

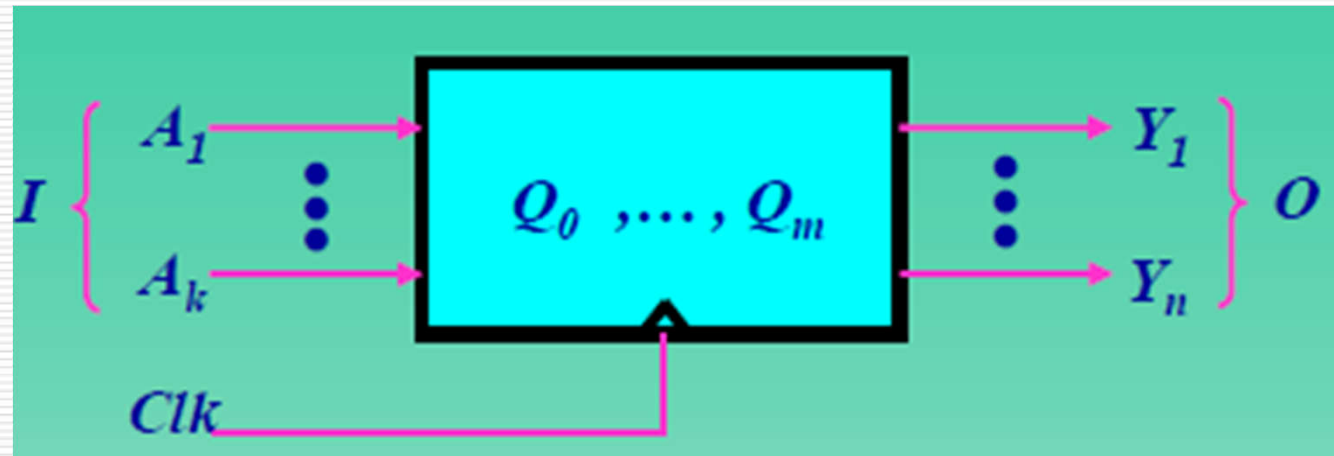
Logică digitală

-Curs 11-12-
FSM (Automate cu
Stări finite)

Outline

- ☐ definiție FSM
 - Moore
 - Mealy
 - ☐ Sinteza circuitelor secventiale
 - Codificarea stărilor
 - ☐ Număr minim de tranziții
 - ☐ Adiacență pe bază de priorități
 - ☐ One hot
 - ☐ Exemple
-

Automate cu stări finite



□ Cvadruplul $\langle S, I, O, f, h \rangle$

- S – mulțimea stărilor
- I – mulțimea intrărilor
- O – mulțimea ieșirilor
- f - funcțiile pt.starea urm.; h – funcțiile pt. ieșire

$$S = Q_1 \times Q_2 \times \dots \times Q_m,$$

$$I = A_1 \times A_2 \times \dots \times A_k,$$

$$O = Y_1 \times Y_2 \times \dots \times Y_n,$$

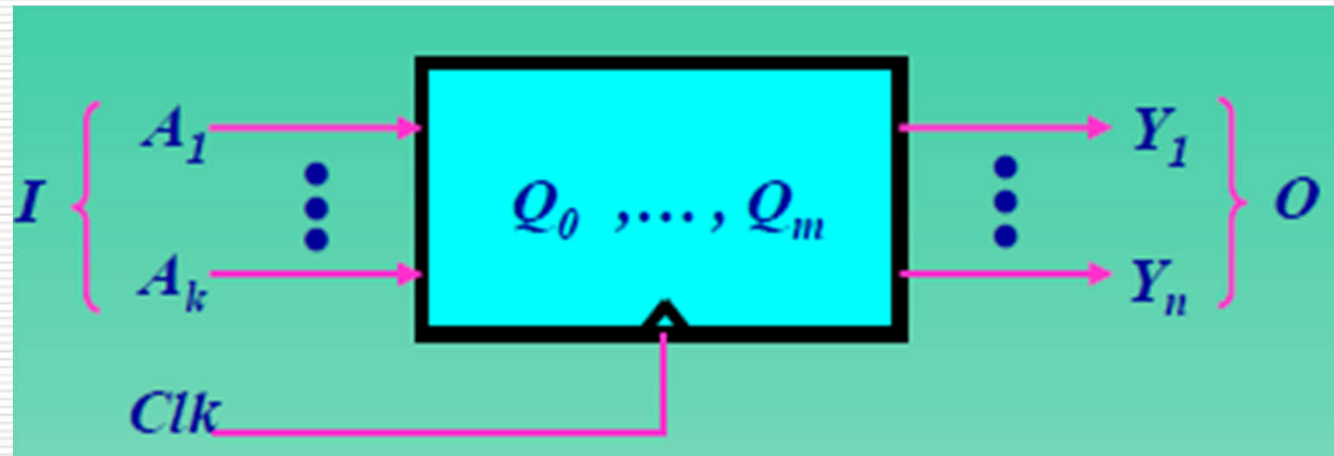
Circuite secvențiale reprezentare

□ **Circuitele secvențiale:**

- **MEALY** sunt caracterizate prin faptul că starea următoare și **ieșirea** la un moment dat depind de starea prezentă și de intrarea prezentă;
- **MOORE** sunt caracterizate prin faptul că **ieșirea** depinde **numai** de starea circuitului. Starea următoare depinde de intrarea prezentă;

- Modelele matematice ale circuitelor secvențiale se numesc în teoria comutațiilor **automate finite**.
-

Automate cu stări finite



□ Cvadruplul $\langle S, I, O, f, h \rangle$

- S – mulțimea stărilor
- I – mulțimea intrărilor
- O – mulțimea ieșirilor
- f - funcțiile pt.starea urm.; h – funcțiile pt. ieșire

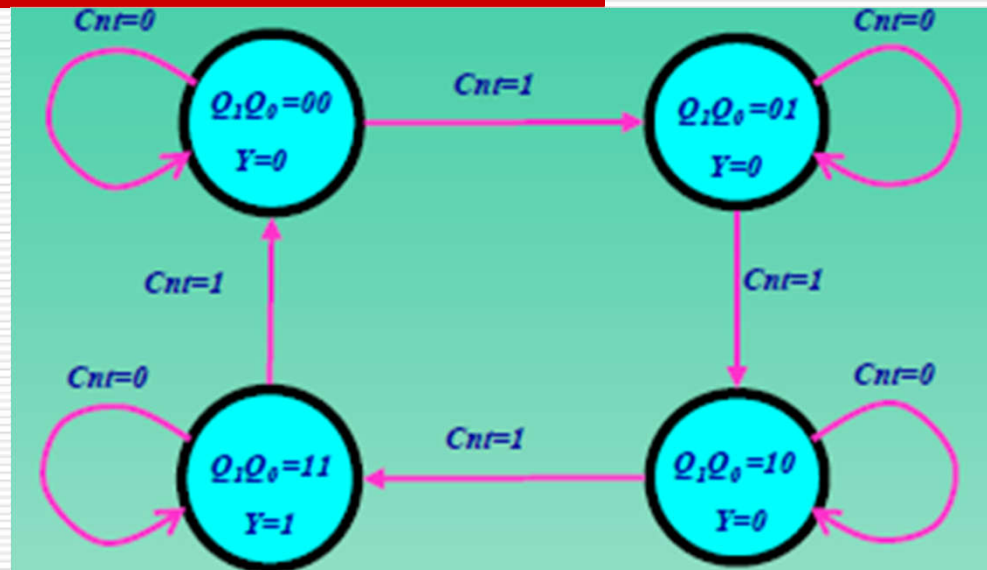
$$f : S \times I \longrightarrow S$$

$$h : S \times I \longrightarrow O \text{ (Mealy-type)}$$

$$S \longrightarrow O \text{ (Moore-type)}$$

Circuite secvențiale: diagrame e stare & tabelul tranzițiilor

□ Moore

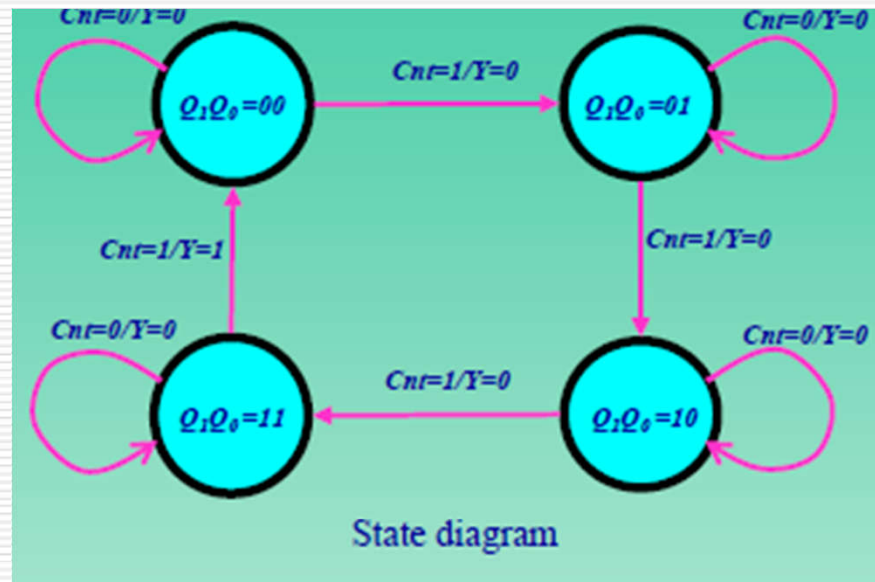


PRESENT STATE Q_1Q_0	NEXT STATE $Q_1(next) \quad Q_0(next)$		OUTPUTS Y
	$Cnt=0$	$Cnt=1$	
00	00	01	0
01	01	10	0
10	10	11	0
11	11	00	1

State and output table

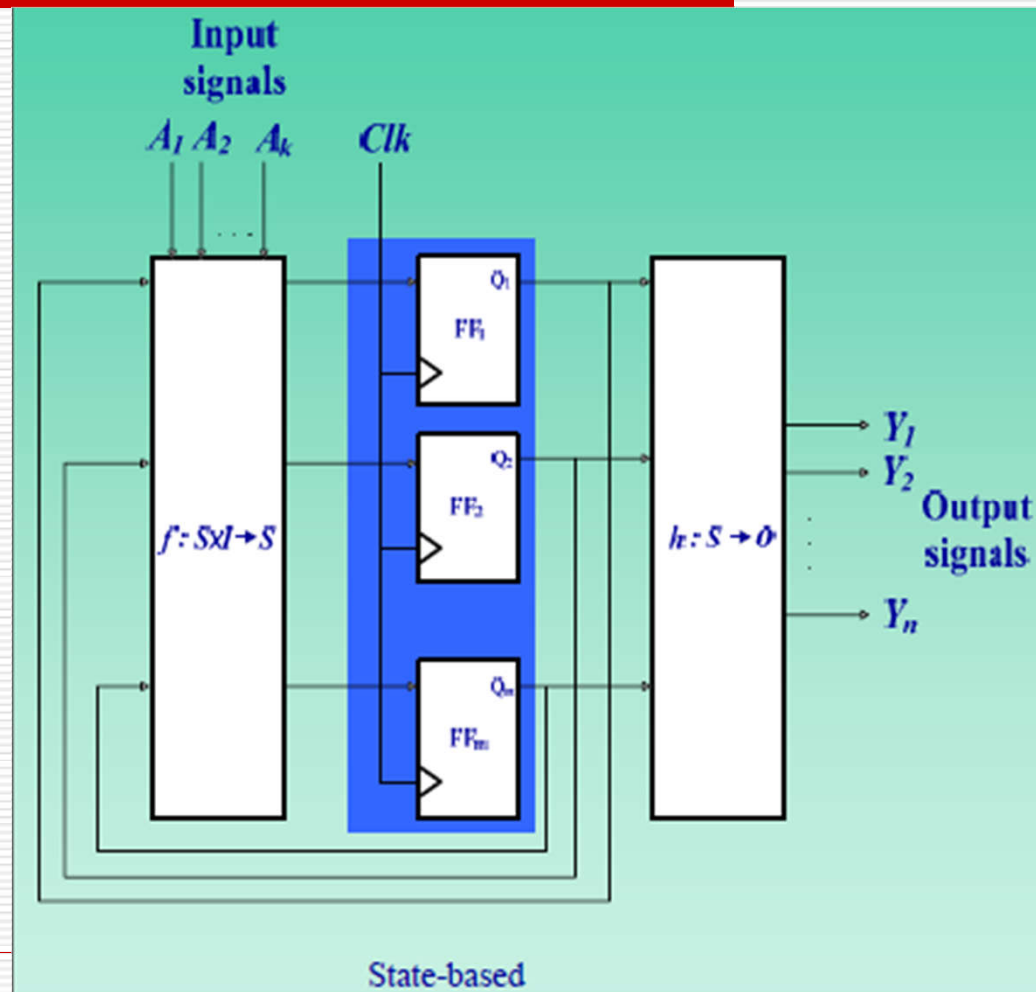
Circuite secvențiale: diagrame e stare & tabelul tranzițiilor

□ Mealy

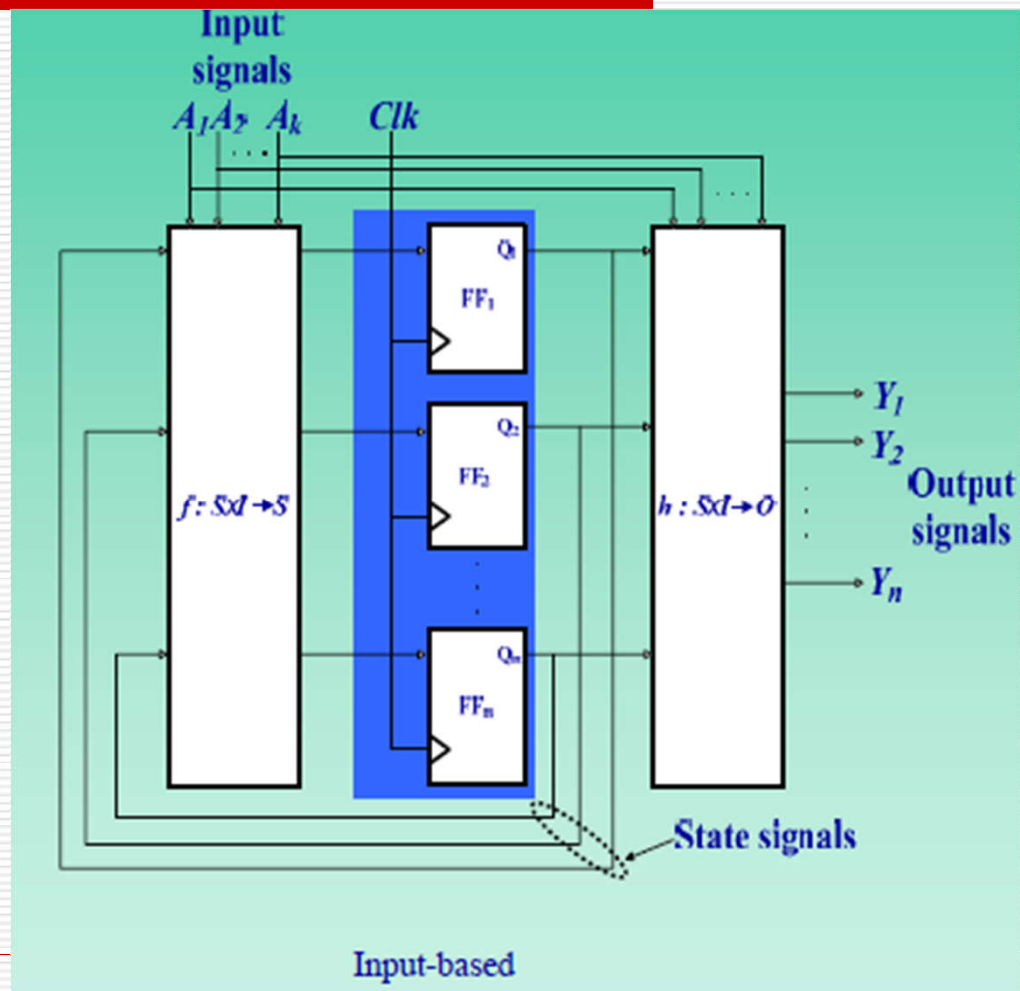


PRESENT STATE Q_1Q_0	NEXT STATE /OUTPUTS $Q_1(next) Q_0(next)/Y$	
	Cnt=0	Cnt=1
0 0	0 0 / 0	0 1 / 0
0 1	0 1 / 0	1 0 / 0
1 0	1 0 / 0	1 1 / 0
1 1	1 1 / 0	0 0 / 1

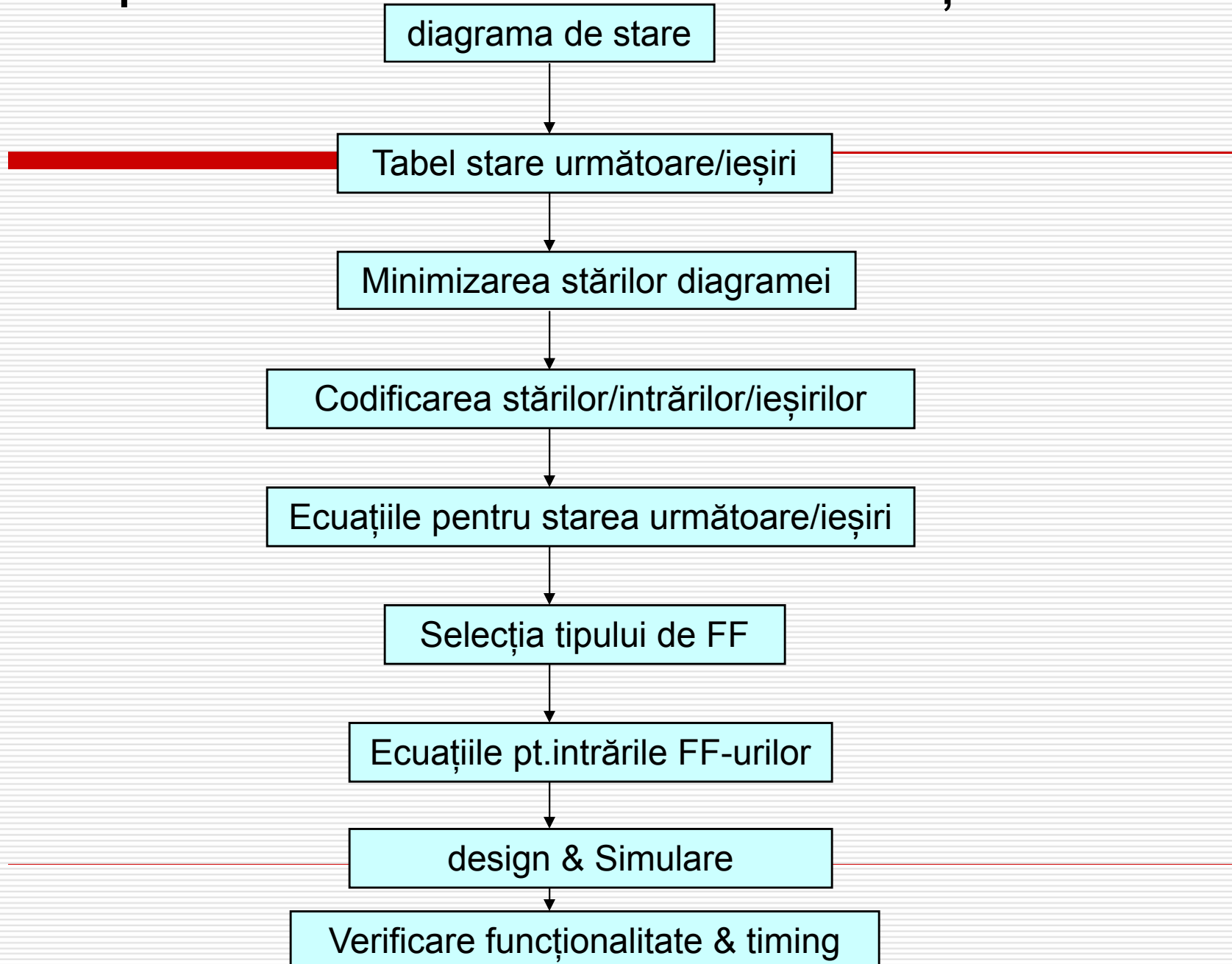
Implementare FSM Moore



Implementare FSM Mealy

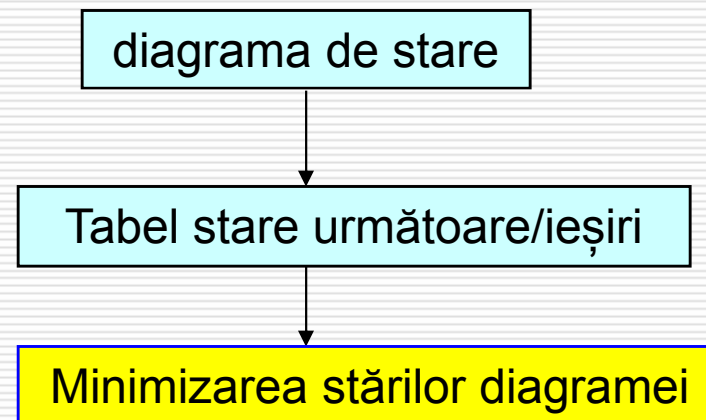


Etape de sinteză circuit secvențial



Etape realizare sistem secvential

- ❑ diagrama completă nu prezintă numărul minim de stări



Minimizarea stărilor

- Minimizarea stărilor își propune reducerea acestora → utilizarea unui număr mai mic de FF-uri
 - Se bazează pe conceptul de comportament echivalent; două stări ale unui FSM au comportament echivalent dacă:
 - Au aceeași secvență de valori de ieșire pentru aceeași secvență de vectori de intrare
 - două stări s_j și s_k sunt evident echivalente ($s_j \equiv s_k$) dacă și numai dacă:
 - (1) au comportament echivalent: **$h(s_j, i) = h(s_k, i)$**
 - (2) au aceleași stări următoare pt. toate secvențele de intrare **$f(s_j, i) = f(s_k, i)$**
-

Minimizarea stărilor

- două stări s_j și s_k sunt echivalente ($s_j \equiv s_k$) dacă și numai dacă:
 - (1) au comportament echivalent: **$h(s_j, i) = h(s_k, i)$**
 - (2) au stări următoare diferite dar acestea sunt echivalente
 - două automate A1 și A2 sunt echivalente dacă pentru fiecare stare s_j din A2 există o stare echivalentă s_k în A1 și invers pentru fiecare stare s_j din A1 există o stare echivalentă s_k în A2
 - Echivalența stărilor unui automat complet definit împarte mulțimea stărilor acestuia în clase de echivalență disjuncte.
 - Relația de echivalență a stărilor automatului complet definit are **proprietatea de tranzitivitate**: dacă și atunci.
-

diagrama de stare



Tabel stare următoare/ieșiri



Minimizarea stărilor diagramei



Codificarea stărilor/intrărilor/ieșirilor

Codificarea stărilor

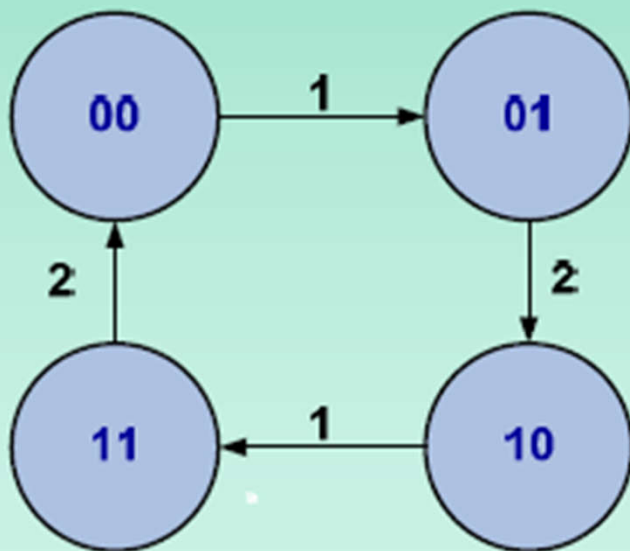
ENCODING NUMBER	s_0	s_1	s_2	s_3
1	00	01	10	11
2	00	01	11	10
3	00	10	01	11
4	00	10	11	01
5	00	11	01	10
6	00	11	10	01
7	01	00	10	11
8	01	00	11	10
9	01	10	00	11
10	01	10	11	00
11	01	11	00	10
12	01	11	10	00
13	10	00	01	11
14	10	00	11	01
15	10	01	00	11
16	10	01	11	00
17	10	11	00	01
18	10	11	01	00
19	11	00	01	10
20	11	00	10	01
21	11	01	00	10
22	11	01	10	00
23	11	10	00	01
24	11	10	01	00

- ❑ Costul/întârzierea unei implementări FSM depind de codificarea stărilor;
- ❑ 4! Posibilități de codificare pentru 4 stări.
- ❑ 3 euristici:
 - Număr minim de tranziții
 - Adiacență pe bază de priorități
 - one-hot

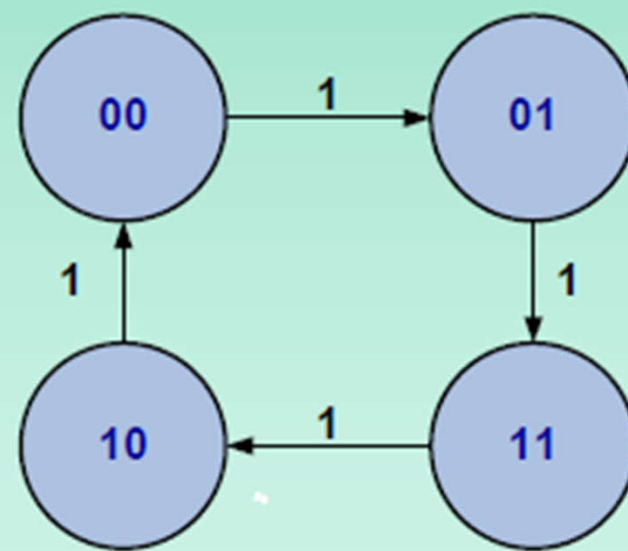
Codificarea stărilor

- **Număr minim de tranziții (minimum bit change):**
 - Stările sunt astfel codificate încât să fie minimizeze tranzițiile biților registrului de stare;
 - Fiecărui arc i se alocă un cost egal cu numărul de biți ai registrului de stare care diferă la tranziția dintre stări;
 - Se minimizează suma costurilor la parcurgerea grafului
 - **Adiacență pe bază de priorități**
 - **one-hot**
-

Număr minim de tranziții



Straightforward encoding



Minimum-bit-change encoding

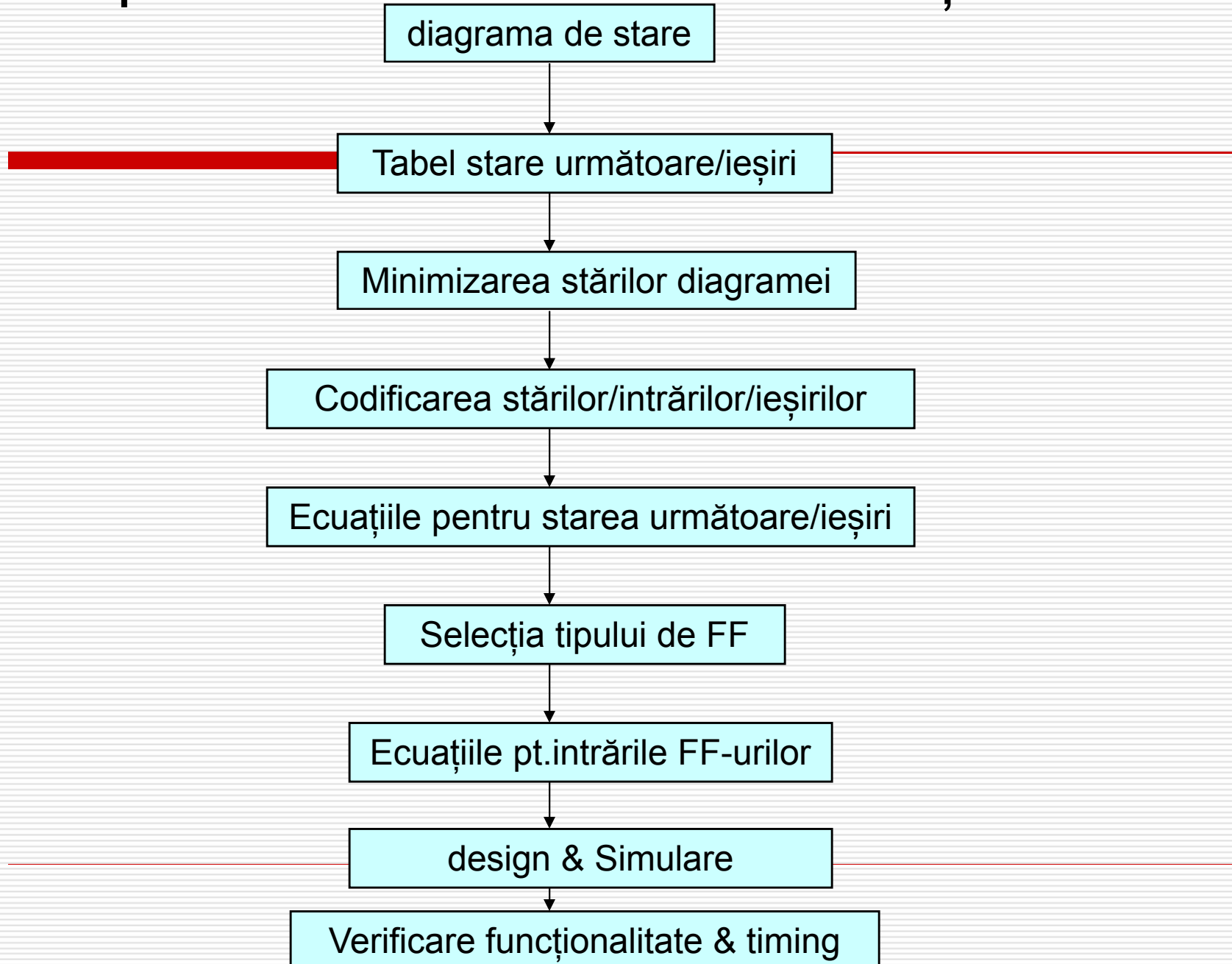
Codificarea stărilor

- **Număr minim de tranziții**
 - **Adiacență pe bază de priorități
(Prioritized adjacency strategy):**
 - **Codificări adiacente pentru stările:**
 - **destinație comună**
 - **Sursă comună**
 - **Ieșire comună**
 - **Next state va apărea în căsuțe
adiacente în K-map;**
 - **one-hot**
-

Codificarea stărilor

- **Număr minim de tranziții**
 - **Adiacență pe bază de priorități**
 - **one-hot**
 - Fiecărei stări i se alocă un bit din registrul de stare → registrul de stare are atâția 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)
-

Etape de sinteză circuit secvențial



Design-ul FSM-urilor folosind diagrame de stare

- ❑ Se preteaza pentru FSM-uri de dimensiune mica, respectiv medie;
 - ❑ Constructiv:
 - Un tabel de stari este o lista exhaustiva de stari urmatoare corespunzatoare unei combinatii: (stare curenta, intrare);
 - O diagrama de stare contine un set de arce etichetate cu conditia aferenta tranzitiei catre starile urmatoare.
-

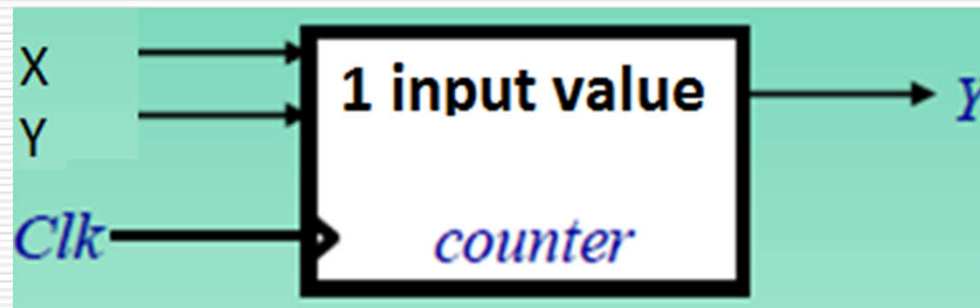
Design-ul FSM-urilor folosind diagrame de stare

- ❑ Se preteaza pentru FSM-uri de dimensiune mica, respectiv medie;
- ❑ Constructiv:
 - Un tabel de stari este o lista exhaustiva de stari urmatoare corespunzatoare unei combinatii: (stare curenta, intrare);

Chiar daca sunt mai multe intrari, numai o singura expresie corespunzatoare tranzitiei este marcata pe arc!

Exemplu

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



Observatii:

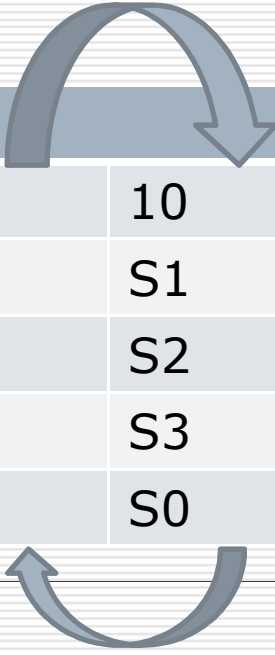
- ❑ Nu trebuie retinuta secventa de valori de la intrare!
- ❑ deoarece iesirea indica numarul de valori de intrare egale cu 1, modulo 4, 4 stari sunt suficiente ($S_0 \div S_3$).
- ❑ Pasul 1: Realizarea tabelului starilor si iesirilor

Semnificatie	S	XY				Z
		00	01	10	11	
Got 0 "1" (mod 4)	S ₀	S ₀	S ₁	S ₁	S ₂	1
Got 1 "1" (mod 4)	S ₁					0
Got 2 "1" (mod 4)	S ₂					0
Got 3 "1" (mod 4)	S ₃					0

Numărător valori intrare egale cu 1

- ❑ Codificarea starilor: Karnaugh order map (00, 01, 11, 10) deoarece in principiu ar trebui sa insemne complexitate mai mica pentru logica de intrare si simplifica transcrierea informatiei in vederea minimizarii.

Codificare Q1Q0	S	XY				Z
		00	01	11	10	
0 0	S0	S0	S1	S2	S1	1
0 1	S1	S1	S2	S3	S2	0
1 1	S2	S2	S3	S0	S3	0
1 0	S3	S3	S0	S1	S0	0



Numărător valori intrare egale cu 1

Codificare Q1Q0	S	XY				Z
		00	01	11	10	
0 0	S0	S0	S1	S2	S1	1
0 1	S1	S1	S2	S3	S2	0
1 1	S2	S2	S3	S0	S3	0
1 0	S3	S3	S0	S1	S0	0
		Q1(next)Q0(next) sau D1, D0				

Inlocuirea simbolurilor starilor cu
codificarea aferenta

Numărător valori intrare egale cu 1

Codificare Q1Q0	S	XY				Z
		00	01	11	10	
0 0	S0	00	01	11	01	1
0 1	S1	01	11	10	11	0
1 1	S2	11	10	00	10	0
1 0	S3	10	00	01	00	0
Q1(next)Q0(next) sau d1, d0						



determinarea expresiei aferente logicii
de intrare (d1 sau d0) prin realizare
diagramelor Karnaugh

d_1 $Q_1Q_0 \backslash XY$	00	01	11	10
00	0	0	1	0
01	0	1	1	1
11	1	1	0	1
10	1	0	0	0

Codificare Q1Q0	S	XY				Z
		00	01	11	10	
0 0	S0	<u>0</u> 0	<u>0</u> 1	<u>1</u> 1	<u>0</u> 1	1
0 1	S1	<u>0</u> 1	<u>1</u> 1	<u>1</u> 0	<u>1</u> 1	0
1 1	S2	<u>1</u> 1	<u>1</u> 0	<u>0</u> 0	<u>1</u> 0	0
1 0	S3	<u>1</u> 0	<u>0</u> 0	<u>0</u> 1	<u>0</u> 0	0
		Q1(next)Q0(next) sau d1, d0				

Expresie logica stare urmatoare

- ☐ Analog se construiește diagrama Karnaugh pentru D0



Numărător valori intrare egale cu 1

d ₀ Q ₁ Q ₀ \XY	00	01	11	10
00	0	1	1	1
01	1	1	0	1
10	1	0	0	0
11	0	0	1	0

Codificare Q ₁ Q ₀	S	XY				Z
		00	01	11	10	
0 0	S0	<u>0</u> 0	<u>0</u> 1	<u>1</u> 0	<u>0</u> 1	1
0 1	S1	<u>0</u> 1	<u>1</u> 1	<u>1</u> 0	<u>1</u> 1	0
1 1	S2	<u>1</u> 1	<u>1</u> 0	<u>0</u> 0	<u>1</u> 0	0
1 0	S3	<u>1</u> 0	<u>0</u> 0	<u>0</u> 1	<u>0</u> 0	0
Q ₁ (next)Q ₀ (next) sau d ₁ , d ₀						

Expresie logica iesire

- Analog se construiește diagrama Karnaugh pentru Z și se minimizează



Homework

- ☐ Realizati scheme logica pentru sistemul secvential.
 - ☐ Desenati diagrama.
-

Design-ul FSM-urilor folosind diagrame de stare

- ❑ Se preteaza pentru FSM-uri de dimensiune mica, respectiv medie;
- ❑ Constructiv:

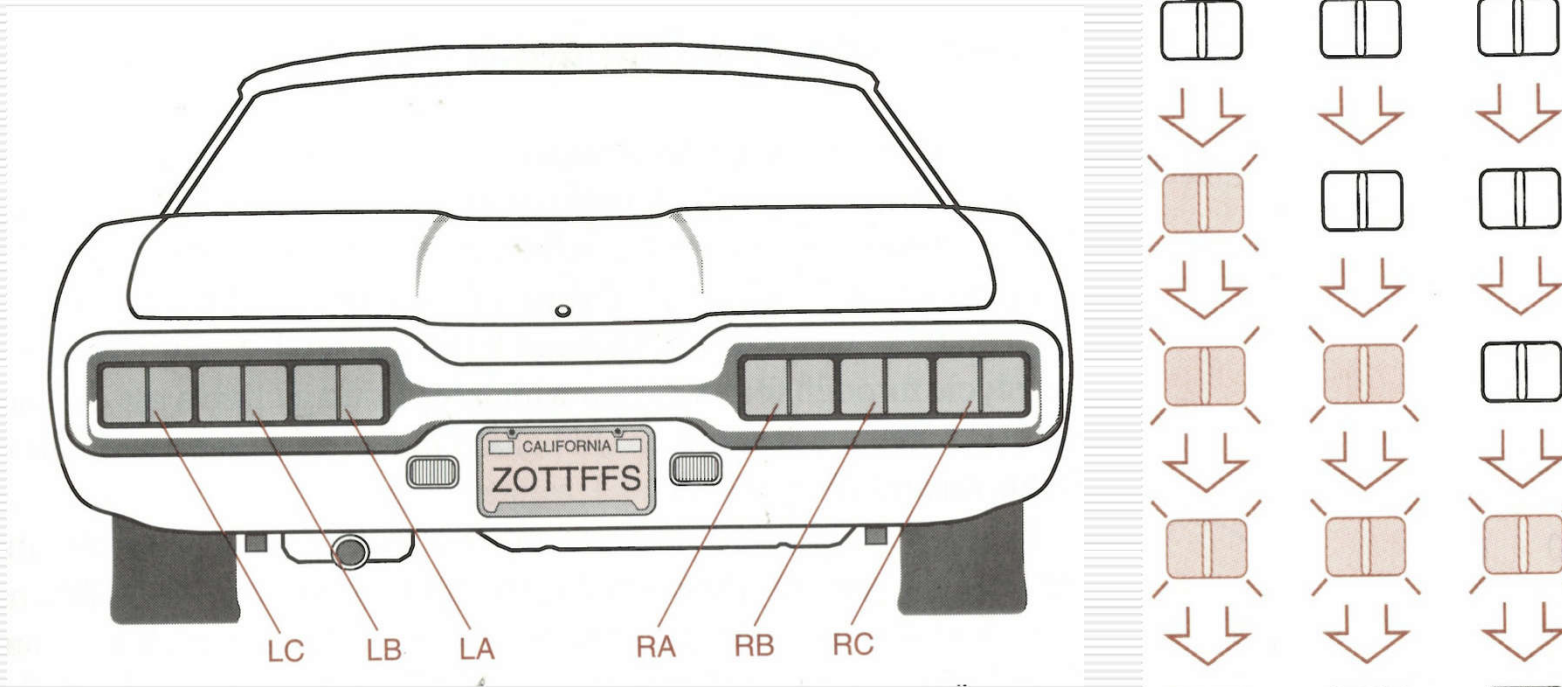
Probleme care pot aparea:

1. Diagrama de stare ambigua! Pot aparea situatii in care sa existe perechi (stare, intrari) pentru care sa nu fie specificata starea urmatoare.
2. Nu este garantat faptul ca conditiile pentru tranzitie asociate arcelor care pleaca dintr-o stare acopera toate combinatiile de intrare exact o data.

Exemplu: Ford Thunderbird 1965

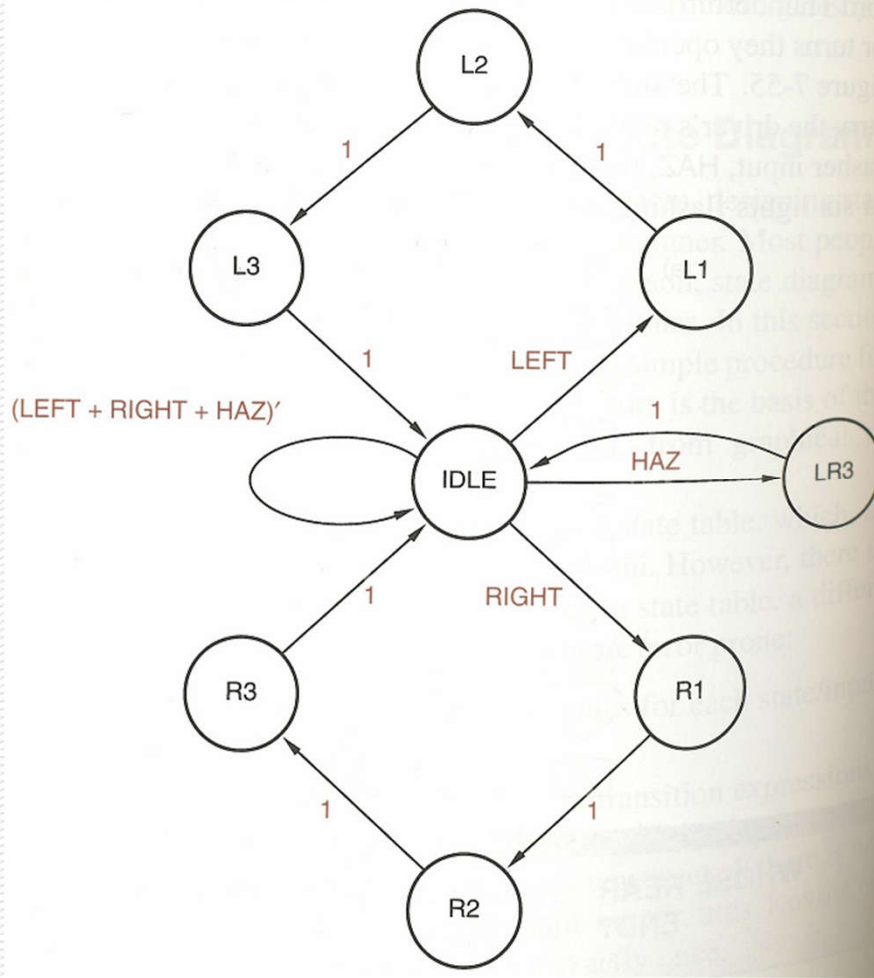
Ex.: semnalizare Ford Thunderbird 1965

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



Ex. Preluat din Digital Design Principles and Practices, Cap 7, J.F. Wakerly, 4th Edition

Ex.: semnalizare Ford diagrama Moore



Output Table

State	LC	LB	LA	RA	RB	RC
IDLE	0	0	0	0	0	0
L1	0	0	1	0	0	0
L2	0	1	1	0	0	0
L3	1	1	1	0	0	0
R1	0	0	0	1	0	0
R2	0	0	0	1	1	0
R3	0	0	0	1	1	1
LR3	1	1	1	1	1	1

Ecuatiile iesirilor

$$LA = L1 + L2 + L3 + LR3$$

$$LB = L2 + L3 + LR3$$

$$LC = L3 + LR3$$

$$RA = R1 + R2 + R3 + LR3$$

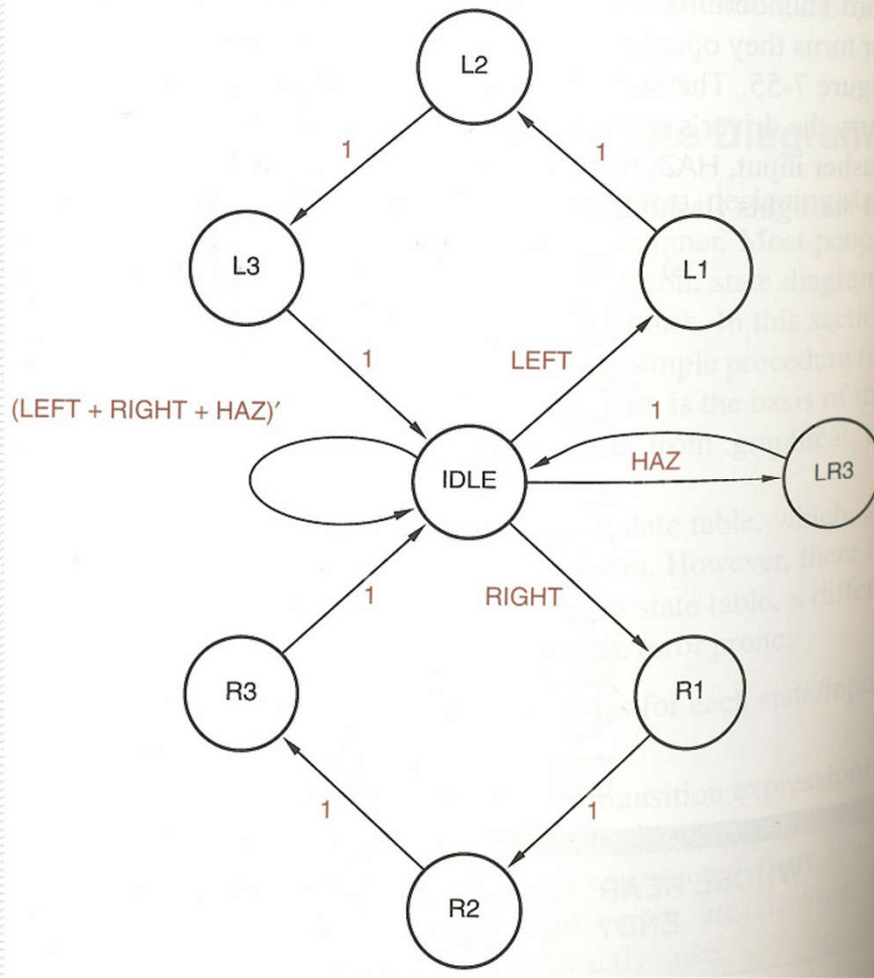
$$RB = R2 + R3 + LR3$$

$$RC = R3 + LR3$$

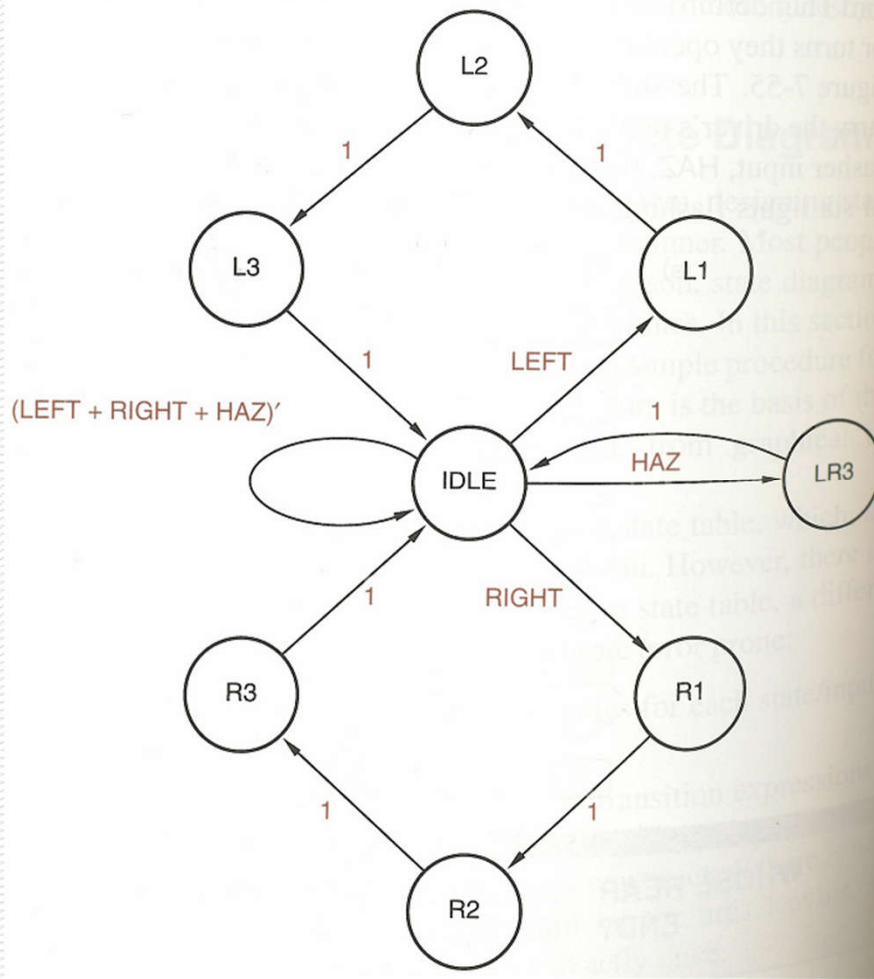
Output Table

State	LC	LB	LA	RA	RB	RC
IDLE	0	0	0	0	0	0
L1	0	0	1	0	0	0
L2	0	1	1	0	0	0
L3	1	1	1	0	0	0
R1	0	0	0	1	0	0
R2	0	0	0	1	1	0
R3	0	0	0	1	1	1
LR3	1	1	1	1	1	1

Probleme?

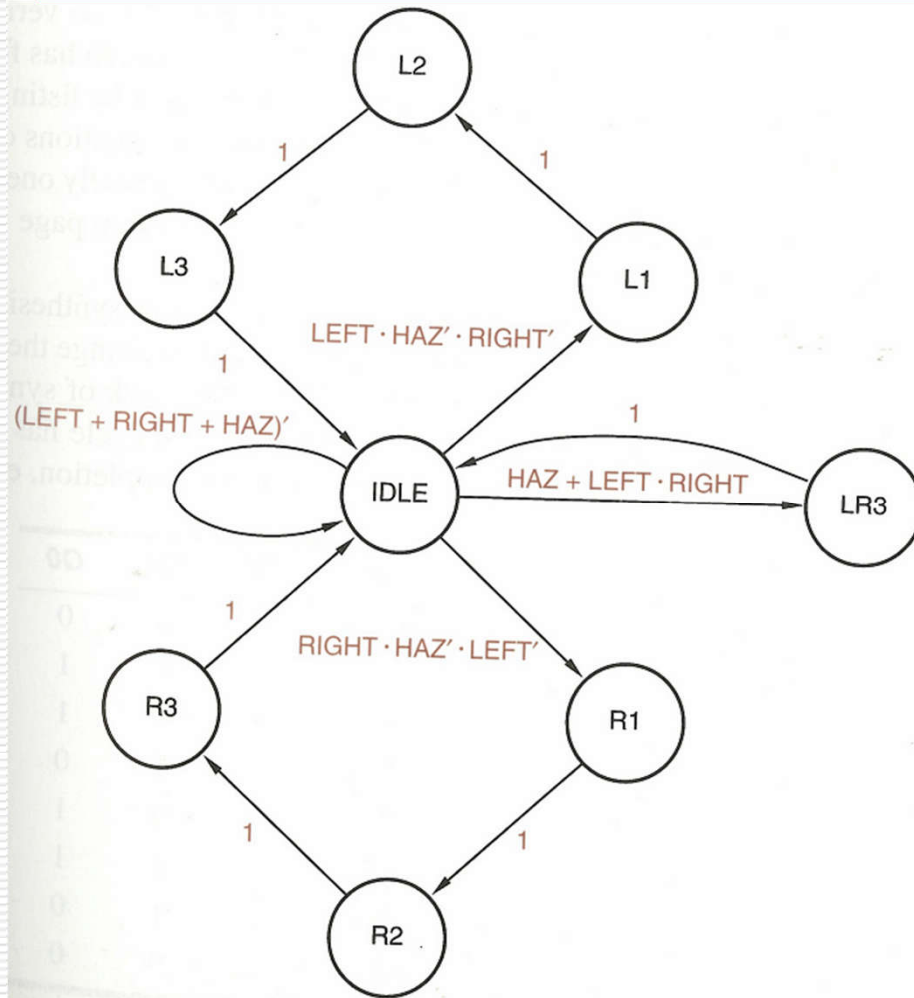


Probleme?



- Ce se intampla daca LEFT si HAZ sunt asertuite simultan?

Probleme?

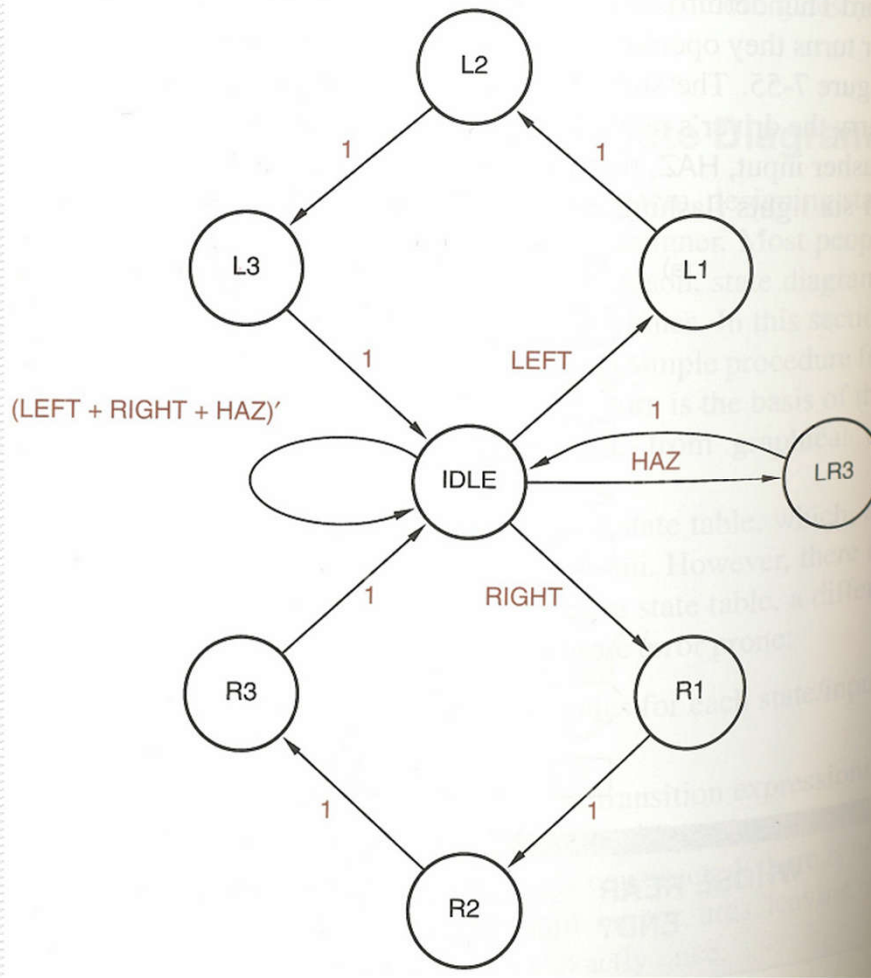


- ❑ Solutie: Ii dam la intrarea HAZ prioritate, iar cazul LEFT si RIGHT asertuite simulatan il tratam ca si hazard.
- ❑ Noua diagrama nu e ambigua: conditiile de pe arce se exclud reciproc si surprind toate combinatiile de intrare!

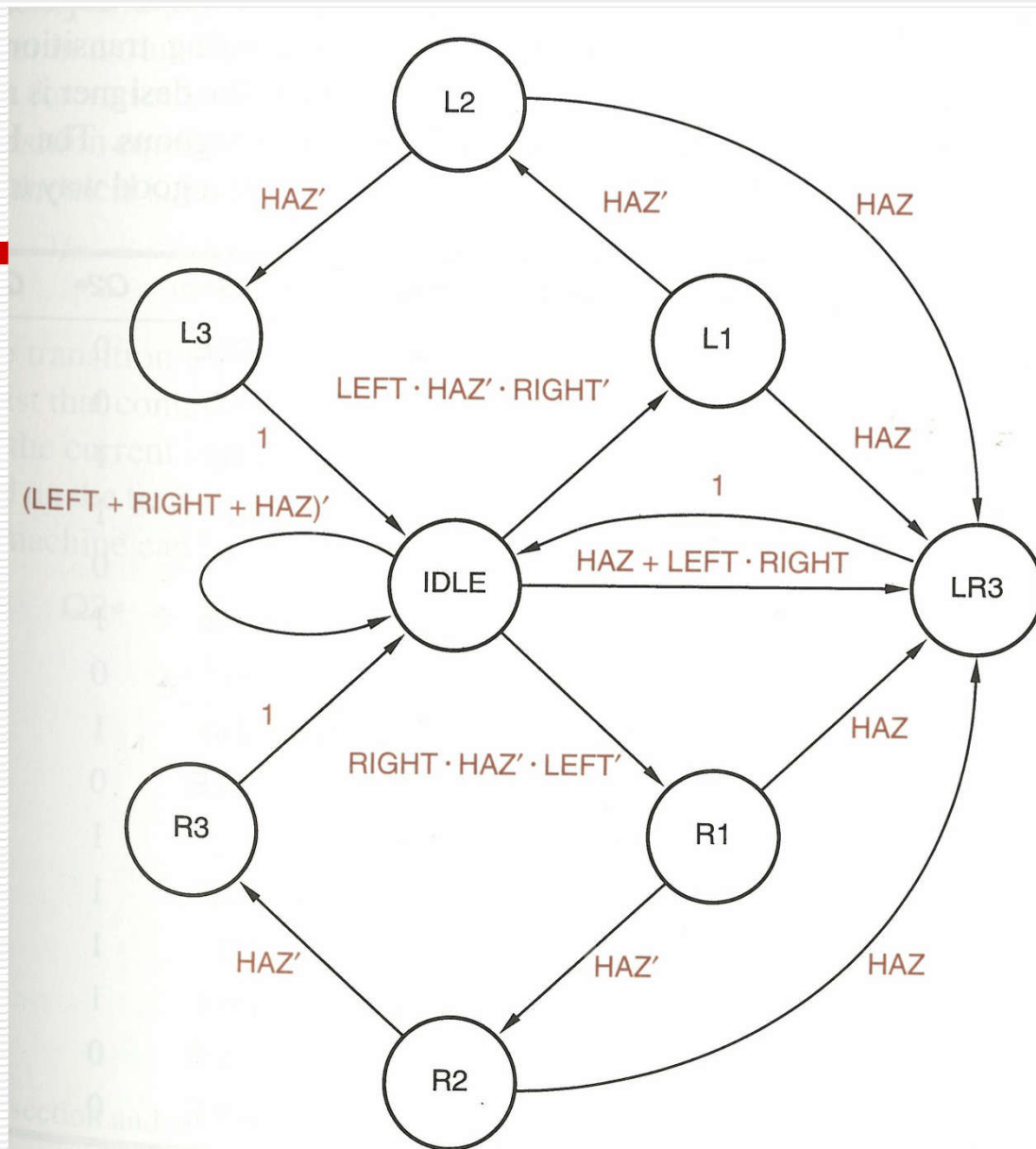
Diagrama Ne-amigua

- ❑ Mutual exclusion: produs logic intre oricare 2 expresii a tranzitiei pentru oricare 2 arce care pleaca dintr-o stare este 0.
 - ❑ All inclusion: suma logica a expresiilor tranzitiilor tuturor arcelor care pleaca dintr-o stare este 1.
-

Probleme?

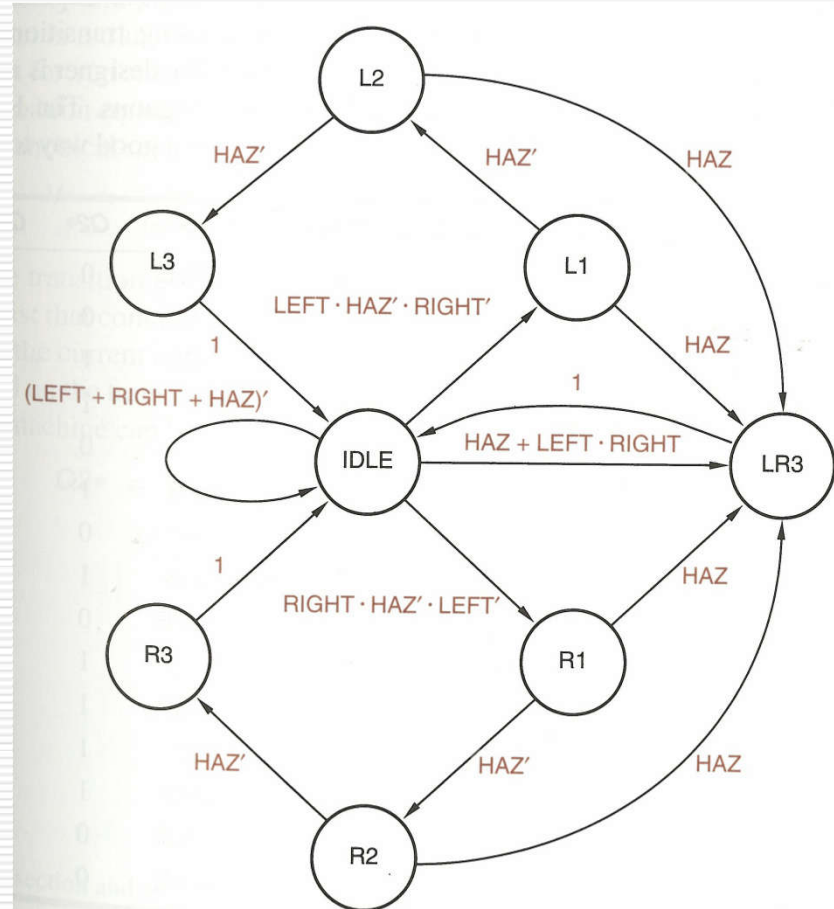


- Ar fi de dorit pentru utilizator ca semnalul de hazard sa fie prioritar.
- Adica daca esti intr-o secventa de semnalizare la stanga in starea L1 si HAZ este activ sa treci direct in starea de hazard LR3

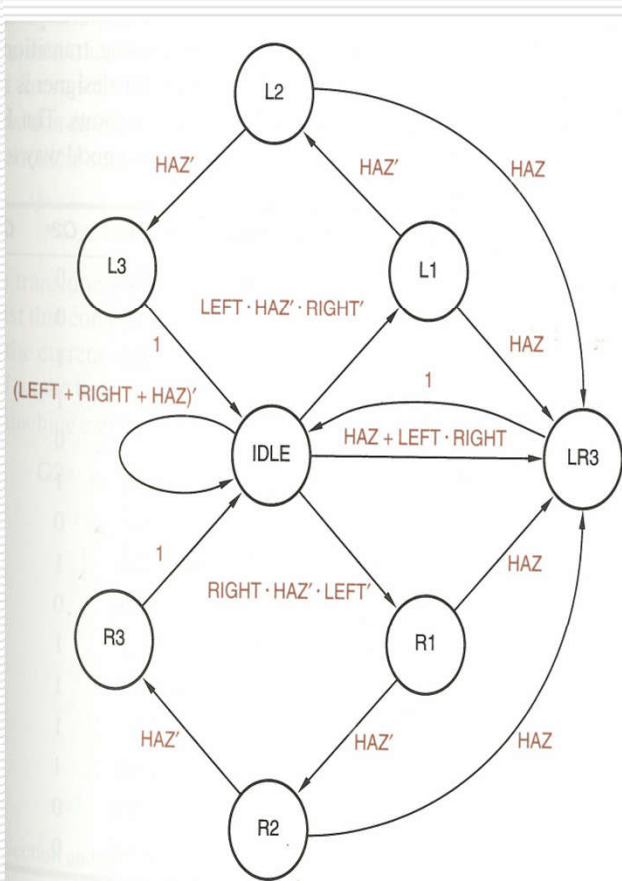


Codificarea starilor

<i>State</i>	<i>Q2</i>	<i>Q1</i>	<i>Q0</i>
IDLE	0	0	0
L1	0	0	1
L2	0	1	1
L3	0	1	0
R1	1	0	1
R2	1	1	1
R3	1	1	0
LR3	1	0	0



Realizarea unei liste de tranzitii



S	Q2	Q1	Q0	Transition Expression	S*	Q2*	Q1*	Q0*
IDLE	0	0	0	$(\text{LEFT} + \text{RIGHT} + \text{HAZ})'$	IDLE	0	0	0
IDLE	0	0	0	$\text{LEFT} \cdot \text{HAZ}' \cdot \text{RIGHT}'$	L1	0	0	1
IDLE	0	0	0	$\text{HAZ} + \text{LEFT} \cdot \text{RIGHT}$	LR3	1	0	0
IDLE	0	0	0	$\text{RIGHT} \cdot \text{HAZ}' \cdot \text{LEFT}'$	R1	1	0	1
L1	0	0	1	HAZ'	L2	0	1	1
L1	0	0	1	HAZ	LR3	1	0	0
L2	0	1	1	HAZ'	L3	0	1	0
L2	0	1	1	HAZ	LR3	1	0	0
L3	0	1	0	1	IDLE	0	0	0
R1	1	0	1	HAZ'	R2	1	1	1
R1	1	0	1	HAZ	LR3	1	0	0
R2	1	1	1	HAZ'	R3	1	1	0
R2	1	1	1	HAZ	LR3	1	0	0
R3	1	1	0	1	IDLE	0	0	0
LR3	1	0	0	1	IDLE	0	0	0

Sinteza

- ❑ Model 2 segment;
 - ❑ Responsabilitatea design-erului este scrierea unei descrieri de automat de stari neambigua;
 - ❑ Realizata de CAD-uri;
 - Eliminarea tranzitiilor duplicate;
-

RTM (Register Transfer Methodology)

- ❑ descriere foarte similară cu cea utilizată pentru algoritmi, și anume descrie fluxul de date prin prisma operațiilor care au loc la nivel de registre.
- ❑ se poate realiza o analogie între operații la nivel de registru și variabilele dintr-un algoritm.
- ❑ Nivelul RT (register transfer) de abstractizare este situat între nivelul poartă logică și nivelul procesor.

RTM

- ❑ situat între nivelul poartă logică și nivelul procesor.
 - ❑ este descris în termeni de operație RT de bază.
 - ❑ o unitate digitală este descrisă ca o succesiune de operații RT de bază, succesiune dictată de o logică (modul) de control.
-

Operația RT elementară

- este în esență o funcție simplă care calculează valoarea a unui sau mai multor registre destinație pe baza conținutului registrelor sursă.

$$R_{DEST} \leftarrow f(R_{SRC1}, R_{SRC2}, \dots, R_{SRC_M})$$

Operația RT elementară

- ❑ Pe frontul crescător al tactului, informația care este încărcată în registre disponibilă la intrare. După o întârziere ea este disponibilă la ieșiri.
- ❑ O logică combinațională calculează funcția f funcție de intrări.
- ❑ Rezultatul evaluării este trimis printr-o logică combinațională la intrările registrului destinație.
- ❑ La proximal front crescător al semnalului de tact, rezultatul de la intrarea registrului destinație este încărcat în registru.

RT exemplu: suma

$$R_1 \leftarrow R_1 + R_2$$



Considerăm următoarele sufixe pentru registre:

Sufix `_reg` – face referire la ieșirea curentă (din timpul impulsului de tact curent) a registrelor;

Sufix `_next` – face referire la datele care sunt la intrarea registrelor, calculații complete sau rezultate intermediare (din timpul impulsului de tact curent);

Atragem atenția asupra comportamentului registrelor de a încărca datele primite la intrare **numai** la momente bine stabilite de timp (în discuția noastră pe frontul crescător al impulsului de tact).

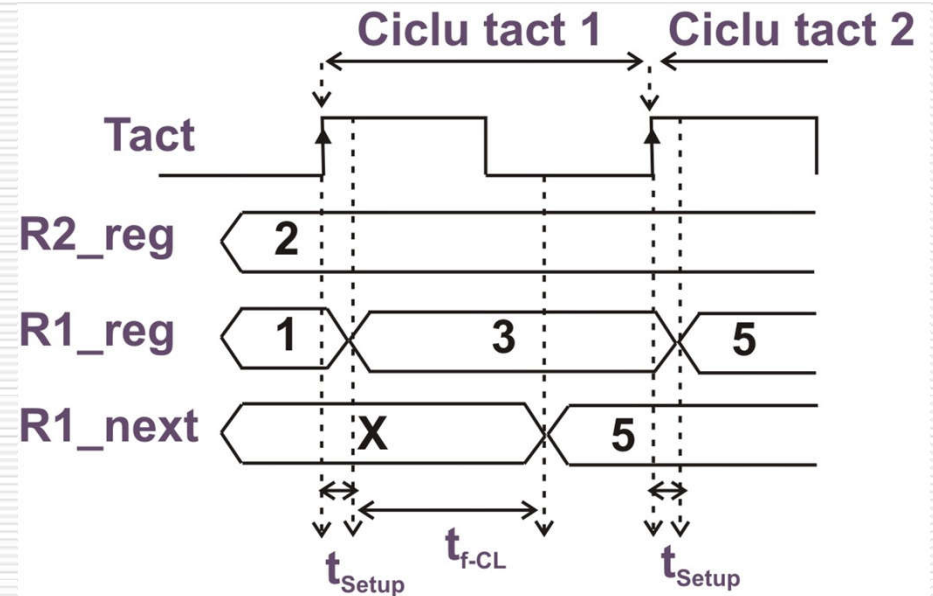
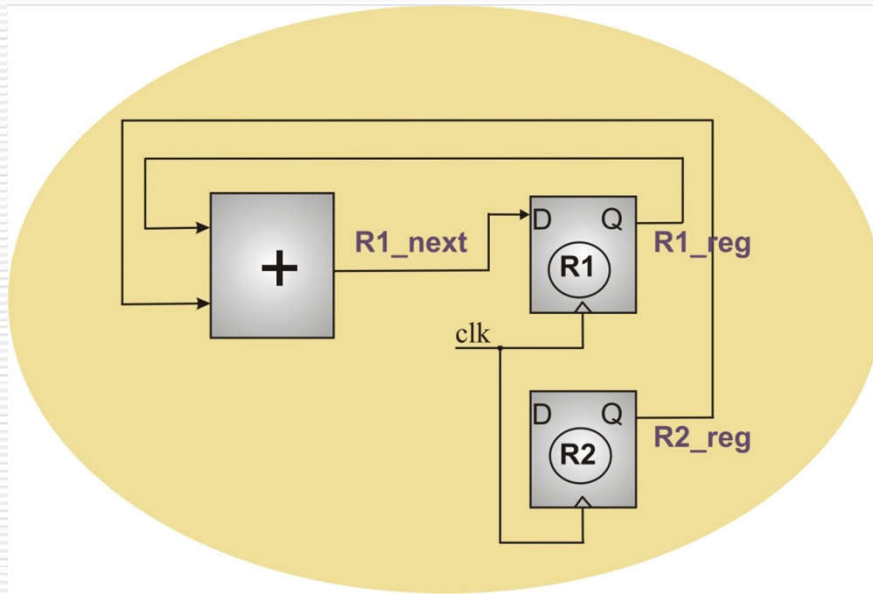
Așadar operația RT se poate exprima astfel:

$$R_1_next \leftarrow R_1_reg + R_2_reg$$

$$R_1_reg \leftarrow R_1_next \quad \text{la intalnirea primului front crescator al tactului}$$

Ex.: suma

□ Rezultat sinteza & diagrama de timp



$$T_{TACT} > t_{f-CL} + t_{SETUP-reg}$$

Mai multe operatii RT

Ex.:

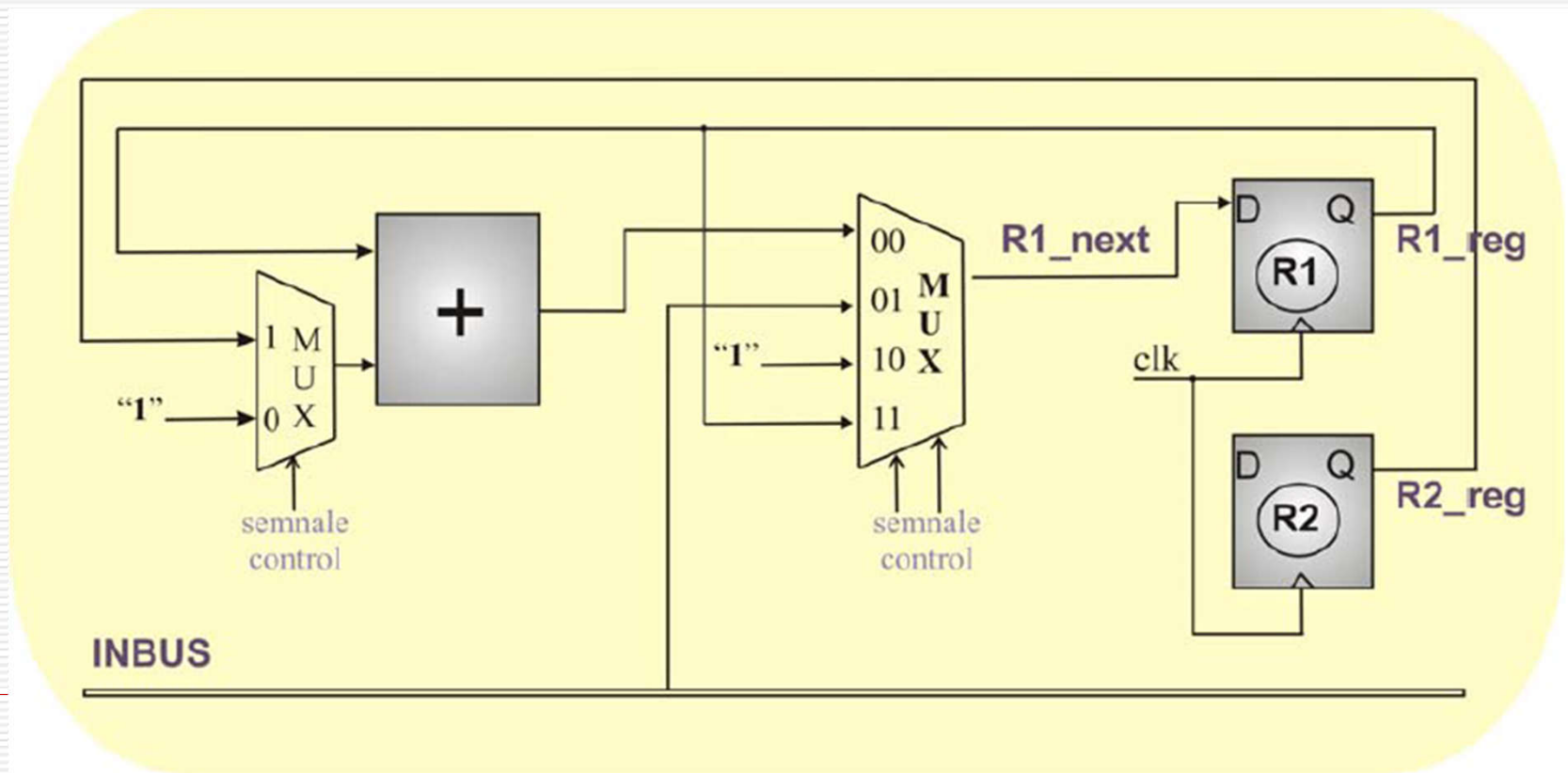
$$R_1 \leftarrow R_1 + R_2$$

$$R1 \leftarrow R1 + 1$$

$$R1 \leftarrow InBus$$

$$R1 \leftarrow 1$$

$$R1 \leftarrow R1 \quad - \text{nop}$$

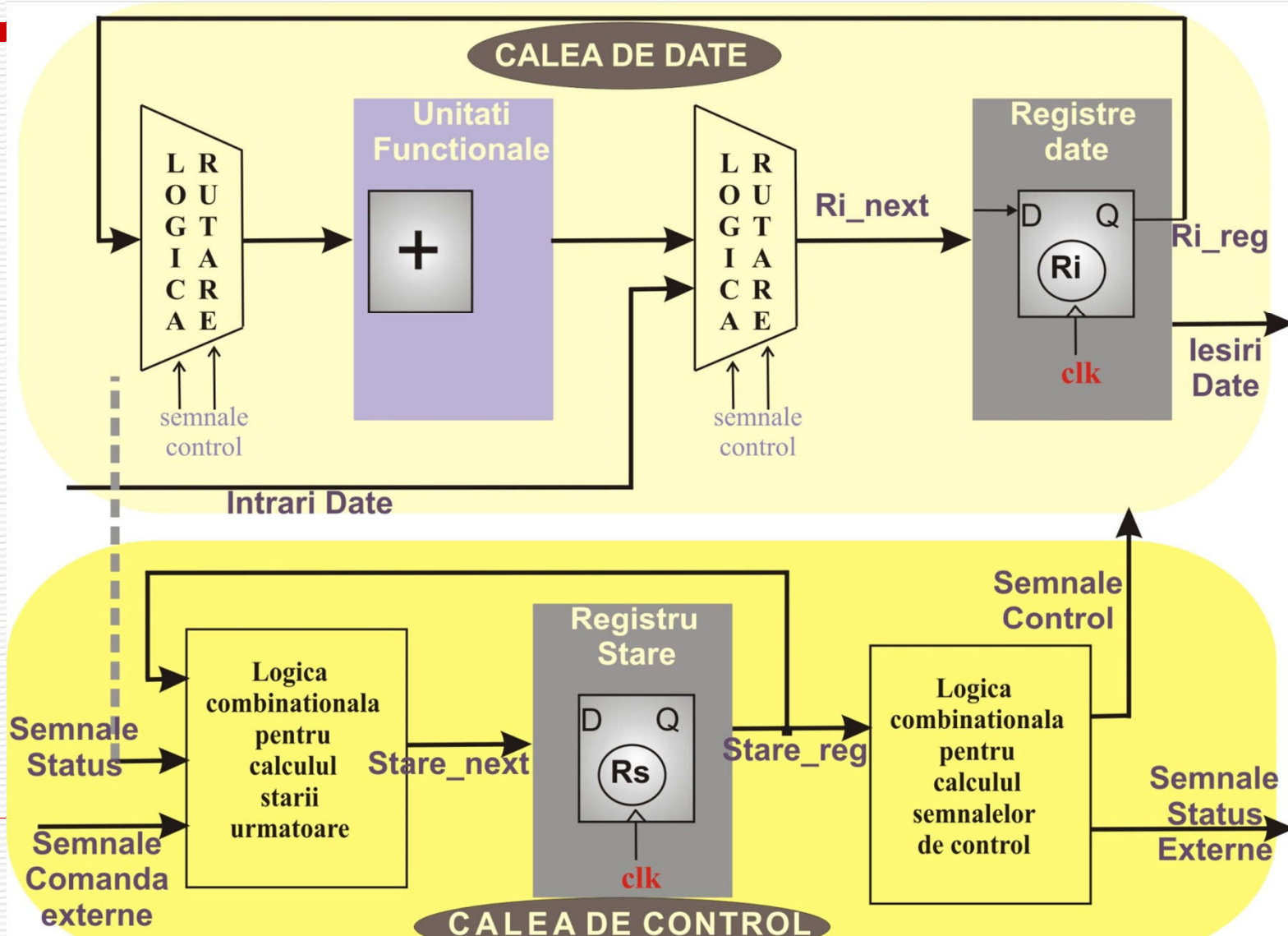


Automat cu stări finite și cale de prelucrare a datelor (FSMD)

O implementare RTL necesită:

- Cale date:
 - prelucrarea și rutarea datelor de către elemente secventiale de memorare.
 - unități funcționale de prelucrare a datelor
 - logică de rutare (de regulă reprezentată de multiplexoare)
 - registre pentru stocarea datelor
 - cale de control: dictează când și ce operație RT se execută (FSM).
-

FSMD



Exemplu 2

- Realizati un automat cu stari finite sincron care are o intrare pe 1 bit, X , si doua iesiri, RdY , $HINT$. Valoarea iesirii Rdy este 1, daca si numai daca $X = 0$, si secventa de valori de intrare pentru X timp de 7 impulsuri de tact succesive a fost "0110111". Iesirea $HINT$ este 1 daca si numai daca intrarea X are o valoare care respecta secventa de mai sus.

Exemplu 2

- Acest exemplu se preteaza pentru un automat cu stari finite de tip ...?
-

Exemplu 2

- ❑ Acest exemplu se preteaza pentru un automat cu stari finite de tip Mealy.
 - ❑ Valoare iesirii RdY depinde de istoria intrarii X (stare) si de valoarea curenta a lui X;
 - ❑ Valoarea lui HINT depined de starea curenta si de intrarea X.
 - ❑ daca HINT este 0, atunci utilizatorul poate folosi info. pentru a schimba X-ul cu valoare corecta pana la proximal pos edge.
-

Exemplu2: secventa unlock

Semnificatie	S	X	
		0	1
Got no valid seq. data	A	B, 01	A, 00
Got "0"	B	B, 00	C, 01
Got "01"	C	B, 00	D, 01
Got "011"	D	E, 01	A, 00
Got "0110"	E	B, 00	F, 01
Got "01101"	F	B, 00	G, 01
Got "011011"	G	E, 00	H, 01
Got "0110111"	H	B, 11	A, 00
		S*, RdY HINT	

Exemplu2: secventa unlock

Codificare	S	X	
		0	1
000	A	001, 01	000, 00
001	B	001, 00	010, 01
010	C	001, 00	011, 01
010	D	100, 01	000, 00
011	E	001, 00	101, 01
101	F	001, 00	110, 01
110	G	100, 00	111, 01
111	H	001, 11	000, 00
		(Q2_nxt, Q1_nxt, Q0_nxt)*, RDY HINT	

Exemplu2: secventa unlock

Inlocuirea simbolurilor starilor cu
codificarea aferenta

Determinarea logicii aferente calculului
noilor stari (minimizare si FF-uri de tip
D)

Determinarea logicii aferente calculului
iesirii (minimizare)

Exemplu2: secventa unlock

Realizarea schemei

Realizarea diagramei de stare

Homework

- ☐ De terminat
 - ☐ Realizati sinteza folosind bistabile de tip J-K
 - ☐ Realizati sinteza folosind FF-urile tip T
-

Întrebări?

**Enough Talking Let's Get To It
!!001race Yourself!!**

