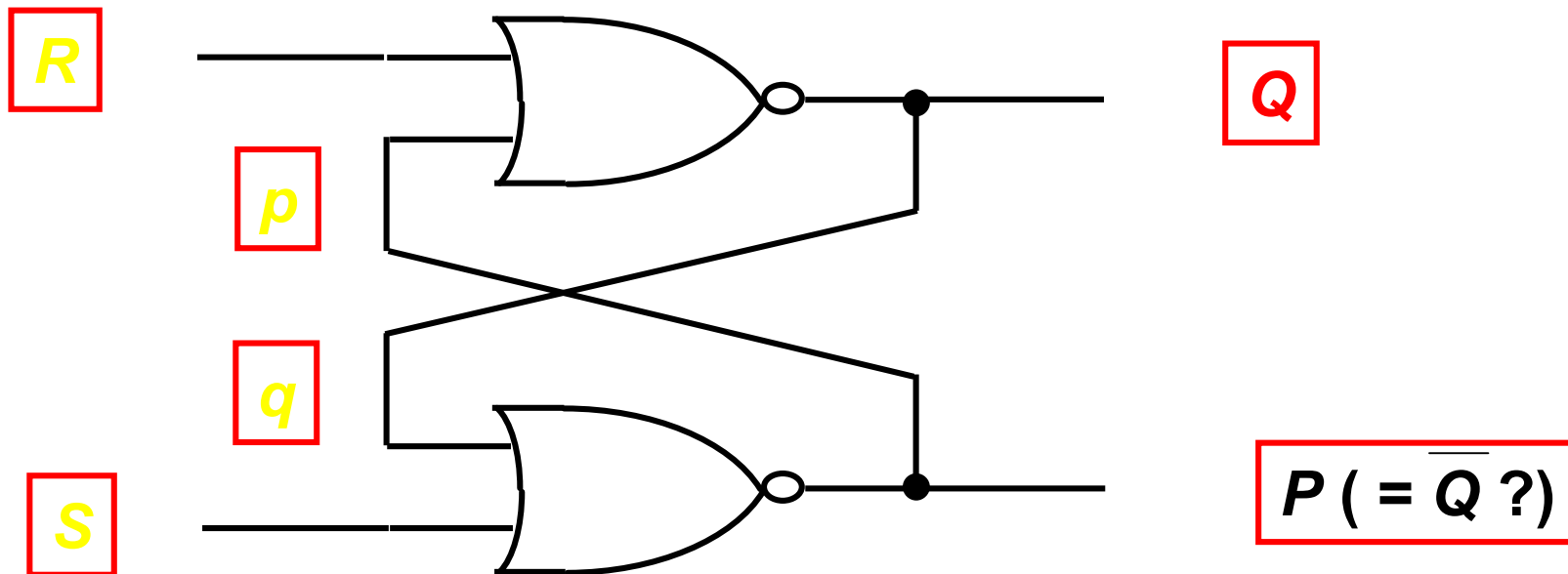


Elemente de memorie - Latch-uri

- Latch RS: 2 intrări (RS), două ieșiri (QP), două conexiuni inverse (qp).
 - Se cere să furnizeze simultan și negația ieșirii, prin P.

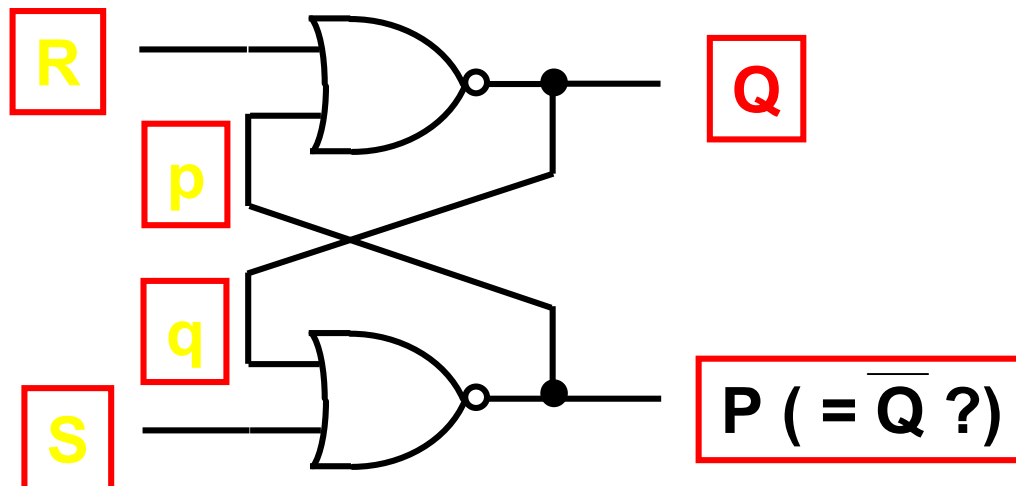
$$Q = \overline{R + p} = \overline{R} \cdot \overline{p}, \quad P = \overline{S + q} = \overline{S} \cdot \overline{q} = (\overline{Q} ?)$$



- Cum aflăm funcționarea circuitului, date valorile lui R și S?

Reprezentarea funcționării elementului de memorie RS

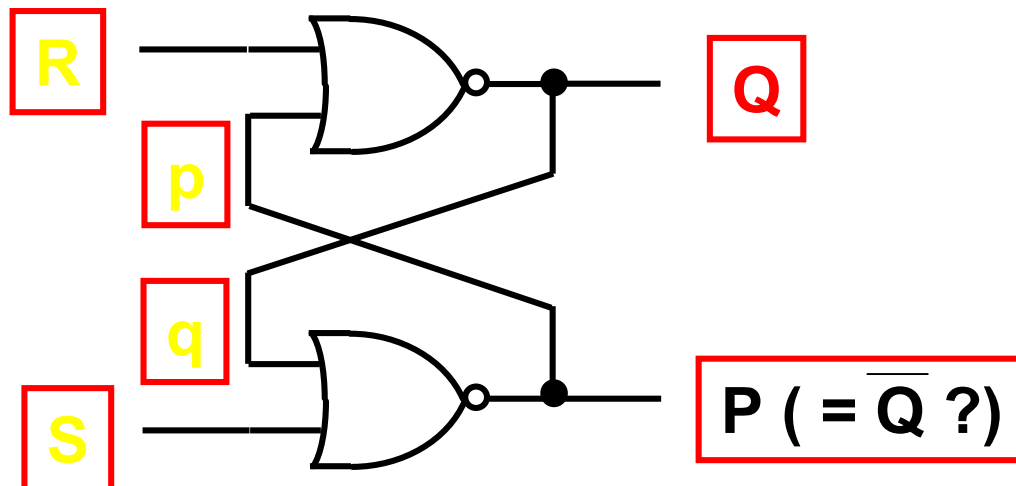
- Se construiește o diagramă Karnaugh în care se scriu ca valori ale funcției perechi de valori QP.
- Prin definiție, $p=P$ și $q=Q$.
 - În stânga egalului, valorile "noi".



qp \ RS	00	01	11	10
00				
01				
11				
10				

Completarea diagramei Karnaugh: Q

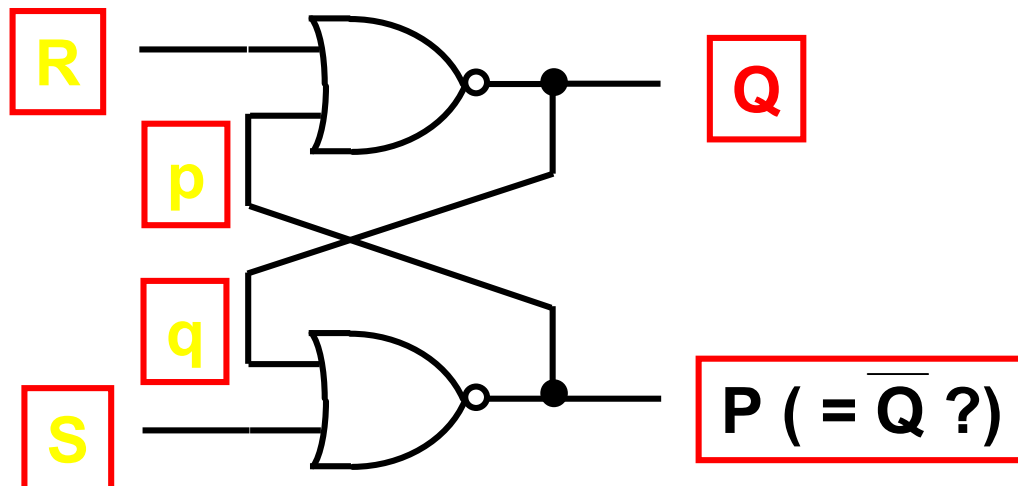
- Știm că $Q = \overline{R} \cdot \overline{p} \dots$
- Blocul (de 4) pentru expresia lui Q



qp \ RS	00	01	11	10
00	1	1	0	0
01	0	0	0	0
11	0	0	0	0
10	1	1	0	0

Completarea diagramei Karnaugh: \overline{Q}

- Știm că $Q = \overline{R} \cdot \overline{p}$ și că $P = \overline{S} \cdot \overline{q}$
- Blocul - cu 4 de 1 - pentru P



$qp \backslash RS$	00	01	11	10
00	11	10	00	01
01	01	00	00	01
11	00	00	00	00
10	10	10	00	00

Stări stabile

- Perechile de valori din diagrama Karnaugh reprezintă ieșirile **QP**.
 - Care dau apoi *stările* qp.
- Interesează *situațiile stabile*: $qp = QP$

qp \ RS	00	01	11	10
00	11	10	00	01
01	01	00	00	01
11	00	00	00	00
10	10	10	00	00

← QP

Identificarea stărilor stabile

- Neexistând ceas (circuit asincron), latch-ul RS oferă outputul când se stabilizează
 - Se caută în diagrama Karnaugh acele locații pentru care $QP = qp$

qp
↓

	00	01	11	10	RS
00	11	10	00	01	QP
01	01	00	00	01	QP = qp !!!
11	00	00	00	00	
10	10	10	00	00	

Funcționarea pentru $R = 0$ și $S = 1$

- Poziționare Q la **1** (setare): $Q = 1, \bar{Q} = 0$

qp

00
01
11
10

00 01 11 10 ← RS

00	11	10	00	01
01	01	00	00	01
11	00	00	00	00
10	10	10	00	00

Funcționarea pentru $R = 1$ și $S = 0$

- Poziționare Q la **0** (resetare): $Q = 0, \bar{Q} = 1$

qp
↓

00 01 11 10 ← RS

00	11	10	00	01
01	01	00	00	01
11	00	00	00	00
10	10	10	00	00

Diagram illustrating the RS flip-flop truth table for the reset condition ($R = 1, S = 0$). The table shows the next state (Q, \bar{Q}) for all combinations of current state (Q, \bar{Q}) and inputs (R, S). The input combination $R=1, S=0$ is highlighted in green. The output state $Q=0, \bar{Q}=1$ is highlighted in red. Red arrows indicate the transition from the current state to the next state for the reset condition.

Funcționarea pentru $R = 0$ și $S = 0$

- Q și \bar{Q} rămân la valorile duale anterioare
 - Provenite din $RS=10$ sau $RS=01$
 - Pentru $qp=11$ – la o valoare nedefinită

qp

↓

	<u>00</u>	01	11	10 ← RS
00	11	10	00	01
01	01	00	00	01
11	00	00	00	00
10	10	10	00	00

Funcționarea pentru $R = 1$ și $S = 1$

- Contradicție
 - Ar trebui să avem simultan $Q = \bar{Q} = 0$

A Karnaugh map for an RS flip-flop. The vertical axis is labeled 'qp' with values 00, 01, 11, 10. The horizontal axis is labeled 'RS' with values 00, 01, 11, 10. The '11' value is underlined. The map contains the following cells:

	00	01	<u>11</u>	10
00	11	10	00	01
01	01	00	00	01
11	00	00	00	00
10	10	10	00	00

Annotations: Red circles highlight the '00' cell at (00, 11) and the '01' cells at (01, 00), (00, 01), (10, 00), and (10, 01). Red arrows point from the '00' cell at (01, 11) to the '00' cells at (00, 11), (01, 00), and (10, 00). A green arrow points from the 'RS' label to the '11' header.

Funcționarea pentru $R = 1$ și $S = 1$

- Dacă $RS = 11$ și apoi $RS=00$, rezultatul este nedefinit (săgețile roșii verticale)

qp
↓

00 01 11 10 ← RS

00	11	10	00	01
01	01	?	00	01
11	00		00	00
10	10	10	00	00

Funcționarea circuitului latch RS

- Tabela de adevăr cu starea internă q reprezentată implicit:
 - De ce 5 linii ?

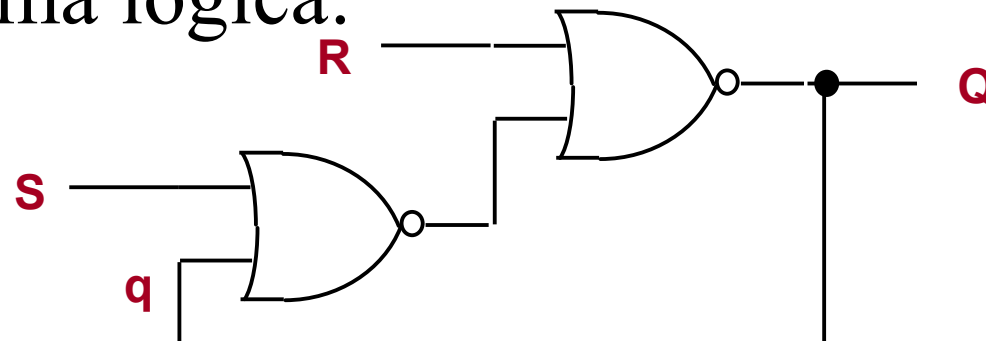
S	R	Q	P	
1	0	1	0	Setare (la "1")
0	1	0	1	Resetare (la "0")
0	0	1	0	Stare neschimbată (după SR = 10)
0	0	0	1	Stare neschimbată (după SR = 01)
1	1	*	*	Combinație imposibilă

Funcționarea latch-ului RS

- Altă reprezentare (fără a considera și \bar{Q})
- $Q = S\bar{R} + \bar{R}q = \bar{R}(S+q)$
 - Din diagrama Karnaugh, conform specificațiilor lui Q

q/SR	00	01	11	10
0	0	0	0	1
1	1	0	0	1

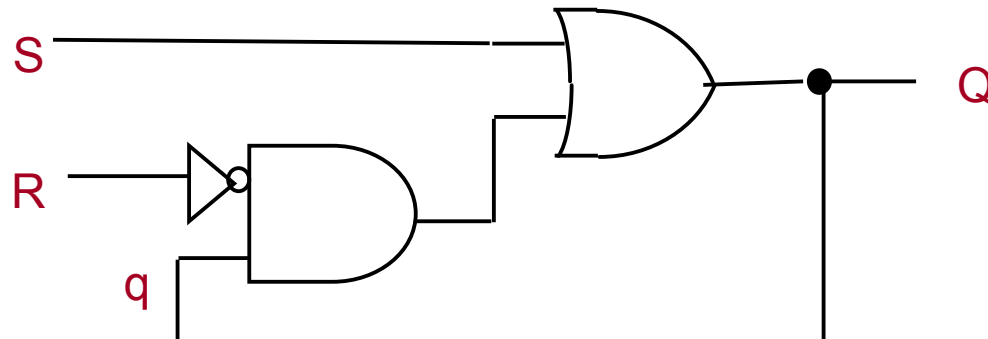
- Diagrama logică:



Altă reducere

- Putem însă considera $Q = S + \overline{R}q$
 - dacă utilizăm combinația imposibilă
 - rezultatul coincide cu specificațiile inițiale pt. $SR=0$

q/SR	00	01	11	10
0	0	0	*	1
1	1	0	*	1

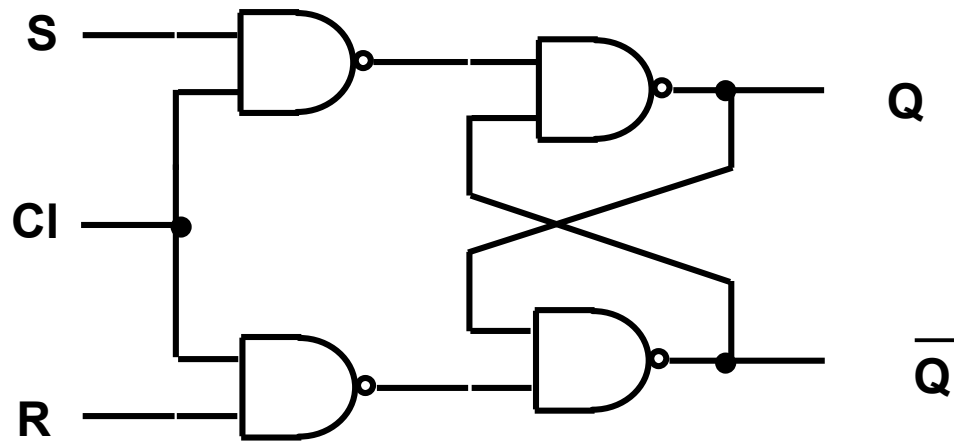


Circuite secvențiale sincrone

- Se adaugă unui latch RS un semnal de sincronizare (impuls ceas - Clock).
- Și flip-flop-urile sunt sincrone - RS, D, T, JK.

Bistabil RS (cu ceas)

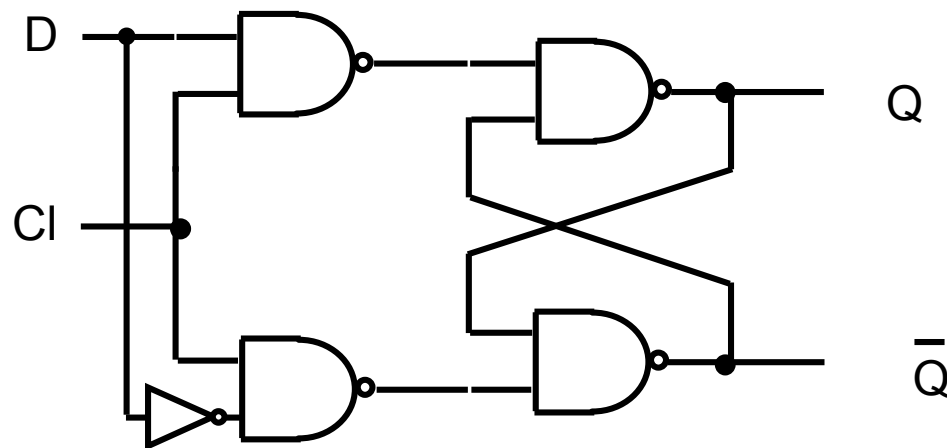
- Porți NAND
- Ceas \rightarrow momente de timp (...t, t+1...)



$q=Q_t$	S	R	Q_{t+1}
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	*
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	*

Bistabil D

- Diagrama logică și tabelul de adevăr.
- Derivat din RS. Modelează situațiile $R \neq S$.

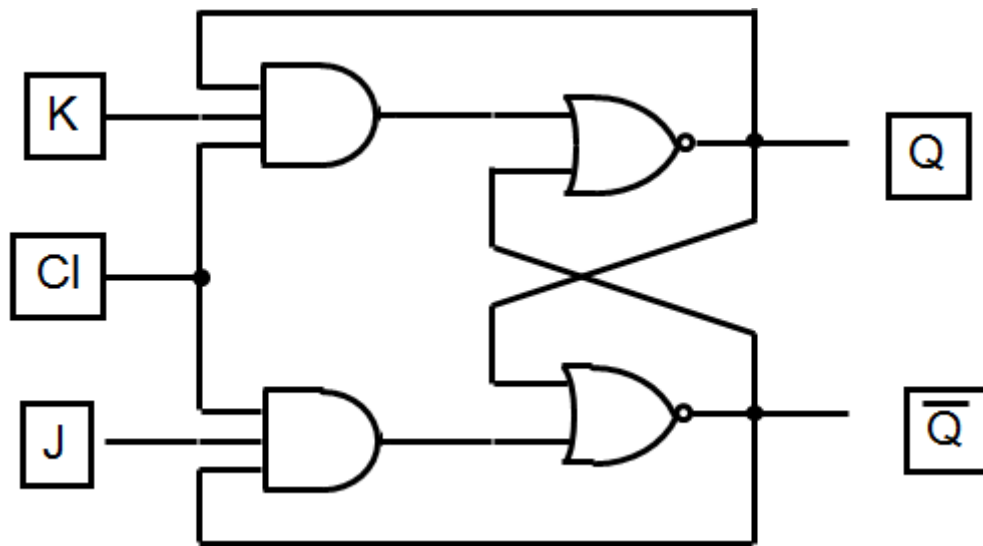


$q=Q_t$	D	Q_{t+1}
0	0	0
0	1	1
1	0	0
1	1	1

- Elimină combinațiile interzise.

Bistabil JK

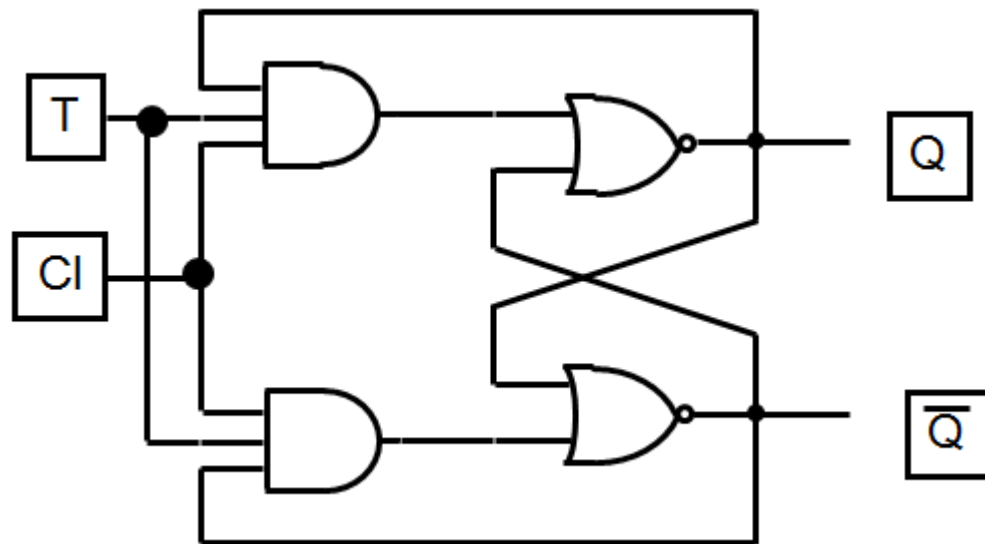
- Diagrama logică și tabelul de adevăr
 - Elimină combinația imposibilă de la RS – feedback-uri suplimentare ($J \leftarrow S$, $K \leftarrow R$)



$q=Q_t$	J	K	Q_{t+1}
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

Bistabil T

- Derivat din JK, analog derivării lui D din RS
- Modelează situațiile $J=K$ de la JK



$q=Q_t$	T	Q_{t+1}
0	0	0
0	1	1
1	0	1
1	1	0

Tabele de adevăr temporale

- Indică evoluția lui $Q_{(t+1)}$ în funcție de intrări (independente sau nu).
- Singura combinație imposibilă – la RS.

S	R	$Q_{(t+1)}$
0	0	$Q_{(t)}$ (neschimbată)
0	1	0 (memorare 0)
1	0	1 (memorare 1)
1	1	* (imposibil)

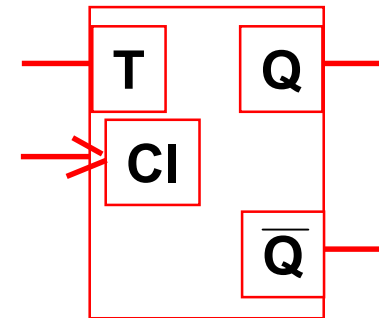
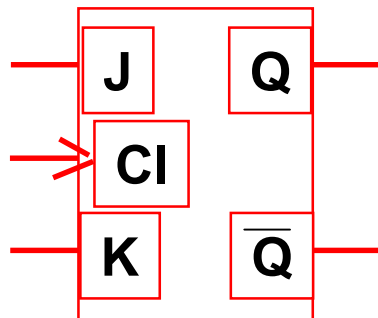
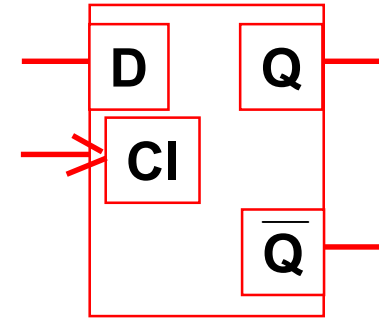
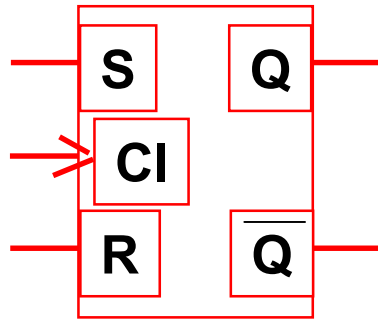
J	K	$Q_{(t+1)}$
0	0	$Q_{(t)}$ (neschimbată)
0	1	0 (memorare 0)
1	0	1 (memorare 1)
1	1	$\overline{Q_{(t)}}$ (negare)

D	$Q_{(t+1)}$
0	0 (memorare 0)
1	1 (memorare 1)

T	$Q_{(t+1)}$
0	$Q_{(t)}$ (neschimbată)
1	$\overline{Q_{(t)}}$ (negare)

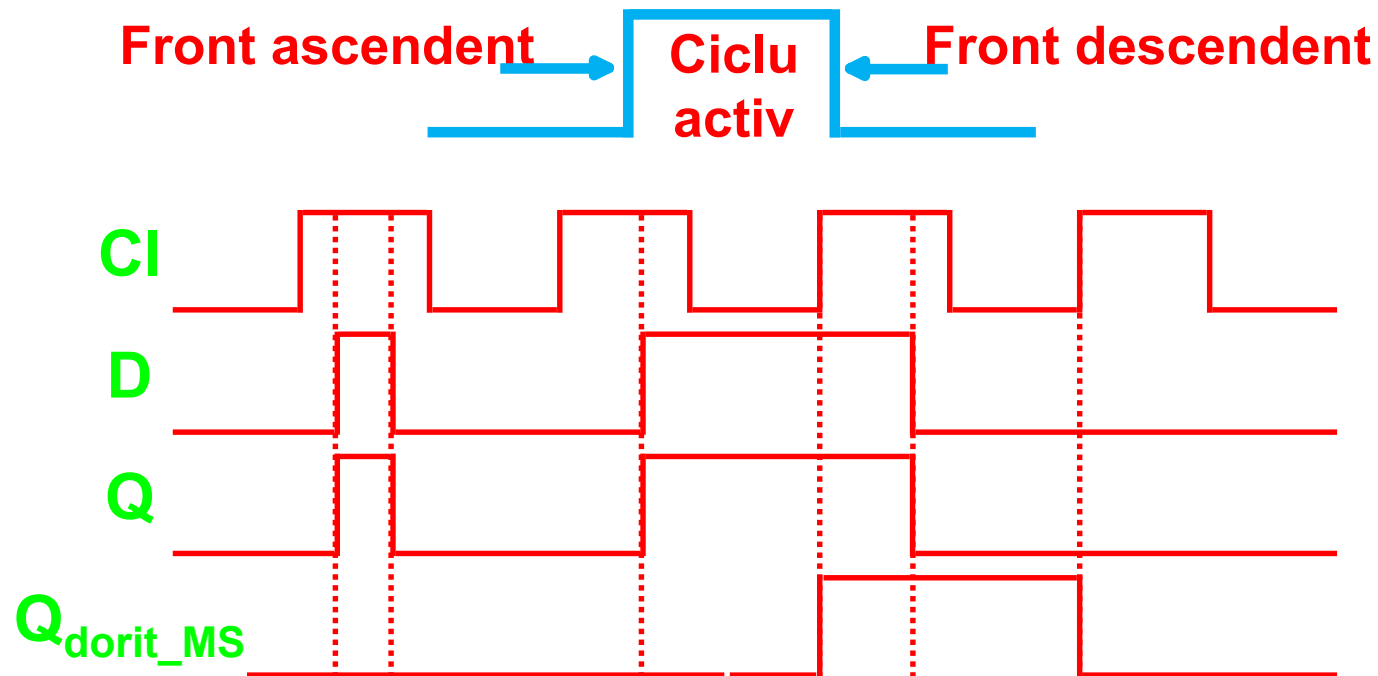
Diagrame bloc pentru bistabili

- Se utilizează simbolii:



Probleme de funcționare

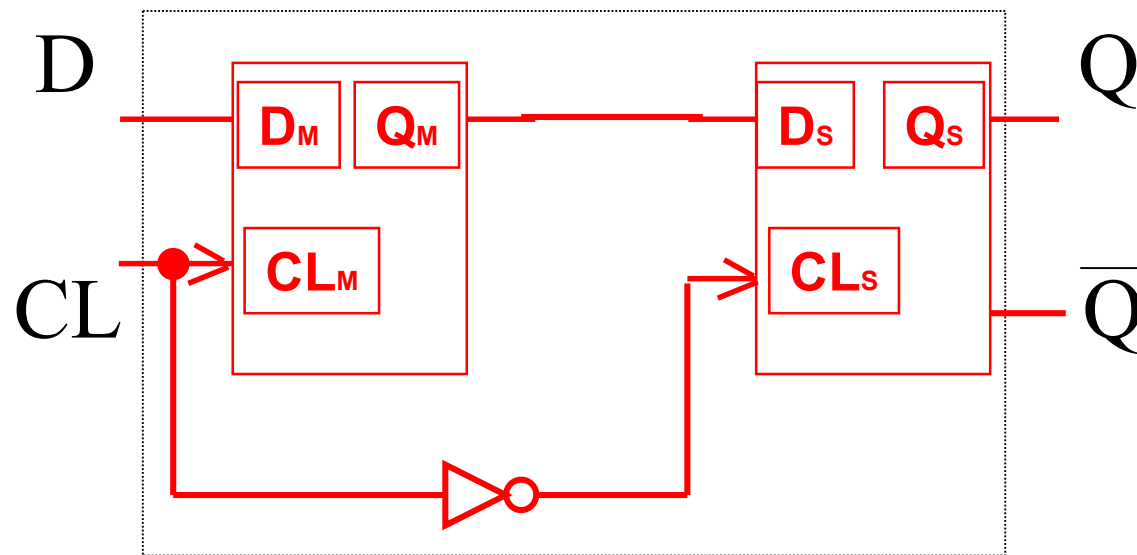
- Pentru a obține un flip-flop D
 - » din latch-uri D cu ceas
 - » ignorare; întârziere (front ascendent)



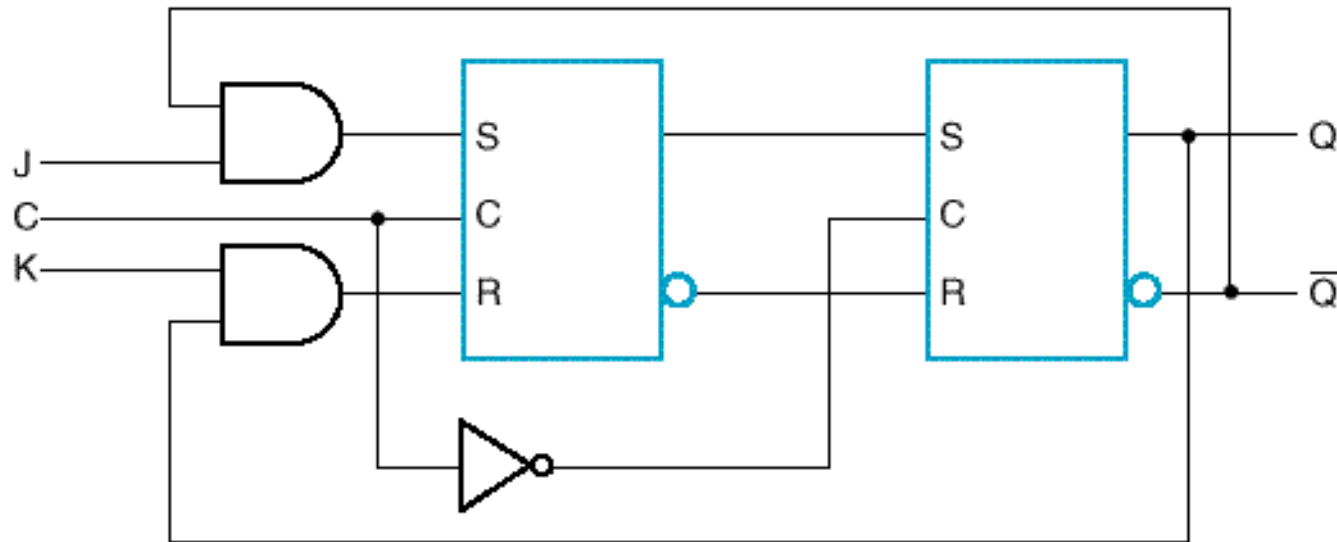
Flip-flop master-slave D

Reduce probabilitatea activării la variații bruște de input.

Face posibilă proiectarea unor circuite secvențiale complexe (ex.: regiștrii cu deplasare).



Flip-flop JK master-slave



(a)

		Next State of Q
J	K	
0	0	Q
0	1	0
1	0	1
1	1	\bar{Q}

Utilizarea controlului prin ceas

- Depinde de elementele de construcție folosite
- Larg folosite flip-flop-urile active pe front
 - crescător sau descrescător
 - de exemplu, la componentele logice programabile (programmable logic devices)
- Numeroase circuite integrate uzuale folosesc latch-uri
 - active pe nivel
 - asincronism (I/O, delay)

Latch-uri și flip-flop-uri

- Circuite D
 - Latch: 4 tranzistoare
 - Flip-flop: utilizat în elemente programabile
 - Alegerea uzuală pentru regiștri
- Intrări de tip "preset" și "clear" sunt utile
 - pornirea / re-setarea sistemului

- Circuite sincrone
 - Intrările, starea și ieșirile se schimbă și sunt luate în considerare la un semnal de referință - **ceas**
 - Exemple: master/slave, circuite active pe front
- Circuite asincrone
 - Intrările, starea și ieșirile se schimbă și sunt luate în considerare independent de vreun semnal de referință
 - Exemplu: latch R-S
- Intrări asincrone în circuite sincrone
 - Intrările se pot schimba la orice moment

Regiștri

- Bitul este "atomul" de informație la nivel fizic; la nivel logic, "atomul" este constituit de un grup de biți
 - 8 (ex.: codul ASCII) sau un multiplu de 8 (virgulă fixă)
- Registru: grup de flip-flop-uri cu semnale de control corelate și funcționând ca un tot
 - Semnalele de ceas, RS (sau echivalente) sunt comune flip-flop-urilor din registru clock
- Exemple
 - Regiștri cu deplasare
 - Contoare

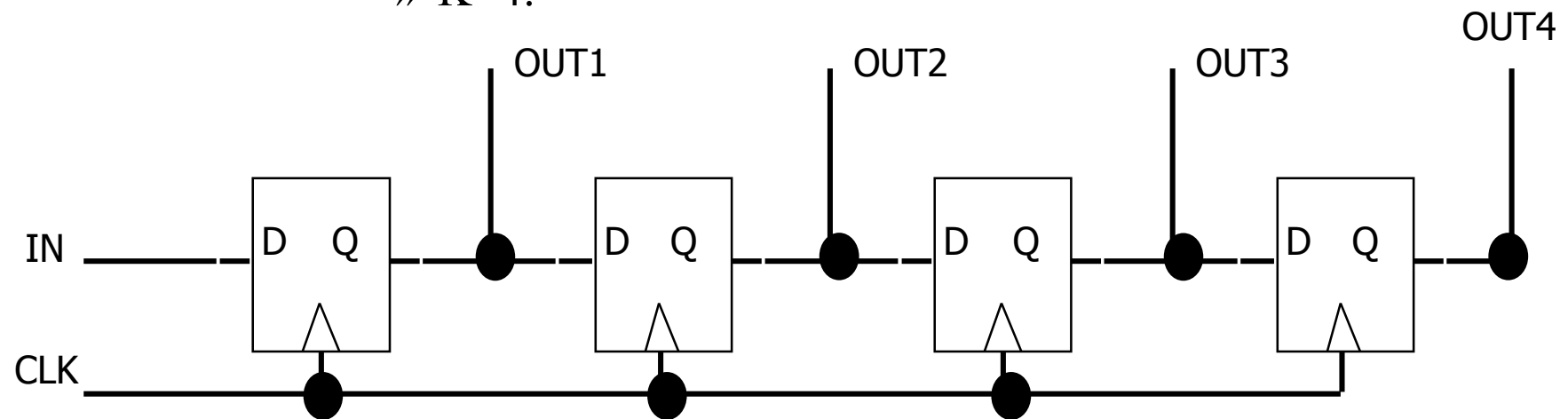
Registrul cu deplasare

- Memorează ultimele k valori input, în ordine

» Specificații: 1101 → 0000;1000;0100;1010;1101

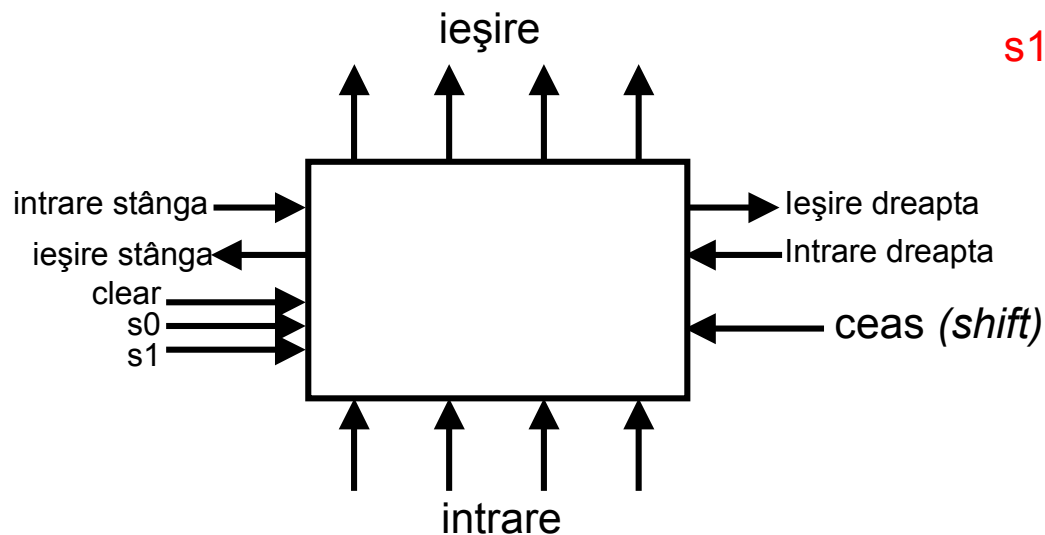
» M-S

» K=4:



Registrul universal cu deplasare

- Intrările pot fi seriale sau paralele (simultane)
- Ieșiri seriale sau paralele
- Poate efectua deplasarea spre stânga sau spre dreapta
- Valori noi sunt introduse de la stânga la dreapta
- Intrări/ieșiri suplimentare pentru *deplasare*
 - biți "pierduți" / "recuperați"



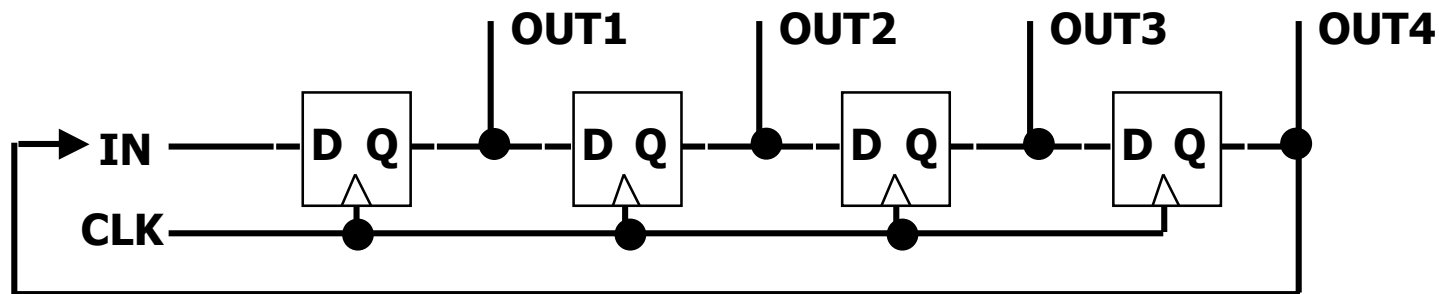
clear pune pe 0 conținutul registrului și ieșirile

s1 și *s0* controlează funcția de deplasare

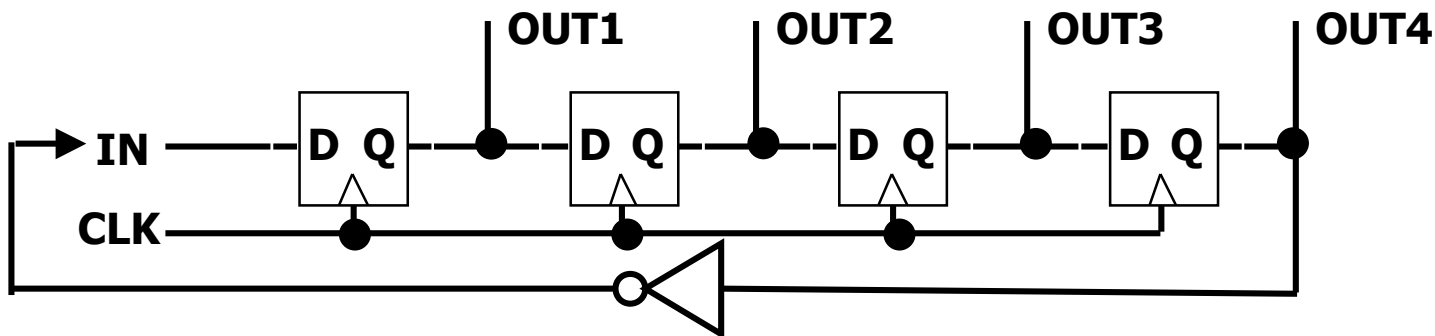
s0	s1	funcție
0	0	păstrează starea
0	1	deplasare dreapta
1	0	deplasare stânga
1	1	încărcare noua intrare

Contoare (circuite de numărare)

- Generează o anumită secvență de combinații la ieșire, reluată de la început după ce se termină.
 - În figură, 1000, 0100, 0010, 0001...
 - Starea inițială trebuie să fie una dintre combinații

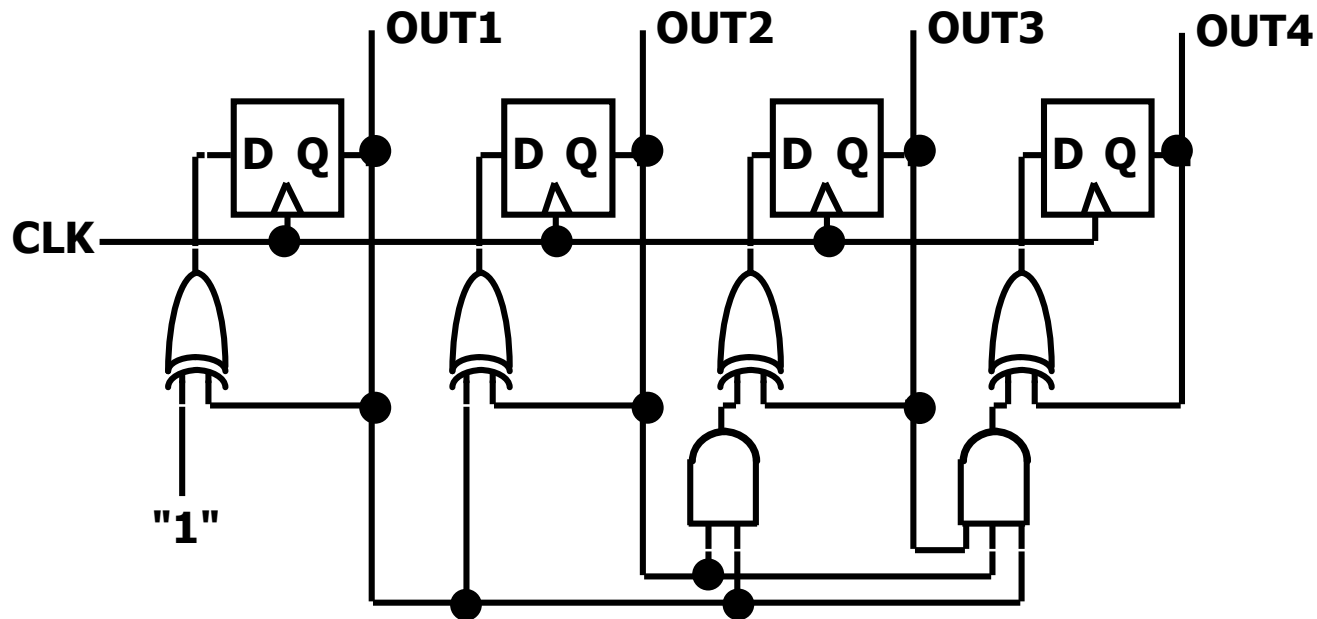


- Contorul Mobius (Johnson)
 - 1000, 1100, 1110, 1111, 0111, 0011, 0001, 0000...



Contor binar

- ieșirea constă din scrierea în baza 2 a numerelor de la 0 la 15, în ordine, reluate apoi de la 0: 0000, 0001, 0010, 0011, 0100, 0101, 0110, 0111, 1000, 1001, 1010, 1011, 1100, 1101, 1110, 1111, 0000, 0001,
 - » 1: 0101010101010101...;
 - » 2: 001100110011...;
 - » 3: 0000111100001111...;
 - » 4: 00000000111111110000000011111111...
- Alte diagrame posibile?



Capitolul al IV-lea

Reprezentări interne

IV.1.
INTRODUCERE

- Reprezentările interne elementare constituie un element al **arhitecturii** oricărui calculator
 - Resursă accesibilă direct programatorilor
- Structurile de date mai complicate se definesc pornind de la reprezentările interne elementare

Reprezentări elementare

- Date numerice
 - Anumite submulțimi finite ale mulțimilor numerelor întregi, respectiv raționale
- Date "alfa-numerice" / logice
 - Caractere; valori de tip boolean
- Instrucțiuni
 - În limbaj mașină
 - Singurele reprezentări interne elementare ne-standardizate
 - Și, evident, neportabile

Importanța studiului reprezentărilor

- Eficiența și siguranța (fiabilitatea) reprezentărilor interne
- Numerice:

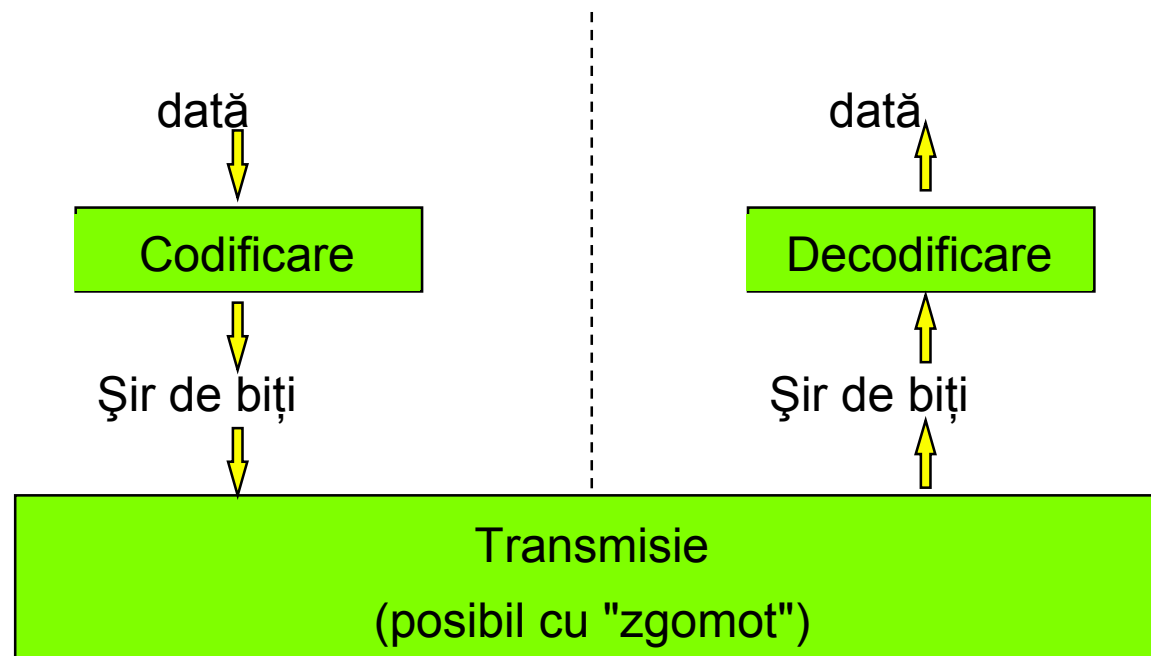
$$r(n_1) \text{ op } r(n_2) \stackrel{?}{=} r(n_1 \text{ op } n_2)$$

- Erori inevitabile și efectul lor
 - Mulțimi de cardinalități diferite
 - » $R \cap [a, b]$ și $Q_m \cap [a, b]$
 - Aproximări, depășiri
 - Tratarea cazurilor de excepție

IV.2. CODURI DETECTOARE ȘI CODURI CORECTOARE DE ERORI

Detectarea de erori

- Fiabilitatea transmisiei și prelucrării reprezentărilor de date



Moduri de detectare/corectare

- Paritate: bit suplimentar
 - *detectie*
 - paritate (im)pară: număr (im)par de 1
- Cod Hamming
 - *corectie*
 - 4 biți de informație, 3 biți de paritate
 - detectarea/corecția mai multor erori simultan

Exemplu: "paritate impară"

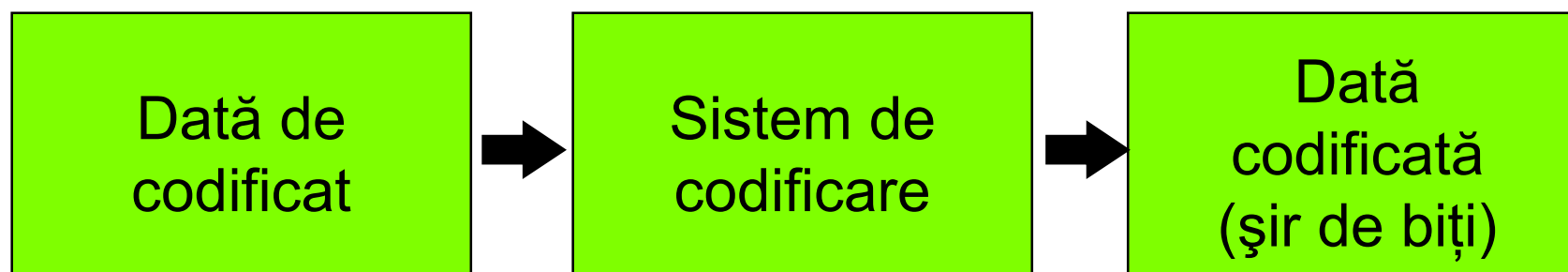
- Emițător:
 - are de trimis valoarea $(110)_2$
 - generează bitul de "paritate impară" $P=1$
 - trimite $(110\underline{1})_2$
- Receptor:
 - primește $(1101)_2$
 - verifică imparitatea numărului de 1 din șir
 - dacă nu detectează erori, elimină bitul de paritate, pentru a obține valoarea transmisă: $(110)_2$

IV.3.

CODIFICĂRI ALFANUMERICE

Codificări alfanumerice

- Reprezentări binare ale datelor alfanumerice
 - Alfabetice, numerice, simbolurile pentru operatori, separatori etc.



Coduri alfanumerice

- ASCII
 - American Standards Committee for Information Interchange code
 - Un caracter se reprezintă pe 7 biți plus un bit de paritate
- EBCDIC (8 biți)
 - Extended Binary Coded Decimal Interchange Code
 - » Extinde codul binar pentru cifre zecimale
- ISO 8859-1 (Latin-1)
 - 8 biți
 - Include și, spre exemplu, litere cu accent
 - $\hat{E} = CA_{(16)}$
- Supraîncărcare a șirurilor de biți
 - Unicode
 - caractere non-latine
 - UCS
 - Universal Character Set

Codul ASCII - exemple

- 1000001 → A
- 1000010 → B
- ...
- 1011010 → Z
- 1100001 → a
- 1100010 → b
- ...
- 1111010 → z
- Ordine lexicografică → comparatorul binar
 - pentru 7 biți – bitul de paritate se ignoră