Eln_Dig **Curs 1:**	***Implementarea FSM sincrone folosind codificarea „One Hot"***

1. Introducere

Metoda clasică de implementare a automatelor cu stări finite sincrone, bazată pe codificarea binară a stărilor, devine greoaie și aproape inoperabilă pentru aplicații de control în care avem un număr mare de semnale de intrare și/sau secvența de stări este lungă. Pentru astfel de cazuri, o soluție mult mai convenabilă din punctul de vedere al reducerii efortului de calcul, constă în adoptarea metodei de codificare cunoscută sub denumirea de "One Hot".

♦ Graful de tranziție a stărilor

Graful de tranziție a stărilor este o metodă intuitiva ce permite descrierea şi verificarea rapidă a funcţionării unui automat. Pentru fiecare nod al grafului, reprezentat printr-un cerculeţ, se asociază o stare a automatului, iar fiecare arc orientat între două noduri corespunde tranziţiei între cele două stări conectate.

♦ **Principalele caracteristici ale codificării *one hot*** sunt prezentate succint în cele ce urmează:

- fiecare stare a automatul dispune de un bistabil propriu (un bistabil dedicat);
- pentru fiecare stare a automatului doar un singur bit este în unu logic, restul sunt în mod obligatoriu în zero logic;
- un exemplu de codificare *one hot* pentru un automat cu 4 stări distincte arată astfel:

  St_1 = **1**000,
  St_2 = 0**1**00,
  St_3 = 00**1**0,
  St_4 = 000**1**;
- numărul bistabililor din registrul de stare este egal cu numărul stărilor din graful de tranziție;
- numărul bistabililor este mare în raport cu alte metode de codificare – acesta este principalul dezavantaj al acestei metode de codificare a stărilor;
- efortul de calcul pentru sinteza schemei logice a automatului este foarte redus în raport cu alte metode de codificare;
- schema automatului este modulară în sensul că, aceiași schemă logică poate fi utilizată pentru implementarea fiecărei stări, diferă doar modul în care sunt făcute conexiunile de intrare/ieșire de la acest modul;
- această metodă este foarte utilă pentru automate cu număr mare de intrări și/sau de stări;
- identificarea stării active a automatului se poate face foarte simplu prin testarea bitului *hot* din codul stării. Se spune că stările sunt deja decodificate;

Metoda *one hot* merită o atenție deosebită deoarece devine atractivă mai ales în cazul automatelor cu număr mare de intrări, acolo unde metoda clasică devine greoaie, necesită un efort mare de calcul și nu mai este atractivă.

Ușurința cu care se poate sintetiza schema logică a automatului prin această metodă de codificare vine și compensează principalul său dezavantaj: numărul mare de bistabili.

În acest curs se prezintă modul de lucru pentru proiectarea și implementarea automatelor sincrone bazate pe metoda "One Hot", pentru o aplicație de control destinată unei uși de garaj.

2. Etape în proiectarea FSM sincron bazat pe metoda "One Hot"
= cazul implementării cu bistabili D =

Pentru a ușura modul de înțelegere a etapelor ce trebuie parcurse pentru proiectarea, implementare și simularea schemelor sincrone de FSM bazate pe metoda de codificare "One Hot", considerăm o aplicație simplă de control, o aplicație pe care o întâlnim la tot pasul: sistem de control pentru ușa de garaj.

♦***Pasul 1: Cunoașterea instalației tehnologice.*** Primul pas în procesul de proiectare constă în înțelegerea foarte bună a procesului tehnologic ce trebuie condus.

Referitor la aplicația de față, avem următoarele componente de intrare (senzori sau butoane de interfațare cu operatorul uman) și de ieșire (elemente de execuție sau lămpi de semnalizare):

Pe partea de intrări, sistemul prezintă următoarele elemente, toate fiind active pe unu logic:
- Un buton cu revenire pentru inițializarea sistemului, notat *Init* ;
- Un buton cu revenire pentru deschidere ușă, notat *Open* ;
- Un buton cu revenire pentru închidere ușă, notat *Close* ;
- Un senzor de poziție ce semnalizează cazul în care ușa este complet închisă, notat *LS_Up* ;
- Un senzor de poziție ce semnalizează cazul în care ușa este complet deschisă, notat *LS_Dn* ;

Pe partea de ieșiri, sistemul prezintă următoarele elemente, toate fiind active pe unu logic :
- Un releu pentru comanda motorului electric necesar la deschiderea ușii, notat *M_Up* ;
- Un releu pentru comanda motorului electric necesar la închiderea ușii, notat *M_Dn* ;
- Lampă de semnalizare ușă complet deschisă, notată *L_Open* ;
- Lampă de semnalizare ușă complet închisă, notată *L_Close* ;
- Lampă de semnalizare ușă întredeschisă, notată *L_Air* ;

♦***Pasul 2: Proiectarea grafului de tranziție a stărilor (Cunoașterea modului de lucru a instalației)***

Din analiza modului în care trebuie să funcționeze sistemul de control pentru ușa de garaj, deducem graful de tranziție din figura 1.

În acest graf, pe tranziții sunt trecute condițiile senzoriale care determină ieșirea din stare. Dacă aceste condiții nu sunt valide (active), automatul rămâne în starea în care era. Spre exemplu, din starea *St0* nu se poate ieși decât prin apăsarea butonului de *Open*.

Trebuie reamintit faptul că, în orice moment de timp, doar o singură stare din graf este activă, restul sunt în mod obligatoriu inactive.

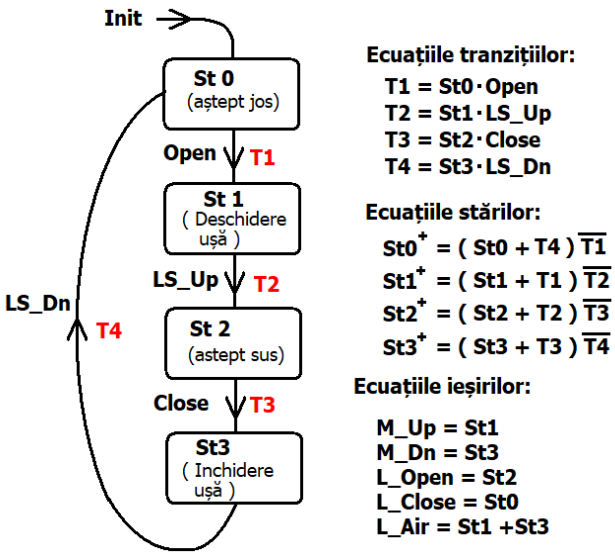

Fig. 1: Graful de tranziție a stărilor pentru controlul unei uși de garaj – varianta fără facilitatea de stop

♦*Pasul 3: Codificarea stărilor*

Având în vedere numărul mare de intrări, este preferabilă codificarea *one hot* deoarece reduce semnificativ efortul de calcul. Pentru cazul de față am ales următoarea codificare:

- **St0 = 1000**;
- **St1 = 0100**;
- **St2 = 0010**;
- **St3 = 0001**;

O problemă importantă în proiectarea unui automat este dată de alegerea modului în care se face inițializarea bistabililor din registrul de stare.

În cazul de față, folosim o **inițializare asincronă:**

- intrarea *Init* este conectată la intrarea *Preset* a bistabilului asociat Stării *St0* ,
- intrarea *Init* este conectată la intrările de *Clear* ale celorlalți bistabili.

În acest mod, prin activarea de scurtă durată a intrării *Init*, avem siguranța că starea bistabililor devine ***1000***, ceea ce corespunde stării inițiale *St0*.

♦*Pasul 4: Deducerea ecuațiilor caracteristice automatului*

a) Determinarea expresiilor pentru tranziții

În această etapă, pentru fiecare tranziție din graf se scrie un produs logic de forma:

T = starea de unde pleacă tranziția x *ce scrie pe tranziție*

Pe baza acestei reguli, pentru tranzițiile din graful alăturat se obțin următoarele expresii

$$T1 = St0 \cdot Open$$

$$T2 = St1 \cdot LS\_Up$$

$$T3 = St2 \cdot Close$$

$$T4 = St3 \cdot LS\_Dn$$

b) Determinarea expresiilor pentru starea viitoare – varianta necesară implementării cu bistabili D

Pentru fiecare stare din graf se scrie o ecuație de forma:

$$St^{+} = (St + \sum T\_intrare\ ) \prod \overline{T\_iesire}$$

În ecuația unei stări sunt prezente trei componente:

- Suma tranzițiilor de intrare în stare (în interiorul parantezei) – practic acestea sunt condițiile de activare a stării respective;
- Codul stării (în interiorul parantezei) - *denumită condiție de automenținere a stării*. Condiția de automenținere este obligatorie deoarece o parte din comenzile de intrare sunt active doar un timp limitat (activare de tip impuls). La activarea acestor comenzi, automatul trebuie să între într-o stare și apoi trebuie să mențină acea stare chiar dacă, între timp comanda de intrare s-a dezactivat. Spre exemplu, la apăsarea temporară a butonului *Open*, automatul trebuie să intre în starea St1 și să mențină această stare, chiar dacă, între timp, butonul Open nu mai este apăsat.
- Produsul format din negatele tranzițiilor de ieșire din stare (în exteriorul parantezei) - practic acestea sunt condițiile de dezactivare/terminare a stării respective;

Pe baza celor de mai sus, pentru ecuațiile de stare se obțin următoarele expresii:

$$St0^{+} = (\ St0 + T4\ ) \cdot \overline{T1}$$

$$St1^{+} = (\ St1 + T1\ ) \cdot \overline{T2}$$

$$St2^{+} = (St2 + T2) \cdot \overline{T3}$$

$$St3^{+} = (St3 + T3) \cdot \overline{T4}$$

c) Determinarea expresiilor pentru ieșiri

Pe baza analizei grafului și a modului de funcționare a sistemului de control deducem următoarele expresii pentru ieșirile din automat:

M_Up = St1

M_Dn = St3

L_Close = ST0

L_Open = ST2

L_Air = ST1+ST3

♦Pasul 5: Trasarea schemei logice a automatului

a) Logica de inițializare

Este cunoscut faptul că după pornirea tensiunii de alimentare starea bistabililor nu poate fi precizată, în sensul că poate diferii de la o pornire la alta. Din acest motiv, în schemele ce conțin bistabili este strict necesar să avem o logică de inițializare, logică ce trebuie să aducă bistabilii într-o stare cunoscută, mereu aceeași, după fiecare pornire a tensiunii de alimentare sau după fiecare comandă de inițializare a schemei. De regulă, inițializarea schemelor se face acționând asupra intrărilor asincrone de *Set/Reset* sau, după caz, asupra intrărilor *Preset/Clear*.

În cazul de față, referitor la inițializare procedăm astfel:

- prin activarea temporară a intrării *Init*, schema trebuie să ajungă în starea **St0 = 1000**;
- intrarea de *Init* se conectează la intrarea de *Preset* ce aparține bistabilului utilizat pentru memorarea stării *St0* ;
- intrarea de *Init* se conectează la toate intrările de *Clear* de la restul bistabililor (bistabili folosiți pentru stările *St1, St2* și *St3*);
- Deoarece intrarea de *Init* este activă pe unu logic, iar intrările de *Preset* sau de *Clear* sunt active pe zero logic, este necesar un inversor conectat așa cum se vede în figura 2, pe traseul desenat cu roșu.

b) Logica de control

După stabilirea modului în care se face inițializarea, pe baza relațiilor deduse la pasul anterior se poate trasa restul schemei logice. Pentru exemplu de față se obține schema desenată în figura 2.

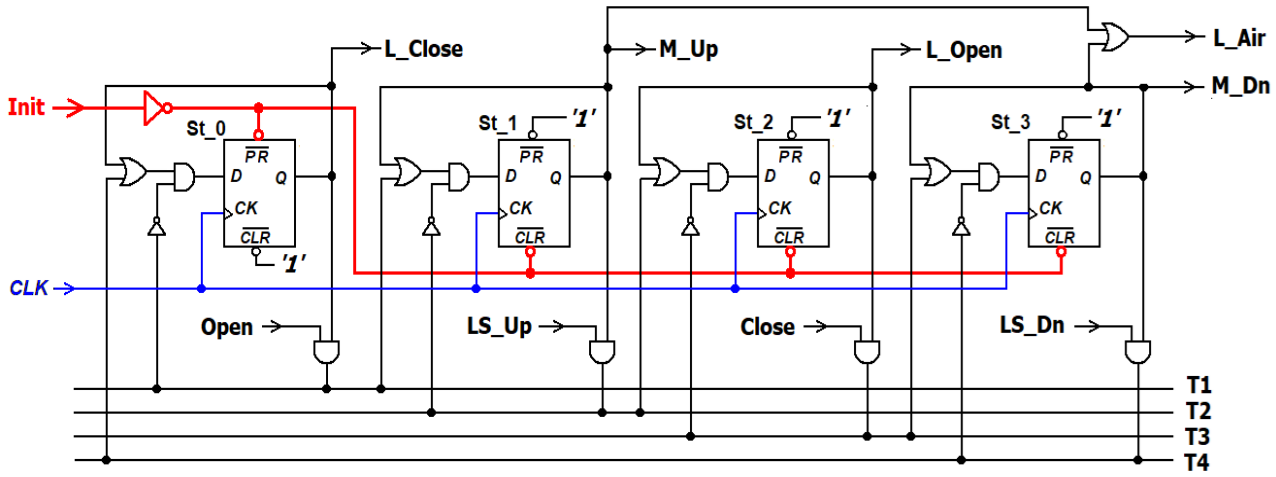

Fig. 2: Schema logică pentru sistemul de control pentru ușa de garaj (varianta 1)

Trebuie precizat și faptul că toate intrările de ceas ale bistabililor trebuie conectate între ele (traseul albastru) și trebuie comandate de către ceasul automatului (un semnal digital periodic cu frecvența de ordinul zecilor de kHz).

O altă variantă de schemă logică, fără inversoare, se obține dacă realizăm următoarele operații pe ecuațiile de stare:

$$St0^{+} = (St0 + T4) \cdot \overline{T1} = A \cdot \overline{T1} = \overline{\overline{A} + T1} = \overline{\overline{(St0+T4)} + T1} = NOR(NOR(St0,T4),T1)$$

$$St1^{+} = (St1 + T1) \cdot \overline{T2} = \ldots = NOR(NOR(St1,T1),T2)$$

$$St2^{+} = (St2 + T2) \cdot \overline{T3} = \ldots = NOR(NOR(St2,T2),T3)$$

$$St3^{+} = (St3 + T3) \cdot \overline{T4} = \ldots = NOR(NOR(St3,T3),T4)$$

Din analiza ecuațiilor anterioare se vede că tranzițiile de ieșire nu mai au bară, din acest motiv nu mai avem nevoie de inversoare. Schema logică obținută pe baza ultimelor ecuații de stare este prezentată în figura 3.

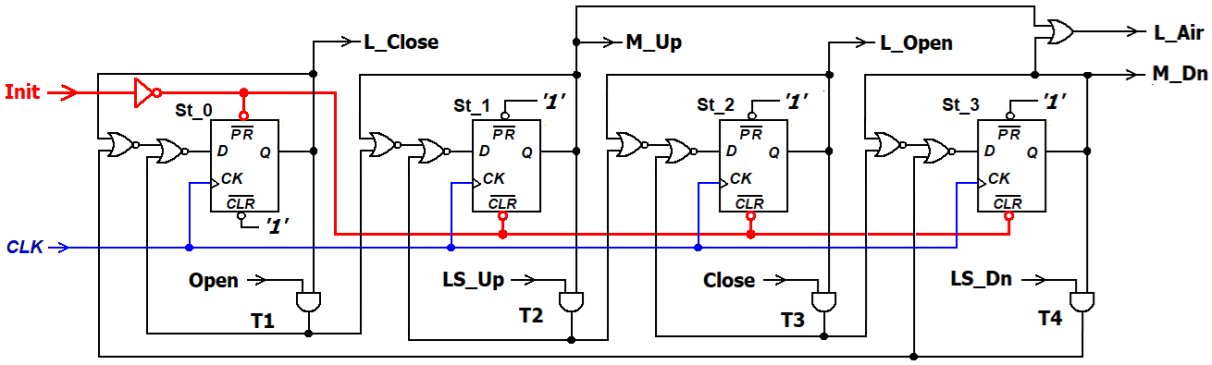

Fig. 3: Schema logică pentru sistemul de control pentru ușa de garaj (varianta 2)

O altă variantă de schemă logică, tot fără inversoare, se obține dacă realizăm următoarele operații pe ecuațiile de stare:

$$St0^{+} = (St0 + T4) \cdot \overline{T1} = \overline{\overline{(St0+T4)} \cdot T1} = \overline{\overline{St0} \cdot \overline{T4} \cdot T1}$$

$$St1^{+} = (St1 + T1) \cdot \overline{T2} = \overline{\overline{(St1+T1)} \cdot T2} = \overline{\overline{St1} \cdot \overline{T1} \cdot T2}$$

$$St2^{+} = (St2 + T2) \cdot \overline{T3} = \overline{\overline{(St2+T2)} \cdot \overline{T3}} = \overline{\overline{St2} \cdot \overline{T2} \cdot \overline{T3}}$$

$$St3^{+} = (St3 + T3) \cdot \overline{T4} = \overline{\overline{(St3+T3)} \cdot \overline{T4}} = \overline{\overline{St3} \cdot \overline{T3} \cdot \overline{T4}}$$

Din analiza ecuațiilor anterioare se vede că toate tranzițiile (indiferent că sunt de intrare sau de ieșire din stare) intervin în ecuațiile de stare doar sub formă negată. Din acest motiv, toate tranzițiile vor fi calculate cu porți NAND iar apoi folosim un NAND și un AND pentru fiecare intrare D. Trebuie avut atenție că, spre deosebire de cazurile anterioare, pentru automenținerea stării trebuie să preluăm semnal de la ieșirea $\overline{Q}$ a fiecărui bistabil. Schema logică obținută pe baza ecuațiilor anterioare ne conduc spre schema prezentată în figura 4.

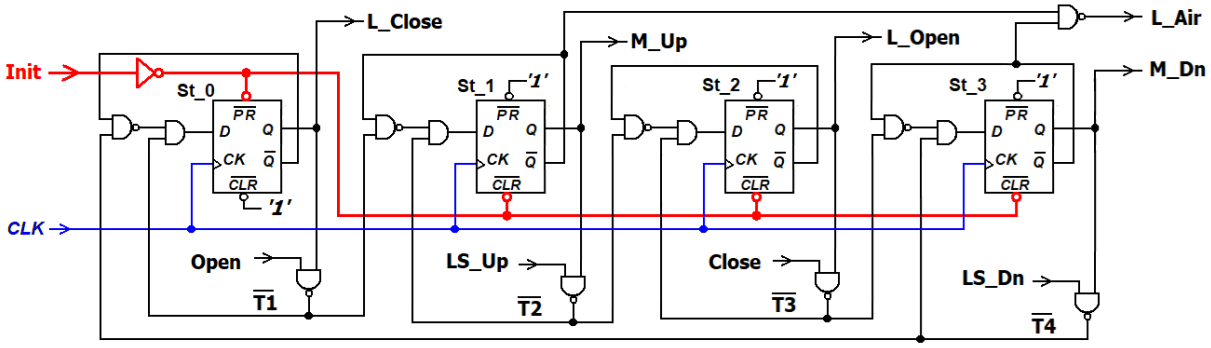

Fig. 4: Schema logică pentru sistemul de control pentru ușa de garaj (varianta 3)

În schema anterioară apare o modificare și pe partea de ieșire, în sensul că ieșirea *L_Air* este implementată cu un NAND, pe baza relației:

$$L\_Air = (St1 + St3) = \overline{\overline{(St1+St3)} \cdot} = \overline{\overline{St1} \cdot \overline{St3}}$$

Această procesare ne permite obținerea unei scheme ce folosește doar două tipuri de porți logice: AND și NAND. Este evident că, dacă este necesar, ultima schemă poate fi implementată destul de ușor doar cu porți NAND.

3. Etape în proiectarea FSM sincron bazat pe metoda "One Hot"
= cazul implementării cu bistabili JK =

♦Pasul 1: Cunoașterea instalației tehnologice.

Considerăm aceiași aplicație ca în cazul anterior cu deosebirea că, pe partea de intrări, apare în plus un buton cu revenire notat cu *Stop.* Apăsarea temporară a acestui buton va avea ca efect oprirea acțiunii de închidere/deschidere ușă.

♦Pasul 2: Proiectarea grafului de tranziție a stărilor.

De această dată, spre deosebire de aplicația anterioară, luăm în calcul și posibilitatea ca ușa să poată fi oprită în timp ce execută comanda de închidere sau de deschidere.

Referitor la oprire:
- oprirea se face prin apăsarea temporară a butonului de *Stop*,
- oprirea are ca efect ajungerea în starea St4;
- ieșirea din St4 (oprire) se poate face în două moduri:
  - prin apăsarea temporară a butonului *Open* – caz în care se lansează acțiunea de deschidere;
  - prin apăsarea temporară a butonului *Close* – caz în care se lansează acțiunea de închidere;

Graful de tranziție a stărilor pentru cazul integrării facilității de stop se poate vedea în figura ce urmează

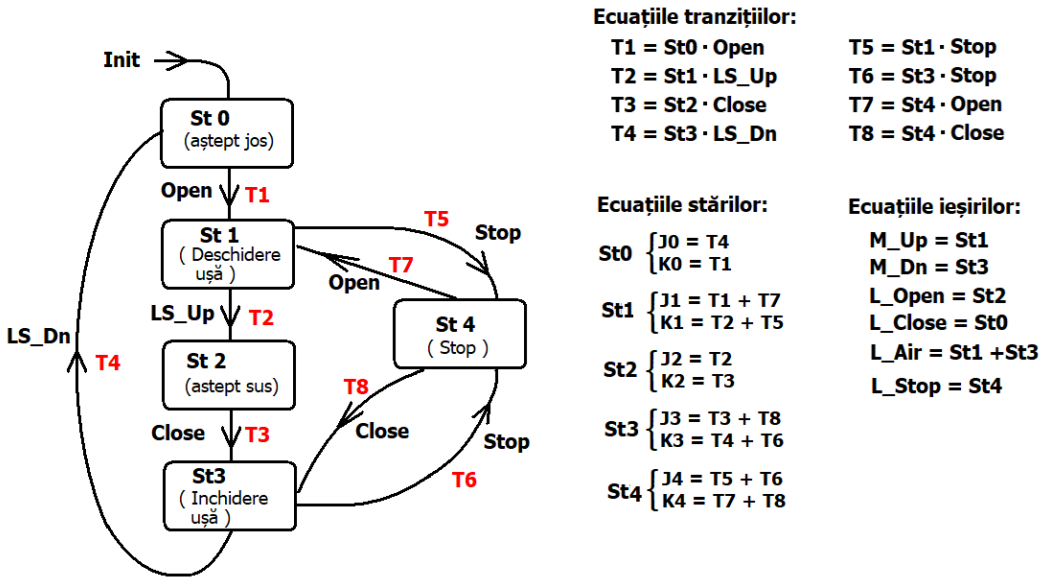

Fig. 5: Graful de tranziție și ecuațiile de stare concepute pentru implementarea cu bistabili JK

♦Pasul 3: Codificarea stărilor

Folosim codificarea *one hot* adoptată și în cazul anterior: **St0 = 10000**; **St1 = 01000**; **St2 = 00100**; **St3 = 00010**; **St4 = 00001**;

♦Pasul 4: Deducerea ecuațiilor pentru tranziții, stări viitoare și ieșiri

a) *Ecuațiile tranzițiilor*

Reamintim că pentru fiecare tranziție din graf se scrie un produs logic de forma:

Ti = starea de unde pleacă tranziția x *ce scrie pe tranziție*

Pe baza acestei reguli, pentru tranzițiile din graful anterior se obțin următoarele expresii:

$T1 = St0 \cdot Open$ $\quad$ $T5 = St1 \cdot Stop$

$T2 = St1 \cdot LS\_Up$ $\quad$ $T6 = St3 \cdot Stop$

$T3 = St2 \cdot Close$ $\quad$ $T7 = St4 \cdot Open$

$T4 = St3 \cdot LS\_Dn$ $\quad$ $T8 = St4 \cdot Close$

b) *Ecuațiile stărilor* diferă față de exemplul anterior prin faptul că în locul bistabililor de tip D trebuie să folosim bistabili de tip JK:

$$St0: \begin{cases} J0 = T4 \\ K0 = T1 \end{cases} \qquad St3: \begin{cases} J3 = T3 + T8 \\ K3 = T4 + T6 \end{cases}$$

$$St1: \begin{cases} J1 = T1 + T7 \\ K1 = T2 + T5 \end{cases} \qquad St4: \begin{cases} J4 = T5 + T6 \\ K4 = T7 + T8 \end{cases}$$

$$St2: \begin{cases} J2 = T2 \\ K2 = T3 \end{cases}$$

c) *Ecuațiile ieșirilor* sunt identice cu cele obținute în exemplele anterioare:

$M\_Up = St1$ $\quad$ $M\_Dn = St3$

$L\_Open = ST2$ $\quad$ $L\_Close = ST0$

$L\_Stop = ST4$ $\quad$ $L\_Air = ST1+ST3$

♦ Pasul 5: Trasarea schemei logice a automatului

a) *Logica de inițializare* – este realizată asincron, la fel ca în exemplul anterior

b) *Logica de control* – se obține ținând cont de relațiile găsite în pasul anterior. Pe baza acestor relații se poate trasa schema ce urmează.

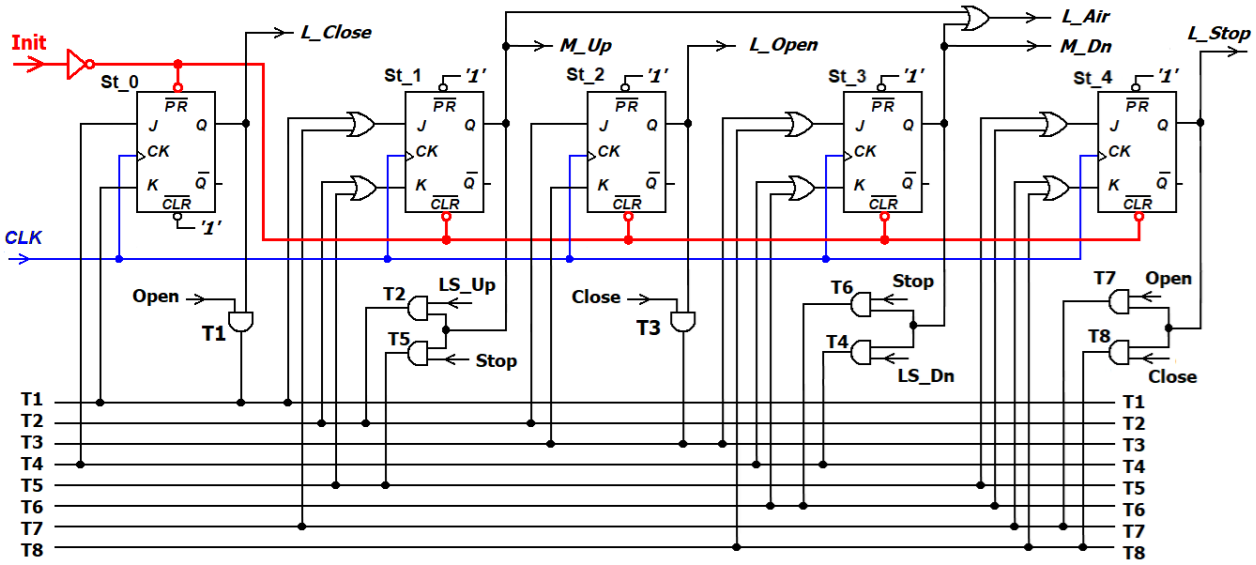

Fig. 6: Schema logică pentru sistemul de control realizată cu bistabili JK (varianta 1)

O altă variantă a schemei logice, bazată doar pe porți NAND, se obține dacă prelucrăm ecuațiile de stare așa cum se prezintă mai jos:

$$St0: \begin{cases} J0 = T4 = \overline{\overline{T4}} \\ K0 = T1 = \overline{\overline{T1}} \end{cases} \qquad St3: \begin{cases} J3 = T3 + T8 = \overline{\overline{T3} \cdot \overline{T8}} \\ K3 = T4 + T6 = \overline{\overline{T4} \cdot \overline{T6}} \end{cases}$$

$$St1:\begin{cases} J1 = T1 + T7 = \overline{\overline{T1}\cdot\overline{T7}} \\ K1 = T2 + T5 = \overline{\overline{T2}\cdot\overline{T5}} \end{cases} \qquad St4:\begin{cases} J4 = T5 + T6 = \overline{\overline{T5}\cdot\overline{T6}} \\ K4 = T7 + T8 = \overline{\overline{T7}\cdot\overline{T8}} \end{cases}$$

$$St2:\begin{cases} J2 = T2 = \overline{\overline{T2}} \\ K2 = T3 = \overline{\overline{T3}} \end{cases}$$

Se constată că este nevoie să calculăm negatele tranzițiilor T1÷T8, iar pe baza acestora se pot implementa semnalele de comandă pentru intrările JK.

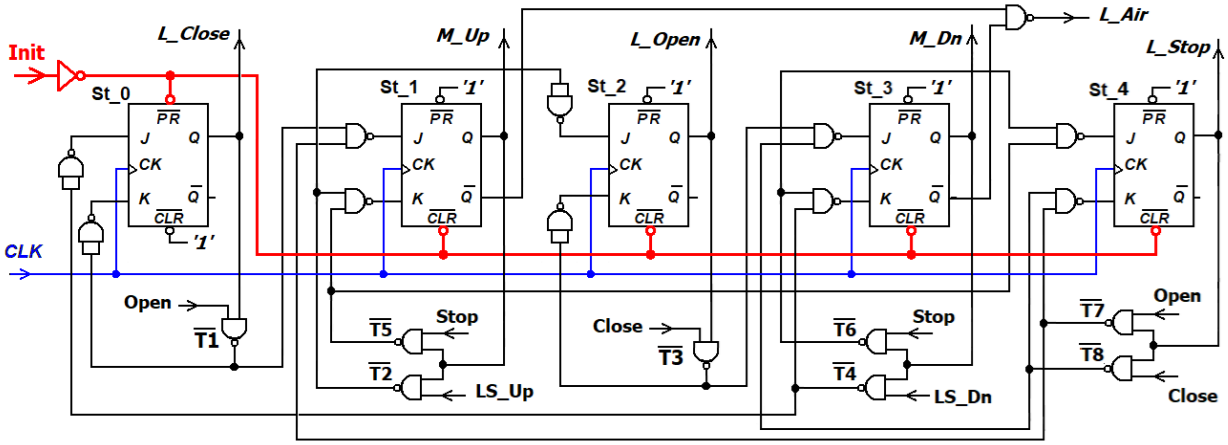

Fig. 7: Schema logică pentru sistemul de control realizată cu bistabili JK (varianta 2)

O analiză a schemei anterioare ne permite o estimare a resurselor hardware necesare implementării unui automat FSM sintetizat prin această tehnică:

- Numărul de bistabili JK este egal cu numărul stărilor distincte din graful de tranziție a stărilor (în cazul de față avem 5 bistabili);
- Numărul de porți NAND pentru logica de tranziție a stărilor este egal cu numărul de tranziții din graf plus dublul numărului de bistabili (în cazul de față avem 8 + 2·5 =18);
- Numărul de porți NAND pentru logica de tranziție a ieșirilor diferă de la o aplicație la alta însă nu poate fi mai mare decât numărul de ieșiri din automat (în cazul de față avem doar un NAND, restul ieșirilor sunt preluate direct din bistabili);