ARHITECTURA SISTEMELOR DE CALCUL - CURS 0x03

ABSTRACTIZAREA DIGITALĂ
CIRCUITE COMBINAȚIONALE, SECVENȚIALE

Cristian Rusu

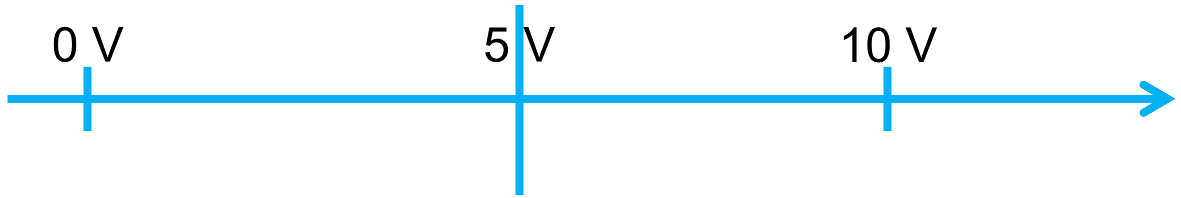
DATA TRECUTĂ

- am discutat despre conceptul de informație
- am văzut cum măsurăm informația
- am calculat entropia lui Shannon
- am văzut algoritmul Huffman pentru codarea variabilă a datelor
- am folosit distanța Hamming între două șiruri de biți
- azi, vom vedea cum implementăm acești biți în circuite

CUPRINS

ABSTRACTIZAREA DIGITALĂ

- **de la continuu la digital**

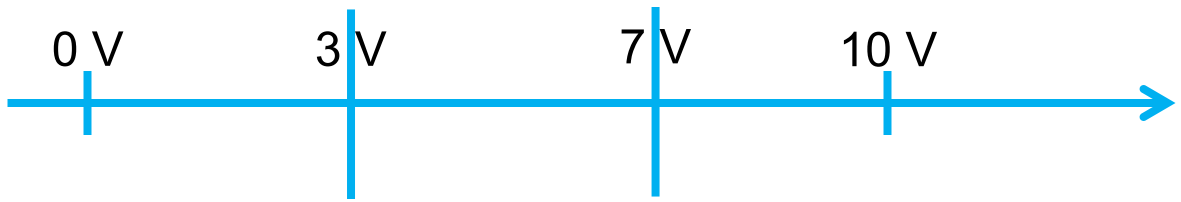


- cea mai simplă ideea: avem un voltaj maxim care poate să fie atins: deci de la 0V la 5V codăm “0” iar de la 5V la 10V avem “1”

- ce dificultăți avem în acestă situație?
  - e dificil să înțelegem ce se întamplă în jurul lui 5V

ABSTRACTIZAREA DIGITALĂ

- **de la continuu la digital**



- o soluție puțin mai sofisticată: avem două limite
- de data asta: “0” este între 0V și 3V iar “1” este între 7V și 10V
- intervalul între 3V și 7V este un “no man’s land”
  - nu putem decide voltajul
  - așteptăm stabilizarea la o valoare < 3V sau > 7V

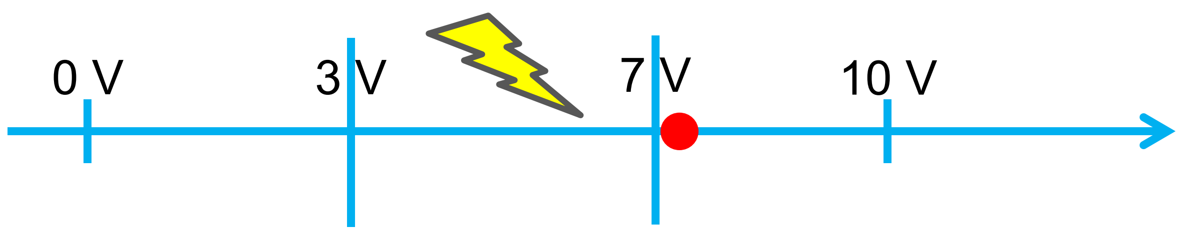
Atenție: sistemul nu este perfect, la 3V suntem “0” dar un zgomot de doar 4V din acest punct ne poate duce la 7V, deci “1”

ABSTRACTIZAREA DIGITALĂ

- **de la continuu la digital**
  - în cele mai multe cazuri, vom conecta dispozitive digitale între ele

  - de exemplu:
    - primul circuit scoate un “0”, dar la limita superioară

    - semnalul este trimis către al doilea circuit, dar apare zgomot pe fir



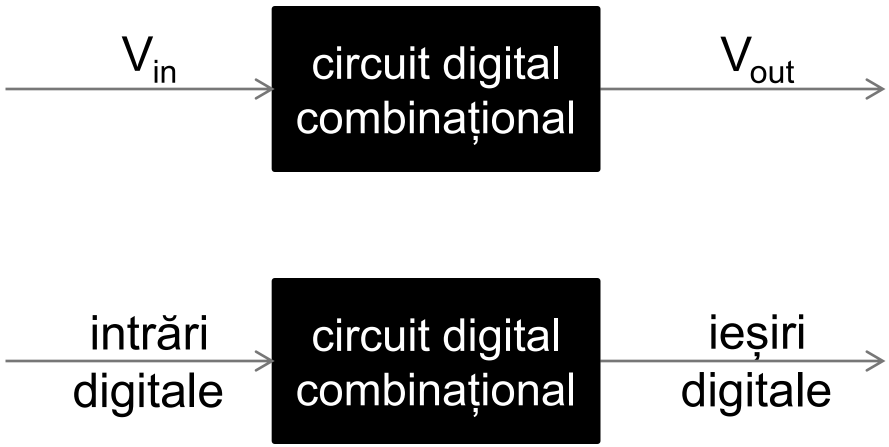
Atenție: sistemul nu este perfect, la 3V suntem “0” dar un zgomot de doar 4V din acest punct ne poate duce la 7V, deci “1”

Soluția? pentru ieșiri vom impune limite mai stricte (sub 2V, peste 8V)

ideal, am vrea ca limitele să fie cât mai înguste: 0V este “0” iar 10V este “1”

CIRCUITE DIGITALE

- **circuit digital combinațional**



CIRCUITE DIGITALE

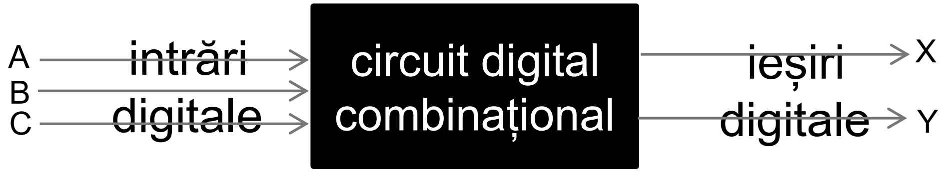
- **circuit digital combinațional**

- intrări digitale, pot fi multe
- ieșiri digitale, pot fi multe
- nu are stări interne
  - pui un semnal digital constant la intrare și ai un alt semnal digital constant la ieșire
  - dar nu poate “memora” nimic
  - nu are o “stare internă” (memorie)
- avem un timp de propagare (t_p): timpul maxim necesar pentru a produce la ieșire semnale digitale corecte și valide din momentul în care la intrare s-au specificat semnale digitale corecte și valide

- de ce se numesc circuite combinaționale?
  - pentru că ieșire este o combinație (o funcție logică care combină) toate (sau o parte) a intrărilor

CIRCUITE DIGITALE

- **circuit digital combinațional**



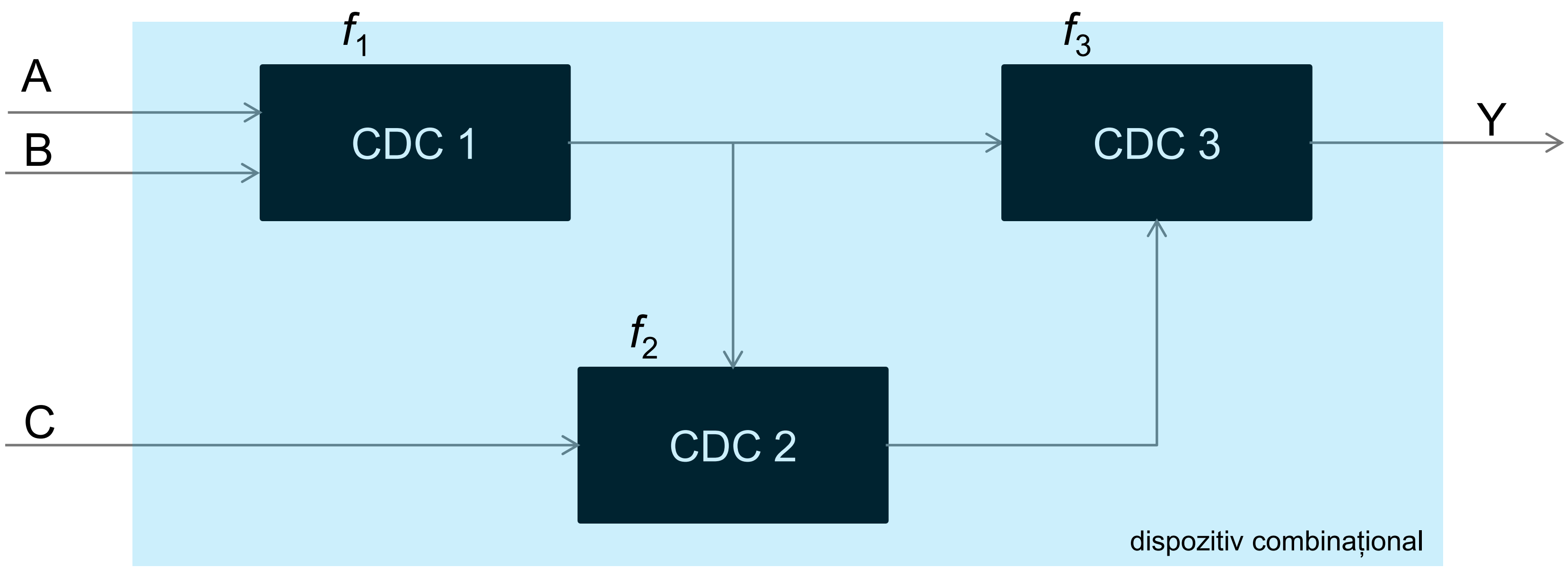
- de ce se numesc circuite combinaționale?
  - pentru că ieșire este o combinație (o funcție logică care combină) toate (sau o parte) a intrărilor
  - deci, pentru fiecare intrare, trebuie sa știm care e ieșirea

A	B	C	X	Y
0	0	0	0	1
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	0

tabel de adevăr

CIRCUITE DIGITALE

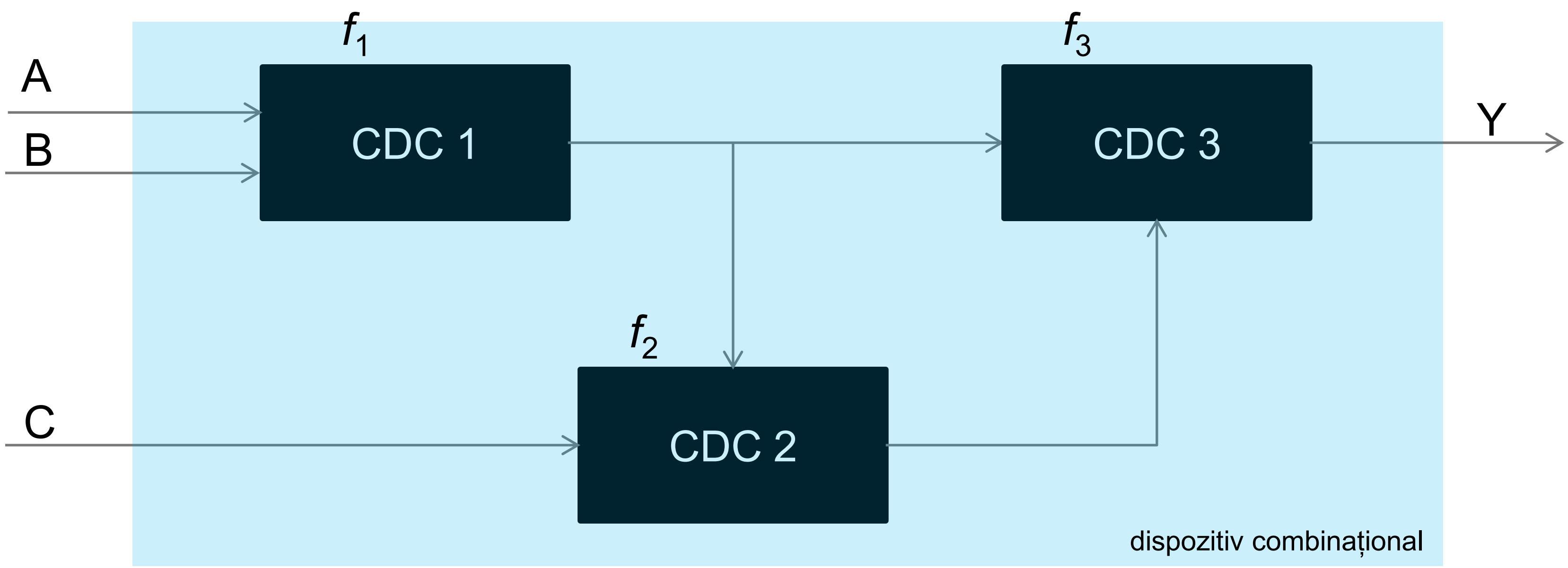
- **dispozitiv combinațional**
  - fiecare element este un circuit combinațional
  - fiecare intrare este conectată la exact o ieșire sau la o constantă
  - nu există niciun ciclu în graful direcțional al dispozitivului



  - care este funcția dispozitivului? $Y = f_3(f_1(A, B), f_2(f_1(A, B), C))$

CIRCUITE DIGITALE

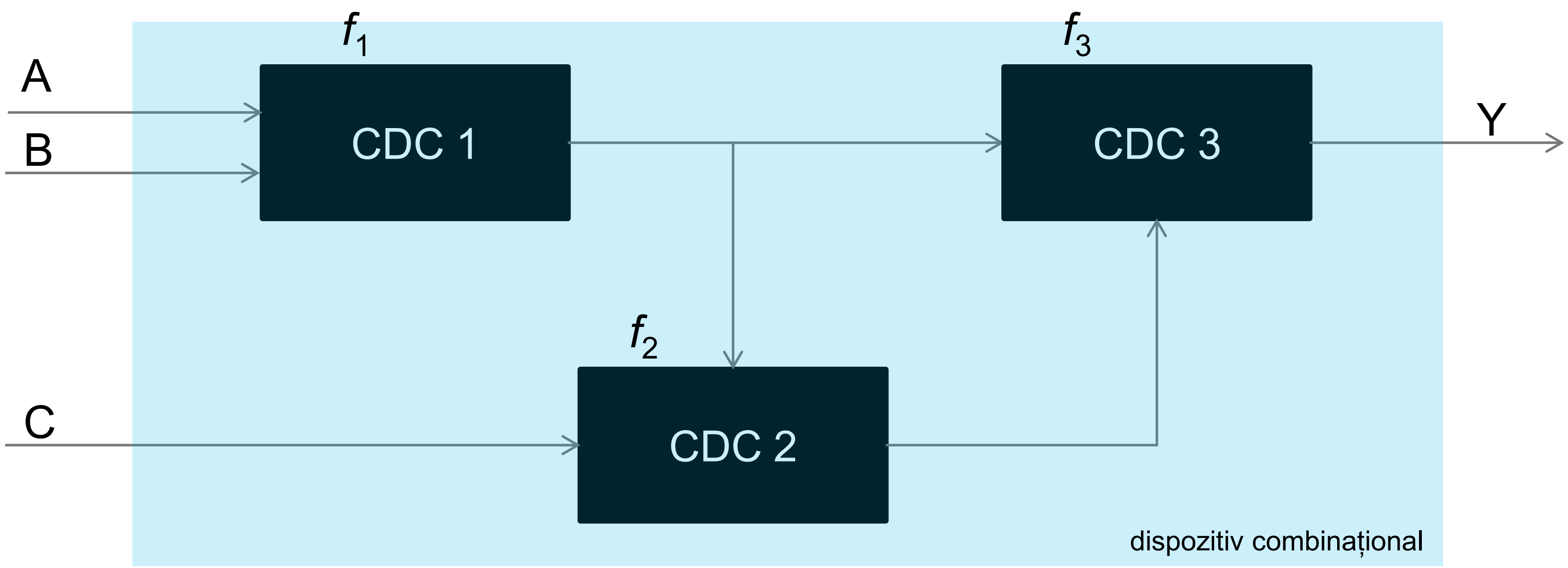
- **dispozitiv combinațional**
  - fiecare element este un circuit combinațional
  - fiecare intrare este conectată la exact o ieșire sau la o constantă
  - nu există niciun ciclu în graful direcțional al dispozitivului



  - timpul total de propagare? $t_{p,total} = t_{p,1} + t_{p,2} + t_{p,3}$ (longest path)

CIRCUITE DIGITALE

- **dispozitiv combinațional**
  - fiecare element este un circuit combinațional
  - fiecare intrare este conectată la exact o ieșire sau la o constantă
  - nu există niciun ciclu în graful direcțional al dispozitivului

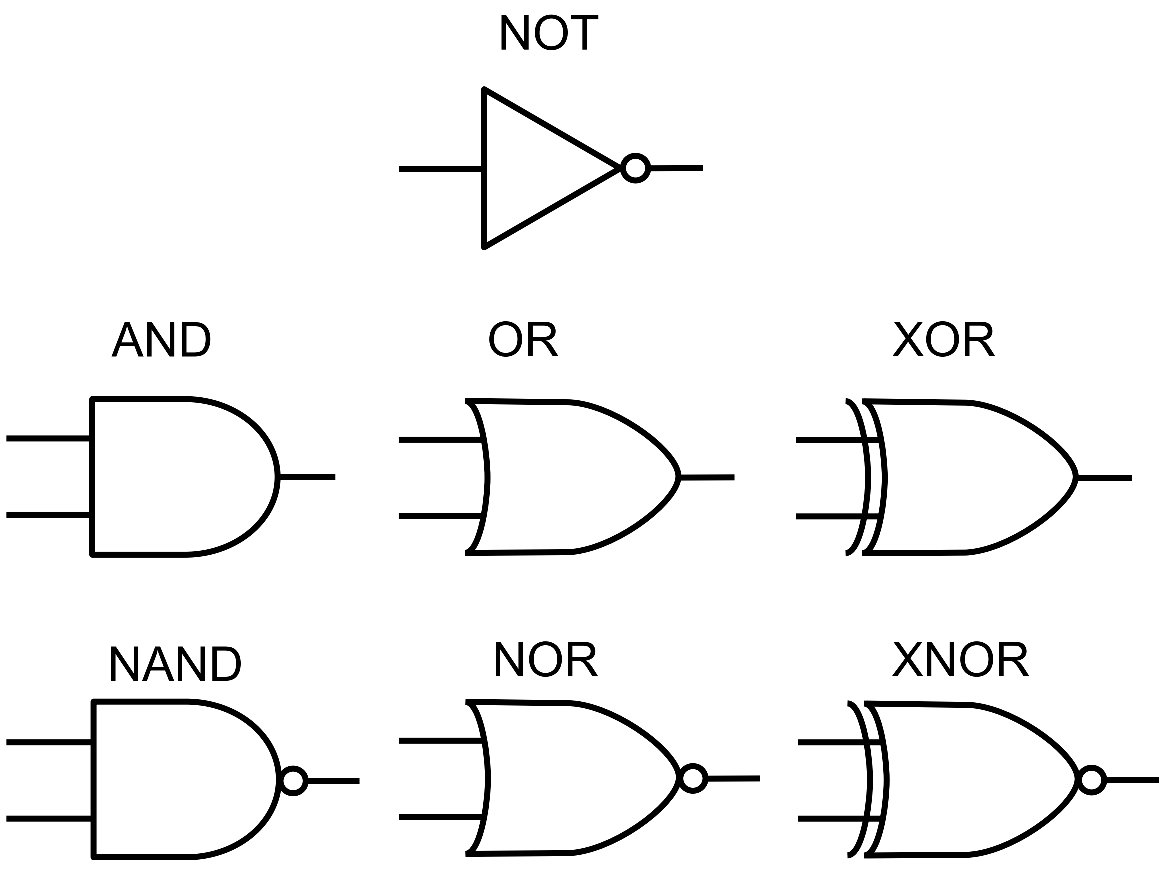


- timpul total de propagare? $t_{p,total} = t_{p,1} + t_{p,2} + t_{p,3}$ (longest path)

  timpul maxim după care avem o ieșire validă dacă avem intrări valide

CIRCUITE DIGITALE

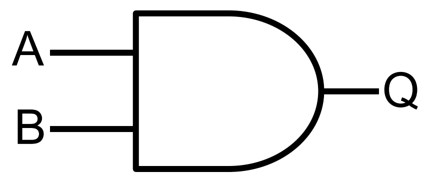
- **circuit digital combinațional**
  - exemple fundamentale

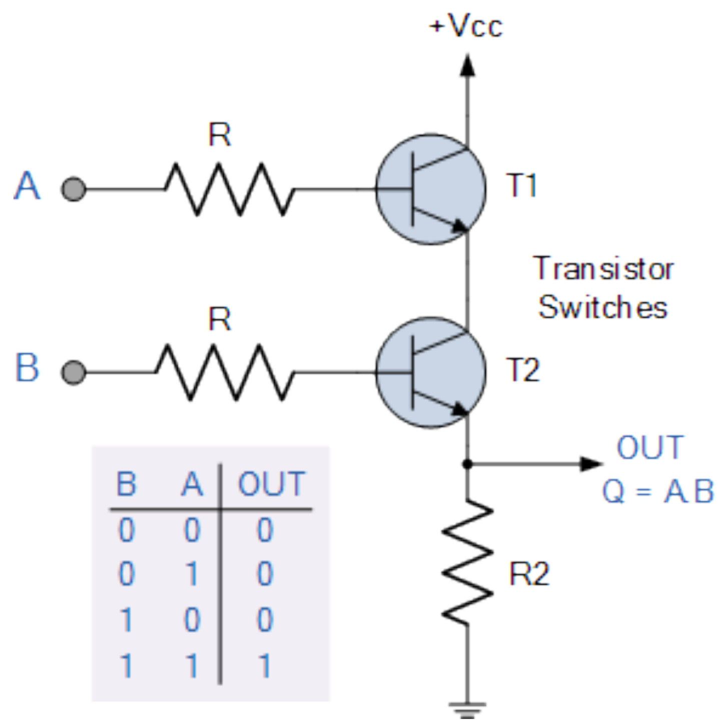


CIRCUITE DIGITALE

- **circuit digital combinațional**
  - exemple fundamentale
  - la baza tuturor se află circuite eletronice bazate de tranzistor

AND



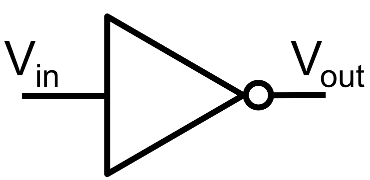


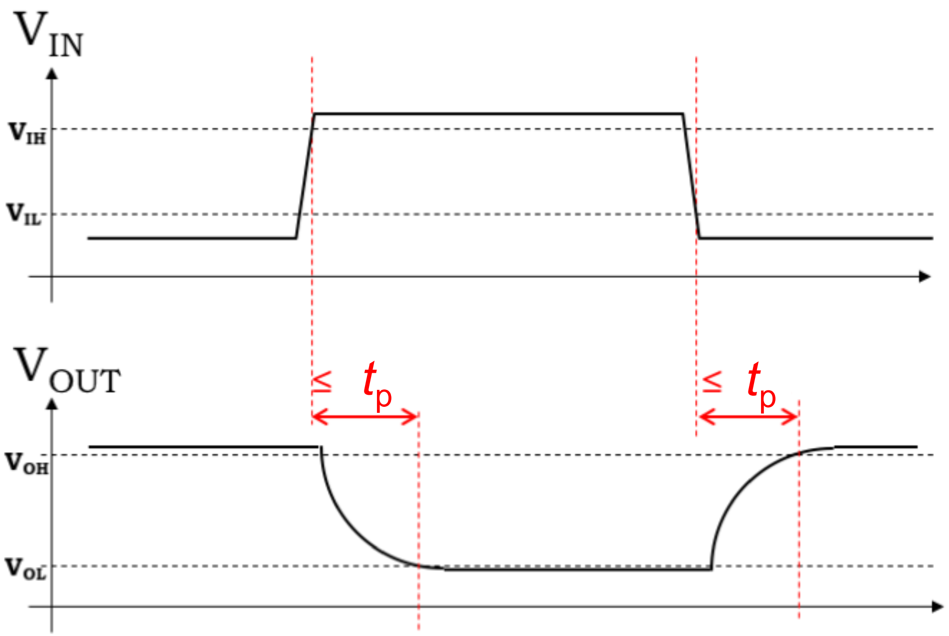
B	A	OUT
0	0	0
0	1	0
1	0	0
1	1	1

https://www.electronics-tutorials.ws/logic/logic_2.html

CIRCUITE DIGITALE

- **circuit digital combinațional**
  - exemple fundamentale: să nu uităm că totul este analogic

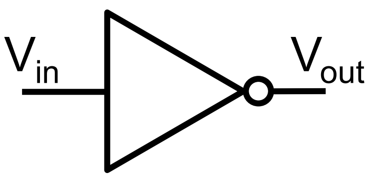


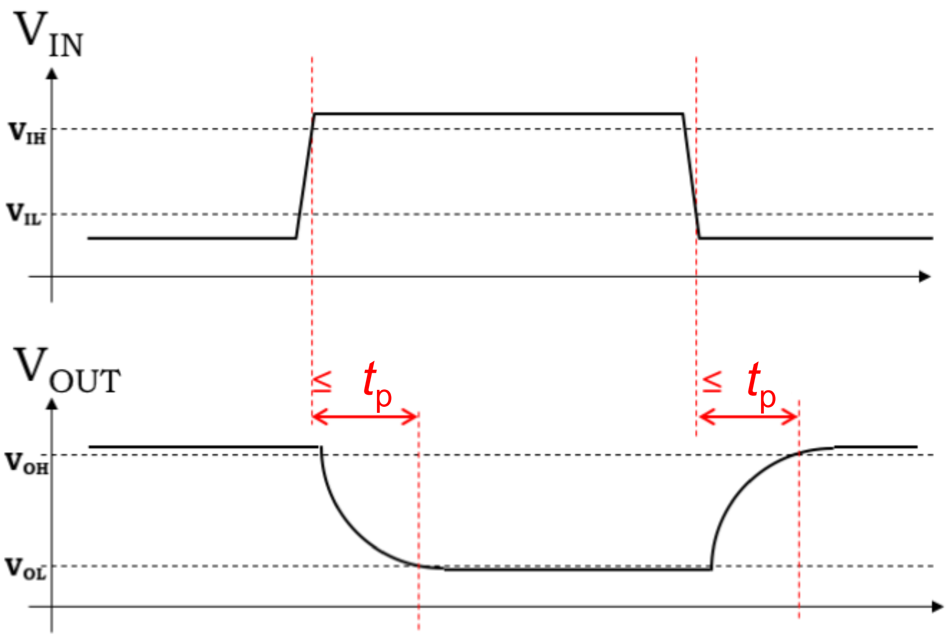


https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-004-computation-structures-spring-2017/c3/c3s1/

CIRCUITE DIGITALE

- **circuit digital combinațional**
  - exemple fundamentale: să nu uităm că totul este analogic



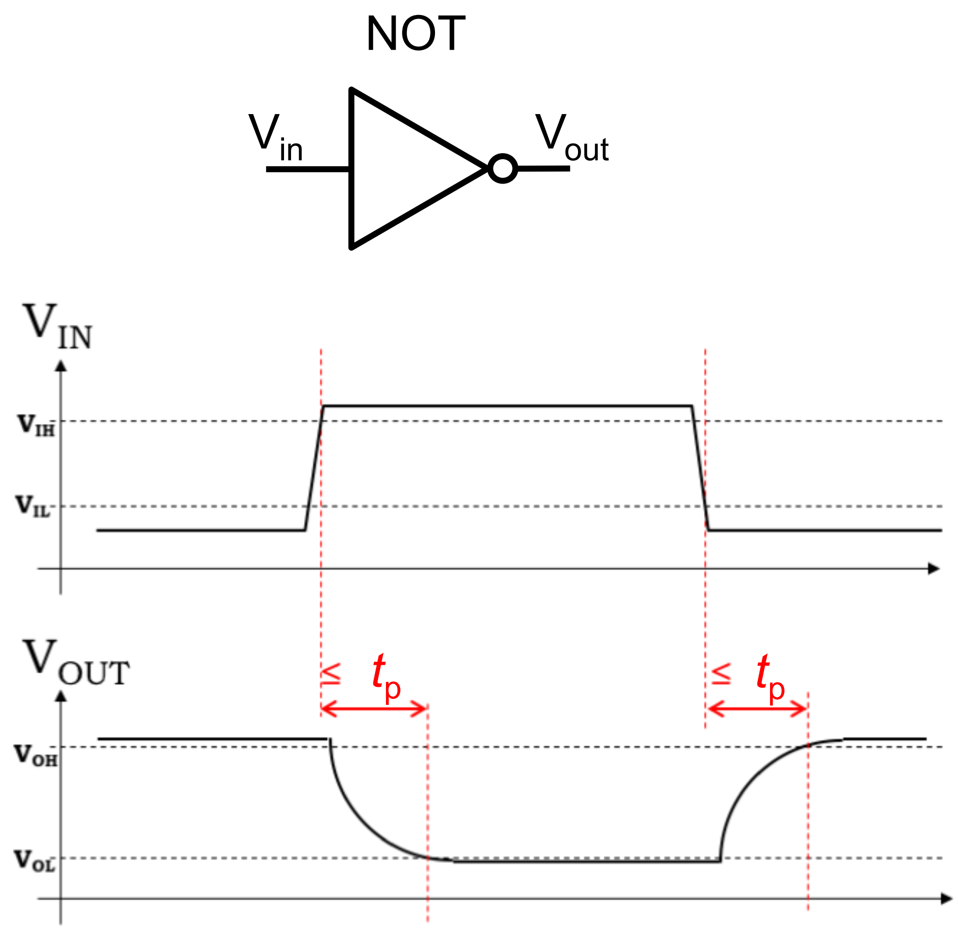


de ce este important acest t_p pentru arhitectura calculatoarelor?

https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-004-computation-structures-spring-2017/c3/c3s1/

CIRCUITE DIGITALE

- **circuit digital combinațional**
  - exemple fundamentale: să nu uităm că totul este analogic



un computer care funcționează la 1GHz trimite comenzi o dată la 1ns

https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-004-computation-structures-spring-2017/c3/c3s1/

ABSTRACTIZAREA DIGITALĂ

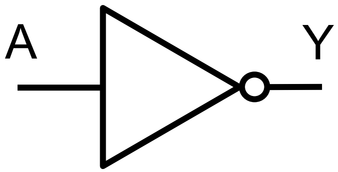
- **două întrebări importante:**
  - de ce folosim semnale digitale în loc de analogice?
  - de ce folosim sistemul binar? ar fi mai avantajos să folosim hex?

ABSTRACTIZAREA DIGITALĂ

- **două întrebări importante:**
  - de ce folosim semnale digitale în loc de analogice?
    - din cauza zgomotului
    - într-un sistem analogic zgomotul se acumulează
    - într-un sistem digital, avem corecțiile de zgomot (avem margini)
  - de ce folosim sistemul binar? ar fi mai avantajos să folosim hex?
    - da, ar fi mai avantajos să folosim hex (e de 4 ori mai avantajos)
    - problema este că în loc de două stări ar trebui acum să avem 16
    - asta înseamnă că trebuie să distingem 16 nivele de voltaj în prezența zgomotului (adică cu tot cu margini de zgomot)
    - probabil 16 nivele e prea mult ... dar probabil 4 nivele ar fi fezabil
    - dacă am avea 4 nivele (adică baza $B = 4$) am fi de două ori mai eficienți

PORȚILE LOGICE DE BAZĂ

- **NOT**

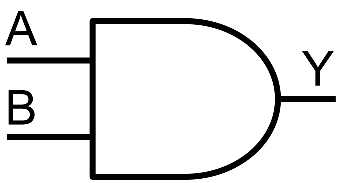


A	Y
0	1
1	0

- **notația:**
  - NOT A (explicit operația)
  - **$\bar{A}$ (A bar, A complement)**
  - ¬A (notația din logică)
  - ~A (not A)
  - - A (minus A)
  - A' (A prime, A complement)
  - !A (bang A)

PORȚILE LOGICE DE BAZĂ

- **AND**

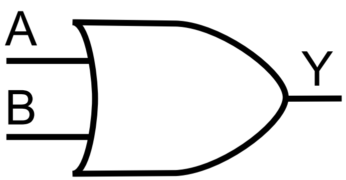


A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

- **notația:**
  - A AND B (explicit operația)
  - A x B (înmulțire)
  - A • B (înmulțire)
  - A * B (înmulțire)
  - A . B (înmulțire)
  - **AB (înmulțire)**
  - A ∧ B (notația din logică)
  - A & B (operația pe biți, C)
  - A && B (operația logică, C)

PORȚILE LOGICE DE BAZĂ

- **OR**

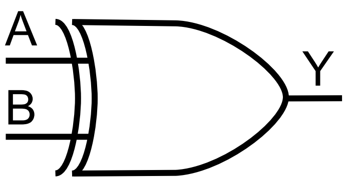


A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

- **notația:**
  - A OR B (explicit operația)
  - **A + B (adunare)**
  - A v B (notația din logică)
  - A | B (operația pe biți, C)
  - A || B (operația logică, C)

PORȚILE LOGICE DE BAZĂ

- **XOR**

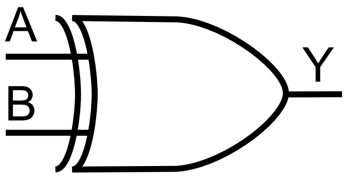


A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

- **notația:**
  - A XOR B (explicit operația)
  - **A ⊕ B (adunare XOR)**
  - A ^ B (operația pe biți, C)
  - A ^^ B (există așa ceva ???)

PORȚILE LOGICE DE BAZĂ

- **XOR**



A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

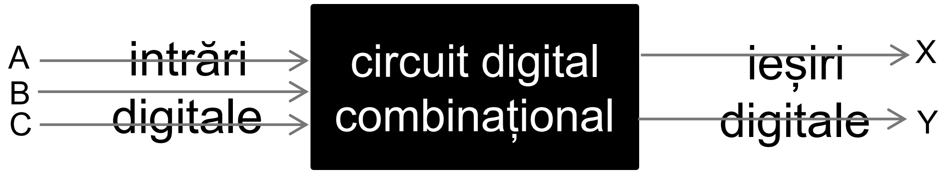
- **notația:**
  - A XOR B (explicit operația)
  - **A ⊕ B (adunare XOR)**
  - A ^ B (operația pe biți, C)
  - A != B (operația logică, C)

ALGEBRĂ BOOLEANĂ

- avețі un curs de Logică în acest semestru
- deci, din acest moment presupun că logica/algebra booleană este cunoscută de voi
- nu vom repeta materia de la logică, dar dacă este ceva foarte important vom trece rapid în revistă conceptele

CIRCUIT DIGITAL COMBINAȚIONAL

- **circuit digital combinațional**



A	B	C	X	Y
0	0	0	0	1
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	0

- **o expresie booleană care conține regulile din tabel?**
  - $X = \overline{A}\,\overline{B}C + \overline{A}B\overline{C} + A\overline{B}\,\overline{C} + ABC$
  - Y = ... (execițiu pentru voi)

forma normală, suma de produse

CIRCUIT DIGITAL COMBINAȚIONAL

- **tabel de adevăr**

A	B	C	X	Y
0	0	0	0	1
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	0

- **expresia booleană echivalentă**
  - $X = \overline{A}\overline{B}C + \overline{A}B\overline{C} + A\overline{B}\overline{C} + ABC$

- **concluzia: NOT, AND și OR sunt universale (pot implementa orice circuit combinațional)**
- **de ce lipsește XOR?**

CIRCUIT DIGITAL COMBINAȚIONAL

- **tabel de adevăr**

A	B	C	X	Y
0	0	0	0	1
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	0

- **expresia booleană echivalentă**
  - $X = \overline{A}\overline{B}C + \overline{A}B\overline{C} + A\overline{B}\overline{C} + ABC$

- **concluzia: NOT, AND și OR sunt universale (pot implementa orice circuit combinațional)**
- **de ce lipsește XOR? $A \oplus B = \overline{A}B + A\overline{B}$**

A	B	A ⊕ B
0	0	0
0	1	1
1	0	1
1	1	0

CIRCUIT DIGITAL COMBINAȚIONAL

- **tabel de adevăr**

A	B	C	X	Y
0	0	0	0	1
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	0

- **expresia booleană echivalentă**
  - $X = \overline{A}\overline{B}C + \overline{A}B\overline{C} + A\overline{B}\overline{C} + ABC$

- **o potențială problemă: dacă avem N intrări, expresia pentru X depinde de un număr (potențial) mare de sume de produse**
- **care e numărul maxim de termeni în sumele din X?**

CIRCUIT DIGITAL COMBINAȚIONAL

- **tabel de adevăr**

A	B	C	X	Y
0	0	0	0	1
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	0

- **expresia booleană echivalentă**
  - $X = \overline{A}\overline{B}C + \overline{A}B\overline{C} + A\overline{B}\overline{C} + ABC$

- **o potențială problemă: dacă avem N intrări, expresia pentru X depinde de un număr (potențial) mare de sume de produse**
- **care e numărul maxim de termeni în sumele din X? $\frac{1}{2}2^N$**

CIRCUIT DIGITAL COMBINAȚIONAL

- **tabel de adevăr**

A	B	C	X	Y
0	0	0	0	1
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	0

- **expresia booleană echivalentă**
  - $X = \overline{A}\,\overline{B}C + \overline{A}B\overline{C} + A\overline{B}\,\overline{C} + ABC$

- **o potențială problemă: dacă avem N intrări, expresia pentru X depinde de un număr (potențial) mare de sume de produse**
- **soluția generală: vrem reprezentări minime**

CIRCUIT DIGITAL COMBINAȚIONAL

- **tabel de adevăr**

A	B	C	X	Y
0	0	0	0	1
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	0

- **expresia booleană echivalentă**
  - $X = \overline{A}\overline{B}C + \overline{A}B\overline{C} + A\overline{B}\overline{C} + ABC$

- **reprezentarea minimă a lui X?**

CIRCUIT DIGITAL COMBINAȚIONAL

- **tabel de adevăr**

A	B	C	X	Y
0	0	0	0	1
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	0

- **expresia booleană echivalentă**
  - $X = \overline{A}\,\overline{B}C + \overline{A}B\overline{C} + A\overline{B}\,\overline{C} + ABC$

    $=$

CIRCUIT DIGITAL COMBINAȚIONAL

- **tabel de adevăr**

A	B	C	X	Y
0	0	0	0	1
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	0

- **expresia booleană echivalentă**
  - $X = \overline{A}\overline{B}C + \overline{A}B\overline{C} + A\overline{B}\overline{C} + ABC$

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

    $=$

CIRCUIT DIGITAL COMBINAȚIONAL

- **tabel de adevăr**

A	B	C	X	Y
0	0	0	0	1
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	0

- **expresia booleană echivalentă**
  - $X = \overline{A}\,\overline{B}C + \overline{A}B\overline{C} + A\overline{B}\,\overline{C} + ABC$
    $= \overline{A}\,(\overline{B}C + B\overline{C}) + A\,(\overline{B}\,\overline{C} + BC)$
    $= \overline{A}\,(B \oplus C) + A\,\overline{(B \oplus C)}$

CIRCUIT DIGITAL COMBINAȚIONAL

- **tabel de adevăr**

A	B	C	X	Y
0	0	0	0	1
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	0

- **expresia booleană echivalentă**
  - $X = \overline{A}\overline{B}C + \overline{A}B\overline{C} + A\overline{B}\overline{C} + ABC$

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

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

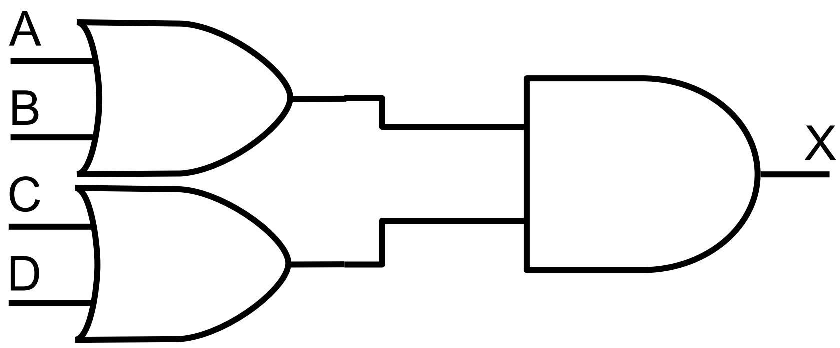
    $= A \oplus B \oplus C$

CIRCUIT DIGITAL COMBINAȚIONAL

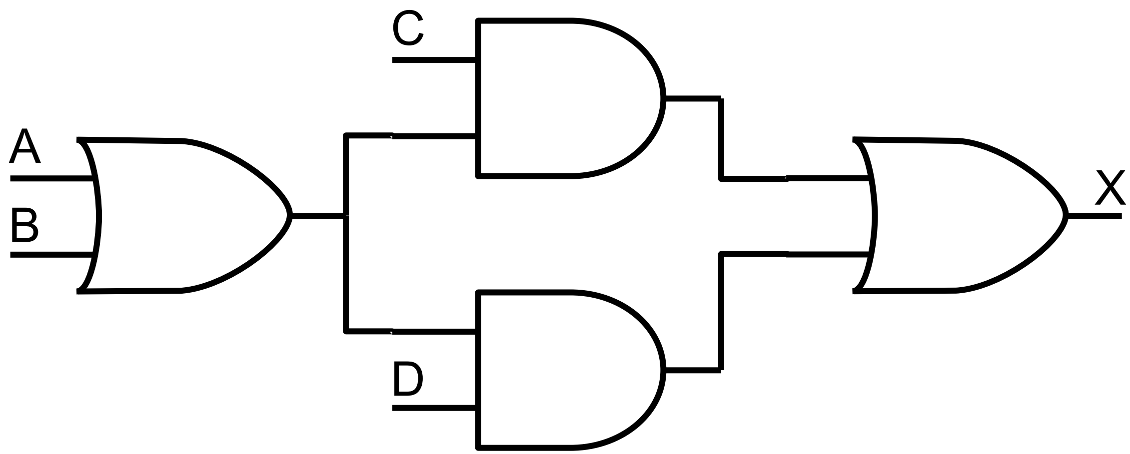
- **considerăm acum**
  - $X = AC + BC + AD + BD$

    $= (A + B)C + (A + B)D$

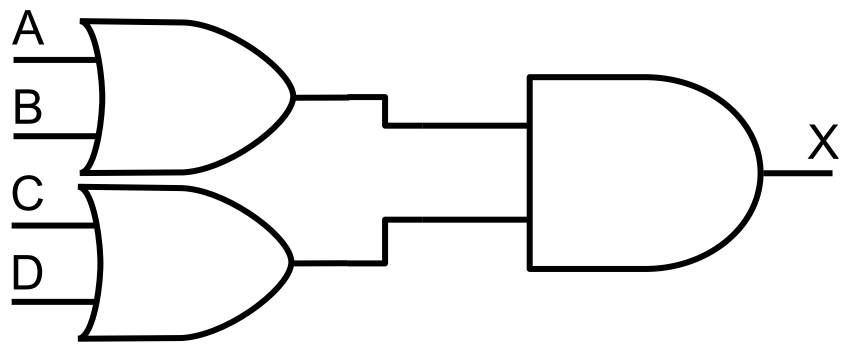
    $= (A + B)(C + D)$



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 ...

CIRCUITE MAI COMPLEXE

- **concluzia importantă:**

- **tot ce facem pe un sistem de calcul trebuie să se reducă la circuite care sunt porți logice**

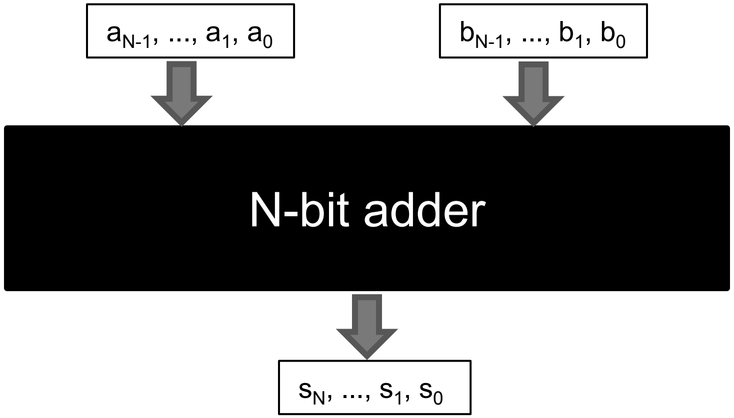
- **altceva nu există la acest nivel**

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

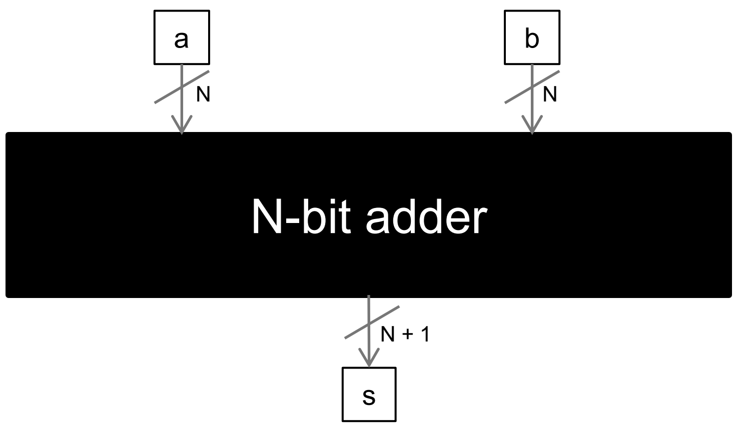
- **circuit de adunare**
  - se dau *a* și *b* pe *N* biți
  - vrem să calculăm *s* = *a* + *b*
    - pe câți biți este *s*?
    - *N+1* biți



- 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:

	1	1	1	0	*c*
	1	1	1	0	*a*
	0	1	1	1	*b*
1	0	1	0	1	*s*

CIRCUITE MAI COMPLEXE

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

1	1	1	0		c
	1	1	1	0	a
	0	1	1	1	b
1	0	1	0	1	s

CIRCUITE MAI COMPLEXE

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

1	1	1	0		c
	1	1	1	0	a
	0	1	1	1	b
1	0	1	0	1	s

CIRCUITE MAI COMPLEXE

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

1	1	1	0		c
	1	1	1	0	a
	0	1	1	1	b
1	0	1	0	1	s

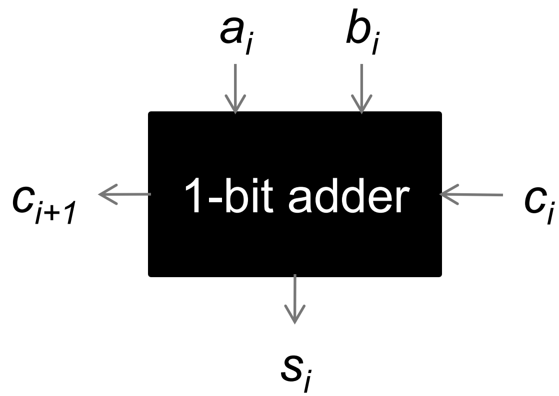
CIRCUITE MAI COMPLEXE

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

1	1	1	0		c
	1	1	1	0	a
	0	1	1	1	b
1	0	1	0	1	s

CIRCUITE MAI COMPLEXE

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



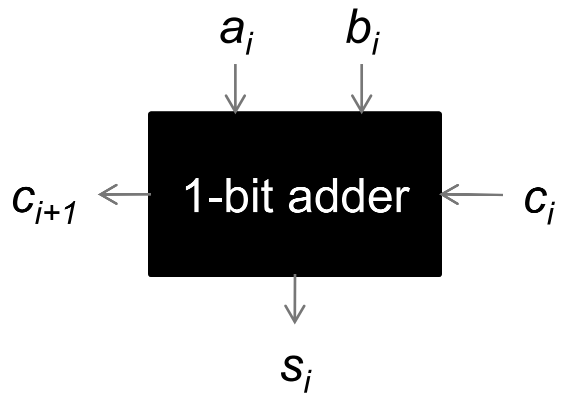
s_i = ?

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



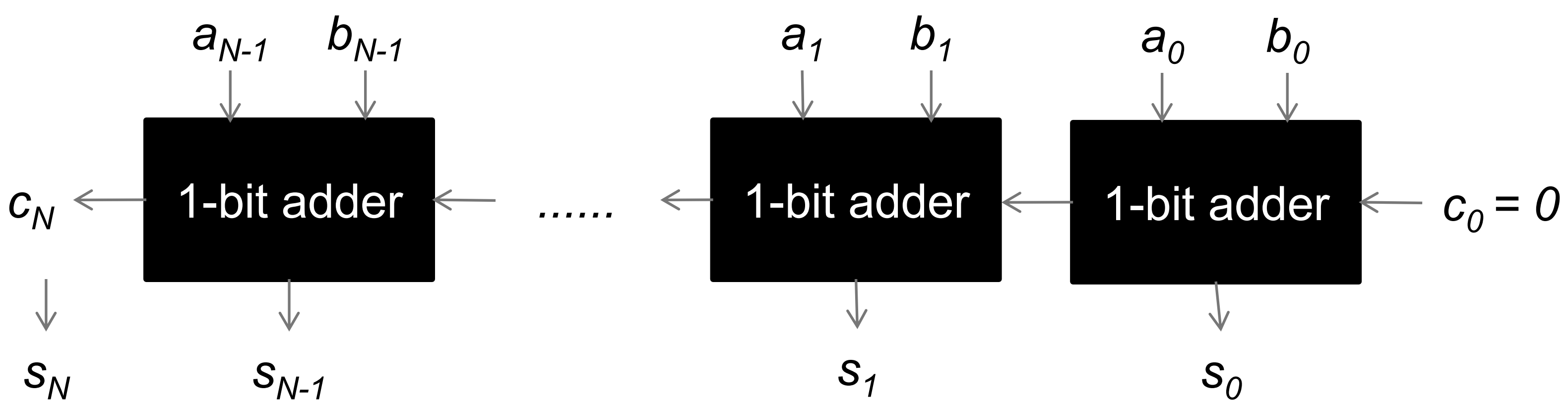
$s_i = a_i \oplus b_i \oplus c_{in}$

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

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

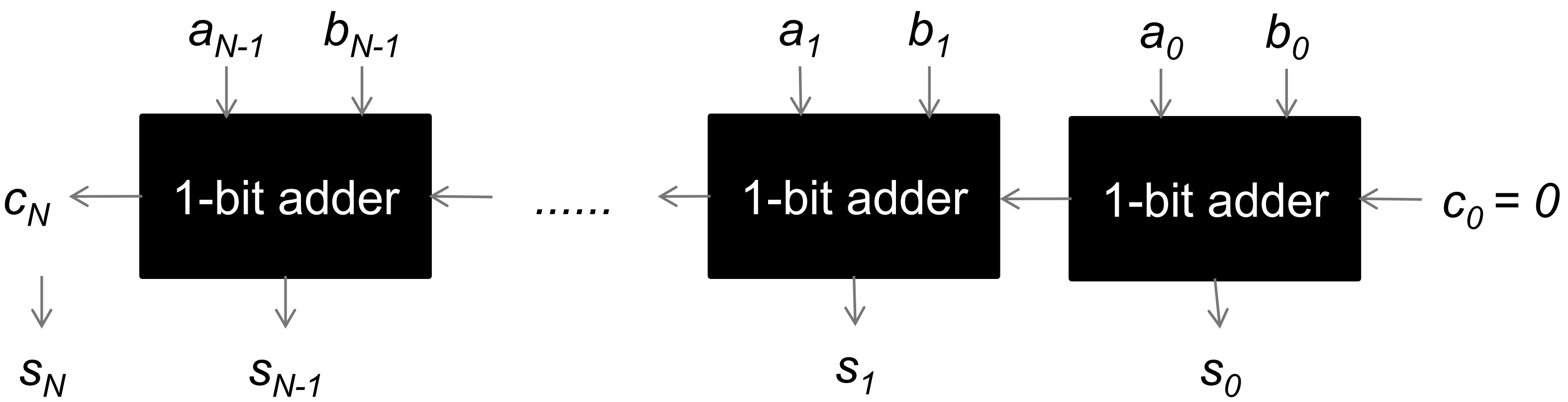


$$s_i = a_i \oplus b_i \oplus c_{\text{in}}$$

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

CIRCUITE MAI COMPLEXE

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



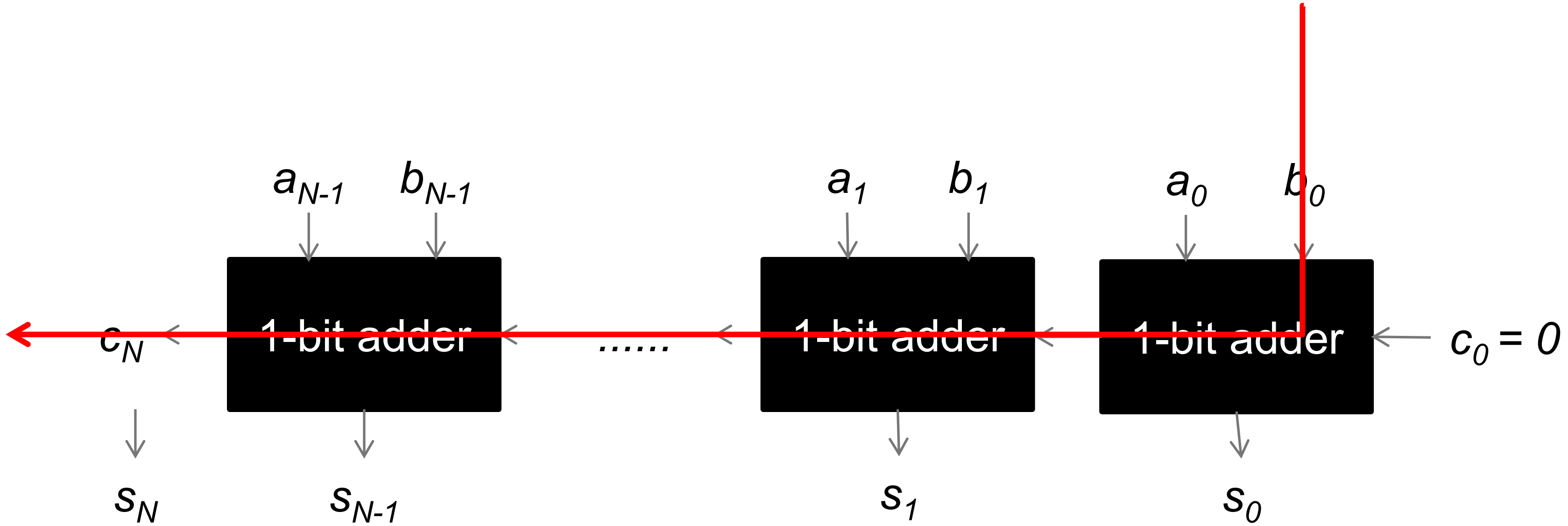
$$s_i = a_i \oplus b_i \oplus c_{\text{in}}$$

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

  - 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



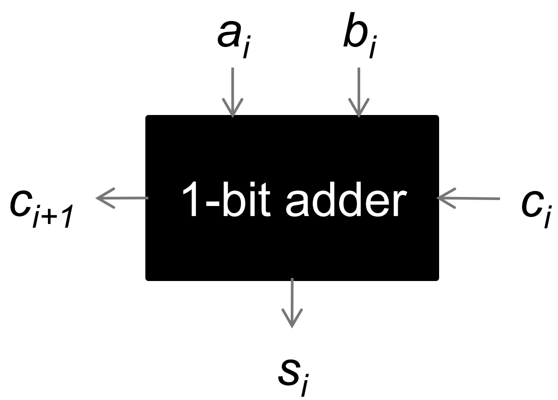
$s_i = a_i \oplus b_i \oplus c_{\text{in}}$

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

- 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



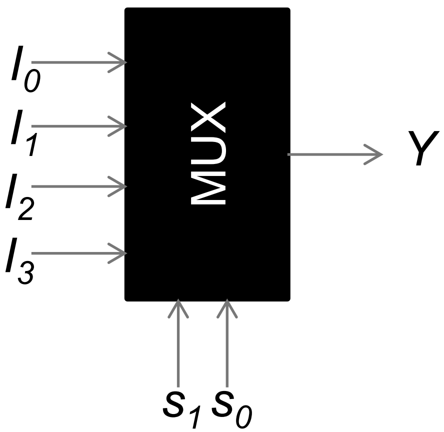
$$s_i = a_i \oplus b_i \oplus c_{in}$$

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

- 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*

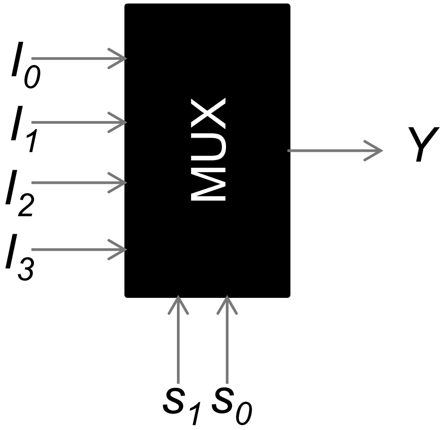


s_1	s_0	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

  - 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*

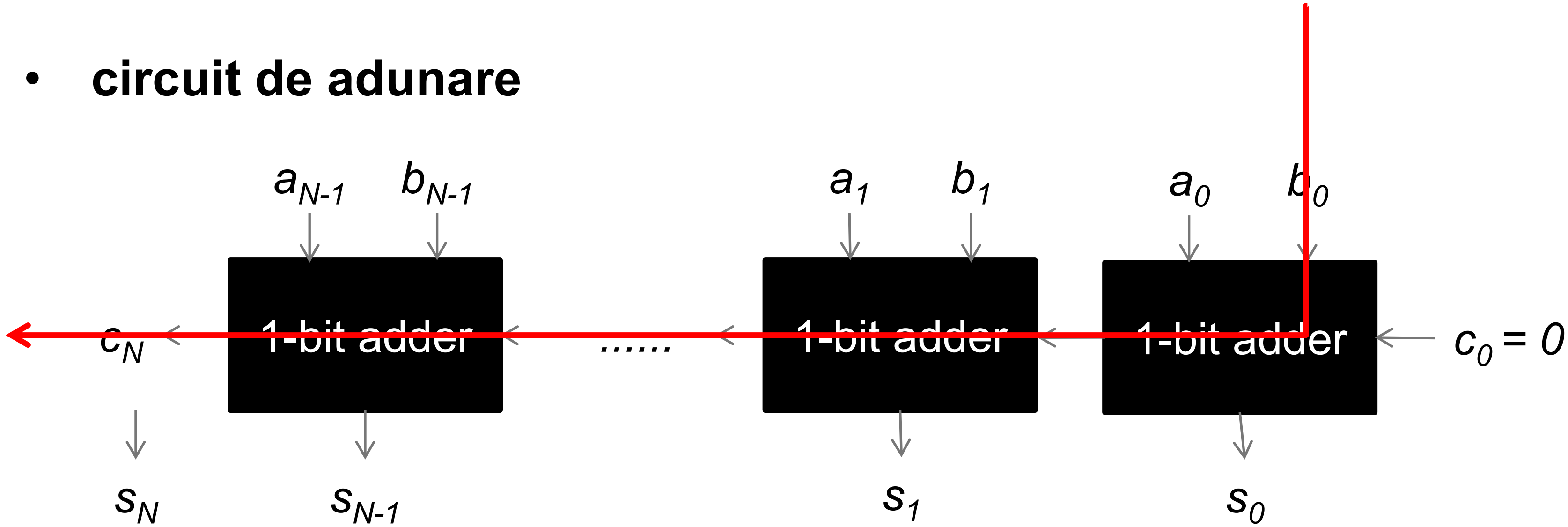


s_1	s_0	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

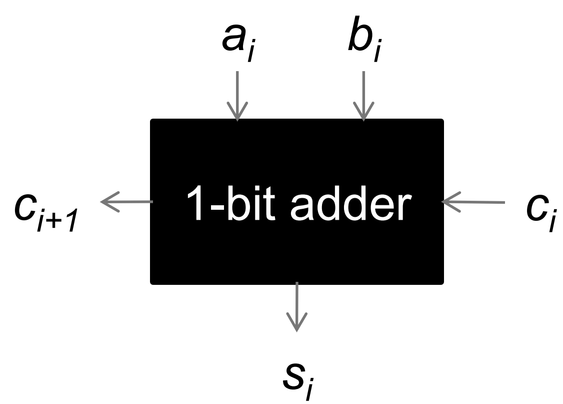
  - de ce e folositor acest circuit?
    - implementare hardware pentru “if” și “case”
    - implementare hardware pentru operații shift
    - vom vedea la seminar

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

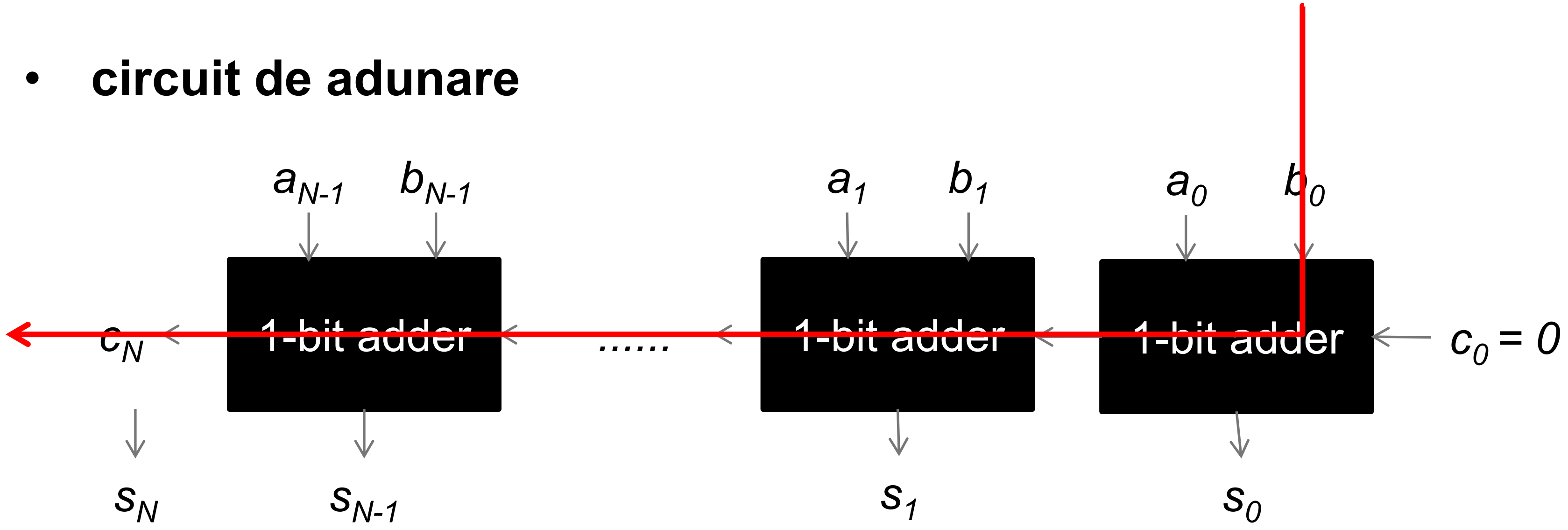


$$s_i = a_i \oplus b_i \oplus c_{in}$$

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

CIRCUITE MAI COMPLEXE

- **circuit de adunare**

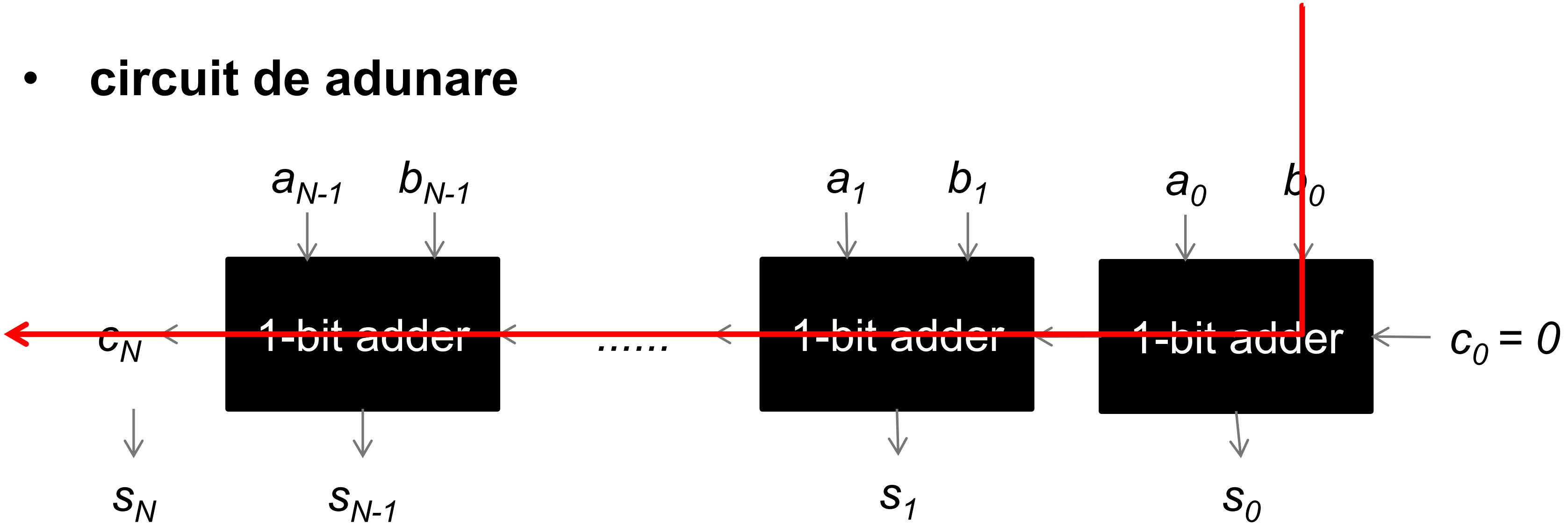


- cât este timpul de propagare în acest caz?
- $N\ t_P$

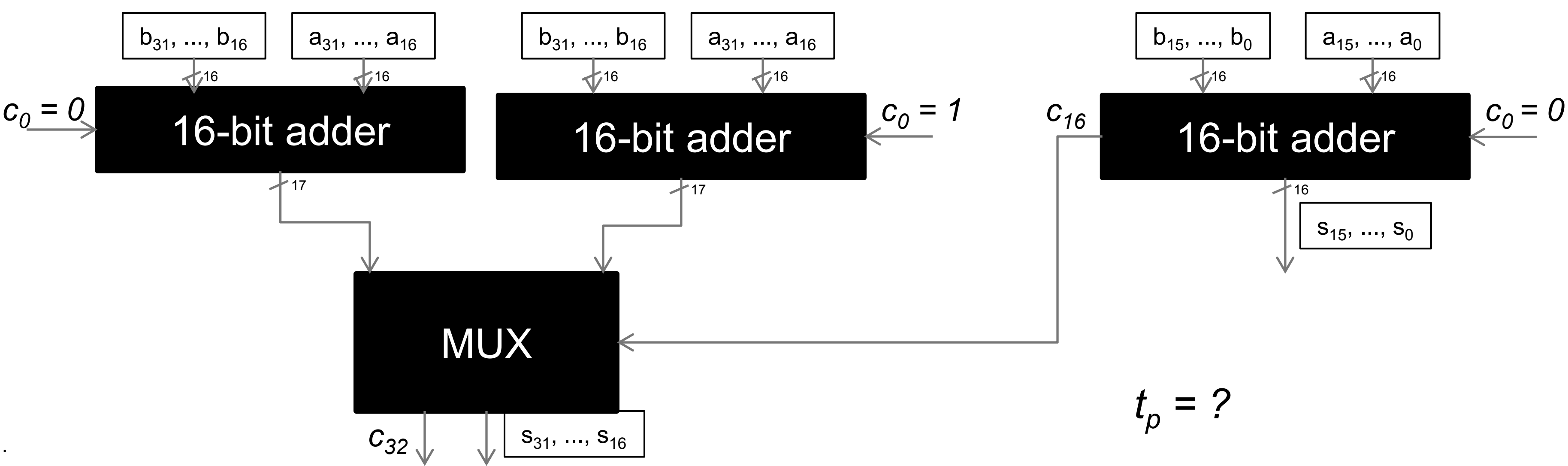
- ce putem face pentru a îmbunătăţii timpul de propagare?

CIRCUITE MAI COMPLEXE

- **circuit de adunare**



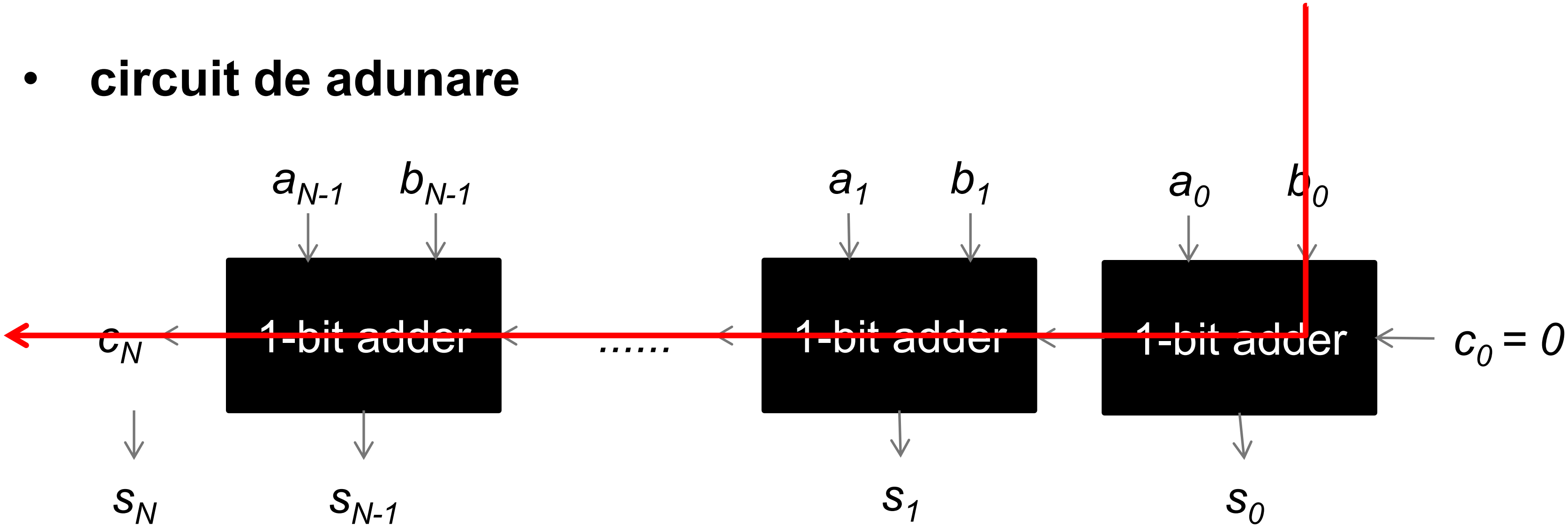
- cât este timpul de propagare în acest caz?
- $N\ t_P$



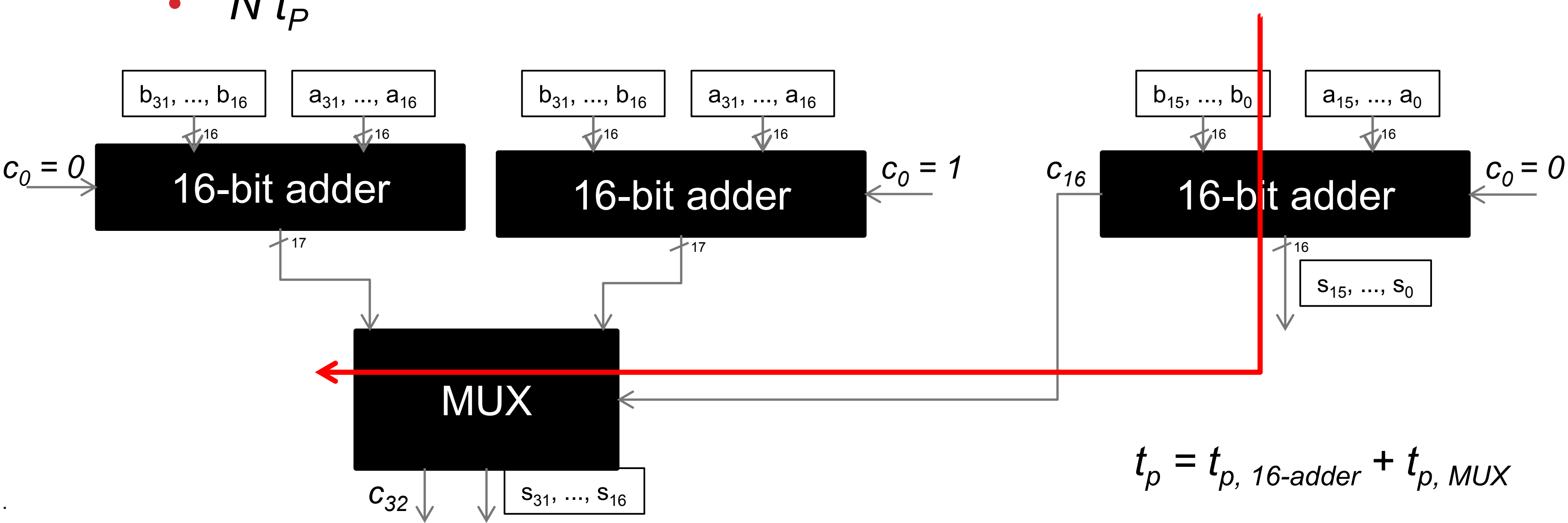
$t_p = ?$

CIRCUITE MAI COMPLEXE

- **circuit de adunare**



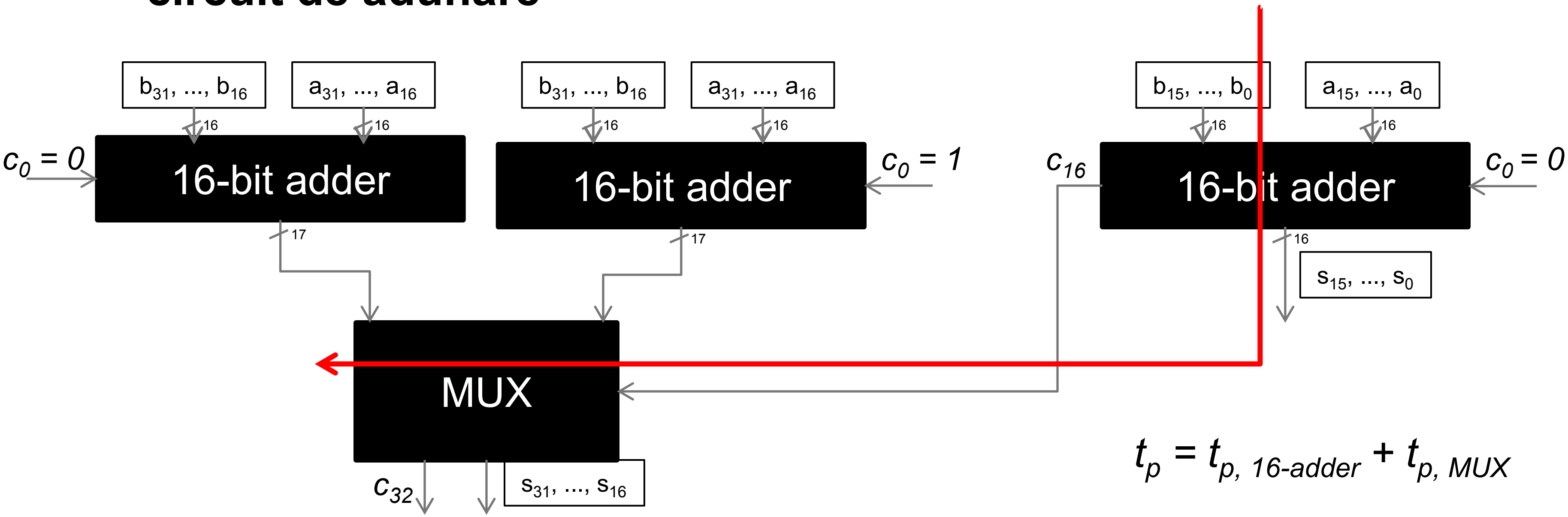
- cât este timpul de propagare în acest caz?
- $N\ t_P$



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

CIRCUITE MAI COMPLEXE

- **circuit de adunare**

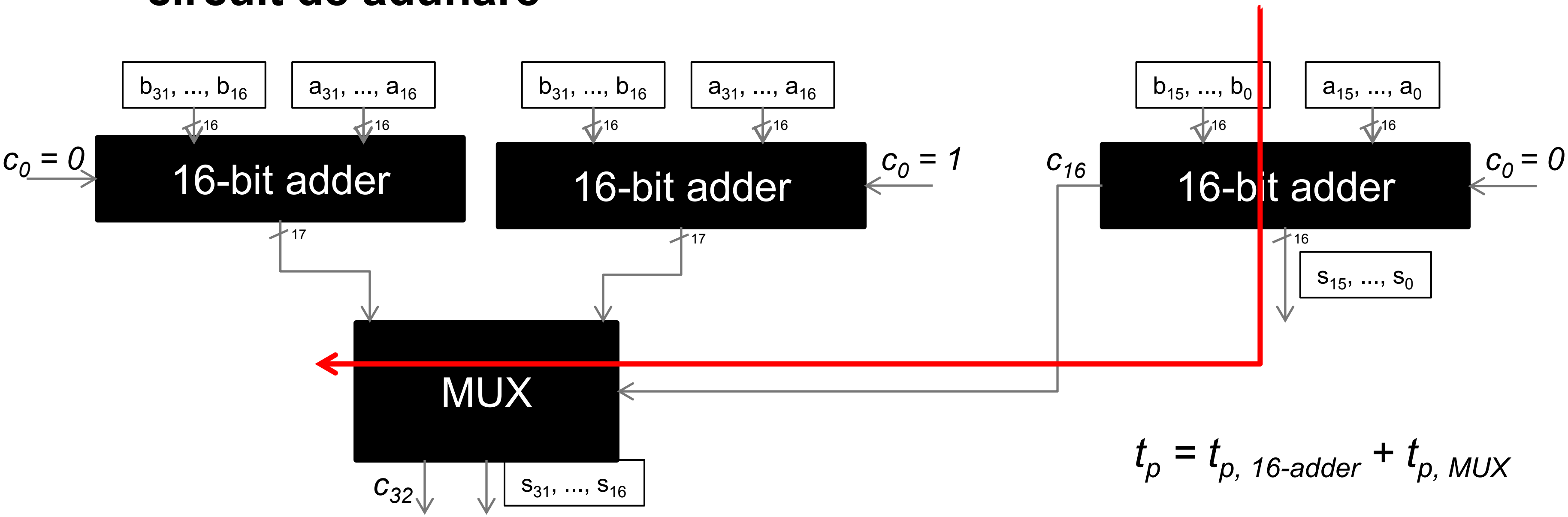


$t_p = t_{p,\ 16\text{-adder}} + t_{p,\ MUX}$

- 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**



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

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

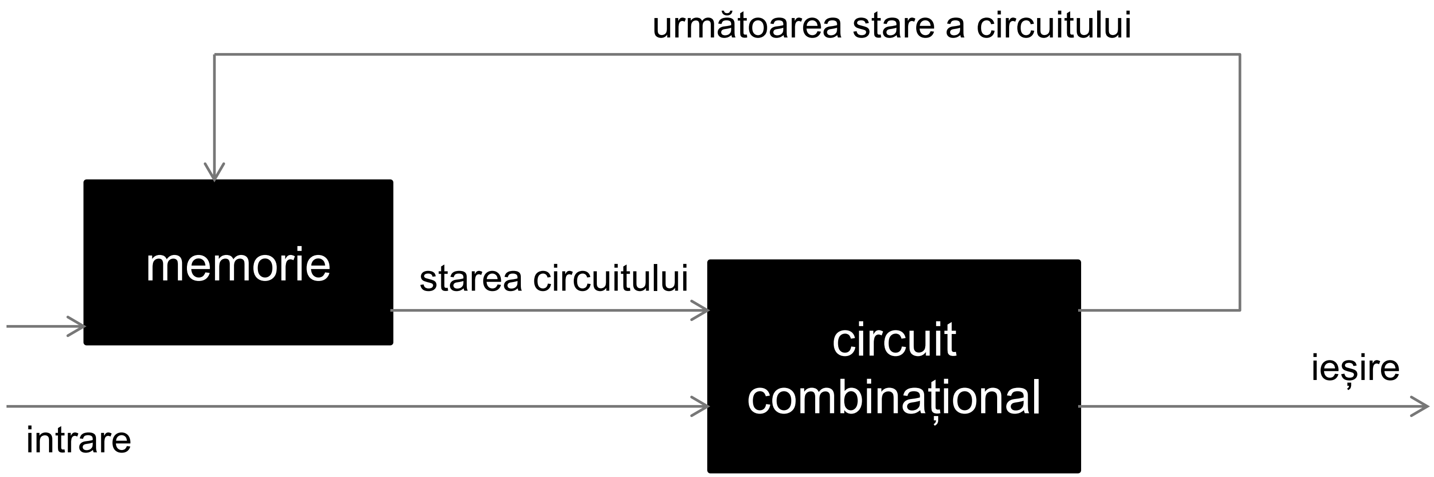
CIRCUITE SECVENȚIALE

- **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

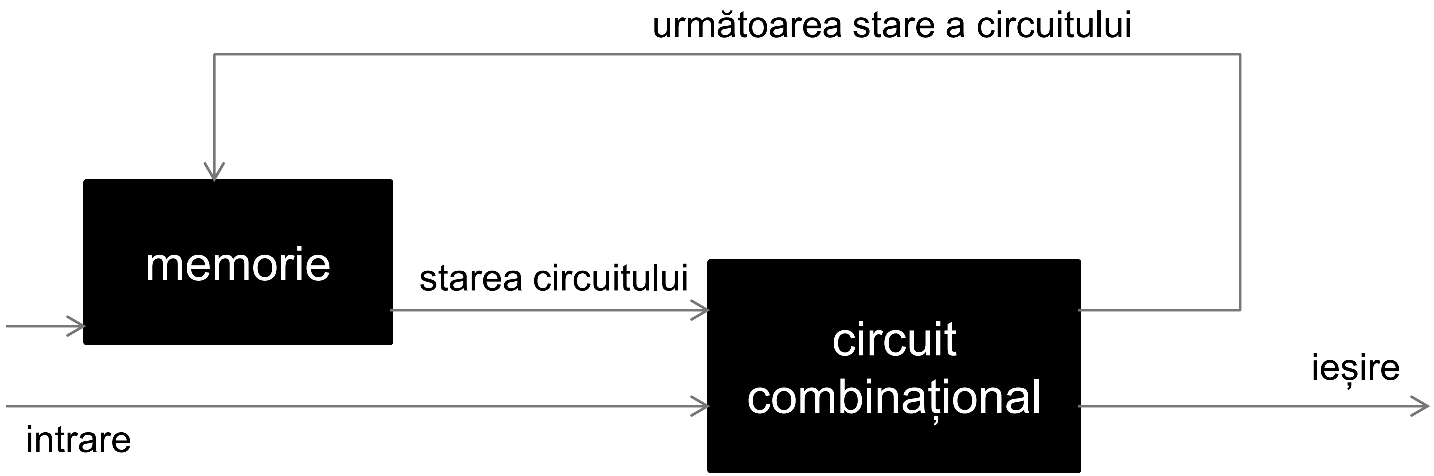
- **avem nevoie să introducem elemente de tip “memorie”**



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

CIRCUITE SECVENȚIALE

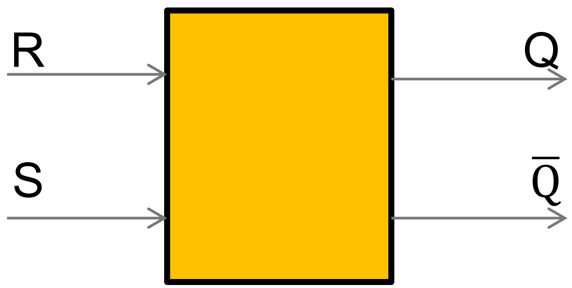
- **avem nevoie să introducem elemente de tip “memorie”**

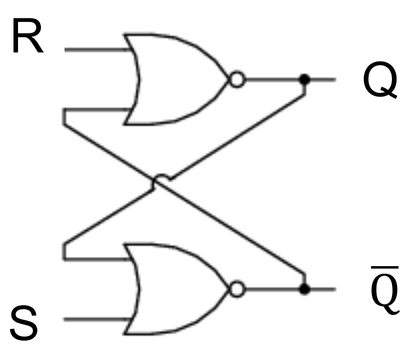


- **am vorbit până acum că biții pe care îi reprezentăm sunt voltaj**
- **dar energia electrică este dificil de stocat (în timp)**
  - fenomen de scurgere (leakage)
- **dacă vrem să memorăm ceva, trebuie să facem un refresh din când în când pentru a actualiza nivelul de energie electrică**

CIRCUITE SECVENȚIALE

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

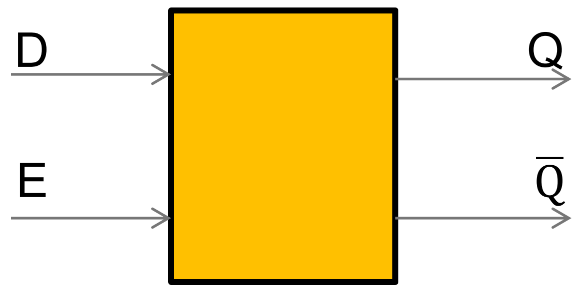


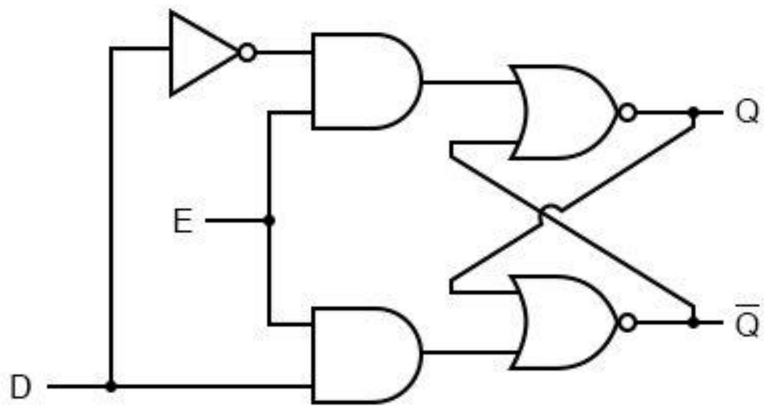


S	R	Q	$\overline{Q}$	
0	0	latch	latch	nu se schimbă nimic
0	1	0	1	aici punem “0” în memorie
1	0	1	0	aici punem “1” în memorie
1	1	0	0	stare invalidă

CIRCUITE SECVENȚIALE

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



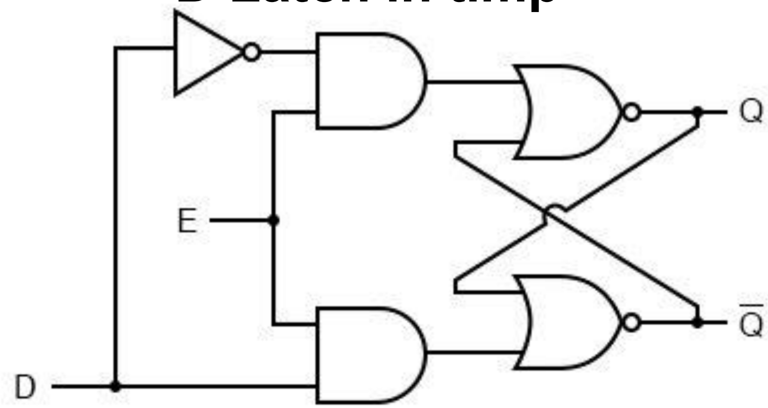


E	D	Q	$\overline{Q}$	
0	0	latch	latch	nu se schimbă nimic
0	1	latch	latch	nu se schimbă nimic
1	0	0	1	aici punem “0” în memorie
1	1	1	0	aici punem “1” în memorie

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

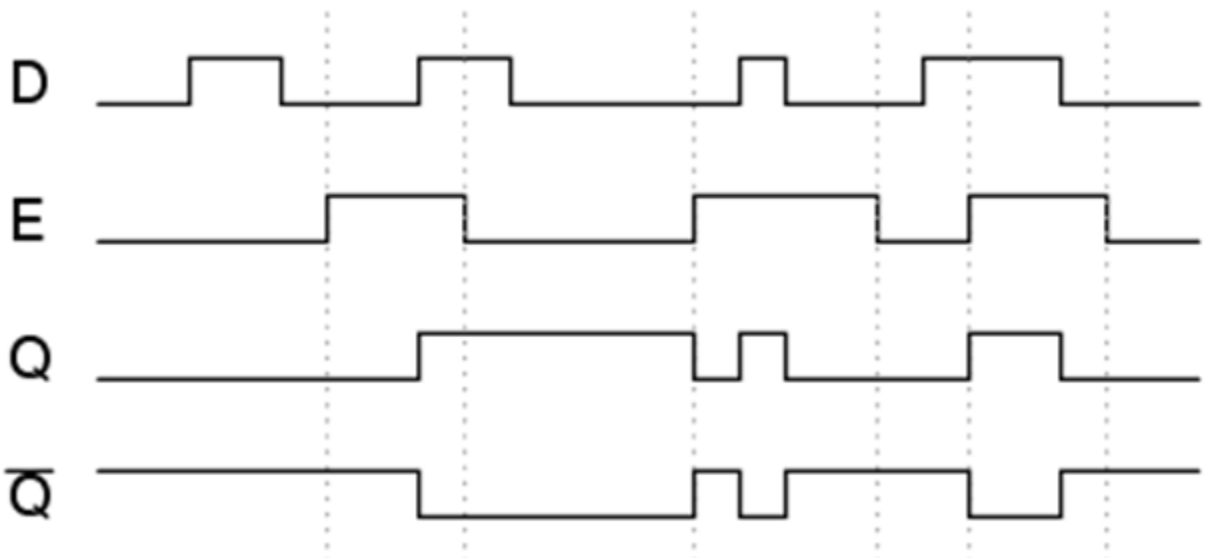
CIRCUITE SECVENȚIALE

- **D Latch în timp**



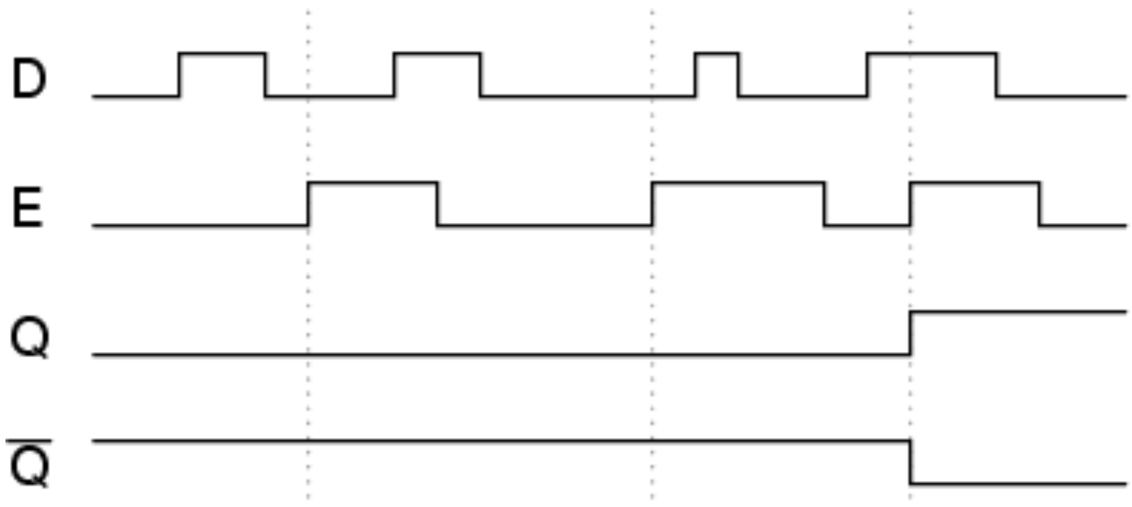
E	D	Q	$\overline{Q}$	
0	0	latch	latch	nu se schimbă nimic
0	1	latch	latch	nu se schimbă nimic
1	0	0	1	aici punem “0” în memorie
1	1	1	0	aici punem “1” în memorie

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



CIRCUITE SECVENȚIALE

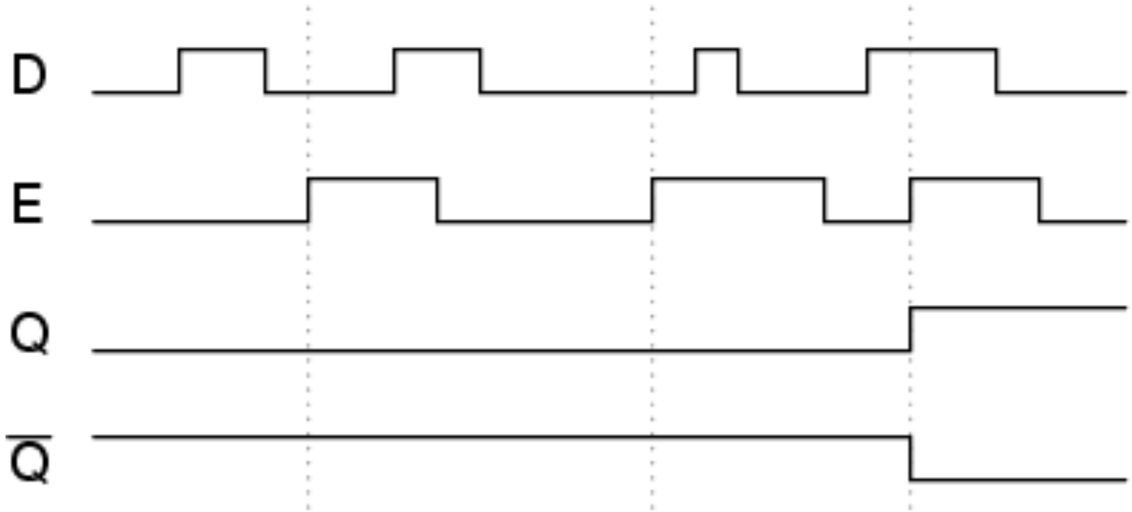
- **D Flip-Flop**
  - D Latch e bun, dar vrem să sincronizăm mai multe dispozitive
  - vrem ca activarea să se facă cand E crește, nu când E e activ



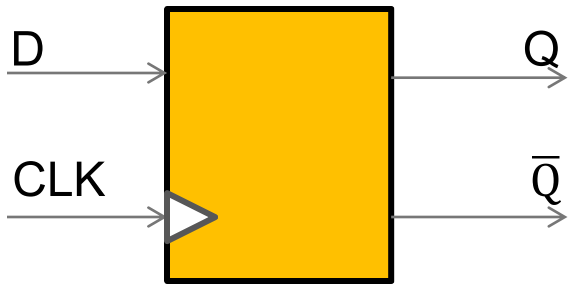
  - E devine clock – adică ceasul sistemului
    - la un interval fix de timp, sistemul face ceva

CIRCUITE SECVENȚIALE

- **D Flip-Flop**



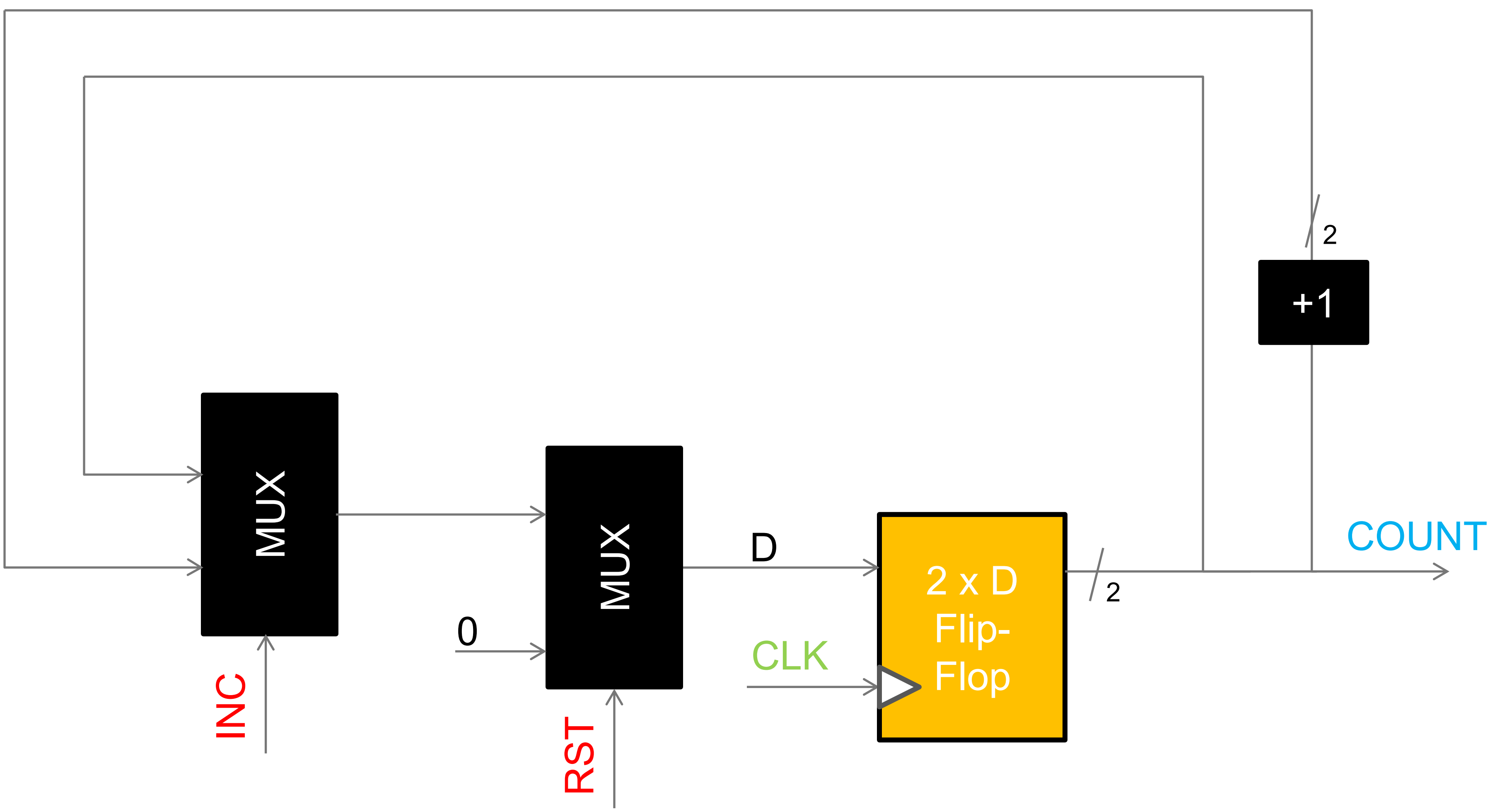
- E devine clock – adică ceasul sistemului
    - la un interval fix de timp (un ciclu), sistemul face ceva



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

CIRCUITE SECVENȚIALE

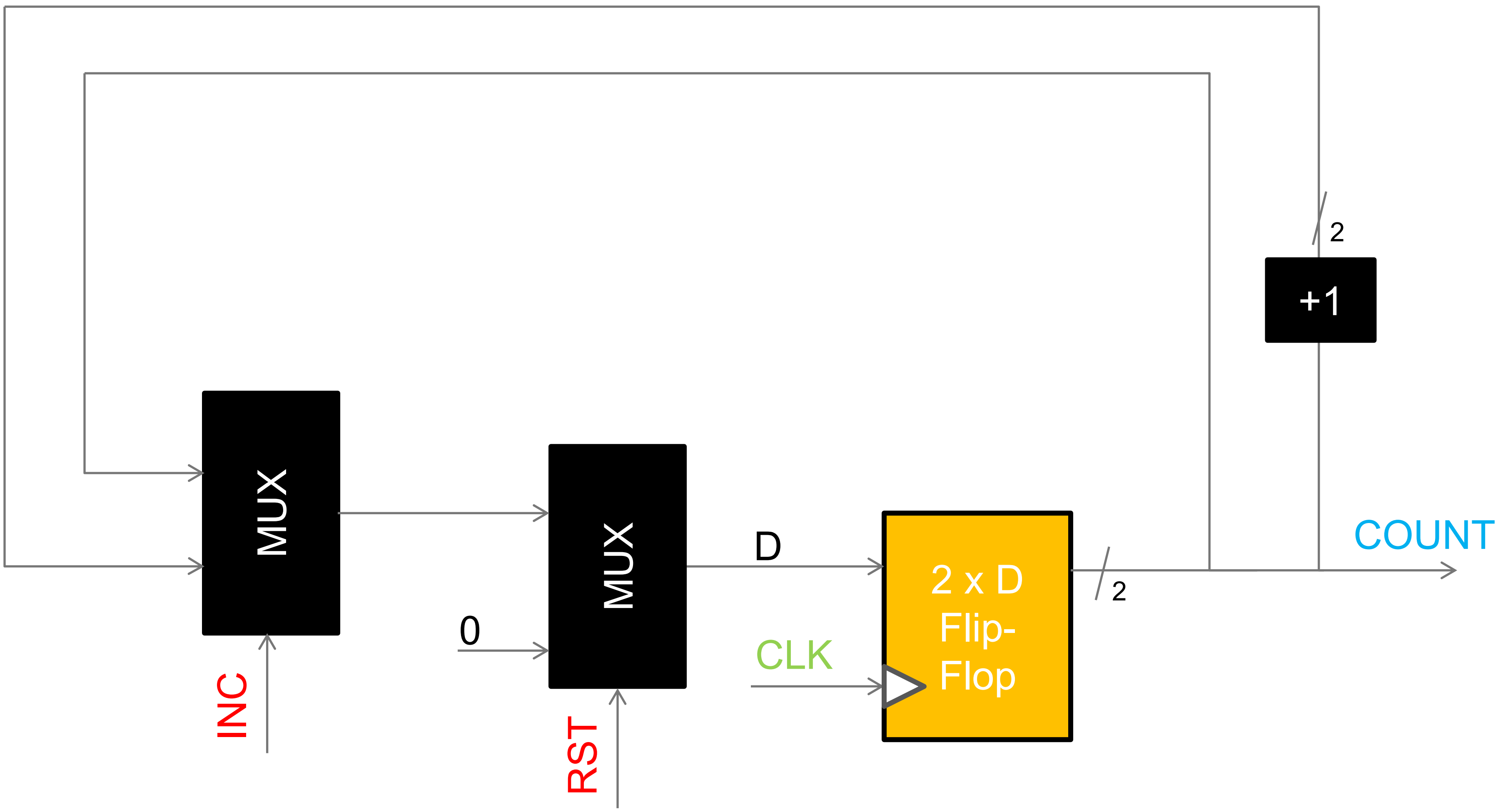
- **un exemplu**



ce face acest circuit?

CIRCUITE SECVENȚIALE

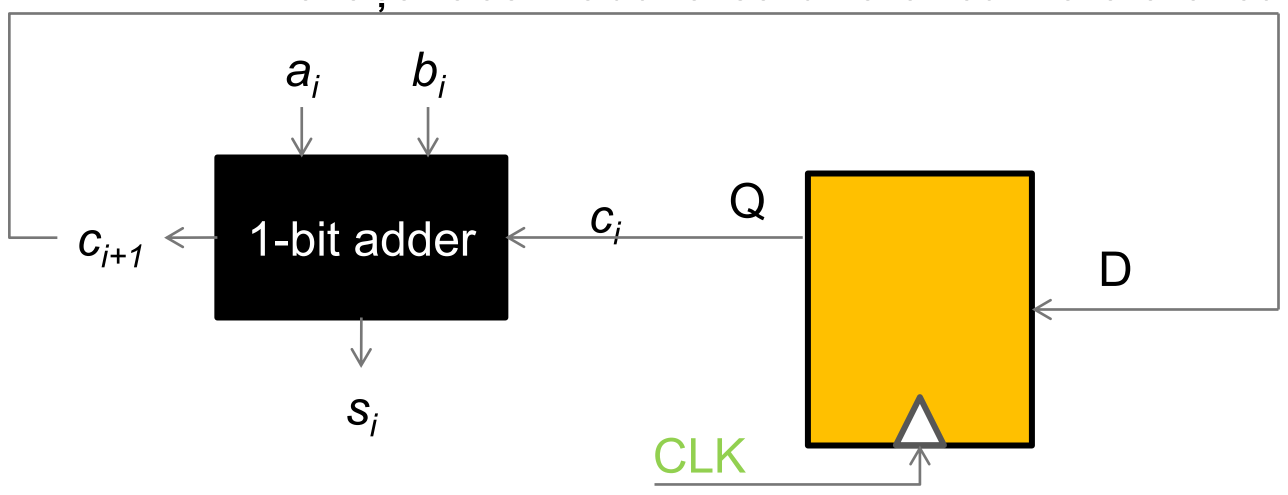
- **un exemplu**



este un counter pe 2 biți

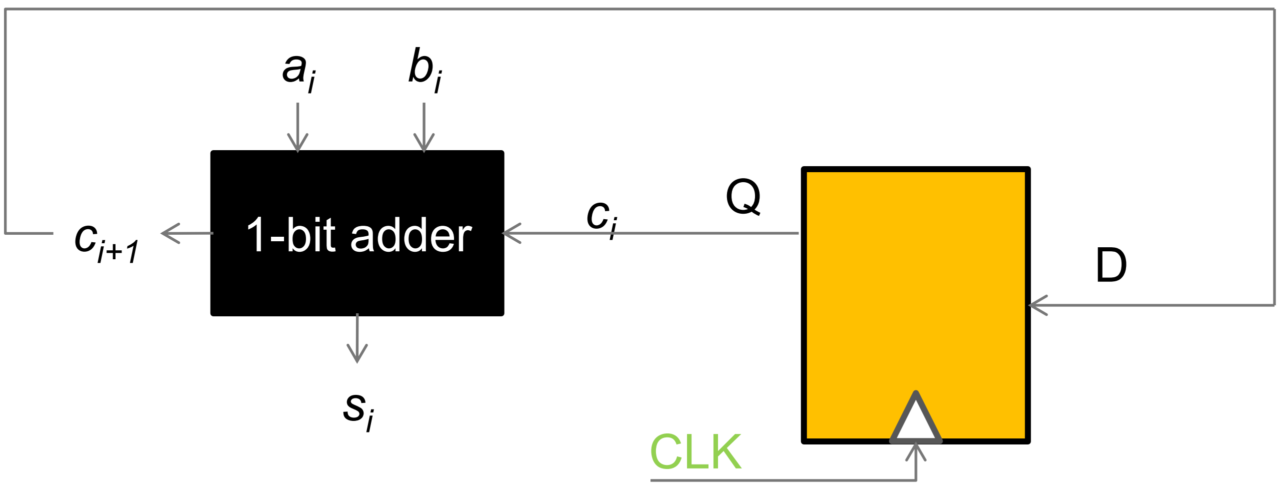
CIRCUITE SECVENȚIALE

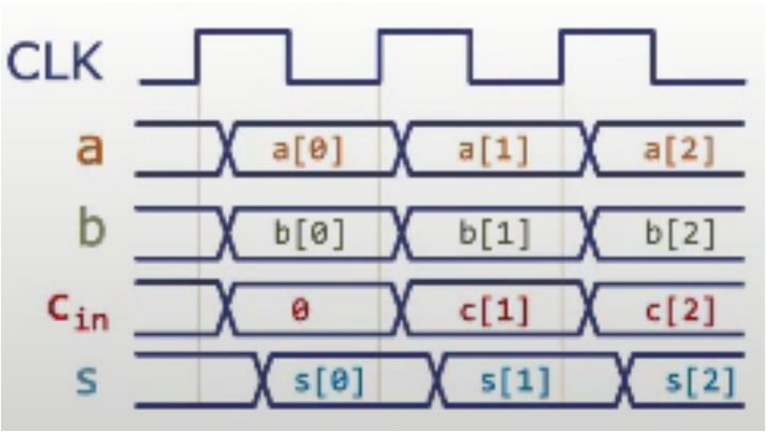
- **avantajele circuitelor secvențiale**
  - avem stare internă
    - exemplu: avem un registru în care memorăm count-ul curent
  - avem variabila de timp
    - intrările/ieșirile nu sunt fixe
    - număr variabil de pași în rezolvare
    - exemplu: construiți un circuit care poate să adune două numere 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**

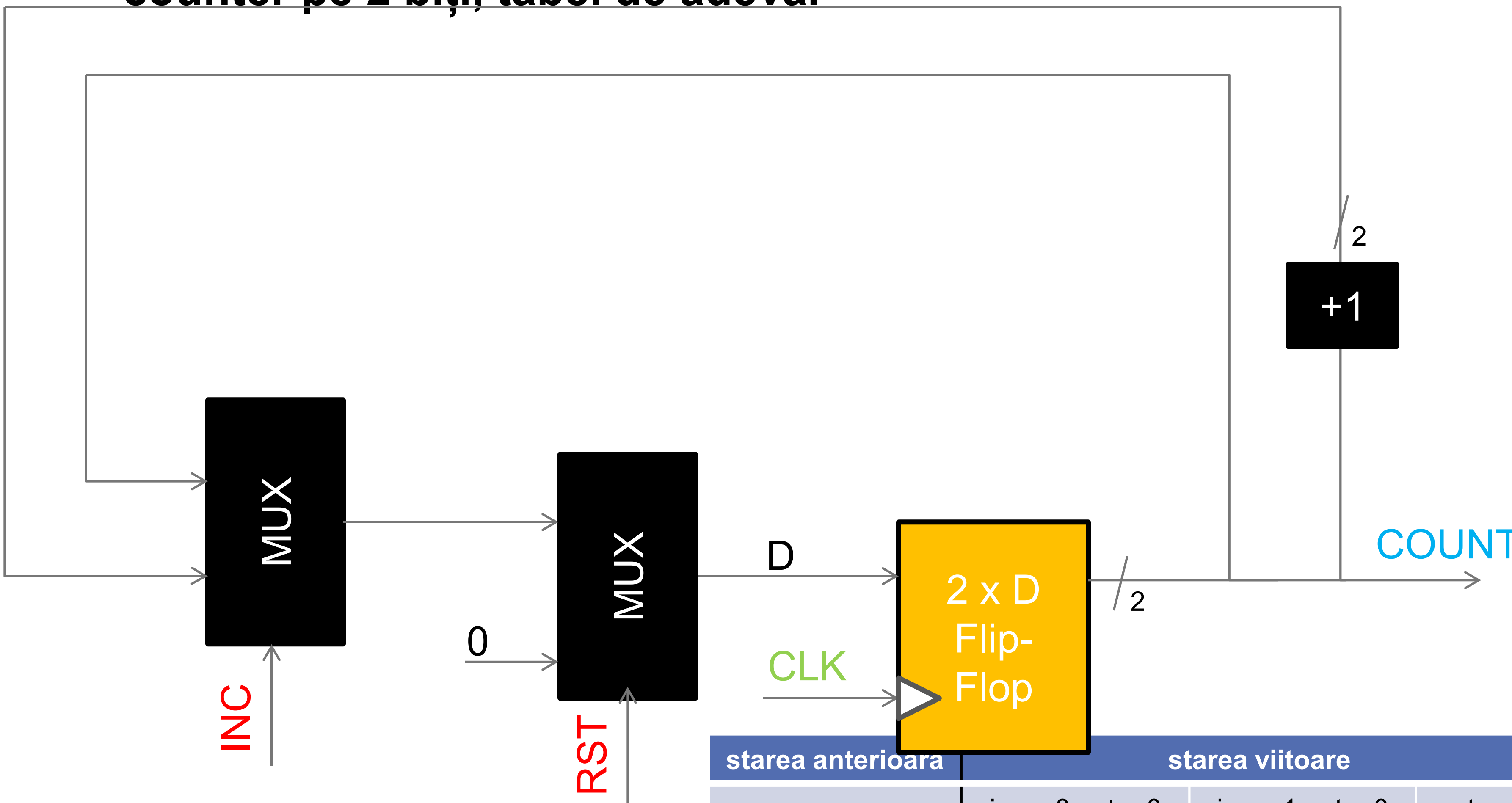




.

CIRCUITE SECVENȚIALE

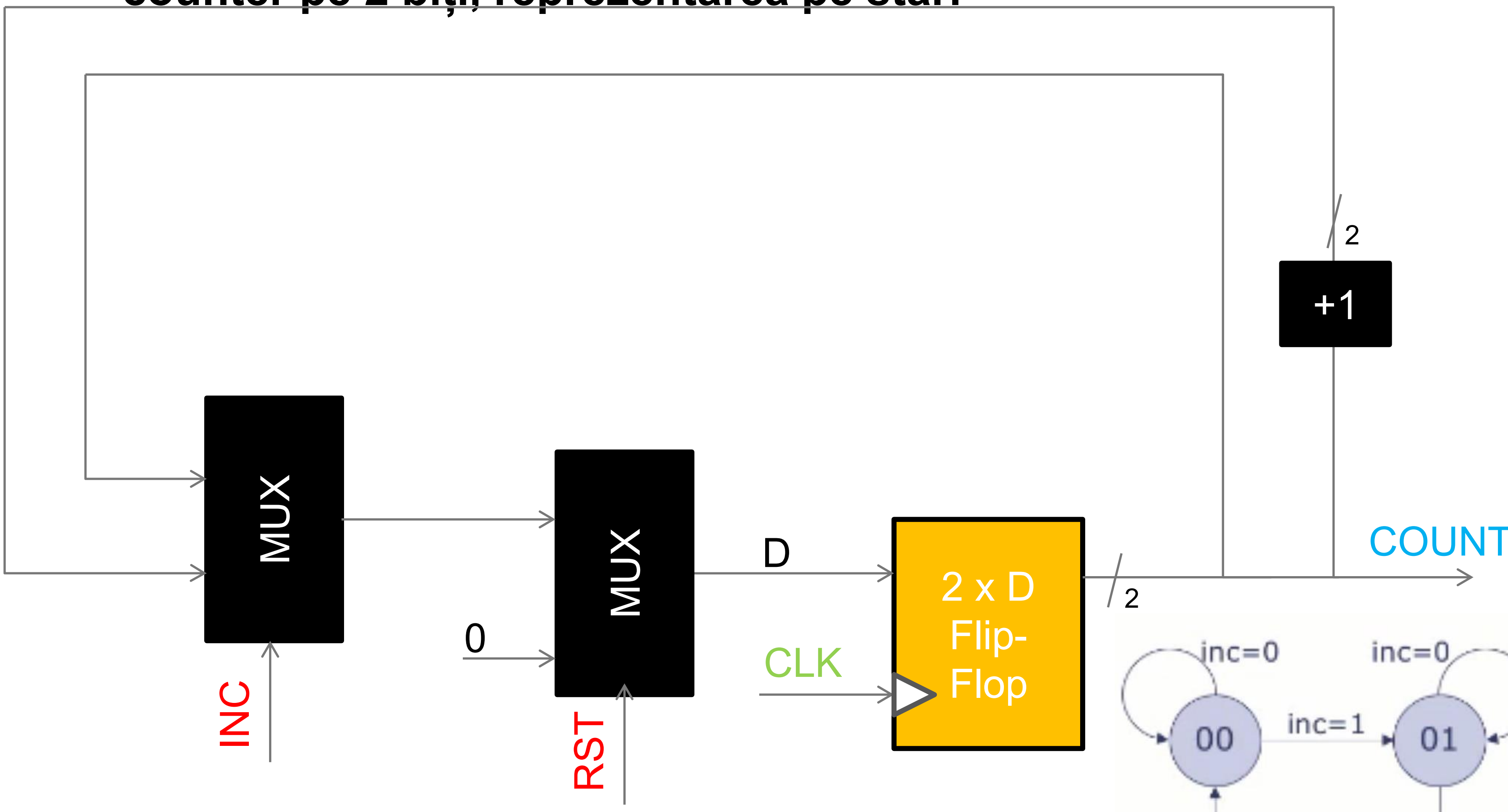
- **counter pe 2 biți, tabel de adevăr**

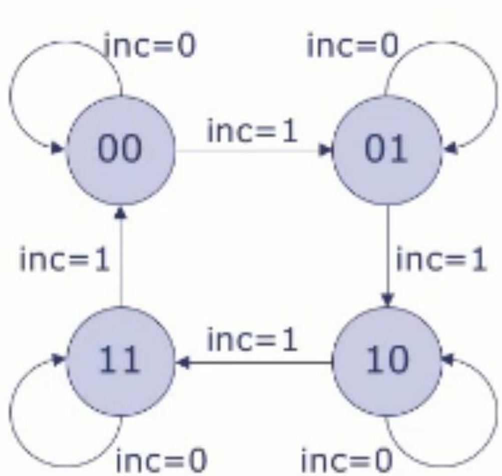


starea anterioara	starea viitoare		
	inc = 0, rst = 0	inc = 1, rst = 0	rst = 1
00	00	01	00
01	01	10	00
10	10	11	00
11	11	00	00

CIRCUITE SECVENȚIALE

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





CE AM FĂCUT ASTĂZI

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

DATA VIITOARE ...

- mai mult despre circuite de bază
- logică booleană
- sinteză logică
- circuite mai complexe
- circuite secvențiale

LECTURĂ SUPLIMENTARĂ

- **PH book**
  - Appendix B

- **Computerphile, Why Use Binary?, https://www.youtube.com/watch?v=thrx3SBEpL8**

- **Real Engineering, Transistors - The Invention That Changed The World, https://www.youtube.com/watch?v=OwS9aTE2Go4**

- **Ben Eater, Making logic gates from transistors, https://www.youtube.com/watch?v=sTu3LwpF6XI**

- **DrPhysicsA, An Introduction to Logic Gates, https://www.youtube.com/watch?v=95kv5BF2Z9E**

LECTURĂ SUPLIMENTARĂ

- 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