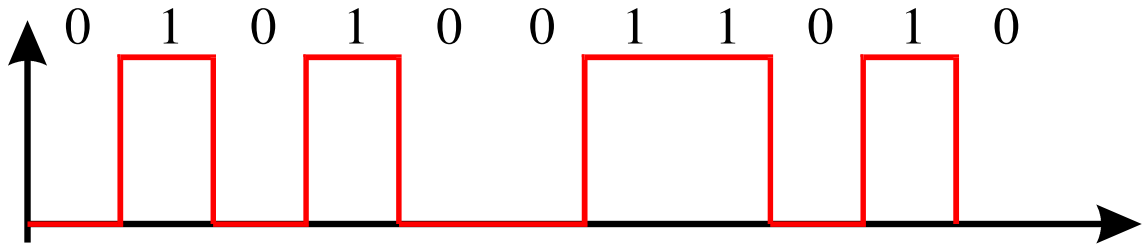
d) zaprzeczeń alternatywy (NOR). (Funkcję NOT otrzymujemy jako: $A \downarrow A = \sim A$.)

Na przykład AND można z NOR uzyskać jako:
$(A \downarrow A) \downarrow (B \downarrow B) = \sim A \downarrow \sim B = \sim (\sim A \vee \sim B) = A \wedge B$

Przy pomocy funkcji NAND lub NOR można utworzyć dowolną funkcję logiczną, (twierdzenie Sheffera) dlatego nazywa się je funkcjami **pełnymi.**

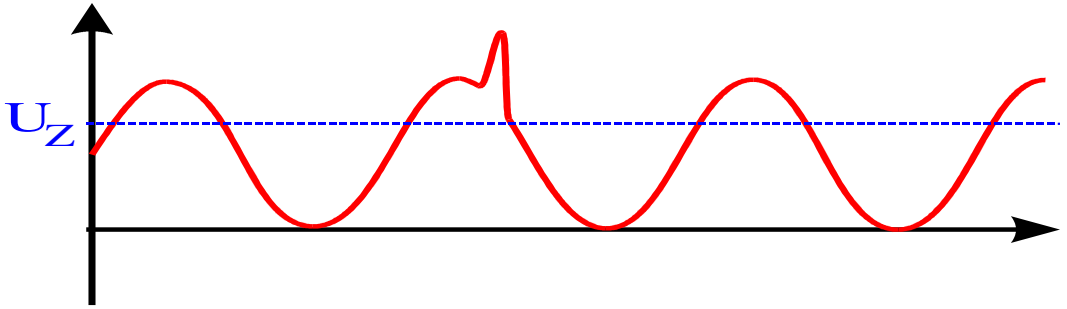
Funkcja XOR (alternatywa rozłączna może zastąpić NOT w punktach a) i b).

Elektroniczne układy cyfrowe

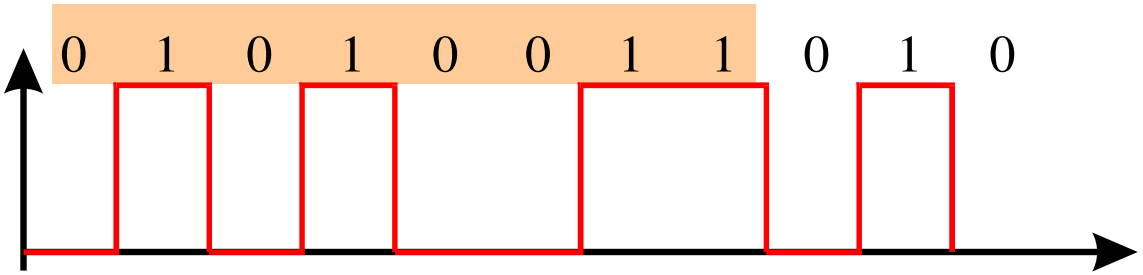


Układy cyfrowe (w odróżnieniu od analogowych) interpretują napięcia jako jeden z dwu stanów 0 (niski, L = Low) albo 1 (wysoki, H = High) i przetwarzają takie dwustanowe sygnały zgodnie z zasadami logiki matematycznej.

Przesyłanie sygnałów



Informacje analogowe przesyłane są jako sygnały napięciowe lub częstościowe.



Informacje cyfrowe przesyłane są w zapisie binarnym.

s z e r e g o w o

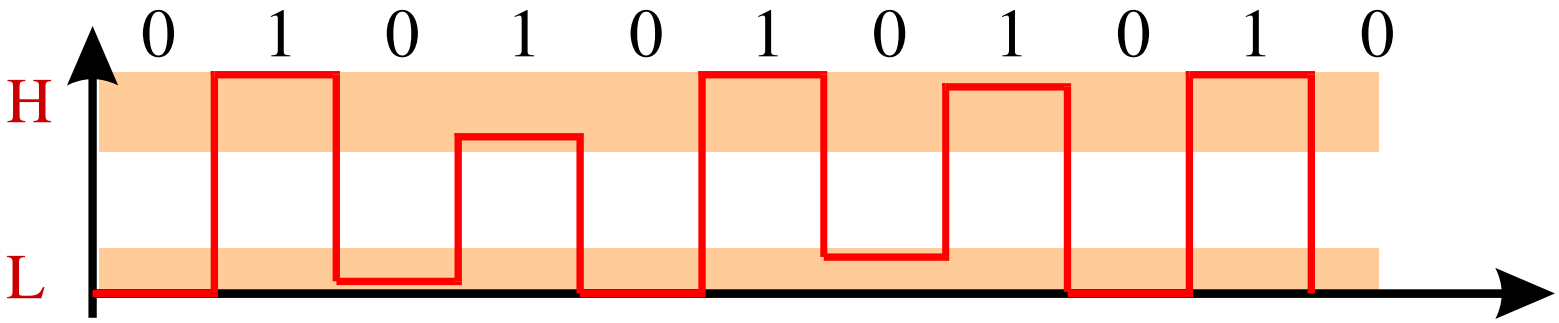
USB – Universal Serial Bus

równolegle

$$01010011_{(2)} = 2^6 + 2^4 + 2^1 + 2^0 = 64 + 16 + 2 + 1 = 83_{(10)}$$

Standard TTL

TTL (ang. Transistor-Transistor Logic) stany:
0 (niski) lub L (Low)
1 (wysoki) lub H (High)



Stany TTL:
Wejścia akceptują:
0: 0 – 0,8 V, **1**: 2 – 5 V, 20 µA.
Wyjścia powinny dawać:
0: 0 – 0,5 V, **1** : 3,5 – 5 V, 0,4 mA

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

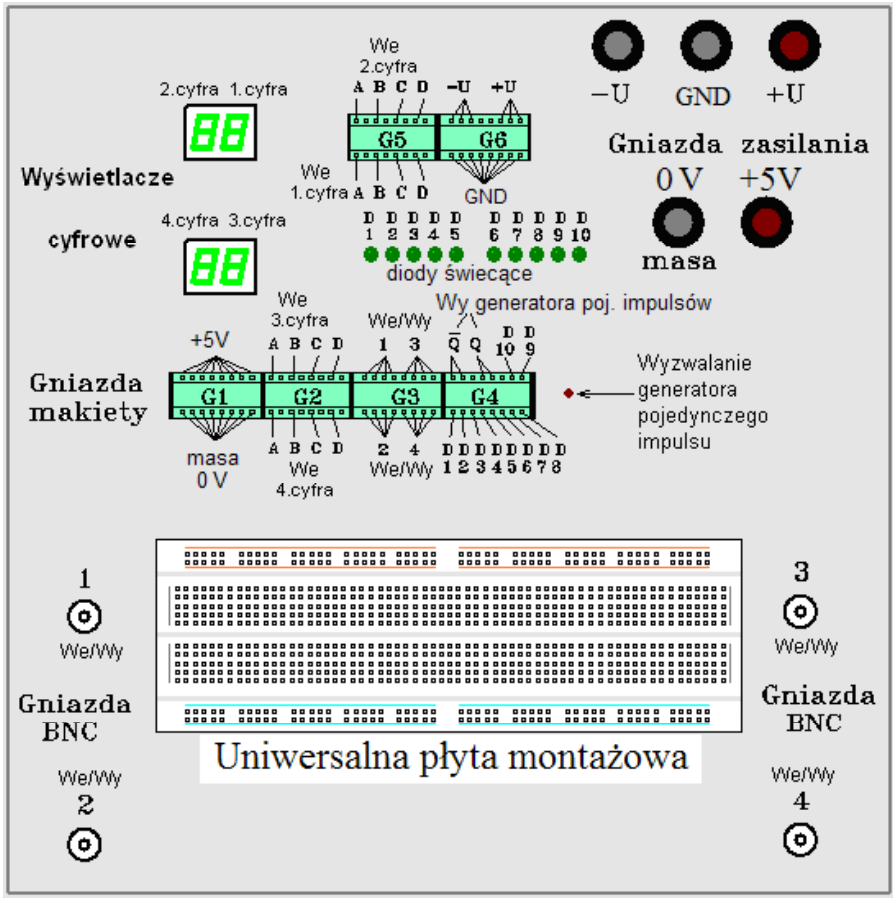
Zasady łączenia układów cyfrowych

- Układy scalone TTL są zasilane napięciem 5 V +/- 0,25V.
- Przed przyłączeniem napięć (zasilacz, generator) do układu należy upewnić się, że wartości tych napięć są zgodne ze standardem TTL.
- Logicznemu zeru odpowiada napięcie 0 – 0,5 V (0 – 0,8 V dla wejść).
- Logicznej jedynce odpowiada napięcie 3,5 – 5 V (2 – 5 V dla wejść).
- Niepodłączone wejście bramki przyjmuje stan 1 (high).

- Nie wolno łączyć ze sobą wyjść bramek.

Makieta do wykonania ćwiczenia

Gniazdo G1 podaje napięcie zasilające, 5 V, i łączy masę układu (0 V).
Gniazda G2 i G5 dekodera BCD obsługują dwa wyświetlacze cyfrowe.
Odpowiednie cyfry dziesiętne w kodzie BCD wyświetlane są na wyświetlaczu cyfrowym po podaniu na jego wejścia (A, B, C, D) słowa czterobitowego.
Gniazdo G3 obsługuje cztery gniazda wejścia /wyjścia typu BNC.
Gniazdo G4 zawiera wyjście generatora impulsów i wejścia diod świecących D1 - D10.
Stany podłączone do wejść diod święcących **D1 - D10** wyświetlane są na makiecie zgodnie z zasadą:
„0" logiczne = dioda nie świeci, „1" logiczna = dioda świeci. Zgodnie ze standardem TTL, niepodłączone wejście układu jest rozumiane jako „l" i powoduje także świecenie diody.



Wyjścia Q oraz **~Q** (zaprzeczenie Q) gniazda G4 są wyjściami ręcznie wyzwalanego generatora pojedynczych impulsów. Po naciśnięciu przycisku "Wyzwalanie", na wyjściu Q pojawi się pojedynczy przebieg w standardzie TTL o postaci: "_⊢⊣_", a jednocześnie na wyjściu ~Q impuls " ⊣_⊢".
Gniazdo G6 zawiera wyprowadzenia napięć z dodatkowych trzech gniazd zasilających (-U, GND, +U) służących do doprowadzenia napięć poza standardem TTL.
Niebieskie, czerwone i szare linie na uniwersalnej płycie montażowej pokazują połączenia wewnętrzne.

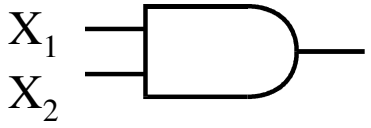
Bramki logiczne

Bramką logiczną nazywamy element realizujący pewną prostą funkcję logiczną (OR, NAND itp.), której argumenty (zmienne logiczne) oraz sama funkcja mogą przybierać jedną z dwóch wartości, {prawda, fałsz} lub {0, 1}.

Oznaczenia bramek

Do nazywania bramek używamy angielskich nazw operatorów.

AND

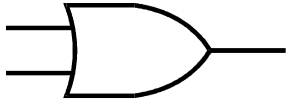


∧	0	1
0	0	0
1	0	1

(X_1 — kolumny, X_2 — wiersze, Y — wartości)

koniunkcja

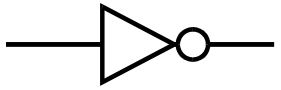
OR



∨	0	1
0	0	1
1	1	1

alternatywa

NOT

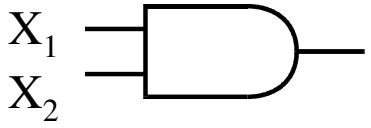


	0	1
~	1	0

negacja

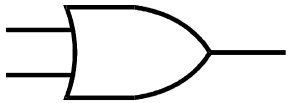
Oznaczenia bramek

AND



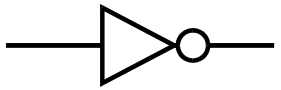
$\wedge$	X_1=0	X_1=1
X_2=0	0	0
X_2=1	0	1

OR



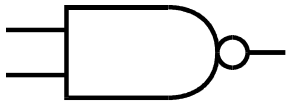
$\vee$	X_1=0	X_1=1
X_2=0	0	1
X_2=1	1	1

NOT



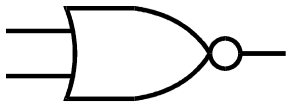
	X_1=0	X_1=1
~	1	0

NAND



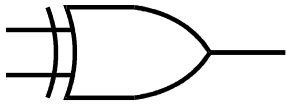
$\uparrow$	X_1=0	X_1=1
X_2=0	1	1
X_2=1	1	0

NOR



$\downarrow$	X_1=0	X_1=1
X_2=0	1	0
X_2=1	0	0

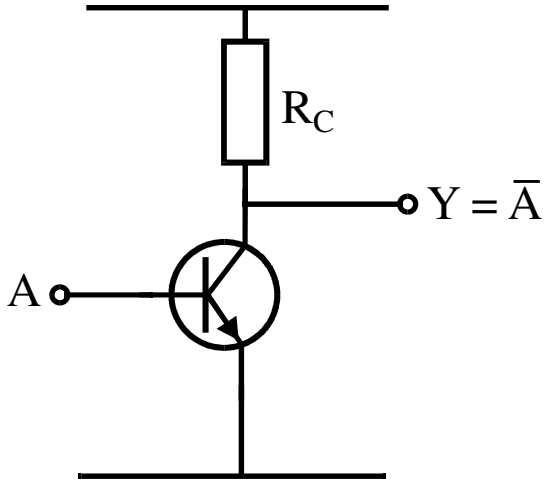
XOR



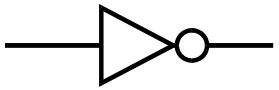
$\underline{\vee}$	X_1=0	X_1=1
X_2=0	0	1
X_2=1	1	0

Zgodnie z twierdzeniem Sheffera, przy pomocy funkcji **NAND** lub **NOR** można utworzyć dowolną funkcję logiczną.

Schemat bramki NOT

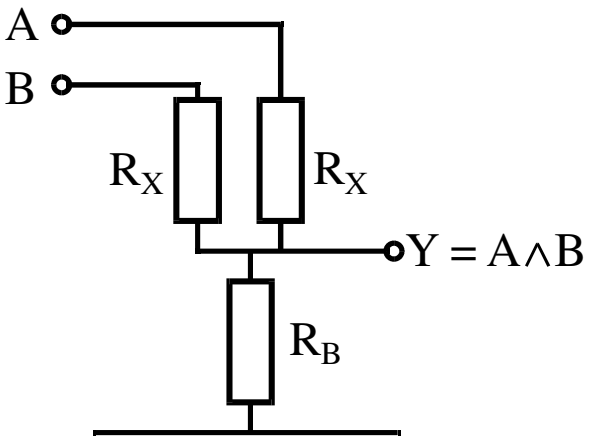


Układ wzmacniacza ze wspólnym emiterem. Gdy na wejściu A jest stan wysoki, tranzystor otwiera się i na wyjściu Y dostajemy stan niski.

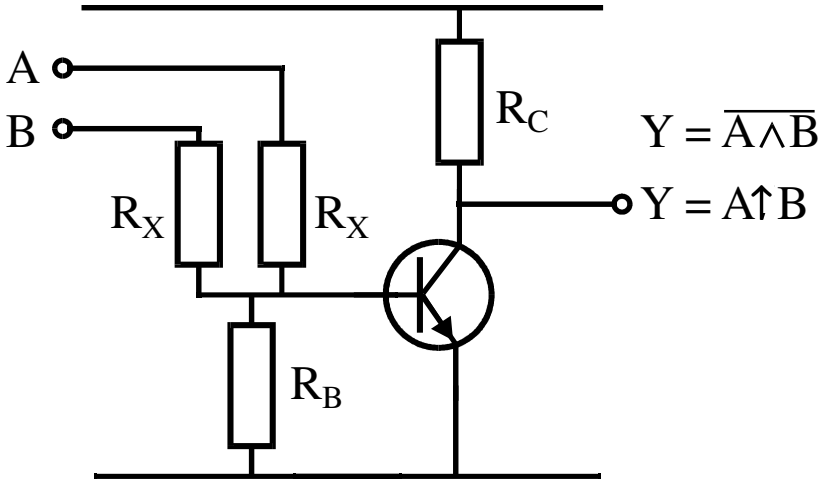


Zatem prosty układ wzmacniacza ze wspólnym emiterem działa jak bramka NOT.

Schemat bramki NAND

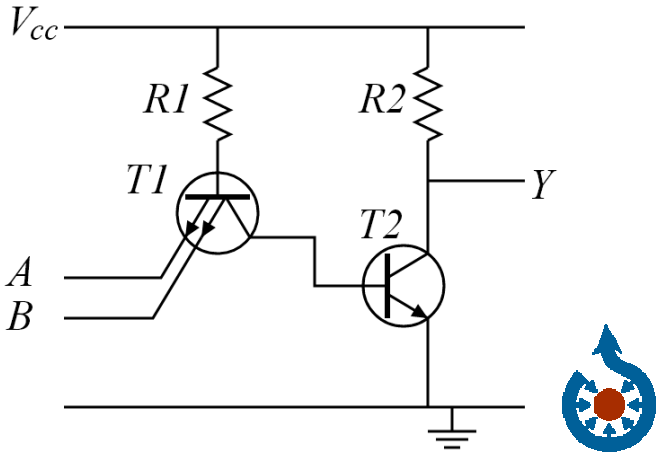


Łącząc przewody możemy uzyskać funkcje AND bez użycia tranzystorów. Jest to tak zwany wire-AND. Nie spełnia on jednak standardu TTL.

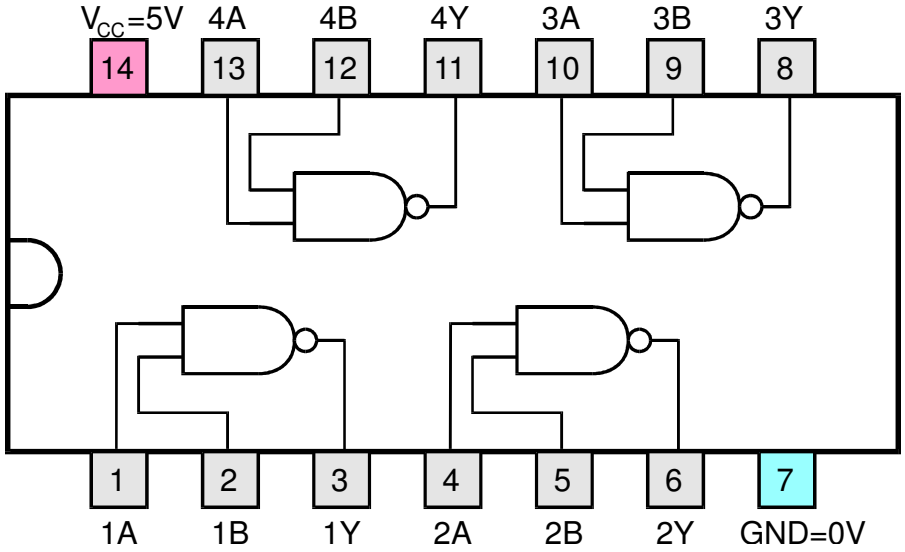


Łącząc wire-AND ze wzmacniaczem otrzymamy funkcjonującą bramkę NAND.

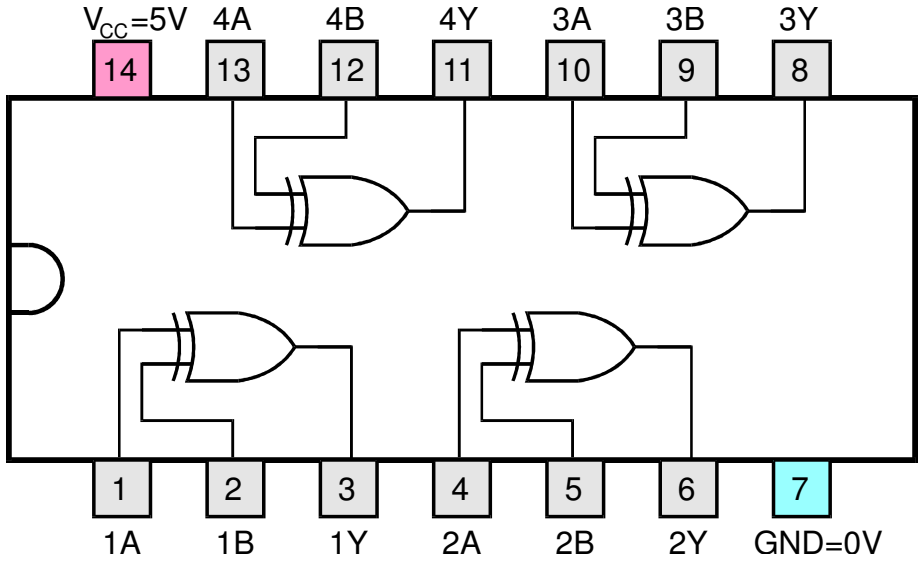
Optymalna konstrukcja bramki NAND wykorzystuje tranzystor z podwójnym emiterem.



Układy scalone z bramkami



Zasilanie 0 i 5 V.
Średni czas przenoszenia sygnału przez bramkę wynosi 1 - 30 ns.



Układ scalony UCY 7400:
4 bramki NAND (= NOT AND)
$Y = \overline{A \wedge B}$

Układ scalony UCY 7486:
4 bramki XOR, $Y = A \underline{\vee} B$

Układy bramek logicznych NAND

Iloczyn logiczny

$$Y = X_1 \wedge X_2$$

Iloczyn logiczny można zrealizować przy pomocy bramek NAND zaprzeczając wyjście bramki NAND, bo NOT NAND = AND.

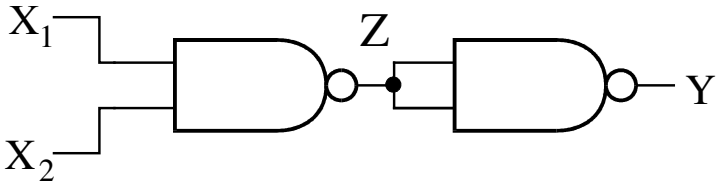


Tabela prawdy

X_1	X_2	Z	Y
0	0	1	0
0	1	1	0
1	0	1	0
1	1	0	1

Układy bramek logicznych NAND

Iloczyn logiczny

$$Y = X_1 \wedge X_2$$

Iloczyn logiczny można zrealizować przy pomocy bramek NAND zaprzeczając wyjście bramki NAND, bo NOT NAND = AND.

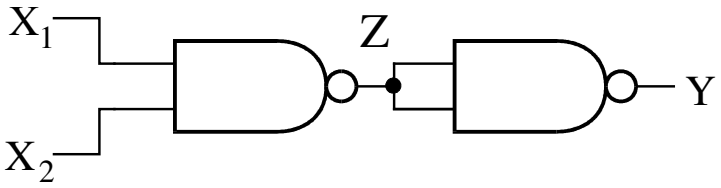


Tabela prawdy

X_1	X_2	Z	Y
0	0	1	0
0	1	1	0
1	0	1	0
1	1	0	1

Suma logiczna

$$Y = X_1 \vee X_2$$

Mamy do dyspozycji bramki NAND, więc aby uzyskać sumę musimy przekształcić równanie korzystając z praw de Morgana:

$$Y = X_1 \vee X_2 \Leftrightarrow \overline{\left(\overline{X_1} \wedge \overline{X_2}\right)}$$

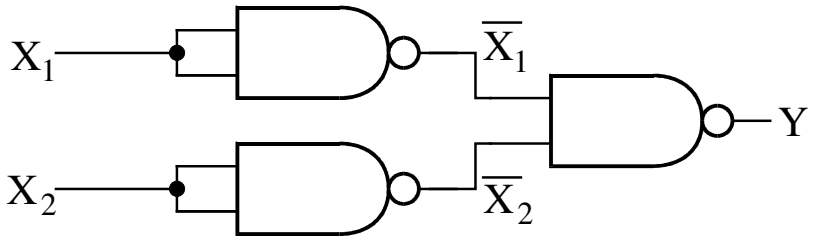


Tabela prawdy

X_1	X_2	Y
0	0	0
0	1	1
1	0	1
1	1	1

XOR z bramek NAND

Suma rozłączna (exclusive or, XOR)

Suma rozłączna jest prawdziwa, gdy zmienne X_1 i X_2 przyjmują różne wartości. Gdy są równe, czyli obie prawdziwe lub obie nieprawdziwe, wtedy suma rozłączna jest fałszywa.

$\veebar$	0	1
0	0	1
1	1	0

$$Y = X_1 \wedge \sim X_2 \ \vee \ \sim X_1 \wedge X_2$$

Na podstawie prawa de Morgana:

$$Y = \sim(\sim(X_1 \wedge \sim X_2) \ \wedge \ \sim(\sim X_1 \wedge X_2))$$
$$Y = \ \sim(X_1 \wedge \sim X_2) \uparrow \ \sim(\sim X_1 \wedge X_2)$$
$$Y = \ (X_1 \uparrow \sim X_2) \uparrow \ (\sim X_1 \uparrow X_2)$$

XOR z bramek NAND

Suma rozłączna (exclusive or, XOR)

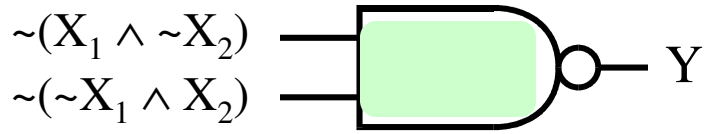
Suma rozłączna jest prawdziwa, gdy zmienne X_1 i X_2 przyjmują różne wartości. Gdy są równe, czyli obie prawdziwe lub obie nieprawdziwe, wtedy suma rozłączna jest fałszywa.

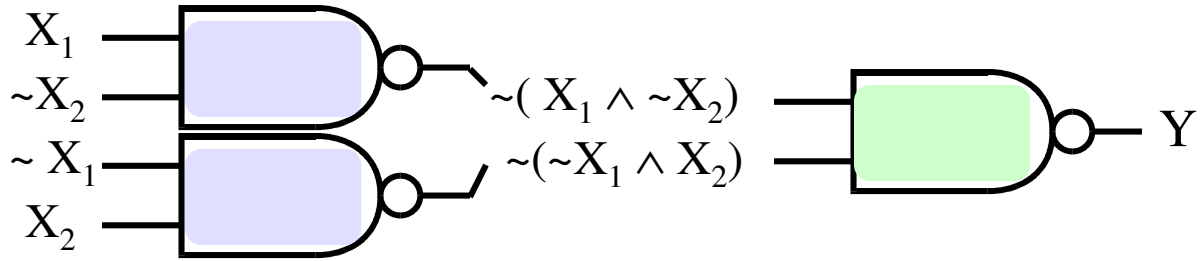
$$Y = X_1 \wedge \sim X_2 \ \vee \ \sim X_1 \wedge X_2$$

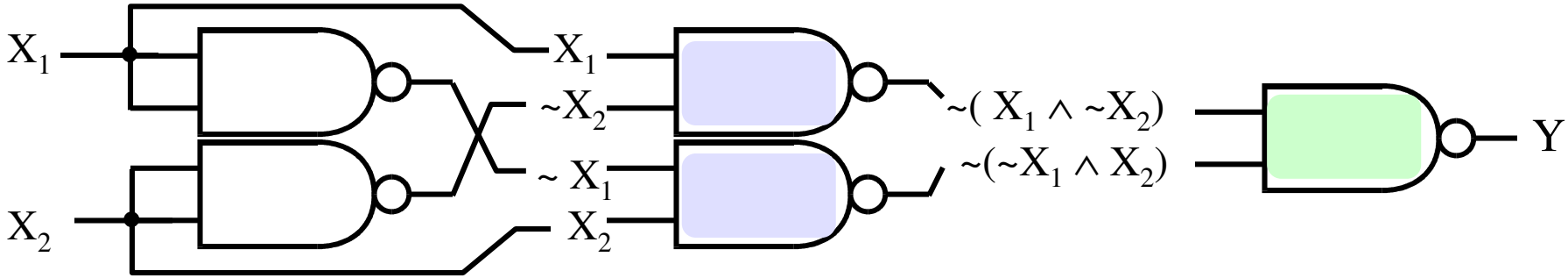
$\veebar$	0	1
0	0	1
1	1	0

Na podstawie prawa de Morgana:

$$Y = (X_1 \uparrow \sim X_2) \uparrow (\sim X_1 \uparrow X_2)$$







Dekoder podzielności przez 2 i 5

Budujemy dekoder wykrywający liczbę 3 bitową podzielną przez 2 lub przez 5. Zakładamy, że liczba 0 jest podzielna przez dowolny czynnik. Dekoder na wyjściu daje 1, gdy wykryje, że liczba spełnia zadany warunek. Ograniczamy się do bramek NAND.

1) Tabela prawdy

n	C	B	A	Y
0	0	0	0	1
1	0	0	1	0
2	0	1	0	1
3	0	1	1	0
4	1	0	0	1
5	1	0	1	1
6	1	1	0	1
7	1	1	1	0

2) Opis logiczny

2 5

$Y = \sim A \vee C \wedge \sim B \wedge A$

3) Przekształcenia

Korzystając z rozdzielności alternatywy, otrzymujemy:

$Y = (\sim A \vee C \wedge \sim B) \wedge (\sim A \vee A)$

$\sim A \vee A = 1$

$(\sim A \vee C \wedge \sim B) \wedge 1 = \sim A \vee C \wedge \sim B$

$Y = \sim A \vee (C \wedge \sim B)$

Korzystając z prawa de Morgana, otrzymujemy:

$Y = \sim(\sim\sim A \wedge \sim(C \wedge \sim B))$

$Y = \sim(A \wedge \sim(C \wedge \sim B))$

Dekoder 2, 5

Budujemy dekoder wykrywający liczbę 3 bitową podzielną przez 2 lub przez 5. Ograniczamy się do bramek NAND.

3) Przekształcenia dały wynik: $Y = \sim(A \wedge \sim(C \wedge \sim B))$.

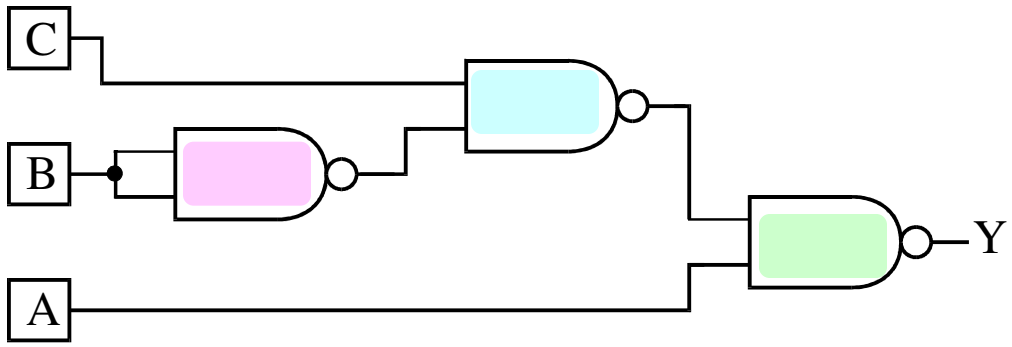
4) Zapisujemy równanie przy pomocy funkcji NAND: $(P \uparrow Q) = \sim(P \wedge Q)$:

$Y = A \uparrow \sim(C \wedge \sim B)$,

$Y = A \uparrow (C \uparrow \sim B)$,

$Y = A \uparrow (C \uparrow (B \uparrow B))$.

5) Rysujemy schemat na bramkach:

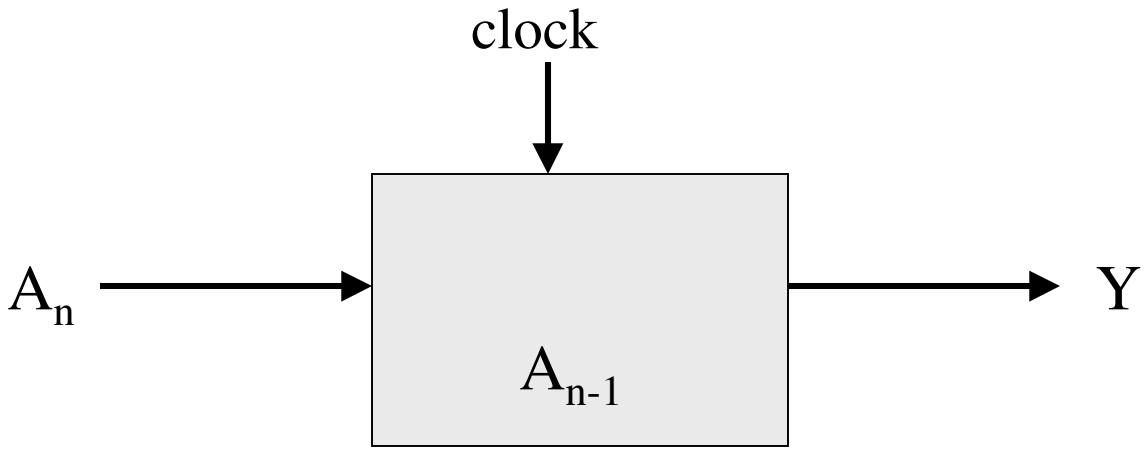


n	C	B	A	Y
0	0	0	0	1
1	0	0	1	0
2	0	1	0	1
3	0	1	1	0
4	1	0	0	1
5	1	0	1	1
6	1	1	0	1
7	1	1	1	0

6) Sprawdzamy

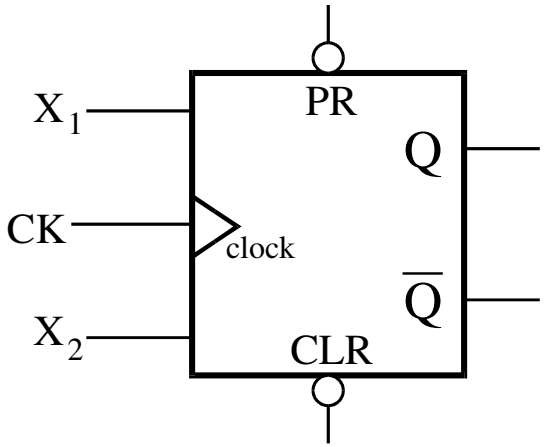
Układy sekwencyjne

Układy sekwencyjne – stan wyjścia zależy od stanu wejścia w danej chwili, jednak może również zależeć od stanu wejścia w poprzednich chwilach.
W układach synchronicznych ustalenie stanu wyjścia następuje w chwili taktu zegara.



Dla odróżnienia, układy złożone z samych bramek nazywamy układami kombinacyjnymi. Stan wyjścia zależy tylko od aktualnych stanów wejść.

Przerzutnik jako układ pamiętający



ang: flip-flop

PR (preset), CLR (clear), – wejścia asynchroniczne (wymuszają natychmiastową zmianę stanu wyjścia Q): preset - Q = 1, clear - Q = 0.
Kółko na wejściu oznacza negację, z tego wynika, że wejścia te reagują na stan zero. Jeżeli wejście układu TTL jest niepodłączone to jest na nim stan 1. Zatem, jeżeli wejścia te nie są potrzebne można je zostawić niepodłączone.
X_1, X_2 – wejścia danych.
CK (clock) - wejście synchronizacyjne (zegarowe).
Q – wyjście, $\overline{Q}$ – zanegowany stan wyjścia Q.

Między taktami zegara przerzutnik pamięta swój stan, wyjście Q nie zmienia się.

Przerzutnik typu T

Przerzutnik T reaguje na impuls zmianą stanu jeżeli na jego wejściu T jest jedynka. Gdy na wejściu T jest zero, wtedy zachowuje zapamiętany stan.

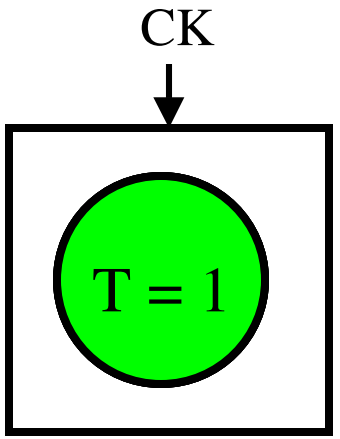
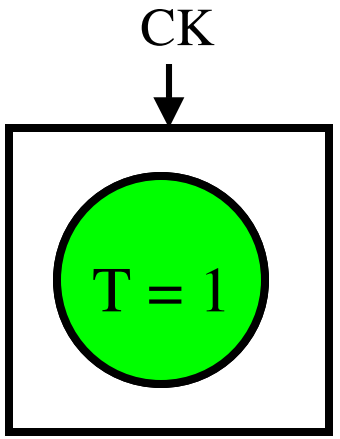


Tabela przerzutnika T

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

Q_n – stan przerzutnika przed impulsem zegarowym,
Q_{n+1} – stan przerzutnika po impulsie.

Przerzutniki typu T i J-K



Przerzutnik T reaguje na impuls zmianą stanu jeżeli na jego wejściu T jest jedynka. Gdy na wejściu T jest zero, wtedy zachowuje zapamiętany stan.

Tabela przerzutnika T

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

Q_n – stan przerzutnika przed impulsem zegarowym,
Q_{n+1} – stan przerzutnika po impulsie.

Przerzutnik J-K zamiast wejścia T ma dwa wejścia: J i K. Jeżeli J = K = 1 lub J = K = 0 to stan zmienia się lub zachowuje, podobnie jak w przerzutniku T.
Natomiast stan wejść (J, K) = (1, 0) ustawia na wyjściu stan 1, a stan wejść (J, K) = (0, 1) ustawia 0 (po impulsie zegara).

Tabela przerzutnika J-K

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

Przerzutnik D

Przerzutnik D po przyjściu impulsu przepisuje stan z wejścia na wyjście.

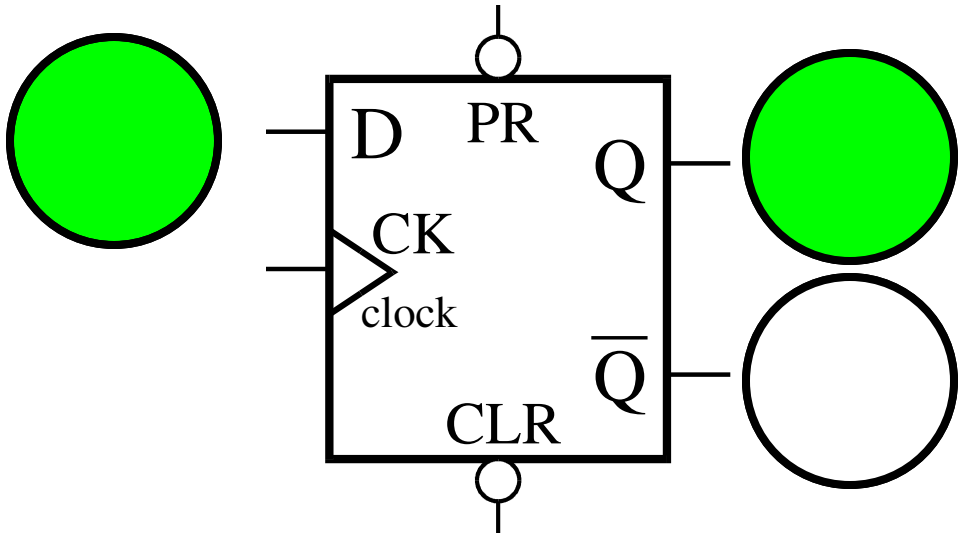
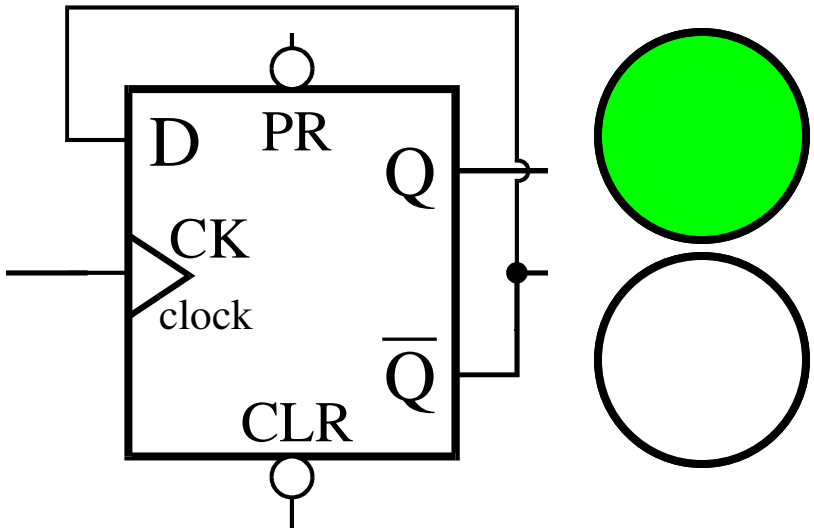


Tabela przerzutnika D

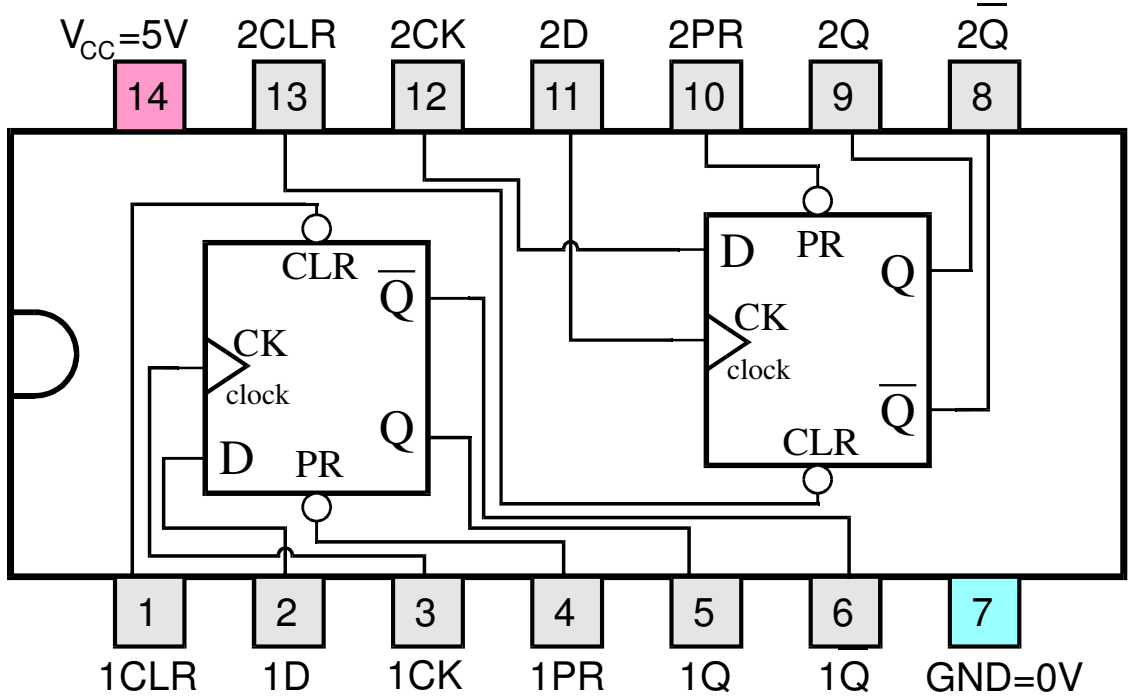
D	Q_{n+1}
0	0
1	1

Dzielnik częstości z przerzutnika D

Jeżeli wyjście zaprzeczone przerzutnika D podłączymy do wejścia to otrzymamy układ podobny do przerzutnika T.
Impuls zegara będzie zmieniał stan wyjścia Q na przeciwny. Po dwóch impulsach zegara przerzutnik wróci do stany pierwotnego. To znaczy, że na wyjściu będą impulsy o 2 razy mniejszej częstości.



Układy scalone z przerzutnikami



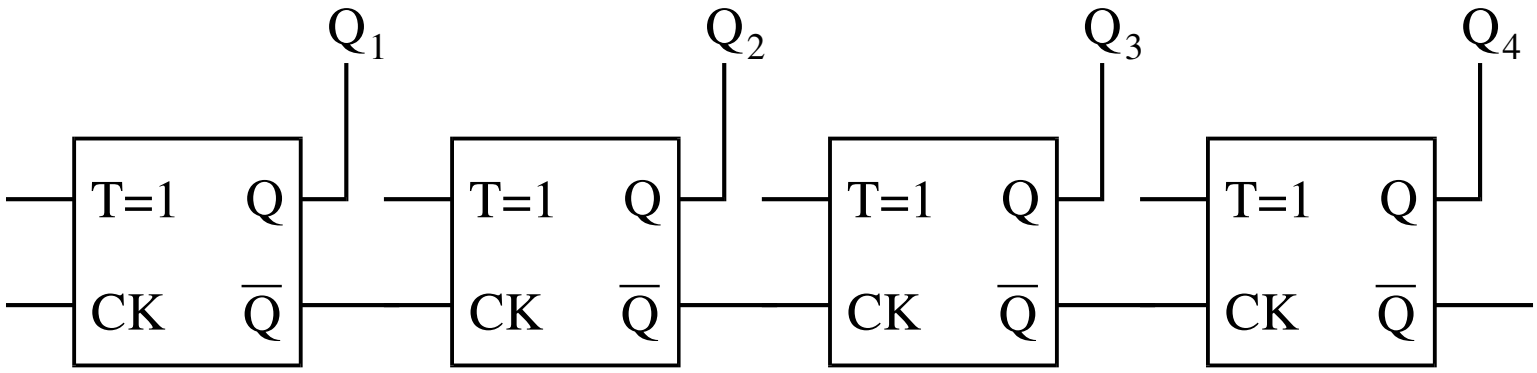
Układy wyposażone są w zaprzeczone wejścia CLR i PR. Przerzutniki zerowane są, gdy na wejściach CLR pojawi się stan 0. Stan 0 na wejściu PR ustawia je na 1.

Bardzo przydatną cechą przerzutników D w wersji z układu UCY7474 jest wyposażenie ich w wyjście zaprzeczenia ~Q. Łącząc wyjście ~Q z wejściem D możemy otrzymać układ pracujący jak przerzutnik T (dla T = 1). Układ ten po każdym impulsie zegara będzie zmieniał stan na przeciwny. Po dwóch impulsach będzie wracał, do stanu wyjściowego, czyli będzie pracował jak licznik liczący do 2.

Licznik z przerzutników T

Szeregowo połączone przerzutniki T tworzą licznik.

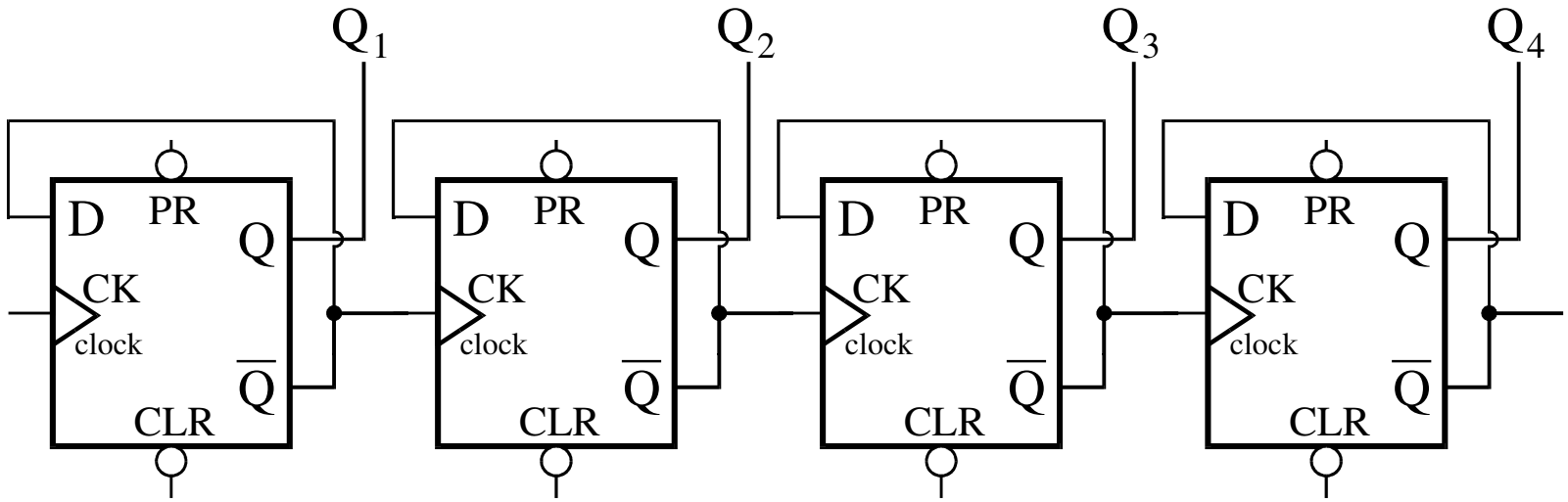
Licznik liczy kolejne takty zegara, tzn. zmiany stanu wejścia CK z 0 na 1



Jeżeli połączymy szeregowo kilka liczników, to na wyjściu pierwszego otrzymamy częstość podzieloną przez 2, na drugim - częstość podzieloną przez 4 na trzecim - przez 8 itd. Jeżeli na wejście takiego licznika ustawionego na zero, podamy pewna liczbę impulsów, to stan wyjść tego licznika przedstawi nam tę liczbę w zapisie binarnym. Tabela przejść takiego licznika przedstawiona jest poniżej. Widać, że wyjście Q_1 pełni role najmłodszego bitu (najmniej znaczącej cyfry, czyli 1), a wejście Q_4 jest najstarszym bitem (najbardziej znaczącą cyfrą, czyli 8).

Licznik zbudowany z przerzutników D

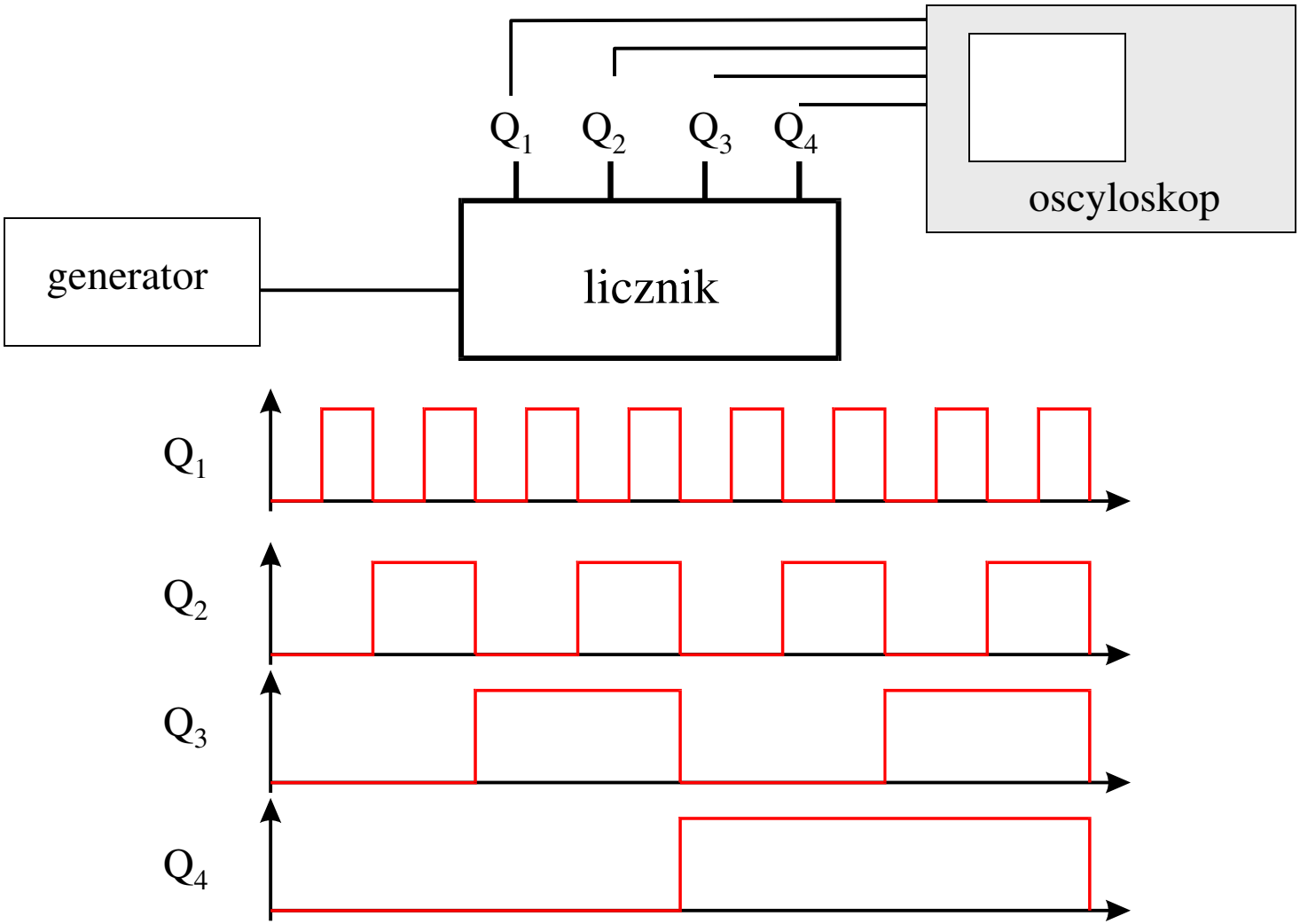
Jeżeli wyjście zaprzeczone przerzutnika D podłączymy do wejścia, to otrzymamy układ podobny do przerzutnika T.



Układ działa tak samo jak licznik wykonany z przerzutników T. Po *n* impulsach otrzymamy:

n = 0, $Q_4..Q_1 = 0000$
n = 1, $Q_4..Q_1 = 0001$
n = 2, $Q_4..Q_1 = 0010$
n = 3, $Q_4..Q_1 = 0011$
n = 4, $Q_4..Q_1 = 0100$
n = 5, $Q_4..Q_1 = 0101$
n = 6, $Q_4..Q_1 = 0110$

Licznik jako dzielnik częstości



Na wyjściach licznika otrzymamy częstość zegara (generatora) podzieloną przez 2 (na pierwszym wyjściu), na drugim wyjściu - częstość podzieloną przez 4 na trzecim - przez 8 itd.

Licznik modulo n

Po *n* impulsach otrzymamy:

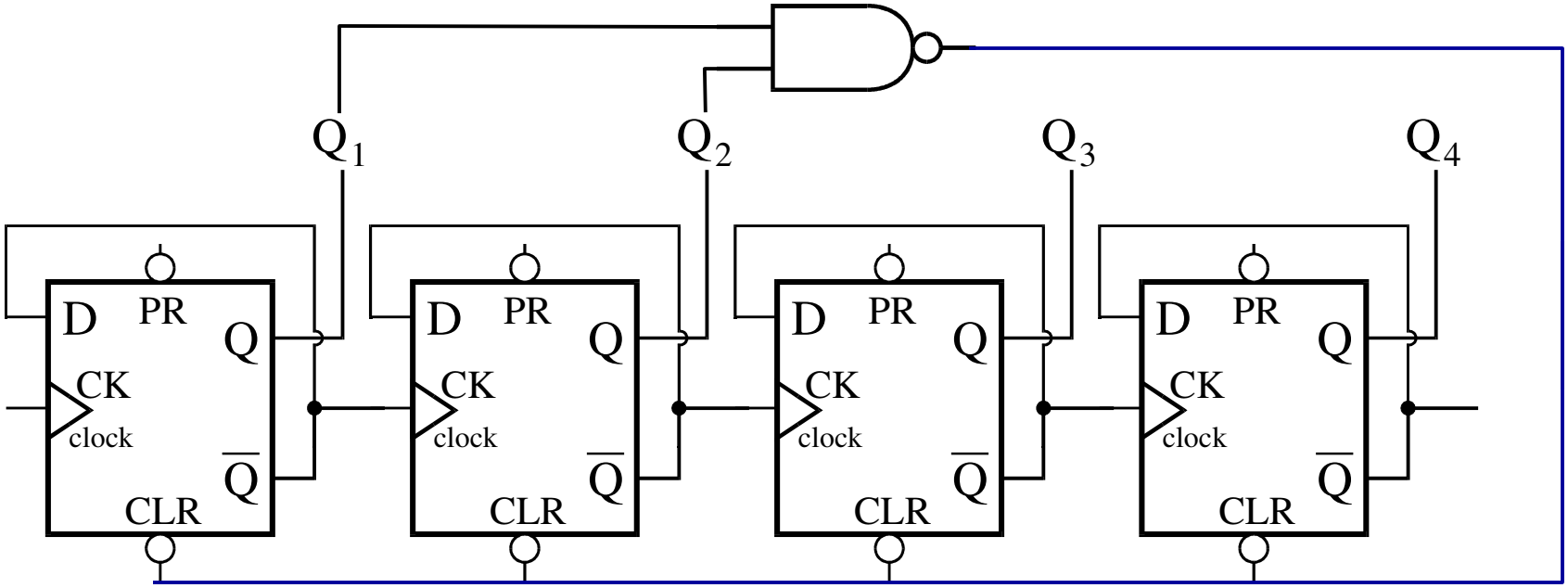
n = 0, $Q_4..Q_1 = 0000$

n = 1, $Q_4..Q_1 = 0001$

n = 2, $Q_4..Q_1 = 0010$

n = 3, $Q_4..Q_1 = 0011$

n = 4, $Q_4..Q_1 = 0100$



Zaawansowane układy cyfrowe

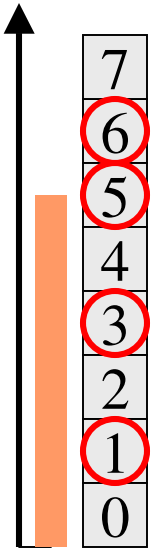
Przetworniki analogowo-cyfrowe

(AD - converters)

Przetworniki cyfrowo-analogowe (DA), zamieniające sygnał cyfrowy na analogowy, można prosto zrealizować sterując cyfrowo źródłami napięcia (lub prądu) o wartościach np. 1 V, 2 V, 4 V, 8 V itd.

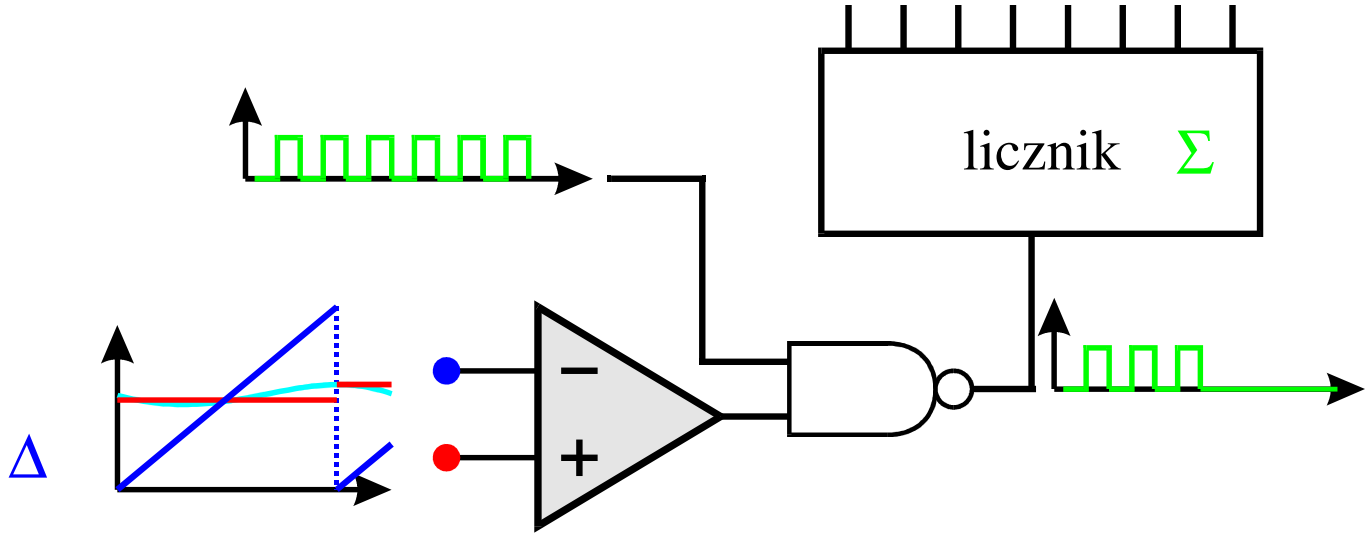
Przetworniki analogowo-cyfrowe (AD) są trudniejszym zadaniem. Natomiast są bardzo potrzebne do pomiarów oraz zamiany obrazu i dźwięku na sygnał cyfrowy. Wyróżniamy 3 metody:

1) Bezpośrednia (flash) - używa 2^n komparatorów, które sprawdzają, czy sygnał mieści się w ich zakresie. Najszybsze, ale możliwe tylko dla małych n.
2) Sukcesywnego porównywania (successive approximation). Porównujemy najstarszy bit, potem drugi itd. Czas = n*czas ustalania napięcia.
3) **Czasowa** (całkowania) (sigma, delta conversion) oparta jest na równoczesnym porównywaniu sygnału piłokształnego (delta) z mierzonym napięciem i zliczaniu impulsów cyfrowych (całkowaniu).



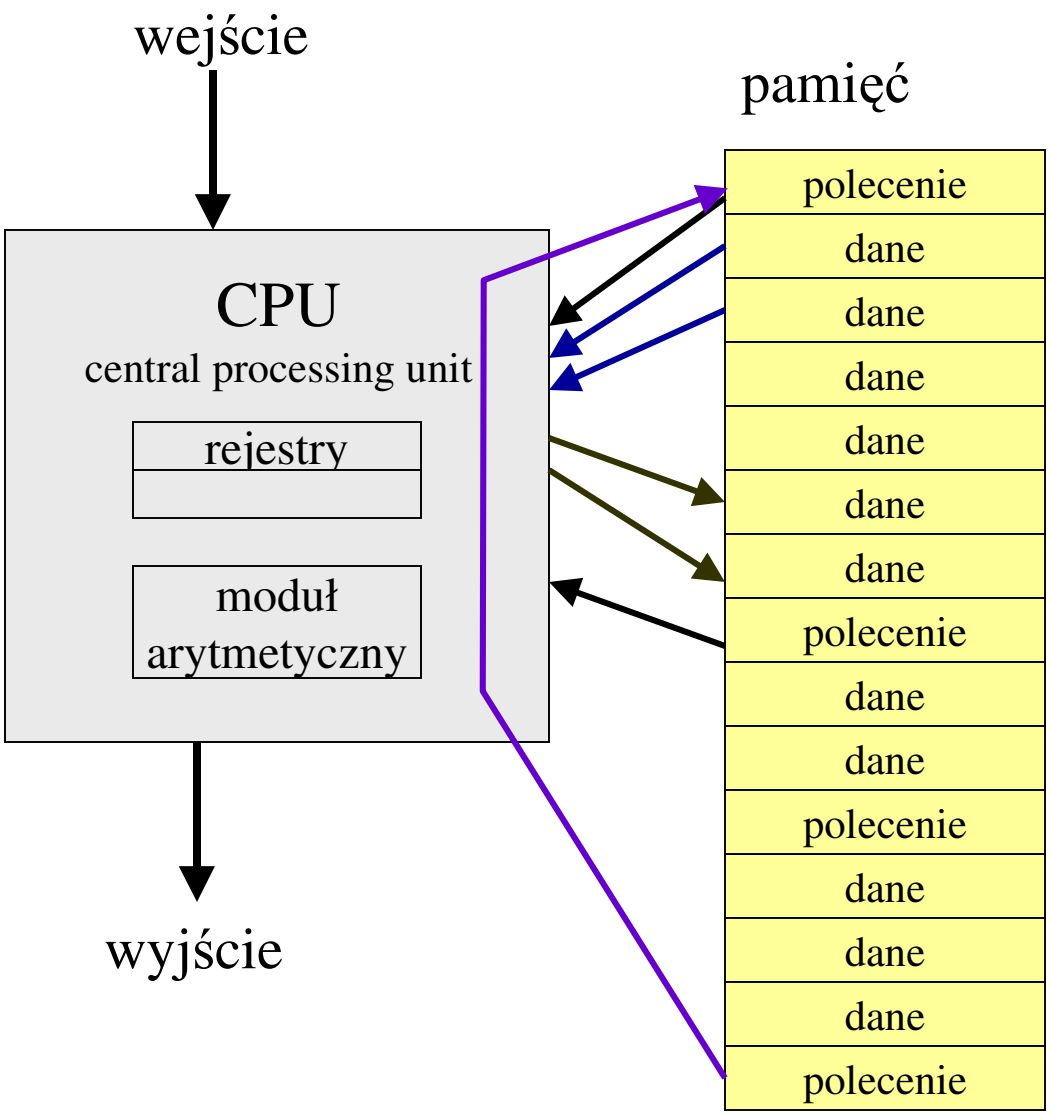
Przetworniki AD z metodą czasową

Metoda pośrednia czasowa (całkowania) (delta, sigma conversion) oparta jest na równoczesnym porównywaniu sygnału piłokształnego (Δ) z mierzonym napięciem i zliczaniu impulsów cyfrowych (całkowaniu, Σ).



Wejściowy sygnał analogowy jest próbkowany i zachowywany na czas pomiaru. Układ pomiarowy generuje sygnał zegarowy i analogowy sygnał narastający. Komparator porównuje próbkę sygnału z napięciem odniesienia i zamyka bramkę, gdy sygnał odniesienia przekroczy wartość sygnału mierzonego. Powoduje to zatrzymanie się licznika na mierzonej wartości.

Procesory



Procesor pobiera kolejne linie z pamięci do rejestrów i realizuje polecenia na wczytanych danych.

Polecenie może też spowodować wczytanie danych z wejścia, wysłanie ich na wyjście lub do pamięci.

Polecenia mogą być warunkowe. Ich efekt działania zależy od wartości rejestrów.

Czytanie pamięci nie musi się odbywać po kolei, bo polecenia mogą generować skoki, także wstecz, tworząc pętle.