

CAP. ALGEBRA BOOLEANĂ

- princip. dualității : $a+a' = 1 \Rightarrow a \cdot a' = 0$.
- idempotentă : $x+x=x$, $x \cdot x=x$

CAP. REPREZENTAREA NUMERELOR ÎN SIST. DE CALCUL

SISTEME DE NUMERATIE POZITIONALE. CONVERSIA NUMERELOR

- din decimal în binar, cu nr. fractionare:

— nr. x baza nouă $\xrightarrow{[bx]}$ în fiz $\xrightarrow{\text{MSB}}$
 $\xrightarrow{\dots}$
 $\xrightarrow{\{bx\}}$ \curvearrowleft $\xrightarrow{\text{LSB}}$

REPREZENTAREA NUMERELOR ÎN VIREGULĂ FIXĂ

- la numere cu semn, MSB: 0: positive

1: negative

, virgula: după bo la nr. %, care e LSB
: după bo la fractionare, care e MSB

$$m = b_0 b_{-1} \dots b_{-m+1} b_{-m}$$

3 Forme de reprezentare

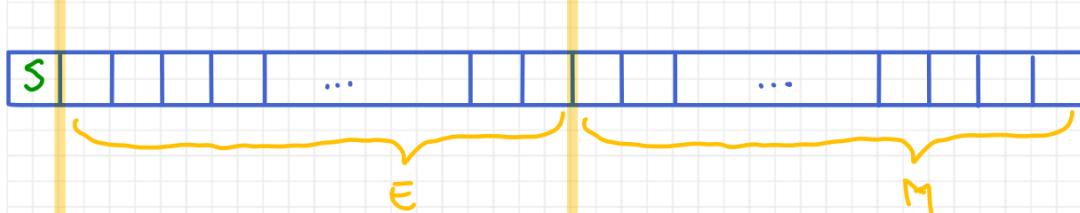
1 bit semn
n biti mărimie

	SM	C1	C2
POTENȚIU	$-2^{n-1} \div 2^{n-1}-1$ 2 repre. pt. zero	$-2^{n-1} \div 2^{n-1}-1$ 2 repre. pt. zero	$-2^{n-1} \div 2^{n-1}-1$ 1 repre. pt. zero
REPREZENTARE	0 - positive 1 - negative	0 - identic cu SM 1 - negativ maxim	0 - identic cu SM 1 - negativ și: $+1$ la γ_2 to.0...01 la fractionar
ex:	+85 0 1010101 -85 1 1010101	+85 0 1010101 -85 1 0101010	+85 0 1010101 -85 1 0101011
ADUNARE	$ \begin{array}{r} +2 \quad 0 \quad 010 \\ +3 \quad 0 \quad 011 \\ \hline +5 \quad 0 \quad 101 \end{array} $ $ \begin{array}{r} -2 \quad 1 \quad 010 \\ +3 \quad 0 \quad 011 \\ \hline -5 \quad 1 \quad 101 \end{array} $	$ \begin{array}{r} +2 \quad 0 \quad 010 \\ +3 \quad 0 \quad 011 \\ \hline +5 \quad 0 \quad 101 \end{array} $ $ \begin{array}{r} -2 \quad 1 \quad 101 \\ +3 \quad 0 \quad 011 \\ \hline -5 \quad 1 \quad 000 \end{array} $ <p style="text-align: center;">+1 0 001</p>	$ \begin{array}{r} +2 \quad 0 \quad 010 \\ +3 \quad 0 \quad 011 \\ \hline +5 \quad 0 \quad 101 \end{array} $ $ \begin{array}{r} -2 \quad 1 \quad 110 \\ +3 \quad 0 \quad 011 \\ \hline +1 \quad 0 \quad 001 \end{array} $
	$ \begin{array}{r} -2 \quad 1 \quad 010 \\ -3 \quad 1 \quad 011 \\ \hline +5 \quad 0 \quad 101 \end{array} $	$ \begin{array}{r} -2 \quad 1 \quad 101 \\ -3 \quad 1 \quad 100 \\ \hline +1 \quad 1 \quad 001 \end{array} $ <p style="text-align: center;">+1 0 001</p>	$ \begin{array}{r} -2 \quad 1 \quad 110 \\ -3 \quad 1 \quad 101 \\ \hline +1 \quad 0 \quad 111 \end{array} $ <p style="text-align: center;">-1 0 101</p>
	$ \begin{array}{r} +2 \quad 1 \quad 010 \\ -3 \quad 0 \quad 011 \\ \hline -5 \quad 1 \quad 101 \end{array} $	$ \begin{array}{r} +2 \quad 0 \quad 010 \\ -3 \quad 1 \quad 100 \\ \hline -1 \quad 1 \quad 110 \end{array} $ <p style="text-align: center;">-(001) ↙</p>	$ \begin{array}{r} +2 \quad 0 \quad 010 \\ -3 \quad 1 \quad 101 \\ \hline -1 \quad 1 \quad 111 \end{array} $ <p style="text-align: center;">-1 1 101</p>

REPREZENTAREA NUMERELOR ÎN VIRGULĂ FLOTANTĂ: $N = M \cdot B^E$

The diagram shows a horizontal line divided into three main sections: a sign bit 'S' on the far left, followed by a section labeled 'E' (exponent) containing several bits, and a section labeled 'M' (mantissa) containing many more bits. Ellipses indicate additional bits in both the exponent and mantissa sections.

mantisa baza exponential



$$\underline{\text{ex: pt. 18: } 18 \cdot 10^0 = 1,8 \cdot 10^1 = \dots = 0,00\dots 18 \cdot 10^{10}}$$

NORMALIZAREA MANTISEI: primul bit din dreapta $\boxed{}$ = 1 în SM
 : primul bit din dreapta $\boxed{}$ = 0
 pt. nr. negative în C1 și C2

REPREZENTAREA EXPONENTULUI: exp. cu zero nu fie cel mai mic posibil
 : val. numărătă exp. să fie 0.
 : toate val. neg. sunt deplasate prin adunarea unui **BIAS**
 : SM pe 8 biți : +127
 : C2 pe 8 biți : +128

IEEE 754 : este în format SM

: +127 la simplă precizie

: +1023 la dublă precizie

: hidden bit : bitul de 1 este mutat la \leftarrow virgulei :

$$S.1M \Rightarrow S1.M$$

SIMPLĂ PRECIE: 32 biti \rightarrow 1 semn
 8 de exponent, cu exces de 124
 23 de mantisa

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
S	exponent mantisa																														

Ex Reprez. 4,625 în IEEE 754

① Conv. B2: 4,625 \rightarrow 100,101 * 2^0

0,625 * 2 = 1,25 \rightarrow 1
 0,25 * 2 = 0,5 \rightarrow 0
 0,5 * 2 = 1,0 \rightarrow 1
 0 * 2 = 0 STOP

② Normalizare \Rightarrow 1,00101 * 2^2

Ajustare exponential: $2+124=129 \Rightarrow 1000001$

③ Deosebire

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	1	0	0	0	0	0	0	1	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DUBLĂ PRECIE: 64 biti \rightarrow 1 semn

11 de exponent, cu exces de 124
 52 de mantisa

0	1		11	12		63
S	exponent mantisa					

VALORI SPECIALE ALE lui IEEE 754:

E	M	N
0	0	± 0
0	$\neq 0$	Demormalized numbers
255	0	$\pm \infty$
255	$\neq 0$	NaN

Coduri BINARE PT NUMERE ZECI MALE:

- tutuc cifra din Blo cu corespondentul ei pe 4b din B2 \Rightarrow BCD

BCD

ex: $N_1 = 459_{10} = 0100 \ 0101 \ 1001$ BCD

$N_2 = 1000 \ 0111 \ 0000 \ 0010$ BCD = 8702_{10}

EXCESUL DE 3 - E3:

- la fiecare tetradă adun 3 : 0011

EXERCITII

2. Reprezentați următoarele numere întregi, în complement față de 2, pe un număr minim de biți. Prezentați codarea acestora pe 8 și 16 biți, în complement față de 2: 66, -73, 51, -84, 200, -200

$$\begin{array}{r|l} 66: 66 & -73: 73 \\ 33 \quad 0 & 36 \quad 1 \\ 16 \quad 1 & 18 \quad 0 \\ 8 \quad 0 & 9 \quad 0 \\ 4 \quad 0 & 4 \quad 1 \\ 2 \quad 0 & 2 \quad 0 \\ 1 \quad 0 & 1 \quad 0 \\ 0 \quad 1 & 0 \quad 1 \\ \hline \Rightarrow 0 \quad 100\ 010 & \Rightarrow 1 \quad 011\ 0111 \end{array}$$

$$\begin{array}{l} 88: 042 \\ 88: 1067 \\ B16: 022 \\ B16: 139 \end{array}$$

$$\begin{array}{r|l} 51: 51 & -84: 84 \\ 25 \quad 1 & 42 \quad 0 \\ 12 \quad 1 & 21 \quad 0 \\ 6 \quad 0 & 10 \quad 1 \\ 3 \quad 1 & 5 \quad 0 \\ 1 \quad 1 & 2 \quad 1 \\ 0 \quad 1 & 1 \quad 0 \\ 0 \quad 1 & 0 \quad 1 \\ \hline \Rightarrow 0 \quad 111011 & \Rightarrow 1 \quad 0101100 \end{array}$$

$$\begin{array}{l} 88: 043 \\ 88: 1054 \\ B16: 03B \\ B16: 12C \end{array}$$

200:200

100	0
50	0
25	0
12	1
6	0
3	0
1	1
0	1

$\Rightarrow 0 \ 1100 \ 1000$

$-200 \rightarrow 1100 \ 1000 \Rightarrow 00110111$

00111000

$\Rightarrow 1 \ 0011 \ 1000$

B8: 0310

B16: 0C8

B8: 1040.

B16: 138

3. Realizați următoarele adunări cu numere întregi exprimate în complement față de 2, pe 8 biți:

$$88 + (-15) \rightarrow 0 \ 101 \ 1000 \quad 1 \ 000 \ 1111 \quad \Rightarrow \begin{array}{r} 0 \ 101 \ 1000 + \\ 1 \ 11 \ 0000 \\ \hline 0 \ 100 \ 1001 \end{array}$$

$$(-14) + 14 \rightarrow \dots 1110 \quad \Rightarrow \begin{array}{r} 1 \ 111 \ 0010 \\ 0 \ 000 \ 1110 \\ \hline 1 \ 000 \ 0000 \end{array}$$

↓
.... 1110

$$(-85) + (-15) \rightarrow 1 \ 101 \ 0101 \quad 1 \ 000 \ 1111 \quad \Rightarrow \begin{array}{r} 1010 \ 1011 + \\ 111 \ 0000 \\ \hline 10011100 \sim 1 \end{array}$$

↓
010 1010
1 010 1011

10011011
11100100

$$(-68) + 15 \rightarrow 1 \ 100 \ 0100 \quad 0 \ 000 \ 1111 \quad \Rightarrow \begin{array}{r} 1011 \ 1100 + \\ 0000 \ 1111 \\ \hline 1100 \ 1011 \sim 1 \end{array}$$

011 1011
1 011 1100

1100 1010
1011 0101

18. Precizați multimea numerelor ce pot fi reprezentate astfel:

- a) numere pozitive, pe 7 biți; $[0, 127]$
- b) numere întregi, codificate ca mărime și semn (MS), pe 7 biți; $[-63; 63]$
- c) numere întregi, codificate în complement față de 1 (C1), pe 7 biți; $[-63; 63]$
- d) numere întregi, codificate în complement față de 2 (C2), pe 7 biți; $[-63; 64]$
- e) numere pozitive/intregi codificate MS/C1/C2, pe 8 biți; $[0, 127 \text{ resp. } 128] / [-127; 127 \text{ n. } 128]$
- f) numere pozitive/intregi codificate MS/C1/C2, pe 4 bytes. $[-(2^{31}-1); 2^{31}-1 \text{ resp. } -0]$

Determinați domeniile de reprezentare a numerelor codificate în MS/C1/C2 pentru un număr N de biți. Utilizați formulele determinante pentru 8/16/32/64 biți.

CAP. REPREZENTAREA FUNCȚIILOR

FUNCȚIU BOOLEENE : sumă de mintermi : $F = \sum (3, 5, 6, 7) \Rightarrow$ formă canonica disjunctivă

: produs de maxtermi : $F = \prod (0, 1, 2, 4) \Rightarrow$ formă canonica conjunctivă

PORȚI LOGICE : NOT, INVERTER, AND, OR, XOR, NAND, NOR, XNOR

DIAGRAME KARNAUGH : grupă cu ordinea

- IUPlicant PRIM: la această grupare nu mai pot fi adăugati mintermi
- IUPlicant PRIM ESENȚIAL: în această grupare există cel puțin un minterm care nu apare în alti iuplicanți primi.
- mintermi adiacenți: diferă DOAR PRIMUL-UL bit

		c				
		00	01	11	10	
ab		00	0000	0001	0011	0010
01		01	0100	0101	0111	0110
11		11	1100	1101	1111	1110
10		10	1000	1001	1011	1010

d

a

b

cd

MINIMIZARE QUINE - McCLUSKEY

Ex) Să ne minimizăm $f_1(a,b,c,d) = \sum(1,3,4,5,6,9,11,12,13)$

Pas 1: mintermii sunt grupați într-un tabel în funcție de nr. variabile menegate (numărău 1-wile) :

0: 0

1: 1, 2, 4, 8

2: 3, 5, 6, 9, 10, 12

3: 7, 11, 13, 14

4: 15

Pas 2: tabelăm doar mintermii din \sum :

GRUPA	mintermul	a	b	c	d	OK
1	1	0	0	0	1	
	4	0	1	0	0	
	3	0	0	1	1	
	5	0	1	0	1	
2	6	0	1	1	0	
	9	1	0	0	1	
	12	1	1	0	0	
	11	1	0	1	1	
3	13	1	1	0	1	

Pas 3: fiecare minterm din grupa i se compară cu fiecare minterm din gr.i+1

: se verifică adiacența : dacă DA : variab ce diforează tulor cu ..-

: grupul de 2 mintermi va fi trecut în grupa i

: în tab precedent se bifăza grupații

GRUPA		interval	a	b	c	d	OK
1	1,3	0	0	-	1		
	1,5	0	-	0	1		
	1,9	-	0	0	1		
	4,5	0	1	0	-		
	4,6	0	1	-	0		
	4,12	-	1	0	0		
	3,11	-	0	1	1		
	5,13	-	1	0	1		
	9,11	1	0	-	1		
	0,13	1	-	0	1		
2	12,13	1	1	0	-		

GRUPA		interval	a	b	c	d	OK
1	1	1	0	0	0	1	✓
	4	0	1	0	0	0	✓
	3	0	0	1	1	1	✓
	5	0	1	0	1	1	✓
	6	0	1	1	0	0	✓
	9	1	0	0	1	1	✓
	12	1	1	0	0	0	✓
	11	1	0	1	1	1	✓
	13	1	1	1	0	1	✓
2							
3							

recursiv.

GRUPA		interval	a	b	c	d	OK
1	1,3,9,11	-	0	-	1		
	1,5,9,13	-	-	0	1		
	4,5,12,13	-	1	0	-		

$$\Rightarrow \Psi_1(a,b,c,d) = \bar{a}b\bar{d} +$$

GRUPA		interval	a	b	c	d	OK
1	1,3	0	0	-	1	1	✓
	1,5	0	-	0	1	1	✓
	1,9	-	0	0	0	1	✓
	4,5	0	1	0	-	1	✓
	4,6	0	1	-	0	0	✗
	4,12	-	1	0	0	0	✓
	3,11	-	0	1	1	1	✓
	5,13	-	1	0	1	1	✓
	9,11	1	0	-	1	1	✓
	0,13	1	-	0	1	1	✓
2	12,13	1	1	0	-	1	✓
3							

Pas 3: nou tabel cu implicanti primii

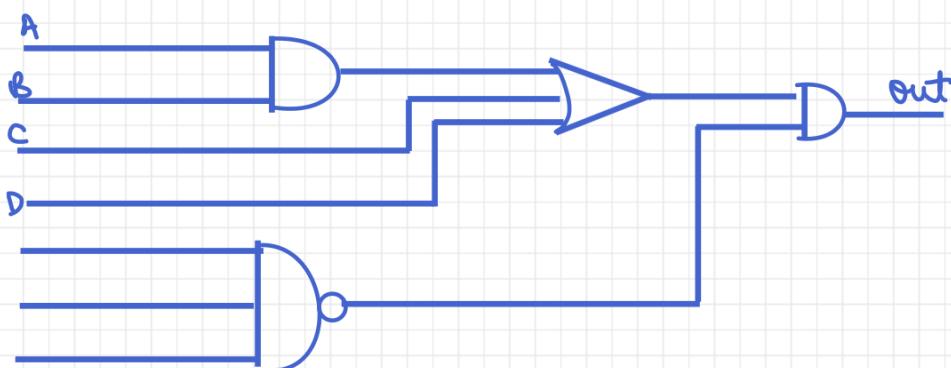
Implicanti primii acoperiti	Mintermi primii	1	3	4	5	6	9	11	12	13	mintermi
$\bar{a}b\bar{d}$	4,6			x	x						
$\bar{b}d$	1,3,9,11	x	x			x	x				
$\bar{c}d$	1,5,9,13	x		x	x	x	x	x			out
$b\bar{c}$	4,5,12,13			x	x			x	x		

$$\Rightarrow \Psi(a,b,c,d) = \bar{a}b\bar{d} + \bar{b}d + b\bar{c}$$

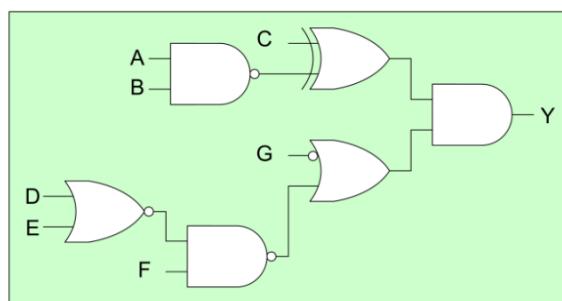
EXERCITII

2. Să se determine structura de porți logice care realizează următoarea funcție logică:

$$Y = (A \cdot B + C + D) \cdot \overline{E \cdot F \cdot G}$$



3. Să se determine funcția logică a structurii de porți logice prezentate în figura 3.2.

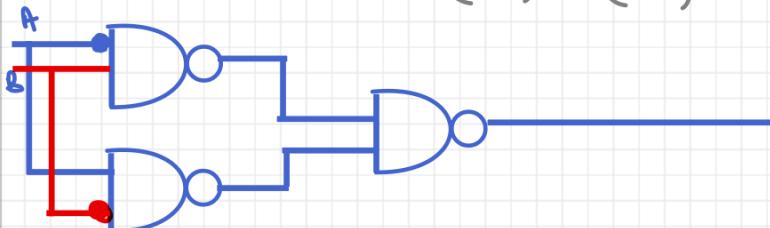


$$f = (\overline{A \cdot B} \oplus C) \cdot (\overline{(D+E)} \cdot \overline{F} + \overline{G})$$

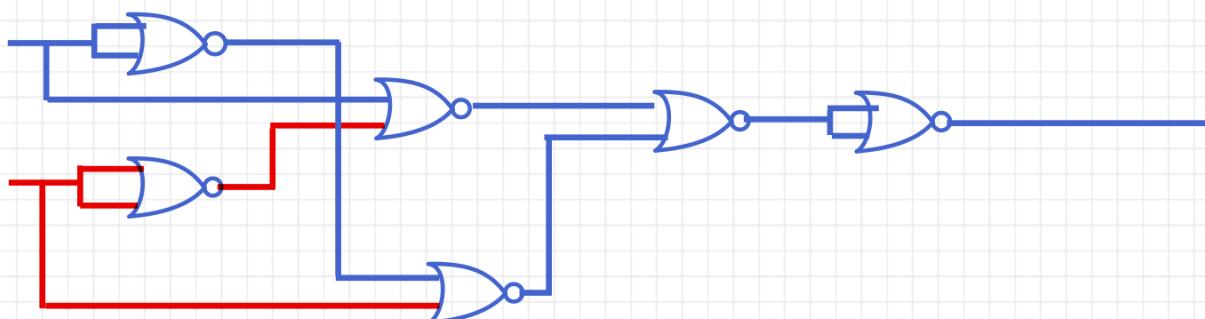
10. Să se implementeze funcția logică $F = A \oplus B$ cu porți:

- a) NAND cu două intrări;
- b) NOR cu două intrări.

$$a) f = A \cdot \overline{B} + \overline{A} \cdot B = \overline{\overline{A \cdot B}} \cdot \overline{\overline{A \cdot B}}$$



$$b) f = A \cdot \overline{B} + \overline{A} \cdot B = \overline{\overline{A} + B} + \overline{A + \overline{B}} = \overline{\overline{A} + B} + \overline{\overline{A} + \overline{B}}$$



1. Scrieți următoarele funcții ca sumă de mintermi:

a) $F_1(A, B, C) = A \cdot B \cdot C + \bar{A} \cdot \bar{B} \cdot C + A \cdot \bar{B} \cdot \bar{C} \rightarrow \sum(1, 4, 6) = \prod(0, 2, 3, 4, 5)$
 b) $F_2(A, B, C) = \bar{A} \cdot C + \bar{B} + A \cdot B \cdot \bar{C} \rightarrow \sum(1, 3, 5, 6, 7)$
 c) $F_3(A, B, C) = A \cdot B \cdot C + \bar{A} \cdot \bar{B} \cdot \bar{C} \rightarrow \sum(0, 4)$
 d) $F_4(A, B, C) = \bar{A} \cdot \bar{C} + C + A \cdot \bar{B} \cdot C \rightarrow \sum(0, 2, 1, 3, 5, 7)$

2. Scrieți următoarele funcții ca produs de maxtermi:

a) $F_1(A, B, C) = (A + B + C) \cdot (\bar{A} + \bar{B} + C) \cdot (A + B + \bar{C}) \quad \prod(1, 4, 6)$
 b) $F_2(A, B, C) = (\bar{A} + C) \cdot \bar{B} \cdot (A + B + \bar{C}) \quad \prod(4, 6, 2, 3, 7, 1)$

4. Scrieți expresiile funcțiilor negate exprimate în ambele forme canonice:

a) $F_1 = \sum(0, 1, 5, 6, 7)$
 b) $F_2 = \sum(2, 4, 6, 11, 14)$

$$\bar{F}_1 = \prod(0, 1, 5, 6, 7) = \sum(2, 3, 4)$$

$$\bar{F}_2 = \prod(2, 4, 6, 11, 14) = \sum(0, 1, 3, 5, 7, 8, 9, 10, 12, 13, 15)$$

1. Determinați tabelul de adevăr al următoarelor funcții și apoi exprimați-le în formele standard FCND și FCNC:

a) $F_1 = (X \cdot Y + Z) \cdot (Y + X \cdot Z)$
 b) $F_2 = \bar{Y} \cdot Z + W \cdot X \cdot \bar{Y} + W \cdot X \cdot \bar{Z} + \bar{W} \cdot \bar{X} \cdot Z$

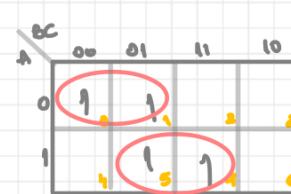
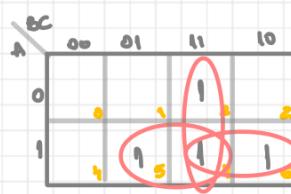
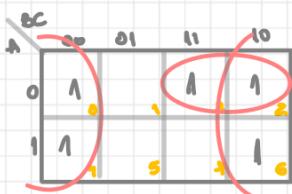
X	Y	Z	$X \cdot Y + Z$	$Y + X \cdot Z$	\bar{f}_{im}
0	0	0	0+0=0	0+0=0	0
0	0	1	0+1=1	0+0=0	1
0	1	0	0+0=0	1+0=1	2
0	1	1	0+1=1	1+0=1	3
1	0	0	0+0=0	0+0=0	4
1	0	1	0+1=1	0+1=1	5
1	1	0	1+0=1	1+0=1	6
1	1	1	1+1=1	1+1=1	7

$$\Rightarrow \text{FCND: } \sum(3, 5, 6, 7) = \dots$$

$$\text{FCNC: } \prod(0, 1, 2, 4) = \dots$$

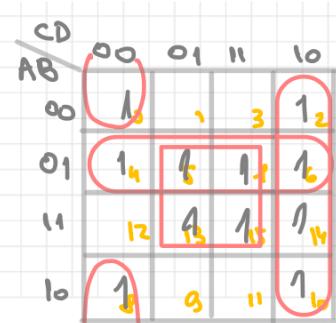
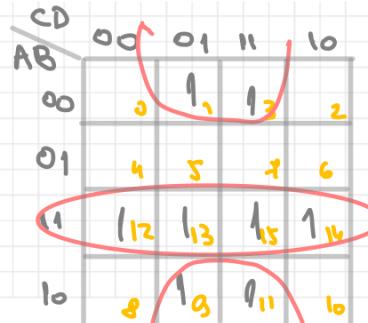
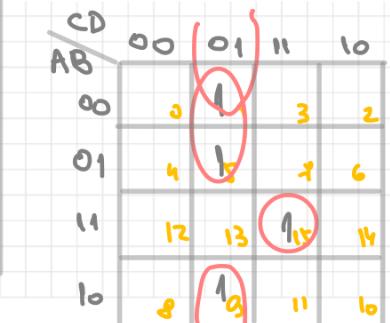
1. Să se minimizeze următoarele funcții de 3 intrări, utilizând diagrame V-K:

a) $F_a(A, B, C) = \sum(0, 2, 3, 4, 6)$
 b) $F_b(A, B, C) = \sum(3, 5, 6, 7)$
 c) $F_c(A, B, C) = \sum(0, 1, 5, 7)$



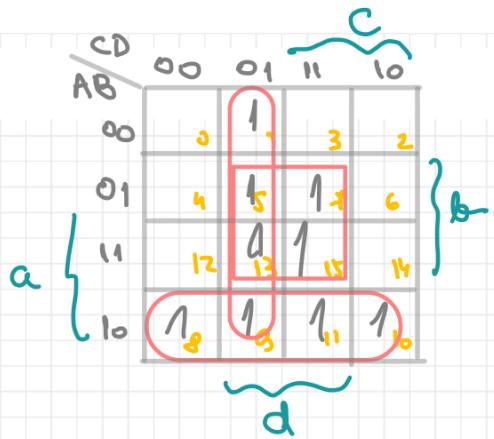
2. Să se minimizeze următoarele funcții de 4 intrări, utilizând diagrame V-K:

a) $F_a(A, B, C, D) = \sum(1, 5, 9, 15)$
 b) $F_b(A, B, C, D) = \sum(1, 3, 9, 11, 12, 13, 14, 15)$
 c) $F_c(A, B, C, D) = \sum(0, 2, 4, 5, 6, 7, 8, 10, 13, 14, 15)$



5. Să se identifice implicantii primi esențiali pentru următoarele expresii:
 a) $F_a(A, B, C, D) = \sum(1, 5, 7, 8, 9, 10, 11, 13, 15)$

$$AB, \bar{C}D, BD$$



CAP. CIRCUITE LOGICE COMBINATORIALE

CLASIFICARE: PROCESARE: + - * / , & | !, comparație, >> <<

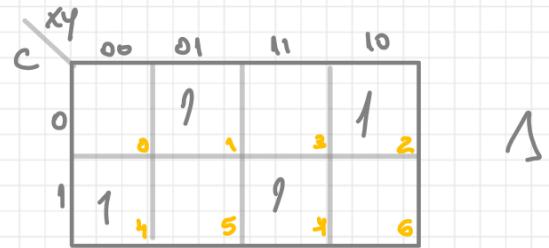
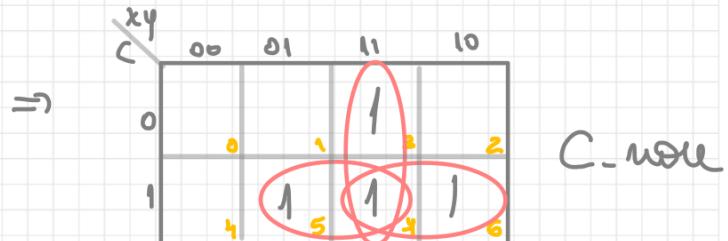
CONVERSIE DATE: codificatoare & decodificatoare

interconnect-wi

alte: ROM, PLA

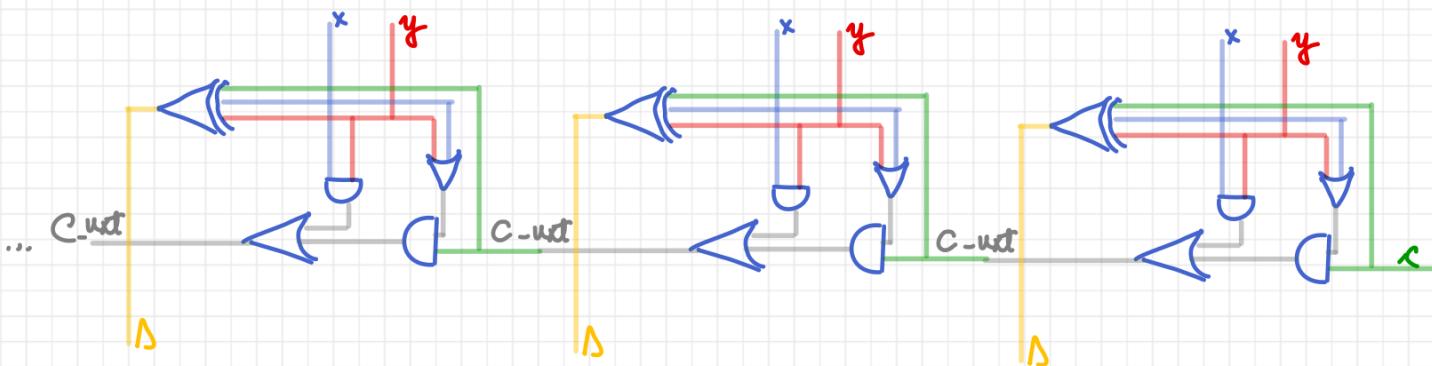
ex) SUMATORUL CU PROPAGARE SERIALĂ A TRANSPORTULUI:

x	y	carry	c_out	Δ
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1



$$\Rightarrow C_{\text{out}} = xy + cy + cx = xy + c(x+y)$$

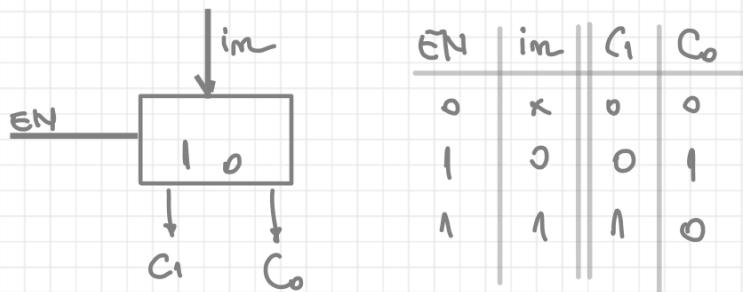
$$\Delta = x \oplus y \oplus c$$



DECODIFICATORUL

- circuit logic combinational ce prezinta un anumit m intrari și peisat la 2^m ieșiri, care activează IEȘIREA (una singură), corespunzătoare valorii combinației vectorului de intrare
- poate avea intrări de activare
- m intrări, m ieșiri \Rightarrow decodificator m-la-m.

Ex: decodif de la 1 la 2:

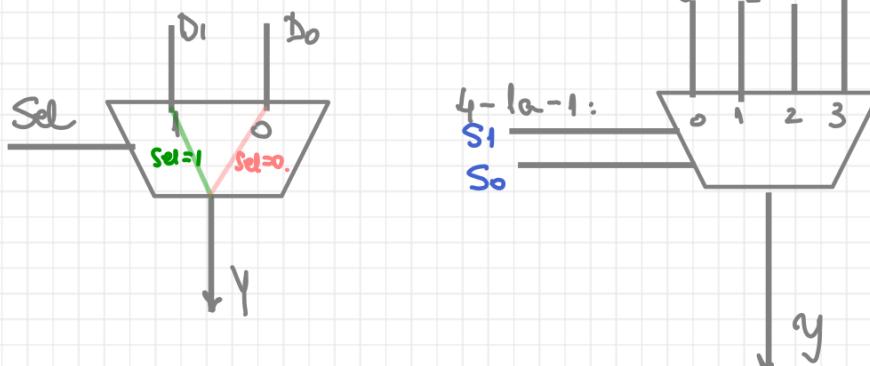


Ex: decodif de la 2 la 4: punem $C_k = 1$ când $(k)_B \wedge E$

MULTIPLEXORUL

- circuit logic combinational ce conectează ieșirea acestuia la una din cele m intrări
- selecția uneia din cele m intrări se face cu ajutorul a log2 m intrări de selecție

Ex: MUX 2-la-1 :



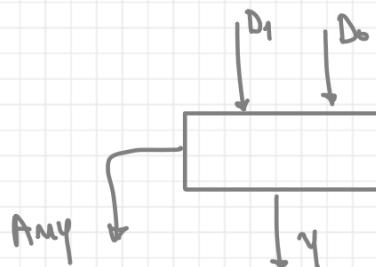
CODIFICATORUL

- circuit logic combinational care realizează într-un sens funcțional inversă decodificatorului

Obs: dacă conectăm un decodificator la ieșirea unui codificator nu obținem identitatea

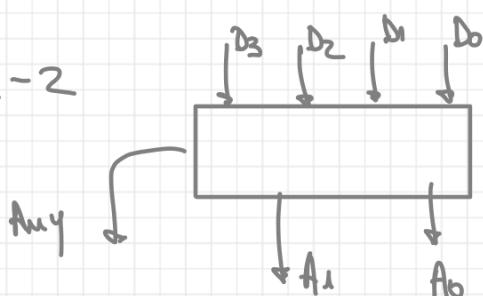
- de la pîmă la 2^n intrări și n ieșiri
- în cele mai multe cazuri, valoarea ieșirii unui codificator este dată de MSB de intrare activ

ex: Codif 2-la-1



D ₁	D ₀	Y	Amy
0	0	0	0
0	1	0	1
1	x	1	1

ex: codif 4-la-2

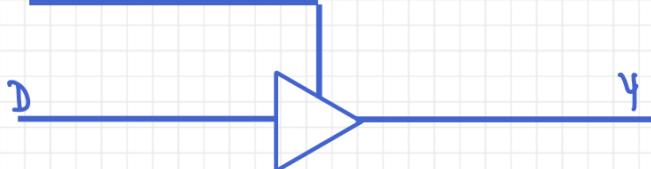


: A₁, A₂; A₂ sau 1 în binar ID-ul MSB-ului activ

• Amy ia 1 dacă oricare e activ

MAGISTRALA

- poartă cu 3 stări, 3 val de ieșire
 - 0
 - 1
 - X - impedanță ne dicată



E	D	Y
0	x	Z
1	x	D

EXERCITII

Decodificatorul $N : 2^N$ are N intrări de date pe care primește codul binar al unui număr în domeniul între 0 și $2^N - 1$. La un moment dat, o singură ieșire a decodificatorului este activă: cea cu indexul egal cu numărul prezentat la intrare. Starea activă a ieșirii poate fi în 0 sau în 1. Se poate considera că decodificatorul generează pe cele 2^N ieșiri toți mintermii celor N variabile prezentate la intrare.

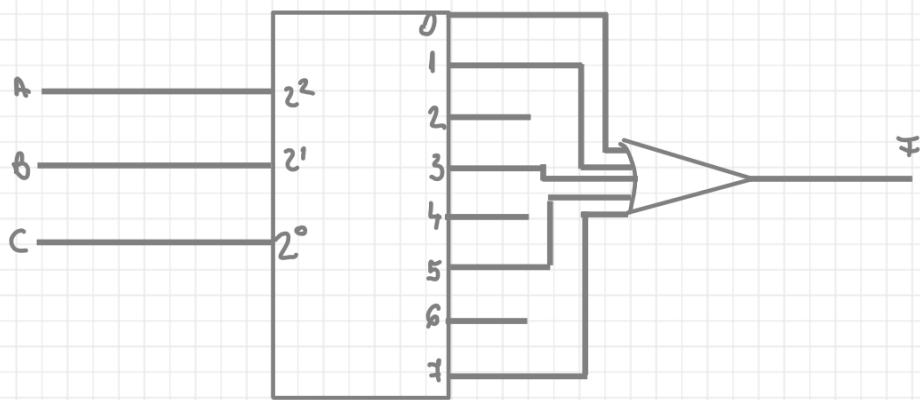
Un decodificator poate avea o intrare de validare. Dacă intrarea de validare este activată, decodificatorul funcționează și prezintă la ieșire valoarea decodificată a intrării (o singură ieșire activată). Dacă intrarea de validare nu este activată, toate ieșirile decodificatorului sunt în stare inactivă. Intrarea de validare oferă suport pentru extinderea capacitatei de decodificare.

Codificatorul priorităr de 2^N biți are 2^N intrări și N ieșiri. La ieșire se prezintă codul binar al celei mai prioritare intrări activate. Codificatorul poate avea o ieșire care semnalează apariția pe intrare a unei configurații valide (cu cel puțin un bit egal cu 1).

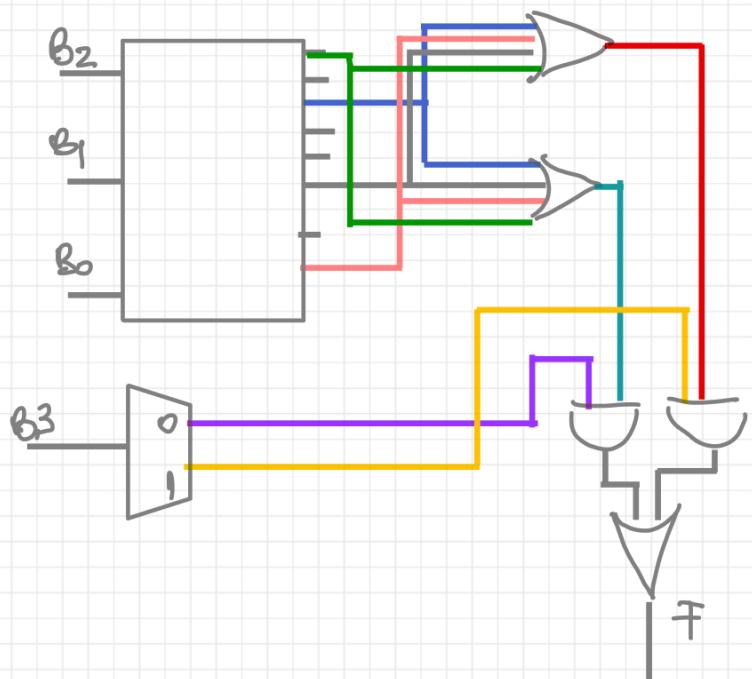
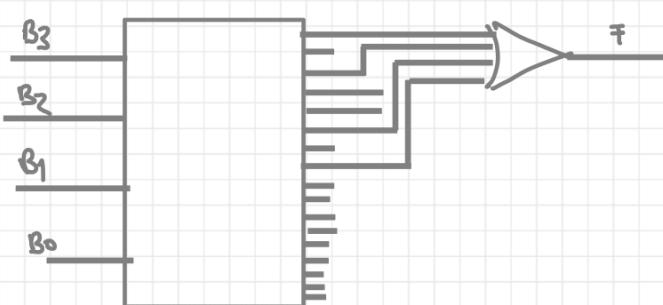
1. Să se implementeze cu decodificator 3:8 funcția: $F(A, B, C) = \sum(0, 1, 3, 5, 7)$.

\Downarrow
8 mintermi de 3 variabile

$$\bar{F} = m_0 + m_1 + m_3 + m_5 + m_7$$

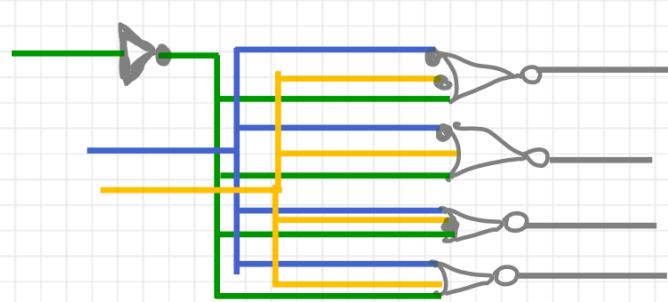


4. Să se implementeze următoarele funcții cu circuit decodificator de 4 biți. Implementați aceleași funcții utilizând un decodificator de 3 biți și un multiplexor 2:1. a) $F_a(A, B, C) = \sum(0, 2, 5, 7)$



7. Proiectați structura internă a unui decodificator de 2 biți cu intrare de validare, implementată cu porți NOR și NOT.

A_1	A_0	EN	O_3	O_2	O_1	O_0
X	X	0	0	0	0	0
0	0	1	0	0	0	1
0	1	1	0	0	1	0
1	0	1	0	1	0	0
1	1	1	1	0	0	0



$$\Rightarrow O_3 = \bar{A}_1 \cdot \bar{A}_0 \cdot EN$$

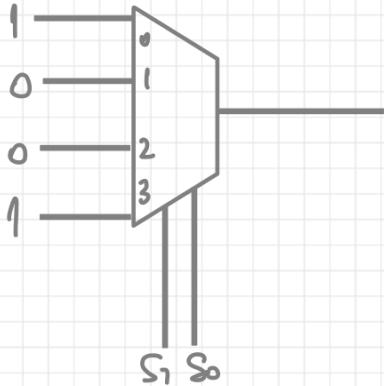
$$O_2 = A_1 \cdot \bar{A}_0 \cdot EN$$

$$O_1 = \bar{A}_1 \cdot A_0 \cdot EN$$

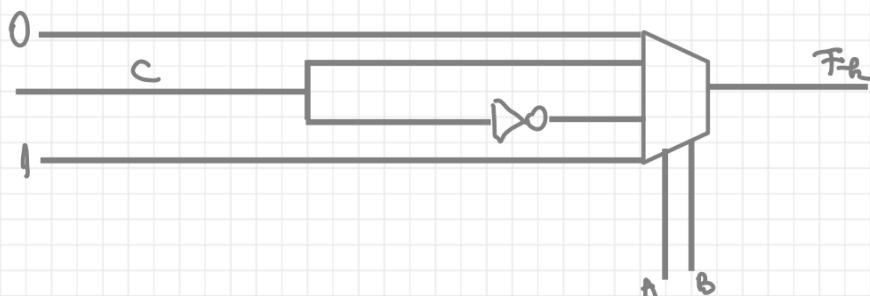
$$O_0 = \bar{A}_1 \bar{A}_0 \cdot EN$$

1. Implementați următoarele funcții conform cerințelor:

a) $F_a(A, B) = \sum(0, 3)$, cu MUX 4:1



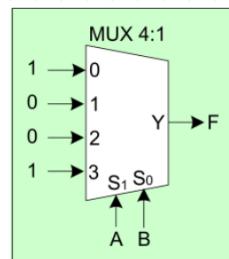
h) $F_h(A, B, C) = \sum(3, 4, 6, 7)$, cu MUX 4:1 și un inversor



	A	B	C	F_h	input
0	0	0	0	0	0
0	0	0	1	0	
0	1	0	0	0	
0	1	0	1	1	C
0	1	1	0	1	
1	0	0	0	1	
1	0	1	0	0	\bar{C}
1	1	0	0	1	
1	1	1	0	1	

3. Care din următoarele funcții este implementată de circuitul din figura 9.9?

- a) $F(A, B) = \overline{A \cdot B}$
- b) $F(A, B) = \overline{A + B}$
- c) $F(A, B) = \overline{A \oplus B}$
- d) $F(A, B) = A \oplus B$
- e) $F(A, B) = A \cdot B$



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

$$\Rightarrow \overline{\overline{A} \overline{B}} + A \overline{B} = \overline{A \oplus B}$$

4. Proiectați un circuit de "vot majoritar" cu 3 intrări și implementați-l cu multiplexor 4:1, fără minimizarea funcției. Ieșirea circuitului are valoarea logică a majorității intrărilor.

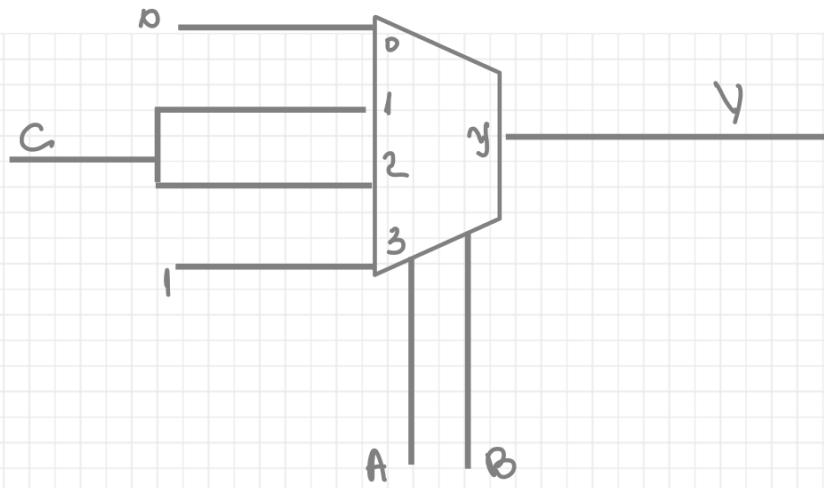
A	B	C	V
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

$$\Rightarrow V = \overline{\overline{A} BC} + A \overline{B} C + A B \overline{C} + A B \overline{C}$$

$$+ A \overline{B} C + A B C$$

$$= AB(\overline{C} + C + C + C) + \overline{A} BC + A \overline{B} C + \overline{AB} \cdot 0.$$

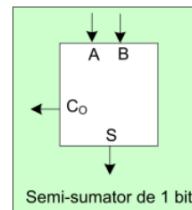
$$= AB \cdot 1 + \overline{AB} \cdot C + \overline{AB} \cdot C + \overline{AB} \cdot 0.$$



CIRCUITE ARITMETICE COMBINATORIALE

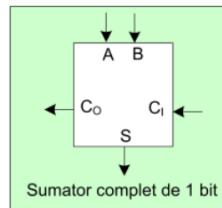
Semi-sumatorul de 1 bit este un circuit logic combinatorial cu două intrări și două ieșiri ce poate fi considerat ca având funcția aritmetică de a aduna 2 biți (operanții) și a genera rezultatul pe un bit (S) și transportul la bitul de ordin superior (C_O). Simbolul bloc și tabelul de adevăr ale semi-sumatorului de 1 bit sunt prezentate în figura 10.1.

Intrări		Ieșiri	
A	B	C_O	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0



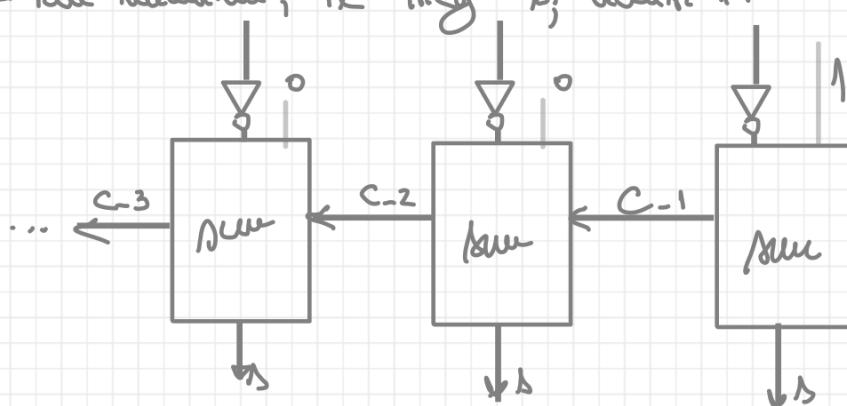
Sumatorul complet de 1 bit este un circuit logic combinatorial cu trei intrări și două ieșiri ce poate fi considerat ca având funcția aritmetică de a aduna 2 biți (operanții) plus transportul de la bitul inferior (C_I) și a genera rezultatul pe un bit (S) și transportul la bitul de ordin superior (C_O). Simbolul bloc și tabelul de adevăr ale sumatorului complet de 1 bit sunt prezentate în figura 10.2.

Intrări			Ieșiri	
A	B	C_I	C_O	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

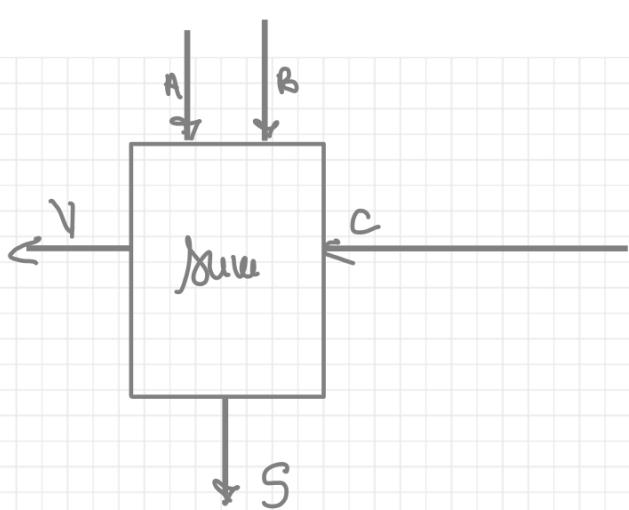


- Proiectați cu porti logice un circuit care convertește un număr de 4 biți în numărul negat (reprezentat în complement față de 2).

→ iau numărul, îl meg și adun 1:



- Proiectați un circuit de "vot majoritar" cu 3 intrări și implementați-l cu sumator complet de 1 bit. Ieșirea circuitului are valoarea logică a majorității intrărilor.



A	B	C	V	t_c	$R+B$	Carry
0	0	0	0	0	0	0
0	0	1	1	1	0	0
0	1	0	1	0	1	1
0	1	1	0	0	0	1
1	0	0	1	0	1	0
1	0	1	0	1	0	1
1	1	0	1	0	1	0
1	1	1	0	1	1	1

5. Să se proiecteze un circuit sumator/scăzător de 4 biți. Selecția funcției se va face cu o intrare suplimentară.

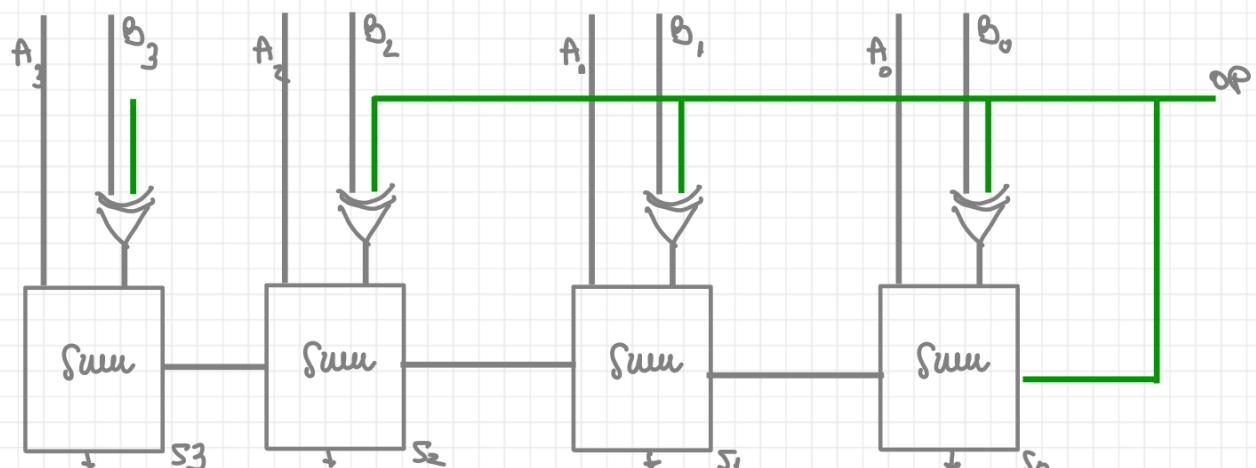
$$A+B / A-B = A+\bar{B}+1$$

A	B	\oplus	$B-n$	t^+
0	0	0	0	0
0	0	1	1	1
0	1	0	1	0
0	1	1	0	1
1	0	0	0	0
1	0	1	1	1
1	1	0	1	0
1	1	1	0	0

$$\Rightarrow B-n = \bar{A}\bar{B}\oplus + \bar{A}B\bar{\oplus} + A\bar{B}\oplus + AB\bar{\oplus} = \\ = \oplus(\bar{A}B + A\bar{B}) + \bar{\oplus}(\bar{A}\bar{B} + AB) = \\ = \oplus(\bar{B}) + \bar{\oplus}(B) = \oplus \oplus B$$

$$\Rightarrow t^+ = \oplus\oplus$$

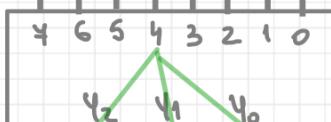
$$\Rightarrow A \pm B = A + B \oplus \oplus P + \oplus P \text{ pe } 1 \text{ bit}$$



1. Să se implementeze un circuit logic combinațional la intrarea căruia se aplică un cuvânt M de 8 biți și un cuvânt N de 3 biți. Ieșirea F a circuitului va fi activă când M este multiplu de 2^N .

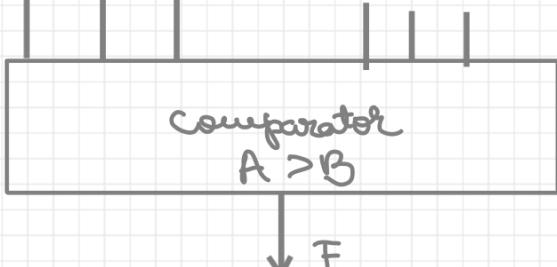
→ căutăm ca ultimii N biți ai lui M să fie zero

$M_0 M_1 M_2 M_3 M_4 M_5 M_6 M_7$



// $y > N$ va lua valoarea 1 din M_i , adică primul 1 din număr

$N_2 N_1 N_0$

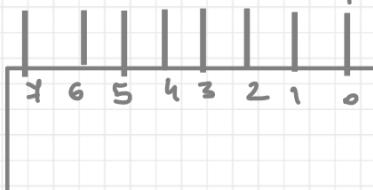


// dacă $y > N \Rightarrow M = k \cdot 2^N$

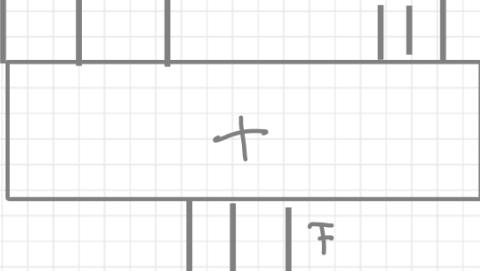
2. Să se implementeze un circuit logic combinațional care calculează relația: $F = [\log_2 M] + 1$, pentru M număr pozitiv reprezentat pe 8 biți. S-a notat cu $[]$ partea întregă a numărului.

$$= \log_2 M + 1$$

$M_7 M_6 M_5 M_4 M_3 M_2 M_1 M_0$

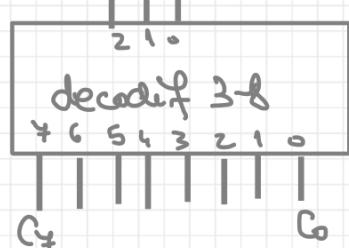
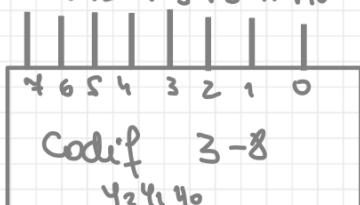


0 0 1



3. Să se proiecteze un circuit logic combinațional cu 8 intrări I_i și 8 ieșiri O_i , $i \in 7 \dots 0$. Circuitul activează în 1 numai ieșirea i de pe poziția celui mai semnificativ bit 1 din cuvântul de intrare.

$M_7 M_6 M_5 M_4 M_3 M_2 M_1 M_0$



obi: ROM-ul e COMBINATIONAL

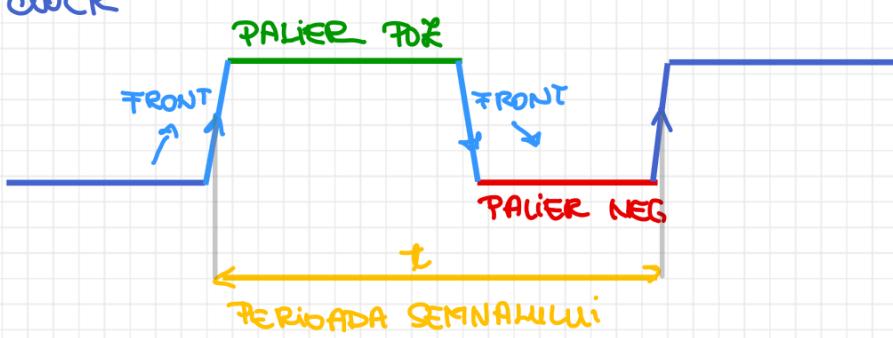
CAP. CIRCUITE LOGICE SEQUENȚIALE

- circuite care depind de starea curentă și un amumit t

COMPONENTE SEQUENȚIALE ASINCRONE: își modifică starea și valoarele de ieșire în funcție de modificările semnalelor de la intrare (always)

COMPONENTE SEQUENȚIALE SINCRONE: își modifică valoarea în funcție de valoarea semnalelor de intrare la momente bine definite de timp dictat de un clock

SEMNALUL DE TACT:

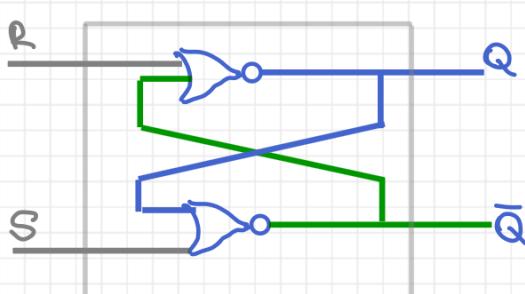


LATCHURI

- active pe PALIERUL semnalului de tact

SR-LATCH

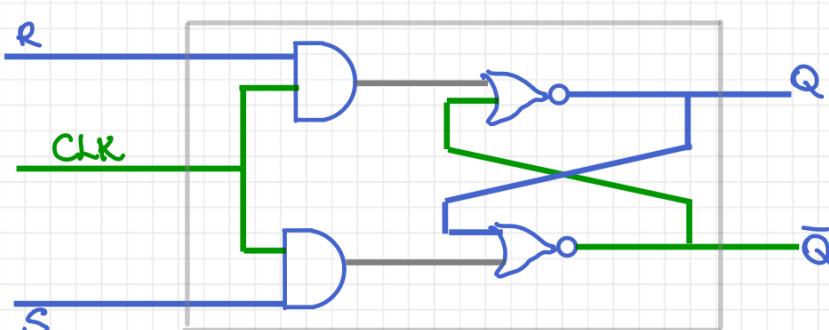
- e asincron (nu primește niciun CLK)



R	S	Q-ult
0	0	Q
0	1	1
1	0	0
1	1	interfis

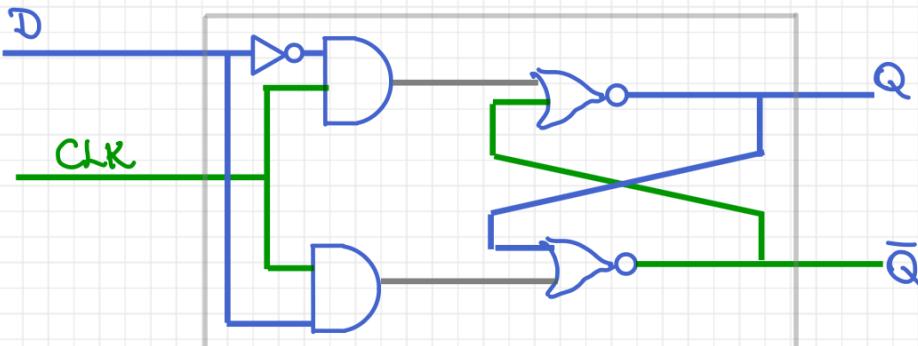
GATED SR - LATCH

- e sincron (primește CLK)



CLK	S	R	Q-ult
0	x	x	Q inactiv
1	0	0	Q hold
1	0	1	reset
1	1	0	set
1	1	1	interfis

GATED D-LATCH

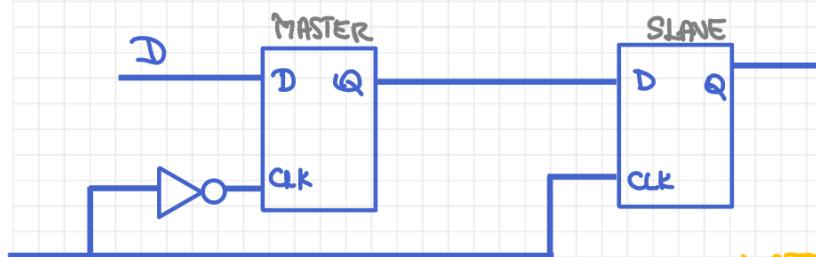


Clk	D	Q - uNt
0	x	Q inactiv
1	0	Q reset
0	1	Q set

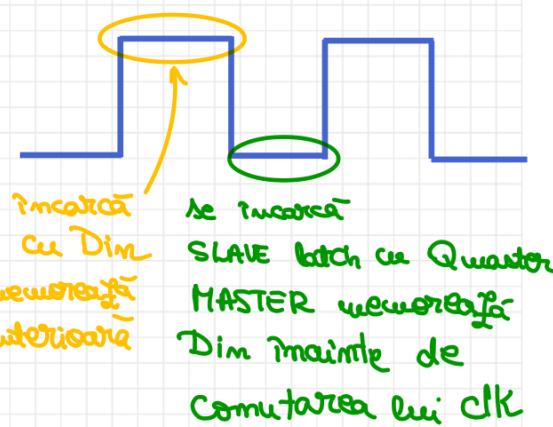
FLIP-FLOPURI

- active pe FRONTUL CRESCĂTOR al semnalului de tact
- 2 variante de arhitectură:

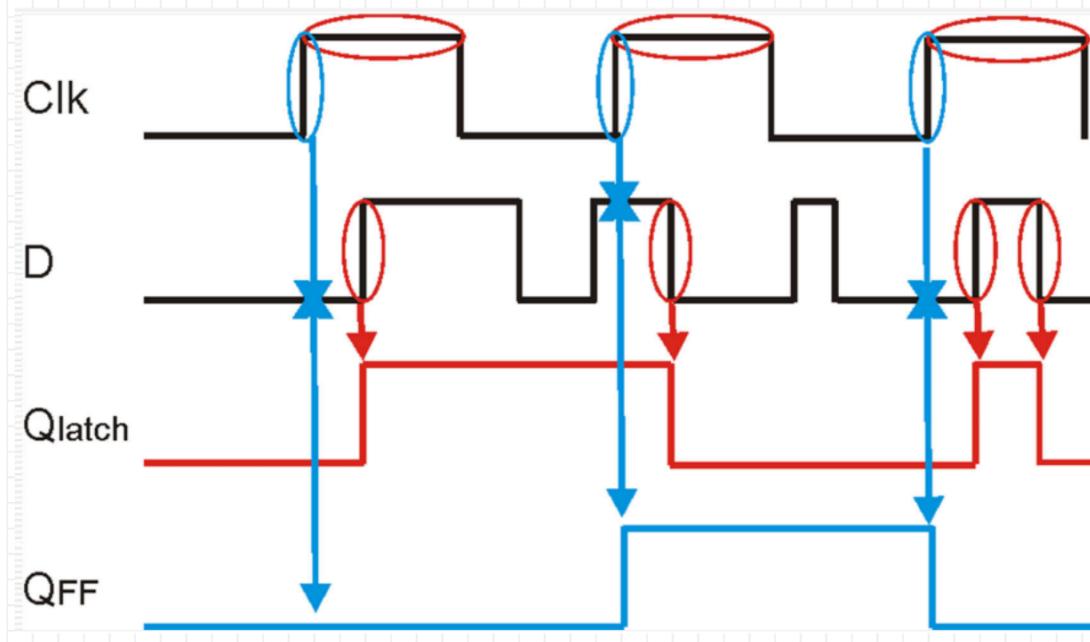
FF MASTER-SLAVE



Ne încreză
MASTER latch cu D în
SLAVE memorază
data anterioră



DIFERENȚA DINTE LATCH SINCRON și FF SINCRON



// tactul

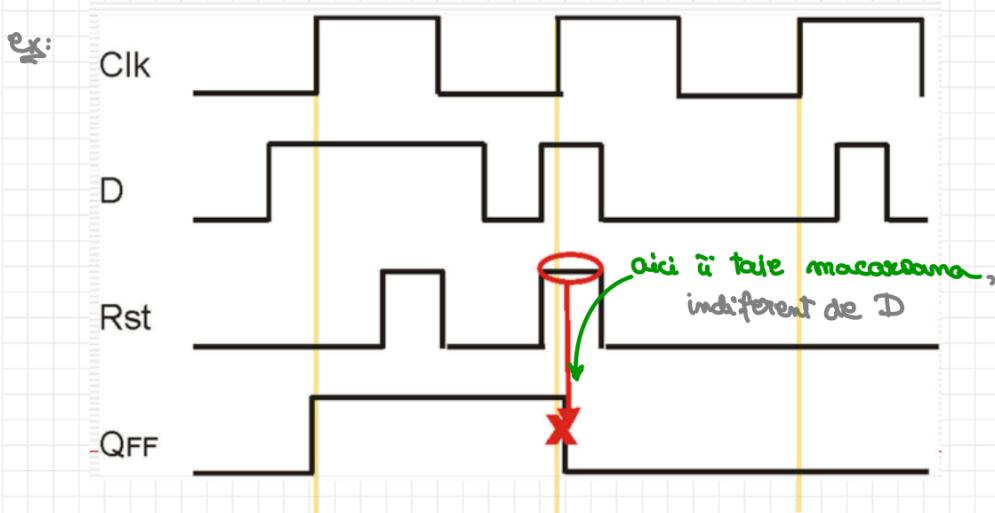
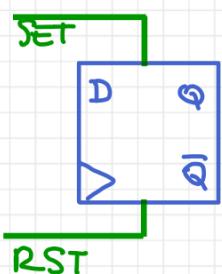
// valori random să
acopere teate
cărurile

CONSTRÂNGERI DE TEMP ÎN OPERAREA FF-URILOR :

- **TIMPUL DE SETUP :** timpul necesar pentru ca semnalul D de intrare să rămână stabil înainte de apariția frontului semnalului de tact
- **TIMPUL DE HOLD:** timpul în care datele de intrare nu pot fi modificate după apariția semnalului de tact, în vederea încărcării corecte a acestora
- **TIMPUL ATENT ÎNTÂRZIERII DATARETE PROPAGĂRII :** timpul necesar basculării FF-ului (clock to Q delay)

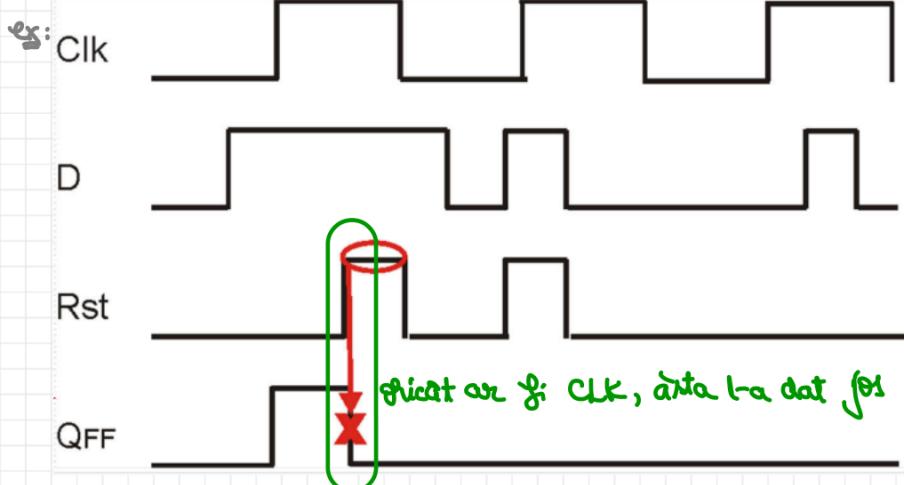
SEGNALUL DE RESET

- funcționalitate: aduce bistabilul într-o stare „initială” cunoscută
- reset este un semnal global: aplicat tuturor elementelor de memorie dintr-un sistem digital
- tipuri de reset: sincron / asincron
- **RESET SINCRON :** activat pe palierul (la latch) sau frontul (la FF) activ al semnalului de clk



```
always
  @(posedge clk)
begin
  if (rst)
    begin
      q <= 0;
    end
  else
    begin
      q <= d;
    end
end
```

• RESET ASINCRON: resetează elementul secvențial INDIIFERENT de clk



```
always
  @((posedge clk, posedge rst))
begin
  if (rst)
    begin
      q <= 0;
    end
  else
    begin
      q <= d;
    end
end
```

MODALITĂȚI DE DESCRIERE ALE CIRCUITELOR SECVENTIALE:

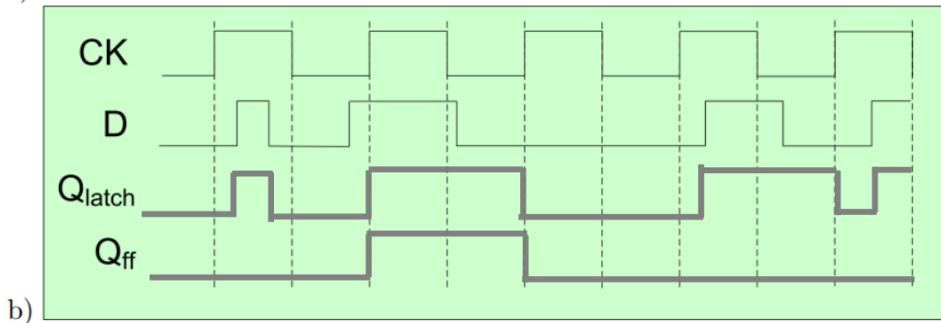
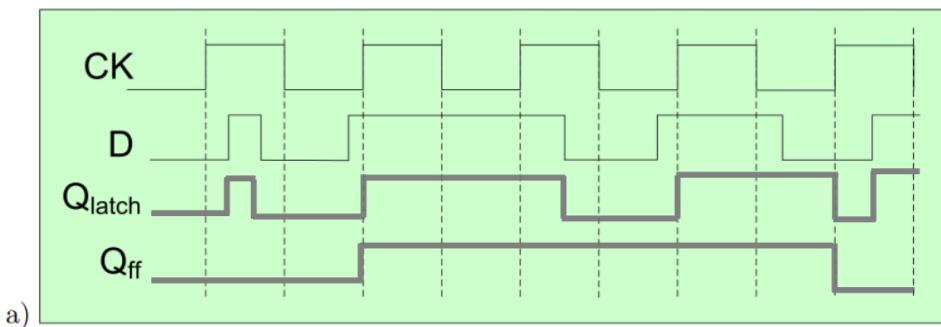
- TABEL CARACTERISTIC : tabelul de tranziție de la o stare la alta
- ECUAȚIE CARACTERISTICĂ : minimizarea din tabel
- TABEUL EXCITĂRILOR : folosit la întregul circuitului
 - : specifică intrările necesare pentru a trece din starea curentă în starea următoare
- GRAFURI DE STARE

TIPURI DE FF-URI

NUME	SIMBOL	TAB. CARACTERISTIC	EC.CARACTERISTICĂ	TAB. DE EXCITARE																																			
SR		<table border="1"> <tr><td>S</td><td>R</td><td>Q_{next}</td></tr> <tr><td>0</td><td>0</td><td>Q</td></tr> <tr><td>0</td><td>1</td><td>0</td></tr> <tr><td>1</td><td>0</td><td>1</td></tr> <tr><td>1</td><td>1</td><td>-</td></tr> </table>	S	R	Q _{next}	0	0	Q	0	1	0	1	0	1	1	1	-	$Q_{next} = S + \bar{R}Q$	<table border="1"> <tr><td>Q</td><td>Q_{old}</td><td>S</td><td>R</td></tr> <tr><td>0</td><td>0</td><td>0</td><td>X</td></tr> <tr><td>0</td><td>1</td><td>1</td><td>X</td></tr> <tr><td>1</td><td>0</td><td>0</td><td>1</td></tr> <tr><td>1</td><td>1</td><td>X</td><td>0</td></tr> </table>	Q	Q _{old}	S	R	0	0	0	X	0	1	1	X	1	0	0	1	1	1	X	0
S	R	Q _{next}																																					
0	0	Q																																					
0	1	0																																					
1	0	1																																					
1	1	-																																					
Q	Q _{old}	S	R																																				
0	0	0	X																																				
0	1	1	X																																				
1	0	0	1																																				
1	1	X	0																																				
JK		<table border="1"> <tr><td>J</td><td>K</td><td>Q_{next}</td></tr> <tr><td>0</td><td>0</td><td>Q</td></tr> <tr><td>0</td><td>1</td><td>0</td></tr> <tr><td>1</td><td>0</td><td>1</td></tr> <tr><td>1</td><td>1</td><td>Q'</td></tr> </table>	J	K	Q _{next}	0	0	Q	0	1	0	1	0	1	1	1	Q'	$Q_{next} = J\bar{Q} + \bar{K}Q$	<table border="1"> <tr><td>Q</td><td>Q_{old}</td><td>J</td><td>K</td></tr> <tr><td>0</td><td>0</td><td>0</td><td>X</td></tr> <tr><td>0</td><td>1</td><td>1</td><td>X</td></tr> <tr><td>1</td><td>0</td><td>X</td><td>1</td></tr> <tr><td>1</td><td>1</td><td>X</td><td>0</td></tr> </table>	Q	Q _{old}	J	K	0	0	0	X	0	1	1	X	1	0	X	1	1	1	X	0
J	K	Q _{next}																																					
0	0	Q																																					
0	1	0																																					
1	0	1																																					
1	1	Q'																																					
Q	Q _{old}	J	K																																				
0	0	0	X																																				
0	1	1	X																																				
1	0	X	1																																				
1	1	X	0																																				
D		<table border="1"> <tr><td>D</td><td>Q_{next}</td></tr> <tr><td>0</td><td>0</td></tr> <tr><td>1</td><td>1</td></tr> </table>	D	Q _{next}	0	0	1	1	$Q_{next} = D$	<table border="1"> <tr><td>Q</td><td>Q_{old}</td><td>D</td></tr> <tr><td>0</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>1</td><td>1</td></tr> <tr><td>1</td><td>0</td><td>0</td></tr> <tr><td>1</td><td>1</td><td>1</td></tr> </table>	Q	Q _{old}	D	0	0	0	0	1	1	1	0	0	1	1	1														
D	Q _{next}																																						
0	0																																						
1	1																																						
Q	Q _{old}	D																																					
0	0	0																																					
0	1	1																																					
1	0	0																																					
1	1	1																																					
T toggle		<table border="1"> <tr><td>T</td><td>Q_{next}</td></tr> <tr><td>0</td><td>Q</td></tr> <tr><td>1</td><td>Q'</td></tr> </table>	T	Q _{next}	0	Q	1	Q'	$Q_{next} = TQ' + \bar{T}'Q$	<table border="1"> <tr><td>Q</td><td>Q_{old}</td><td>T</td></tr> <tr><td>0</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>1</td><td>1</td></tr> <tr><td>1</td><td>0</td><td>1</td></tr> <tr><td>1</td><td>1</td><td>0</td></tr> </table>	Q	Q _{old}	T	0	0	0	0	1	1	1	0	1	1	1	0														
T	Q _{next}																																						
0	Q																																						
1	Q'																																						
Q	Q _{old}	T																																					
0	0	0																																					
0	1	1																																					
1	0	1																																					
1	1	0																																					

EXERCITII

3. Care este diferența dintre un "latch D" și un "bistabil D"? Completăți formele de undă ale ieșirilor latch-ului și ale bistabilului în cazul stimулilor de intrare prezențați în figura 13.6.



1. Descrieți funcționarea bistabilelor D/T/JK sub forma unor tabele de determinare a stării viitoare pe baza stării prezente și a intrărilor.

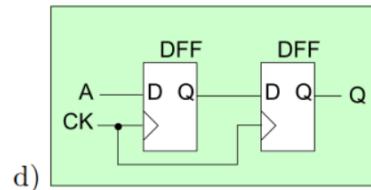
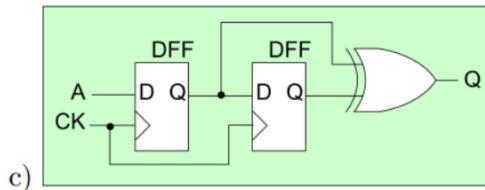
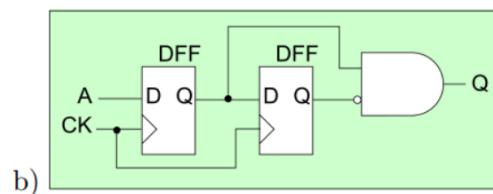
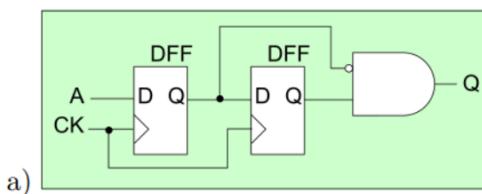
D	Q	Q^+	Acțiune
0	X	0	copiază intrarea D
1	X	1	copiază intrarea D

T	Q	Q^+	Acțiune
0	0	0	păstrează starea
0	1	1	păstrează starea
1	0	1	complementează starea
1	1	0	complementează starea

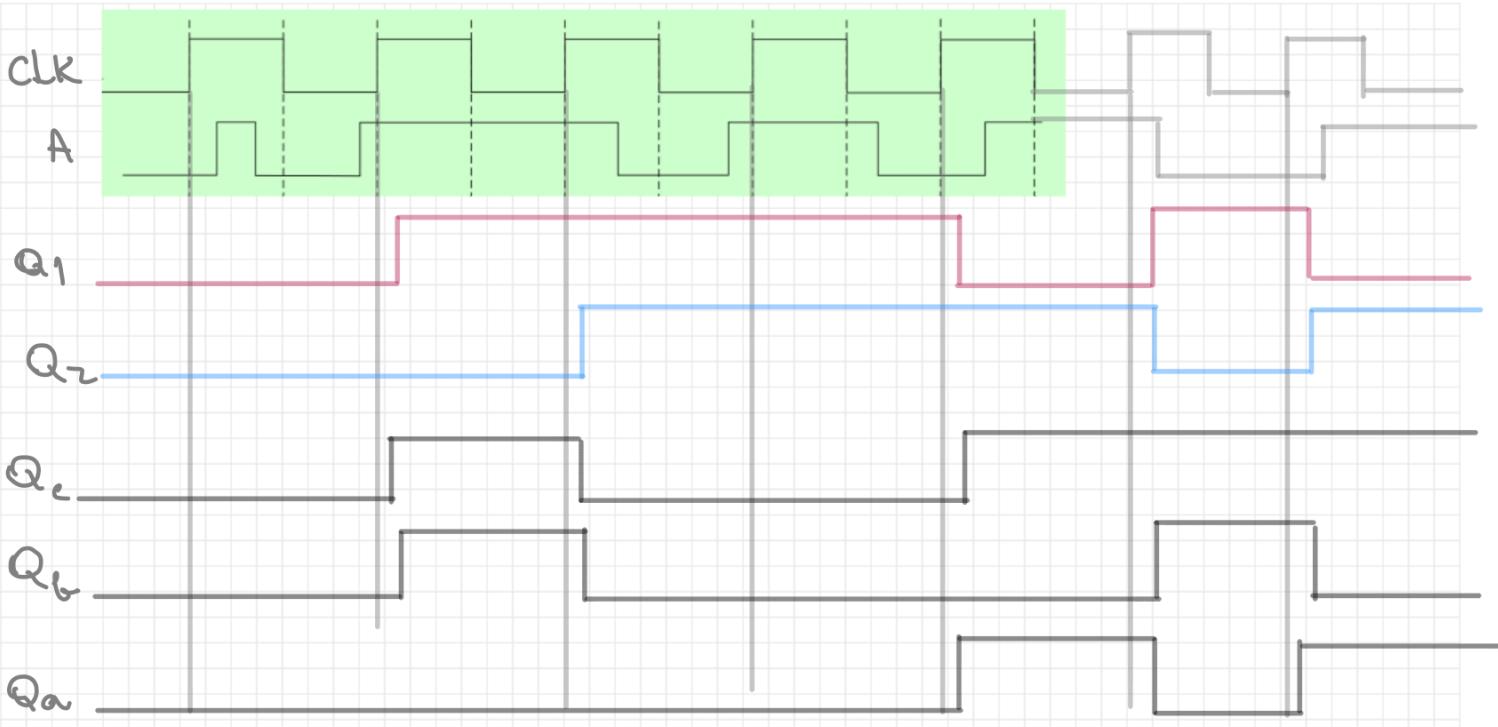
J	K	Q	Q^+	Acțiune
0	0	0	0	păstrează starea
0	0	1	1	păstrează starea
0	1	X	0	resetează starea
1	0	X	1	setează starea
1	1	0	1	complementează starea
1	1	1	0	complementează starea

2. Asociați circuitele prezentate în figura 13.8 cu funcțiile implementate de acestea:

- a) circuit de întârziere cu două perioade de ceas;
- b) detector de front pozitiv;
- c) detector de front;
- d) detector de front negativ.



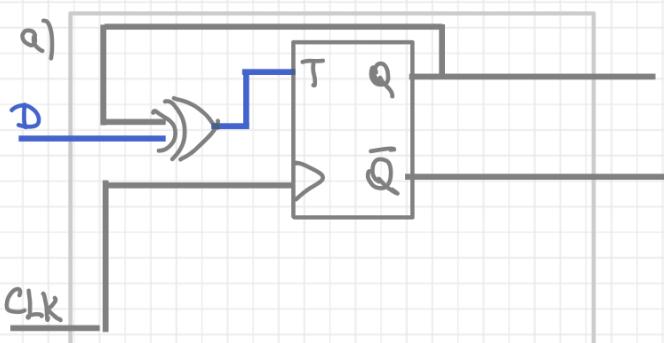
FRONT NEGATIV
(M-S cu clk menegat)



9. Realizați următoarele conversii de bistabile (implementați bistabilul de tip precizat în partea stângă folosind bistabilul propus în partea dreaptă și porți logice suplimentare).

- a) bistabil D cu bistabil T,
- b) bistabil T cu bistabil D,
- c) bistabil JK cu bistabil D,

- d) bistabil D cu bistabil JK,
- e) bistabil T cu bistabil JK,
- f) bistabil JK cu bistabil T.



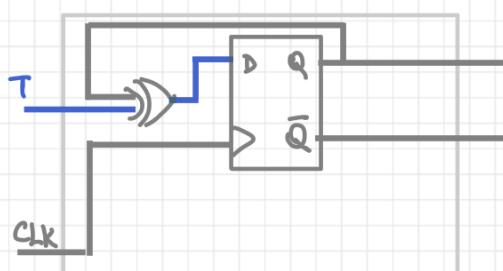
T	Qout	D	Q	Qout	T
0	Q	0	0	0	0
1	Q̄	1	1	1	1

$$\Rightarrow T = \overline{D}\bar{Q} + \bar{D}Q = D \oplus Q$$

b)

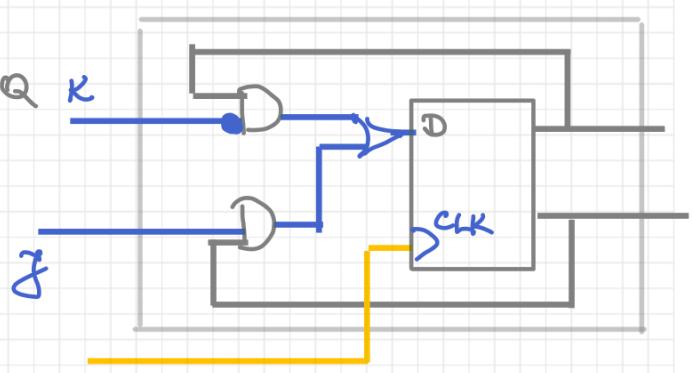
T	Q	Qout	D
0	0	0	0
1	0	1	1
1	1	0	0
0	1	1	1

$$\Rightarrow D = \overline{T}\bar{Q} + \bar{T}Q = T \oplus Q$$



J	K	Q	Q_{next}	D
0	*	0	0	0
1	*	0	1	-
*	1	1	0	0
*	0	1	1	-

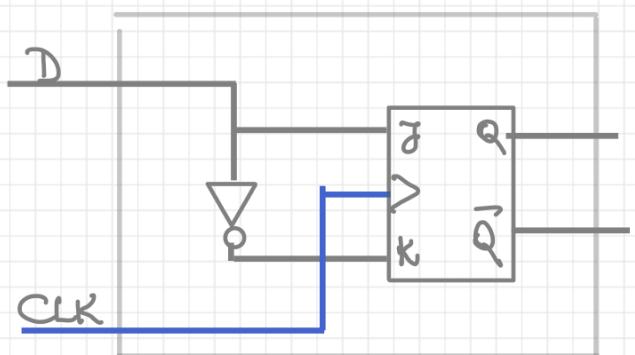
$$\Rightarrow D = \bar{J}Q + KQ$$



D	Q	Q_{next}	J	K
0	0	0	0	*
1	0	1	-1	*
0	1	0	*	1
1	1	1	*	0

$$\Rightarrow J = D$$

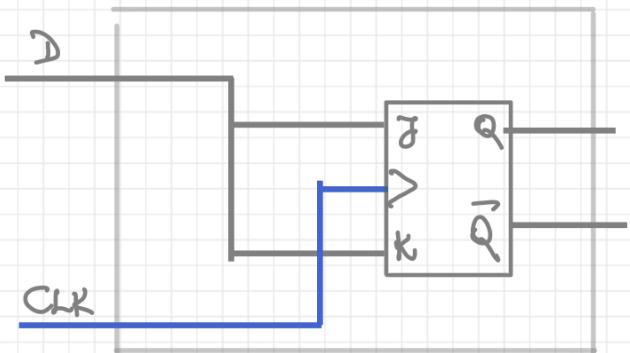
$$K = \bar{D}$$



T	Q	Q_{next}	J	K
0	0	0	0	*
1	0	1	-1	*
1	1	0	*	1
0	1	1	*	0

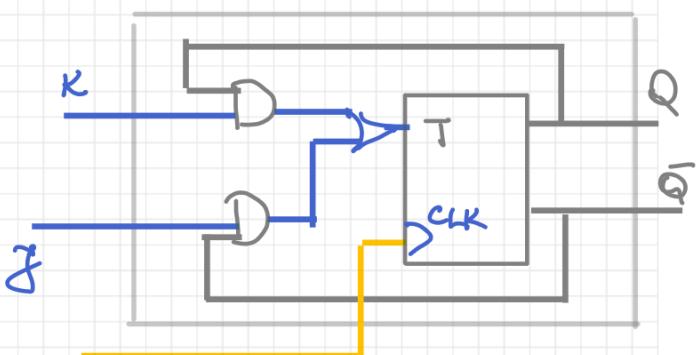
$$\Rightarrow J = D$$

$$K = D$$



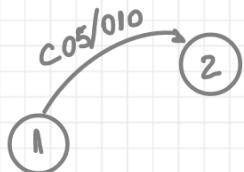
J	K	Q	Q_{next}	T
0	*	0	0	0
1	*	0	1	-
*	1	1	0	-
*	0	1	1	0

$$\Rightarrow T = \bar{J}Q + KQ$$



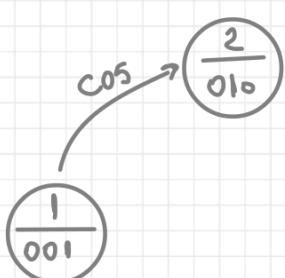
REPREZENTAREA CIRCUITELOR SEQUENȚIALE . CLASIFICARE

- CIRCUITE MEALY : stare următoare și ieșirea la un moment dat depind de STAREA PREZENTĂ și INTRAREA PREZENTĂ



- : fiecare mod se notează cu simbolul stării pe care o reprezintă
- : arcul ce pleacă din mod se notează :
- intrarea care a generat tranziția circuitului
- ieșirea generată în timpul transiției

- CIRCUITE MOORE : ieșirea depinde numai de STAREA CIRCUITULUI

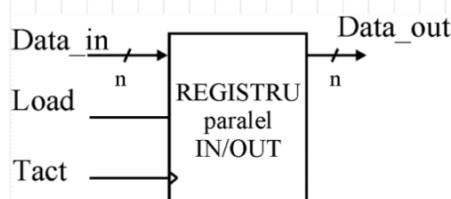


- : stare următoare depinde de starea prezintă
- : modurile diagramei de stări contin simbolul stării și legile
- : arcul care menține aceeași intrare care a generat tranziția

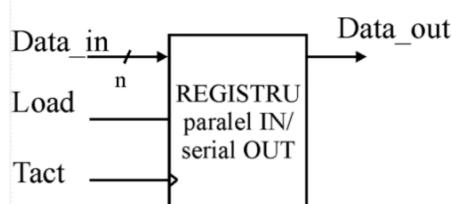
REGISTRE

- Reprezentă o colecție / grupare de n bitabile
- nr. maxim de valori a unui register pe n biti - 2^n val. binare
- folosit pentru memorarea unui cuvânt de date / unei stări curente a sistemului

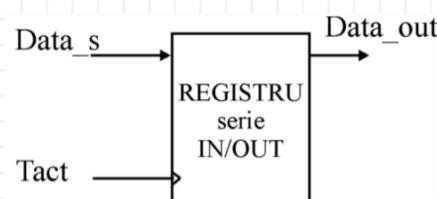
CLASIFICARE



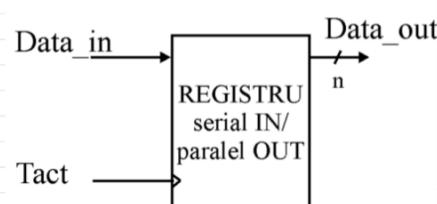
PiPO



PiSO



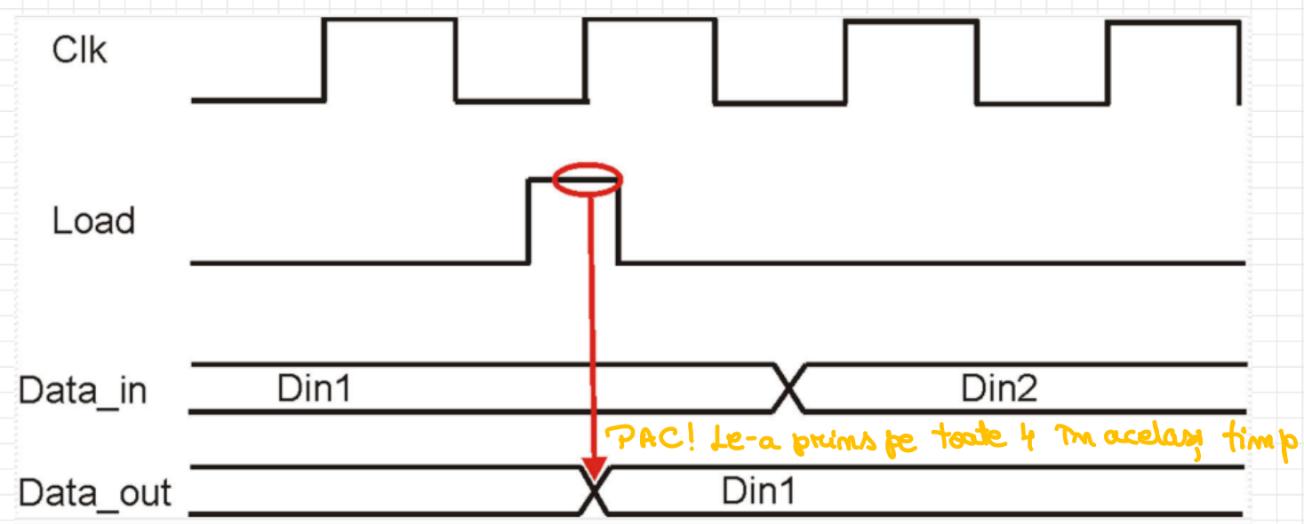
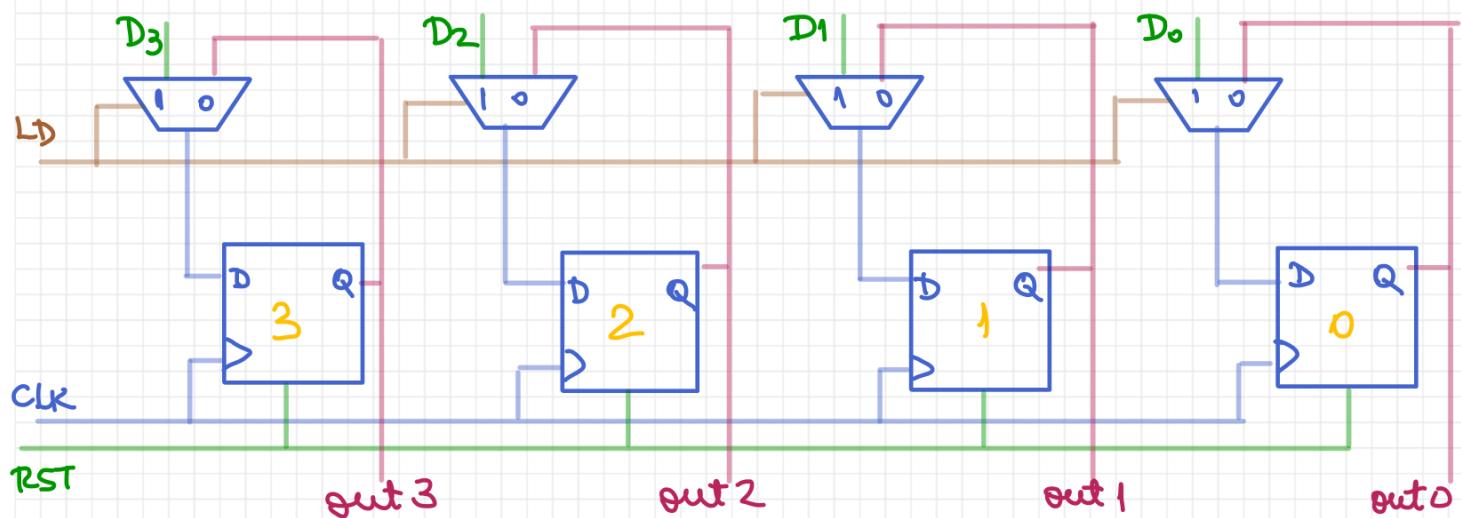
SiSO



SiPO

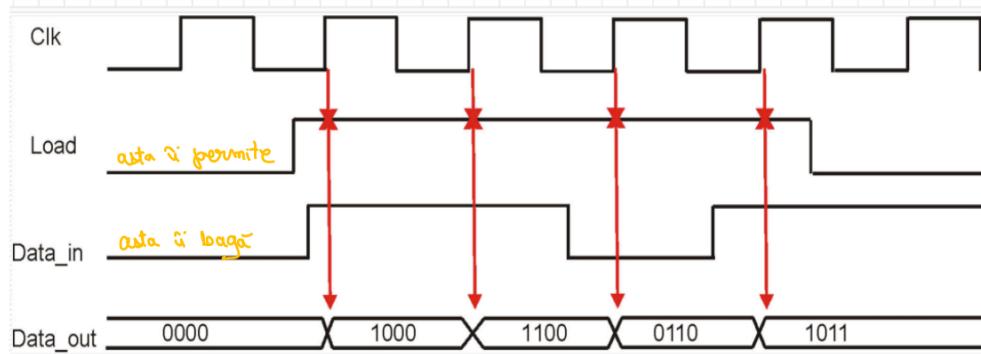
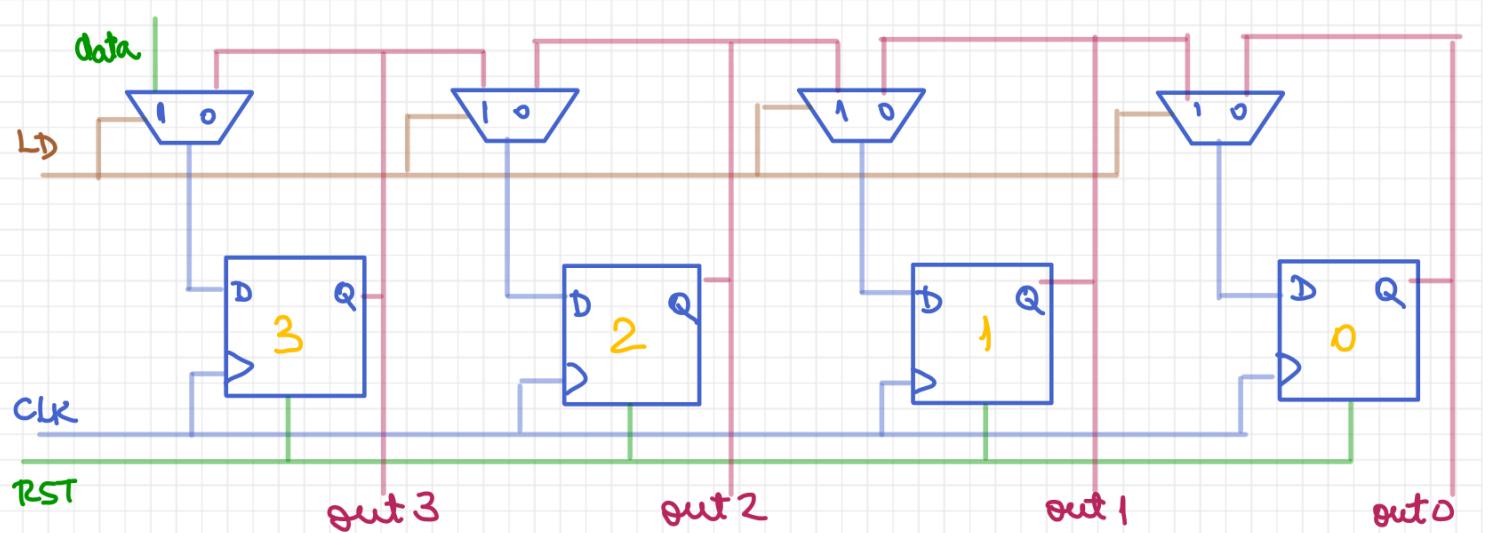
REGISTRU CU INTRARE PARALELĂ - IEŞIRE PARALELĂ

- prezentă semnal de LOAD
 - la fiecare front crescător, valoarea registrului se actualizează cu Data In, dacă este activ semnalul de LD :
- | | |
|----|---------|
| LD | Q-ură |
| 0 | Q |
| 1 | Data In |



REGISTRU CU INTRARE SERIALĂ - IEŞIRE PARALELĂ

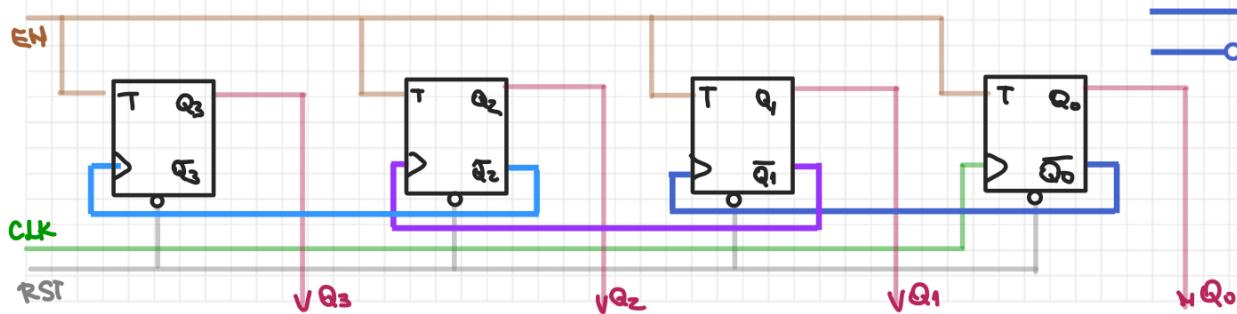
- apare funcția de shiftoare în interiorul registrului
- datele se introduc serial în registru \Rightarrow singură intrare de date
- la fiecare activare a semnalului de LD, datele se deplasează în cadrul registrului (shift)
- incarcarea a n biti necesită n cicluri de CLK



NUMĂRĂTOARE (COUNTERE)

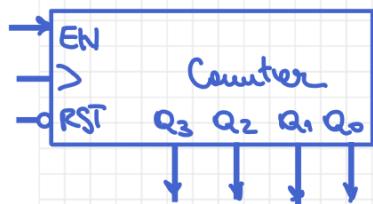
- circuite secvențiale SINCRONE AUTONOME (multimea înțărărilor vidă), care folosesc (acoperă) o secvență de stări impuse de proiectant
- de regulă este initializat cu „0”, după care, la fiecare impuls de numărat, comută într-o nouă stare
- caracterul ASINCRON al unui număratore este dat de faptul că impulsul de tact nu comandă simultan toate bistabilele număratoreului
- J counterie ↑, ↓ sau haversibile (↔)

NUMĂRĂTOR ASINCRON



NUMĂRĂTOR SINCRON

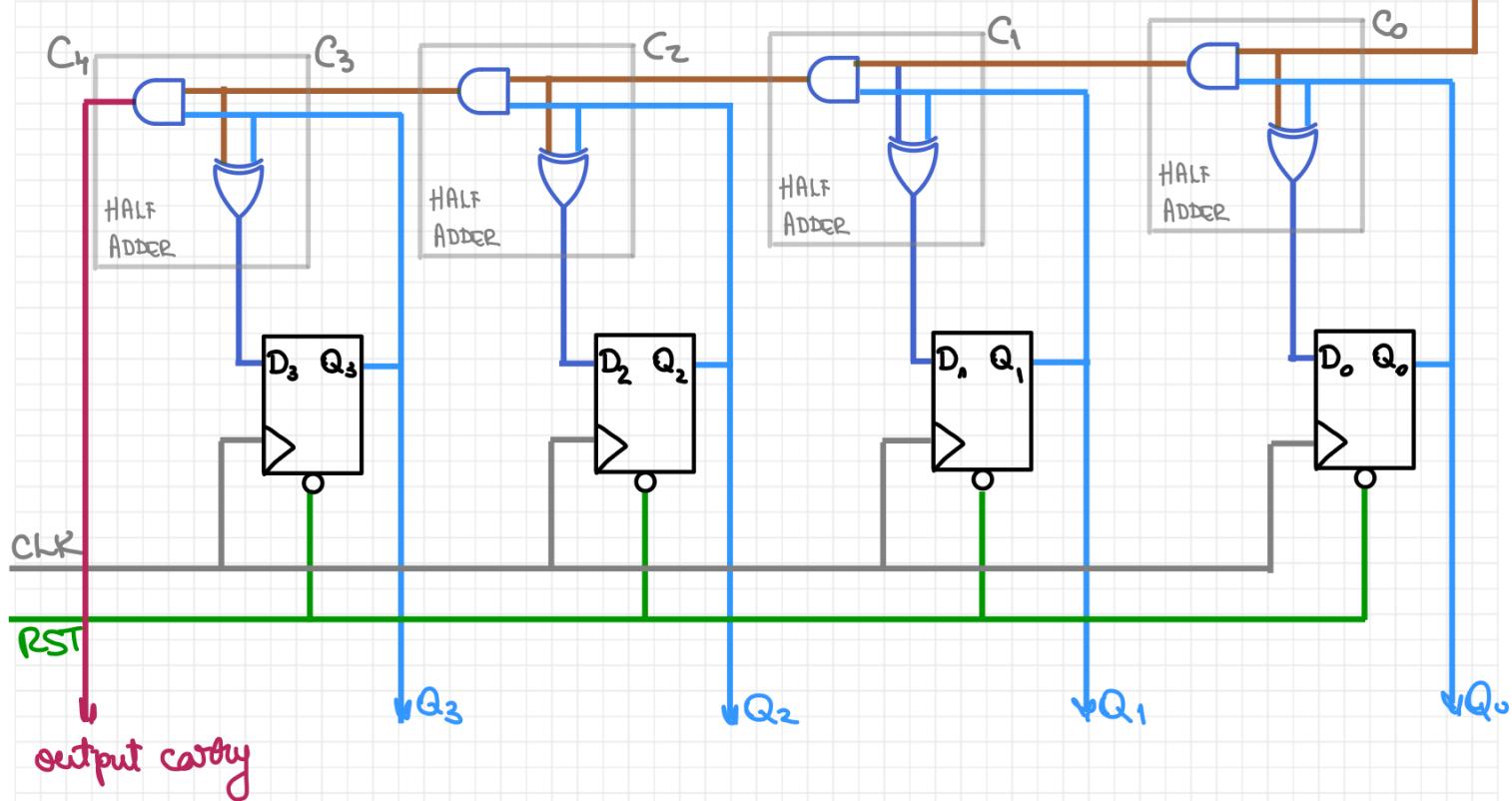
- incrementarea și / sau decrementarea continuă când primesc semnal de activare



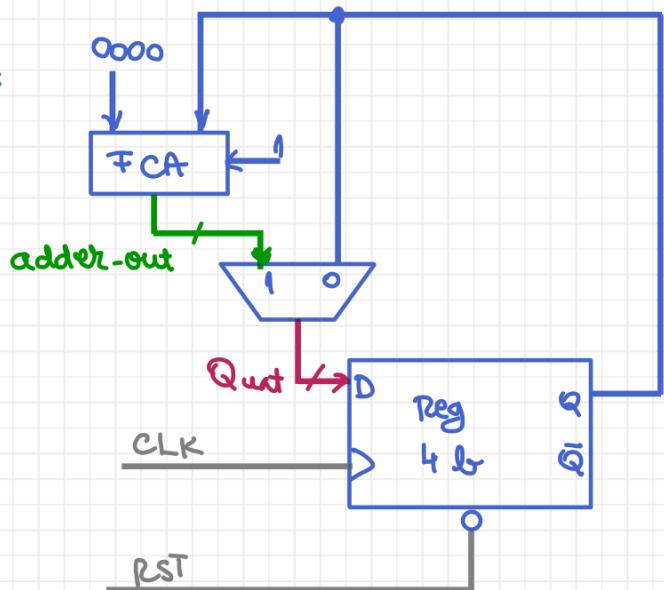
E	Operatie
0	No Change
1	Count

Qi	Ci	Di	Ci+1
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

EN



- Simplificat:



Ex Realizati un numarator folosind FF-wri de tip J-K, care numara după următoarea secvență: $0 \rightarrow 2 \rightarrow 6 \rightarrow 4 \rightarrow 5 \rightarrow 4$

- tab. excit pt. J-K:

Q	Q-uit	J	K
0	0	0	x
0	1	1	x
1	0	x	1
1	1	x	0

- tab. stare:

STARE	Q ₂	Q ₁	Q ₀	J ₂	K ₂	J ₁	K ₁	J ₀	K ₀
0	0	0	0	0	x	1	x	0	x
2	0	1	0	1	x	x	0	0	x
6	1	1	0	x	0	x	0	1	x
4	1	1	1	x	0	x	1	x	0
5	1	0	1	x	0	0	x	x	1
4	1	0	0	x	1	0	x	0	x

- minimizare:

	Q ₁	Q ₀₀₀	01	11	10
J ₀	0	0	x 1	x 3	0 2
	1	0	4	x 5	x 7 1 6

$$J_0 = Q_2 Q_1$$

	Q ₁	Q ₀₀₀	01	11	10
J ₁	0	1 0	x 1	x 3	x 2
	1	0 4	0 5	x 7	x 6

$$J_1 = \overline{Q}_2$$

	Q ₁	Q ₀₀₀	01	11	10
J ₂	0	0 0	x 1	x 3	1 2
	1	x 4	x 5	x 7	x 6

$$J_2 = Q_1$$

	Q ₁	Q ₀₀₀	01	11	10
K ₀	0	x 0	x 1	x 3	x 2
	1	x 4	1 5	0 7	x 6

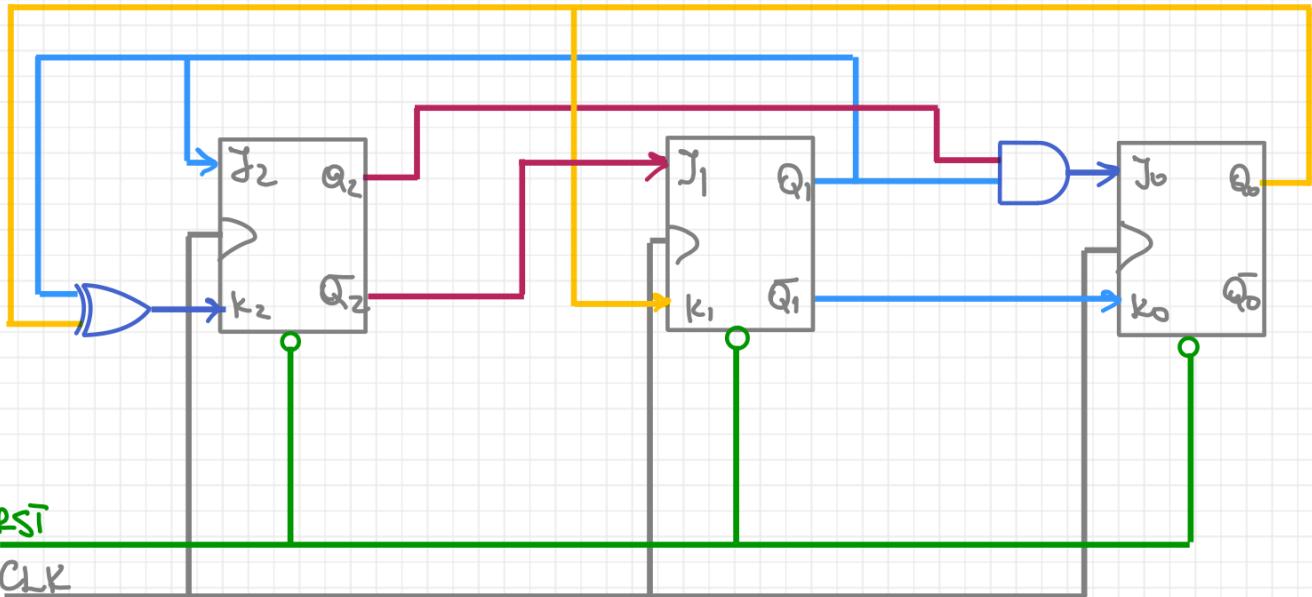
$$K_0 = \overline{Q}_1$$

	Q ₁	Q ₀₀₀	01	11	10
K ₁	0	x 0	x 1	x 3	0 2
	1	x 4	x 5	1 7	0 6

$$K_1 = Q_0$$

	Q ₁	Q ₀₀₀	01	11	10
K ₂	0	x 0	x 1	x 3	x 2
	1	1 4	0 5	0 7	0 6

$$K_2 = \overline{Q}_0 \overline{Q}_1$$

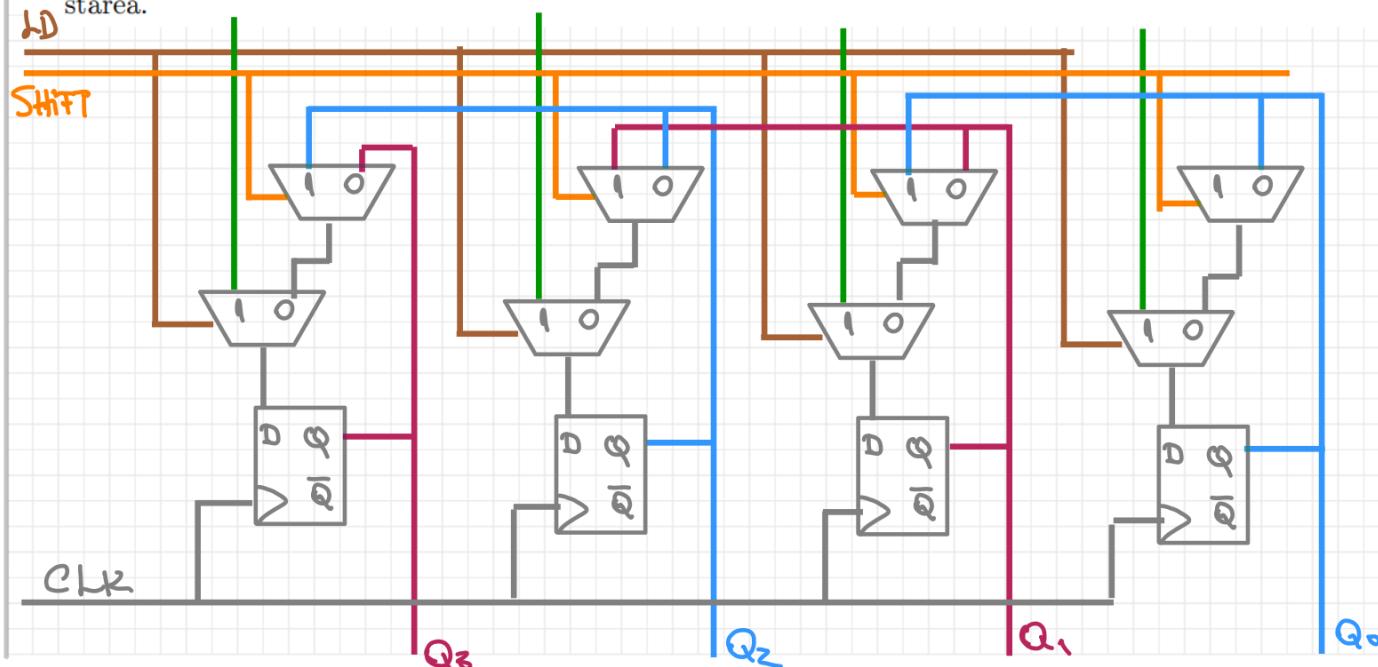


EXERCITII

1. Un registru de 4 biți având inițial conținutul 1011 este deplasat spre stânga secvențial, primind pe intrarea de date serială secvența 1010111. Precizați conținutul registrului după fiecare front activ de ceas.

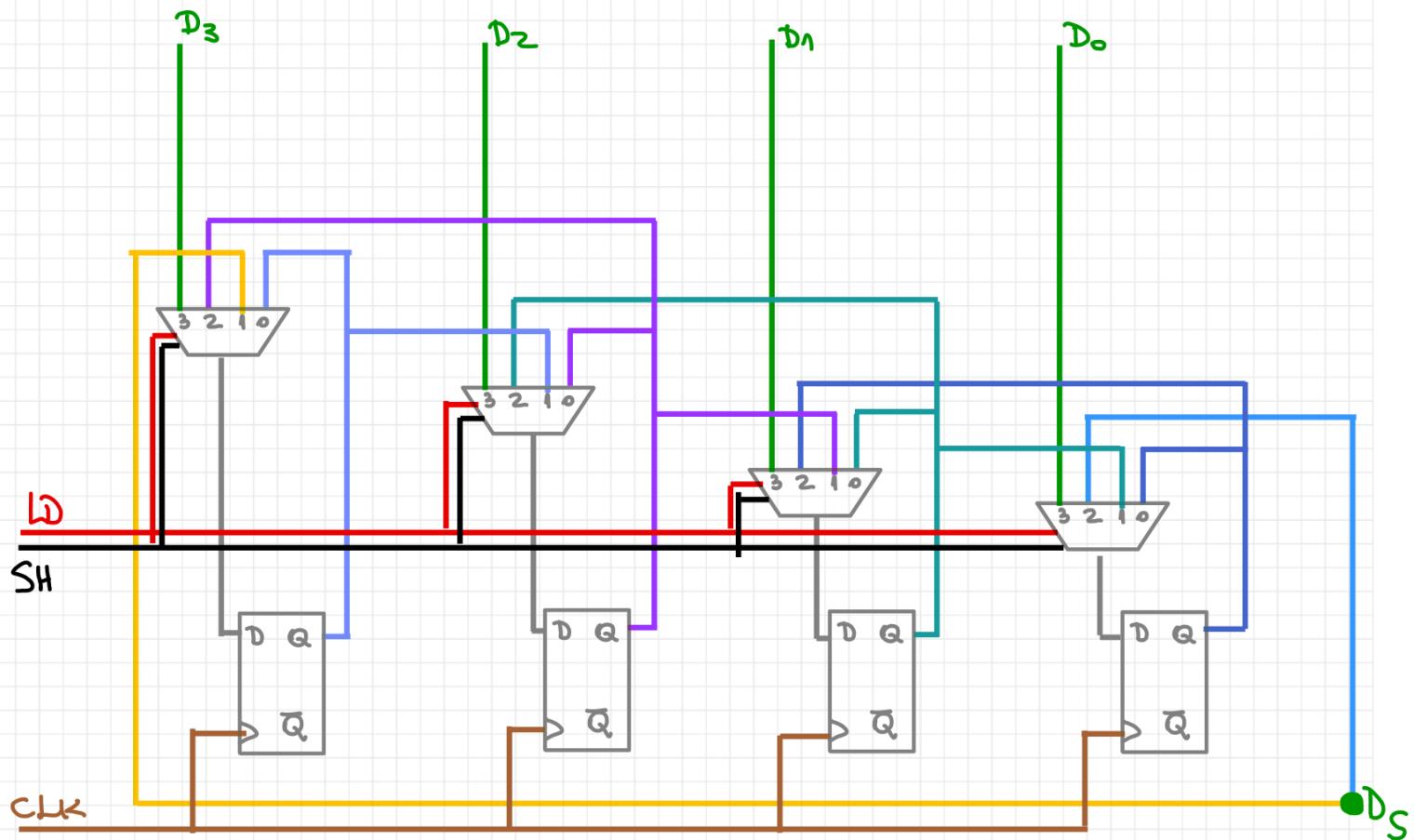
CLK	Intrare	STARE			
		Q ₃	Q ₂	Q ₁	Q ₀
1	1	1	0	1	1
2	0	0	1	1	1
3	1	1	1	0	0
4	0	1	0	0	1
5	1	0	0	1	0
6	1	0	1	0	1
7	1	1	0	1	1
8	-	0	1	1	1

4. Proiectați cu porți și bistabile un registru de 4 biți cu facilități de preîncărcare (dacă $LD = 1$) și deplasare stânga (dacă $LD = 0$ și $SH = 1$). Dacă intrările de control sunt inactive ($LD = SH = 0$) circuitul își păstrează starea.



5. Proiectați cu multiplexoare și bistabile un registru de 4 biți cu facilitățile prezentate în tabelul următor:

<i>LD</i>	<i>Shift</i>	Operăția
<i>S</i> ₁	<i>S</i> ₀	
0	0	păstrează valoarea curentă
0	1	deplasare stânga cu încărcare serială
1	0	deplasare dreapta cu încărcare serială
1	1	încărcare paralelă



CAP. FSM (AUTOMATE CU STĂRÎ FINITE)

• reprezentarea cuadruplicul $\langle S, I, O, f, \phi \rangle$

multimea stărilor multimea intrărilor multimea reprezenterii funcțile fct. mult state funcția pt. ieșire

CLASIFICARE:

• CIRCUITE MEALY: starea următoare și ieșirea la un moment dat

depind de STAREA PREZENTĂ și INTRAREA PREZENTĂ

: fiecare mod se notează cu simbolul stării pe care o reprezintă

: arcul ce pleacă din mod se notează :

intrarea care a generat
transiția circuitului

ieșirea generată
în timpul transiției

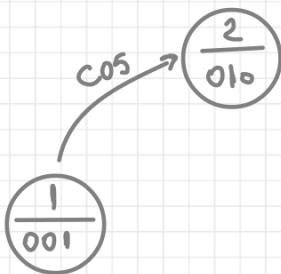
• CIRCUITE MOORE: ieșirea depinde NUMAI de STAREA CIRCUITULUI

: starea următoare depinde de starea prezentă

: modurile diagramei de stări conțin simbolul stării și ieșirile

: arcul care merge doar întrarea care a generat transiția

GRAF

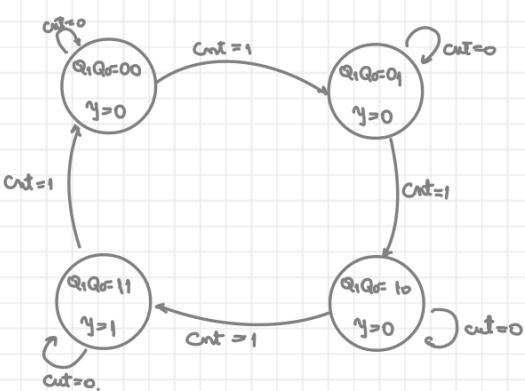


CATEGORIE

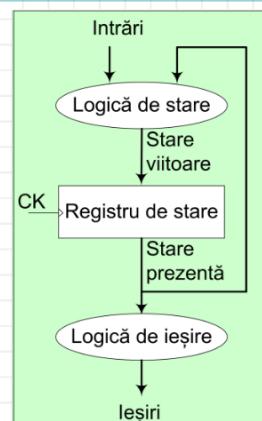
TAB DE STĂRI + IEȘIRI

ORGANIZARE

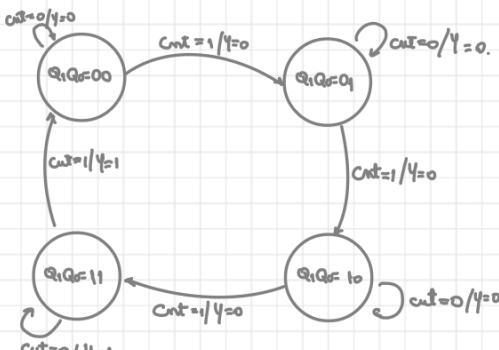
MOORE



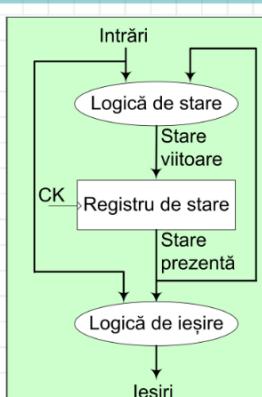
Curr. State	Cnt	State - viitor	Output	
Q1	Q0	Q1	Q0	
0	0	0	0	0
		1	0	1
0	1	0	0	1
		1	1	0
1	0	0	1	0
		1	1	1
1	1	0	1	1
		1	0	0



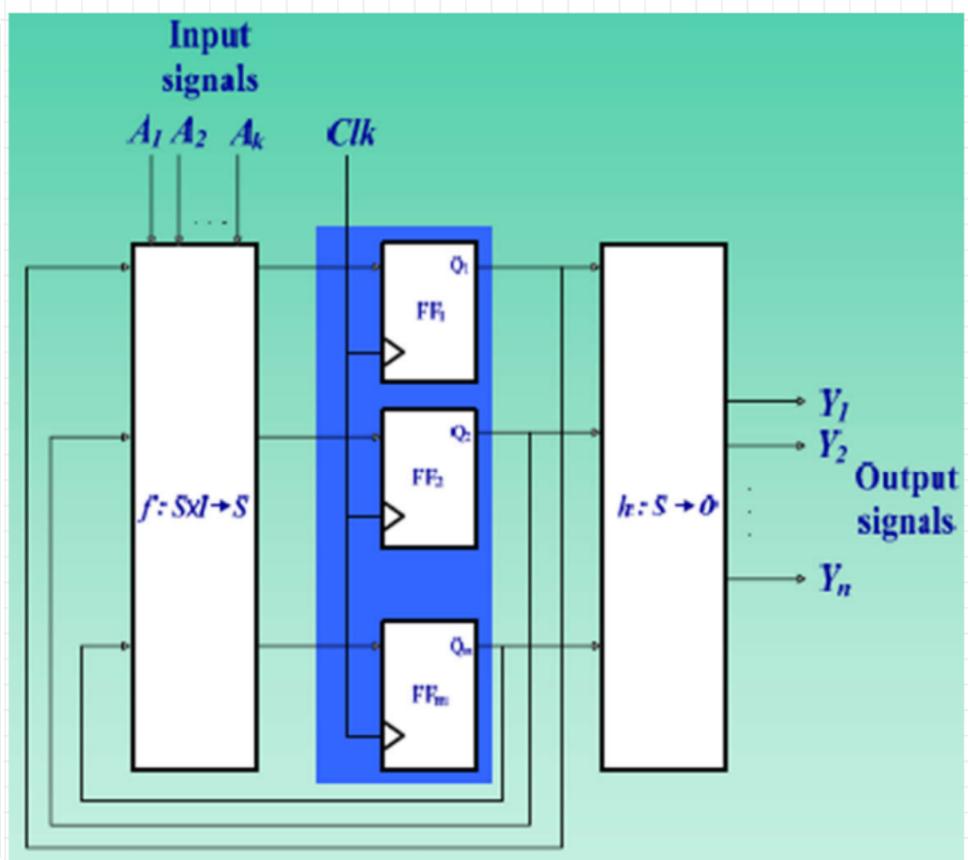
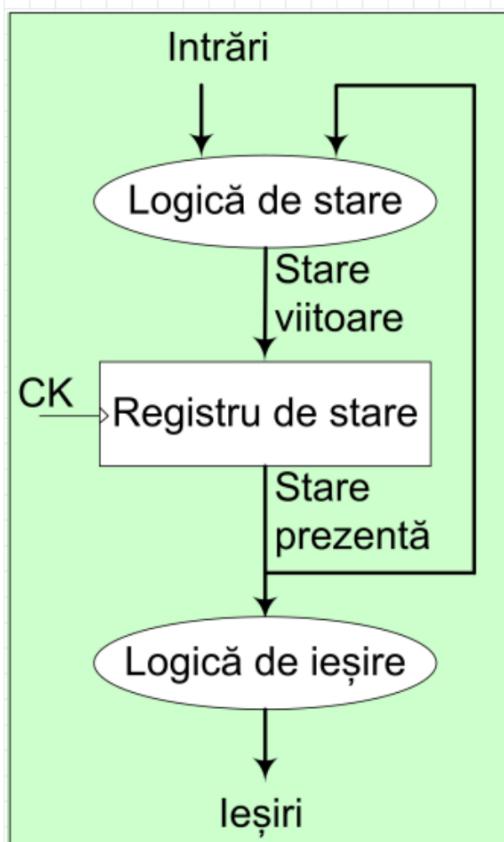
MEALY



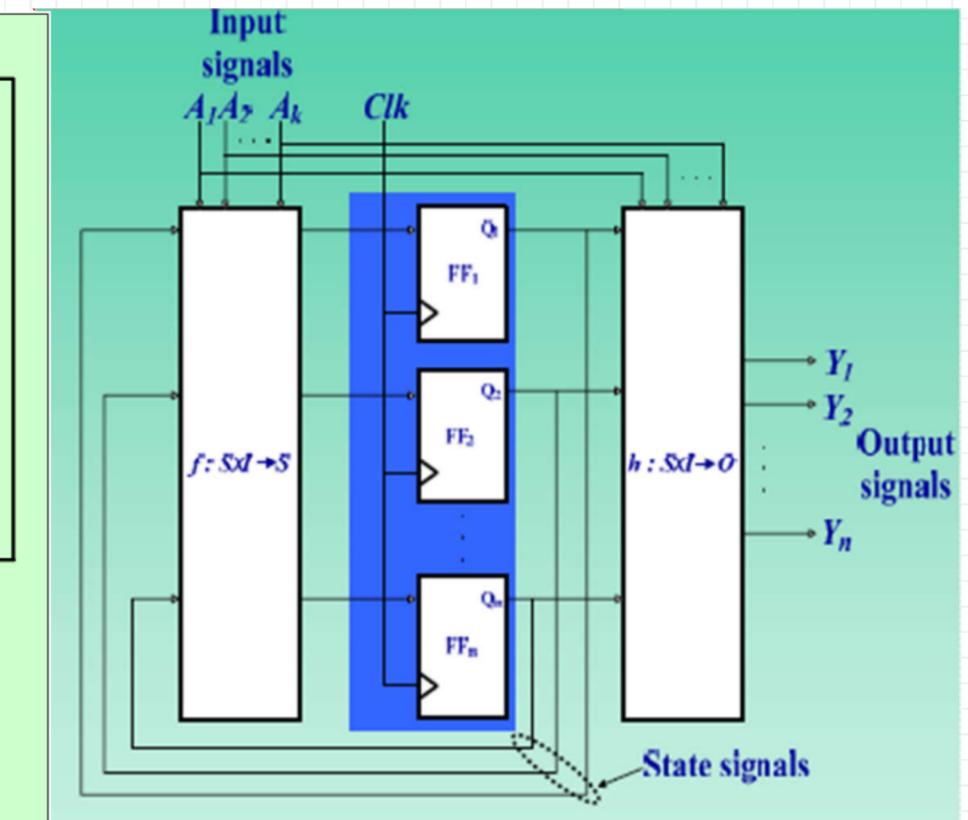
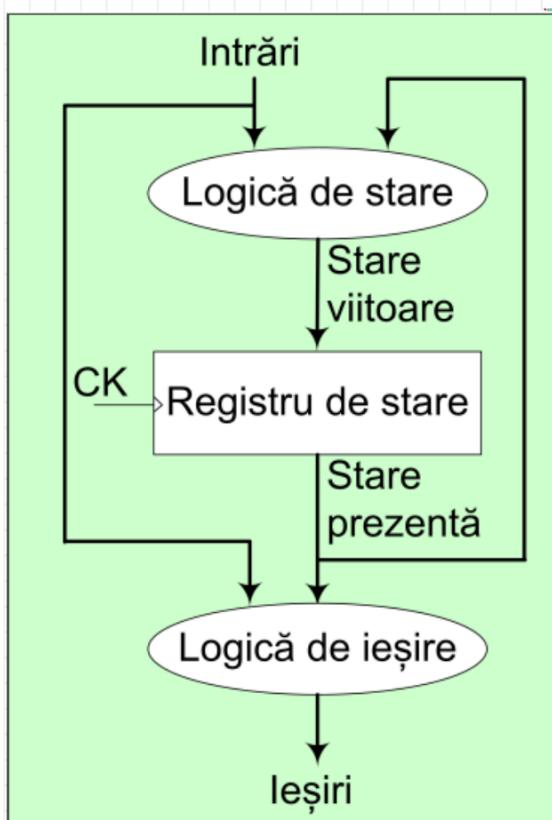
Curr. State	Cnt	State - viitor
Q1	Q0	Q1
0	0	00 / 0
		1 01 / 0
0	1	0 01 / 0
		1 10 / 0
1	0	0 10 / 0
		1 11 / 0
1	1	0 11 / 1
		1 00 / 1



IMPLEMENTARE FSM MOORE



IMPLEMENTARE FSM MEALY



ETAPELE REALIZĂRII UNUI SISTEM SEVENTIAL

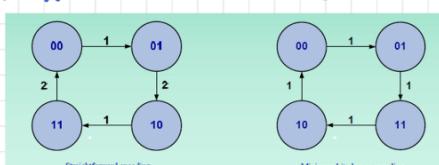
- diagrama de stare (graful)
- tabel stare următoare / ieșiri
- minimizarea stăriilor diagramei (VK)
- codificarea stăriilor / intrărilor / ieșirilor
- ecuația pentru starea următoare / ieșiri
- selecția tipului de FF
- ecuația pentru intrările FF - următor
- design

MINIMIZAREA STĂRIILOR

- încercăm să reducem stările , bazându-ne pe comportamentul echivalent: aceeași secvență de valori de ieșire pentru același secvență de vectori de intrare
- 2 stări sunt evidențial echivalente ($S_j \equiv S_k$) dacă și numai dacă:
 - au comportament echivalent: $f(S_j, i) = f(S_k, i)$
 - au aceeași stare următoare pt. toate secvențele de intrare:
$$f(S_j, i) = f(S_k, i)$$
- 2 stări sunt echivalente ($S_j \equiv S_k$) dacă și numai dacă:
 - au comportament echivalent: $f(S_j, i) = f(S_k, i)$
 - au stări următoare diferite , dar acestea sunt echivalente

CODIFICAREA STĂRIILOR

• NUMĂR MINIM DE TRANZIȚII:



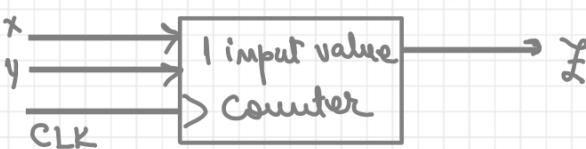
- Stările sunt astfel codificate încât să fie minimizeze tranzitiiile bitilor registrului de stare;
- Fiecare arc i se alocă un cost egal cu numărul de biți ai registrului de stare care diferă la tranzitia dintre stări;
- Se minimizează suma costurilor la parcurgerea grafului

• ADIACENȚĂ PE BAZĂ DE PRIORITATE

- Codificări adiacente pentru stările:
 - destinație comună
 - Sursă comună
 - Ieșire comună
- Next state va apărea în căsuțe adiacente în K-map;

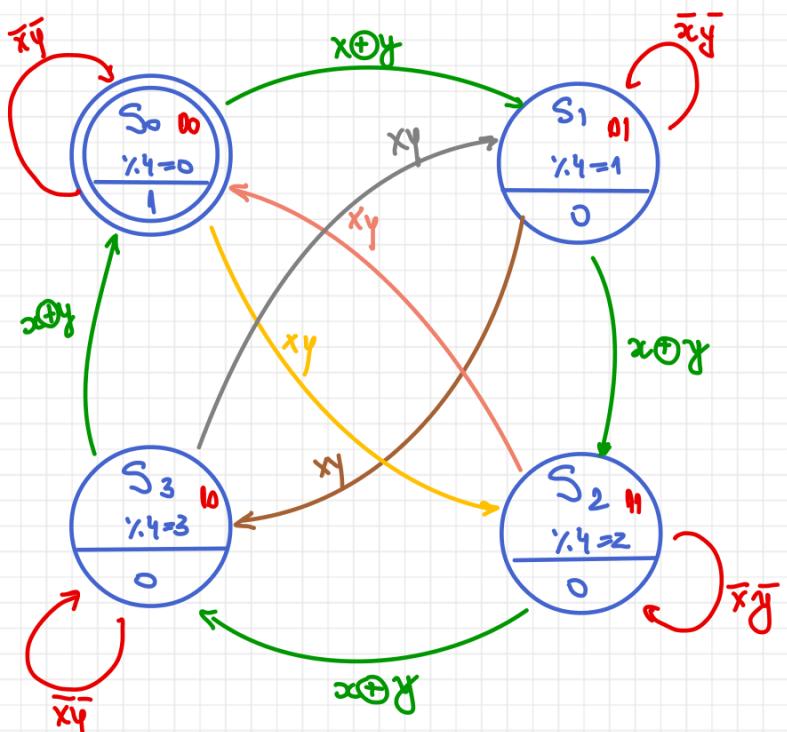
- ONE-HOT:
 - Fiecărei stări i se alocă un bit din registrul de stare → registrul de stare are atâtia biți câte stări sunt în diagramă
 - Nu se pretează pentru diagrame cu multe stări;
 - La un moment dat un singur bit (cel corespunzător stării este pe 1)

Ex Realizați design-ul pentru un automat cu stări cu 2 intrari: X și Y și o ieșire Z. Ieșirea este 1 dacă numarul de valori de intrare 1 pentru X și Y de la reset este multiplu de 4, și 0 în caz contrar.

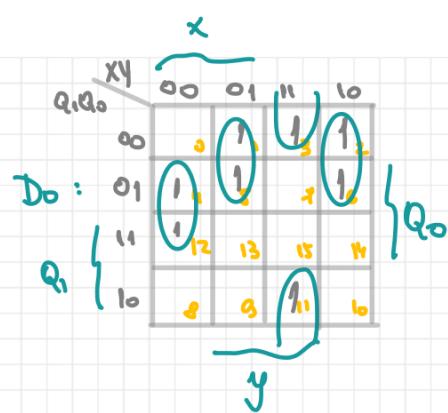
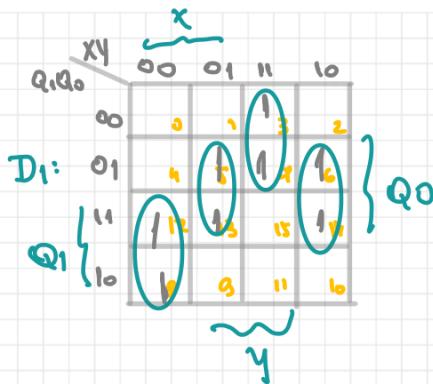


avem 4 stări: $S_0 \div S_3$

STAREA Q1 Q0	SEMANTICĂ	X	Y	D ₁	D ₀	STATE-NXT	Z
S ₀ 00	am 0 de 1	0	0	S ₀	00	00	1
		0	1	S ₁	01	01	0
		1	0	S ₁	01	01	0
		1	1	S ₂	11	11	0
S ₁ 01	am 1 de 1	0	0	S ₁	01	01	0
		0	1	S ₂	11	11	0
		1	0	S ₂	11	11	0
		1	1	S ₃	10	10	0
S ₂ 11	am 2 de 1	0	0	S ₂	11	11	0
		0	1	S ₃	10	10	0
		1	0	S ₃	10	10	0
		1	1	S ₀	00	00	1
S ₃ 10	am 3 de 1	0	0	S ₃	10	10	0
		0	1	S ₀	00	00	1
		1	0	S ₀	00	00	1
		1	1	S ₁	01	01	0



STAREA Q ₁ Q ₀	00	01	11	10	Z
00	00	01	11	01	1
01	01	11	10	11	0
11	11	10	00	10	0
10	10	00	01	00	0



$$\Rightarrow D_1 = Q_1 \cdot \bar{x} \cdot \bar{y} + Q_0 \cdot \bar{x} \cdot y + \bar{Q}_1 \cdot x \cdot y + Q_0 \cdot x \cdot \bar{y}$$

$$= Q_0 (\bar{x} \cdot y + x \cdot \bar{y}) + Q_1 \cdot \bar{x} \cdot \bar{y} + \bar{Q}_1 \cdot x \cdot y$$

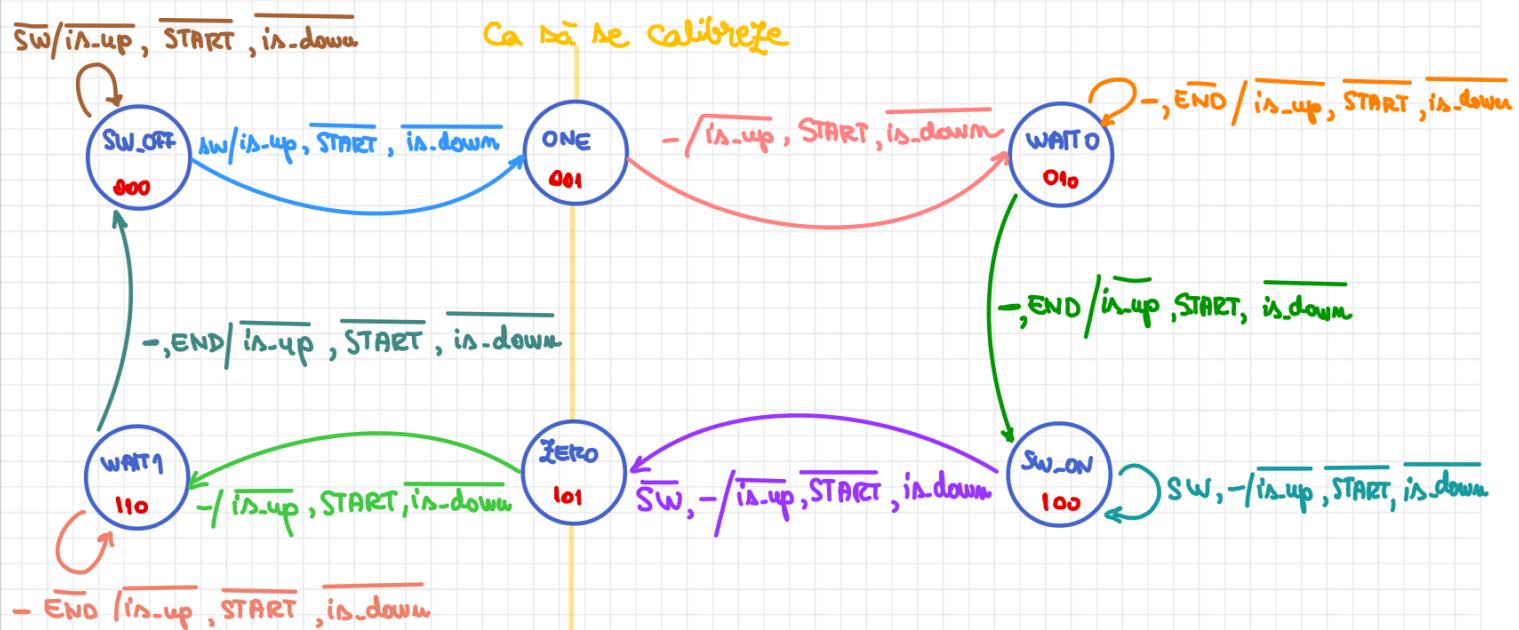
$$= Q_0 \cdot (x \oplus y) + Q_1 \cdot \bar{x} \cdot \bar{y} + \bar{Q}_1 \cdot x \cdot y$$

$$\Rightarrow D_0 = Q_0 \cdot \bar{x} \cdot \bar{y} + \bar{Q}_1 \cdot \bar{x} \cdot y + \bar{Q}_0 \cdot x \cdot y + \bar{Q}_1 \cdot x \cdot \bar{y}$$

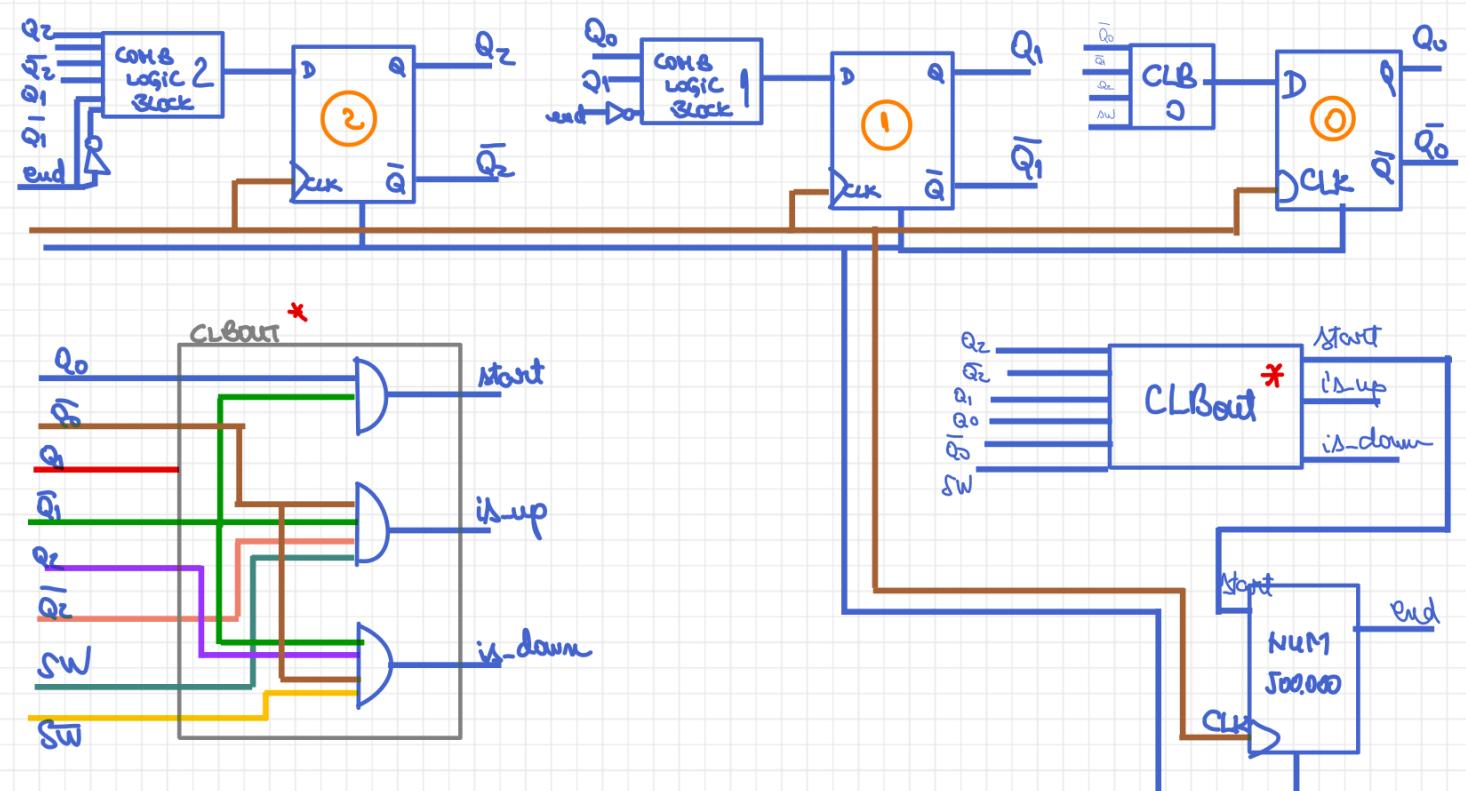
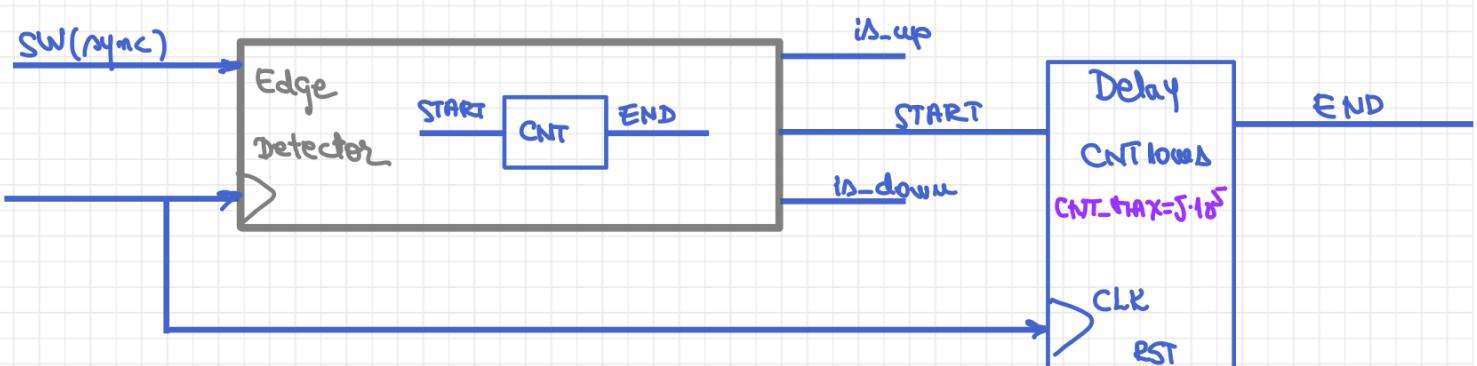
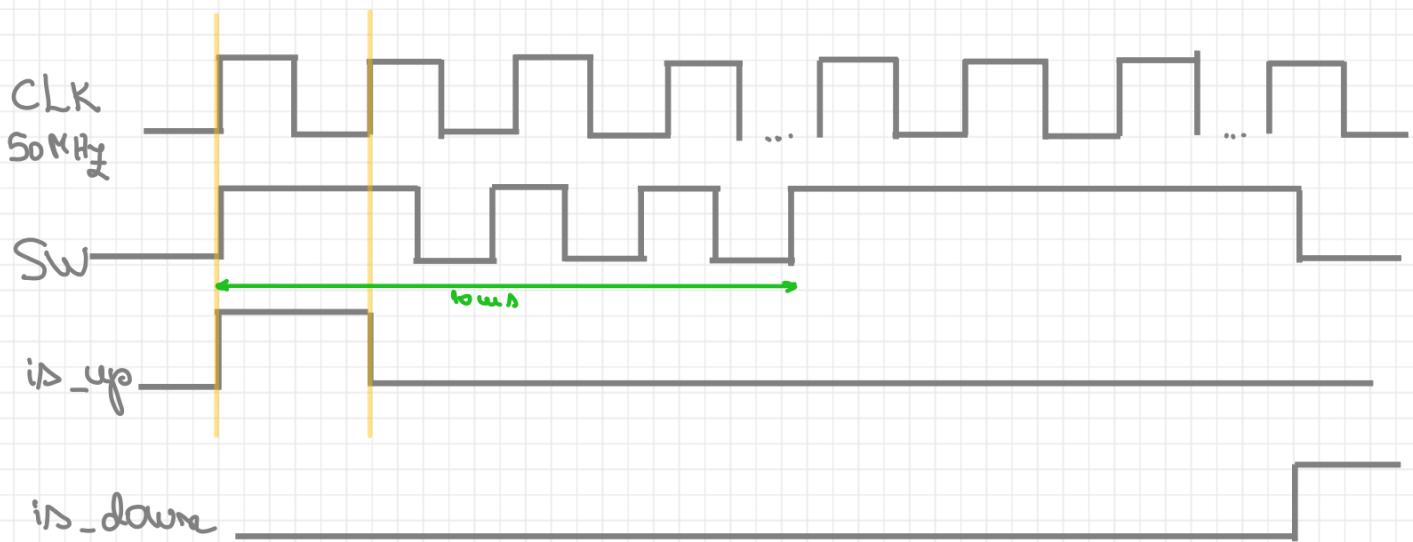
$$= \bar{Q}_1 (x \oplus y) + Q_0 \cdot \bar{x} \cdot \bar{y} + \bar{Q}_0 \cdot x \cdot y$$

• de aici cîcă ar fi temă. Să îi urâm pe la probă!

Ex) Simoarenițarea PW de la „Omulet” cu clk de la FPGA;



• is-up by is-down sum edge detectors



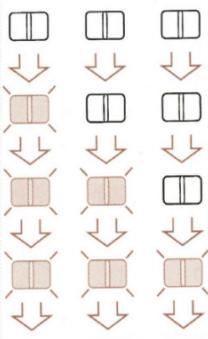
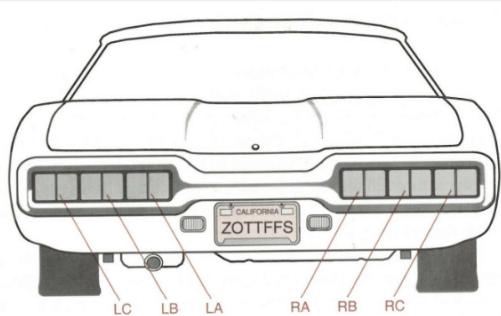
STARE CURENTĂ	INTRĂRI			STARE URMATOARE = (pt D-wii) val intrări			IESIRI'				
	Q ₂	Q ₁	Q ₀	SW	END	D ₂ (Q ₂ ^{NXT})	D ₁ (Q ₁ ^{NXT})	D ₀ (Q ₀ ^{NXT})	Start	IS-up	IS-down
SW_OFF	0	0	0	0	0	0	0	0	0	0	0
				0	1	0	0	0	0	0	0
				1	0	0	0	0	1	0	1
				1	1	0	0	0	1	0	0
ONE	0	0	1	*	*	0	1	0	1	0	0
WAIT0	0	1	0	*	0	0	1	0	0	0	0
				*	1	1	0	0	0	0	0
				*	1	1	0	0	0	0	0
SW_ON	1	0	0	0	*	1	0	1	0	0	1
				1	*	1	0	0	0	0	0
ZERO	1	0	1	*	*	1	1	0	1	0	0
WAIT1	1	1	0	*	0	1	1	0	0	0	0
				*	1	0	0	0	0	0	0
	0	1	1	*	*	d	d	d	d	d	d
	1	1	1	*	*	d	d	d	d	d	d

⇒ VK cu D₂, D₁, D₀, START, IS-up și IS-down

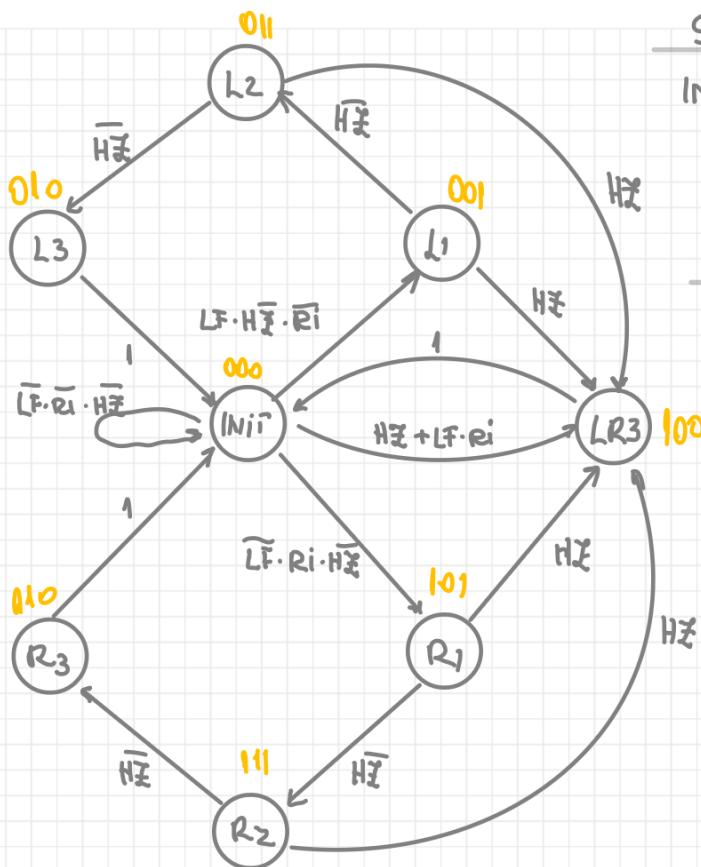
• modelarea ... fiecare după sufltel lui!

ex) FORDU', Moore

- Realizati diagrama de stare pentru sistemul de semnalizare a unei masini Ford.



STARE	LC	LB	LA	RA	RB	RC
INIT	0	0	0	0	0	0
L ₁	0	0	1	0	0	0
L ₂	0	1	1	0	0	0
L ₃	1	1	1	0	0	0
R ₁	0	0	0	1	0	0
R ₂	0	0	0	1	1	0
R ₃	0	0	0	1	1	1
LR ₃	1	1	1	1	1	1



S	Q ₂	Q ₁	Q ₀	Transition	S ^{out}	Q ₂ ^{out}	Q ₁ ^{out}	Q ₀ ^{out}
INIT	0	0	0	$\overline{LF} \cdot \overline{R}_i \cdot \overline{H\bar{Z}}$	INIT	0	0	0
				$\overline{L}_i \cdot \overline{R}_i \cdot \overline{H\bar{Z}}$	L _i	0	0	1
				$\overline{L}_i \cdot \overline{R}_i \cdot \overline{H\bar{Z}}$	R _i	1	0	1
				$H\bar{Z} + \overline{L}_i \cdot \overline{R}_i$	LR ₃	1	0	0
	L ₁	0	0	1	L ₂	0	1	1
	L ₂	0	1	1	LR ₃	1	0	0
	L ₃	0	1	1	LR ₃	1	0	0
	R ₁	1	0	1	R ₂	1	1	1
	R ₂	1	1	1	LR ₃	1	0	0
	R ₃	1	1	0	LR ₃	1	0	0
	INIT	0	0	0	INIT	0	0	0

SAU:

S[st.current]	Q ₂	Q ₁	Q ₀	LEFT	RIGHT	H <bar>Z</bar>	S*[store wum.]	D ₂ Q ₂ *	D ₁ Q ₁ *	D ₀ Q ₀ *
IDLE	0	0	0	0	0	0	IDLE	0	0	0
	0	0	1				LR ₃	1	0	0
	0	1	0				R ₁	1	0	1
	0	1	1				LR ₃	1	0	0
	1	0	0				L ₁	0	0	1
	1	0	1				LR ₃	1	0	0
	1	1	0				LR ₃	1	0	0
	1	1	1				LR ₃	1	0	0

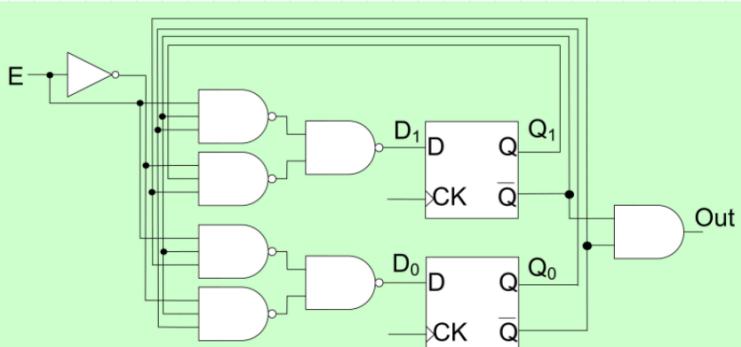
EXERCITII

Analiza circuitelor logice secvențiale constă în *determinarea comportamentului automatului pe baza structurii acestuia* (porți și bistabile). Comportamentul circuitului logic secvențial poate fi descris sub formă de organigramă sau graf de tranziții.

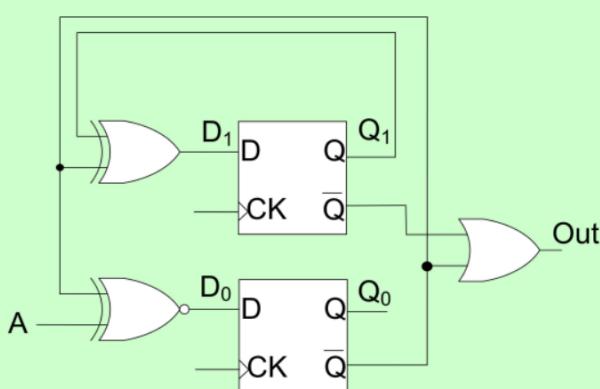
Pentru analiza unui circuit secvențial, se recomandă parcurgerea următoarelor etape:

- Circuitul este un *automat* dacă în structura sa apar bistabile și bucle între ieșirile și intrările acestora.
- Se determină caracteristicile automatului:
 - *Numărul de intrări* se determină prin observarea intrărilor primare (din exteriorul circuitului).
 - *Numărul de ieșiri* se determină prin observarea ieșirilor. Ieșirile pot fi direct din regiszru (un bit de stare este transmis la ieșire) sau prin intermediul unui circuit combinațional (având intrările provenite de la ieșirile bistabilelor și optional de la intrările primare).
 - Determinarea *tipului automatului* se face prin observarea modului în care se generează ieșirile. Dacă există o cale combinațională între o intrare și o ieșire, atunci automatul este de tip Mealy. Dacă toate ieșirile depind exclusiv de stare (ieșirile bistabilelor) atunci automatul este de tip Moore.
 - *Numărul de biți care codifică stările* este egal cu numărul de bistabile prezente în schemă.
 - *Numărul maxim de stări* este egal cu $2^{\text{numărul de biți de cod}}$.

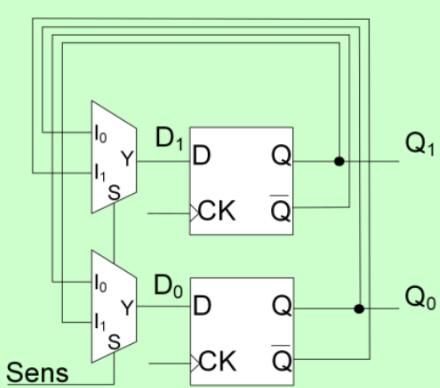
1. Analizați circuitele logice secvențiale prezentate :



Moore - Ieșirea depinde exclusiv de ieșirile bistabilelor



Moore - Mealy



Mealy : Există cale combinațională între intrare și ieșire

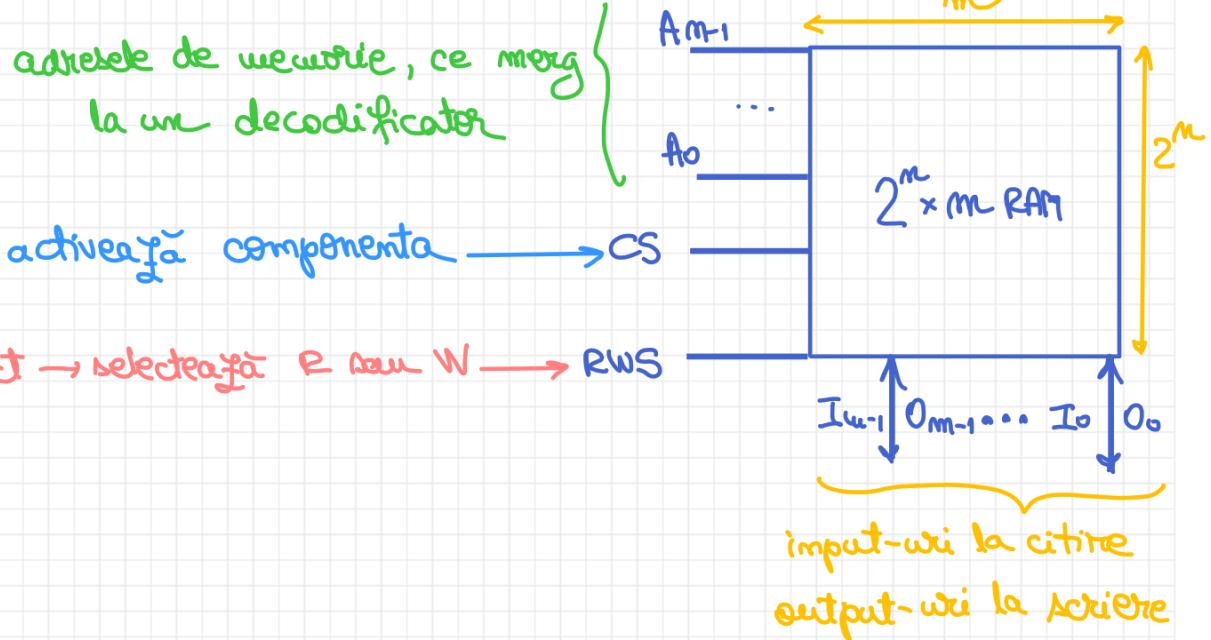
CAP. MEMORII

CLASIFICARE:

- după modul de operare: SCRİERE și CİTİRE: ACCES RANDOM: SRAM DRAM
 - ACCES NON-RANDOM: FIFO LIFO
- NON-VOLATILE SCRİERE și CİTİRE: EPROM EEPROM FLASH
- ROM (nu mai citire) COMB! : măști programate
- după modul de adresare: prim ADRESĂ: SRAM DRAM
 - prim CONȚINUT: accesări pe baza unui tag
- după METRICI : DENSITATEA MEMORIEI ($\text{biti}/\mu\text{m}^2$) și CAPACITATE
 - TIMP DE ACCES (timpul necesar de citire/scrivere)
 - THROUGHPUT (cate date se încarcă/sec.)
- CONSUM DE PUTERE

RANDOM ACCESS MEMORY (RAM) → Memorie cu acces aleator

ADRESA DE MEMORIE		CONTINUT
Binar	Decimal	
00...000	0	10100111...11
00...001	1	0101100...00
00...010	2	0101001..01
...	...	10101011..00
01...000	...	0001010...01
01...001	...	11111111...11
...	...	0001000...01
11...111	$2^n - 1$	01011101...1

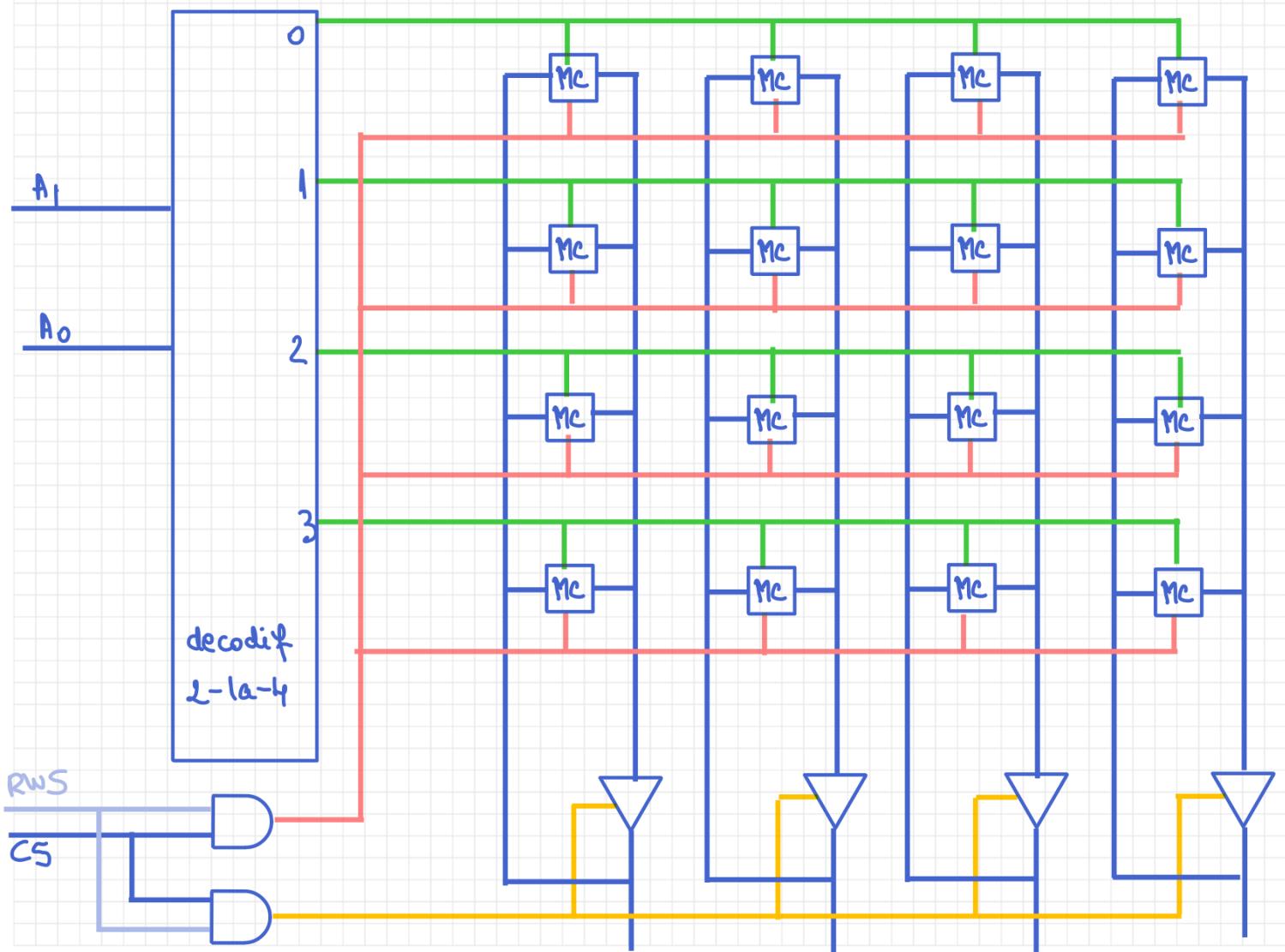


Ex) De cate linii de adresa avem memorie de a accesa o memorie de:

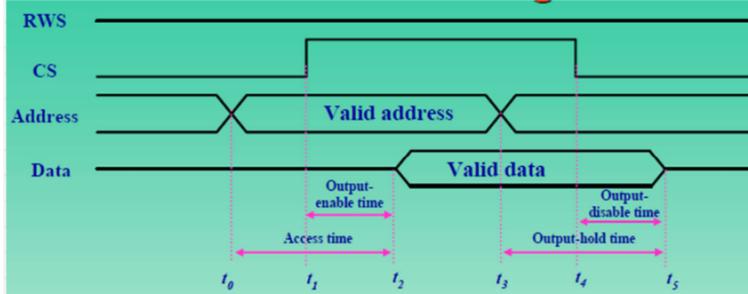
$$: 1 \text{ kbit?} \rightarrow 1 \text{ kbit} = 1024 \text{ b} = 2^{10} \text{ b} \Rightarrow 10 \text{ linii}$$

$$: 64 \text{ Kbiti?} \rightarrow 64 \text{ kb} = 2^6 \cdot 2^{10} \text{ b} = 2^{16} \text{ b} \Rightarrow 16 \text{ linii}$$

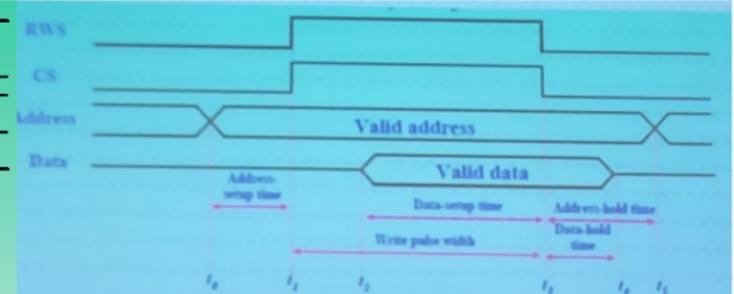
ORGANIZARE:



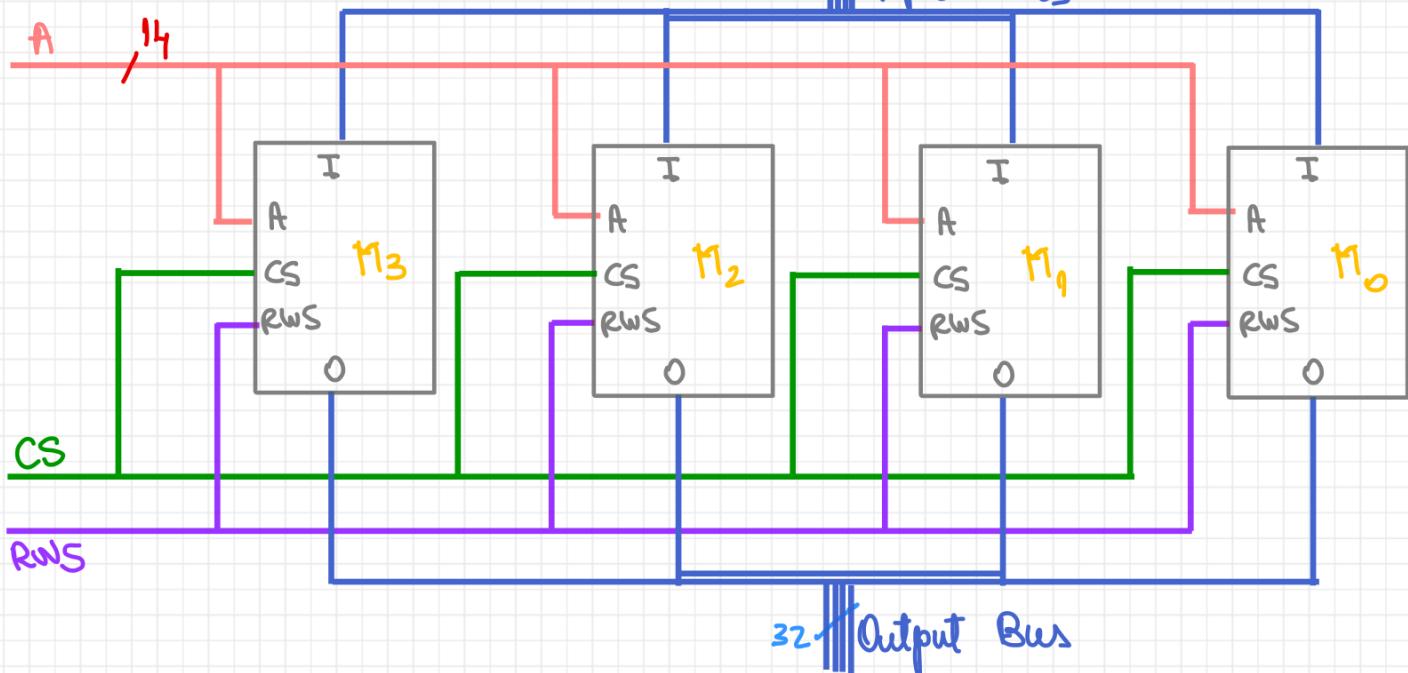
Ciclu RAM CITIRE



Ciclu RAM SCRIRE



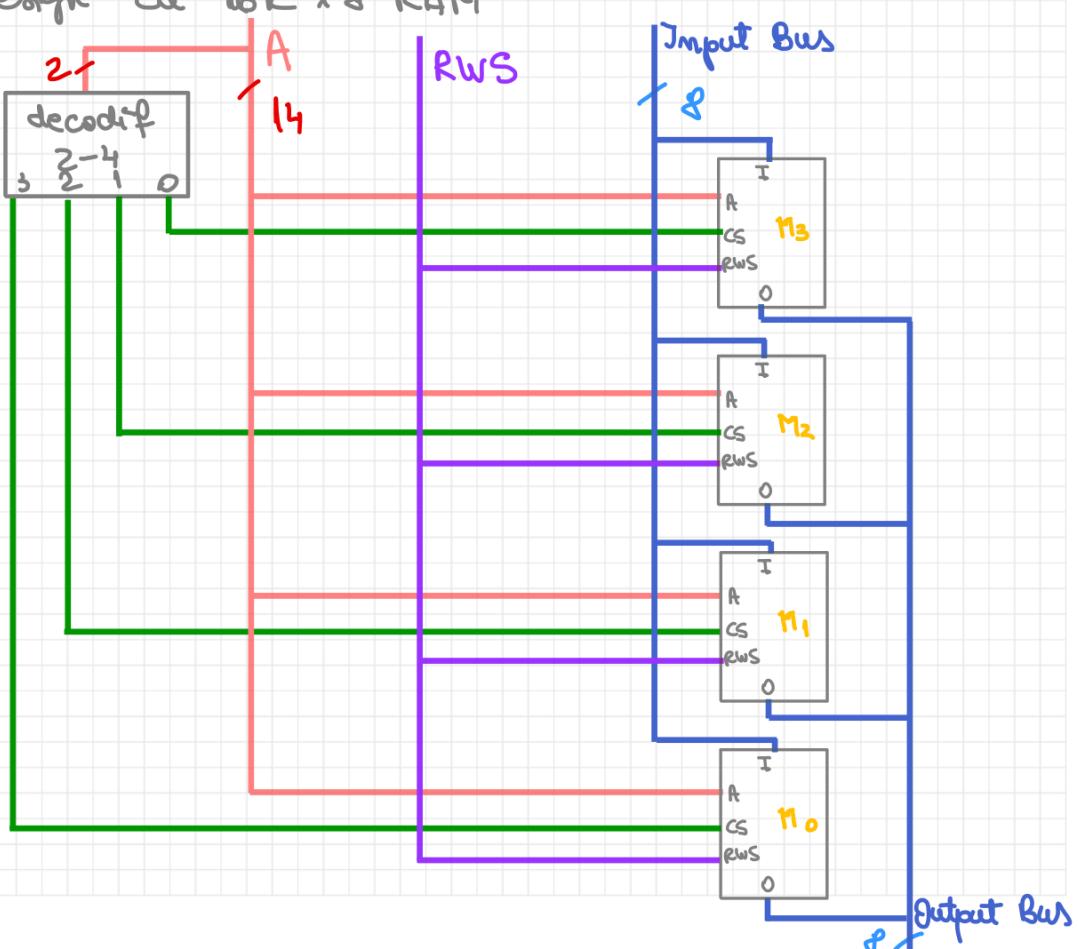
Ex: 16 K x 32 RAM design cu 16 K x 8 RAM //aici intra 32 Input Bus



Ex: 64 K x 8 RAM design cu 16 K x 8 RAM

• obs. că acum nu mai avem continutul apart din 4 memorii, ci memoria e spartă în 4 grupuri de adrese. Rezulta:

GR	A_{15}	A_{14}	A_{13}	\dots	A_0
M_0	0	0	000...00		111...11
M_1	0	1	000...00		111...11
M_2	1	0	000...00		111...11
M_3	0	1	000...00		111...11

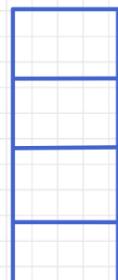


SRAM VS DRAM

- SRAM memorază starea căt timp e alimentat
- DRAM are nevoie de alimentare + refresh (controler mai complex)
- DRAM are densitate mai mare ($1 \text{ transistor/celula}$) în comparație cu SRAM-ul, dar:
 - are memoie de cicle refresh
 - citirea e distructivă, deci trebuie rezervată informația imediat după
- Long story short: DRAM : \oplus : densitate mare
 \ominus : ciclu de refresh \Rightarrow timpuri de acces sunt variabili

Ex: STVA

- h word, m bit push-down stack, cu:
 - m inputuri
 - m outputuri
- Semnale de control : push / pop : $0 \rightarrow \text{push}$
 $1 \rightarrow \text{pop}$
 - : enable
 - : semnale de stare: empty și full



EN	PP	?
0	X	Nimic
1	0	push
1	1	pop

Centroale		PP	EN	Centroale	
S ₁	S ₀			D	E
X	0	X	0	K	0
1	1	0	1	0	1
0	1	1	1	1	1



$$S_1 = EN$$

$$S_0 = \overline{PP} \cdot EN$$

$$D = PP$$

$$E = EN$$

$$EMP = \overline{Q}_2 \cdot \overline{Q}_1 \cdot \overline{Q}_0, FUL = Q_2 \cdot Q_1 \cdot \overline{Q}_0$$

- detector de empty și full

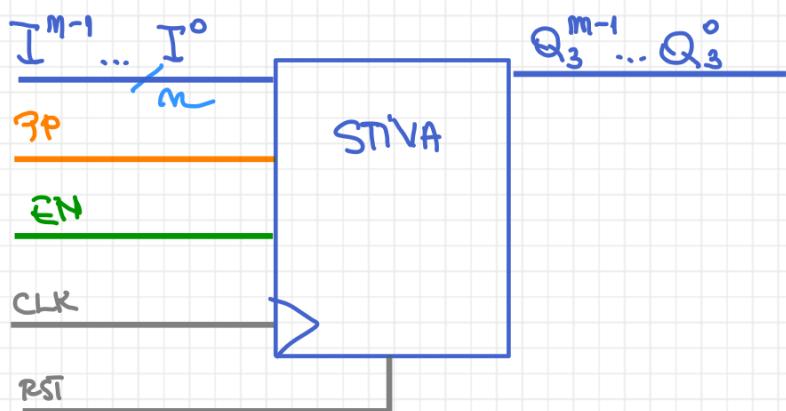
Val	EMP	FUL
000	1	0
001	0	0
010	0	0
011	0	0
100	0	1

• CWR-State State-unt

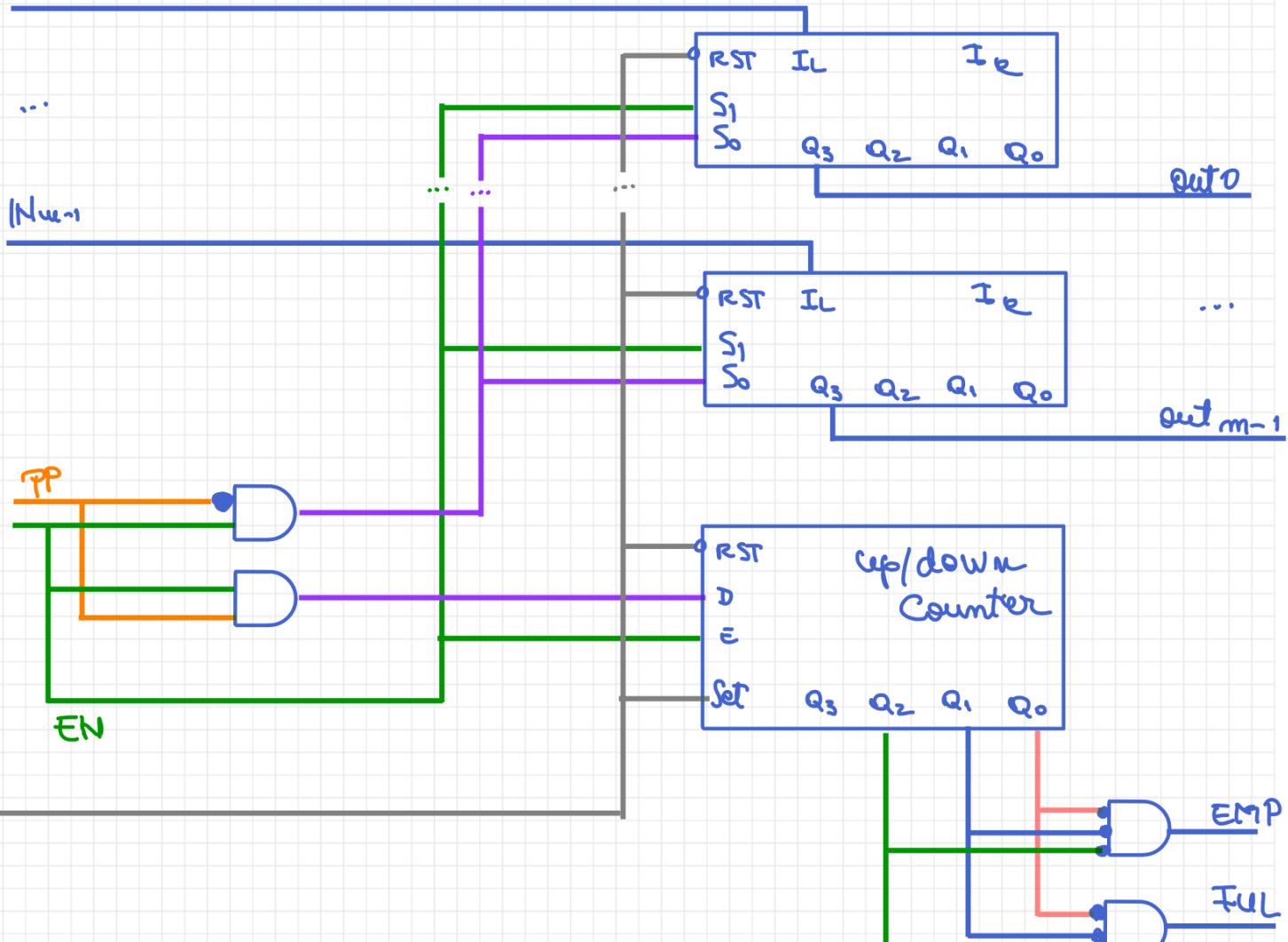
	S_1	S_0	Q_3	Q_2	Q_1	Q_0
write	0	0	Q_3	Q_2	Q_1	Q_0
load	0	1	I_3	I_2	I_1	I_0
\leftarrow	1	0	Q_2	Q_1	Q_0	I_R
\rightarrow	1	1	I_L	Q_2	Q_2	Q_1

• counter: LD EN D-summerator ?

0	0	X	X
0	1	0	++
0	n	0	--
1	x	X	LD



$[N_0]$



Ex: Coafda FIFO : 4 cuvinte

• tab de operatii :

RWS	EN	op	Stare initială	D	E
X	0	no change	S ₁ S ₀	0 0	X 0
0	1	read	0 0	1 1	
1	1	write	1 1	0 1	

$\Rightarrow S_1 = RWS \cdot EN$

$S_0 = RWS \cdot EN *$

$D = EN \cdot \bar{RWS}$

$E = EN$

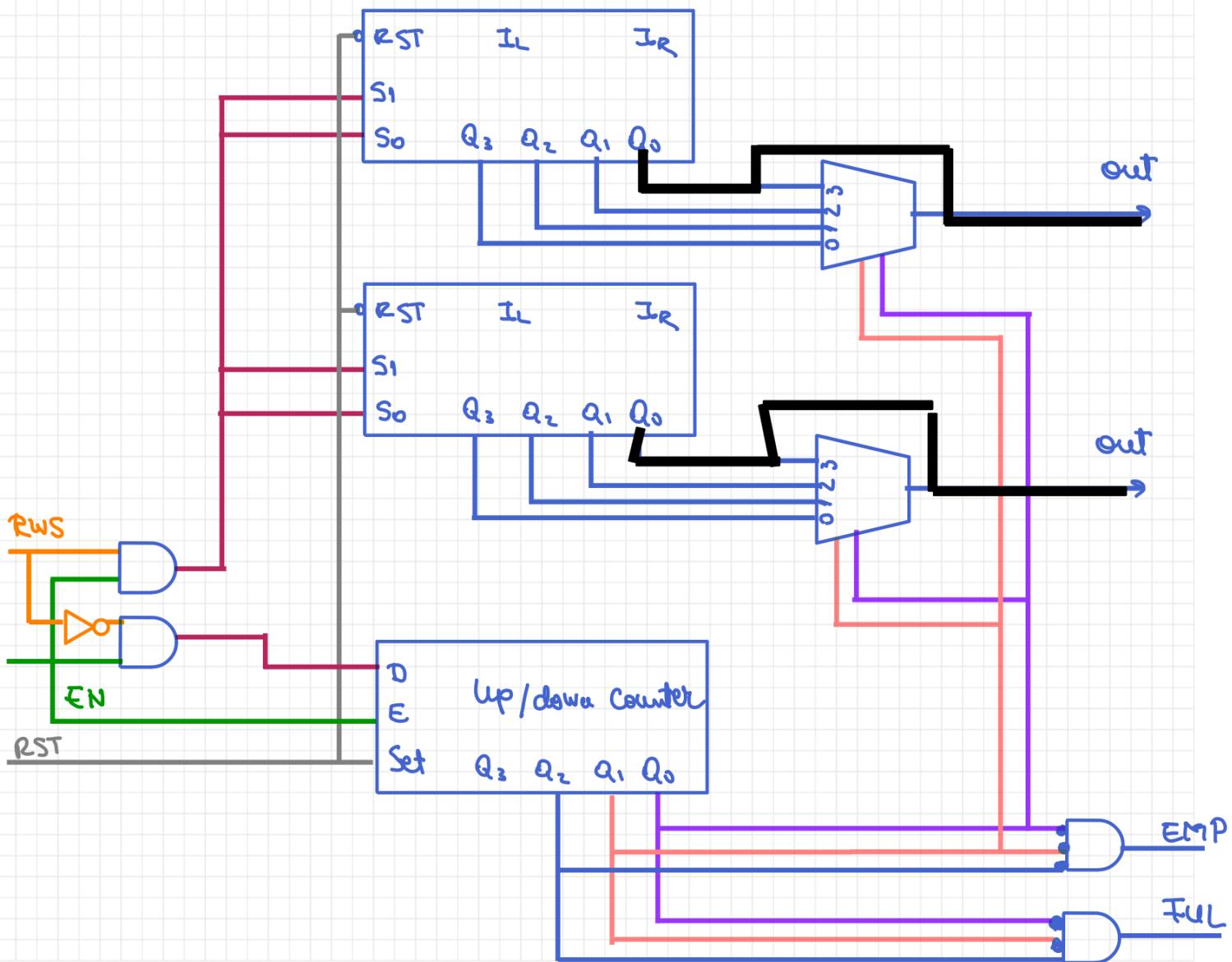
- Reg. de deplasare

Present State		operation	Next state			
S ₁	S ₀		Q ₃	Q ₂	Q ₁	Q ₀
0	0	no change	Q ₃	Q ₂	Q ₁	Q ₀
0	1	load	I ₃	I ₂	I ₁	I ₀
1	0	<<	Q ₂	Q ₁	Q ₀	IR
1	1	>>	IL	Q ₃	Q ₂	Q ₁

- numarator

ID	E	D	Operatie
0	0	X	no change
0	1	0	++
0	1	1	--
1	X	X	load

FULL: 100 , EMPTY: 000



EXERCITII

Memoria ROM (Engl. "Read Only Memory") este o memorie accesibilă doar în citire (în timpul funcționării normale). Aceasta este un circuit combinațional a cărui funcție de transfer este descrisă sub forma unui tabel adresă/dată.

Memoria RAM (Engl. "Random Access Memory") este o memorie accesată în mod normal atât pentru scrierea cât și pentru citirea datelor. Memoria RAM are structura unui circuit secvențial, conținând elemente de memorare a datelor. În funcție de modul de stocare a informației, memoriile RAM se clasifică după cum urmează:

- **Memoria SRAM** (Engl. "Static RAM") stochează informația într-o celulă de memorie cu o structură bazată pe un latch (două inversoare conectate în buclă). Celula de memorie (figura 18.1-a) se numește **statică** deoarece cele două inversoare active sunt capabile să își păstreze starea atât timp cât circuitul este alimentat.

Simbolul bloc al unei memorii SRAM (figura 18.1-b) prezintă bus-uri de date de intrare și de ieșire, bus-ul de adrese și semnale de control care comandă validarea chip-ului (CS - Chip Select) și scrierea datelor (WR - Write).

- **Memoria DRAM** (Engl. "Dynamic RAM"), stochează informația sub formă de sarcină electrică într-o celulă de memorie cu o structură bazată pe un capacitor. Celula de memorie (figura 18.2-a) se numește **dinamică** deoarece sarcina capacitivă stocată se diminuează în timp datorită pierderilor capacitorului. Din acest motiv, memoriile DRAM necesită periodic o acțiune de "împrospătare" (Engl. "refresh") a sarcinii electrice stocate pentru refacerea periodică a informației înscrise în memorie. Aceasta este un dezavantaj al memoriilor DRAM față de cele SRAM. Însă, memoriile DRAM au avantajul unor celule de memorie de dimensiune mult mai mică care determină posibilitatea realizării unor memorii de dimensiuni mult mai mari într-o capsulă.

1. Se dorește realizarea unui modul de memorie $2K \times 8$ folosind chip-uri de memorie RAM 128×8 .

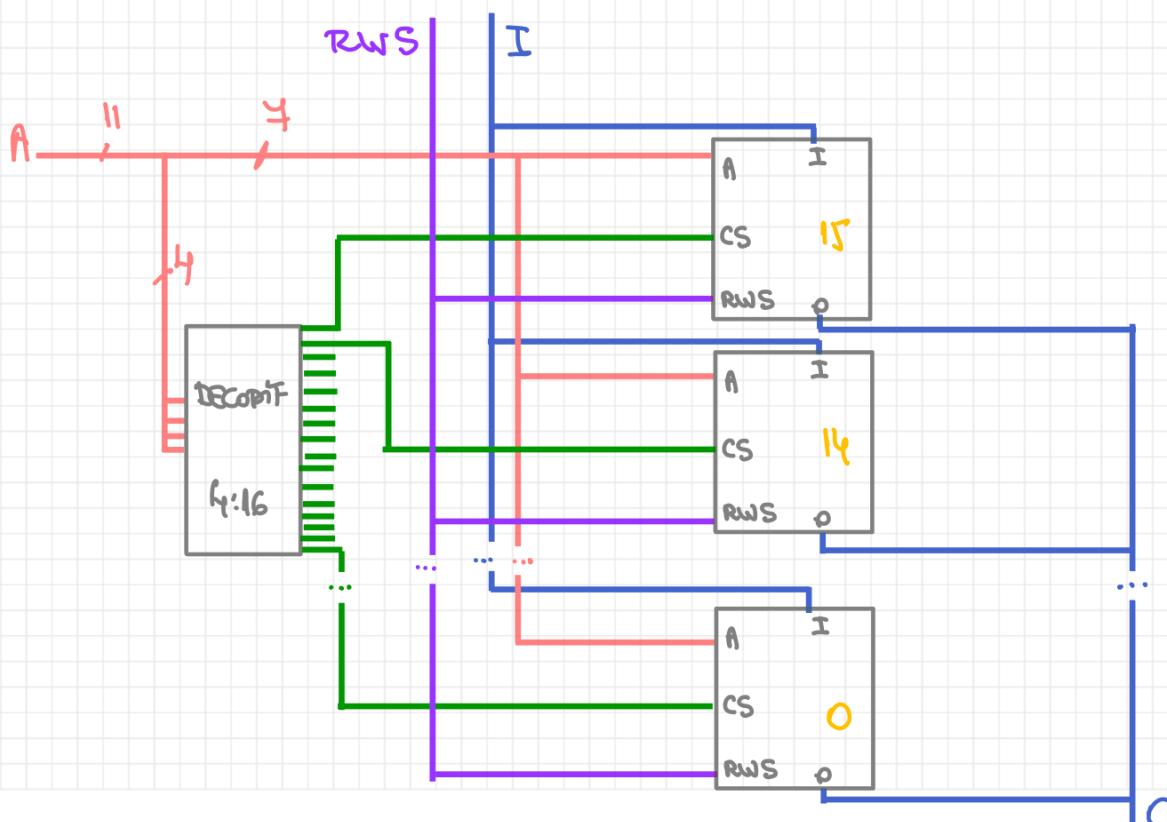
- Câte chip-uri sunt necesare?
- Câte linii de adresă sunt necesare pentru a adresa modulul de memorie?
- Câte linii de adresă trebuie conectate împreună la toate chip-urile?
- Câte linii de adresă trebuie decodate pentru intrările de selecție ale chip-urilor?
- Ce dimensiune are decodificatorul necesar?

a) $128 \times 16 = 2048 = 2^11 \Rightarrow 16$ chipuri

b) $2^k = 2^{11} \Rightarrow 11$

c) \neq la chip, \neq în afara

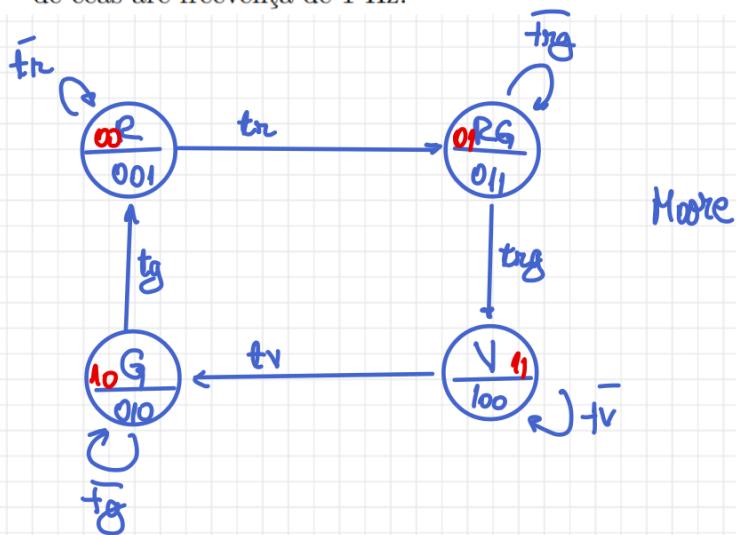
e) decodif $4 \rightarrow 16$: 4 variante la intrare $\Rightarrow 2^4$ la ieșire



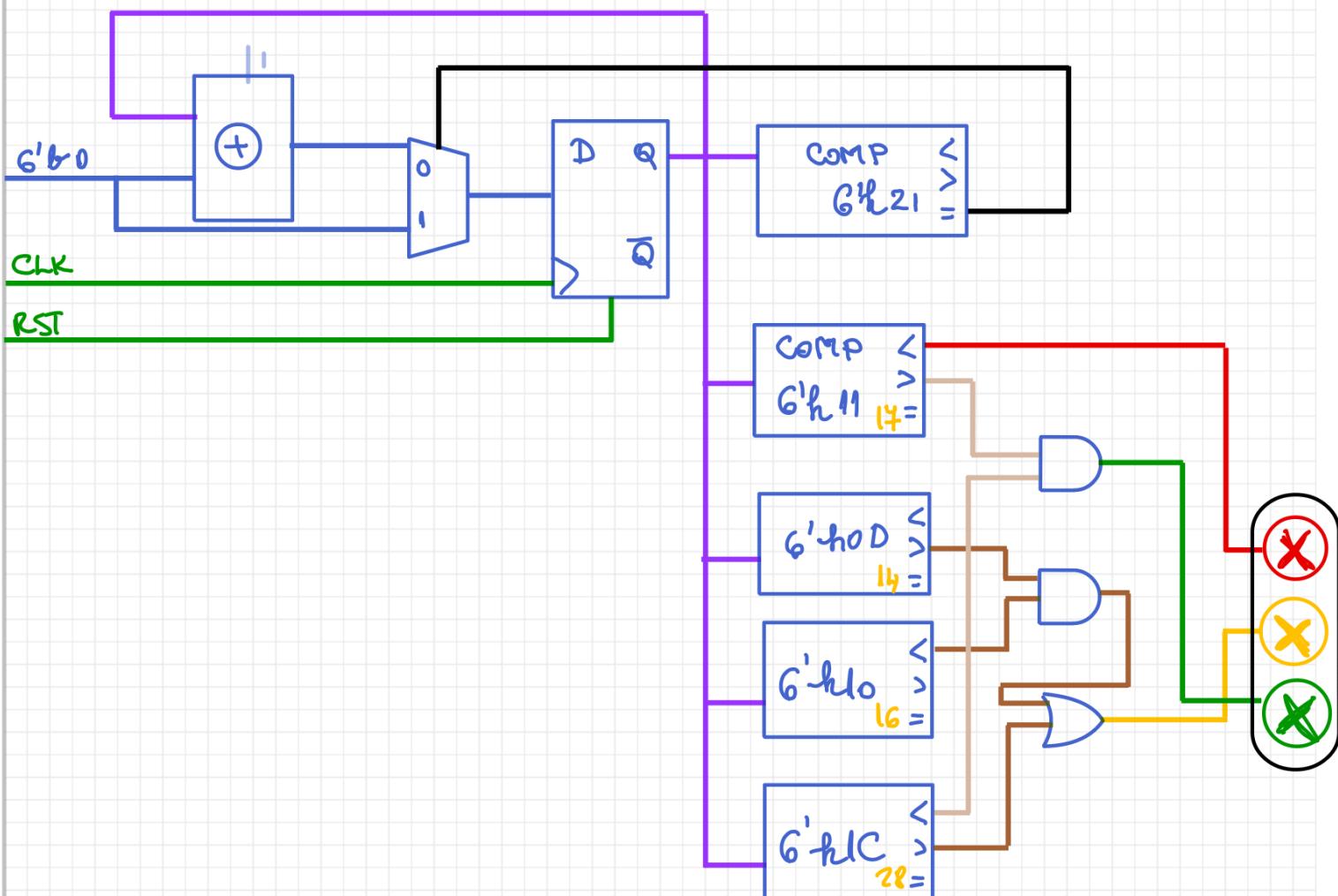
1. Proiectați un circuit de secvențiere care comandă un semafor. Circuitul are trei ieșiri active în 1 logic: verde, galben și roșu. Secvența de comandă este următoarea:

- a) roșu aprins 14 secunde
- b) roșu și galben aprinse 3 secunde
- c) verde aprins 12 secunde
- d) galben aprins 4 secunde

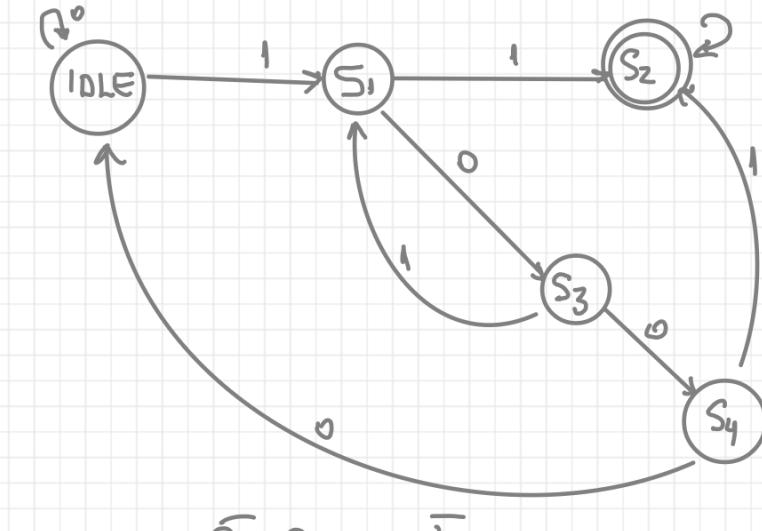
Proiectați circuitul utilizând un numărător cu presetare potrivit și logică adițională. Presupuneți că semnalul de ceas are frecvență de 1 Hz.



• Încercăm să proiectăm un numărător până la 33



4. Într-un sistem de comunicație serială, un mesaj poate începe fie cu secvența 11 fie cu secvența 1001. Să se realizeze un circuit secvențial sincron pentru detectarea secvențelor de început de mesaj. Intrarea se numește DATA iar singura ieșire se numește GĂSIT. Când RESET = 1, automatul intră în starea inițială și GĂSIT = 0. Când RESET = 0, automatul este în mod normal de operare. Circuitul semnalizează cu GĂSIT=1 după detectarea primei secvențe 11 sau 1001. Ieșirea GĂSIT revine la 0 doar după ce RESET devine 1 din nou. Proiectați un automat Moore care să implementeze circuitul de recunoaștere a secvențelor de început. Deducreți tabelul de tranziții al automatului și precizați semnificația fiecărei stări. Nu se vor utiliza mai mult de 5 stări.



STARE	i_m	$NL\ w\ t$	găsit
IDLE	000	0	000
		1	001
S1	001	0	011
		1	010
S2	010	*	010
			1
S3	011	0	100
		1	001
S4	100	0	000
		1	010

$$\Rightarrow D_2 = \bar{Q}_2 Q_1 Q_0 \bar{i_m}$$

$$D_1 = \bar{Q}_2 \bar{Q}_1 Q_0 + \bar{Q}_2 Q_1 \bar{Q}_0 + Q_2 \bar{Q}_1 \bar{Q}_0 \bar{i_m}$$

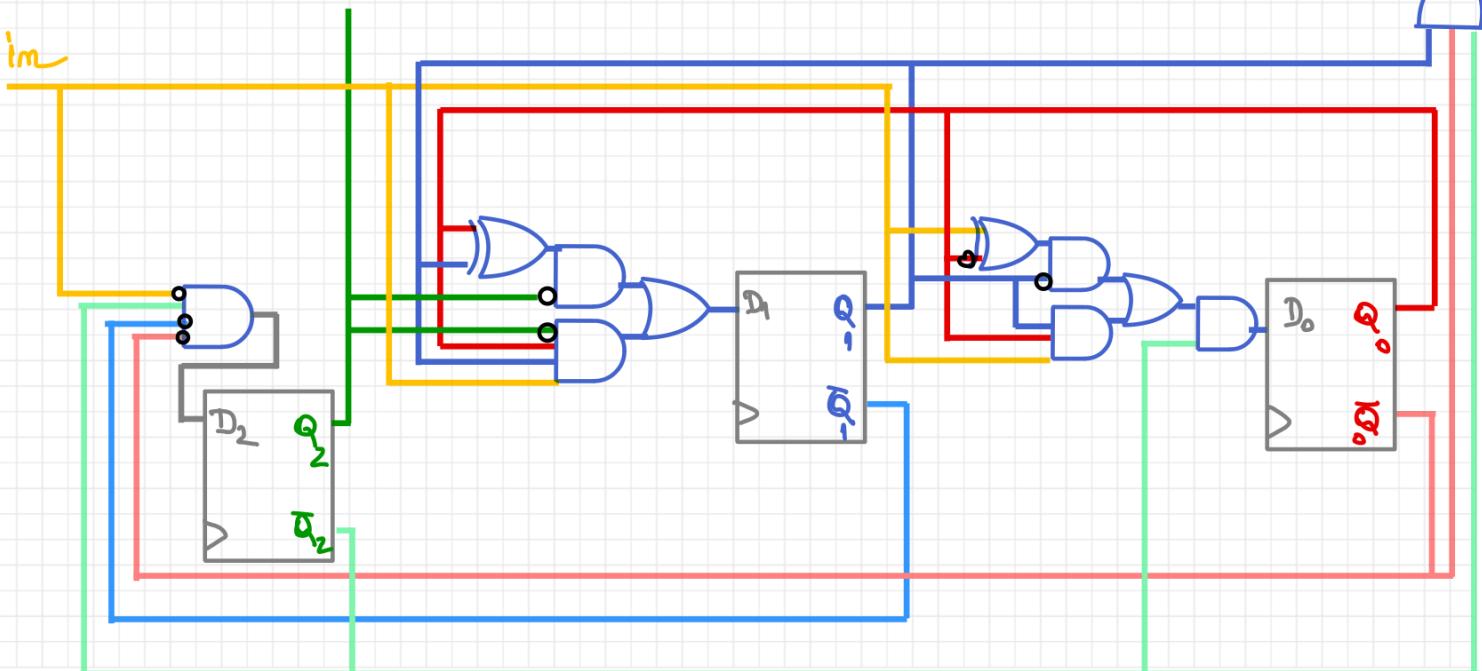
$$= \bar{Q}_2 (Q_1 \oplus Q_0) + Q_2 \bar{Q}_1 \bar{Q}_0 \bar{i_m}$$

$$D_0 = \bar{Q}_2 \bar{Q}_1 \bar{Q}_0 \bar{i_m} + \bar{Q}_2 \bar{Q}_1 Q_0 \bar{i_m} + \bar{Q}_2 Q_1 \bar{Q}_0 \bar{i_m}$$

$$= \bar{Q}_2 \bar{Q}_1 (\bar{Q}_0 \oplus i_m) + \bar{Q}_2 Q_1 \bar{Q}_0 \bar{i_m}$$

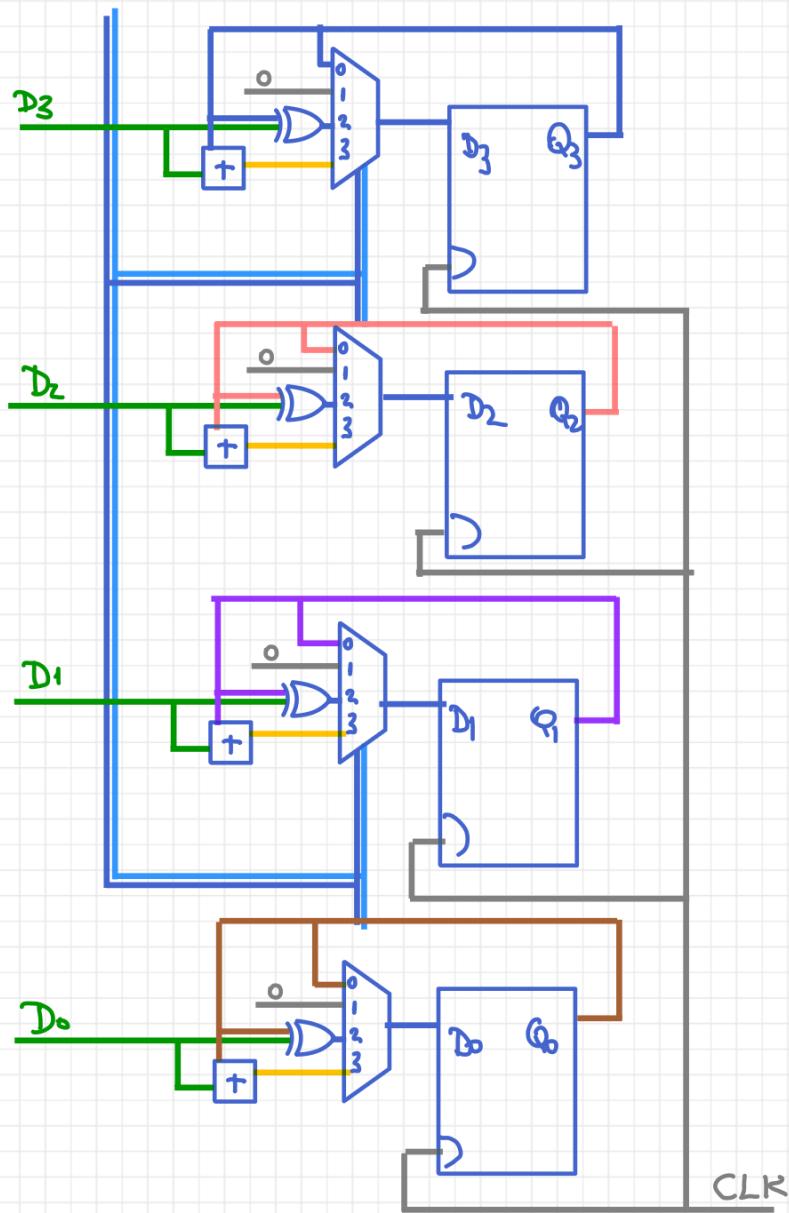
$$= \bar{Q}_2 \cdot [\bar{Q}_1 (\bar{Q}_0 \oplus i_m) + Q_1 \cdot Q_0 \cdot \bar{i_m}]$$

găsit



7. Proiectați un circuit secvențial care să efectueze următoarele operații:

S_1	S_0	$Q_{[3:0]}^+$	Acțiune
0	0	Q	Păstrează starea
0	1	0000	Initializează starea
1	0	$Q \oplus D_{in}$	XOR între starea curentă și intrarea de 4 biți
1	1	$Q + D_{in}$	Adună intrarea la starea curentă



CAP. VERILOG HDL

REG VS. WIRE

- wire-ile sunt utilizate pentru legături între module și pentru assignarea de rezultate parțiale în circuite combinatoriale, prim urmăre:
 - un element de tip WIRE și schimbă valoarea doar în blocuri de tip assign sau ca ieșire a unui modul
 - un element de tip REG și schimbă valoarea doar în blocuri de tip always
 - mai multe mici reg nu își pot schimba valoarea în mai mult de un bloc
 - assign e la combinatoriale, reg e la sevențiale

ASPECTE LEGATE DE MODELAREA CORECTĂ A UNUI BLOCK COMBINATIONAL

ASPECTE LEGATE DE MODELAREA CORECTĂ A UNUI BLOCK SEVENȚIAL

DIFERENȚA DINTRE CEI 2 OPERATORI DE ATRIBUIRE

- Blocking (=) : evaluarea și atribuirea au loc imediat

always @ (a or b or c)

begin

x = a | b;

1. Evaluate $a | b$, assign result to x

y = a ^ b ^ c;

2. Evaluate $a ^ b ^ c$, assign result to y

z = b & ~c;

3. Evaluate $b \& (\sim c)$, assign result to z

end

- NON-BLOCKING (<=) : evaluarea se face imediat, atribuirea la finalul blocului

always @ (a or b or c)

begin

x <= a | b;

1. Evaluate $a | b$ but defer assignment of x

y <= a ^ b ^ c;

2. Evaluate $a ^ b ^ c$ but defer assignment of y

z <= b & ~c;

3. Evaluate $b \& (\sim c)$ but defer assignment of z

end

4. Assign x, y, and z with their new values

Why two ways of assigning values?

Conceptual need for two kinds of **assignment** (in always blocks):

Blocking: Evaluation and assignment are immediate		
Non-Blocking: Assignment is postponed until all r.h.s. evaluations are done		
When to use: (only in always blocks!)	Sequential Circuits	Combinational Circuits