ARHITECTURA SISTEMELOR DE CALCUL - CURS 0x04

CIRCUITE COMBINAȚIONALE ȘI SECVENȚIALE

Cristian Rusu

DATA TRECUTĂ

- abstractizarea digitală
- circuite digitale
- circuite combinaționale
- am văzut un singur exemplu de circuit pe bază de tranzistor
- discuție binary vs hex

CUPRINS

- simplificări logice
- circuit de adunare
- exemple de circuite combinaționale și secvențiale
- referințe bibliografice

CIRCUIT DIGITAL COMBINAȚIONAL

- **tabel de adevăr**

A	B	C	X
0	0	0	0
1	0	0	1
0	1	0	0
1	1	0	1
0	0	1	0
1	0	1	0
0	1	1	1
1	1	1	1

- X = ?

CIRCUIT DIGITAL COMBINAȚIONAL

- **tabel de adevăr**

A	B	C	X
0	0	0	0
1	0	0	1
0	1	0	0
1	1	0	1
0	0	1	0
1	0	1	0
0	1	1	1
1	1	1	1

- $X = A\overline{B}\overline{C} + AB\overline{C} + \overline{A}BC + ABC$

  =

CIRCUIT DIGITAL COMBINAȚIONAL

- **tabel de adevăr**

A	B	C	X
0	0	0	0
1	0	0	1
0	1	0	0
1	1	0	1
0	0	1	0
1	0	1	0
0	1	1	1
1	1	1	1

- $X = A\overline{B}\overline{C} + AB\overline{C} + \overline{A}BC + ABC$

  $= A\overline{C}\,(\overline{B} + B) + (\overline{A} + A)BC$

  $=$

CIRCUIT DIGITAL COMBINAȚIONAL

- **tabel de adevăr**

A	B	C	X
0	0	0	0
1	0	0	1
0	1	0	0
1	1	0	1
0	0	1	0
1	0	1	0
0	1	1	1
1	1	1	1

- $X = A\overline{B}\overline{C} + AB\overline{C} + \overline{A}BC + ABC$

  $= A\overline{C}\,(\overline{B} + B) + (\overline{A} + A)BC$

  $= A\overline{C} + BC$

CIRCUIT DIGITAL COMBINAȚIONAL

- **tabel de adevăr alternativ: reducerea cu wildcard**

A	B	C	X
0	0	0	0
1	0	0	1
0	1	0	0
1	1	0	1
0	0	1	0
1	0	1	0
0	1	1	1
1	1	1	1

A	B	C	X
0	?	0	0
1	?	0	1
?	0	1	0
?	1	1	1

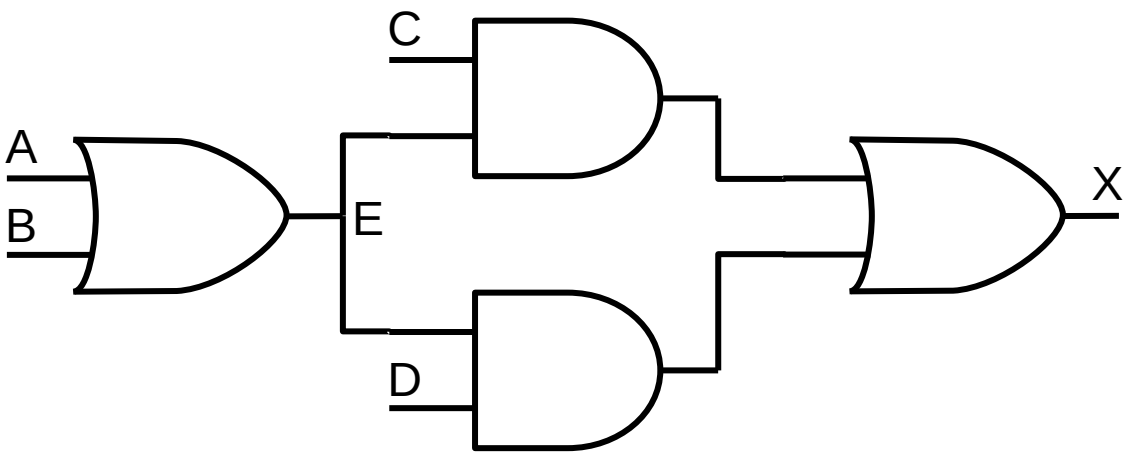
- $X = A\bar{C} + BC$

CIRCUIT DIGITAL COMBINAȚIONAL

- **considerăm acum**
  - X = AC + BC + AD + BD
    =

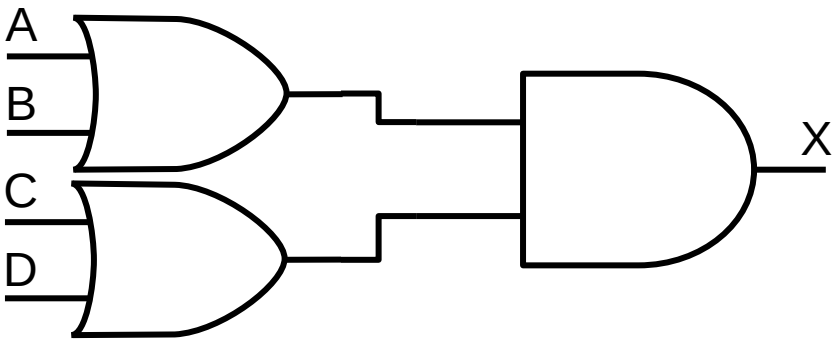
CIRCUIT DIGITAL COMBINAȚIONAL

- **considerăm acum**
  - X = AC + BC + AD + BD
    = (A + B)C + (A + B)D
    = EC + ED

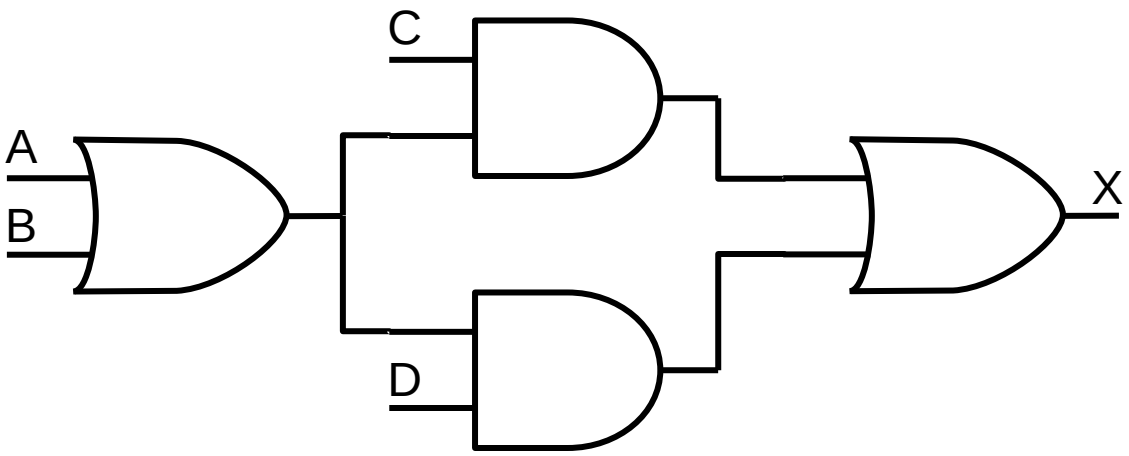


CIRCUIT DIGITAL COMBINAȚIONAL

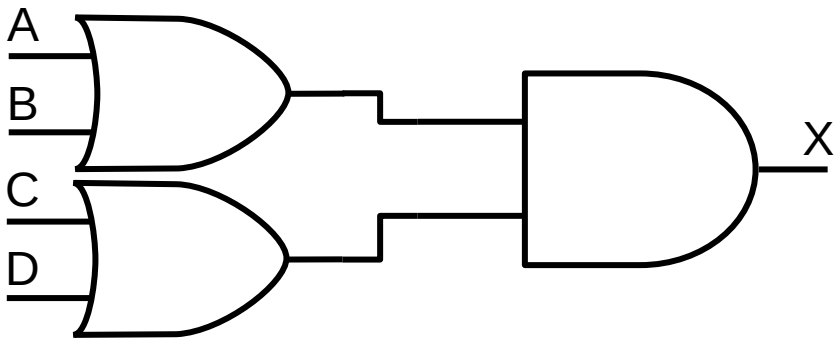
- **considerăm acum**
  - X = AC + BC + AD + BD
    = (A + B)C + (A + B)D
    = (A + B)(C + D)



CIRCUIT DIGITAL COMBINAȚIONAL



- **versus**



în general, e bine să mergem până la capătul calculelor de grupare
dar, rețineți că ne interesează și timpul de propagare

număr porți vs. stagii de calcul vs. tipul de porți ...

CIRCUIT DIGITAL COMBINAȚIONAL

- **nu toate circuitele sunt la fel de eficiente**

poartă	întârziere (ps)	suprafață (μm^2)
AND-2	50	25
NAND-2	30	15
OR-2	55	26
NOR-2	35	16
AND-4	90	40
NAND-4	70	30
OR-4	100	42
NOR-4	80	32

- **ce observați?**

CIRCUIT DIGITAL COMBINAȚIONAL

- **nu toate circuitele sunt la fel de eficiente**

poartă	întârziere (ps)	suprafață (μm^2)
AND-2	50	25
NAND-2	30	15
OR-2	55	26
NOR-2	35	16
AND-4	90	40
NAND-4	70	30
OR-4	100	42
NOR-4	80	32

- **ce observați?**
  - N*-? sunt mai rapide și mai mici ca suprafață decât *-?
  - *-2 sunt mai rapide și mai mici ca suprafață decât *-4

motivul: unele tehnologii sunt bazate pe tranzistoare CMOS, iar circuitele analogice sunt mai eficiente pe logică negată (mai puține componente electrice)

CIRCUITE MAI COMPLEXE

- **concluzia importantă:**
- **tot ce facem pe un sistem de calcul trebuie să se reducă la circuite care sunt porți logice**
- **altceva nu există la acest nivel**

.

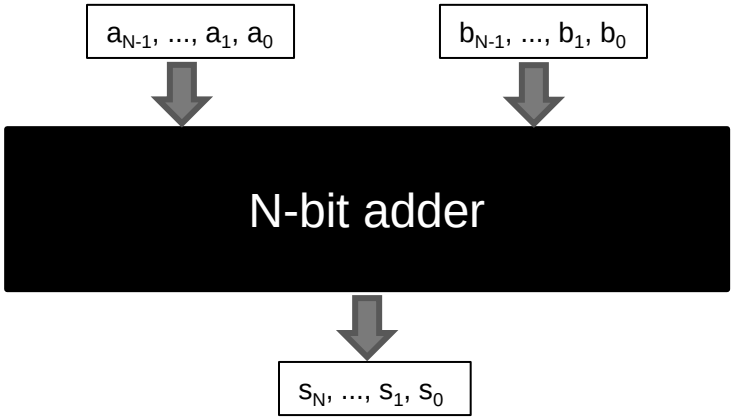
CIRCUITE MAI COMPLEXE

- **circuit de adunare**
  - se dau *a* și *b* pe *N* biți
  - vrem să calculăm *s* = *a* + *b*
    - pe câți biți este *s*?
    -

.

CIRCUITE MAI COMPLEXE

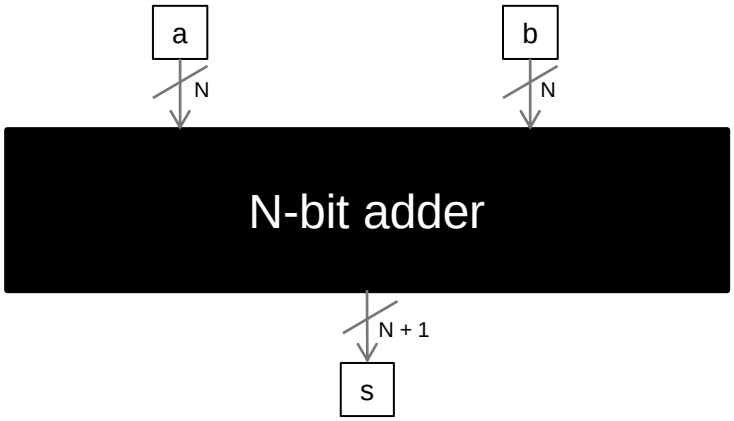
- **circuit de adunare**
  - se dau *a* și *b* pe *N* biți
  - vrem să calculăm *s* = *a* + *b*
    - pe câți biți este *s*?
    - *N+1* biți



    - câte intrări avem?
    - câte ieșiri?
    - cât de mare va fi circuitul combinațional?

CIRCUITE MAI COMPLEXE

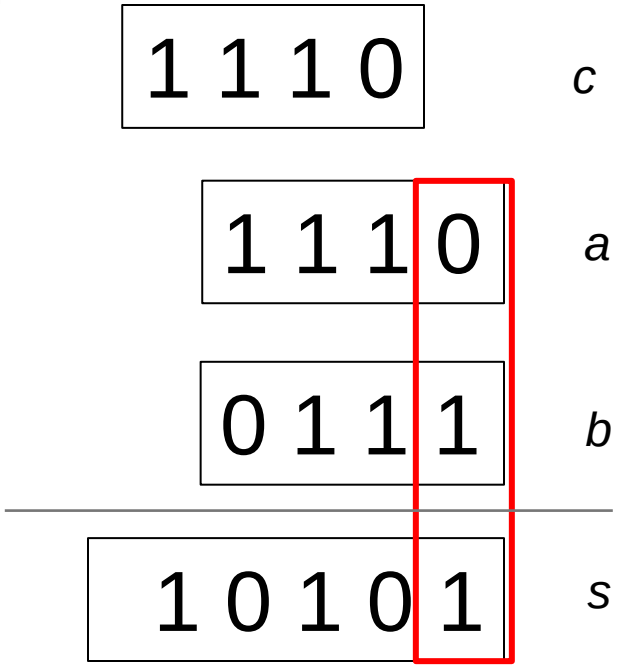
- **circuit de adunare**
  - se dau *a* și *b* pe *N* biți
  - vrem să calculăm *s* = *a* + *b*
    - pe câți biți este *s*?
    - *N+1* biți



    - câte intrări avem? 2N
    - câte ieșiri? N + 1
    - cât de mare va fi circuitul combinațional? $(N+1)2^{2N-1}$

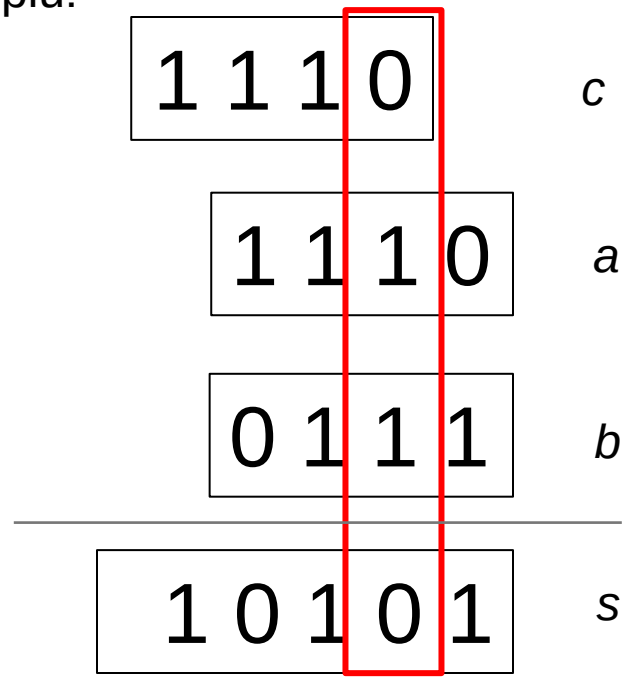
CIRCUITE MAI COMPLEXE

- **circuit de adunare**
  - cum am făcut până acum adunarea binară?
    - $s = a + b$
    - exemplu:



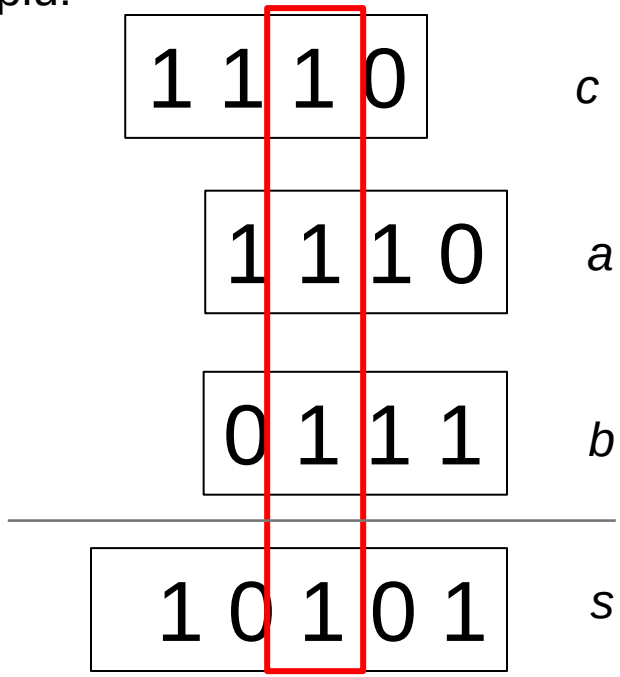
CIRCUITE MAI COMPLEXE

- **circuit de adunare**
  - cum am făcut până acum adunarea binară?
    - $s = a + b$
    - exemplu:



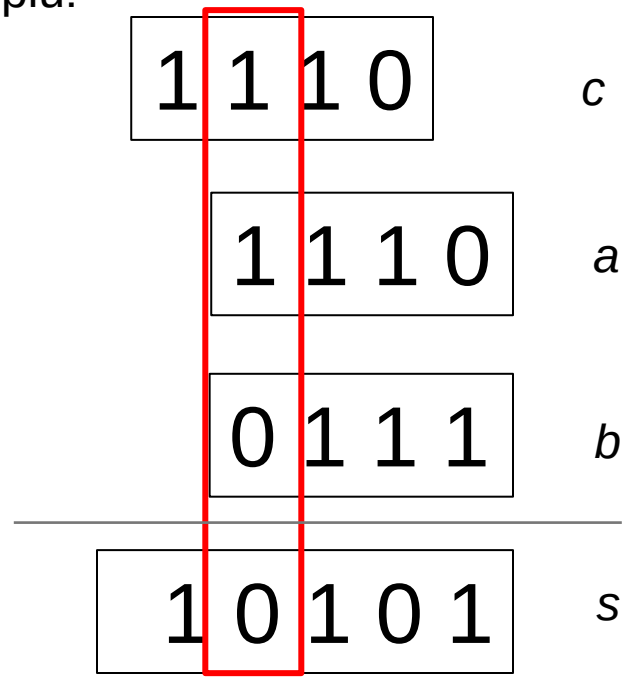
CIRCUITE MAI COMPLEXE

- **circuit de adunare**
  - cum am făcut până acum adunarea binară?
    - $s = a + b$
    - exemplu:



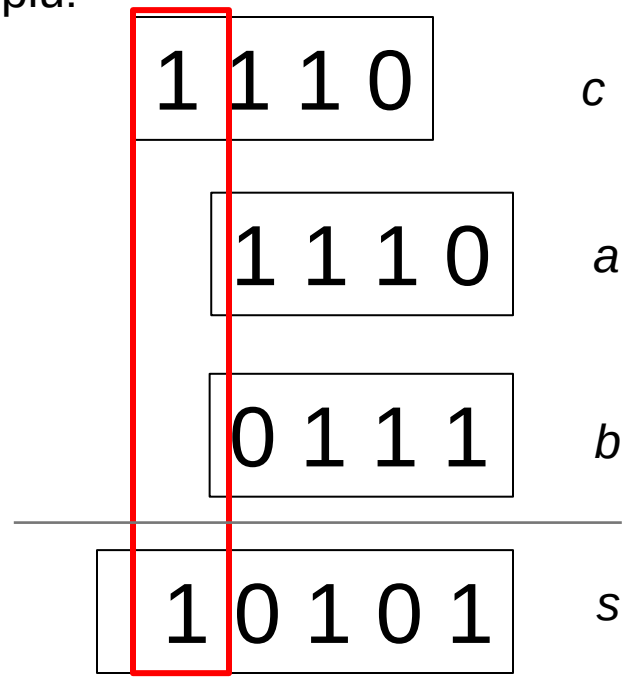
CIRCUITE MAI COMPLEXE

- **circuit de adunare**
  - cum am făcut până acum adunarea binară?
    - $s = a + b$
    - exemplu:



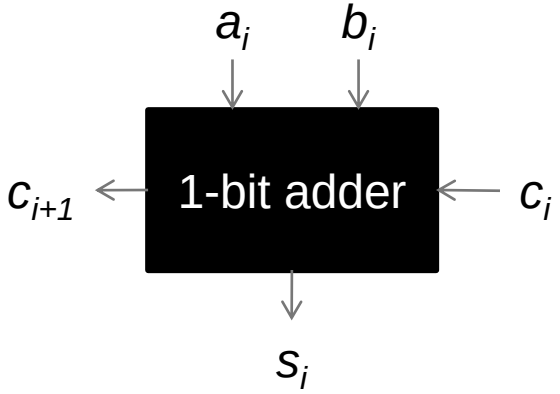
CIRCUITE MAI COMPLEXE

- **circuit de adunare**
  - cum am făcut până acum adunarea binară?
    - $s = a + b$
    - exemplu:



CIRCUITE MAI COMPLEXE

- **circuit de adunare**
  - ideea: definim un circuit bloc fundamental pe care bazăm totul



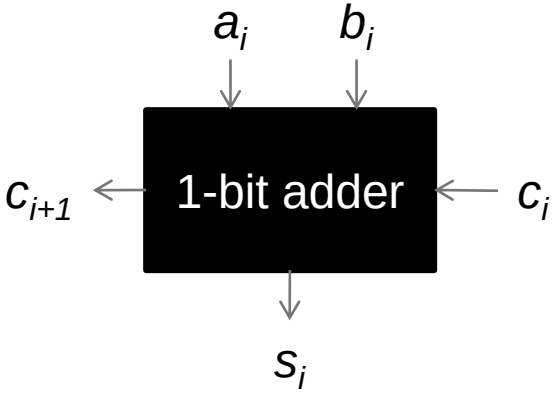
s_i = ?

c_{out} = ?

a_i	b_i	c_{in}/c_i	s_i	c_{out}/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

CIRCUITE MAI COMPLEXE

- **circuit de adunare**
  - ideea: definim un circuit bloc fundamental pe care bazăm totul



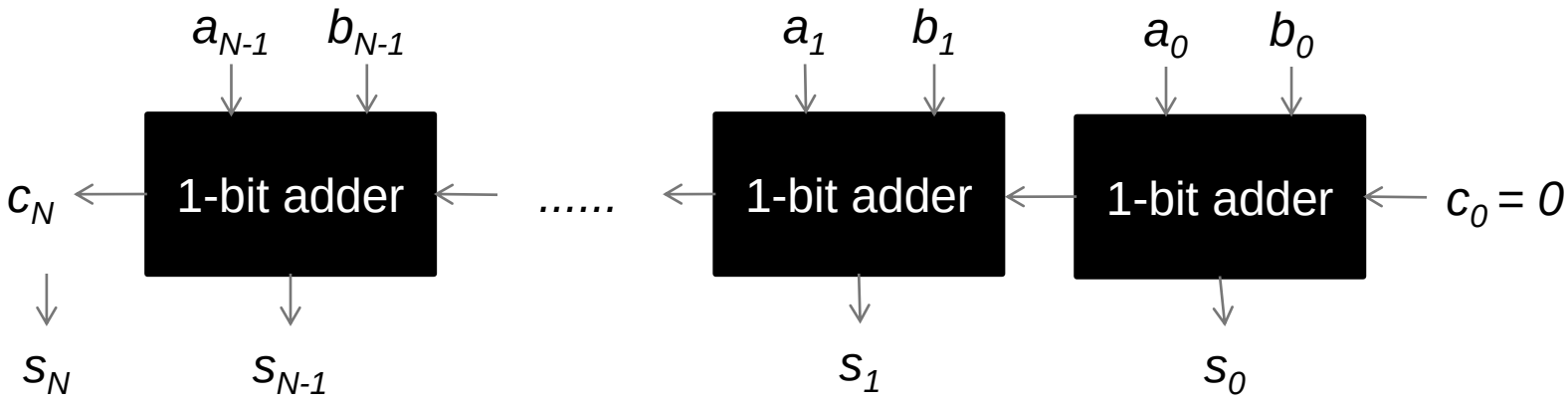
$$s_i = a_i \oplus b_i \oplus c_{in}$$

$$c_{out} = a_i b_i + (a_i + b_i) c_{in}$$

a_i	b_i	c_{in}/c_i	s_i	c_{out}/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

CIRCUITE MAI COMPLEXE

- **circuit de adunare**
  - ideea: definim un circuit bloc fundamental pe care bazăm totul
  - ideea: folosim circuitul fundamental în cascadă

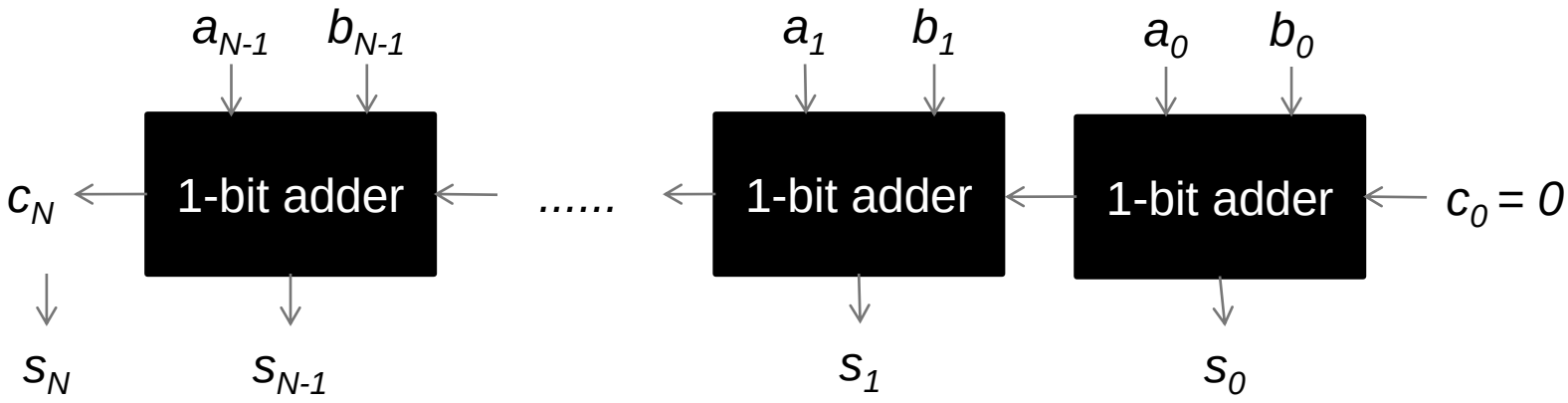


$$s_i = a_i \oplus b_i \oplus c_{in}$$

$$c_{out} = a_i b_i + (a_i + b_i) c_{in}$$

CIRCUITE MAI COMPLEXE

- **circuit de adunare**
  - ideea: definim un circuit bloc fundamental pe care bazăm totul
  - ideea: folosim circuitul fundamental în cascadă



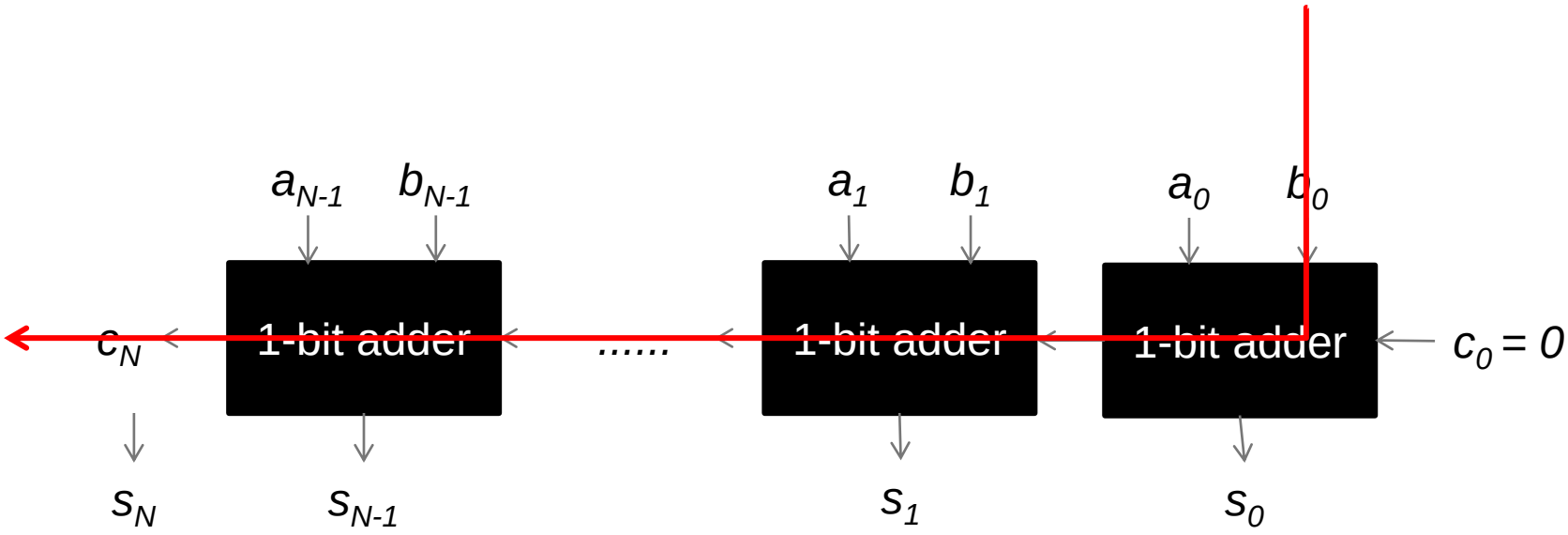
$$s_i = a_i \oplus b_i \oplus c_{in}$$

$$c_{out} = a_i b_i + (a_i + b_i) c_{in}$$

- data trecută am vorbit de t_P
- cât este timpul de propagare în acest caz?

CIRCUITE MAI COMPLEXE

- **circuit de adunare**
  - ideea: definim un circuit bloc fundamental pe care bazăm totul
  - ideea: folosim circuitul fundamental în cascadă



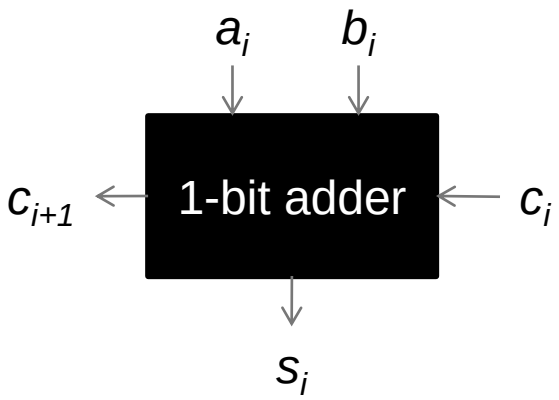
$$s_i = a_i \oplus b_i \oplus c_{in}$$

$$c_{out} = a_i b_i + (a_i + b_i) c_{in}$$

- data trecută am vorbit de t_P
- cât este timpul de propagare în acest caz?
- $N\ t_{P,\ 1\text{-bit adder}}$

CIRCUITE MAI COMPLEXE

- **circuit de adunare**
  - care este problema fundamentală în circuitul de mai sus?
    - trebuie să așteptăm să putem calcula acei biți de carry



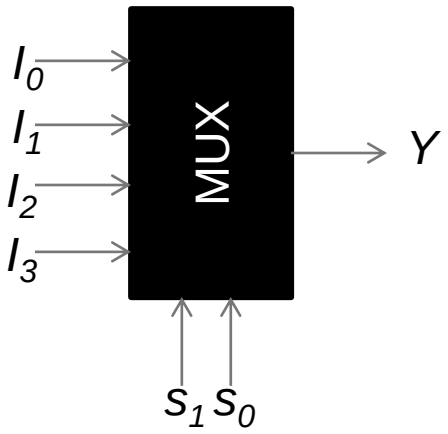
$$s_i = a_i \oplus b_i \oplus c_{in}$$

$$c_{out} = a_i b_i + (a_i + b_i) c_{in}$$

- notăm $g_i = a_i b_i$ și $p_i = a_i + b_i$
  - *dacă $g_i = 1$ atunci $c_{out} = 1$*
  - *dacă $g_i = 0$ și $p_i = 0$ atunci $c_{out} = 0$*
  - *dacă $g_i = 0$ și $p_i = 1$ atunci $c_{out} = c_{in}$*

CIRCUITE MAI COMPLEXE

- **multiplexare**
  - un circuit care selectează un semnal digital de la intrare pe baza unui semnal de activare *s*

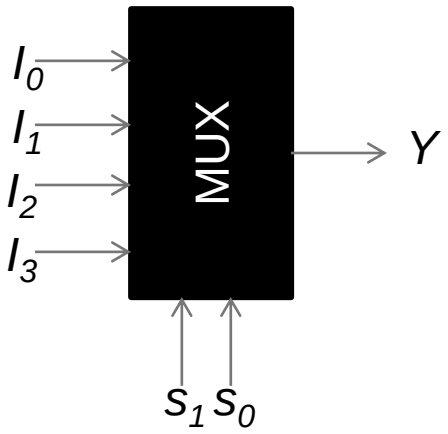


S_1	S_0	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

  - de ce e folositor acest circuit?

CIRCUITE MAI COMPLEXE

- **multiplexare**
  - un circuit care selectează un semnal digital de la intrare pe baza unui semnal de activare *s*

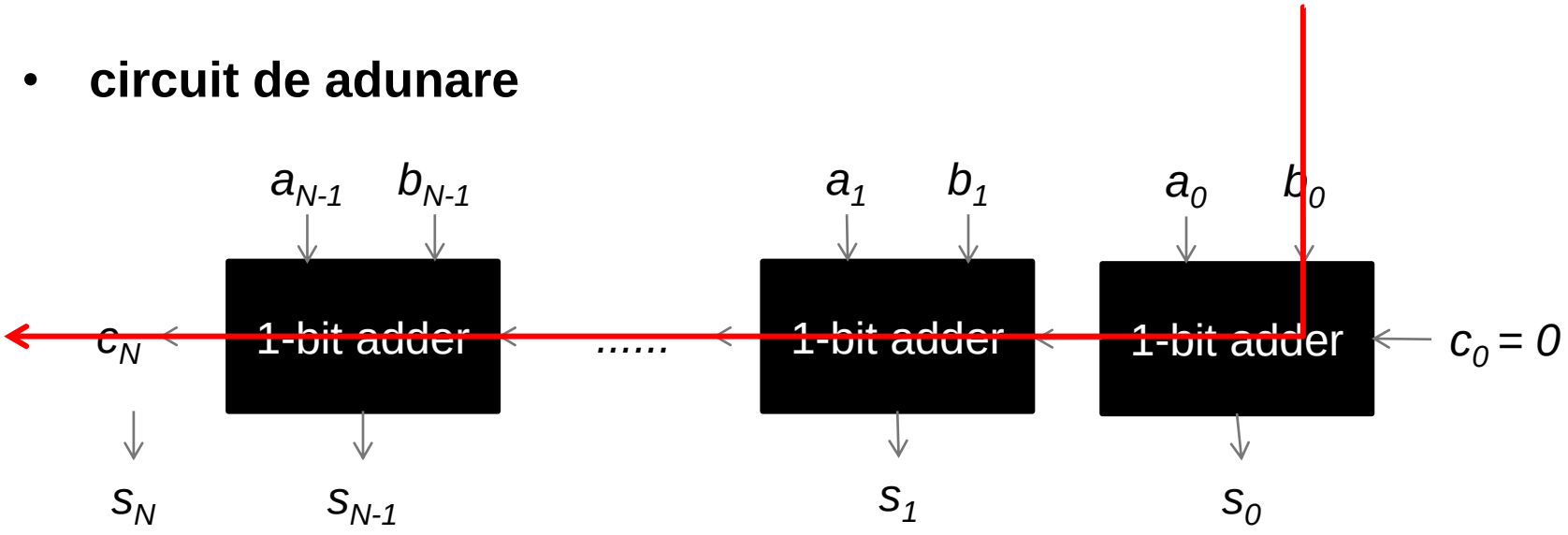


S_1	S_0	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

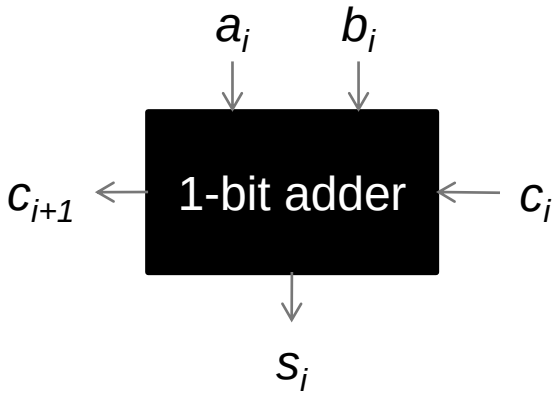
  - de ce e folositor acest circuit?
    - implementare hardware pentru “if” și “case”
    - implementare hardware pentru operații shift
    - vom vedea la seminar

CIRCUITE MAI COMPLEXE

- **circuit de adunare**



- care este problema fundamentală în circuitul de mai sus?
  - trebuie să așteptăm să putem calcula acei biți de carry

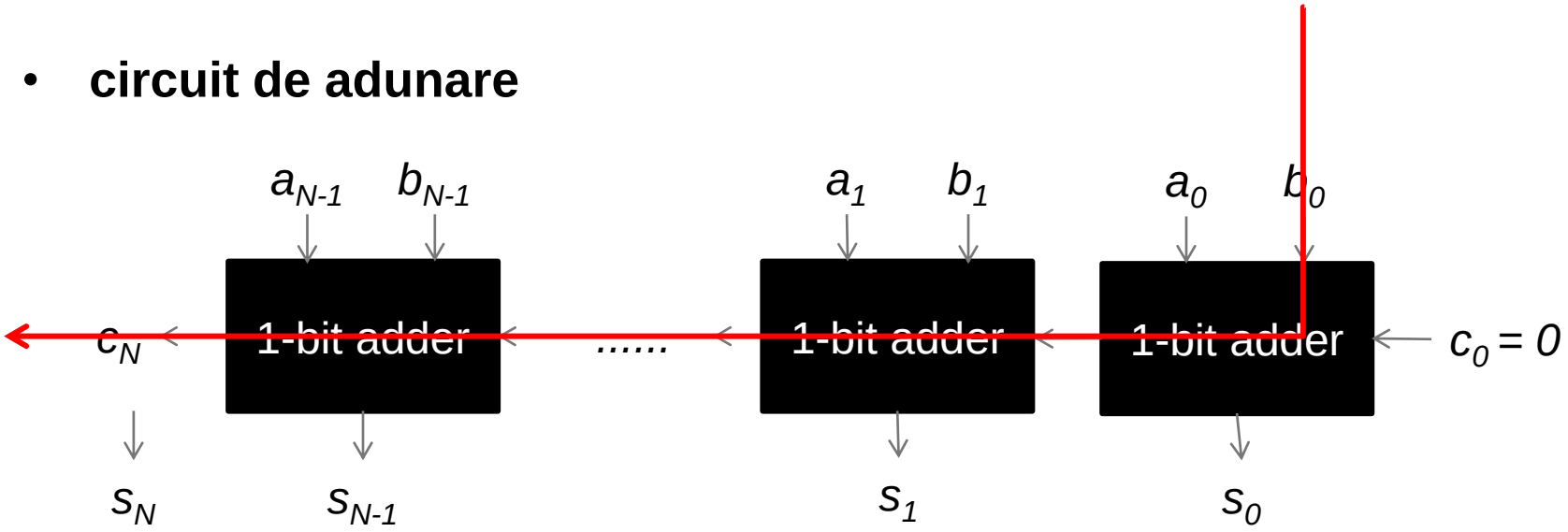


$$s_i = a_i \oplus b_i \oplus c_{in}$$

$$c_{out} = a_i b_i + (a_i + b_i) c_{in}$$

CIRCUITE MAI COMPLEXE

- **circuit de adunare**

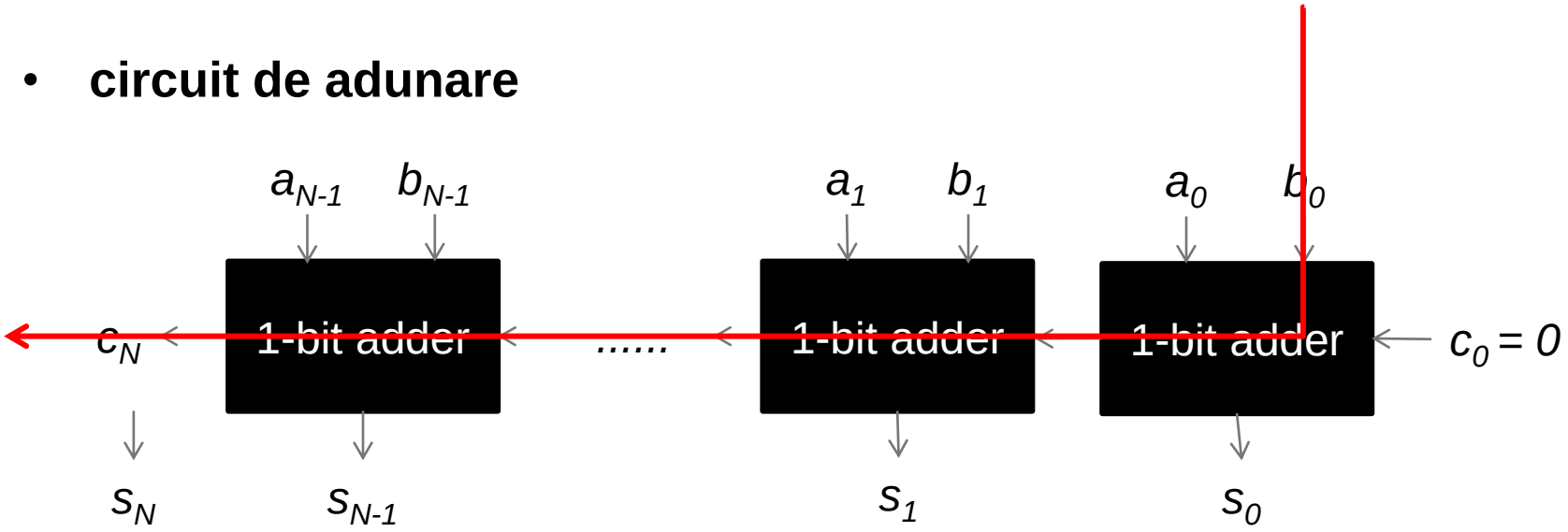


- cât este timpul de propagare în acest caz?
- $N\ t_P$

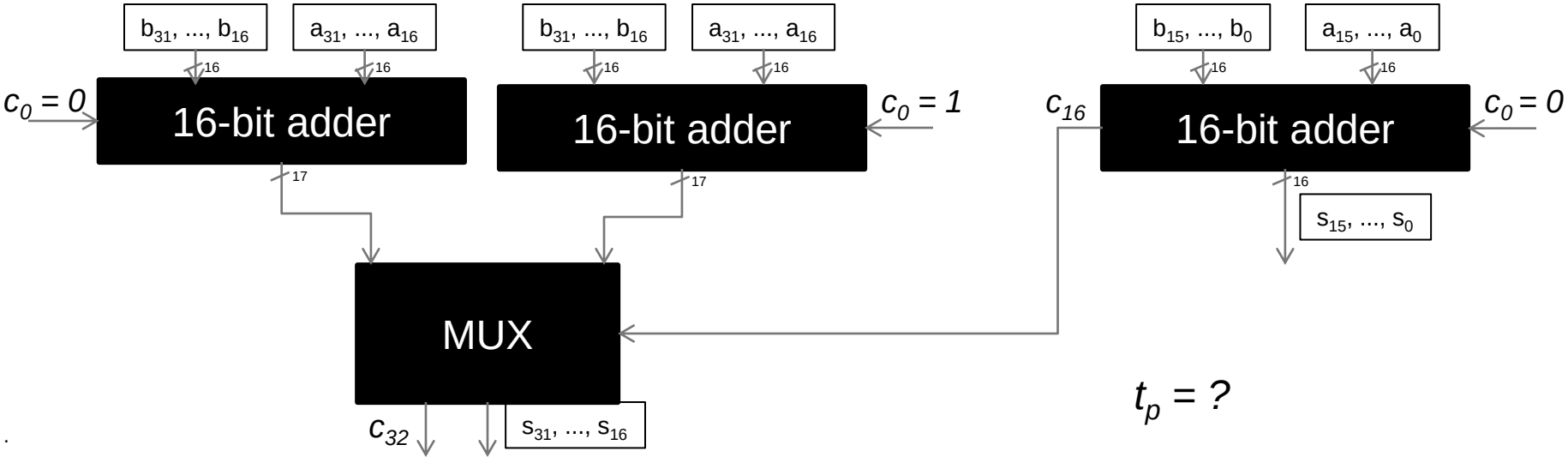
- ce putem face pentru a îmbunătății timpul de propagare?

CIRCUITE MAI COMPLEXE

- **circuit de adunare**

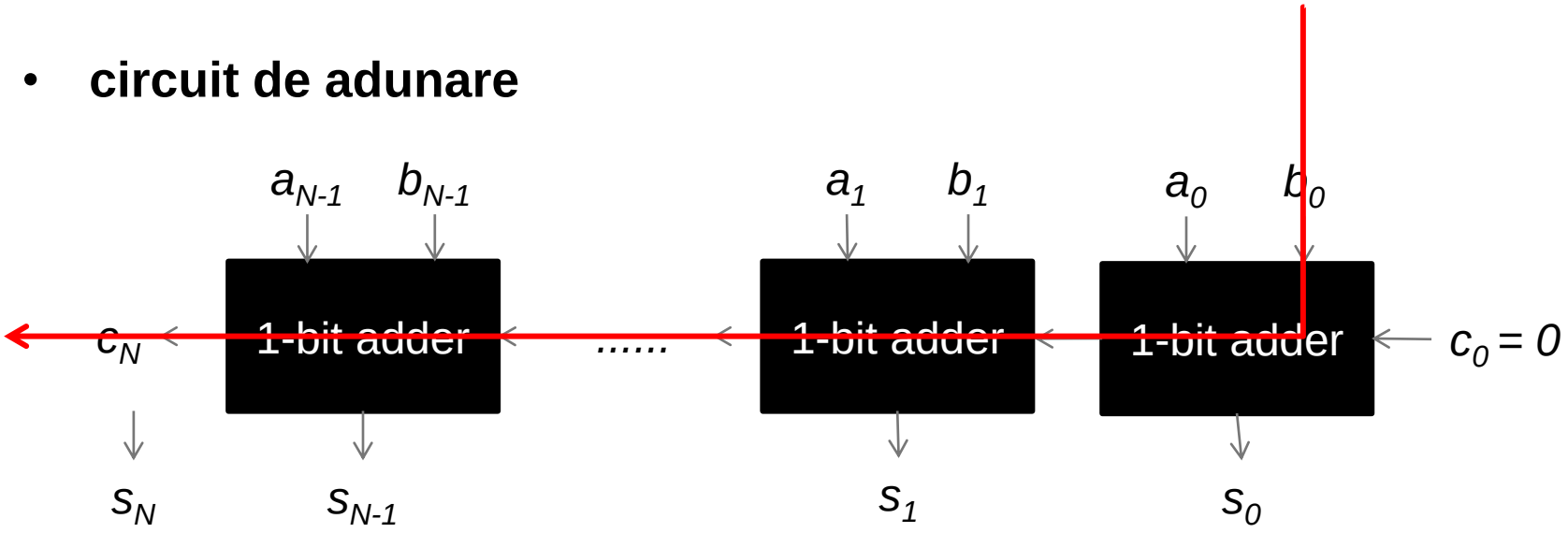


- cât este timpul de propagare în acest caz?
- $N\ t_P$

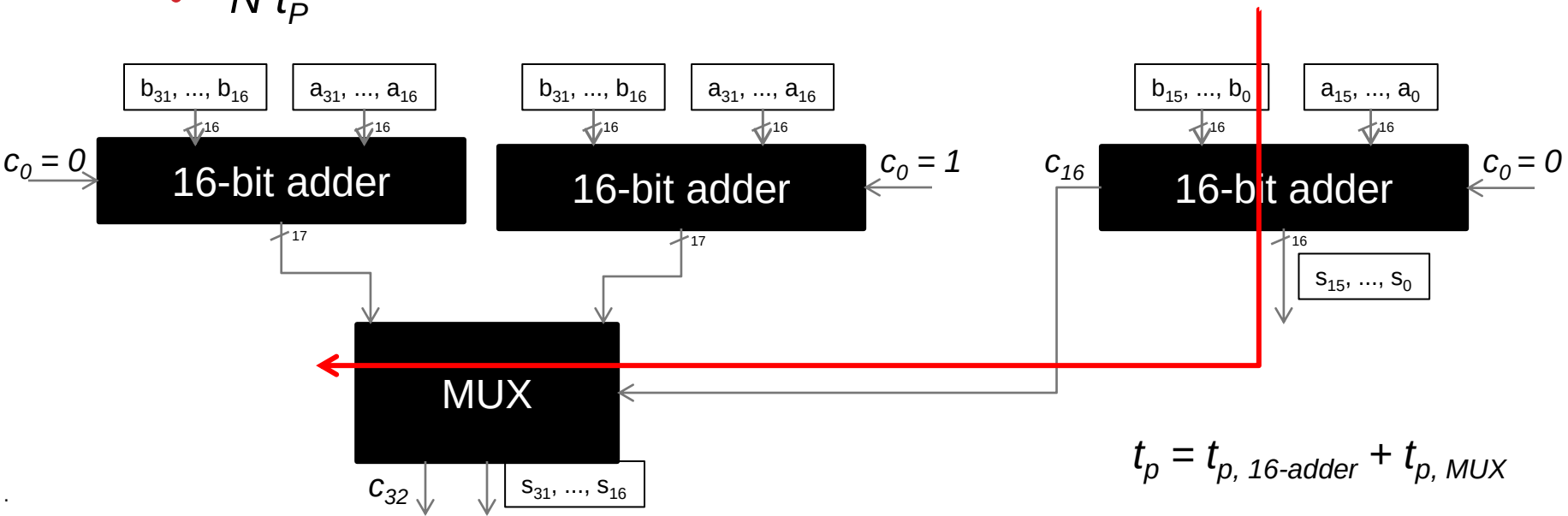


CIRCUITE MAI COMPLEXE

- **circuit de adunare**



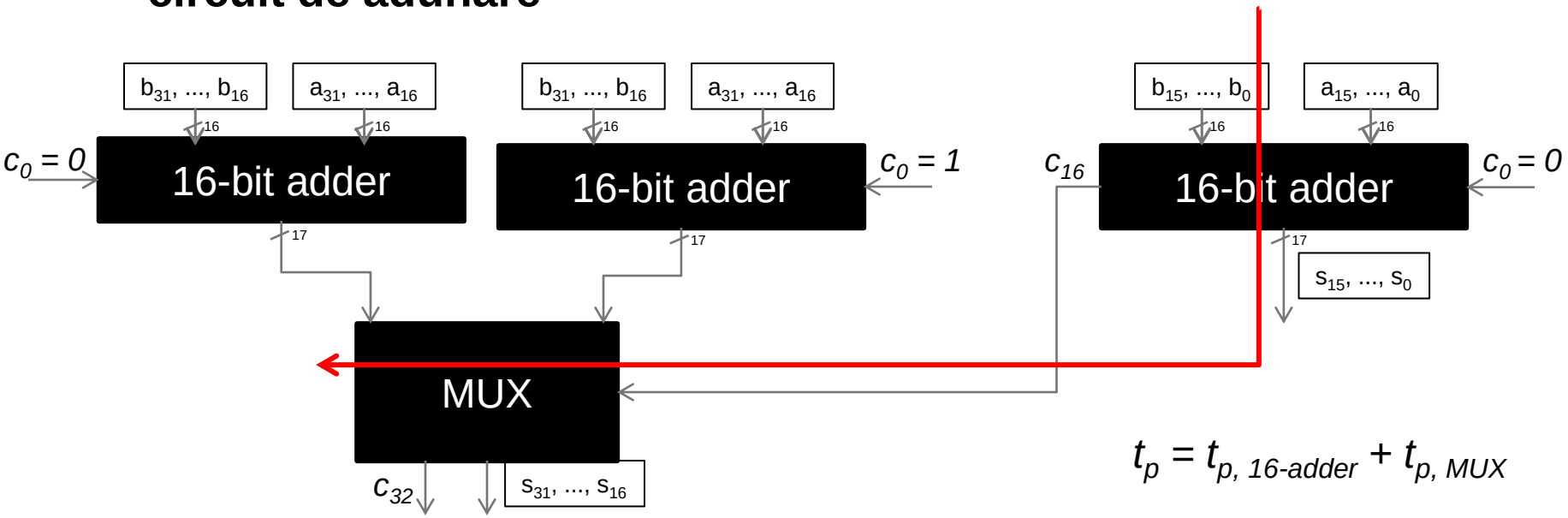
- cât este timpul de propagare în acest caz?
- $N\ t_P$



$t_p = t_{p,\ 16\text{-}adder} + t_{p,\ MUX}$

CIRCUITE MAI COMPLEXE

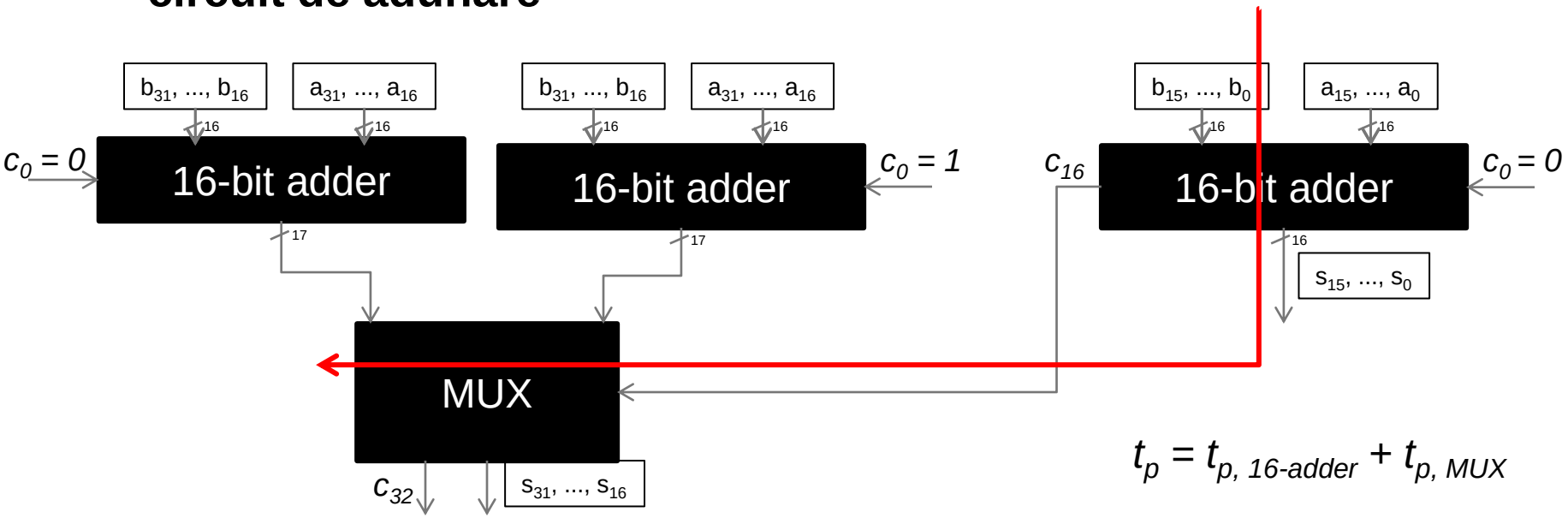
- **circuit de adunare**



- aici am considerat un exemplu de adunare pe 32 de biți
- putem aplica aceeași idee și pentru circuitele de 16 biți de sus
- $t_p = ?$

CIRCUITE MAI COMPLEXE

- **circuit de adunare**



- aici am considerat un exemplu de adunare pe 32 de biți
- putem aplica aceeași idee și pentru circuitele de 16 biți de sus
- $t_p = O(log_2 N)$

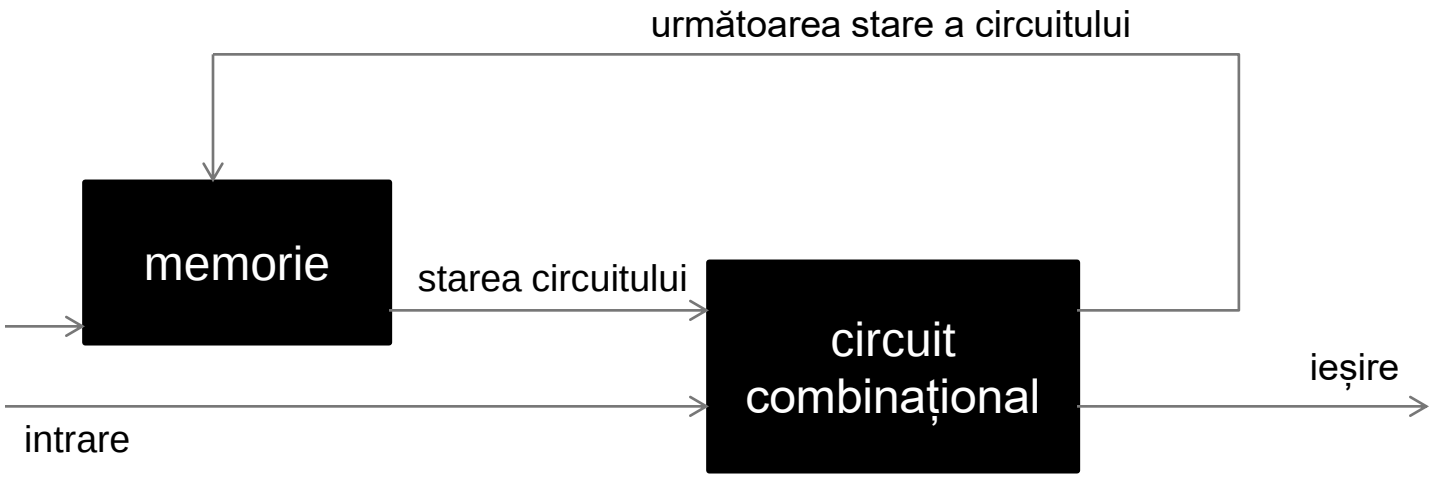
CIRCUITE SECVENȚIALE

- **circuitele combinaționale sunt eficiente (în general) dar au o problema majoră: sunt one-shot**
  - nu putem itera
  - nu permit niciun fel de “logică internă” sau “memorie internă”
  - nu există o stare internă a circuitului
  - sunt prea simple, asociază o funcție logică a intrărilor cu o ieșire (după un anumit timp)
  - unele lucruri nu pot fi implementate folosind doar logică combinațională

- **circuitele secvențiale adresează unele dintre limitările circuitelor combinaționale**

CIRCUITE SECVENȚIALE

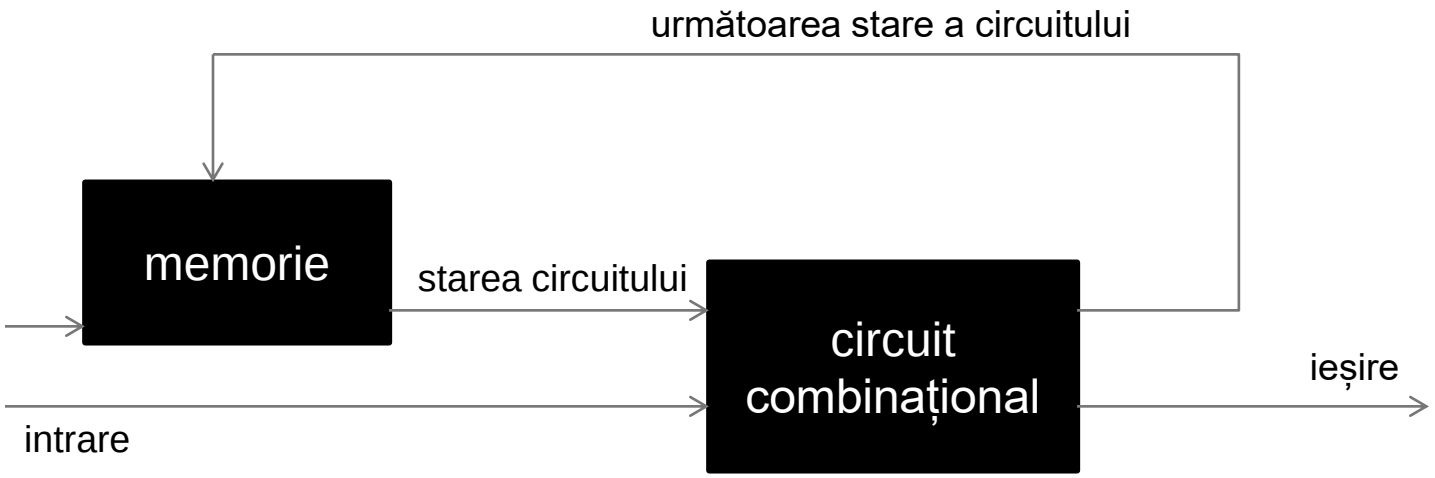
- **avem nevoie să introducem elemente de tip “memorie”**



- **în felul acesta adăugăm o stare circuitului**

CIRCUITE SECVENȚIALE

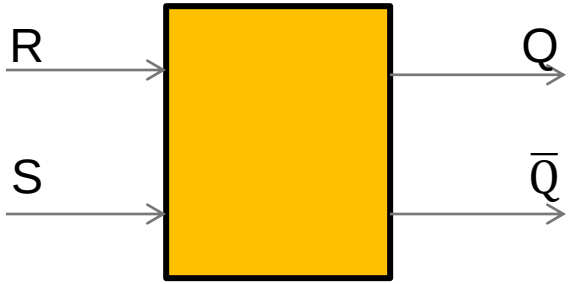
- **avem nevoie să introducem elemente de tip “memorie”**

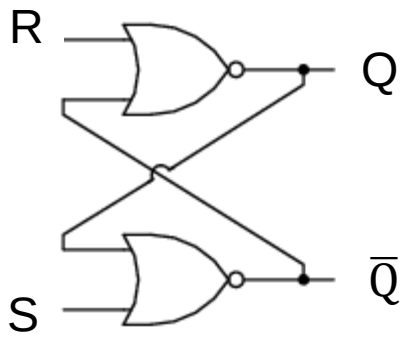


- **am vorbit până acum că biții pe care îi reprezentăm sunt voltaj**
- **dar energia electrică este dificil de stocat (în timp)**
  - fenomen de scurgere (leakage)
- **dacă vrem să memorăm ceva, trebuie să facem un refresh din când în când pentru a actualiza nivelul de energie electrică**

CIRCUITE SECVENȚIALE

- **SR Latch (Set-Reset Latch)**
  - memorează un bit de informație

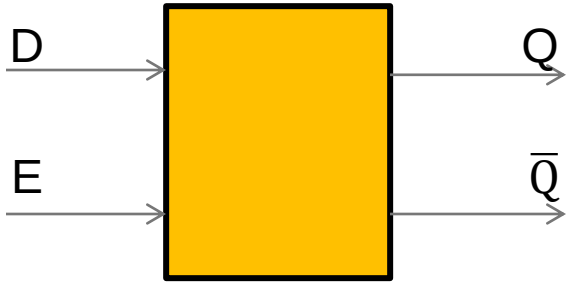


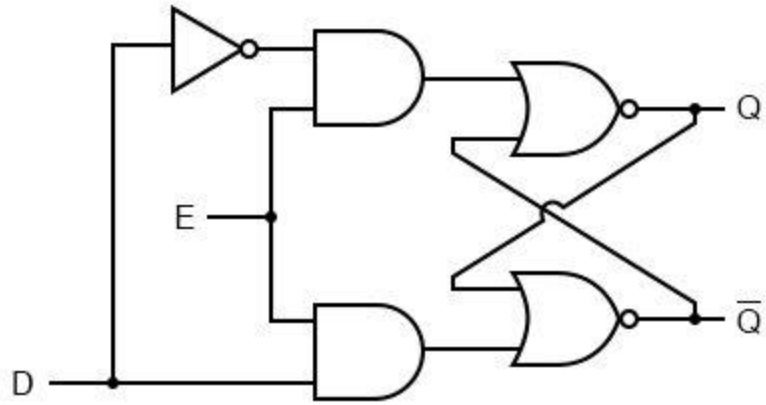


S	R	Q	$\overline{Q}$	
0	0	latch	latch	nu se schimbă nimic
0	1	0	1	aici punem “0” în memorie
1	0	1	0	aici punem “1” în memorie
1	1	0	0	stare invalidă

CIRCUITE SECVENȚIALE

- **SR Latch (Set-Reset Latch)**
  - e bun dar are două intrări, putem face ceva cu o singură intrare?
  - **D Latch**



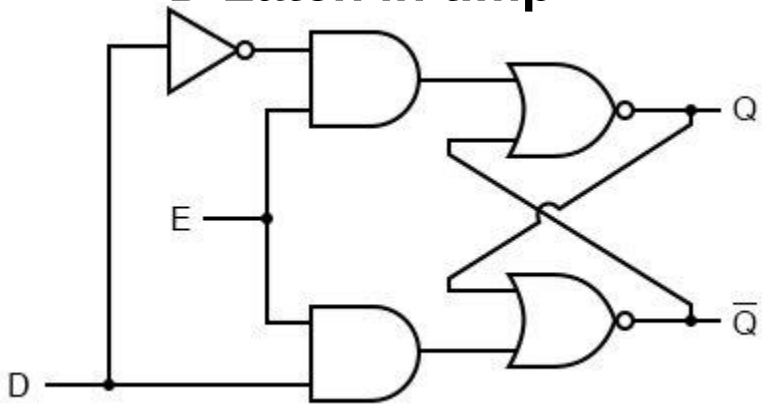


E	D	Q	$\overline{Q}$	
0	0	latch	latch	nu se schimbă nimic
0	1	latch	latch	nu se schimbă nimic
1	0	0	1	aici punem “0” în memorie
1	1	1	0	aici punem “1” în memorie

E de la Enable, adică activare
dacă E = 0 nu se întâmplă nimic

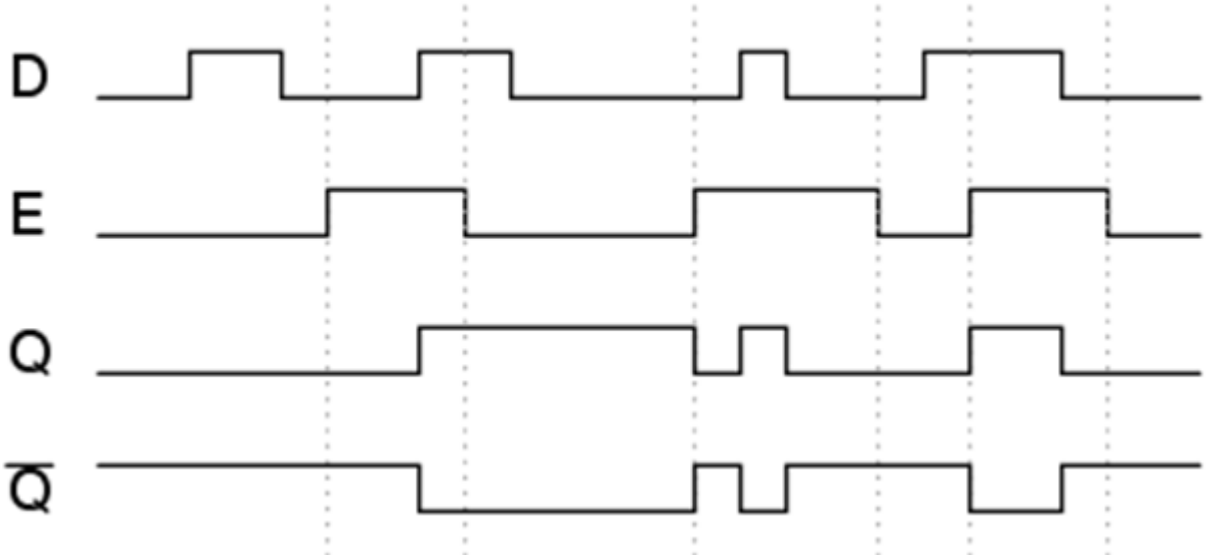
CIRCUITE SECVENȚIALE

- **D Latch în timp**



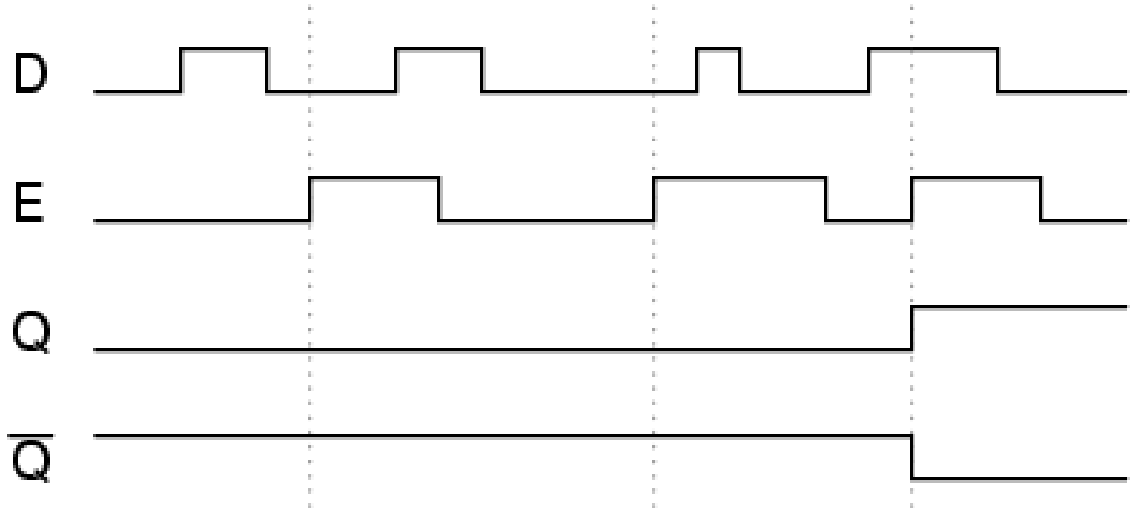
E	D	Q	$\overline{Q}$	
0	0	latch	latch	nu se schimbă nimic
0	1	latch	latch	nu se schimbă nimic
1	0	0	1	aici punem “0” în memorie
1	1	1	0	aici punem “1” în memorie

E de la Enable, adică activare
dacă E = 0 nu se întâmplă nimic



CIRCUITE SECVENȚIALE

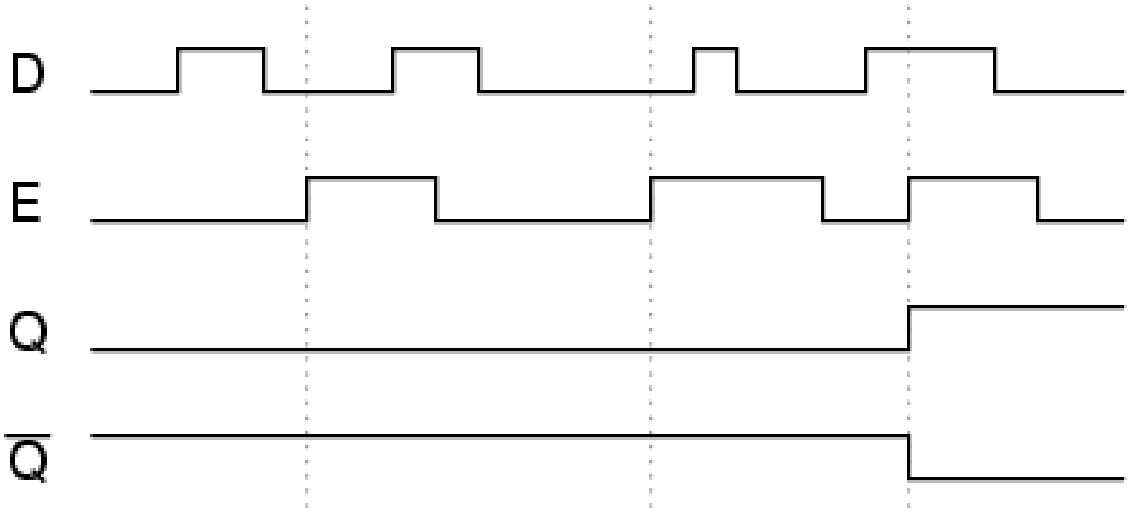
- **D Flip-Flop**
  - D Latch e bun, dar vrem să sincronizăm mai multe dispozitive
  - vrem ca activarea să se facă cand E crește, nu când E e activ



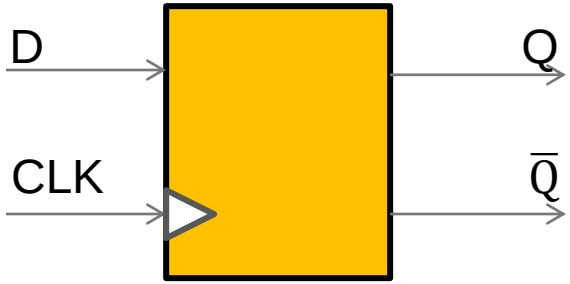
  - E devine clock – adică ceasul sistemului
    - la un interval fix de timp, sistemul face ceva

CIRCUITE SECVENȚIALE

- **D Flip-Flop**



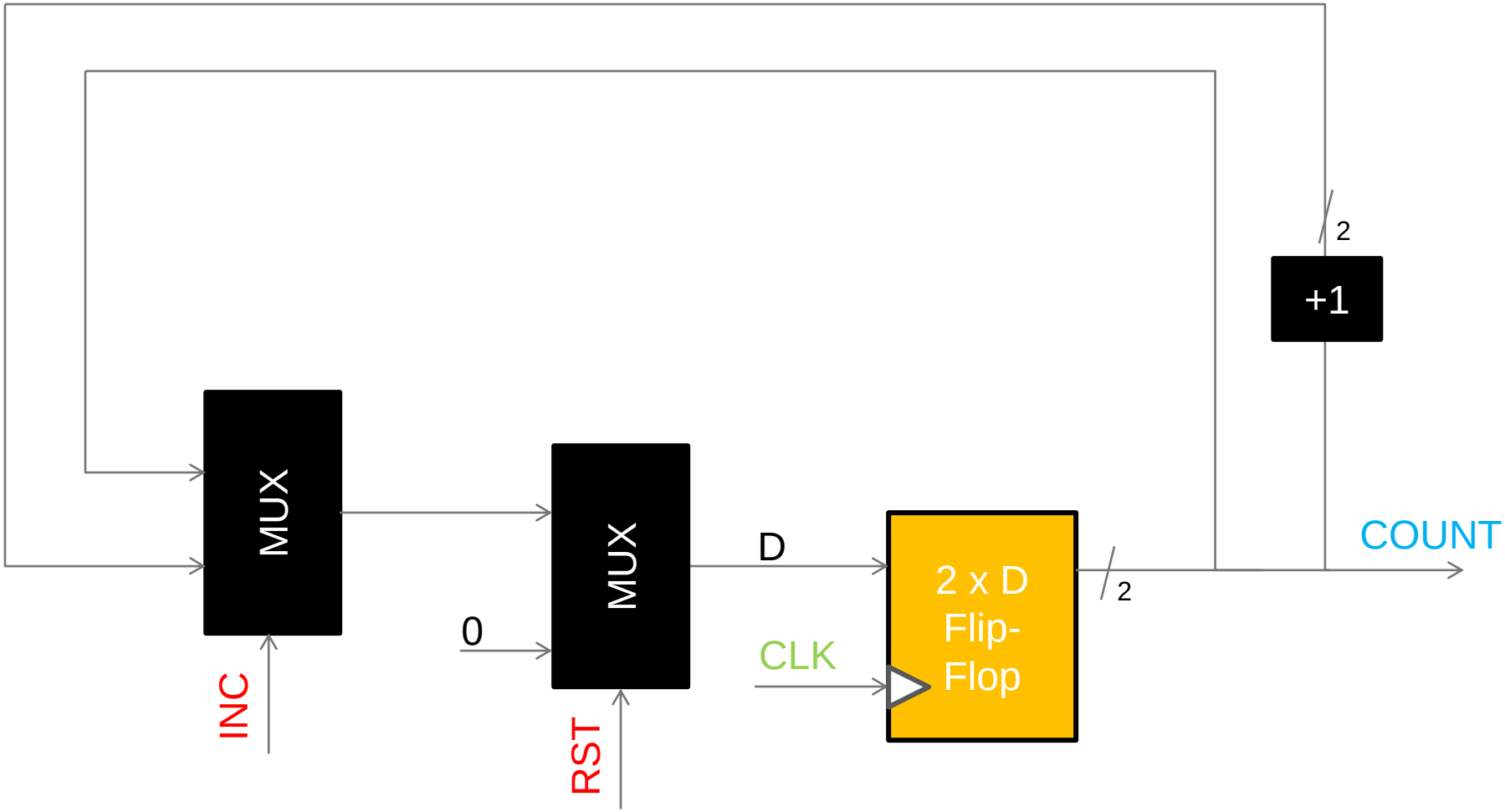
- E devine clock – adică ceasul sistemului
  - la un interval fix de timp (un ciclu), sistemul face ceva



un set de câteva D Flip Flops care au același CLK = un registru

CIRCUITE SECVENȚIALE

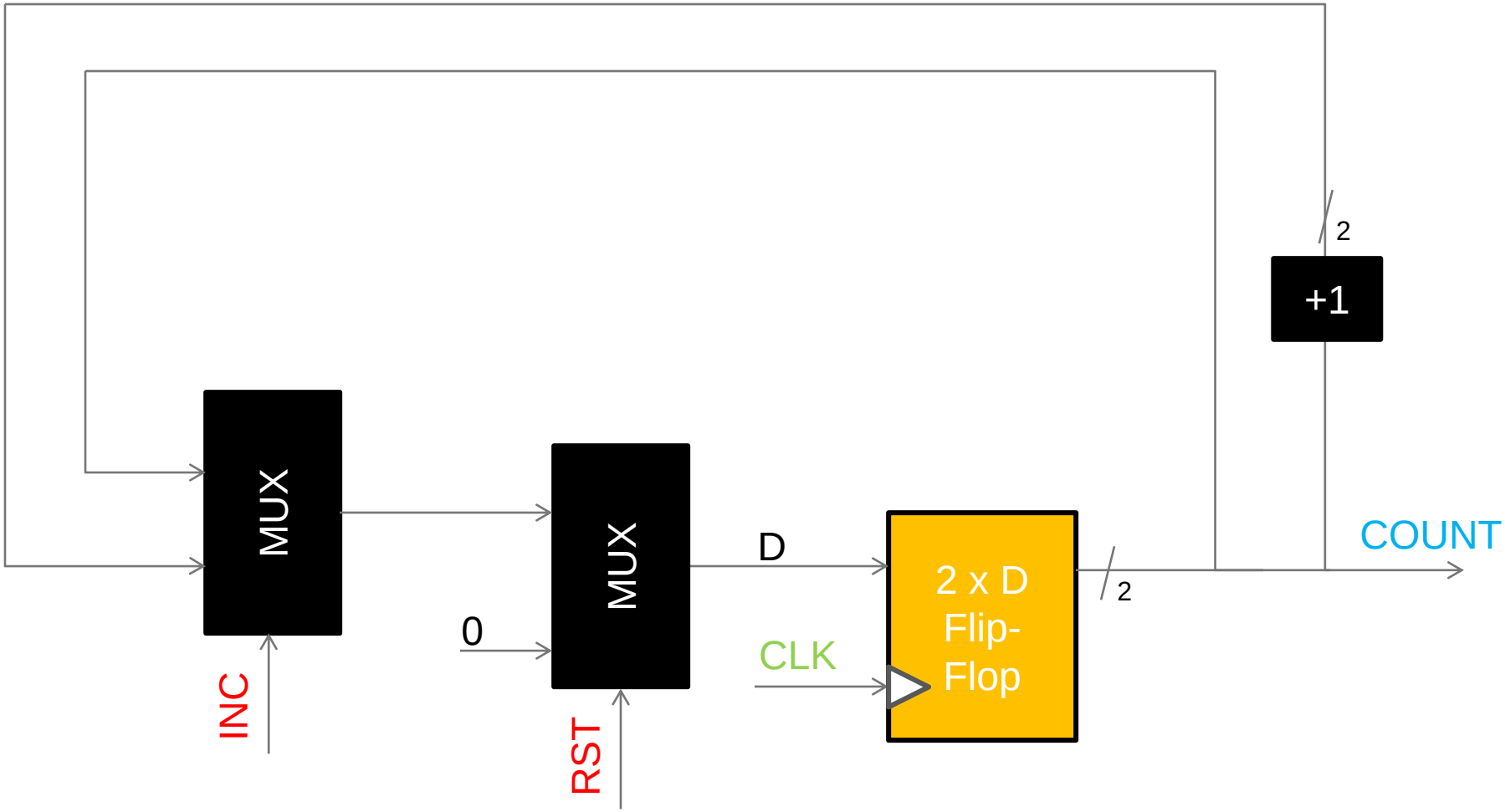
- **un exemplu**



ce face acest circuit?

CIRCUITE SECVENȚIALE

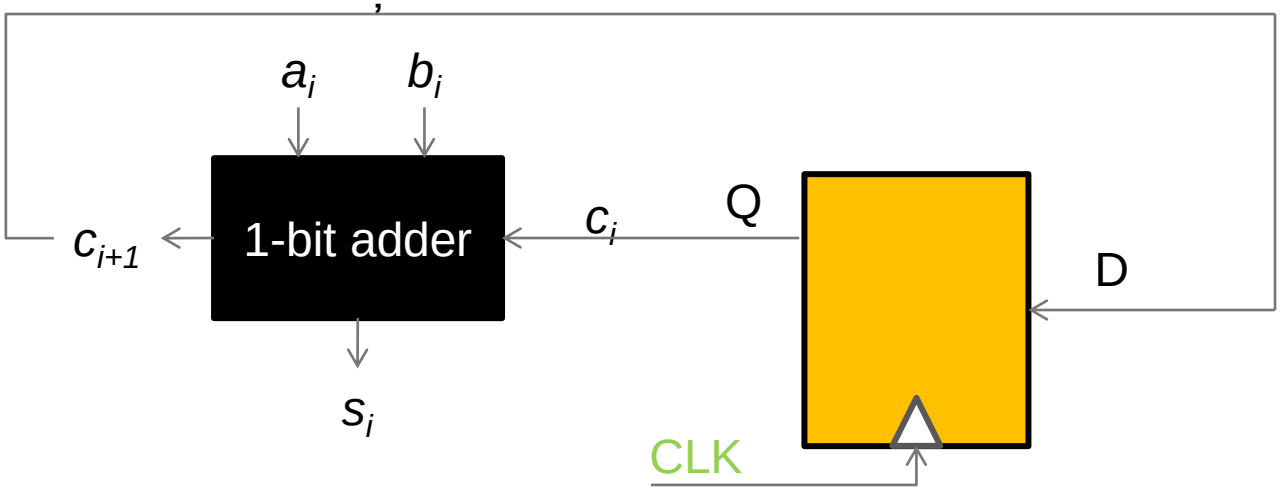
- **un exemplu**



este un counter pe 2 biți

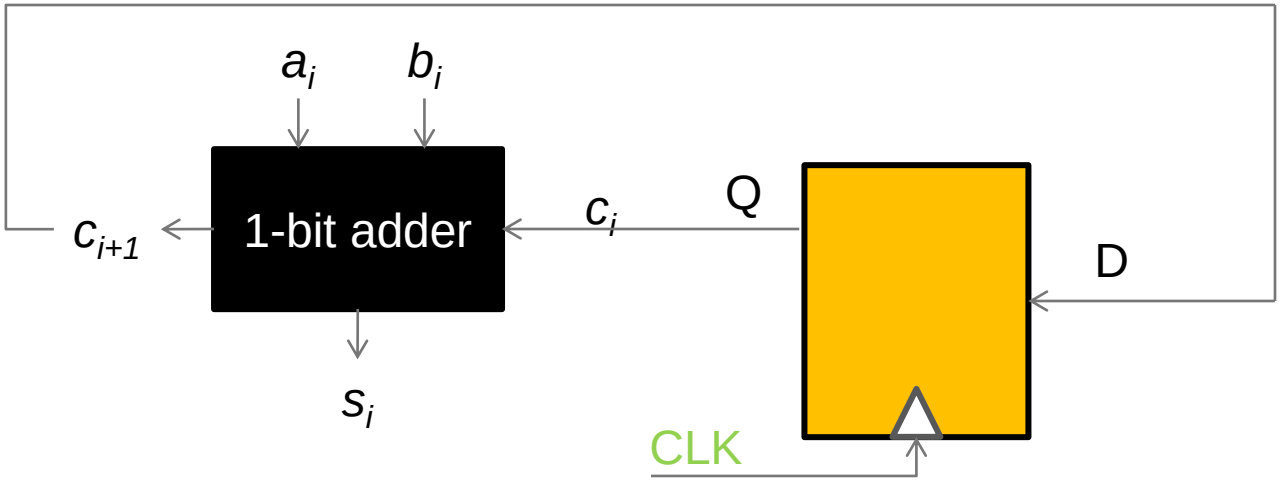
CIRCUITE SECVENȚIALE

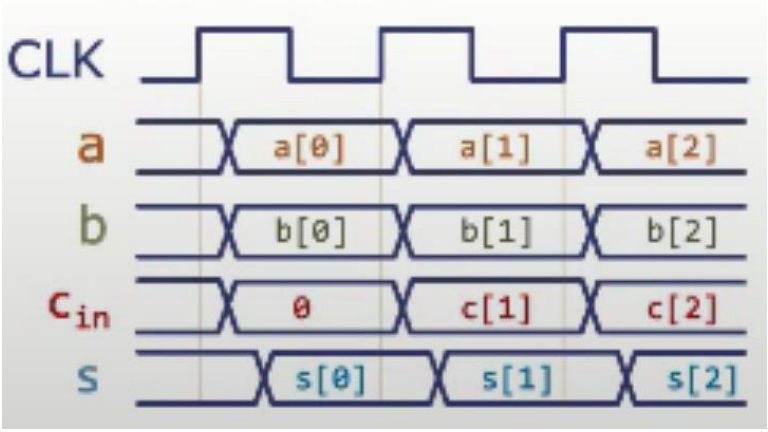
- **avantajele circuitelor secvențiale**
  - avem stare internă
    - exemplu: avem un registru în care memorăm count-ul curent
  - avem variabila de timp
    - intrările/ieșirile nu sunt fixe
    - număr variabil de pași în rezolvare
    - exemplu: construiți un circuit care poate să adune două numere de dimensiune (număr de biți) variabil – adică nu prestabilim pe câți biți e fiecare număr
      - noi așa facem adunarea binară. cum arată circuitul?



CIRCUITE SECVENȚIALE

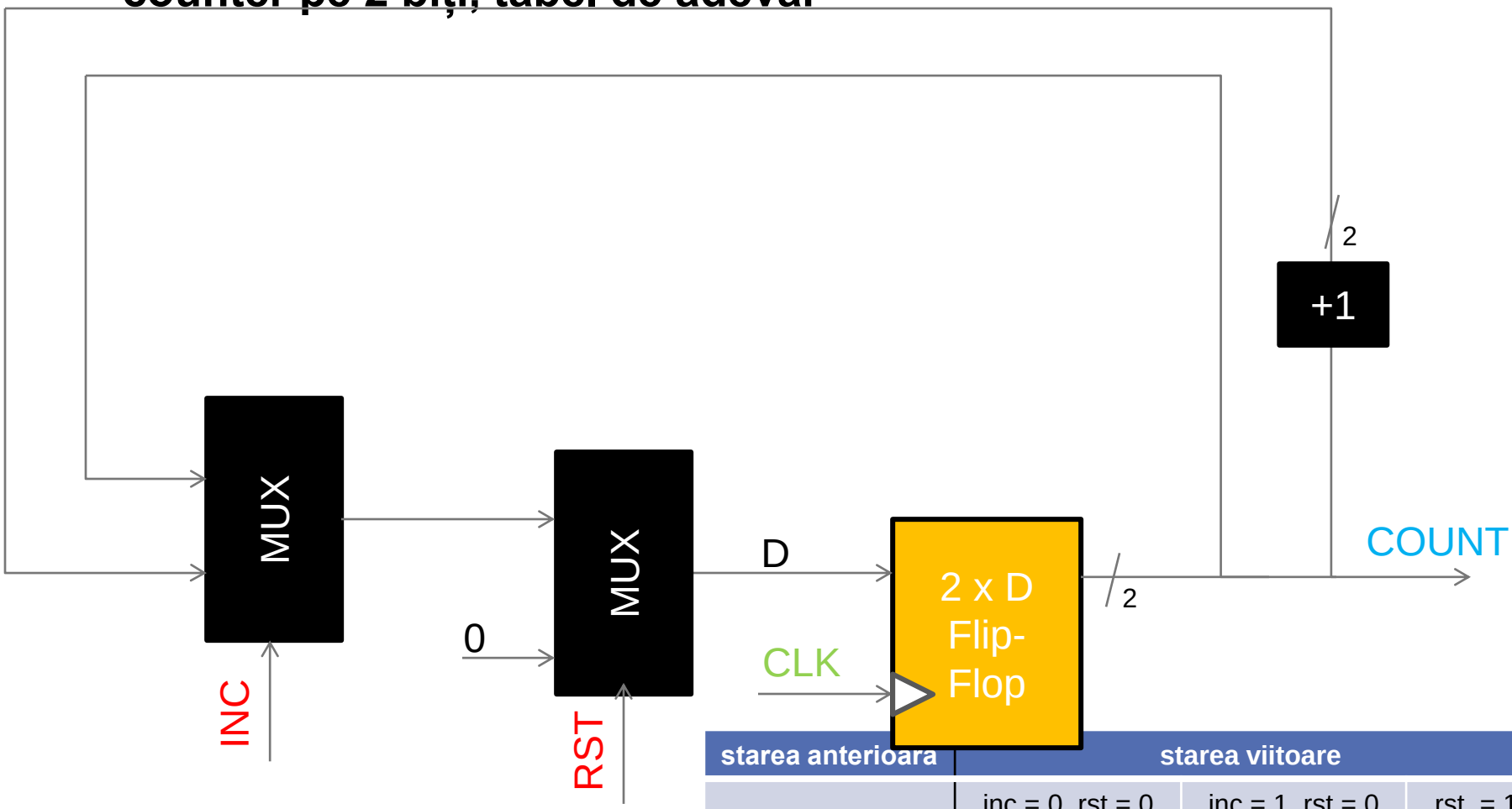
- **circuit de adunare, în timp**





CIRCUITE SECVENȚIALE

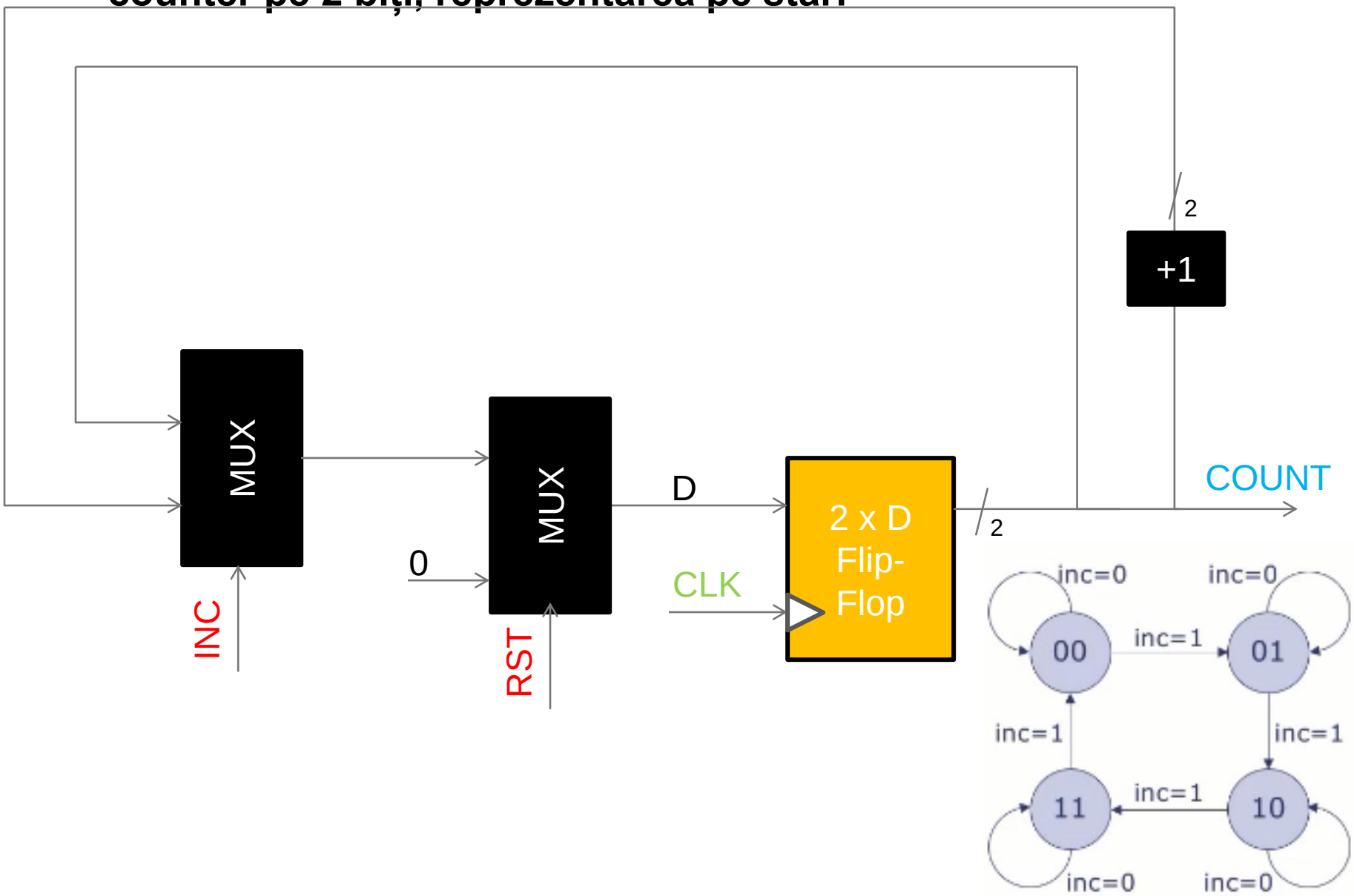
- **counter pe 2 biți, tabel de adevăr**



starea anterioara	starea viitoare		
	inc = 0, rst = 0	inc = 1, rst = 0	rst = 1
00	00	01	00
01	01	10	00
10	10	11	00
11	11	00	00

CIRCUITE SECVENȚIALE

- **counter pe 2 biți, reprezentarea pe stări**



CE AM FĂCUT ASTĂZI

- **tabele de adevăr și expresii booleene**
- **caracteristici ale porților logice**
- **circuitul de adunare combinațional**
- **circuite secvențiale**
  - SR Latch
  - D Latch
  - D Flip Flop
  - circuit de adunare secvențial

DATA VIITOARE ...

- **seminarul următor are multe exerciții cu circuite combinaționale și secvențiale**
- **mai multe circuite secvențiale mai complexe**
- **mai multe despre reprezentarea pe stare**
- **înmulțirea binară**

LECTURĂ SUPLIMENTARĂ

- **PH book**
  - Appendix B

- **Computerphile, Logic gates playlist, https://www.youtube.com/playlist?list=PLzH6n4zXucko1YVRjhnquPaNOfZrymVQH**

- **Computerphile, Binary Addition & Overflow, https://www.youtube.com/watch?v=WN8i5cwjkSE**

- **Ben Eater, SR latch, https://www.youtube.com/watch?v=KM0DdEaY5sY**

- **Ben Eater, D latch, https://www.youtube.com/watch?v=peCh_859q7Q**

- **Ben Eater, D flip-flop, https://www.youtube.com/watch?v=YW-_GkUguMM**

- **How do computers remember?, https://www.youtube.com/watch?v=I0-izyq6q5s**

- **Visualizing binary data, https://www.youtube.com/watch?v=hEDQpqhY2MA**

.