Układy kombinacyjne

Kombinacyjne bloki funkcjonalne

- układy komutacyjne
- układy arytmetyczne

Układy komutacyjne

- multiplekser (MUX)
- demultiplekser (DMX)
- koder, dekoder, enkoder, transkoder

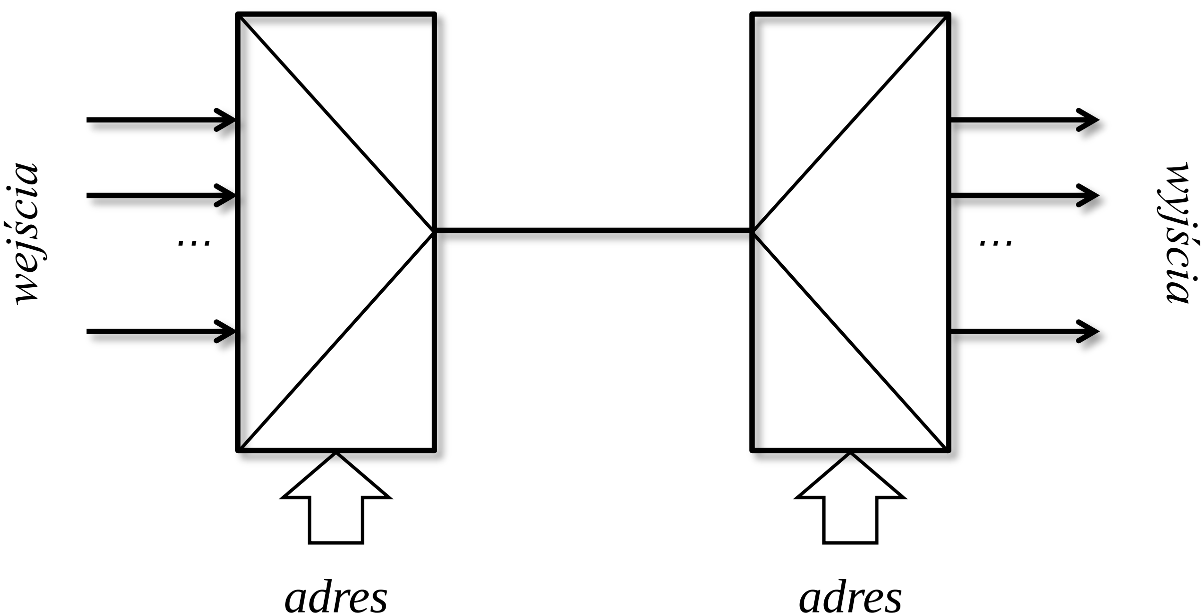
Multiplekser i demultiplekser

Multiplekser jest układem przetwarzającym równoległy kod dwójkowy na kod szeregowy.

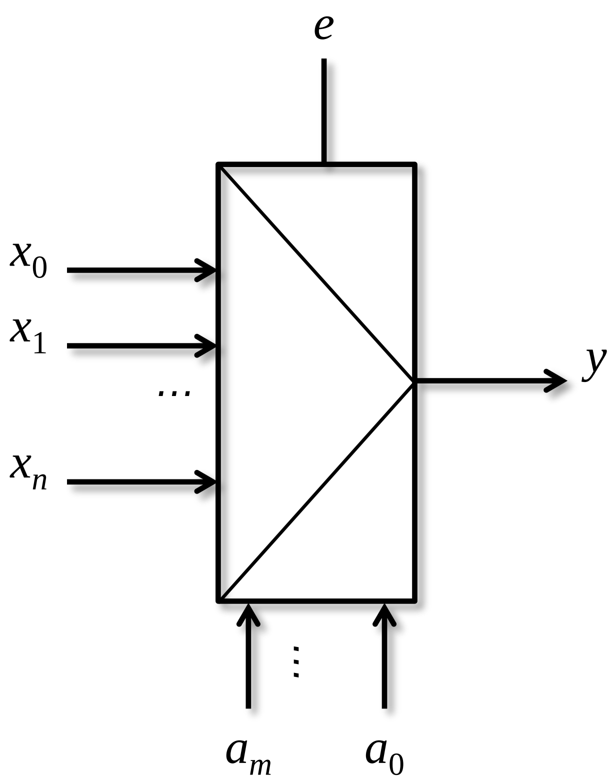
Demultiplekser, układem przetwarzającym kod szeregowy na równoległy.

Oba układy to przełączniki sterowane kodowo za pomocą dekodera adresu.

Układy te pozwalają na zrealizowanie tzw. multipleksowego systemu transmisji danych.



Multiplekser (MUX)



Multiplekser – funkcjonalny blok kombinacyjny posiadający:

m wejść adresowych,

$n = 2^m$ wejść informacyjnych,

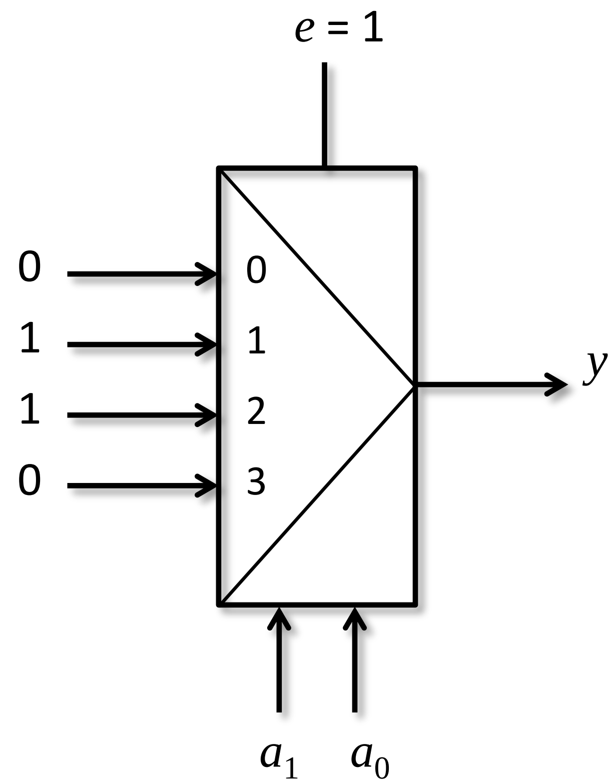
jedno wyjście informacyjne,

e wejście zezwolenia (*enable*).

$x_0, x_1, \ldots, x_n$ – wejścia informacyjne

$a_0, a_1, \ldots, a_\mathrm{m}$ – wejścia adresowe

Multiplekser (MUX)

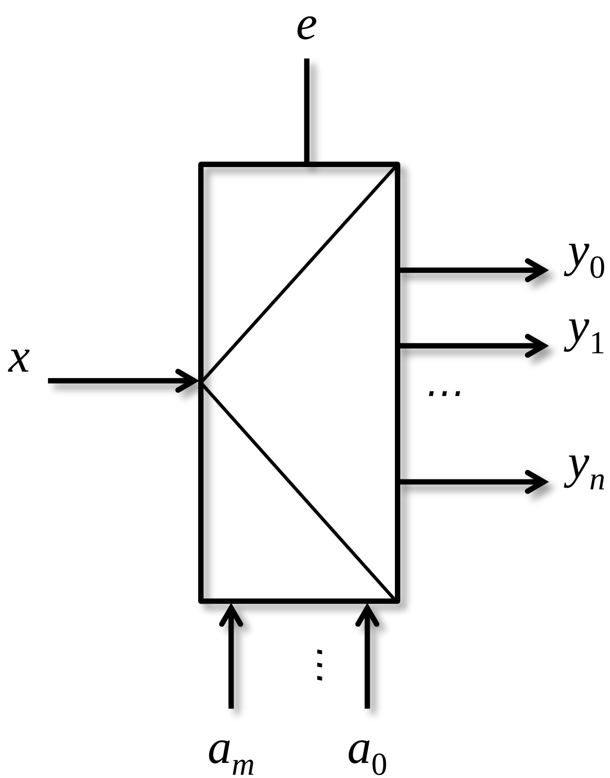


Multiplekser jako ***przełącznik*** realizuje funkcję $y = /a_1a_0d + a_1a_0d.$

Stan na wyjściu y będzie taki, jak stan wejścia d wskazywanego (wybranego) stanem na wejściach adresowych a_k.

Multipleksery łączy się kaskadowo w celu eliminacji tworzenia multiplekserów o dużej liczbie wejść adresowych. (*dlaczego*?)

Demultiplekser (DMX)



Demultiplekser – funkcjonalny blok kombinacyjny posiadający:

m wejść adresowych,

jedno wejście informacyjne,

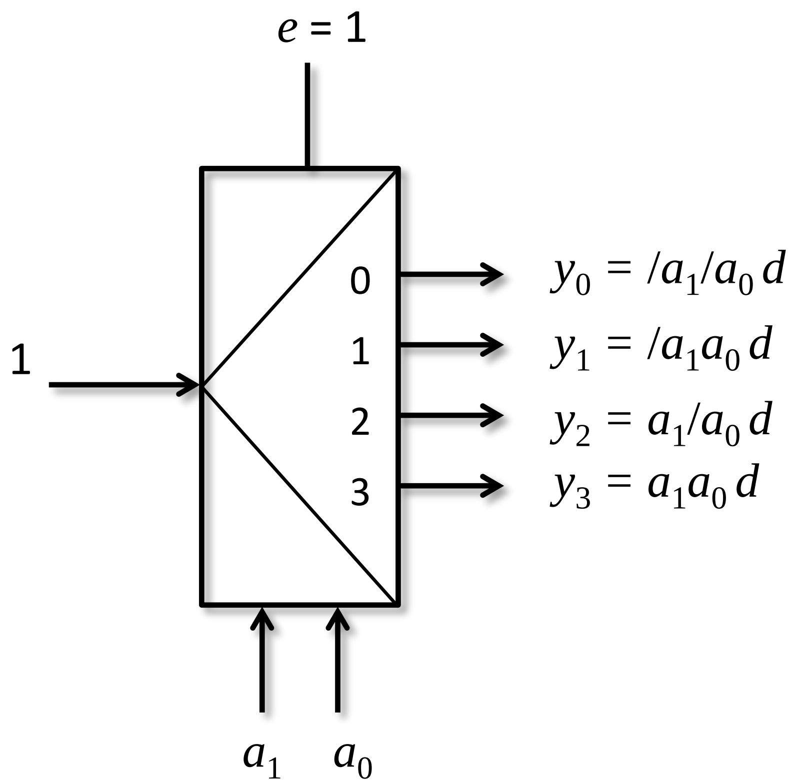
$n = 2^m$ wyjść informacyjnych,

e wejście zezwolenia (*enable*).

$y_0, y_1, \ldots, y_n$ – wyjścia informacyjne

$a_0, a_1, \ldots, a_\mathrm{m}$ – wejścia adresowe

Multiplekser (MUX)



Multiplekser jako ***przełącznik***.

Stan na wyjściu y_k wskazywanym (wybranym) stanem na wejściach adresowych a_k,

będzie taki, jak stan wejścia d.

Kody, kodery i reszta

Kody

- dwójkowo – dziesiętne (BCD)
- refleksyjne
- detekcyjne i korekcyjne

Kody

Aby urządzenie cyfrowe, zbudowane z elementów dwustanowych, mogło wykonywać operacje na liczbach dziesiętnych, każda cyfra dziesiętna musi być przedstawiona przy pomocy liczb dwójkowych.

Dla cyfr od 0 do 9 należy użyć czterobitowych liczb dwójkowych.
Z 16 kombinacji bitów w tetradzie dwójkowej wystarczy 10 kombinacji.

Jaka jest zatem liczba możliwych kodów BCD?

Kody

O wyborze rodzaju kodu decydują jego własności:

łatwość wykonania operacji arytmetycznych,

odporność na zakłócenia,

wykrywanie i korekcja błędów.

Kody dwójkowo – dziesiętne (BCD)

łatwość wykonywania operacji arytmetycznych

Kod 8421, znany jako BCD – kod wagowy (istnieje bezpośredni związek pomiędzy wagą a pozycją cyfry).

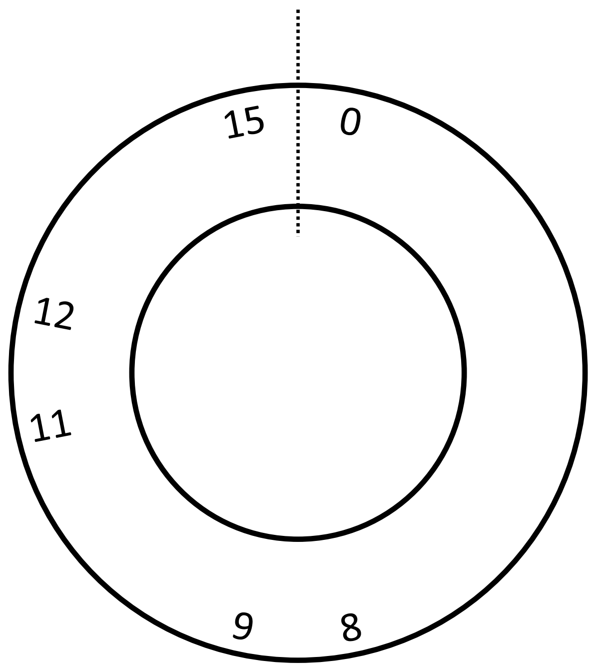
Wagi jak w kodzie binarnym, stąd łatwość wykonywania operacji arytmetycznych, tymi samymi metodami, co dla liczb dwójkowych.

Kod „+3” ($D = B + 3$) – kod niewagowy, samouzupełniający (do 9 przez negację wszystkich bitów)

Kod 2421 – kod wagowy, samouzupełniający, przydatny w układach zliczających.

Wszystkie mają małą odporność na zakłócenia – np. zmiany na pozycjach mogą nie występować jednocześnie – zmiana 0111 na 1111, zamiast na 1000 (w sterowaniu to problem).

własności kodów 8421, +3 i 2412



Kody refleksyjne

odporność na zakłócenia

Możliwość powstawania błędów niejednoczesnej zmiany na pozycjach kodu jest wyeliminowana w kodach, w których nie więcej niż jeden bit zmienia swoją wartość przy przejściu między kolejnymi zakodowanymi wartościami.

Przykładem kodu refleksyjnego jest kod Graya.

0	0 0 0 0
1	0 0 0 1
2	0 0 1 1
3	0 0 1 0
4	0 1 1 0
5	0 1 1 1
6	0 1 0 1
7	0 1 0 0
8	1 1 0 0
9	1 1 0 1
10	1 1 1 1
11	1 1 1 0
12	1 0 1 0
13	1 0 1 1
14	1 0 0 1
15	1 0 0 0

Kody detekcyjne i korekcyjne

wykrywanie i korekcja błędów

Przykładami kodów detekcyjnych są:

kod w kontrolą parzystości,

kody „1 z 10” i kod „2 z 5”.

Przykładem kodu korekcyjnego jest:

kod Hamminga

	z kontrolą parzystości	„1 z 10”	„2 z 5”
0	0 0 0 0 0	0 0 0 0 0 0 0 0 0 **1**	0 0 0 1 1
1	0 0 0 1 1	0 0 0 0 0 0 0 0 **1** 0	0 0 1 0 1
2	0 0 1 0 1	0 0 0 0 0 0 0 **1** 0 0	0 1 0 0 1
3	0 0 1 1 0	0 0 0 0 0 0 **1** 0 0 0	1 0 0 0 1
4	0 1 0 0 1	0 0 0 0 0 **1** 0 0 0 0	0 0 1 1 0
5	0 1 0 1 0	0 0 0 0 **1** 0 0 0 0 0	0 1 0 1 0
6	0 1 1 0 0	0 0 0 **1** 0 0 0 0 0 0	0 1 0 1 0
7	0 1 1 1 1	0 0 **1** 0 0 0 0 0 0 0	0 1 1 0 0
8	1 0 0 0 1	0 **1** 0 0 0 0 0 0 0 0	1 0 1 0 0
9	1 0 0 1 0	**1** 0 0 0 0 0 0 0 0 0	1 1 0 0 0

	Hamminga
0	0 0 0 0 0 0 0
1	0 0 0 0 1 1 1
2	0 0 1 1 0 0 1
3	0 0 1 1 1 1 0
4	0 1 0 1 0 1 0
5	0 1 0 1 1 0 1
6	0 1 1 0 0 1 1
7	0 1 1 0 1 0 0
8	1 0 0 1 0 1 1
9	1 0 0 1 1 0 0

Enkoder, dekoder i transkoder

Enkoder – kod wejściowy to „1 z *n*”, wyjściowy dowolny dwójkowy.

głównie do wprowadzania danych w postaci liczb dziesiętnych (kod wyjściowy to najczęściej 8421)

Dekoder – wejściowy dowolny dwójkowy, kod wyjściowy to „1 z *n*”.

Transkoder – kod wejściowy inny niż „1 z *n*”, wyjściowy inny niż „1 z 10”

Układy arytmetyczne

- sumator
- subtraktor
- komparator

a	b	c_i	s	c_{i+1}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

s

c_i \ $a\ b$	00	01	11	10
0	0	1	0	1
1	1	0	1	0

$$f_s = /ab/c_i + a/b/c_i + /a/bc_i + abc_i = c_i \oplus a \oplus b$$

c_{i+1}

c_i \ $a\ b$	00	01	11	10
0	0	0	1	0
1	0	1	1	1

$$f_c = ab + ac_i + bc_i$$

Układy sekwencyjne

Sprzężenie zwrotne

STAN STABILNY

stan jednoznacznie określony – jak w układach kombinacyjnych

STAN BISTABILNY

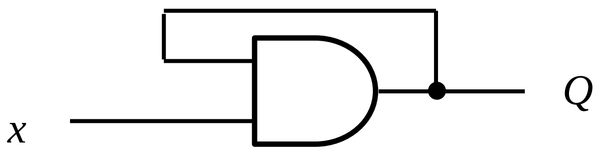
stan 0 lub 1 zależny od stanu poprzedniego (stanu zapamiętanego, wewnętrznego)

STAN ASTABILNY

wystąpienie sprzeczności $Q = /Q$

Konsekwencje sprzężenia zwrotnego

x	Q'
0	0
1	Q



$$Q' = x \cdot Q$$

$Q \equiv Q\,(t) \qquad Q' \equiv Q\,(t+1)$

dla $x = 0$, $Q' = Q \cdot 0 = 0$ STAN STABILNY

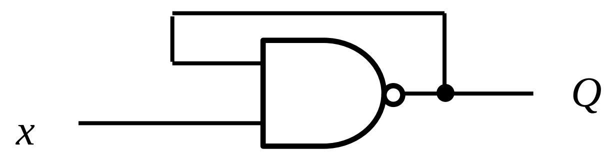
dla $x = 1$, $Q' = Q \cdot 1 = Q$ STAN BISTABILNY

stan 0 lub 1 zależny od stanu poprzedniego (stanu zapamiętanego)

Występowanie STANU BISTABILNEGO charakteryzuje układ sekwencyjny.

Jeżeli w układzie występuje STAN BISTABILNY, to układ nie jest już kombinacyjny.

Konsekwencje sprzężenia zwrotnego



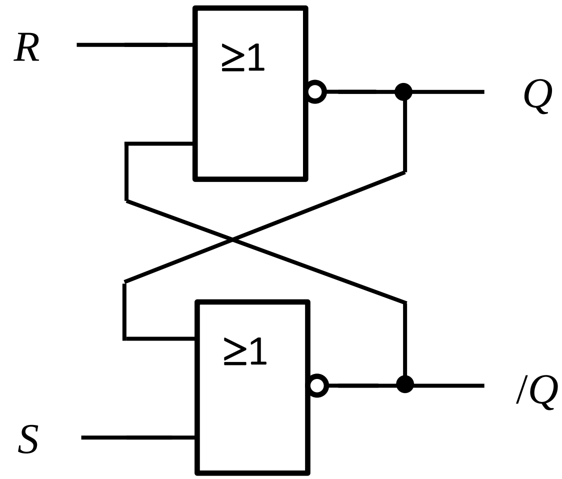
$$Q' = /(x \cdot Q)$$

dla $x = 0$, $Q' = /(Q \cdot 0) = /0 = 1$ STAN STABILNY

dla $x = 1$, $Q' = /(Q \cdot 1) = /Q$ STAN ASTABILNY

Asynchroniczny przerzutnik RS

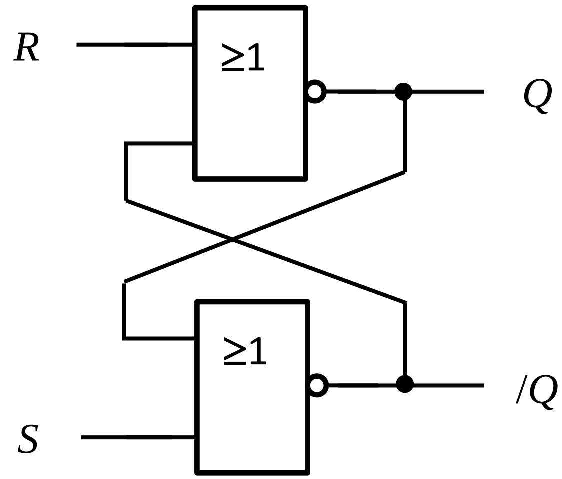
W przypadku przerzutnika RS wszystkie kombinacje stanów początkowych Q i /Q oraz pobudzeń R, S (badane tzw. lawiną zmian stanów), prowadzą do osiągania stanów stabilnych bądź bistabilnych, bez niebezpieczeństwa osiągania stanów astabilnych.



S	R	Q	$/Q$	
0	0	Q	$/Q$	STAN BISTABILNY
0	1	0	1	STANY STABILNE
1	0	1	0	
1	1	0	0	

Asynchroniczny przerzutnik RS

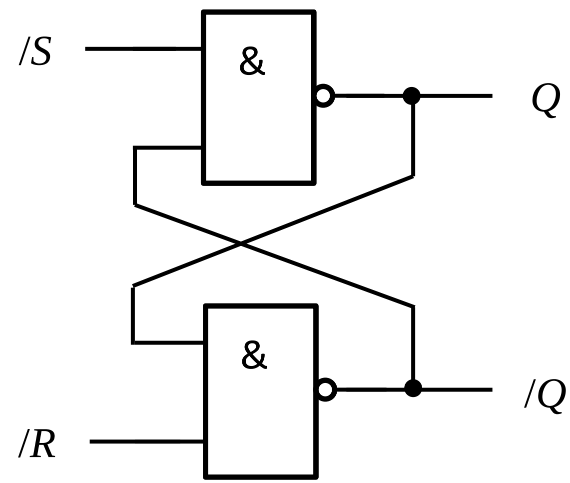
Elementarny i w pełni funkcjonalny układ sekwencyjny zdolny do pamiętania jednego bitu informacji



S	R	Q	$/Q$	
0	0	Q	$/Q$	„pamiętaj” stan
0	1	0	1	reset (ustaw Q = 0)
1	0	1	0	set (ustaw Q = 1)
1	1	0	0	stan zabroniony – pobudzenie sprzeczne

Asynchroniczny przerzutnik RS

Elementarny i w pełni funkcjonalny układ sekwencyjny zdolny do pamiętania jednego bitu informacji

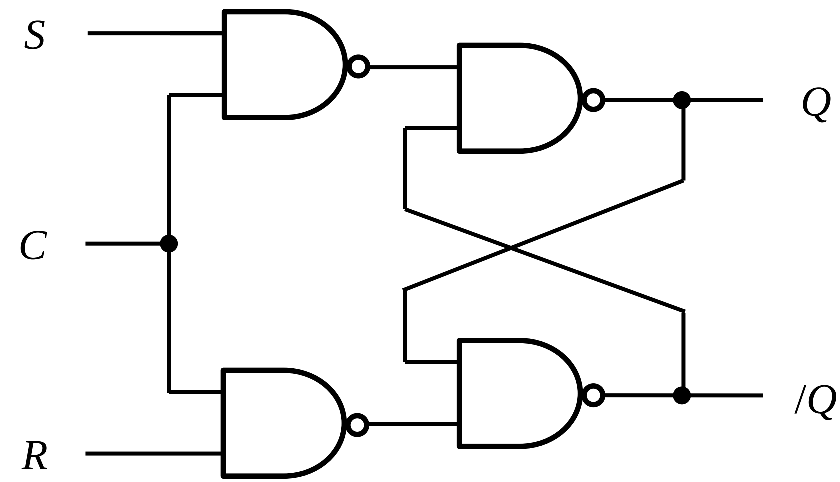


set i reset aktywne w stanie niskim

/S	/R	Q	/Q
0	0	1	1
0	1	1	0
1	0	0	1
1	1	Q	/Q

Synchroniczny przerzutnik RS

$/S$	$/R$	Q_{t+1}
0	0	Q_t
0	1	0
1	0	1
1	1	x



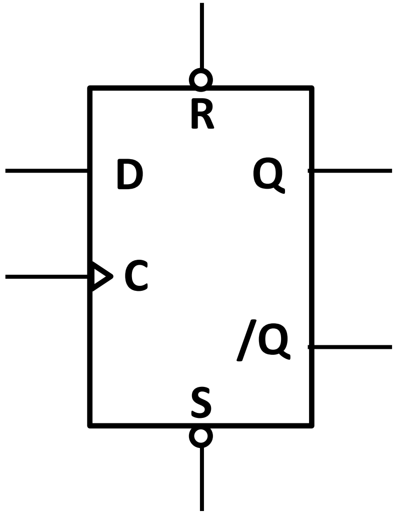
Dla C = 0 zmiany sygnałów R i S nie mają wpływu na stan Q.

Dla C = 1 zmiany zachodzą zgodnie z powyższą tabelą.

Zmiana sygnału C z 1 na 0 powoduje zatrzaśnięcie stanu wyjścia – układ typu *latch*.

Przerzutnik D

D	Q_t	Q_{t+1}
0	0	0
0	1	0
1	0	1
1	1	1



Wyjście Q przyjmuje wartość D.

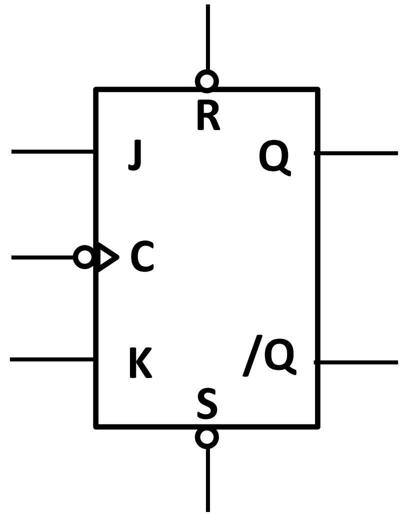
Przerzutnik posiada dwa stany.

Zmiana stanu następuje (tu:) ze zboczem narastającym sygnału C.

Przerzutnik posiada asynchroniczne wejścia: zerujące ($Q = 0$), RESET i ustawiające ($Q = 1$), SET.

Przerzutnik JK

J	K	Q_{t+1}
0	0	Q_t
0	1	0
1	0	1
1	1	$/Q_t$



$$Q_{t+1}\ J\ /Q_t + /K\ Q_t$$

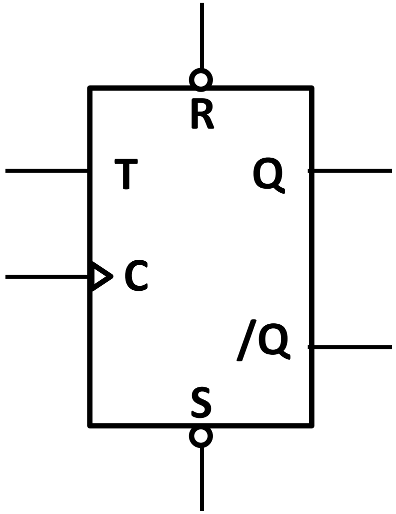
Przerzutnik posiada dwa stany.

Zmiana stanu następuje (tu:) ze zboczem opadającym sygnału C.

Przerzutnik posiada asynchroniczne wejścia: zerujące ($Q = 0$), RESET i ustawiające ($Q = 1$), SET.

Przerzutnik T (Trigger)

D	Q_t	Q_{t+1}
0	0	0
0	1	1
1	0	1
1	1	0



$Q_{t+1}\ J\ /Q_t + /K\ Q_t$

Przerzutnik posiada dwa stany.

Zmiana stanu następuje (tu:) ze zboczem narastającym sygnału C.

Przerzutnik posiada asynchroniczne wejścia: zerujące (Q = 0), RESET i

ustawiające (Q = 1), SET.

Formalny opis układów sekwencyjnych

$X = < x_0, x_1, \ldots, x_{n\text{-}1} >$ wektor n binarnych sygnałów WE

$Y = < y_0, y_1, \ldots, y_{m\text{-}1} >$ wektor m binarnych sygnałów WY

$\mathbf{X} = \{ \mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_N \}$ zbiór stanów (słów) WE ($N \leq 2^n$) – alfabet wejściowy

$\mathbf{Y} = \{ \mathbf{Y}_1, \mathbf{Y}_2, \ldots, \mathbf{Y}_M \}$ zbiór stanów (słów) WY ($M \leq 2^m$) – alfabet wyjściowy

$\mathbf{Q} = \{ \mathbf{Q}_1, \mathbf{Q}_2, \ldots, \mathbf{Q}_N \}$ zbiór stanów wewnętrznych – alfabet wewnętrzny

Stan wewnętrzny $\mathbf{Q}$ reprezentowany jest wektorem bitowym złożonym z sygnałów binarnych Q_i symbolizujących stan elementarnych układów pamięci (przerzutników).

Pracę układu sekwencyjnego opisują dwie funkcje:

δ: $\mathbf{Q} \times \mathbf{X} \to \mathbf{Q}$ funkcja przejść oraz

λ: $\mathbf{Q} \times \mathbf{X} \to \mathbf{Y}$ funkcja wyjść

Funkcja przejść określa stan, do którego układ przechodzi w zależności od stanu bieżącego oraz pobudzenia na wejściu: $\delta(\mathbf{Q} \times \mathbf{X}) = \mathbf{Q}'$.

Funkcją wyjść określa stan wyjść układu w funkcji stanu bieżącego oraz pobudzenia na wejściu: $\lambda(\mathbf{Q} \times \mathbf{X}) = \mathbf{Y}$.

Układ cyfrowy definiuje się jako piątkę $A = (\mathbf{Q}, \mathbf{X}, \mathbf{Y}, \delta, \lambda)$.

Systemy skończenie stanowe

…, który znajdować się może w jednej ze skończonych konfiguracji, czyli stanów.

Stan zawiera informacje dotyczące stanów poprzednich, niezbędną do określenia zachowania systemu przy kolejnych pobudzeniach.

Przykłady:

mechanizm sterujący windy – pamięta aktualną pozycję, kierunek ruchu i listę niezrealizowanych żądań

system sterowania światłami na skrzyżowaniu – aktualny stan, czas przebywania w nim

komputer (?)

ludzki mózg (???)

Water Jug Problem

Posiadamy dwa pojemniki, nieprzezroczyste, nieforemne, bez oznaczeń, o pojemnościach 3 i 4 jednostki. Mamy nieograniczony dostęp do płynu. W pojemniku 4 mamy umieścić 2 jednostki płynu.

Możliwe operacje to:
(**p**) napełnij 3,
(**k**) napełnij 4,
(**t**) opróżnij 3,
(**j**) opróżnij 4,
(**i**) uzupełnij 3,
(**e**) uzupełnij 4,
(**a**) przelej do wypełnienia z 3 do 4,
(**o**) przelej do wypełnienia z 4 do 3.

(#) – oznaczenie działania

Water Jug Problem

Mamy dwie drogi rozwiązania zadania złożone z łąńcuch działań: **papaja** i **kosokot**. Oprócz nich, nieskończenie wiele innych rozwiązań zawierających bezużyteczne cykle.

Powstał pewien język składający się ze wszystkich łańcuchów będących etykietami dróg.

Dwie kwestie, którymi rozwiązanie WTP różni się od typowego systemu skończenie stanowego.

1. posiada tylko jeden stan końcowy – raczej jest ich wiele.
2. każda droga ma swój odpowiednik powrotny – raczej rzadkie.

Automat skończony

Automat skończony składa się ze skończonego zbioru stanów i zbioru przejść (ruchów/działań/operatorów), ze stanu do stanu zachodzących przy różnych symbolach wejściowych wybranych z pewnego alfabetu Σ.

Dla każdego symbolu wejściowego istnieje dokładnie jedno przejście odpowiadające temu symbolowi. Przejście to może prowadzić do tego samego stanu.

Wyróżniony jest stan początkowy, oznaczony q_0.

Z każdym automatem skończonym związany jest diagram przejść, będący grafem skierowanym, w którym wierzchołki odpowiadają stanom automatu, natomiast krawędzie odpowiadają przejściom pomiędzy stanami, jeśli takie przejście istnieje.

Stany automatu oznaczamy symbolami q_i, etykiety krawędzi symbolami z_k.

Automat skończony akceptuje łańcuch s, jeżeli ciąg przejść odpowiadający symbolom łańcucha s prowadzi od stanu początkowego q_0 do dowolnego stanu końcowego.

Automat skończony przedstawiamy formalnie jako piątkę uporządkowaną

$(Q, \mathbf{\Sigma}, \delta, q_0, F)$, gdzie:

Q – skończony zbiór stanów,

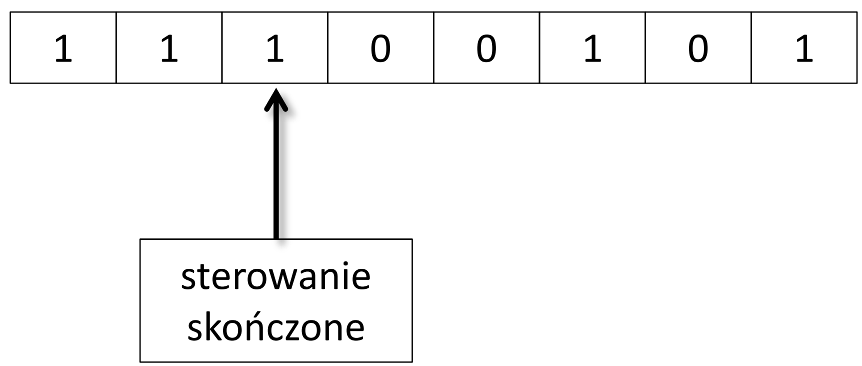
$\mathbf{\Sigma}$ – skończony alfabet wejściowy,

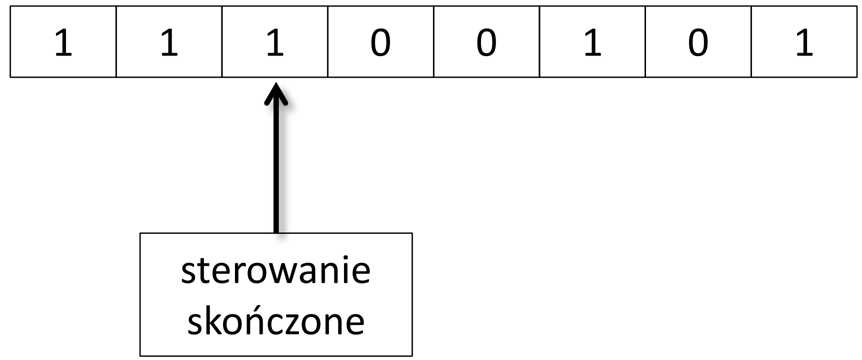
δ – funkcja przejść odwzorowująca $Q\ x\ \mathbf{\Sigma}$ w Q,

q_0 – stan początkowy,

F – zbiór stanów końcowych $F \subseteq Q$.

Automat skończony można wyobrazić sobie jako układ skończonej liczbie stanów (sterowanie skończone), który znajduje się w pewnym stanie należącym do Q i czyta ciąg symboli z $\boldsymbol{\Sigma}$ zapisany na taśmie.





AS w stanie q czyta symbol z, przechodzi w stan $\delta(q, z)$ i przesuwa głowicę o jeden symbol w prawo.

Jeśli $\delta(q, z)$ jest stanem końcowym, to uważamy, że AS zaakceptował łańcuch zapisany na taśmie, z wyłączeniem stanu na który przesunęła się właśnie głowica.

Mówimy, że łańcuch s jest akceptowany przez automat skończony $M = (Q, \boldsymbol{\Sigma}, \delta, q_0, F)$, ,

jeżeli $\delta(q, s) = p$ dla jakiegoś p należącego do F.

Język akceptowany przez M, oznaczany $L(M)$, to zbiór $\{ s \mid \delta(q_0, s) \subset F \}$.

Język nazywamy regularnym* (zbiorem regularnym), jeśli jest on językiem akceptowanym przez jakiś automat skończony.

* termin „regularny” pochodzi od „wyrażeń regularnych”, które określa tę samą klasę języków, co automaty skończone.

Automat skończony z wyjściem

A automacie skończonym sygnał wyjściowy przyjmuje wartość zero bądź jeden (akceptuje / nie akceptuje). Nie zawsze wystarczająco.

Stąd rozwiązania z wyborem sygnału wyjściowego z pewnego alfabetu.

1. wyjście związane ze stanem – Automat Moore'a
2. wyjście związane z przejściem – Automat Mealy'ego

Automat Moore'a (1)

Automat Moore'a to (formalnie) szóstka uporządkowana

$(Q, \Sigma, \delta, \Delta, q_0, \lambda)$, gdzie:

Q – skończony zbiór stanów,
Σ – skończony alfabet wejściowy,

Δ – skończony alfabet wyjściowy,

λ – odwzorowanie Q w Δ określającym wyjście związane z każdym stanem,

δ – funkcja przejść odwzorowująca $Q \times \Sigma$ w Q,

q_0 – stan początkowy.

Automat Moore'a (2)

Wyjściem M odpowiadającym wejściu $z_1, z_2, \ldots, z_n$, $n \geq 0$, jest $\lambda(q_1)\lambda(q_2)\ldots\lambda(q_n)$, gdzie $q_0, q_1, \ldots, q_n$ jest ciągiem stanów, takim że $\delta(q_{i-1}, z_i) = q_i$, dla $1 \leq i \leq n$

Automat Moore'a (a układ sekwencyjny)

Stany wewnętrzne zależą od stanów WE. Zatem,

δ: $\mathbf{Q}$ x $\mathbf{X} \rightarrow \mathbf{Q}$	funkcja przejść oraz
λ: $\mathbf{Q} \rightarrow \mathbf{Y}$	funkcja wyjść

Automat Mealy'ego (1)

Automat Mealy'ego to (formalnie) szóstka uporządkowana

$(Q, \Sigma, \delta, \Delta, q_0, \lambda)$, gdzie:

Q – skończony zbiór stanów,
Σ – skończony alfabet wejściowy,

Δ – skończony alfabet wyjściowy,

λ – odwzorowanie Q x Σ w Δ,

δ – funkcja przejść odwzorowująca $Q\ x\ \Sigma$ w Q,

q_0 – stan początkowy.

Automat Mealy’ego (2)

λ – odwzorowanie Q x $\boldsymbol{\Sigma}$ w Δ oznacza, że $\delta(q_0, z)$ podaje wejście związane z przejściem

ze stanu q przy wejściu z.

Wyjściem M odpowiadającym wejściu $z_1, z_2, \ldots, z_n$, $n \geq 0$, jest $\lambda(q_0, z_1)\lambda(q_1, z_2)\ldots\lambda(q_{n-1}, z_n)$,

gdzie $q_0, q_1, \ldots, q_n$ jest ciągiem stanów, takim że $\delta(q_{i-1}, z_i) = q_i$, dla $1 \leq i \leq n$

Powyższy ciąg ma długość n, inaczej niż w automacie Moore’a ($n + 1$)

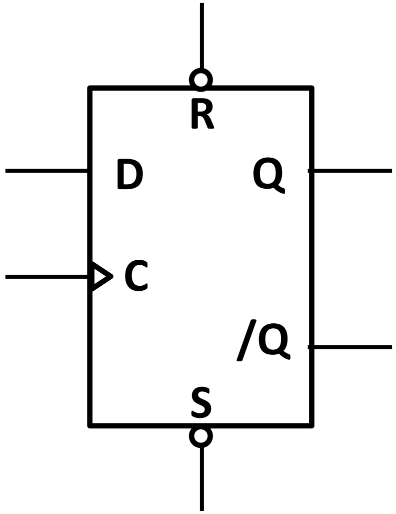
Automat Moore’a vs Automat Mealy’ego

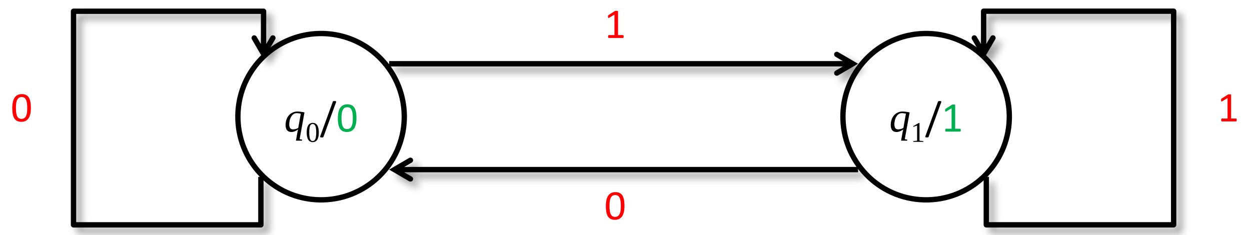
Dla każdego automatu Moore’a istnieje równoważny mu (w znaczeniu odpowiedzi), automat Mealy’go i na odwrót.

Zaletą automatu Moore’a jest odseparowanie wyjść od zmian sygnałów – być może przypadkowych.

Przerzutnik D

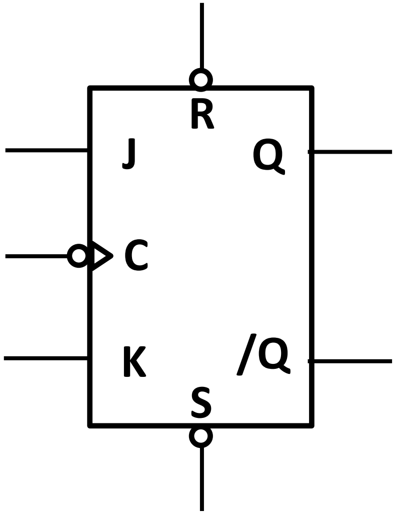
D	Q_t	Q_{t+1}
0	0	0
0	1	0
1	0	1
1	1	1

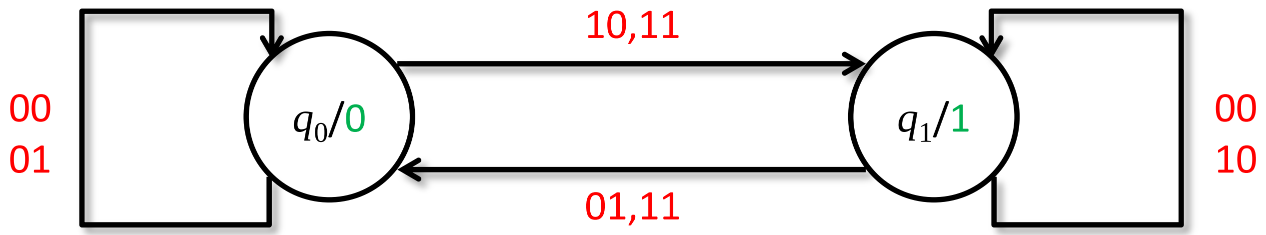




Przerzutnik JK

J	K	Q_{t+1}
0	0	Q_t
0	1	0
1	0	1
1	1	$/Q_t$





Przerzutnik T (Trigger)

D	Q_t	Q_{t+1}
0	0	0
0	1	1
1	0	1
1	1	0