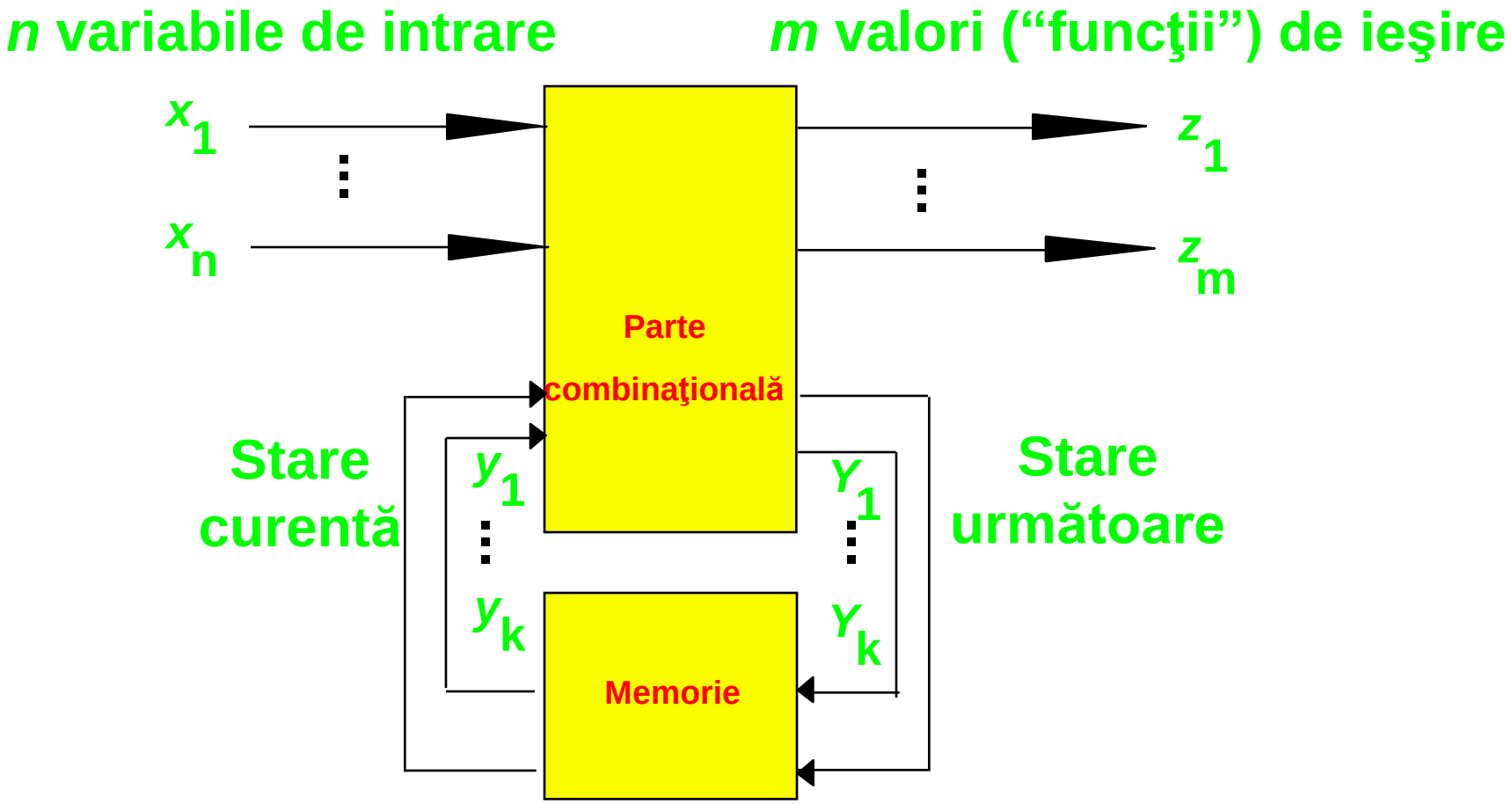
Capitolul al III-lea
Circuite secvenţiale

- "Circuite cu memorie"
- Exemplu: ascensorul
  - Input: număr (natural) indicând etajul-destinaţie
  - Output: număr (întreg) indicând câte etaje trebuiesc urcate/coborâte
    - Nu sunt suficiente toate valorile de intrare pentru a avea descrierea completă a funcţionării circuitului
  - Memorie: suma algebrică a tuturor inputurilor

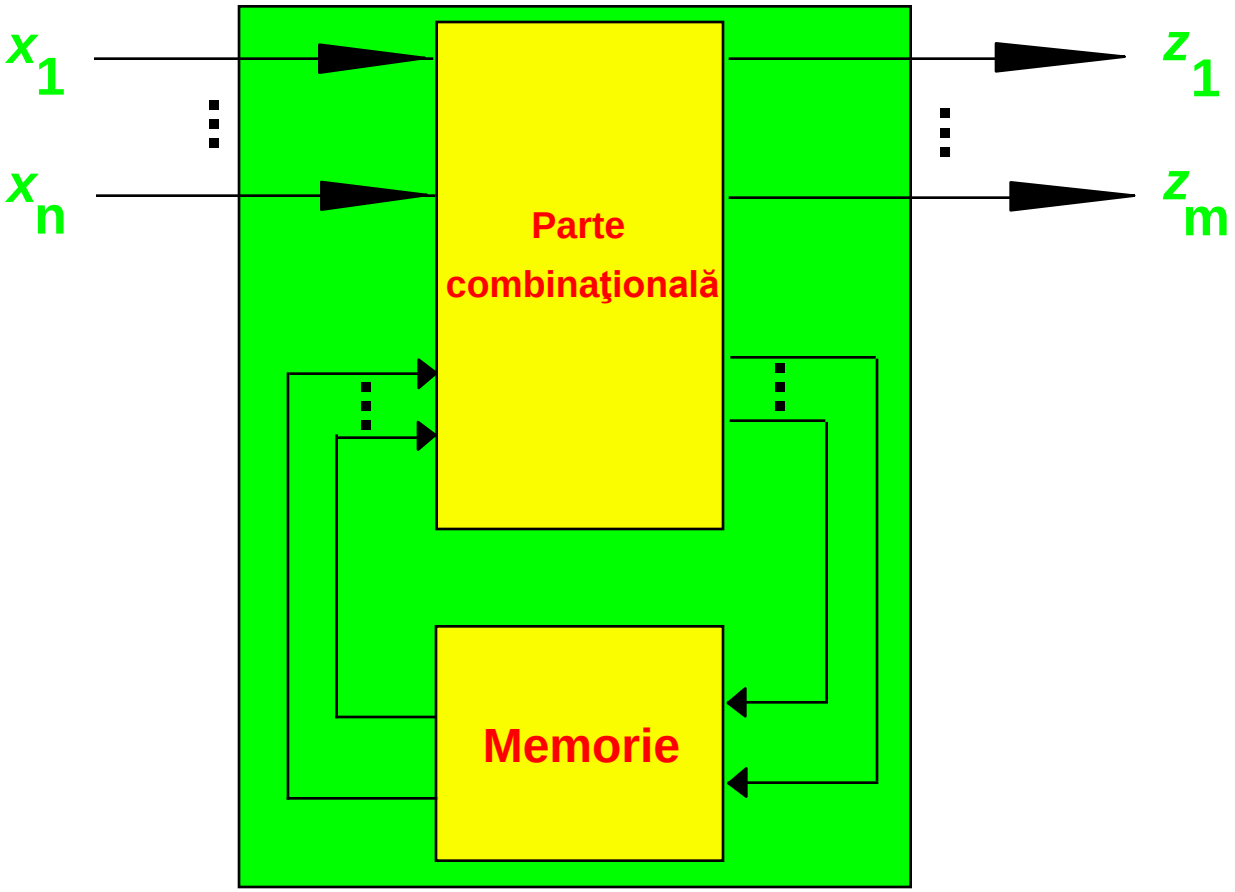
- Outputul depinde de intrare şi de memorie – starea internă
- Ascensor:
  - Starea se schimbă la momente precise, stabilite
    - fie la momentul unui semnal regulat (ceas)
    - fie la momentul în care se întâmplă un eveniment
      - liftul ajunge în dreptul uşii unui nou etaj
- Primul mod se numeşte **sincron**, al doilea – mod **asincron**
- Ascensorul se află la începutul funcţionării într-o stare numită **stare iniţială**
  - etajul de plecare în prima cursă

Circuit secvenţial



Circuit secvenţial – diagrama bloc

Control doar asupra input-urilor independente



***n* variabile de intrare**

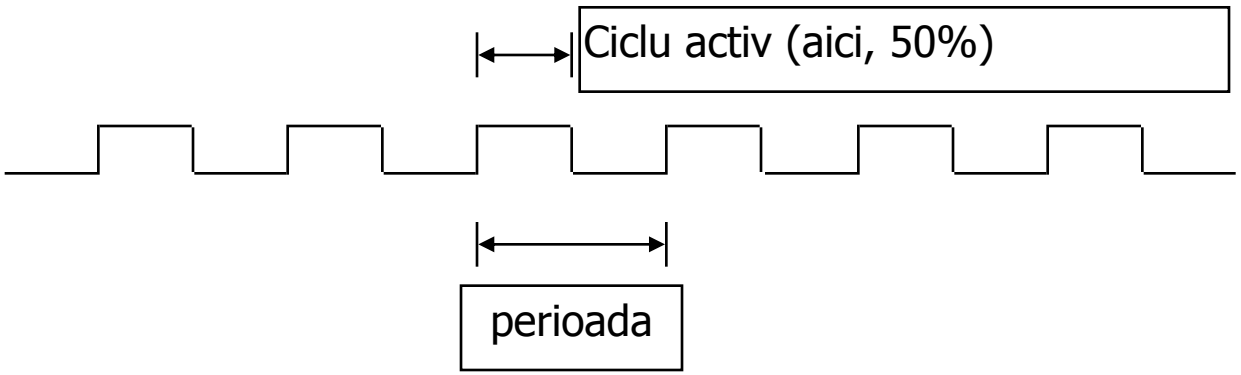
***m* valori (“funcţii”) de ieşire**

Ceas

- Transmiterea semnalului prin porţi şi prin liniile de comunicare se face cu întârzieri:
  - Timp de tranziţie între cele două niveluri
  - Întârziere de propagare input-output
- O soluţie pentru funcţionare corectă este utilizarea unui semnal ceas (Clock), care să **sincronizeze** funcţionarea, oferind şi suficient timp pentru stabilizarea tuturor semnalelor.

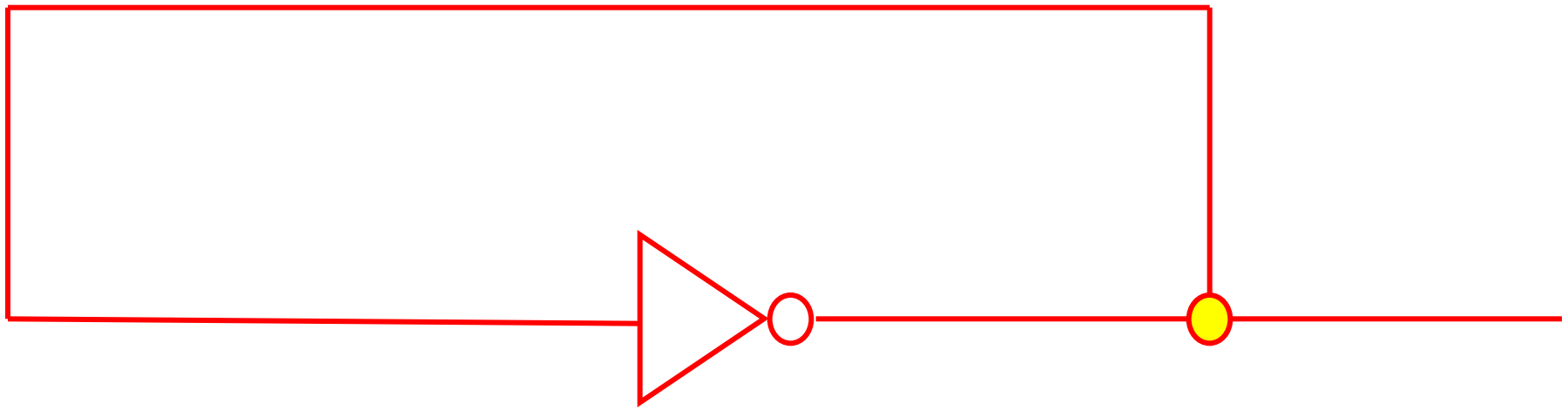
CEAS

- Circuit pentru controlul prin tacte de timp
  - Durata tactului suficientă pentru stabilizarea inputului
  - Permiţând apoi efectul asupra stării memorate
- Semnalul de ceas este periodic
  - Perioada – timpul scurs între două "bătăi" ale ceasului
  - Ciclul activ ("de lucru") – în exemplu, timpul cât semnalul ceas are nivelul "High", exprimat ca procentaj din perioadă
    - » Sunt posibile implementări "activ pe nivel" (maxim, ca în exemplu, sau minim), sau "activ pe front" (crescător sau descrescător)



Diagramă logică

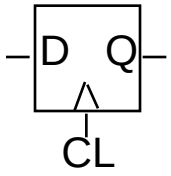
- Cea mai simplă diagramă logică ce produce semnal tip "ceas"
  - Conexiune inversă (feed- back)
  - Starea inițială



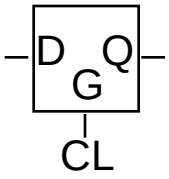
Tipuri de circuite secvenţiale

- După modul de "sesizare" a ceasului
  - Latch-uri - elemente de memorie RS
    - Cu ceas, activ pe nivel; pot fi şi asincrone
  - Flip-flop-uri / bistabili (activ pe front)
    - RS, D, JK, T
- Regiştri, contori
- Modelare abstractă: maşini cu număr finit de stări (automate)

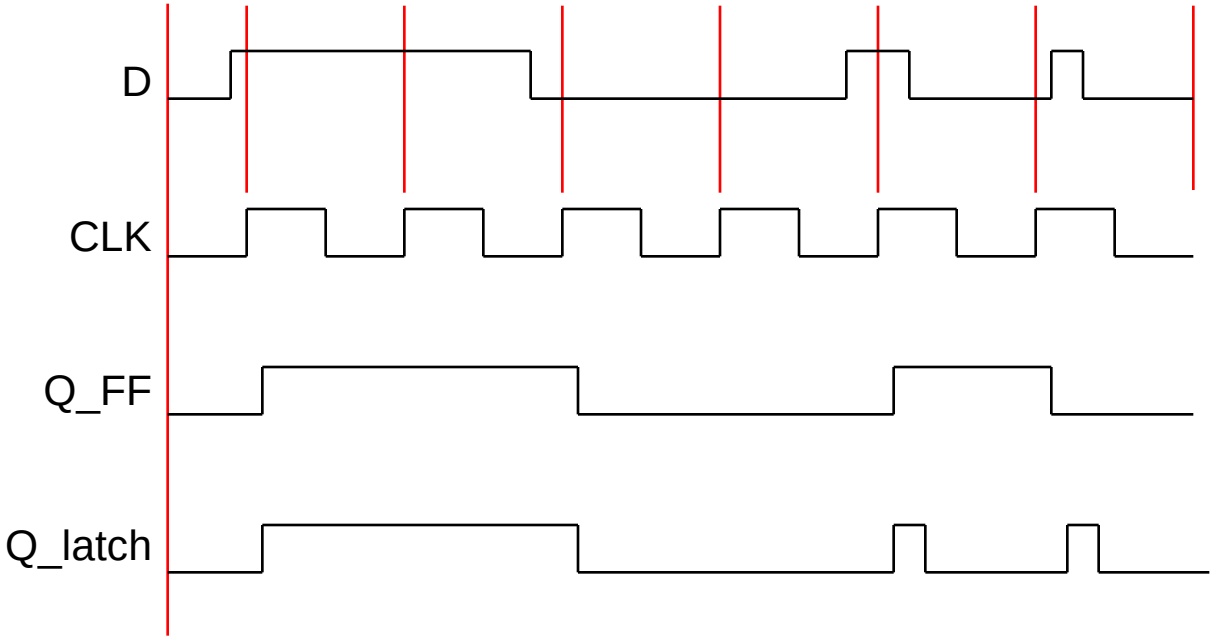
Comparaţie între funcţionarea latch-urilor şi a flip-flop-urilor



Activare
pe front crescător
(flip-flop)



Activare
pe nivel
(latch cu ceas)



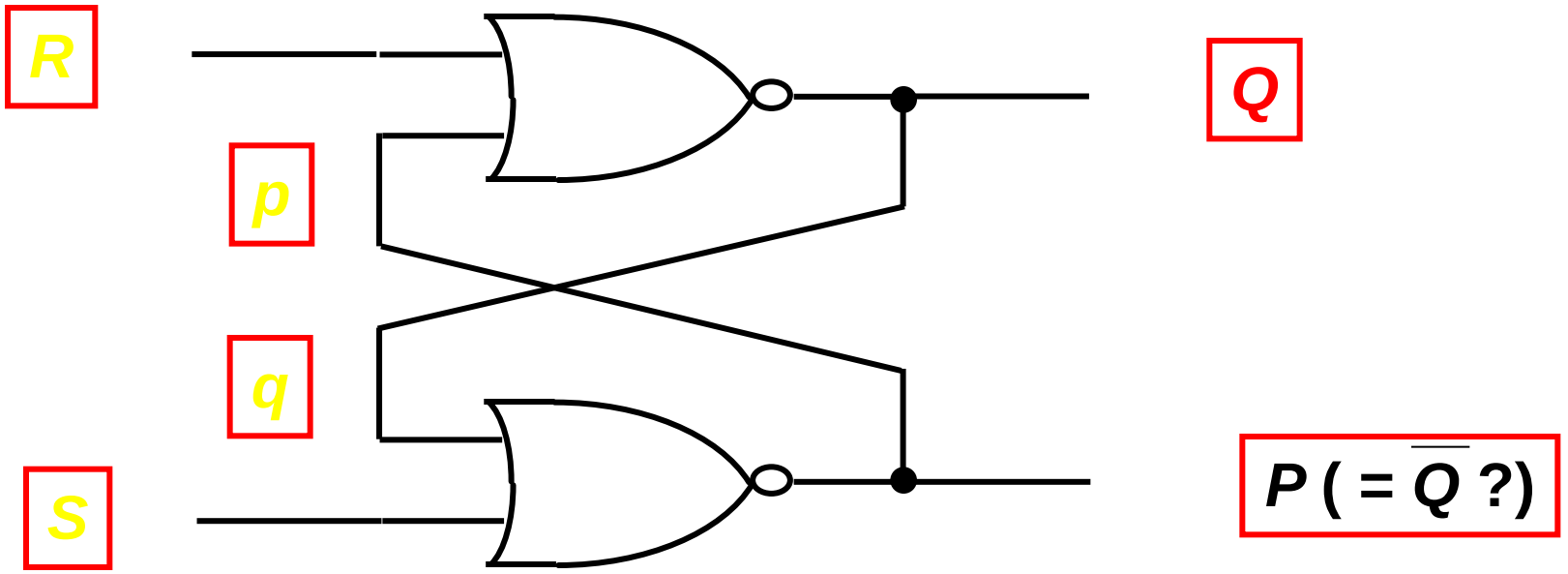
Comportare identică, cu excepţia cazului în care
inputul se schimbă când semnalul ceas este la nivel "High"

- Studiul latch-urilor şi al flip-flop-urilor răspunde la întrebarea "Cum trebuie să arate un circuit care implementează **bitul**?"
- Specificaţiile circuitului "bit":
  - Să se poată scrie în el un 0 sau un 1
  - Să se memoreze acea valoare până la scrierea alteia
  - Să se poată citi ultima valoare scrisă
- Circuitul "bit" nu poate fi circuit combinaţional (condiţia a doua)
- Cum poate arăta circuitul secvenţial "bit"?

Elemente de memorie - Latch-uri

- Latch RS: 2 intrări (RS), două ieşiri (QP), două conexiuni inverse (qp).
  - Se cere să furnizeze simultan şi negaţia ieşirii, prin P.

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

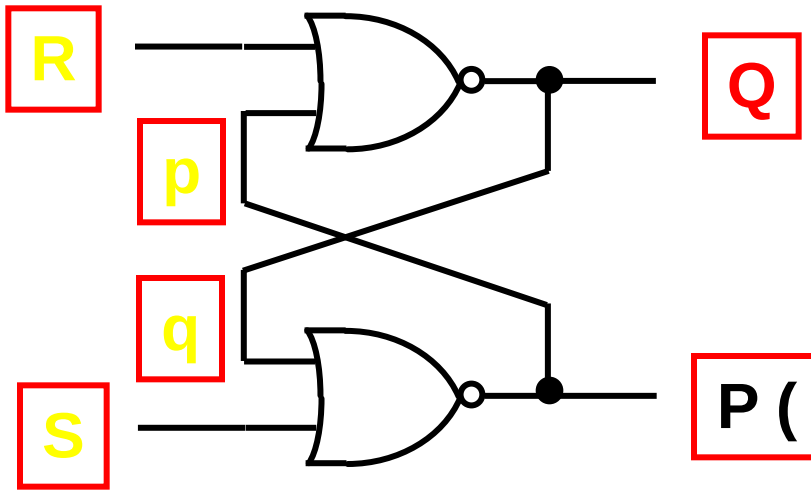


- Cum aflăm funcţionarea circuitului, date valorile lui R şi S?

Reprezentarea funcţionării elementului de memorie RS

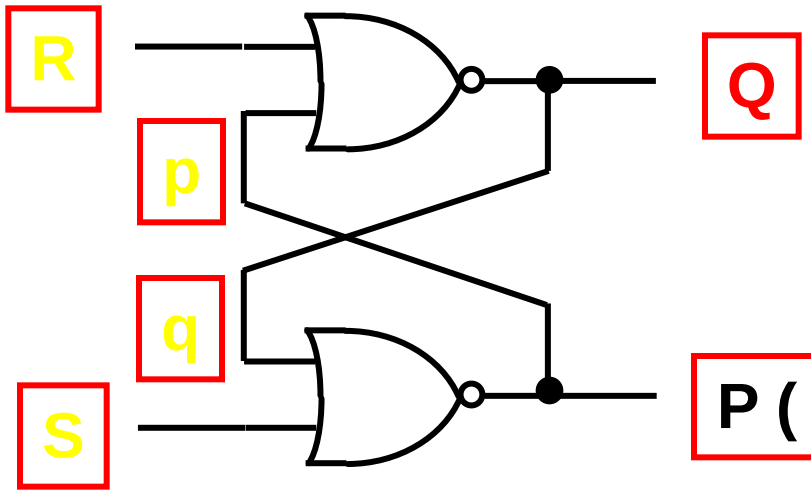
- Se construieşte o diagramă Karnaugh în care se scriu ca valori ale funcţiei perechi de valori QP.
- Prin definiţie, p=P şi q=Q.
  - În stânga egalului, valorile "noi".

qp \ RS	00	01	11	10
00				
01				
11				
10				



Completarea diagramei Karnaugh: **Q**

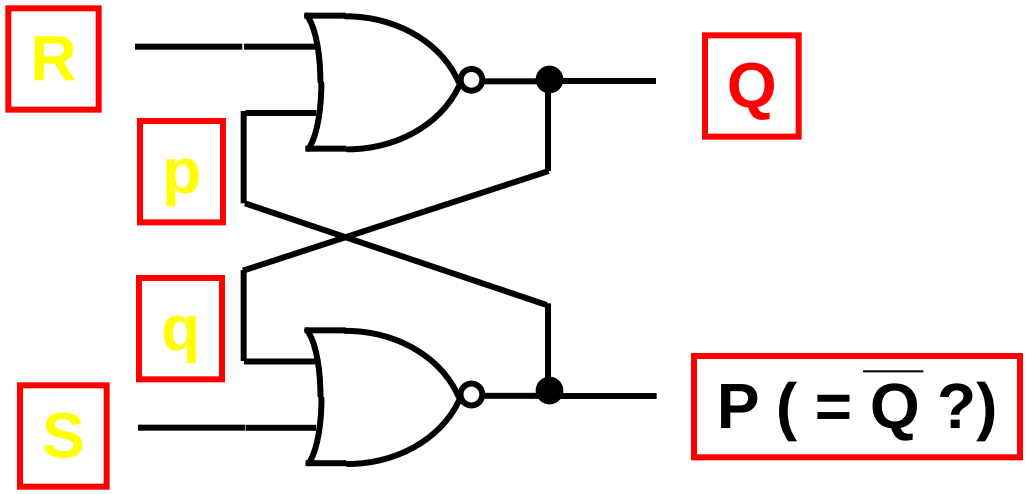
- Ştim că **$Q = \overline{R} \cdot \overline{p}$** ...
- Blocul (de 4) pentru expresia lui **Q**



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: $\overline{Q}$

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



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

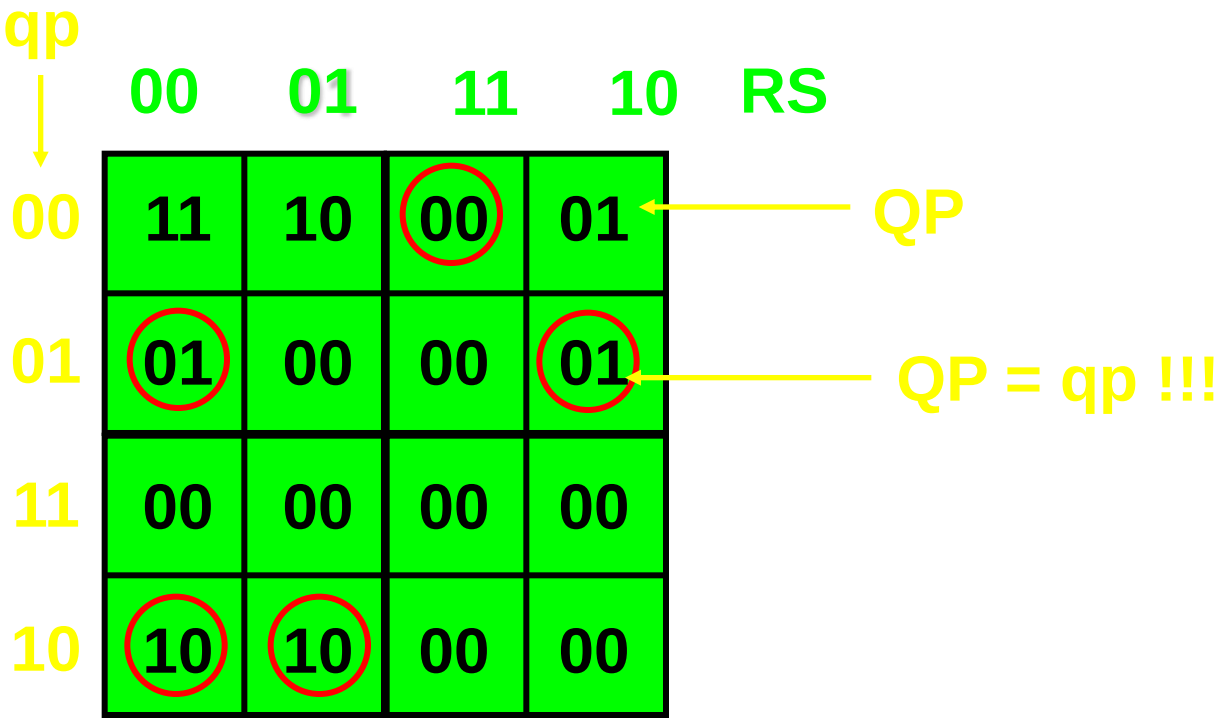
Stări stabile

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

qp \ RS	00	01	11	10
00	11	10	00	01
01	01	00	00	01 ← QP
11	00	00	00	00
10	10	10	00	00

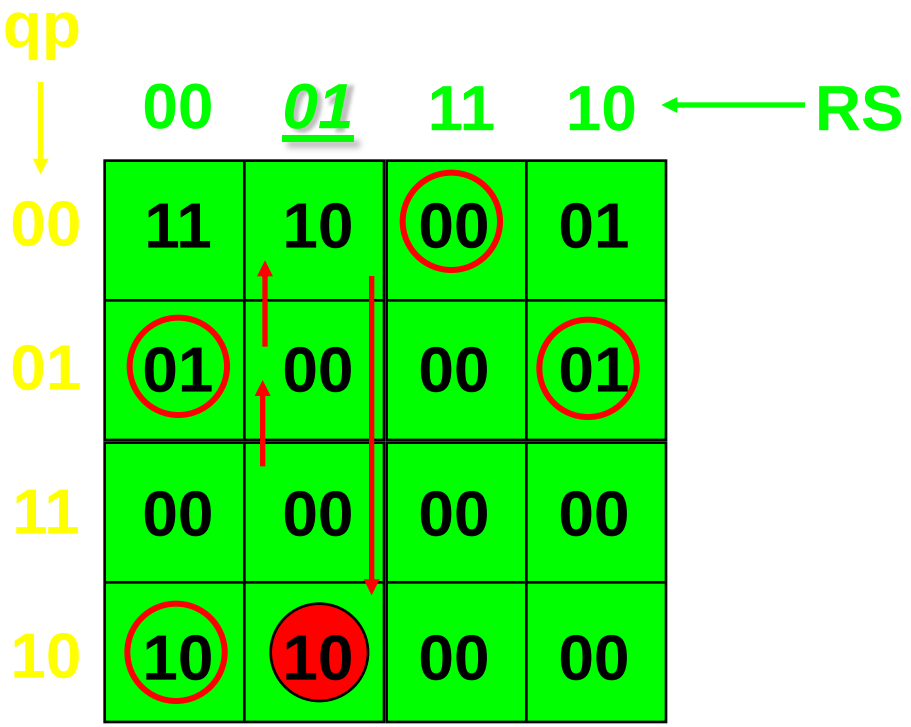
Identificarea stărilor stabile

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



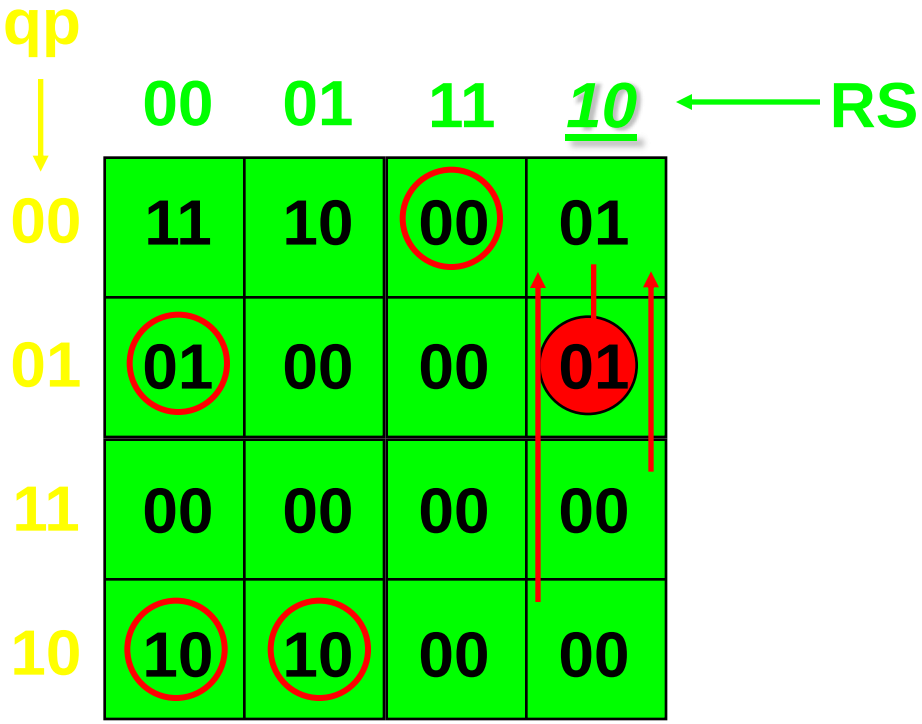
Funcţionarea pentru R = 0 şi S = 1

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



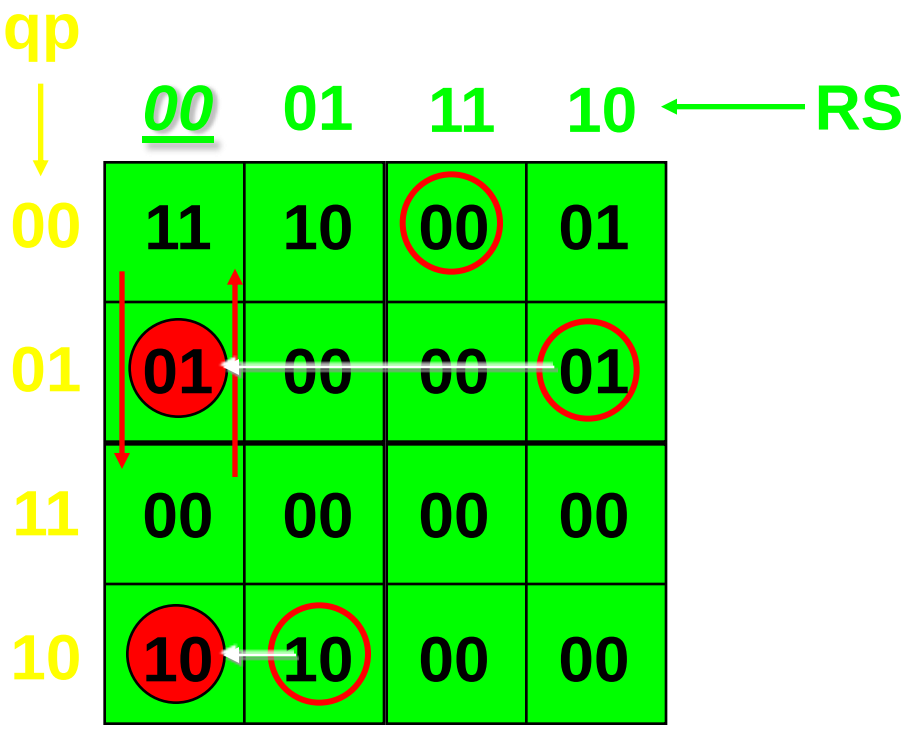
Funcţionarea pentru R = 1 şi S = 0

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



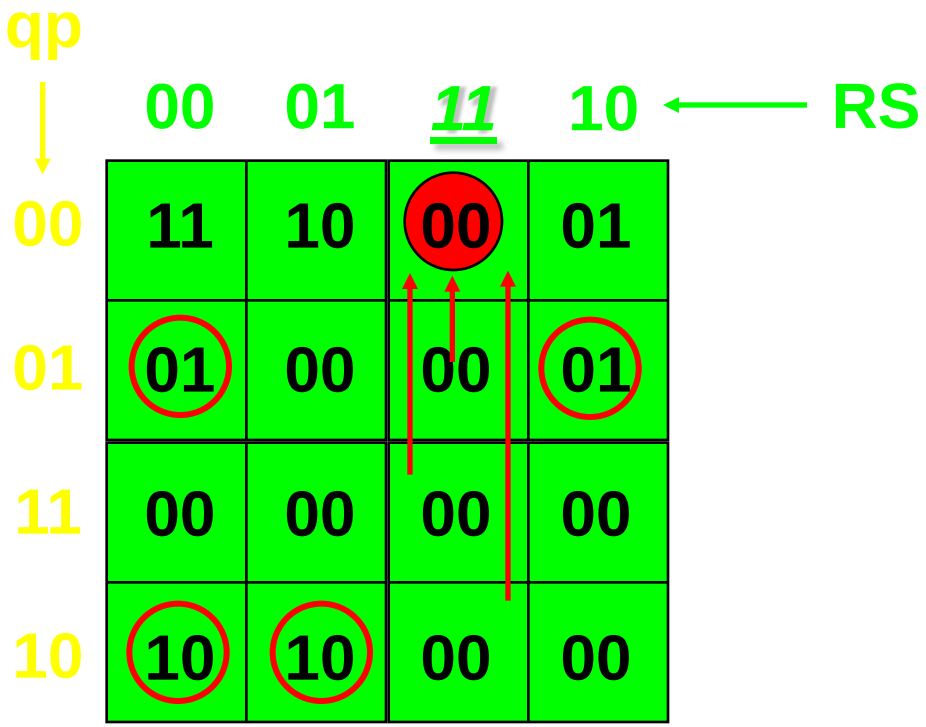
Funcţionarea pentru R = 0 şi S = 0

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



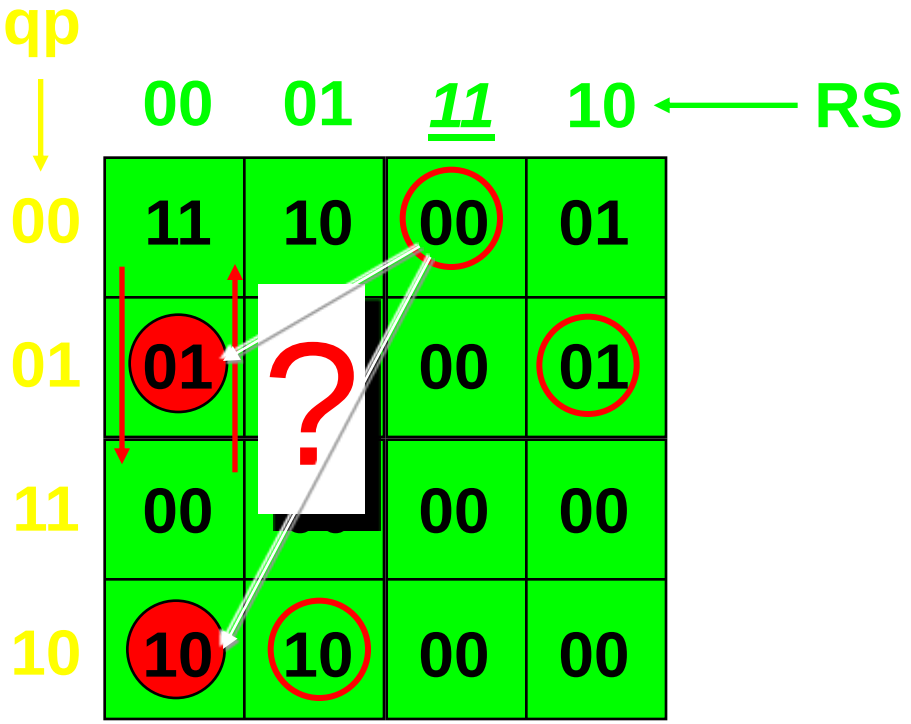
Funcţionarea pentru R = 1 şi S = 1

- Contradicţie
  - Ar trebui să avem simultan $Q=\overline{Q}=0$



Funcţionarea pentru R = 1 şi S = 1

- Dacă RS = 11 şi apoi RS=00, rezultatul este nedefinit (săgeţile roşii verticale)



Funcţionarea circuitului latch RS

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

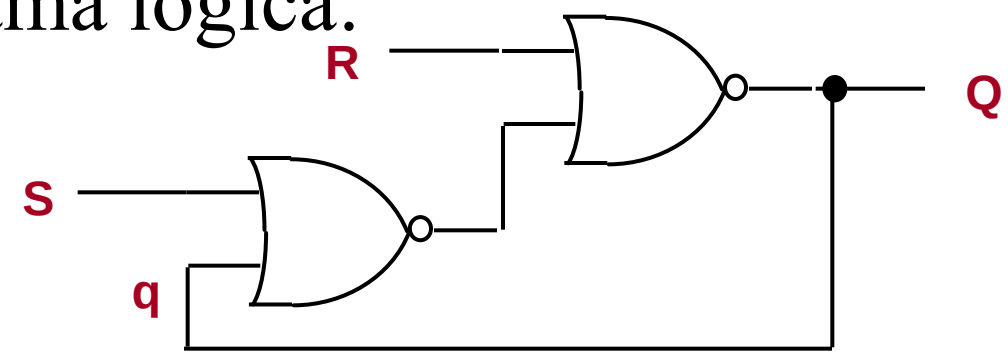
S	R	Q	P	
1	0	**1**	**0**	Setare (la "1")
0	1	**0**	**1**	Resetare (la "0")
0	0	**1**	**0**	Stare neschimbată (după SR = 10)
0	0	**0**	**1**	Stare neschimbată (după SR = 01)
1	1	*	*	Combinaţie imposibilă

Funcţionarea latch-ului RS

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

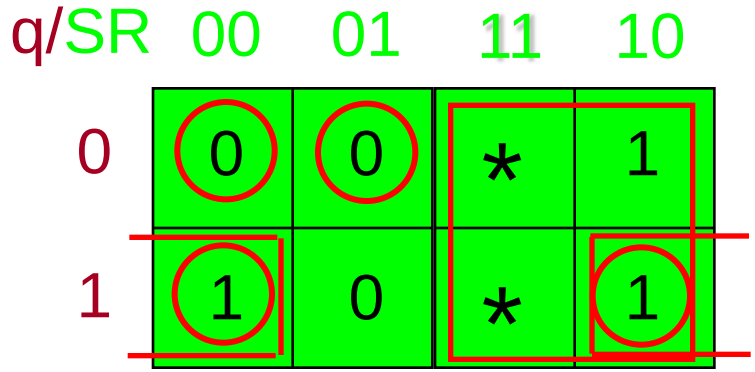
q/SR	00	01	11	10
0	0	0	0	1
1	1	0	0	1

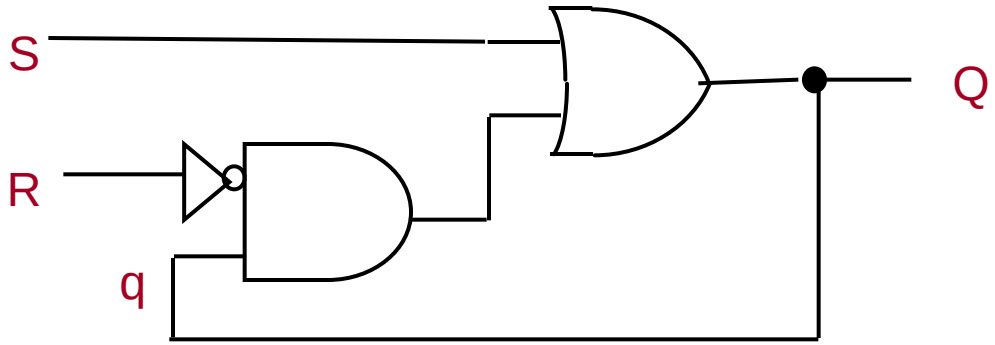
- Diagrama logică:



Altă reducere

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



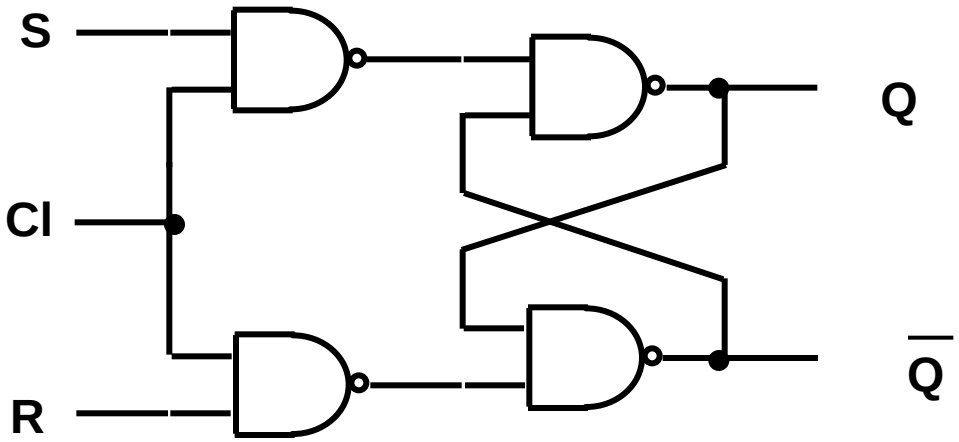


Circuite secvenţiale sincrone

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

Bistabil RS (cu ceas)

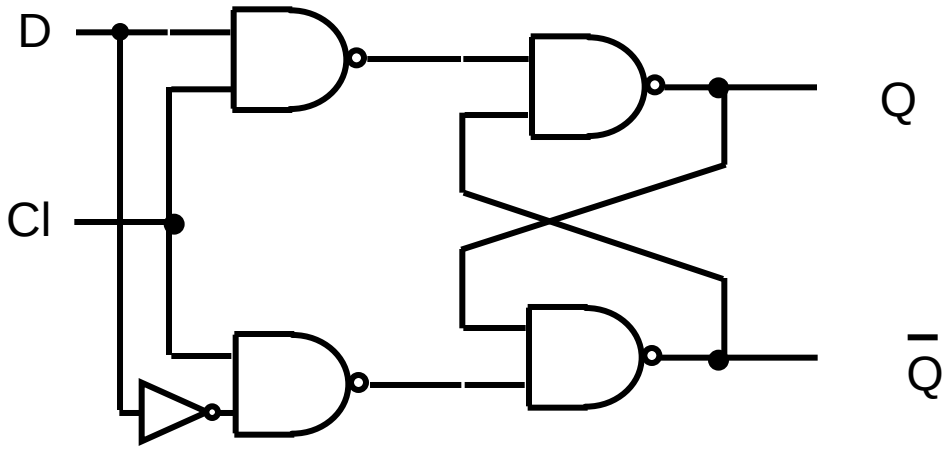
- Porţi NAND
- Ceas→momente de timp (…t, t+1…)



$q=Q_t$	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 şi tabelul de adevăr.
- Derivat din RS. Modelează situaţiile R≠S.

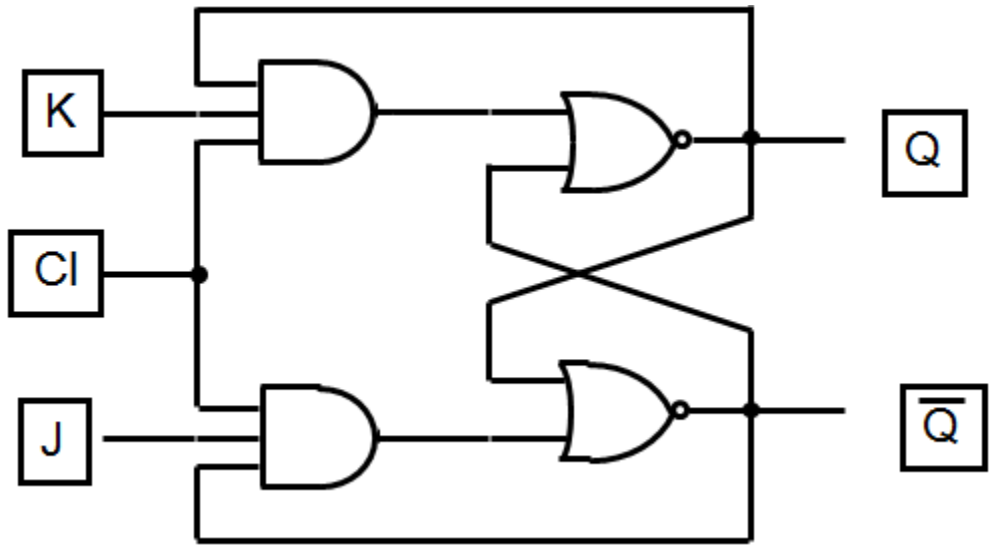


$q=Q_t$	D	Q_{t+1}
0	0	0
0	1	1
1	0	0
1	1	1

- Elimină combinaţiile interzise.

Bistabil JK

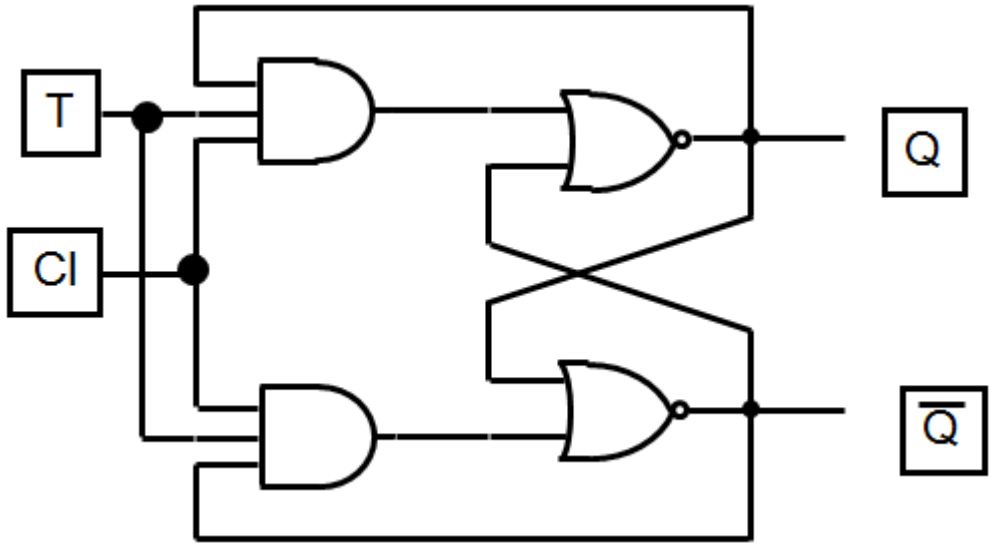
- Diagrama logică şi tabelul de adevăr
  - Elimină combinaţia imposibilă de la RS – feedback-uri suplimentare (J←S, K←R)



$q=Q_t$	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 situaţiile 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 evoluţia lui $Q_{(t+1)}$ în funcţie de intrări (independente sau nu).

•Singura combinaţie 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)

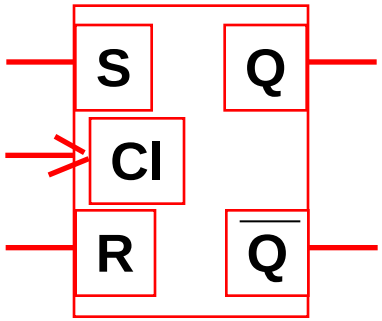
J	K	$Q_{(t+1)}$	
0	0	$Q_{(t)}$	(neschimbată)
0	1	0	(memorare 0)
1	0	1	(memorare 1)
1	1	$\overline{Q_{(t)}}$	(negare)

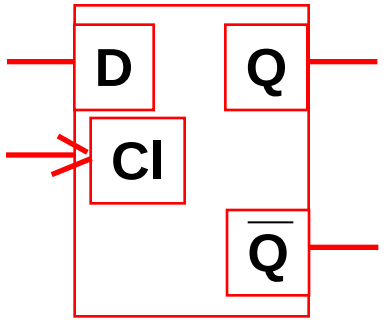
D	$Q_{(t+1)}$	
0	0	(memorare 0)
1	1	(memorare 1)

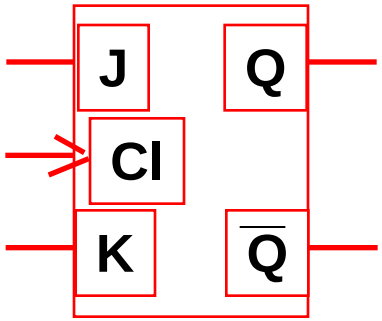
T	$Q_{(t+1)}$	
0	$Q_{(t)}$	(neschimbată)
1	$\overline{Q_{(t)}}$	(negare)

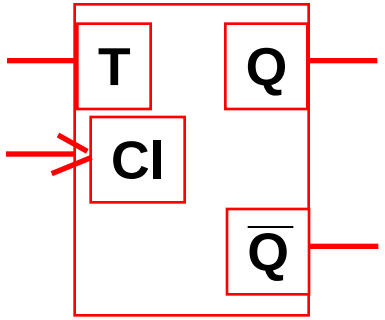
Diagrame bloc pentru bistabili

- Se utilizează simbolii:



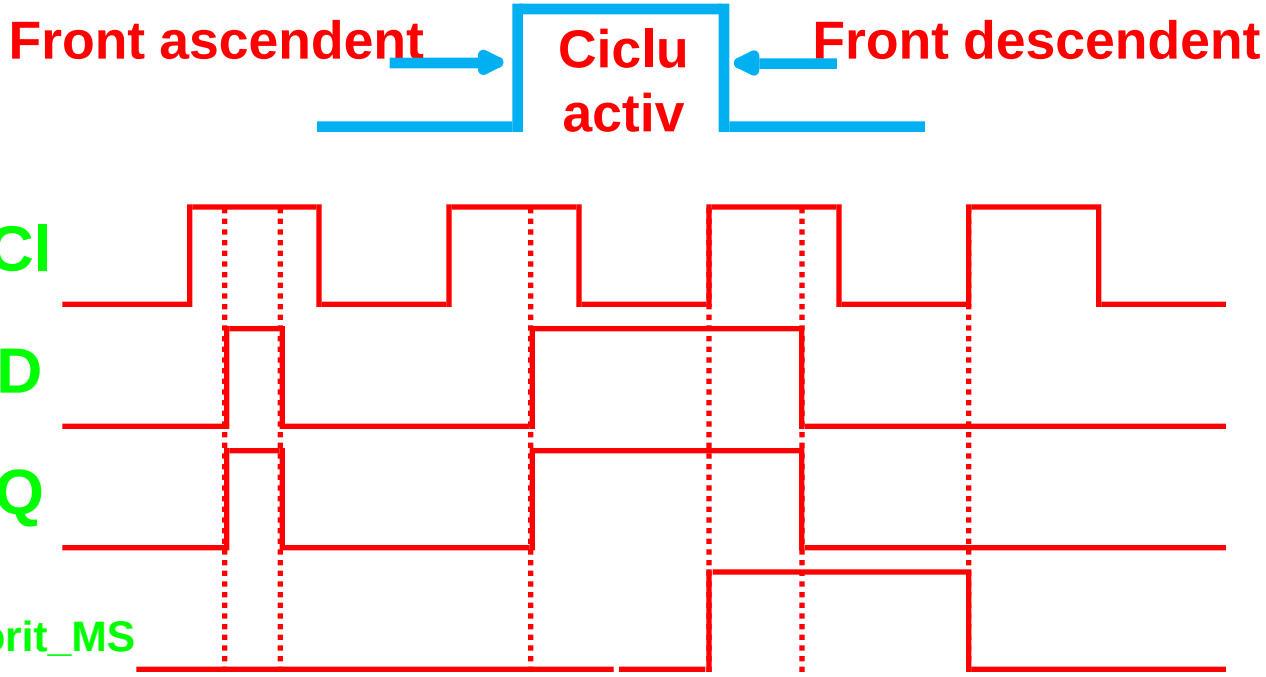






Probleme de funcţionare

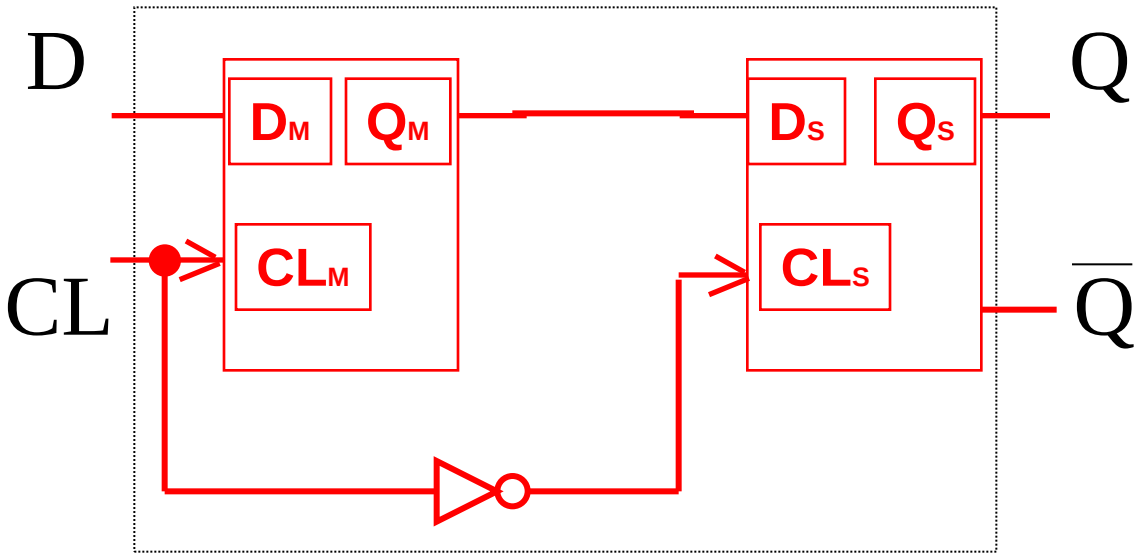
- Pentru a obţine un flip-flop D
  - » din latch-uri D cu ceas
  - » ignorare; întârziere (front ascendent)



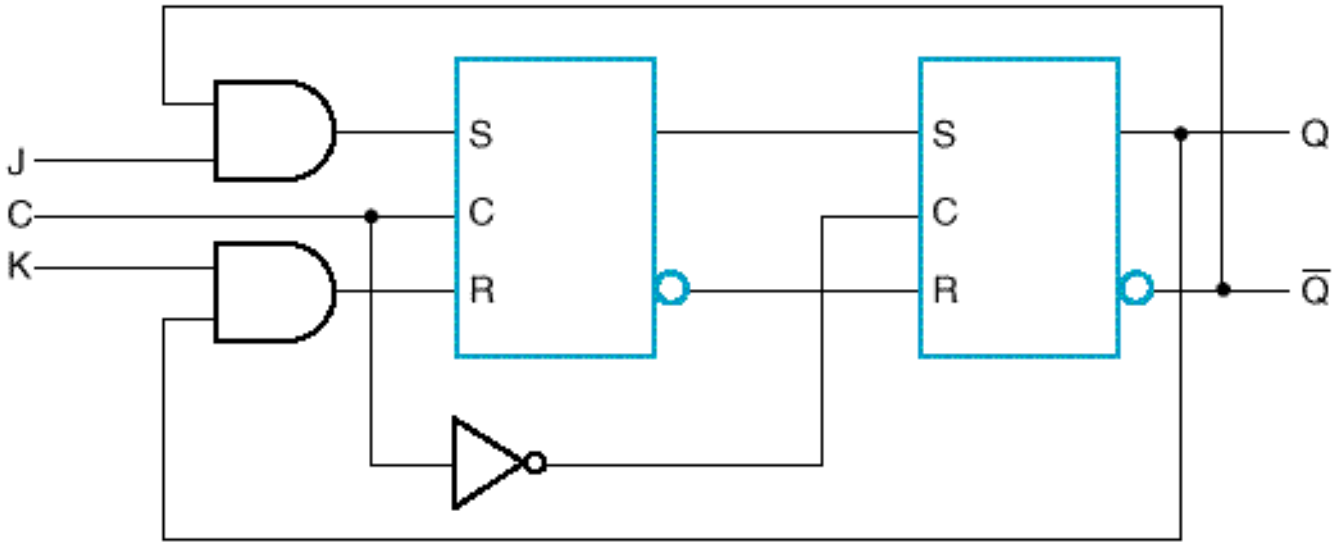
Flip-flop master-slave D

Reduce probabilitatea activării la variaţii bruşte de input.

Face posibilă proiectarea unor circuite secvenţiale complexe (ex.: regiştrii cu deplasare).



Flip-flop JK master-slave



(a)

J	K	Next State of Q
0	0	Q
0	1	0
1	0	1
1	1	$\overline{Q}$

Utilizarea controlului prin ceas

- Depinde de elementele de construcţie 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 şi flip-flop-uri

- Circuite D
  - Latch: 4 tranzistoare
  - Flip-flop: utilizat în elemente programabile
  - Alegerea uzuală pentru regiştri

- Intrări de tip "preset" şi "clear" sunt utile
  - pornirea / re-setarea sistemului

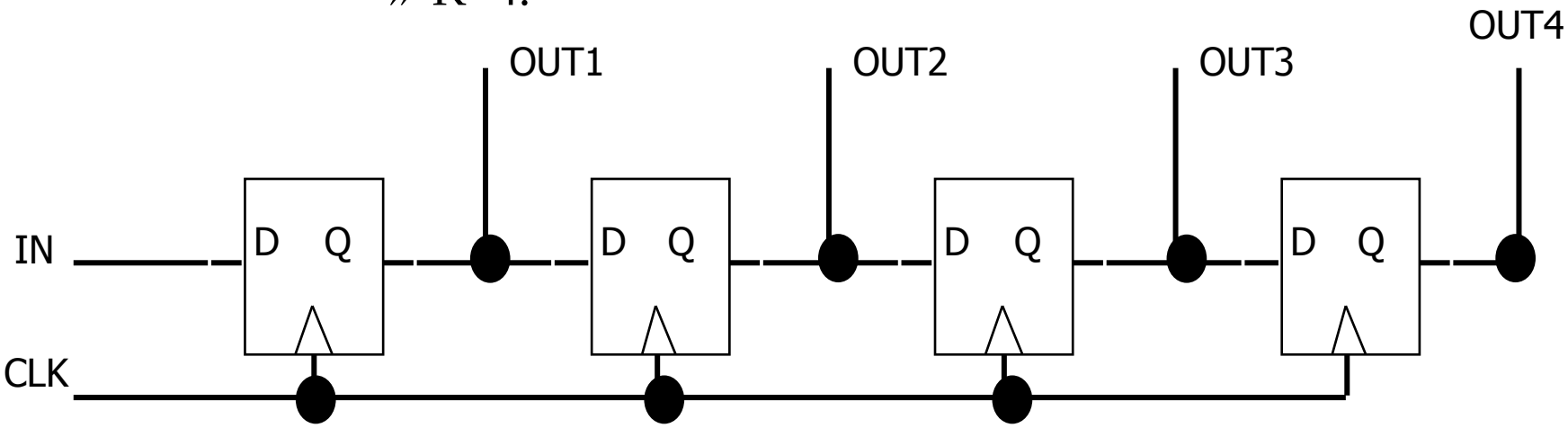
- Circuite sincrone
  - Intrările, starea şi ieşirile se schimbă şi sunt luate în considerare la un semnal de referinţă - **ceas**
  - Exemple: master/slave, circuite active pe front
- Circuite asincrone
  - Intrările, starea şi ieşirile se schimbă şi sunt luate în considerare independent de vreun semnal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

Regiştri

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

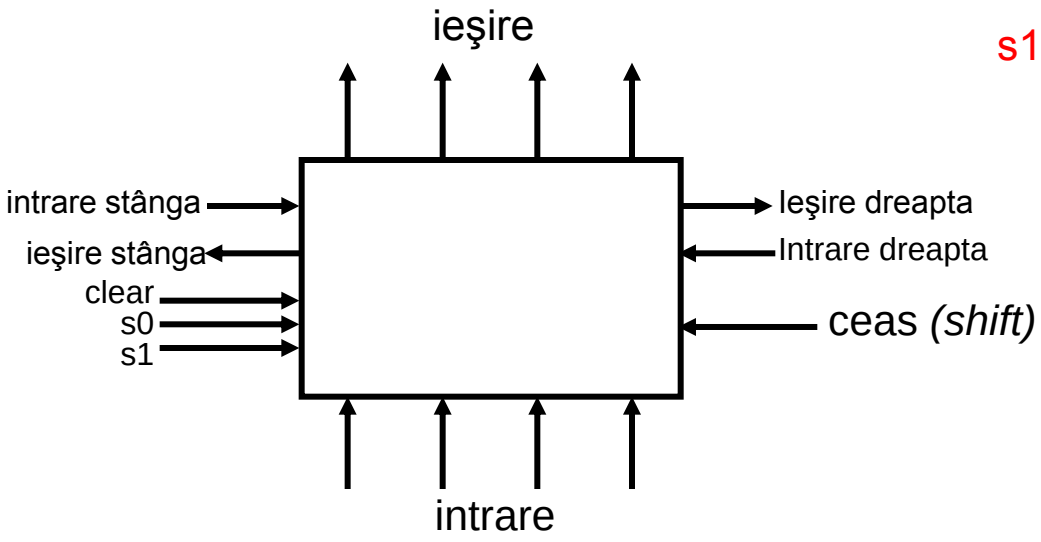
Registrul cu deplasare

- Memorează ultimele k valori input, în ordine
  - » Specificaţii: 1101→0000;1000;0100;1010;1101
  - » M-S
  - » K=4:



Registrul universal cu deplasare

- Intrările pot fi seriale sau paralele (simultane)
- Ieşiri seriale sau paralele
- Poate efectua deplasarea spre stânga sau spre dreapta
- Valori noi sunt introduse de la stânga la dreapta
- Intrări/ieşiri suplimentare pentru *deplasare*
  - biţi "pierduţi" / "recuperaţi"



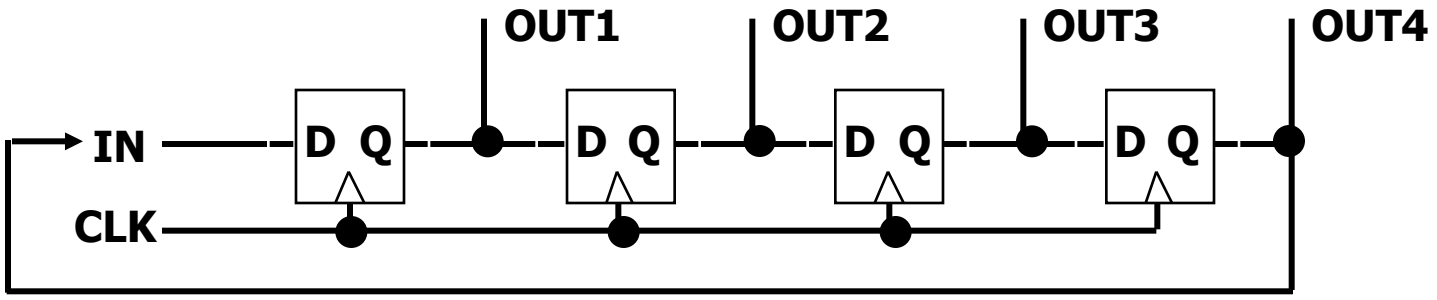
clear pune pe 0 conţinutul registrului şi ieşirile

s1 şi s0 controlează funcţia de deplasare

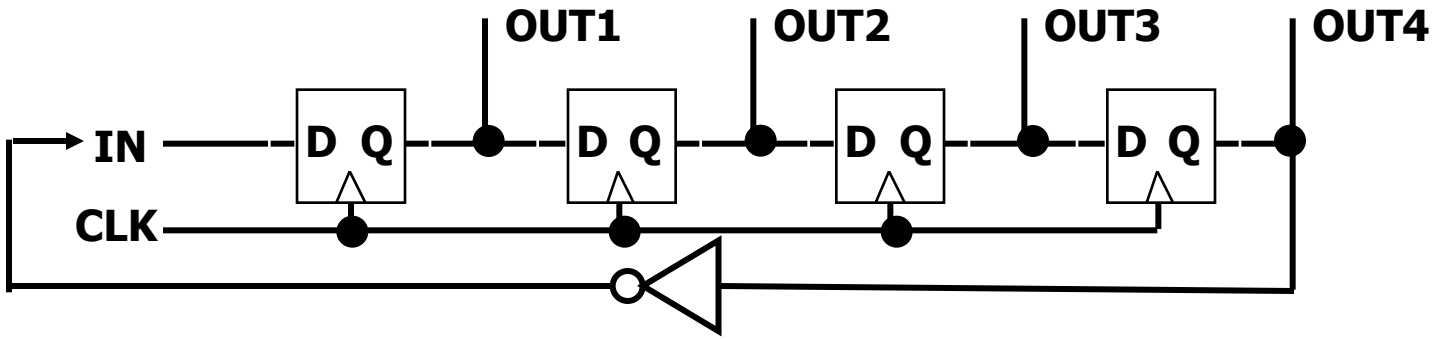
s0	s1	funcţie
0	0	păstrează starea
0	1	deplasare dreapta
1	0	deplasare stânga
1	1	încărcare noua intrare

Contoare (circuite de numărare)

- Generează o anumită secvenţă de combinaţii la ieşire, reluată de la început după ce se termină.
  - În figură, 1000, 0100, 0010, 0001…
  - Starea iniţială trebuie să fie una dintre combinaţii



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



Contor binar

- Ieşirea 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?