Eln_Dig **Curs 2:**	***I. Implementarea FSM sincrone, folosind codificarea „One Hot", cu bistabili JK*** ***II. Implementarea FSM asincrone, folosind codificarea „One Hot", cu bistabili JK***

I. Implementarea FSM sincrone, folosind codificarea „One Hot", cu bistabili JK

♦*Pasul 1: Cunoașterea instalației tehnologice.*

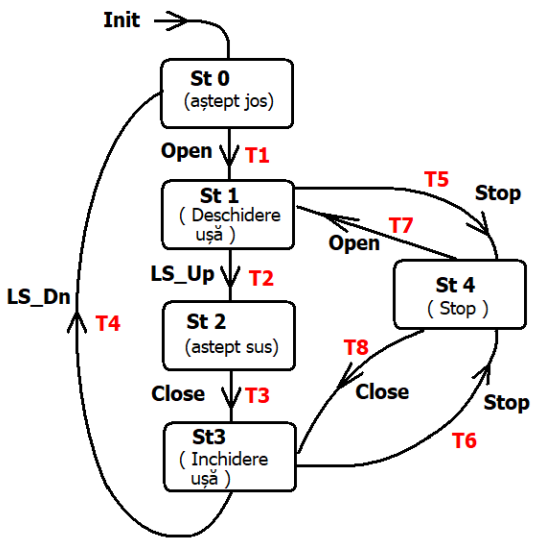
Considerăm aceiași aplicație ca în cazul anterior cu deosebirea că, pe partea de intrări, apare în plus un buton cu revenire notat cu *Stop.* Apăsarea temporară a acestui buton va avea ca efect oprirea acțiunii de închidere/deschidere ușă.

♦*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



Ecuațiile tranzițiilor:

$T1 = St0 \cdot Open$
$T2 = St1 \cdot LS\_Up$
$T3 = St2 \cdot Close$
$T4 = St3 \cdot LS\_Dn$
$T5 = St1 \cdot Stop$
$T6 = St3 \cdot Stop$
$T7 = St4 \cdot Open$
$T8 = St4 \cdot Close$

Ecuațiile stărilor:

$St0 \begin{cases} J0 = T4 \\ K0 = T1 \end{cases}$
$St1 \begin{cases} J1 = T1 + T7 \\ K1 = T2 + T5 \end{cases}$
$St2 \begin{cases} J2 = T2 \\ K2 = T3 \end{cases}$
$St3 \begin{cases} J3 = T3 + T8 \\ K3 = T4 + T6 \end{cases}$
$St4 \begin{cases} J4 = T5 + T6 \\ K4 = T7 + T8 \end{cases}$

Ecuațiile ieșirilor:

$M\_Up = St1$
$M\_Dn = St3$
$L\_Open = St2$
$L\_Close = St0$
$L\_Air = St1 + St3$
$L\_Stop = St4$

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.

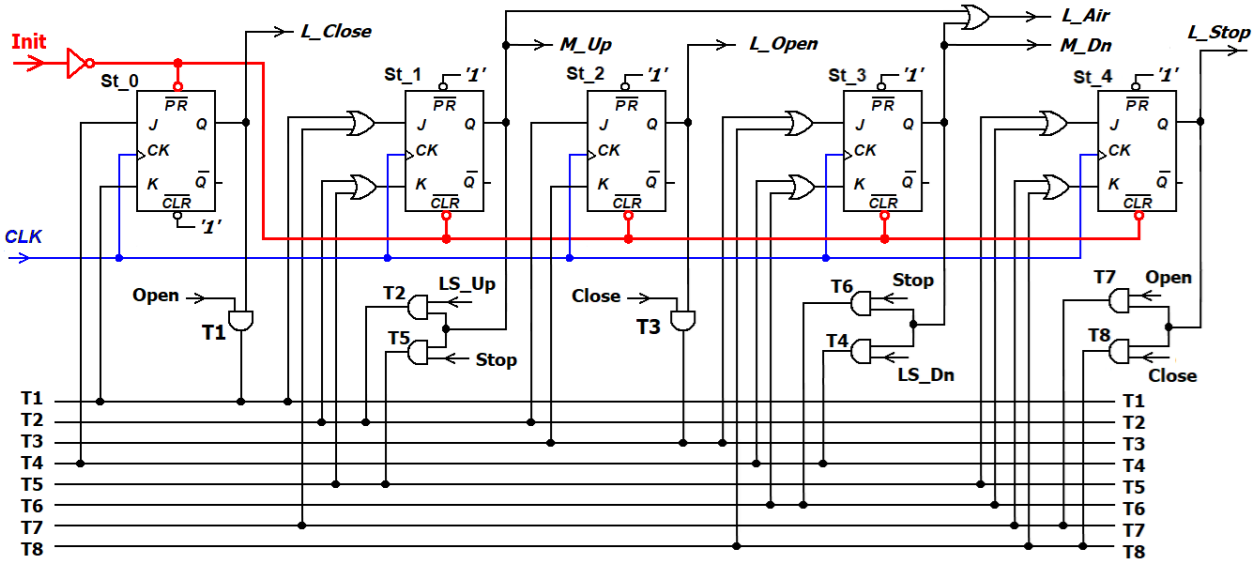


Fig. 6: Schema logică pentru sistemul de control realizată cu bistabili JK (varianta 1)

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.

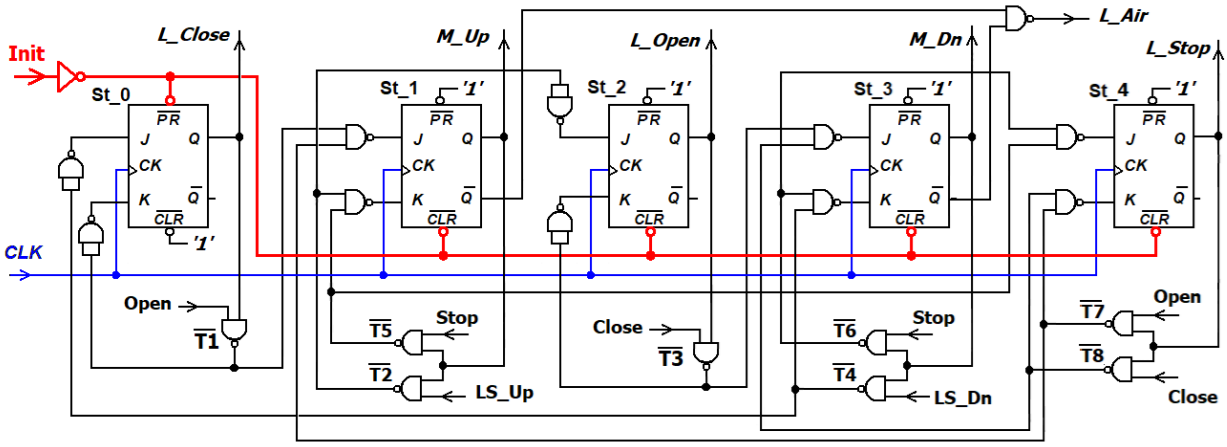


Fig. 7: Schema logică pentru sistemul de control realizată cu bistabili JK (varianta 2)

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);

II. Implementarea FSM asincrone, folosind codificarea „One Hot"

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. *Atenție! Acest pas a fost descris în cursul anterior.*

♦ *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 8.

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.

Graful de tranziție a stărilor prezintat în exemplul anterior rămâne nemodificat așa cum se poate vedea în figura ce urmează

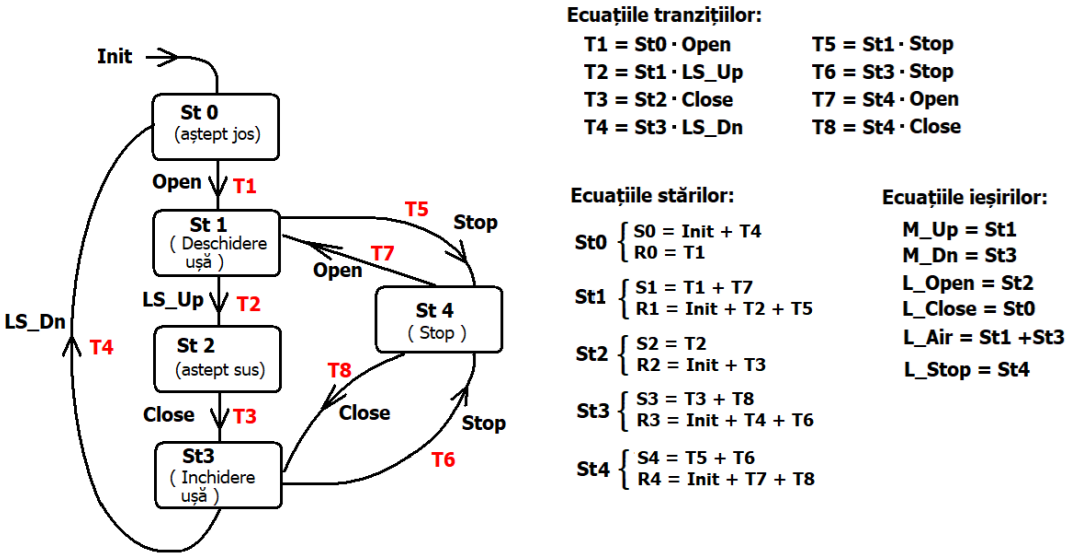


Fig. 8: Graful de tranziție și ecuațiile de stare concepute pentru implementarea asincronă (cu latch-uri)

♦ *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 elementelor de memorie 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 · Open
T2 = St1 · LS_Up
T3 = St2 · Close
T4 = St3 · LS_Dn
T5 = St1 · Stop
T6 = St3 · Stop
T7 = St4 · Open
T8 = St4 · Close

b) *Determinarea expresiilor pentru starea viitoare – varianta necesară pentru implementarea asincronă*

Pentru implementarea asincronă a automatelor FSM, pentru fiecare stare din graf se folosește una din următoarele două variante:

- un latch asincron format din două NAND-uri (latch-ul elementare);
- intrările asincrone (Preset și Clear) ale unui bistabil de orice tip – caz în care, intrările sincrone sunt legate la masă;

Pentru scrierea ecuațiilor de stare, trebuie respectate următoarele reguli:

- Pentru fiecare stare trebuie scrise două ecuații:
  - una pentru intrarea de *Set* (în cazul bistabililor această intrare poate fi referită și cu denumirea de *Preset*);
  - alta pentru intrarea de *Reset* (în cazul bistabililor această intrare poate fi referită și cu denumirea de *Clear*);
- Pentru intrarea de *Set*, este necesar să sumăm toate tranzițiile ce intră în starea aflată în discuție;
- Pentru intrarea de *Reset*, este necesar să sumăm toate tranzițiile ce ies din starea aflată în discuție,
- O atenție deosebită trebuie acordată intrării de inițializare (*Init*), această intrare trebuie să apară în fiecare stare o singură dată după cum urmează:
  - pe intrarea de *Set* ce aparține stării de pornire/inițializare a grafului;
  - pe intrările de *Reset* pentru restul stărilor;

Ecuațiile stărilor diferă față de exemplele anterioare prin faptul că în locul bistabililor de tip D trebuie să folosim bistabili de tip JK:

$St0: \begin{cases} S0 = Init + T4 \\ R0 = T1 \end{cases}$	$St3: \begin{cases} S3 = T3 + T8 \\ R3 = Init + T4 + T6 \end{cases}$
$St1: \begin{cases} S1 = T1 + T7 \\ R1 = Init + T2 + T5 \end{cases}$	$St4: \begin{cases} S4 = T5 + T6 \\ R4 = Init + T7 + T8 \end{cases}$
$St2: \begin{cases} S2 = T2 \\ R2 = Init + T3 \end{cases}$	

Ecuațiile anterioare trebuie prelucrate deoarece majoritatea elementelor de memorie au intrările asincrone active pe nivelul de zero.

$$St0:\begin{cases}\overline{S0} = \overline{Init + T4} = \overline{Init}\cdot\overline{T4}\\ \overline{R0} = \overline{T1}\end{cases}\qquad St3:\begin{cases}\overline{S3} = \overline{T3 + T8} = \overline{T3}\cdot\overline{T8}\\ \overline{R3} = \overline{Init + T4 + T6} = \overline{Init}\cdot\overline{T4}\cdot\overline{T6}\end{cases}$$

$$St1:\begin{cases}\overline{S1} = \overline{T1+T7} = \overline{T1}\cdot\overline{T7}\\ \overline{R1} = \overline{Init + T2 + T5} = \overline{Init}\cdot\overline{T2}\cdot\overline{T5}\end{cases}\qquad St4:\begin{cases}\overline{S4} = \overline{T5 + T6} = \overline{T5}\cdot\overline{T6}\\ \overline{R4} = \overline{Init + T7 + T8} = \overline{Init}\cdot\overline{T7}\cdot\overline{T8}\end{cases}$$

$$St2:\begin{cases}\overline{S2} = \overline{T2}\\ \overline{R2} = \overline{Init + T3} = \overline{Init}\cdot\overline{T3}\end{cases}$$

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

M_Up = St1
L_Open = ST2
L_Stop = ST4

M_Dn = St3
L_Close = ST0
$L\_Air = St1 + St3 = \overline{\overline{St1}\cdot\overline{St3}}$

♦Pasul 5: Trasarea schemei logice a automatului

a) Logica de inițializare

În cazul de față, referitor la inițializare, așa după cum am mai precizat deja, procedăm astfel:

- intrarea *Init*, se conectează la intrarea de *Set* pentru starea **St0 = 1000**;
- intrarea de *Init* se conectează la toate intrările de *Reset* de la restul latch-urilor (latch-uri folosite pentru stările *St1, St2* și *St3*);

Deoarece intrarea de *Init* este activă pe unu logic, iar intrările de *Set* sau de *Reset* sunt active pe zero logic, este necesar un inversor conectat așa cum se vede în figura 3, de pe traseul desenat cu roșu.

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

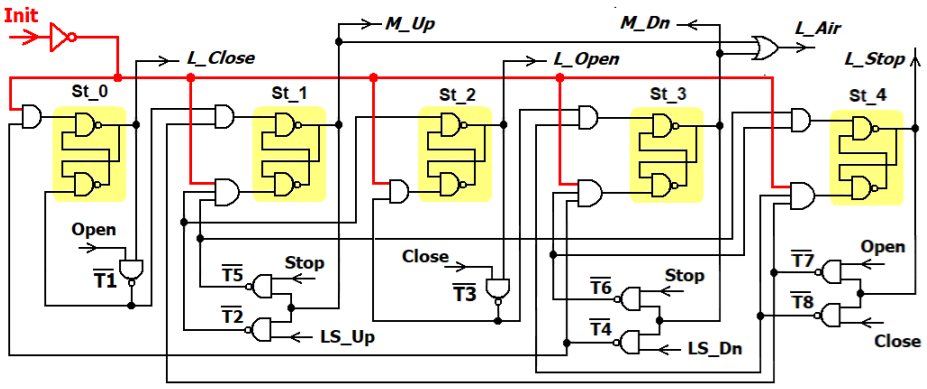


Fig. 10 Schema logică pentru sistemul de control realizată cu latch-uri (varianta 1)

În final facem precizarea că logica de inițializare și logica de control sunt realizate împreună, în mod asincron. Circuitele marcate în galben sunt latch-uri asincrone cu intrări active pe zero.

O altă variantă de implementare, folosind doar intrările asincrone de la bistabili de tip D, este ilustrată în figura 11.

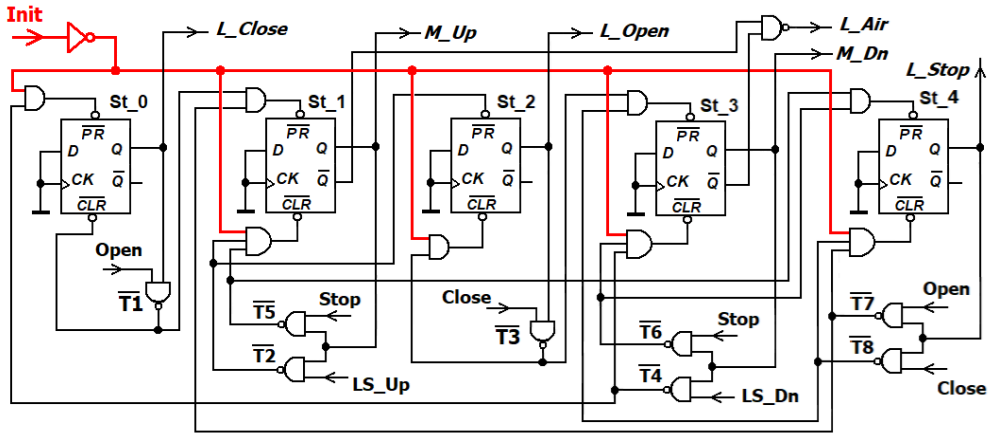


Fig. 11. Schema logică pentru sistemul de control realizată cu latch-uri (varianta 2)

În ultima schemă,

- elementul de memorie nu mai este un latch, implementat la nivel de poartă logică, ci un bistabil D folosit ca latch prin faptul că din resursele sale folosim doar intrările asincrone;
- ecuațiile pentru tranziții, pentru stări și pentru ieșiri sunt identice cu cele folosite la schema din figura 10
- O mică deosebire apare la ieșirea *L_Air*, unde, pe baza teoremei lui DeMorgan, poarta OR a fost înlocuită cu un NAND, prin faptul că bistabilii ne oferă și negatul stării.