2. Proiectarea şi realizarea practică a automatelor Richards

Obiectivele temei: Prezentarea etapelor ce trebuie parcurse în proiectrea şi implementarea sistemelor de control bazate pe automate FSM. Pentru partea de proiectare se are în vedere metodologia propusă de Charles Richards iar pe partea de implementare se are în vedere utilizarea circuitelor logice discrete de complexitate mică sau medie.

La o primă analiză, dacă avem în vedere amploarea utilizării microprocesoarelor în aplicaţiile de control, automatele prezentate în această secţiune par a fi depăşite (uzate moral). Cu toate acestea, automatele Richards revin în actualitate; ele sunt utilizate în proiectarea sistemele ce urmează a fi realizate cu circuite de tip CPLD sau FPGA. Explicaţia este destul de simplă: necesarul de resurse hardware pentru realizarea unui automat Richards este mult mai mic faţă de realizarea unui microprocesor. În consecinţă, pentru secvenţe simple de control, utilizarea automatelor Richards devine mai atractivă.

Competenţe obţinute:

- înţelegerea modului de funcţionare al automatelor Richards;
- cunoaşterea avantajelor/dezavantajelor diferitelor variante constructive de automate Richards;
- interpretarea corectă a diagramelor de tranziţie a stărilor specifice automatelor Richards;
- proiectarea unei scheme logice pornind de la diagrama de tranziţie a starilor;
- implementarea schemei logice cu circuite discrete de complexitate mică sau medie.

1. Breviar teoretic

1.1. Caracteristici generale

Este cunoscut faptul că automatele Mealy sau Moore înglobează programul de control (graful de tranziţie) la nivel hardware. Aceasta înseamnă că, o mică schimbare în programul de lucru al automatului, va avea ca efect modificarea semnificativă a schemei logice a automatului. Din acest motiv spunem că automatele Mealy sau Moore au o flexibilitate redusă. Pe de altă parte, proiectarea automatelor Mealy sau Moore nu poate fi făcută de către un tehnician cu pregătire medie, deoarece necesită cunoştinţe avansate de teoria circuitelor logice.

În anul 1973, în revista *Electronics*, Charles Richards publică un articol intitulat „*An easy way to design complex program controllers*”, în care prezintă o arhitectură de automat sincron ce prezintă următoarele caracteristici demne de remarcat:

- nucleul de bază este format din 3÷5 circuite logice de uz general (un numărător sincron, unul sau două MUX-uri, unul sau două DMUX-uri);
- nucleul de bază se menţine nemodificat chiar dacă se schimbă secvenţa de control, ceea ce înseamnă că poate fi folosit pentru o gamă largă de aplicaţii fără a fi necesară reproiectarea sa;
- în structura automatului, pe lângă nucleul de bază mai apare un bloc de calcul al adresei de salt, acesta este singurul bloc funcţional ce trebuie reproiectat/reconfigurat la modificarea aplicaţiei/secvenţei de control;
- automatul permite implementarea rapidă a secvenţelor lungi de control (zeci sau chiar sute de stări), cu grad mare de complexitate, cu număr mare de variabile de intrare/ieşire;
- proiectarea automatului şi eventualele modificări ale secvenţei de control pot fi realizate de către un tehnician cu pregătire medie, nefiind necesare parcurgerea etapelor specifice proiectării automatelor Mealy/Moore;

Aşadar, automatele Richards sunt mult mai flexibile, mai uşor de implementat şi mai apropiate de necesităţile practice de control, în raport cu automatele Mealy sau Moore.

Cu toate avantajele prezentate mai sus, automatele Richards nu rezolvă complet problema flexibilităţii. Adevărata rezolvare vine de la automatele de tip CROM, unde, programul de lucru este înscris într-o memorie ROM, iar partea hardware este nemodificată, în totalitate, indiferent de programul de lucru al automatului.

Automatele Richards au fost utilizate, destul de mult, până la apariţia şi utilizarea pe scară largă a microprocesoarelor, moment în care au căzut într-un con de umbră.

În prezent, automatele Richards revin în atenţia specialiştilor ce realizează aplicaţii cu circuite reconfigurabile de tip CPLD sau FPGA. Pentru astfel de aplicaţii, automatul Richards se poate folosi oriunde este nevoie de un “mic sistem de control” ce nu justifică utilizarea unui microprocesor. Avantajul vine din faptul că implementarea automatului Richards ocupă mult mai puţine resurse din CPLD/FPGA, în raport cu un microprocesor.

În final, facem precizarea că, *din punct de vedere al comportamentului intrare/ieşire, un automat Richards se comportă similar unui automat de tip Moore*. Diferenţele sunt date de modul de proiectare şi implementare.

1.2. Prezentarea automatului Richards – varianta I

Deşi are o utilitate practică mai restrînsă, cea mai simplă variantă de automat Richards, vezi schema bloc din figura 1, este prezentată pentru a face o trecere graduală spre variante mai complexe.

În schema bloc de principiu din figura 1 intervin trei circuite logice de complexitate medie:

- numărătorul binar (***N1***) – are ca rol memorarea stării automatului precum şi incrementarea stării automatului dacă anumite condiţii de intrare sunt îndeplinite;
- multiplexorul (***M1***) – pentru fiecare stare a automatului, acest circuit ne permite să alegem o variabilă de intrare, pentru a fi supusa testării. Starea logică a variabilei de intrare selectate decide dacă stare automatului se schimbă sau rămâne pe loc;
- demultiplexorul (***D1***) – permite iniţializarea unei comenzi distincte pentru fiecare stare a numărătorului/automatului.

Pe lângă resursele prezentate în figura 1, mai avem nevoie de un oscilator pentru generarea unui semnal digital periodic necesar pentru comanda intrării de ceas a numărătorului.

a) Funcţionare

Funcţionarea schemei bloc din figura 1 este similară pentru orice stare a numărătorului/automatului. În cele de urmează vom considera că numărătorul se află în starea $Q_2Q_1Q_0=000$, ca urmare a unei comenzi de **reset**.

Pentru această stare a automatului, multiplexorul ***M1*** primeşte pe intrările sale de selecţie codul binar ***000***. Ca urmare a acestui cod, circuitul ***M1*** permite trecerea variabilei de intrare X_0 spre ieşirea sa. Mai departe, de la ieşirea MUX-ului, variabila X_0 ajunge la intrarea de validare ***CE*** a numărătorului (sub formă directa) şi la intrarea de validare $\overline{E}$ a demultiplexorului (sub formă inversată – vezi inversorului ***P1***).

Prin urmare, în starea $Q_2Q_1Q_0=000$, întreaga funcţionare a automatului este decisă de starea logică pe care o are variabila de intrare X_0, după cum urmează:

- **pentru $X_0=0$**
  - intrarea ***CE*** *(Count Enable)* a numărătorului primeşte *'**0**'*, ceea ce are ca efect blocarea funcţiei de numărare. Aceasta înseamnă că starea numărătorului, şi în consecinţă şi a automatului, nu poate fi modificată atâta timp cât $X_0=0$, adică atâta timp cât condiţia testată este neadevarată, X_0 = ***FALS***.
  - circuitul ***D1*** primeşte un *'**1**'* pe intrarea de validare $\overline{E}$, ceea ce înseamnă că toate ieşirile acestuia vor fi în starea lor inactivă *('**1**')* . Cu alte cuvinte, nicio comandă nu poate fi activată.
- **pentru $X_0=1$**
  - numărătorul primeşte *'**1**'* pe intrarea de validare a numărării (***CE=1)***, ceea ce înseamnă că funcţia de numărare este permisă, starea numărătorului/automatului este incrementată cu o unitate la prima tranziţie activă de pe intrarea de ceas.
  - circuitul ***D1*** primeşte *'**0**'* pe intrarea de validare $\overline{E}$ (vezi inversorului P_1) lucru ce permite activarea ieşirii $\overline{0}$. Cu alte cuvinte este permisă activarea ieşirii principale $\overline{Y_0}$.

Investeste in OAMENI!

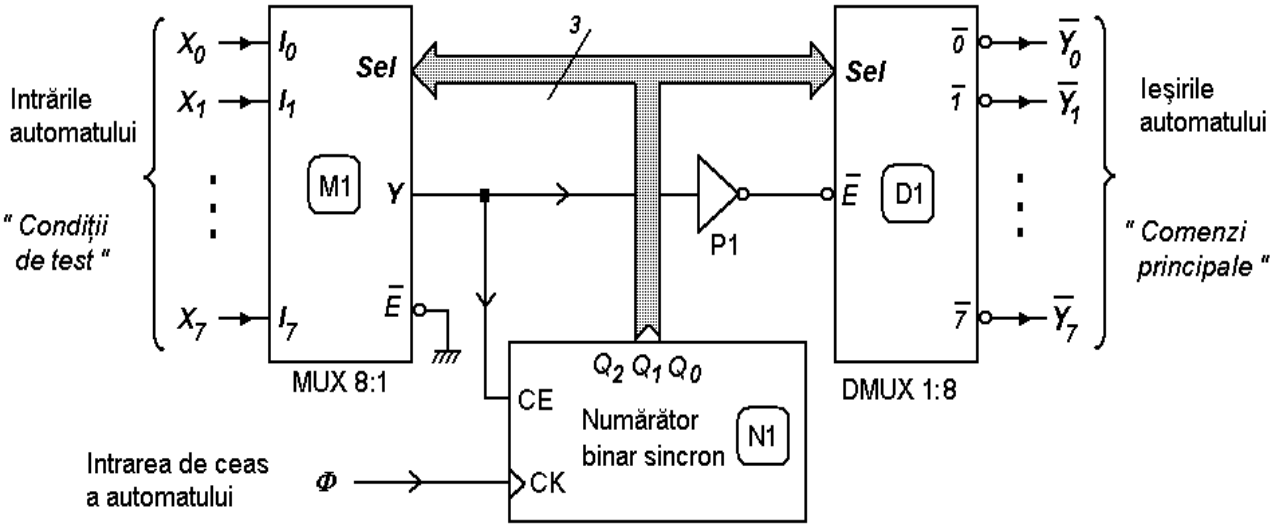

Fig. 1: *Schema bloc de principiu a unui automat Richards – varianta I*

După apariţia tranziţiei active pe intrarea de ceas a numărătorului, acesta trece din starea ***Q₂Q₁Q₀=000*** în starea ***Q₂Q₁Q₀=001***, ieşirea de comandă $\overline{Y_0}$ se dezactivează, iar multiplexorul selectează pentru testare variabila de intrare ***X₁***. În această nouă stare, funcţionarea automatului va fi decisă de variabila ***X₁***. Dacă ***X₁=0*** starea automatului nu poate fi schimbată iar dacă ***X₁=1*** se activează ieşirea $\overline{Y_1}$ după care, la prima tranziţie a semnalului de ceas, automatul trece în starea următoare.

Aşadar, pentru o stare oarecare a automatului, să spunem starea ***i***, este selectată variabila de intrare X_i , iar aceasta va avea un rol hotărâtor în funcţionarea automatului:

- dacă ***Xᵢ=0***, *ceea ce s-ar traduce prin:* ***"condiţie testată falsă"***, starea automatului nu poate fi modificată;
- dacă ***Xᵢ=1***, ***"condiţie testată adevărată"***, se activează ieşirea $\overline{Y_i}$ iar automatul poate avansa în spaţiul stărilor cu o unitate

Putem spune că automatul trece de la starea ***i***, la starea ***i+1***, numai dacă condiţia de test este adevărată (***Xᵢ =1)***. Dacă condiţia de test este falsă (***Xᵢ =0)***, automatul rămâne în starea ***i***, până când condiţia testată devine adevărată, adică până când apare ***Xᵢ =1***.

Referitor la funcţionarea automatului din figura 1, se pot face următoarele observaţii importante:

- Este absolut obligatoriu ca numărătorul să fie sincron, pentru ca funcţionarea sa să nu fie afectată de stările intermediare nedorite ce sunt specifice numărătoarelor asincrone. Mai clar, dacă s-ar utiliza un numărător asincron în schema din figura 1, există riscul ca schema să genereze comenzi neprevăzute, pe intervalele scurte de timp în care numărătorul asincron trece dintr-o stare în alta.

– Automatul iniţiază comenzi doar pentru situaţia în care *condiţia testată este adevărată* (variabila de intrare selectată de *M1* are valoarea '*1*').

  În unele situaţii practice poate fi util să avem posibilitatea de a genera comenzi şi pentru *condiţia testată=fals*. Pentru aceasta, la schema din figura 1 trebuie să adăugăm un DMUX suplimentar, conectat astfel încât să fie validat în antifază cu ***D1***. Modul de conectare al acestui DMUX, precum şi funcţionarea unui automat de acest fel, va fi explicat într-o secţiune viitoare;

– Ieşirile $\overline{Y_0} \div \overline{Y_7}$, denumite şi comenzile principale, sunt active pe zero logic, iar durata de activarea este cel mult egală cu o perioada a semnalului de ceas Φ.

– Din punct de vedere practic, comenzile $\overline{Y_0} \div \overline{Y_7}$ sunt folosite pentru iniţializarea, respectiv terminarea unei acţiuni (spre exemplu start motor, stop motor). Ele nu rămân active între momentele de start şi de stop, deci nu pot fi folosite pentru comanda propriu-zisa a unui element de execuţie. În consecinţă, vom avea nevoie de elemente de memorie (bistabili) care să memoreze comanda de start şi să o menţină activă până la o comandă de stop. Aceste elemente de memorie nu sunt trecute în figura 1.

– Pentru orice stare a automatului nu se poate testa decât o singură variabilă de intrare. Dacă pentru starea ***i***, trebuie testate mai multe variabile de intrare, atunci, la intrarea $\boldsymbol{X_i}$, trebuie amplasat un CLC care va genera *'1'* doar atunci când variabilele de intrare au configuraţia dorită.

– Automatul din figura 1 nu permite întreruperea secvenţei de numărare, aceasta înseamnă că nu acceptă diagrame de funcţionare în care apar salturi de la o stare la alta - total diferită în ceea ce priveşte codificarea sa. Acceptarea salturilor presupune modificarea schemei din figura 1, aşa cum se arată în secţiunea următoare.

b) Diagrama de tranziţie a stărilor

Pentru explicarea funcţionării acestor automate, inventatorul lor propune utilizarea unei diagrame de tranziţie a stărilor formată din două elemente :

- blocuri de test (simbolizate prin romburi sau triunghiuri) ;
- blocuri de tip funcţie (simbolizate prin dreptunghiuri).

În blocurile de test se trece starea automatului (numărul stării) şi numele variabilei de intrare ce trebuie testate. În blocurile de tip funcţie se trece doar numele ieşirii (comenzii principale) ce trebuie activată. Un exemplu de diagramă de tranziţie a stărilor, pentru automatul din figura 1, este prezentată în figura 2.

Diagrama din figura 2 explică foarte clar funcţionarea automatului: trecerea de la o stare la alta este permisă doar atunci când variabila selectată în procesul de testare este adevărată (are valoarea '1').

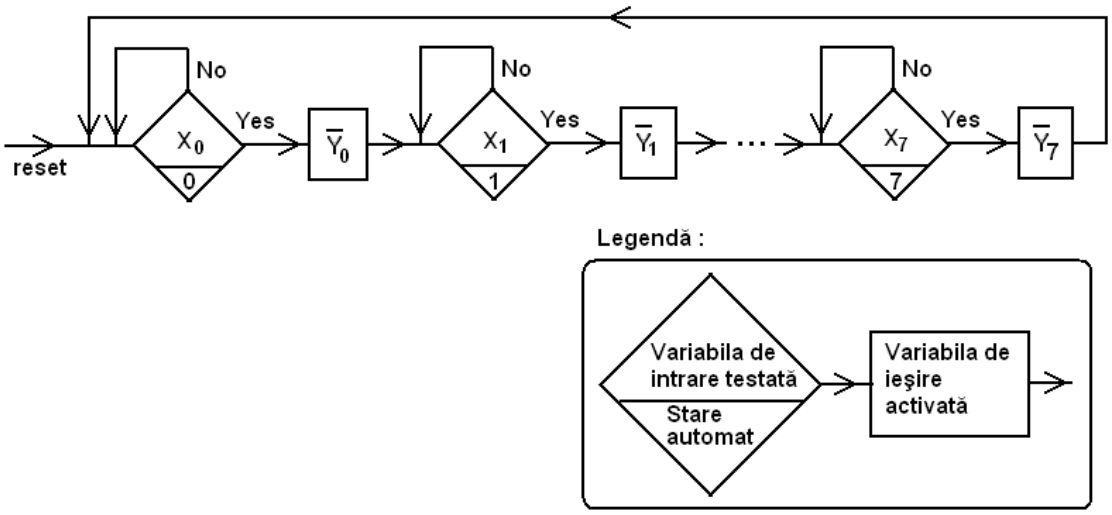

Fig. 2: *Diagrama de tranziţie a stărilor pentru automatul Richards din figura 1.*

c) Ecuaţii logice

Funcţionarea automatului poate fi descrisă şi prin ecuaţii logice de forma:

$Y_0 = (\text{stare } 0)(X_0)\,\Delta$

...

$Y_7 = (\text{stare } 7)(X_7)\,\Delta$

Aceste ecuaţii ne spun că ieşirea $\boldsymbol{Y_i}$ se activează numai dacă starea automatului/numărătorului este (starea ***i***) iar variabila $\boldsymbol{X_i=1}$. Simbolul ***Δ*** este folosit pentru a indica faptul că urmează incrementarea stării numărătorului.

2.3. Prezentarea automatului Richards – varianta II

Un automat mult mai util din punct de vedere practic este acela care execută o comandă pentru cazul în care *variabila testata este adevărată* şi o altă comandă pentru cazul în care *variabila testată este falsă*. În plus, trebuie să existe şi posibilitatea de a executa salturi de la o stare la alta, adică de a întrerupe secvenţa de numărare în funcţie de stările logice găsite pe intrările $\boldsymbol{X_i}$.

Schema bloc de principiu pentru un automat care satisface cerinţele exprimate mai sus este prezentata în figura 3. În această schemă regăsim componentele folosite în versiunea anterioară de automat, la care se mai adaugă demultiplexorul ***D2*** şi un circuit combinaţional responsabil de calculul adresei de salt.

Investeste in OAMENI!

Trebuie remarcat şi faptul că numărătorul ***N1*** trebuie să fie presetabil – cerinţă ce nu era strict necesară pentru schema bloc din figura 1.

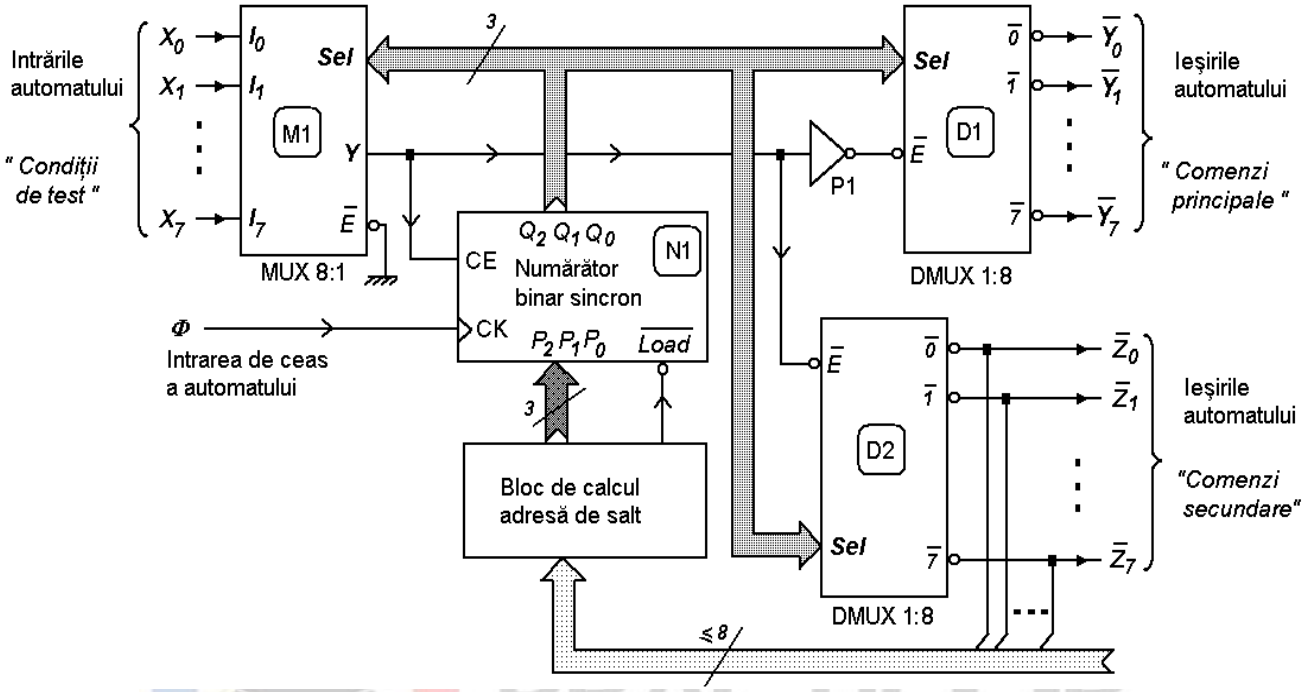

Fig. 3: *Schema bloc de principiu a unui automat Richards – Varianta II*

Rolul componentelor din figura 3 este prezentat succint în cele ce urmează:

- numărătorul binar presetabil (***N1***) – are ca rol: memorarea stării automatului, incrementarea stării automatului sau realizarea salturilor dacă anumite condiţii de intrare sunt îndeplinite;
- multiplexorul (***M1***) – selectează variabila de intrare ce trebuie testată în fiecare stare a automatului;
- demultiplexorul (***D1***) – permite iniţializarea *comenzilor principale*, numai în situaţiile în care variabila de intrare selectată de ***M1*** este adevărată;
- demultiplexorul (***D2***) – permite iniţializarea *comenzilor secundare*, numai în situaţiile în care variabila de intrare selectată de ***M1*** este falsă;
- blocul de calcul al adresei de salt – este folosit pentru calculul adresei de salt şi pentru comanda intrării $\overline{Load}$ a numărătorului (intrarea de comandă a încărcării paralele).

a) Funcţionare

Ca şi în cazul anterior, vom considera că numărătorul se află în starea $Q_2Q_1Q_0=000$, ca urmare a unei comenzi de **reset.** Pentru această stare, multiplexorul ***M1*** selectează variabila de intrare X_0 şi o conectează:

- în mod direct la intrare ***CE*** a numărătorului ***N1*** precum şi la intrarea de validare $\overline{E}$ a demultiplexorului secundar ***D2***;
- sub formă inversată (vezi inversorul P1) la intrarea de validare $\overline{E}$ a demultiplexorului principal ***D1***.

În urma acestor conexiuni, întreaga funcţionare a automatului este decisă de starea logică pe care o are variabila de intrare X_0, după cum urmează:

➢ **Pentru $X_0=0$**, elementele din schemă funcţionează după cum urmează:

- incrementarea numărătorului ***N1*** nu este posibilă deoarece intrarea ***CE*** (de validare a numărării) primeşte o valoare de ***0***, în condiţiile în care intrarea este activă pe 1. Starea automatului rămâne nemodificată atâta vreme cât $X_0=0$. Ieşirea din această stare se poate face dacă X_0 trece în ***1*** sau dacă se acţionează asupra intrării de reset;
- ieşirile circuitului ***D1*** nu pot fi activate deoarece intrarea sa de validare, $\overline{E}$, activă pe zero logic, primeşte unu logic din cauza inversorului P_1. In consecinţă, nicio ieşire principală nu poate fi activată;
- demultiplexorul ***D2*** este validat deoarece valoarea de zero logic de pe intrarea X_0 ajunge, via ***M1***, la intrarea de validare a lui D_2. Aceasta are ca efect activarea ieşirii secundare $\overline{Z_0}$;

  Faptul că intrările de validare ale celor două circuite DMUX se face în contratimp ne dă garanţia că, niciodată, nu se vor activa simultan cele două categorii de ieşiri. Richards propune denumirea de ***comenzi principale*** pentru cele generate pe *condiţia testată=adevărat*, respectiv ***comenzi secundare*** pentru cele generate pe *condiţia testată= fals*.

  Aşadar, ***D1*** este folosit pentru generarea comenzilor principale iar ***D2*** pentru generarea de comenzi secundare.

- pentru *condiţia testată= fals* există posibilitatea executării unui salt în diagrama de tranziţie a stărilor, acţionând asupra intrării de încărcare paralela a numărătorului. Generarea comenzii de salt precum şi a adresei de salt (starea în care trebuie să ajungă automatul după execuţia saltului) este responsabilitatea **blocului de calcul al adresei de salt**.

  Atenţie ! Saltul este permis doar pentru condiţia testată=fals şi niciodată pentru condiţia testată=adevărat.

  Execuţia unui salt este o opţiune, nu o obligaţie. Aceasta înseamnă că nu este obligatoriu să avem salturi pentru orice situaţie în care condiţia testată este falsă.

Execuţia saltului se poate face numai dacă numărătorul are facilitatea de încărcare paralelă.

- **Pentru** ***X0=1***, elementele din schemă funcţionează după cum urmează:
  - incrementarea numărătorului este permisă deoarece intrarea ***CE***, activă pe unu logic, primeşte chiar unu logic. Starea automatului va fi incrementată la prima tranziţie activă de pe intrarea de ceas Φ.
  - circuitul ***D1*** este validat, aceasta înseamnă că este permisă generarea de comenzi principale. În exemplul de faţă se activează ieşirea $\overline{Y_0}$, cu observaţia că durata de activare este cel mult egală cu perioada semnalului de ceas.
  - circuitul ***D2*** nu este validat (primeşte unu logic pe intrarea de validare $\overline{E}$), aceasta înseamnă că nu este permisă validarea niciunei comenzi secundare;
  - pentru această situaţie, sau altele similare, blocul de calcul al adresei de salt nu trebuie să permită activarea unei comenzi de încărcare paralelă a numărătorului;

Schema logică a blocului de calcul al adresei de salt este puternic dependentă de diagrama de tranziţie a stărilor şi, cel mai adesea, cuprinde câteva porţi NAND cu două sau trei intrări. Practic, acest bloc funcţional este cel care particularizează funcţionarea automatului, el trebuie reproiectat dacă se fac modificări în programul de lucru al automatului.

2. Activităţi de proiectare

O implementare cu circuite logice comerciale a schemei de principiu din figura 3 se obţine, destul de rapid, cu ajutorul unor circuite uzuale de complexitate mică şi medie.

Singurul bloc funcţional care necesită un mic efort de proiectare este blocul de calcul al adresei de salt. De altfel, acesta este singurul care se modifică în situaţia în care trebuie operate schimbări în programul de lucru al automatului.

În cele ce urmează prezentăm modul de implementare a schemei logice a unui automat având ca punct de plecare schema bloc din figura 3 şi diagrama de tranziţie a stărilor din figura 4. În aceste condiţii, o posibilă schemă finală este prezentată în figura 6. În realizarea acestei scheme s-a ţinut cont de următoarele aspecte:

- Multiplexorul ***M1***, de tip 8:1, este implementat cu ajutorul circuitului 74151. La acest circuit ieşirea multiplexorului este disponibilă sub formă directă (Y la pinul 5), respectiv sub formă negată ($\overline{Y}$ la pinul 6). Această facilitate este echivalentă cu integrarea inversorului ***P1*** în capsula circuitului integrat de tip 74151. Din acest motiv, în schema finală din figura 6, inversorul ***P1*** nu mai poate fi pus în evidenţă, aşa cum este în cazul schemei bloc.

Investeste in OAMENI!

- Demultiplexoarele ***D1*** şi ***D2*** sunt implementate cu decodificatoare zecimale de tip 7442. Utilizarea unui decodificator zecimal ca DMUX 1:8 este posibilă dacă: folosim intrările *CBA* ca intrări de selecţie, intrarea *D* ca intrare de validare iar ieşirile $\overline{8}$ şi $\overline{9}$ rămân nefolosite.
- Numărătorul ***N1*** este implementat cu un numărător sincron, binar, presetabil, pe 4 biţi, de uz comercial, de tip *74161*. În utilizarea acestui circuit trebuie să ţinem cont că are 4 biţi (faţă de un necesar de 3 biţi), iar denumirea intrărilor/ieşirilor este puţin diferită faţă de cea a numărătorului generic din figura 3. Astfel:
  - Ieşirile ce arată starea numărătorului *74161* sunt notate cu $Q_DQ_CQ_BQ_A$, în loc de $Q_3Q_2Q_1Q_0$.
  - Dintre cele 4 ieşiri se folosesc doar primele trei ieşiri (Q_2 Q_1 Q_0). Aceasta înseamnă că numărul maxim de stări distincte ale automatului este egal cu 8.
  - Intrările folosite la încărcarea paralelă sunt notate cu ***DCBA***, în loc de $P_3P_2P_1P_0$.
  - Intrările de date folosite în mod efectiv la încărcarea paralelă a numărătorului sunt primele trei, ***CBA***. Intrarea *D* este legată la masă deoarece nu este folosită niciodată.
  - Intrarea de validare a numărării ***CE*** se obţine prin legarea în paralel a intrărilor ***ENT*** şi ***ENP***.
  - Ieşirea de semnalizare a umplerii numărătorului, ***RCO***, nu este folosită niciodată
  - Intrarea de ştergere a numărătorului este comandată de către un comutator cu revenire de tip normal deschis.
- Blocul de calcul al adresei de salt este alcătuit din porţi NAND şi o poartă AND. Numărul porţilor, numărul de intrări şi modul de conectare este puternic dependent de programul de lucru al automatului.

Pentru a demonstra uşurinţa cu care se poate implementa o diagramă de tranziţie a stărilor, vom considera diagrama de tranziţie din figura 4.

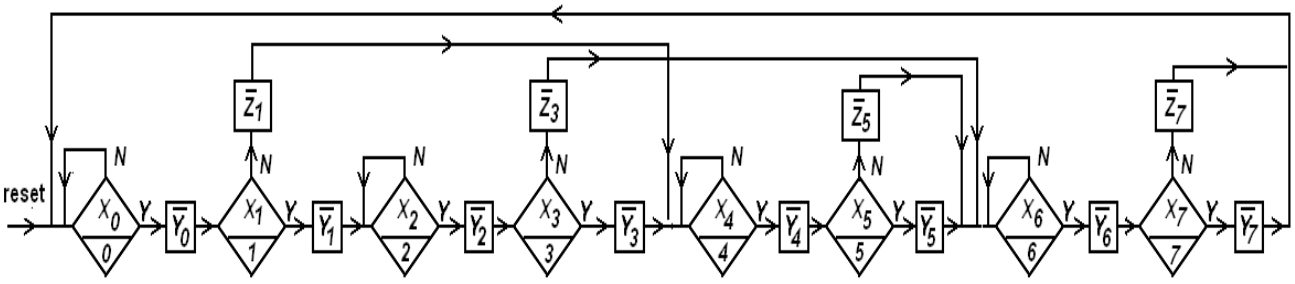

Fig. 4: *Diagramă de tranziţie a stărilor (pentru automatul Richards, varianta II)*

Din analiza diagramei se observă că salturile sunt efectuate numai din stările 1, 3, 5 şi 7 şi numai pentru condiţia testată falsă. Spre exemplu, în starea 1, automatul testează variabila de intrare notată cu X_1. Programul de lucru va merge pe o ramură, sau pe alta, în funcţie de rezultatul testului. Dacă $X_1=0$ se execută

funcţia $\overline{Z_1}$ (comanda secundară cu numărul 1) iar apoi se face un salt la starea 4. Dacă ***X1=1*** se execută funcţia $\overline{Y_1}$ (comanda principală cu numărul 1) iar apoi se trece, prin incrementare, la starea următoare, starea cu numărul 2.

Pentru proiectarea blocului de calcul al adreselor de salt, din diagrama anterioară se extrag următoarele relaţii:

$\overline{Z_1}$ =0 şi X1=0 → salt la starea 4 → CBA=100 şi $\overline{LD}$ =0

$\overline{Z_3}$ =0 şi X3=0 → salt la starea 6 → CBA =110 şi $\overline{LD}$ =0

$\overline{Z_5}$ =0 şi X5=0 → salt la starea 6 → CBA =110 şi $\overline{LD}$ =0

$\overline{Z_7}$ =0 şi X7=0 → salt la starea 0 → CBA =000 şi $\overline{LD}$ =0

Alte situaţii → saltul este interzis sau nu este necesar → $\overline{LD}$ =1

Relaţiile anterioare ne conduc la următorul tabel de adevăr:

Tabelul 4.1

Stare automat	***Xi***	**Ieşirile demultiplexorului D2** $\overline{Z_7}$ $\overline{Z_6}$ $\overline{Z_5}$ $\overline{Z_4}$ $\overline{Z_3}$ $\overline{Z_2}$ $\overline{Z_1}$ $\overline{Z_0}$	***C B A***	$\overline{LD}$	**Observaţii**
0 0 0	0	1 1 1 1 1 1 1 0	* * *	1	Nu se face salt
0 0 1	0	1 1 1 1 1 1 0 1	1 0 0	0	**Salt în starea 4**
0 1 0	0	1 1 1 1 1 0 1 1	* * *	1	Nu se face salt
0 1 1	0	1 1 1 1 0 1 1 1	1 1 0	0	**Salt în starea 6**
1 0 0	0	1 1 1 0 1 1 1 1	* * *	1	Nu se face salt
1 0 1	0	1 1 0 1 1 1 1 1	1 1 0	0	**Salt în starea 6**
1 1 0	0	1 0 1 1 1 1 1 1	* * *	1	Nu se face salt
1 1 1	0	0 1 1 1 1 1 1 1	0 0 0	0	**Salt în starea 0**
* * *	1	0 1 1 1 1 1 1 1	* * *	1	Saltul nu este permis

Investeste in OAMENI!

În construcţia tabelului anterior am ţinut cont de mai multe aspecte ce sunt impuse de funcţionarea schemei bloc:

- ieşirile $\overline{Z_i}$ aparţin unui DMUX, de aici tragem concluzia că, niciodată, nu vom găsi două sau mai multe ieşiri activate simultan;
- pentru calculul expresiilor logice ale ieşirilor ***CBA*** respectiv $\overline{LD}$, nu mai este necesar să luăm în calcul stările automatului şi nici variabilele de intrare (toate acestea au intervenit deja, fie în adresarea, fie în validarea funcţionării circuitului ***D2***);
- comanda de încărcare paralelă a numărătorului trebuie să fie activă doar acolo unde se doreşte realizarea unui salt ($\overline{LD}$=0). În astfel de cazuri starea logică de la ieşirile ***CBA*** trebuie să fie egală cu scrierea binară a stări în care trebuie să ajungă automatul în urma execuţiei saltului.
- comanda de încărcare paralelă a numărătorului trebuie să fie inactivă acolo unde nu este necesar un salt ($\overline{LD}$=1). În astfel de cazuri starea logică de la ieşirile ***CBA*** nu este relevantă (poate avea orice valoare) deoarece această informaţie nu este preluată de numărător.
- ultima linie din tabelul de adevăr corespunde situaţiilor în care variabilele de intrare sunt adevărate. În astfel de cazuri comanda de încărcare paralelă a numărătorului trebuie să fie obligatoriu inactivă ($\overline{LD}$=1) iar informaţia de pe ieşirile ***CBA*** nu are relevantă.

Pe baza tabelului de adevăr anterior, se poate arăta că ieşirile folosite pentru comanda intrărilor de încărcare paralelă sunt date de expresiile:

$$\boldsymbol{C}=\overline{Z_1}+\overline{Z_3}+\overline{Z_5}=\overline{Z_1\cdot Z_3\cdot Z_5}$$

$$\boldsymbol{B}=\overline{Z_3}+\overline{Z_5}=\overline{Z_3\cdot Z_5}$$

$$\boldsymbol{A}=0$$

iar pentru comanda intrării de încărcare paralelă se pot scrie relaţiile:

$$LD=\overline{Z_1}+\overline{Z_3}+\overline{Z_5}+\overline{Z_7}=\overline{Z_1\cdot Z_3\cdot Z_5\cdot Z_7}$$

$$\overline{LD}=\overline{\overline{Z_1\cdot Z_3\cdot Z_5\cdot Z_7}}=Z_1\cdot Z_3\cdot Z_5\cdot Z_7$$

$$\overline{LD}=Z_1\cdot Z_3\cdot Z_5\cdot Z_7$$

Două variante de implementare a blocului de calcul a adresei de salt, deduse din relaţiile de mai sus, sunt prezentate în figura 5.

Investeste in OAMENI!

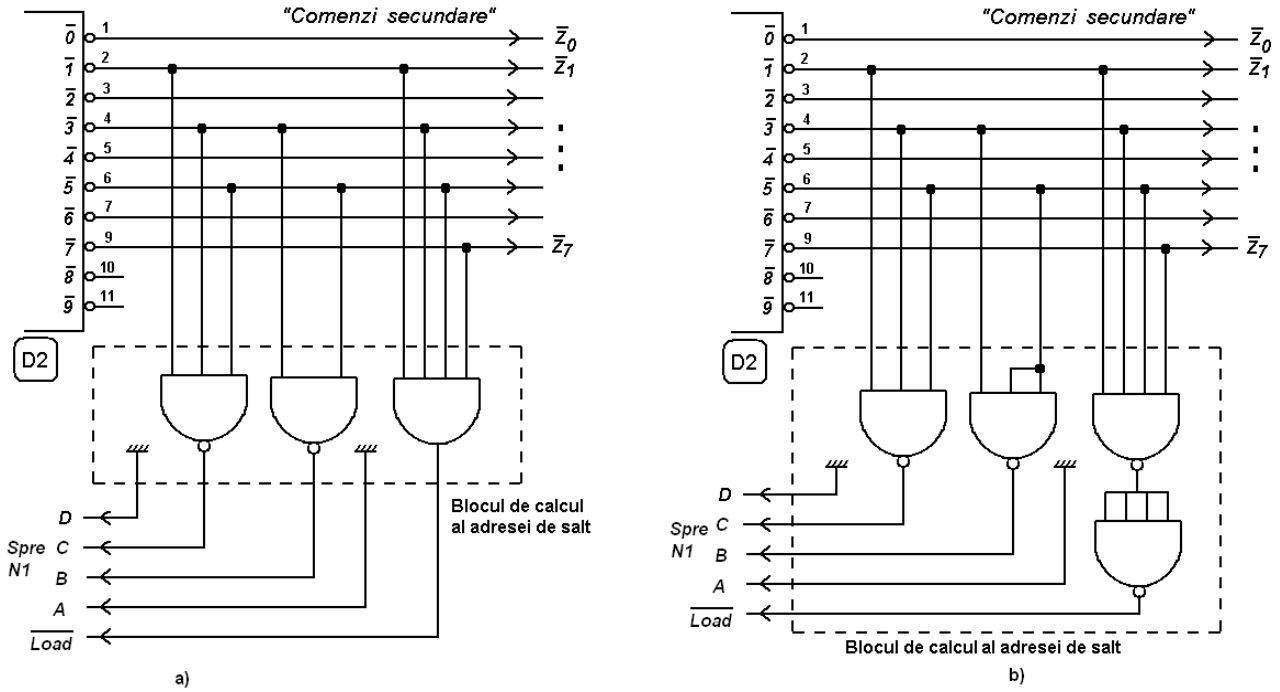

Fig. 5: *Variante de implementare a blocului de calcul al adresei de salt pentru diagrama de tranziţie a stărilor din figura 4*

Schemele din figura 5 sunt echivalente din punct de vedere logic, dar, din punct de vedere al realizării practice, ele diferă: schema din figura ***a*** necesită trei circuite integrate (câte un circuit integrat pentru fiecare tip de poartă) iar schema din figura ***b*** necesita doar doua circuite integrate (unul conţine porţile NAND cu 4 intrări iar celălalt porţile NAND cu 3 intrări). În consecinţă schema din figura 5b este mai avantajoasă din punct de vedere al costurilor.

Schema electrică finală a automatului Richards este prezentată în figura 6. Pentru această schemă avem un necesar de 6 circuite integrate.

Trebuie precizat că schema din figura 6 conţine două zone:

a) o zonă fixă ce nu se schimbă chiar dacă apar modificări în programul de lucru al automatului, formată din circuitele ***M1***, ***N1***, ***D1*** şi ***D2***;
b) o zonă dependentă de aplicaţie – blocul de calcul al adresei de salt, formată din porţile logice încadrate prin linie întreruptă.

Aşadar, schimbarea programului de lucru sau utilizarea automatului într-o altă aplicaţie de control presupune reproiectarea blocului de calcul al adresei de salt (zona punctată). Reducerea efortului de proiectare se poate face ţinând cont de următoarele reguli generale de conectare a porţilor din zona amintită mai sus.

Investeste in OAMENI!

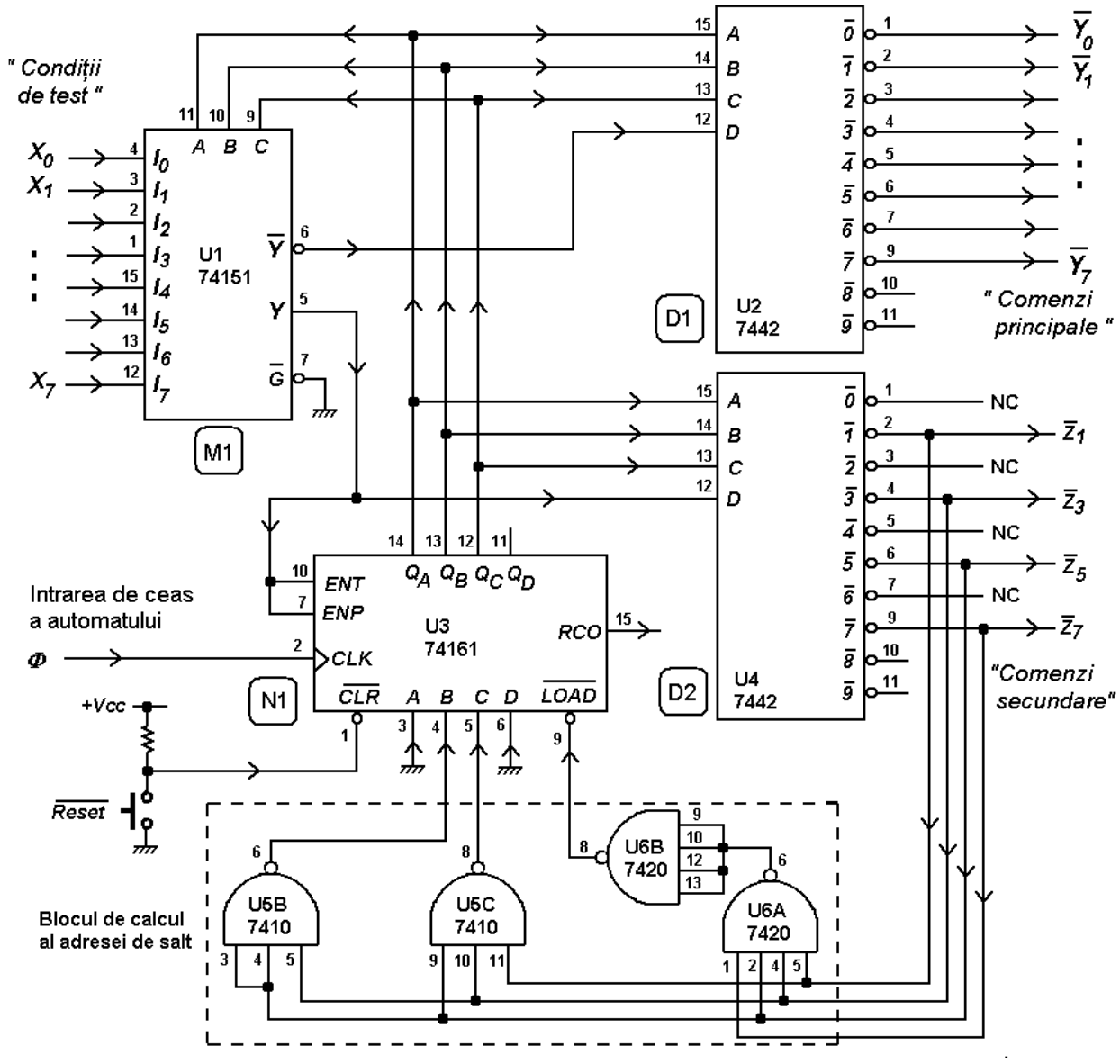

Fig. 6: *Schema electrică a automatului Richards ce lucrează după digrama de tranziţie a stărilor din figura 4.*

Observaţii:

Din exemplul anterior se pot deduce următoarele reguli generale ce pot fi utilizate în proiectarea blocului de calcul al adresei de salt fără a mai fi necesar să scriem nicio relaţie. Aceste reguli sunt prezentate şi exemplificate pentru cazul anterior în cele ce urmează.

- Pentru fiecare intrare de încărcare paralelă a numărătorului avem nevoie de cel mult o poarta NAND.
- Pentru fiecare poartă NAND, numărul de intrări este egal cu numărul valorilor de unu logic din tabelul de tranziţie al automatului.

*În exemplul anterior se remarcă faptul că poarta NAND care comandă intrarea **C** trebuie să aibă 3 intrări deoarece, în tabelul anterior, pe coloana lui **C** găsim trei valori de unu logic. În mod similar deducem că pentru intrarea **B** avem nevoie de o poartă cu 2 intrări iar pentru **A** nu avem nevoie de nicio poartă.*

- Intrările porţilor NAND se leagă la toate ieşirile $\overline{Z_i}$ ce sunt active în dreptul valorilor de unu logic.

  *În exemplul anterior, cele trei intrări ale porţii trebuie conectate la $\overline{Z_1}$, $\overline{Z_3}$ şi $\overline{Z_5}$ deoarece C=1 atunci când se activează $\overline{Z_1}$, $\overline{Z_3}$ respectiv $\overline{Z_5}$. În mod similar intrările porţii ce comandă intrarea **C** trebuie conectate la $\overline{Z_3}$ şi $\overline{Z_5}$.*

- Pentru intrarea de încărcare paralelă avem nevoie de o poartă AND ce trebuie să aibă câte o intrare pentru fiecare stare din care se face salt.
  Pentru exemplul anterior, se execută salturi din stările 1, 3, 5 şi 7, motiv pentru care avem nevoie de un AND cu 4 intrări conectate la $\overline{Z_1}$, $\overline{Z_3}$, $\overline{Z_5}$ şi $\overline{Z_7}$.

Aceste reguli simple ne permit să proiectăm rapid blocul de calcul al adresei de salt – singurul bloc funcţional ce trebuie reproiectat la schimbarea programului de lucru al automatului.

Bibliografie

1. Charles L. Richards, *„An easy way to design complex program controllers”*, Electronics, vol. 46, no.3, Feb 1, 1973, p 107-113