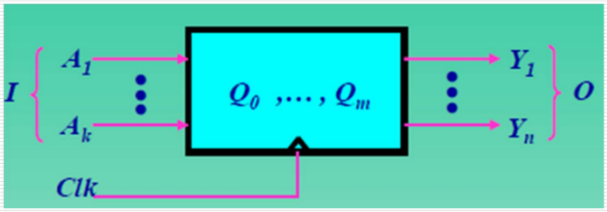
L6
(curs 10 - 510)

19.04.2024

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



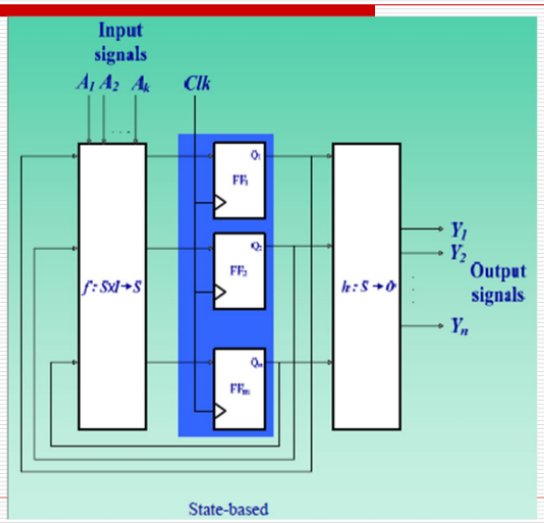
- Cvadruplul $\langle S, I, O, f, h \rangle$
  - S – mulțimea stărilor
  - I – mulțimea intrărilor
  - O – mulțimea ieșirilor
  - f- funcțiile pt.starea urm.; h – funcțiile pt.ieșire

$$S = Q_1 \times Q_2 \times \ldots \times Q_m,$$
$$I = A_1 \times A_2 \times \ldots \times A_k,$$
$$O = Y_1 \times Y_2 \times \ldots \times Y_n,$$

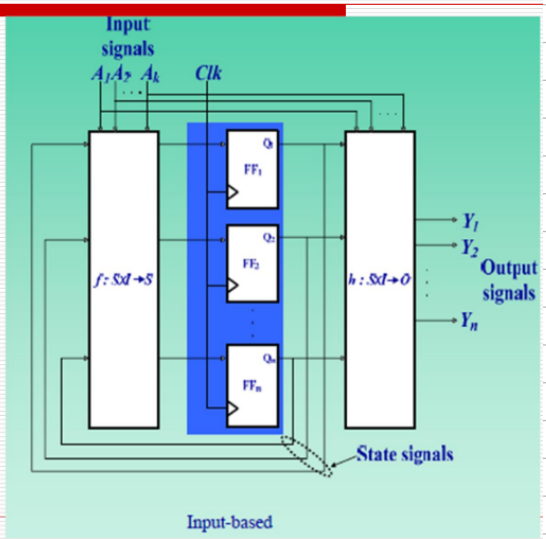
$$f : S \times I \rightarrow S$$
$$h : S \times I \rightarrow O \ (\textit{Mealy-type})$$
$$S \rightarrow O \ (\textit{Moore-type})$$

- **Circuitele secvenţiale:**
  - **MEALY** sunt caracterizate prin faptul că starea următoare şi **ieşirea** la un moment dat depind de starea prezentă si de intrarea prezentă;
  - **MOORE** sunt caracterizate prin faptul că **ieşirea** depinde **numai** de starea circuitului. Starea următoare depinde de intrarea prezentă;
- Modelele matematice ale circuitelor secvenţiale se numesc in teoria comutaţiilor **automate finite**.

Implementare FSM Moore



Implementare FSM Mealy



Etape de sinteză circuit secvențial

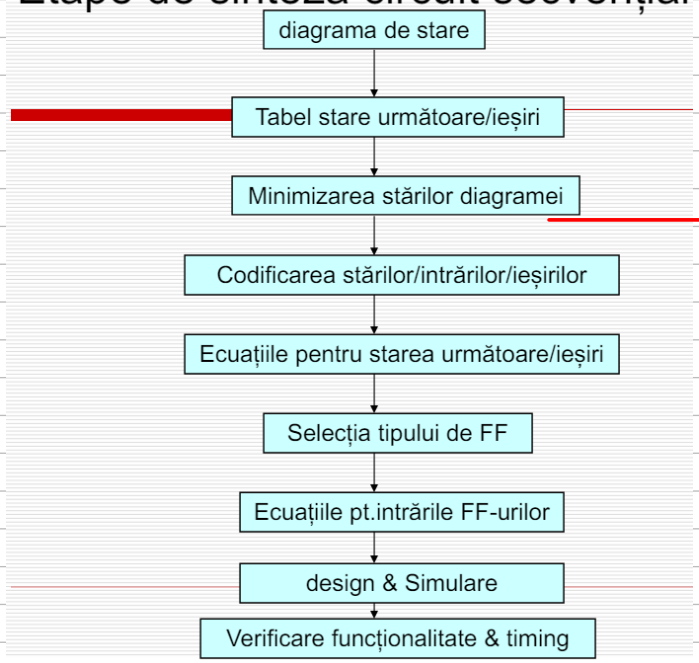


diagrama completă nu prezintă nr. minim de stări

- 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 aceași secventă de valori de ieșire pentru aceași secventă 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 $f(s_j, i) = f(s_k, i)$
- 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 şi A2 sunt echivalente dacă pentru fiecare stare s_j din A2 există o stare echivalentă s_k în A1 şi invers pentru fiecare stare s_j din A1 există o stare echivalentă s_k în A2
- Echivalenţa stărilor unui automat complet definit împarte mulţimea stărilor acestuia în clase de echivalenţă disjuncte.
- Relaţia de echivalenţă a stărilor automatului complet definit are **proprietatea de tranzitivitate:** dacă şi atunci.

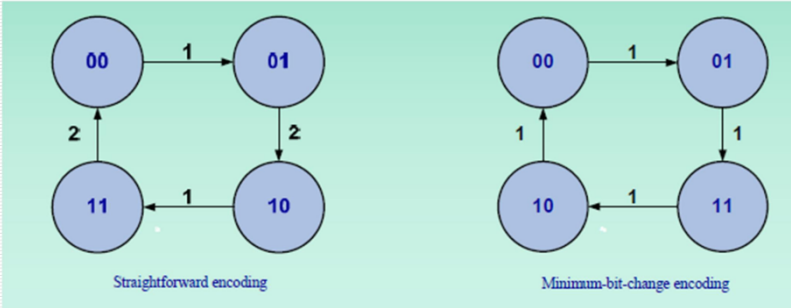
Codificarea stărilor

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**

- **Număr minim de tranziții (minimum bit change):**
  - Stările sunt astfel codificate încât să fie minimizate 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



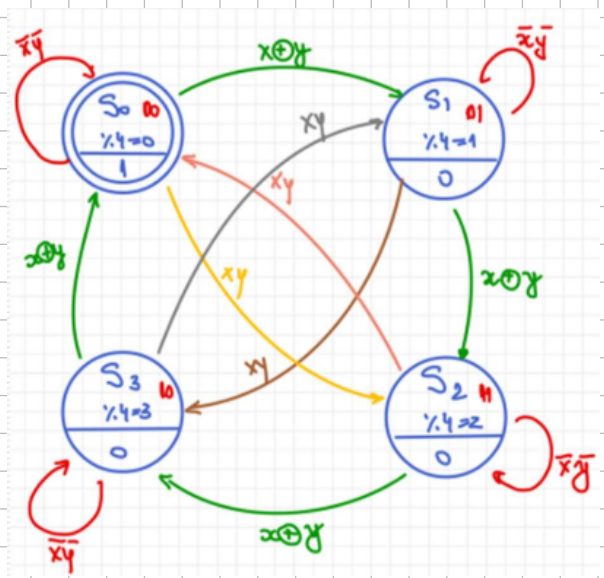
- **Număr minim de tranziții**
- **Adiacență pe bază de priorități (Prioritized adjancency strategy):**
  - **Codificări adiacente pentru stările:**
    - **destinație comună**
    - **Sursă comună**
    - **Ieșire comună**
  - **Next state va apărea in căsuțe adiacente în K-map;**
- **one-hot**
  -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

- Stare initiala care este necesara la reset (ex. 000, sau 111);
- Minimizati numarul de biti ai variabilei de stare care comuta;
- Maximizati grupurile de stari care genereaza tranzitii catre stari din cadrul grupului;
- Nu va limitati la ordinea crescatoare!
- Impartiti starile in asa fel incat sa corespunda la modificari ale unor grupuri de semnale de intrare!
- Adaugati biti suplimentari la reprez. starii …

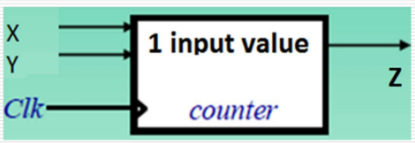
Design-ul FSM-urilor folosind diagrame de stare

- 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.

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



- Realizați design-ul pentru un automat cu stari cu 2 intrari: X si Y si o iesire Z. Iesirea este 1 daca numarul de valori de intrare 1 pentru X si Y de la reset este multiplu de 4, si 0 in caz contrar *.



- 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 (S0÷ S3).
- Pasul 1: Realizarea tabelului starilor si iesirilor

Semnificatie	S	XY				Z
		00	01	10	11	
Got 0 "1" (mod 4)	S0	S0	S1	S1	S2	1
Got 1 "1" (mod 4)	S1					0
Got 2 "1" (mod 4)	S2					0
Got3 "1" (mod 4)	S3					0

Numărător valori de 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

Inlocuirea simbolurilor starilor cu codificarea aferenta

d_1 Q_1Q_0\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	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

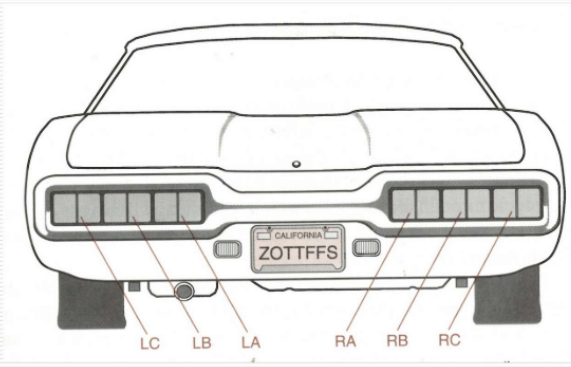
Q1(next)Q0(next) sau d1, d0

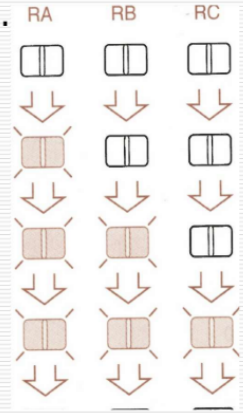
d_0 Q_1Q_0\XY	00	01	11	10
00	0	1	1	1
01	1	1	0	1
10	1	0	0	0
11	0	0	1	0

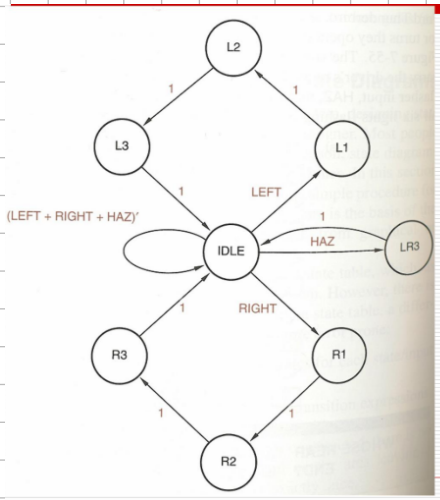
$$D_1 = Q_1\cdot\bar{x}\cdot\bar{y} + Q_0\cdot\bar{x}\cdot y + \bar{Q_1}\cdot x\cdot y + Q_0\cdot x\cdot\bar{y}$$
$$= Q_0(\bar{x}\cdot y + x\cdot\bar{y}) + Q_1\cdot\bar{x}\cdot\bar{y} + \bar{Q_1}\cdot x\cdot y$$
$$= Q_0\cdot(x\oplus y) + Q_1\cdot\bar{x}\cdot\bar{y} + \bar{Q_1}\cdot x\cdot y$$
$$D_0 = Q_0\cdot\bar{x}\cdot\bar{y} + \bar{Q_1}\cdot\bar{x}\cdot y + \bar{Q_0}\cdot x\cdot y + \bar{Q_1}\cdot x\cdot\bar{y}$$
$$= \bar{Q_1}(x\oplus y) + Q_0\cdot\bar{x}\cdot\bar{y} + \bar{Q_0}\cdot x\cdot y$$

Ex.: semnalizare Ford Thunderbird 1965

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





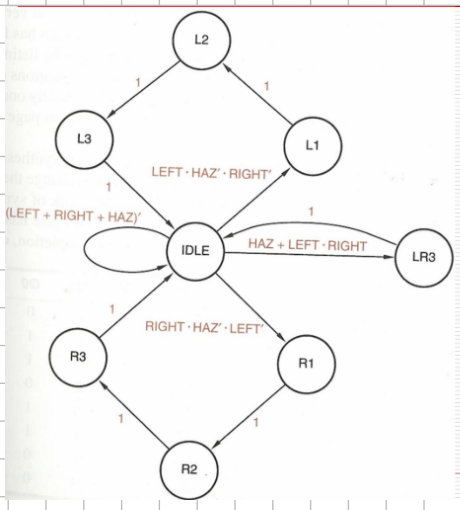


Output Table

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

$LA = L1 + L2 + L3 + LR3$
$LB = L2 + L3 + LR3$
$LC = L3 + LR3$
$RA = R1 + R2 + R3 + LR3$
$RB = R2 + R3 + LR3$
$RC = R3 + LR3$

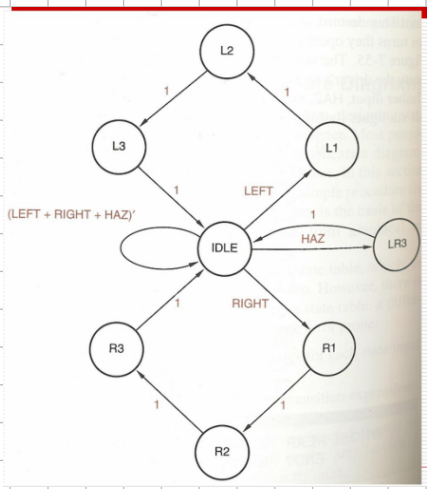
Ce se intampla daca LEFT SI HAZ sunt asertuite simultan? ---- ambiguizare



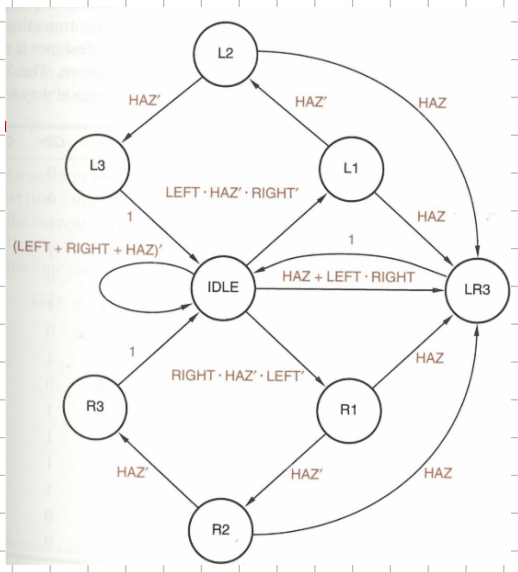
- Solutie: Ii dam la intrarea HAZ prioritate, iar cazul LEFT si RIGHT assertuite simultan il tratam ca si hazard.
- Noua diagrama nu e ambigua: conditiile de pe arce se exclud reciproc si surprind toate combinatiile de intrare!

Diagrama Ne-amigua

- Mutual exclusion: produs logic intre oricare 2 expresiii 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.



- 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



State	Q2	Q1	Q0
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

S	Q2	Q1	Q0	Transition Expression	S*	Q2*	Q1*	Q0*
IDLE	0	0	0	(LEFT + RIGHT + HAZ)'	IDLE	0	0	0
IDLE	0	0	0	LEFT · HAZ' · RIGHT'	L1	0	0	1
IDLE	0	0	0	HAZ + LEFT · RIGHT	LR3	1	0	0
IDLE	0	0	0	RIGHT · HAZ' · 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;

Examplu 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.

- 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 proximul pos edge.

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
011	D	100, 01	000, 00
100	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	

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)

Realizarea schemei

Realizarea diagramei de stare