AUTOMATE PROGRAMABILE ALGORITMICE

1. INTRODUCERE

Un circuit de control a unei aplicaţii , realizat din circuite interconectate şi numit *automat* , în urma testării unor semnale logice aplicate la intrările sale , emite semnale logice de comandă la terminalele de ieşire . Semnalele logice de la intrările şi ieşirile unui automat reprezentând informaţii binare , rezultă că un automat realizează funcţia de control prin procesare de informaţie .

Definiţie

Automatul ale cărui semnale logice de ieşire depind în orice moment doar de semnalele logice de intrare se numeşte *automat combinaţional* sau *circuit logic combinaţional* .

Un automat combinaţional materializează prin fiecare ieşire

$y_j \qquad j = 1 , 2 , \dots , p$

câte o funcţie logică

$h_j : X \rightarrow \{0 , 1\}$

care pentru o combinaţie logică

$x = x(x_1 , x_2 , \dots , x_k) \; \varepsilon \; x = \underbrace{\{0,1\}x\dots.x\{0,1\}}_{\text{de k ori}}$

aplicată la cele k intrări ale sale , asociază o stare logică de ieşire

$y_j = h_j(x) \; \varepsilon \; \{0,1\}$

Reunind funcţiile logice

$h_j : X \rightarrow \{0,1\} \quad j = 1,2,\dots.,p$

ale celor p ieşiri ale automatului combinaţional , într-o funcţie logică vectorială

$h = \{h_1 , h_2 , \dots , h_p\} : X \rightarrow Y$

ce asociază unei combinaţii logice

$x = (x_1 , x_2 , \dots , x_k) \; \varepsilon \; X = \underbrace{\{0,1\}x\dots.x\{0,1\}}_{\text{de k ori}}$

aplicată la cele k intrări , o combinaţie logică

$y = (y_1 , y_2 , \dots , y_p) = (h_1(x) , h_2(x) , \dots , h_p(x)) \; \varepsilon \; Y = \underbrace{\{0,1\}x\dots.x\{0,1\}}_{\text{de p ori}}$

generată la cele p ieşiri, un automat combinaţional are din punct de vedere matematic următoarea :

Definiţie

Se numeşte *automat combinţional* gruparea A = (X , Y , h) , în care X = mulţimea combinaţiilor logice de intrare , numită *alfabet de intrare* , Y = mulţimea combinaţiilor logice de ieşire , numită *alfabet de ieşire* , iar funcţia logică :

$h : X \rightarrow Y$

este numită *funcţie de ieşire* .

Un automat combinaţional este simbolizat ca în figura 1 .

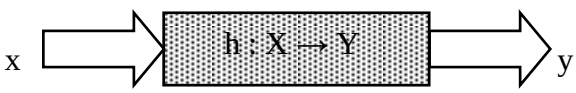


Fig.1

Definiţie

Un automat ce determină efectuarea unei sarcini prin executarea de operaţii într-o secţiune temporală de etape , adică în mod secvenţial , se numeşte *automat secvenţial* .

Un automat secvenţial poate reacţiona diferit la o aceeaşi combinaţie logică de intrare , în funcţie de nivelele logice memorate la ieşirile unor circuite componente , fiecare combinaţie între ieşirile acestor circuite cu memorare caracterizând deci o *stare internă* a automatului .

În funcţie de starea internă în care se găseşte şi de combinaţia logică de pe intrările sale , un automat secvenţial trece într-o nouă stare internă şi generează pe ieşirile sale o combinaţie logică necesară controlului aplicaţiei .

Deci , în orice moment *starea* unui automat secvenţial este caracterizată atât prin starea sa internă cât şi prin combinaţia logică emisă pe ieşirile sale .

Definitii

1). Trecerea unui automat într-o altă stare se numeşte *tranziţie* .

2). Durata dintre două tranziţii succesive se numeşte *secvenţă* .

Starea unui automat secvenţial nu se modifică într-o secvenţă . Într-o tranziţie , identificarea stării interne în care se trece poartă numele de *secvenţiere* , iar determinarea combinaţiei logice de ieşire se numeşte *comandă* .

Tranziţiile prin care se realizează evoluţia unui automat secvenţial printr-un şir de stări , pot avea loc fie la momente de timp oarecare când se spune că automatul secvenţial este *asincron* , fie la momente de timp precis determinate cu impulsurile unui generator de tact , când se spune că automatul secvenţial este *sincron* .

În practică se preferă automatele sincrone întrucât au funcţionarea precis controlată .

Într-un automat secvenţial A , notând X = mulţimea combinaţiilor logice de intrare , Y = =mulţimea combinaţiilor logice de ieşire şi Z = mulţimea stărilor interne , secvenţierea reprezintă o funcţie logică de tranziţie

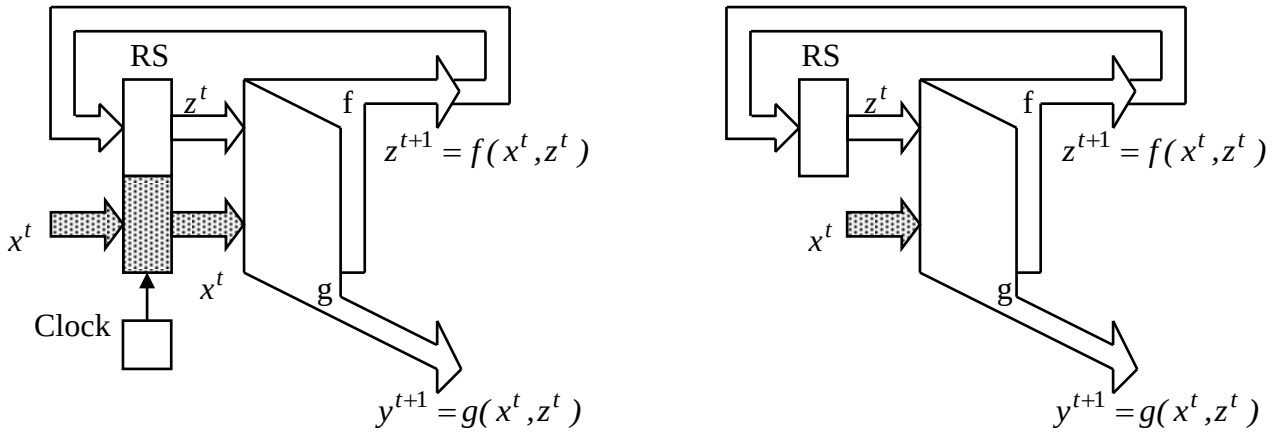
f : X x Z -> Z ,

iar comanda o funcţie logică de ieşire

g : X x Z -> Y .

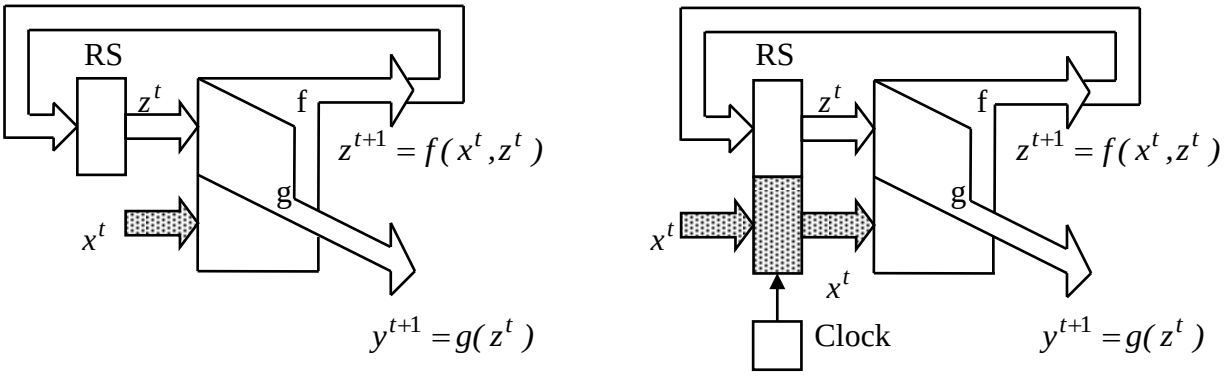
Din cele prezentate , rezultă că un automat secvenţial conţine câte un automat combinaţional pentru materializarea funcţiilor logice de tranziţie şi de ieşire precum şi elemente cu memorare pentru menţinerea stării interne curente ce pot fi realizate spre exemplu sub forma unui registru de stare .

După modul de definire a funcţiilor de tranziţie şi de ieşire , s-au determinat două structuri posibile de realizare a automatelor secvenţiale , ce se clasifica astfel în : *automate Mealy* (figura 2) şi *automate Moore* (figura 3) .



a) Mealy sincron b) Mealy asincron

Fig. 2 – Structura automatului Mealy



a) Moore asincron b) Moore sincron

Fig. 3 – Structura automatului Moore

Din punct de vedere matematic automatele secvenţiale Mealy şi Moore au următoarele :

Definiţii

1. Se numeşte automat *Mealy* gruparea A = (X,Z,Y,f,g) în care X = mulţimea combinaţiilor logice de intrare numită *alfabet de intrare* , Z = mulţimea stărilor interne , Y = mulţimea combinaţiilor logice de ieşire numită *alfabet de ieşire* , secvenţierea se realizează cu funcţia :

f : X x Z -> Z ,

iar comanda , cu funcţia de ieşire :

g : X x Z -> Y

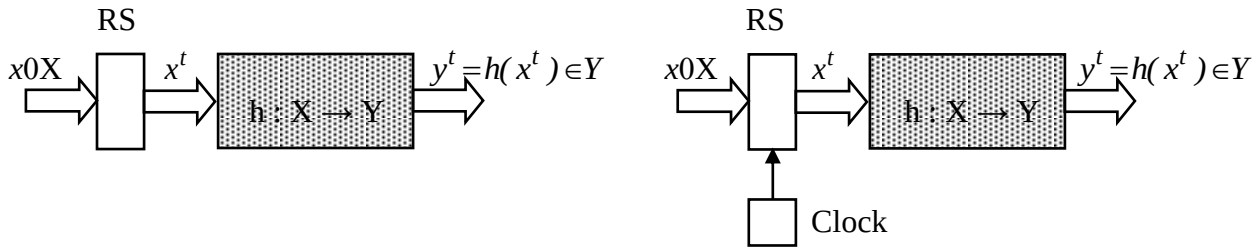
2. Automatul Mealy A = (X,Z,Y,f,g) în care :

g : Z -> Y

se numeşte automat *Moore* .

Rezultă că atât automatul Moore cât şi automatul combinaţional reprezintă cazuri particulare ale automatului Mealy .

Pentru un automat combinaţional , modificând doar la anumite momente de timp combinaţia logică de la intrarea sa , se obţine o funcţionare secvenţială , în structura unui asemenea automat memorarea combinaţiei logice de intrare realizându-se în modul sincron ce ar putea fi un registru de stare (figura 4) .



a) automat asincron b) automat sincron

Fig. 4 – Automat secvenţial cu un automat combinaţional

Într-un automat secvenţial A = (X,Z,Y,f,g) funcţiile logice , de tranziţie f şi de ieşire g , pot fi materializate , fie în logică cablată , fie prin circuite cu memorare , programabile .

O logică cablată poate fi asimilată unui circuit cu memorare complicat , greu de realizat , neprogramabile şi cu timp mare de propagare .

Definiţie

Automatul secvenţial în care atât secvenţierea cât şi comanda sunt programate în circuite cu memorare se numeşte *Automat Programabil Algoritmic* (APA) .

Un APA are o viteză mare de operare limitată doar de timpul de acces în circuitele cu memorare .

O stare a unui APA se atinge prin localizarea sa în circuitul cu memorare unde este descrisă printr-un câmp de biţi al semnalelor logice de comandă şi un alt câmp de biţi reprezentând codul stării interne curente necesar în identificarea următoarei stări interne în care va tranzita automatul .

Într-un APA , tranziţiile au loc sincron cu impulsurile unui generator având perioada stabilită în funcţie de tipul de acces al circuitului cu memorare , iar o stare a automatului are durata unei perioade de tact .

Structura unui APA este cea a unui automat Mealy sincron în care partea combinaţională este realizată cu o memorie , iar registrul de stare menţine într-o secvenţă starea internă din care s-a tranzitat şi combinaţia logică de intrare ce a fost implicată în tranziţie (figura 5).

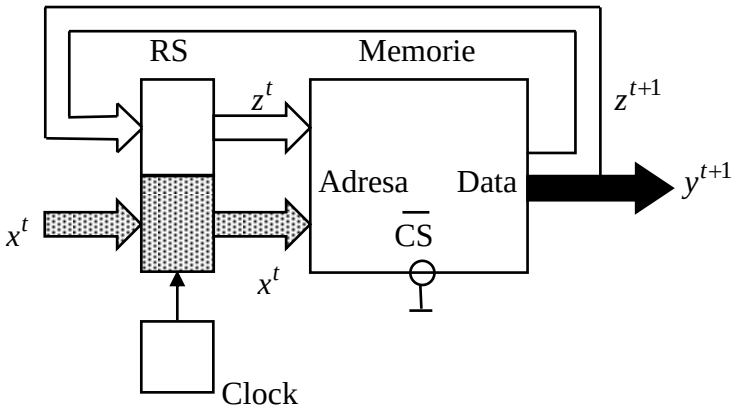


Fig. 5 – Structură automat programabil algoritmic

Într-un APA , memorarea se poate realiza atât în circuite de memorie , cât şi în arii logice programabile notate PLA (Programable Logic Array).

Sub comanda unui automat secvenţial , fiecare sarcină se realizează după un algoritm ce specifică operaţiile de efectuat într-o anumită succesiune temporală de etape şi care se reprezintă grafic sub forma unei organigrame.

Un automat secvenţial constituie o *maşină algoritmică de stare* , notată ASM (Algorithmic State Machine), pentru că în controlul sarcinii de efectuat evoluează printr-o succesiune de stări fixată de un algoritm , reprezentarea grafică a algoritmului fiind numită *organigramă ASM* .

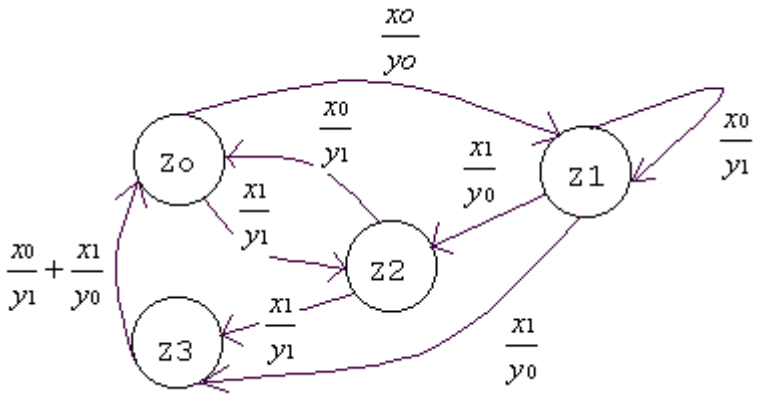
Funcţionarea unui automat secvenţial în tranziţiile posibile dintr-o stare internă a sa, are într-o organigramă ASM o reprezentare distinctă numită *bloc ASM* , ce indică starea internă curentă , testările de efectuat pentru identificarea tranziţiei , stările interne în care se poate tranzita şi comenzile date în fiecare tranziţie posibilă . Calea de legătură dintre două stări interne implicate într-o tranziţie constituie o *conexiune de stare* .

Un bloc ASM , reprezintă deci, ansamblul conexiunilor de stare pornind dintr-o aceeaşi stare internă .

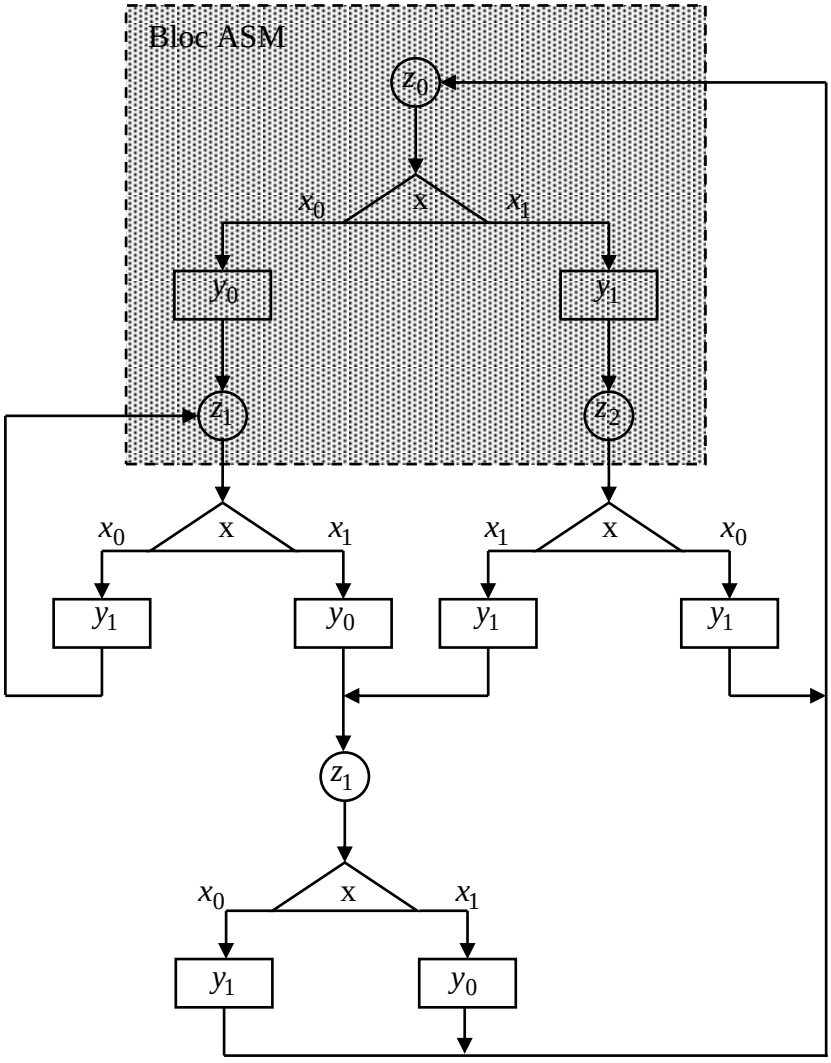
Cele trei elemente din componenţa unei organigrame ASM sunt :

- *cercul* ce indică o stare internă a automatului secvenţial ;
- *triunghiul* (sau *rombul*) ce reprezintă elementul de decizie la testarea unei condiţii ce poate fi falsă ori adevărată ;
- *dreptunghiul* ce conţine comenzile date la sfârşitul unei tranziţii .

Automatul secvenţial , al cărui graf este dat în fig. 6 , este descris prin organigrama ASM din fig. 7 , în care sunt puse în evidenţă un bloc ASM şi conexiunile de stare componente .



a) Graful unui automat secvenţial



b) Organigramă ASM

Fig. 7 – Descriere aplicaţie prin organigramă ASM

2. Memoriile într-un APA

Realizarea unui automat secvenţial constă în materializarea prin logică cablată , sau cu memorii , a funcţiilor logice de tranziţie şi de comandă , conform organigramei funcţionale a aplicaţiei de controlat .

Când numărul de stări ale automatului secvenţial este mai mare ca 20 , se recomandă , justificat economic , utilizarea în locul logicii cablate a memoriilor care conferă simplitate şi flexibilitate pentru APA . După variantele de utilizare a memoriilor se disting mai multe variante de APA .

Aşa cum se arată în fig. 8 , orice funcţie logică vectorială descrisă printr-o tabelă de adevăr , poate fi materializată printr-un automat combinaţional realizat cu o memorie , în care o combinaţie logică de intrare va adresa o locaţie , al cărei conţinut , furnizat la biţii de date , reprezintă combinaţia logică de ieşire indicată în tabela de adevăr a funcţiei .

Un automat combinaţional realizat cu o memorie având *k* biţi de adresă şi *q* biţi de date , deşi admite 2^k combinaţii logice de intrare şi maxim 2^q combinaţii logice de ieşire , oferă flexibilitate maximă în realizarea unei funcţii logice doar prin schimbarea informaţiei înscrise în locaţiile de memorie .

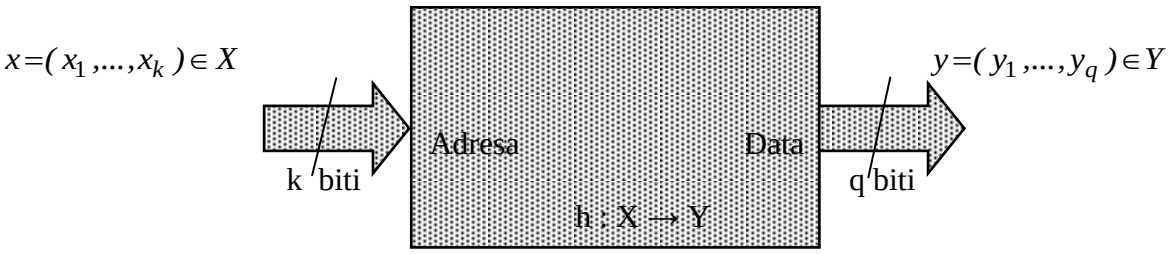
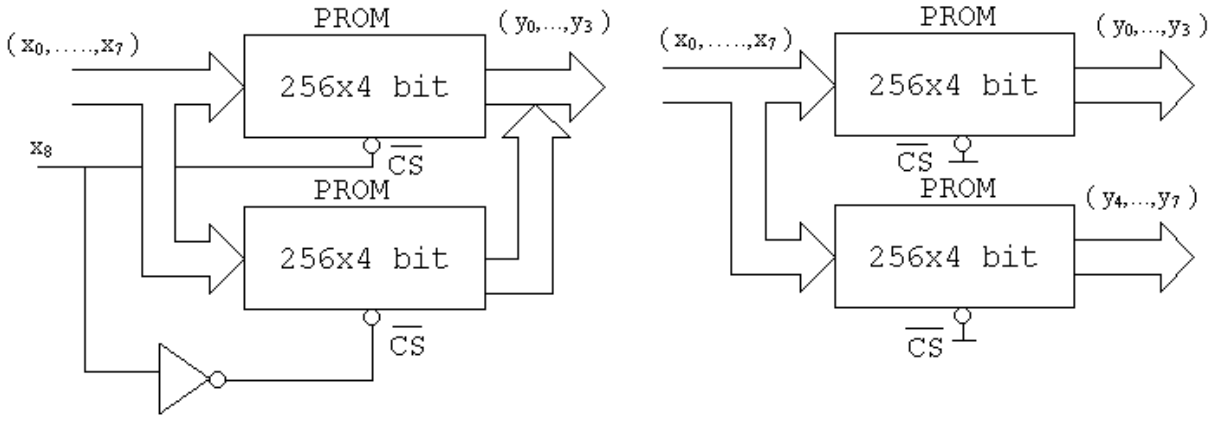


Fig. 8 – Automat combinaţional ce materializează funcţia logică vectorială h : X –> Y

Într-un automat combinaţional folosind memorii şi nu logică cablată , prin eliminarea decodării combinaţiilor logice de intrare , se realizează un transfer direct intrare-ieşire într-un interval de timp scurt limitat doar de timpul de acces al memoriei . Primele utilizări ale memoriei în logica combinaţională au fost de tipul tabelelor de conversie .

Dacă numărul de intrări şi/sau ieşiri ale unui automat combinaţional este mai mare ca cel disponibil la un circuit de memorie se vor folosi mai multe circuite de memorie interconectate pentru a realiza extensia pe intrare şi/sau ieşire aşa cum se arată în fig. 9 .



a) Extensie pe intrare b) Extensie pe ieşire

Fig. 9 Extensia pe intrarea şi ieşirea unui automat combinaţional cu memorii

Când din cei k+p biţi ai unei locaţii din memoria unui automat combinaţional , sunt alocaţi pentru comanda aplicaţiei doar k biţi , aceştia vor fi utilizaţi în adresarea indirectă a maxim 2^k combinaţii logice de ieşire într-o memorie suplimentară , aşa cum se arată în fig. 10 .

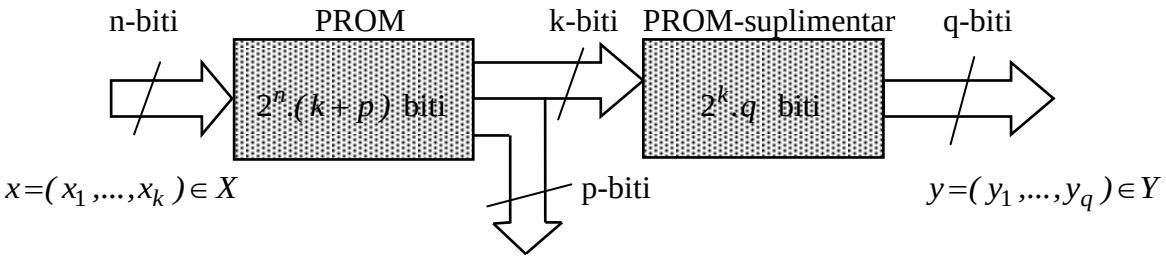


Fig. 10 – Automat combinaţional cu memorii folosind adresare indirectă

În memoria suplimentară a unui automat combinaţional alegerea între locaţiile conţinând diferite combinaţii logice de ieşire posibile pentru o aceeaşi combinaţie logică de intrare , se face cu semnale logice de condiţionare folosite ca biţi de adresă (fig. 11) .

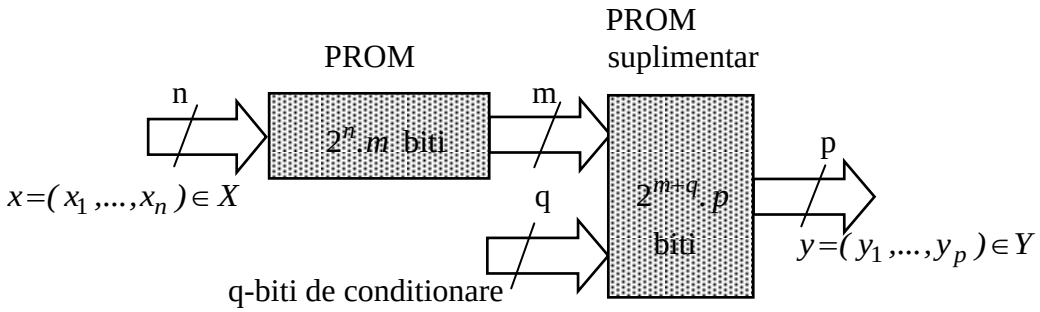


Fig. 11 – Automat combinaţional cu ieşiri condiţionate

Un automat combinaţional , în care cei *q* biţi de condiţionare reprezintă ieşirile unui numărător al impulsurilor unui generator de tact , se transformă într-un APA la care , pe durata unei combinaţii logice de intrare , se generează succesiv 2^q combinaţii logice de ieşire ce comandă o structură din circuite logice interconectate să realizeze în mod secvenţial o funcţie logică complexă . Această structură de APA , reprezentată în figura 12 , stă la baza realizării unităţilor aritmetice şi logice, o combinaţie logică de intrare , ce reprezintă codul unei instrucţiuni identificând zona memoriei de comandă ce conţine combinaţiile logice de ieşire prin care se comandă efectuarea secvenţială a instrucţiunii .

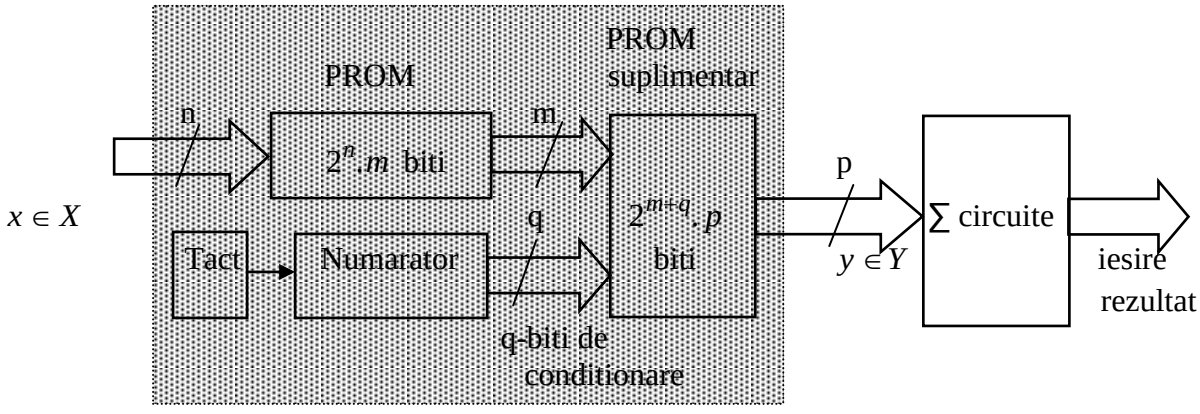


Fig. 12 – Unitate aritmetică şi logică cu APA

Locaţiile memoriei de ieşire a unui APA se împart pe grupuri de biţi , numite câmpuri , destinate a realiza comanda aplicaţiei controlate , testarea unor condiţii şi conexiunea către următoarea stare internă în care se va tranzita , operaţii prezente în organigrama ASM a oricărei aplicaţii .

3. APA cu memorie în format fix

Dacă fiecare locaţie a memoriei unui APA are aceeaşi împărţire pe câmpuri , se spune că memoria este în *format fix* .

Prin câmpul numit *conexiune stare* se realizează de fapt o ramnificare în memoria automatului către adresa următoarei stări interne în care se va tranzita . După modul de identificare în memorie a adresei stării interne în care se va tranzita , se deosebesc următoarele tipuri de APA cu memorie în format fix :

a) APA cu memorie adresabilă printr-un câmp conexiune stare

La acest tip de APA , cu structura dată în figura 13 , memoria se adresează atât cu biţii combinaţiei logice de intrare , cât şi cu biţii câmpului conexiune stare , dezavantajul major fiind gradul redus de ocupare a memoriei când combinaţiile logice de intrare sunt formate din mulţi biţi . Un astfel de APA are însă o viteză mare de operare , perioada unei stări putând fi redusă la valoarea

$$T_S \geq t_{\text{acces memorie}} + t_{\text{răspuns registru stare}}$$

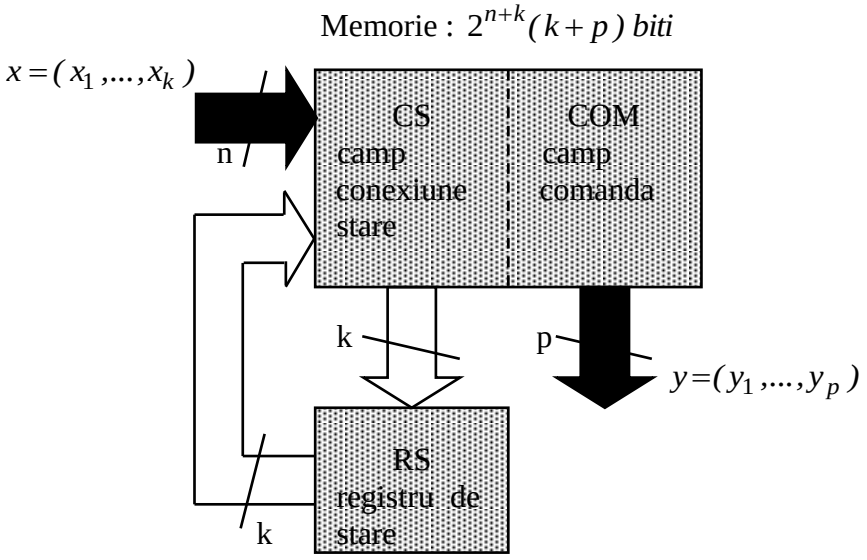


Fig. 13

Capacitatea mare de memorie rămasă nefolosită a impus găsirea de noi structuri de APA ca să permită utilizarea unor memorii de capacitate redusă .

b) APA cu memorie adresabilă prin două câmpuri conexiune stare

Pentru acest tip de APA , reprezentat în figura 14 , este necesară o logică suplimentară pentru tranzitarea într-una din cele două stări posibile indicate în câmpurile conexiune stare . Astfel , prin câmpul *test* , cu multiplexorul MUX 1 , se selectează bitul de intrare a cărui stare logică decide care din cele două zone conexiune stare va fi transferată cu multiplexorul MUX 2 în registrul de stare RS.

Memorie : $2^n\,(m+2n+p)\ biti$

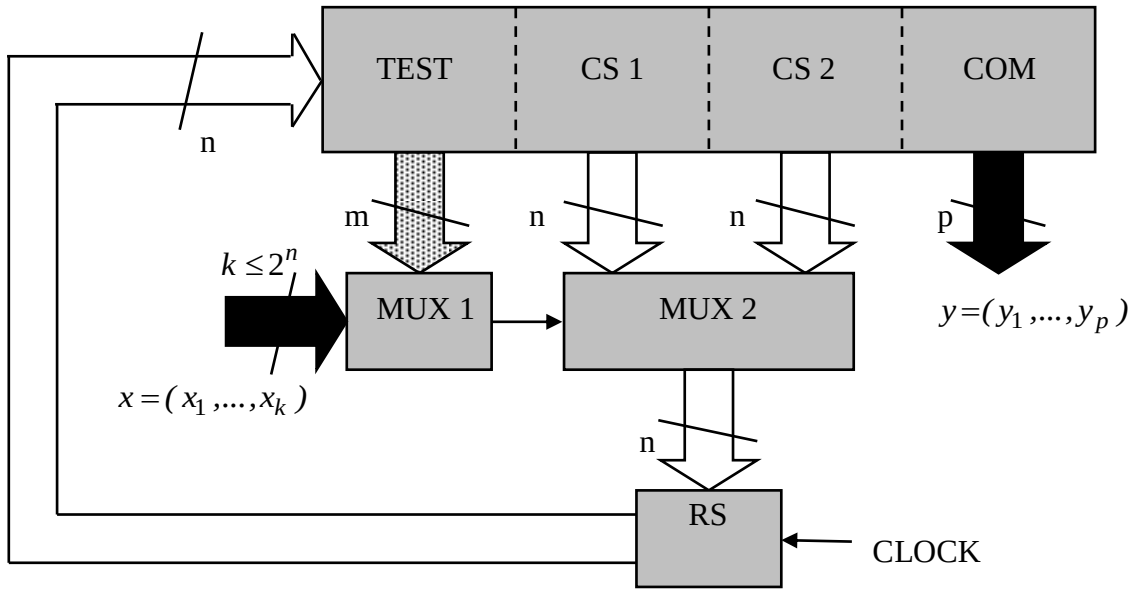
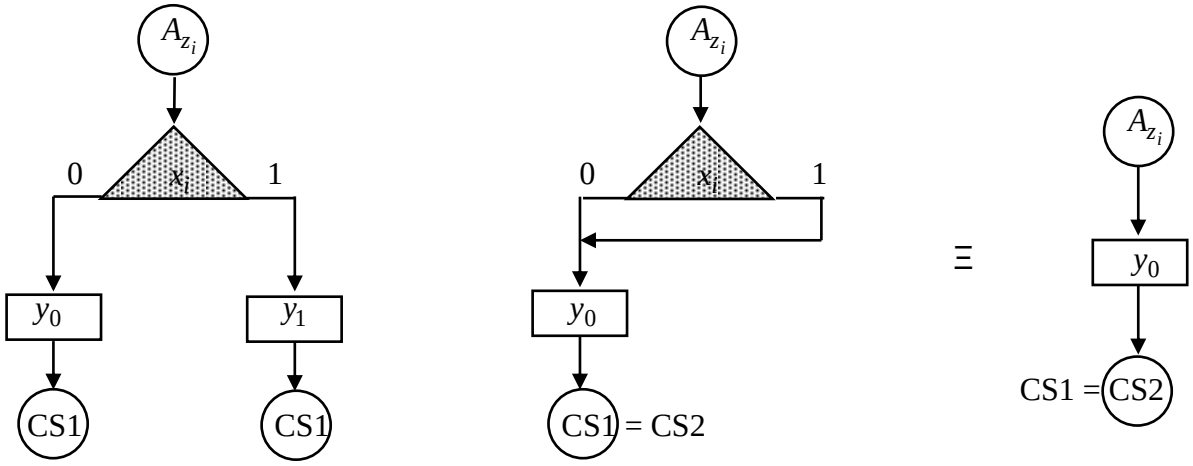


Fig. 14



a) b)

Fig. 15

Starea z_r , din figura 15 , în care se tranzitează dintr-o stare z_j , prin testarea intrării x_i , se determină prin adresa locaţiei corespunzătoare din memorie conform relaţiilor :

$$A_{Zr} = \begin{cases} CS_1\text{, dacă } x_i = 0 \\ CS_2\text{, dacă } x_i = 1 \end{cases},$$

relaţie ce descrie un bloc ASM cu reprezentarea din figura 15a .

Pentru un bloc ASM cu reprezentarea din figura 15b cele două câmpuri conexiune stare , notate CS_1 şi CS_2 trebuie să fie identice .

Memoria unui asemenea tip de APA se caracterizează printr-o scădere a numărului de locaţii şi o creştere a numărului de biţi pe locaţii .

O variantă de APA cu memorie adresabilă prin două câmpuri de conexiune stare , prezentată în figura 16 , se caracterizează prin eliminarea câmpului test din locaţiile de memorie al căror număr de biţi se reduce şi prin complexitatea multiplexorului MUX 2 . Această variantă se recomandă când un câmp conexiune stare are acelaşi număr de biţi ca al unei combinaţii logice de intrare .

Memorie : $2^n(2n+p)$ biti

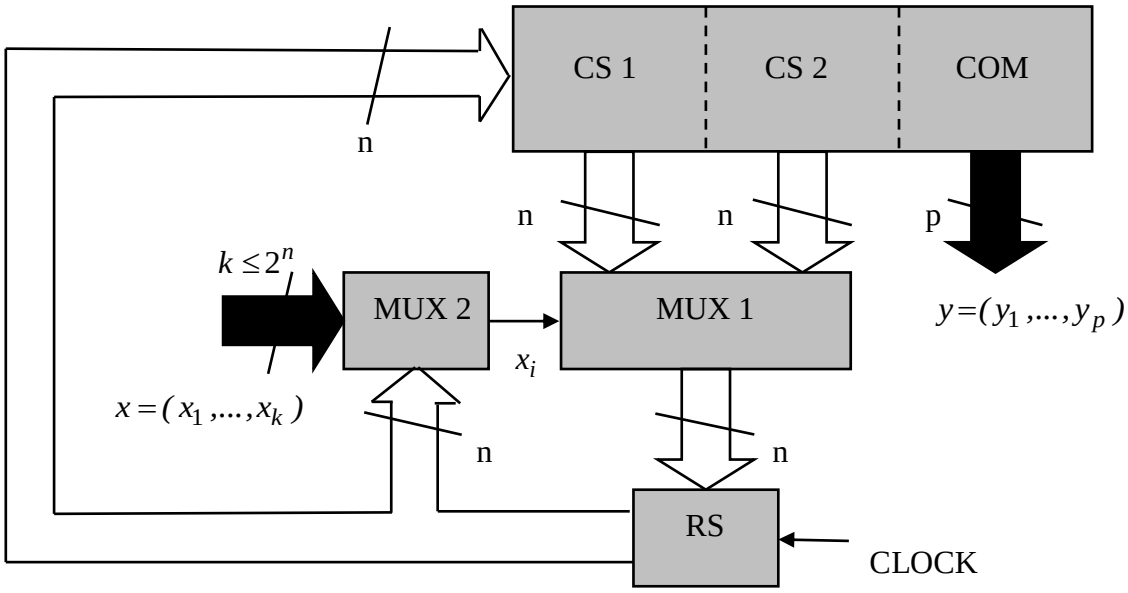


Fig.16

c) <u>APA cu memorie adresabilă printr-un câmp conexiune stare şi prin incrementare</u>

Structura unui asemenea tip de APA , dată în figura 17, oferă avantajul existenţei unui singur câmp conexiune stare în locaţiile de memorie al căror număr de biţi se reduce astfel şi se recomandă în aplicaţiile cu număr mic de intrări a căror selecţie se face printr-un câmp test redus .

Memorie : $2^n(m+n+p)$ *biti*

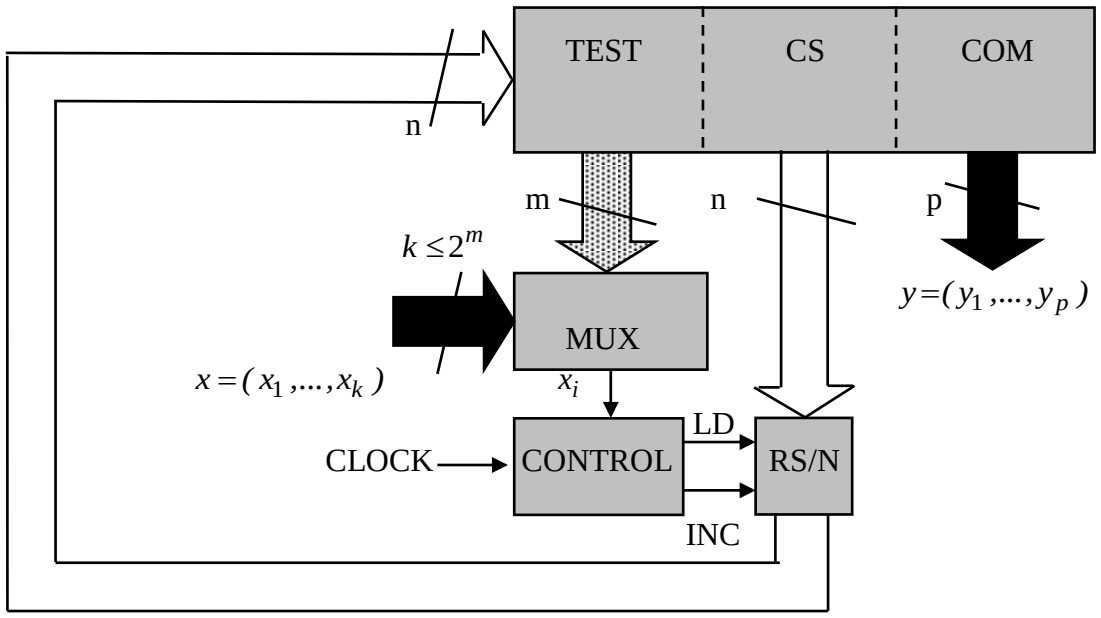


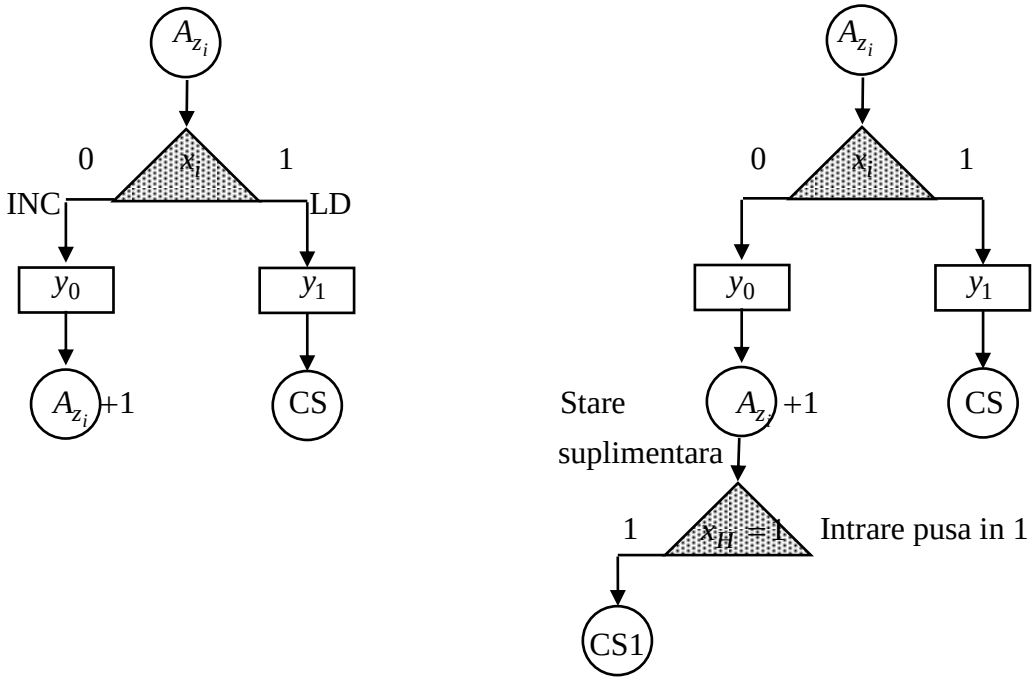
Fig 17

La un astfel de APA , dintr-o stare internă curentă z_j , în funcţie de starea logică a intrării x_i indicată în câmpul test , se poate tranzita într-o stare internă z_r aflată în memoria automatului fie la adresa următoare stării curente z_j , fie la adresa indicată în câmpul conexiune stare CS , conform relaţiilor :

$$A_{Zr} = \begin{cases} A_{Zj} + 1, \text{ când } x_i = 0 \\ CS \quad , \text{ când } x_i = 1 \quad , \end{cases}$$

relaţii ce descriu un bloc ASM cu reprezentarea din figura 18 .

Adresarea memoriei automatului se face cu un registru de stare cu numărare RS/N , care pentru incrementare este comandat cu semnal INC , iar pentru încărcare cu adresa din câmpul conexiune stare este comandat cu semnalul LD .



a) b)

Fig. 18

Cu un astfel de APA, pentru a realiza salturi pe ambele ramnificări dintr-o decizie , atunci , pe ramura cu incrementare , se introduce o stare suplimentară din care se face salt prin testarea unei intrări x_H pusă în starea logică 1 , structura unui asemenea bloc ASM având reprezentarea din figura 18b .

d) <u>APA folosind combinaţii de memorii</u>

Un astfel de APA, având structura din figura 19 , se utilizează în aplicaţiile ce necesită atât ieşiri condiţionate , cât şi ieşiri necondiţionate .

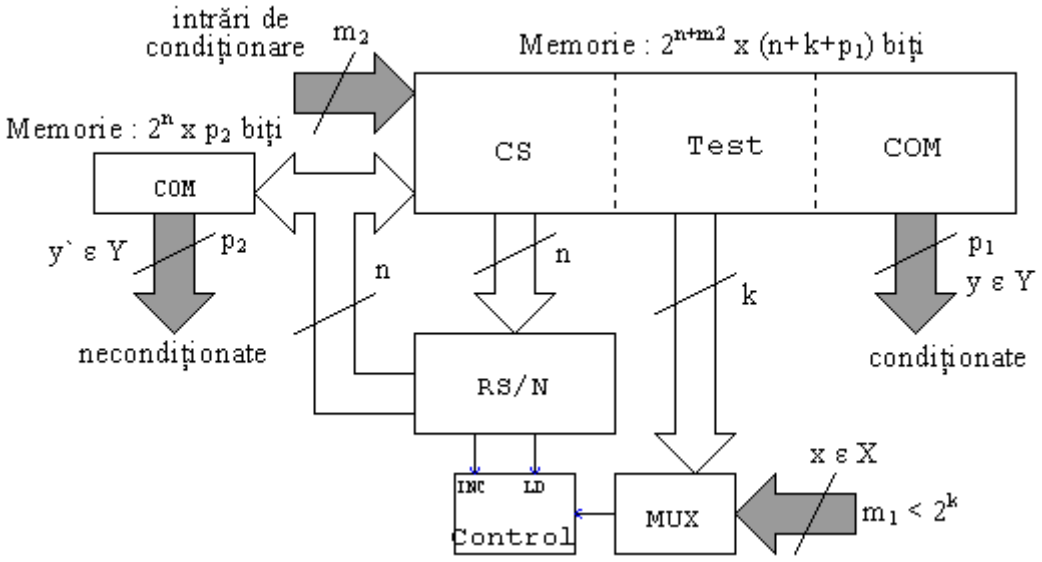


Fig. 19

4. APA organizată cu memorie programată în format variabil

Un APA cu memorie adresabilă în format fix , deşi are un număr mare de biţi pe o locaţie de memorie , prezintă avantajul că tranziţia către o stare internă se face doar într-un tact al generatorului pilot . Datorită standardizării circuitelor de memorie pe locaţii de opt biţi , o memorie adresabilă în format fix va fi organizată pe locaţii formate dintr-un multiplu de opt biţi şi realizată prin mai multe circuite de memorie adresată în paralel . Acest fapt determină un grad redus de ocupare pentru o memorie adresabilă în format fix .

Creşterea gradului de ocupare a memoriei se realizează în *APA organizată cu memorie programată în format variabil* în care fiecare stare a automatului este descrisă în mai multe locaţii de memorie organizate pe octeţi . Astfel , o stare a unui APA cu memorie programată în format variabil va fi parcursă în atâtea tacte ale generatorului pilot , câte cuvinte intră în descrierea unei stări , fiecare cuvânt având propria împărţire pe câmpuri de biţi. Identificarea fiecărui cuvânt dintr-o stare a unui APA cu memorie programată în format variabil se realizează prin biţii câmpului de identificare format , conţinut .

După modul de împărţire pe octeţi a câmpurilor *test , conexiune stare şi comandă* ale unei stări , se deosebesc următoarele tipuri de APA cu memorie programată în format variabil :

a) APA organizată cu memorie în format variabil şi separare pe grupuri de câmpuri

Structura de bază a unui asemenea automat este dată în figura 20 , în care o tranziţie se realizează fie prin salt la adresa din câmpul conexiune stare al stării curente , fie la adresa determinată prin incrementare cu 1 a adresei de încheiere a stării curente , după cum starea logică S_{Xi} a intrării x_i selectată prin câmpul test este 1 , respectiv 0 . Fiecare stare dintr-un asemenea automat este descrisă prin două cuvinte a căror identificare se face printr-un câmp identificare notat IF şi format dintr-un singur bit .

Pentru acest tip de automat , generarea adreselor de explorare a stării curente şi de tranzitare în următoarea stare se realizează conform relaţiilor :

$$ADR_{k+1} = \begin{cases} ADR_k + 1 & \text{, dacă } IF + IF \cdot \overline{S} = 1 \\ CS & \text{, dacă } S \cdot \overline{IF} = 1 \end{cases} ,$$

o tranziţie realizându-se ca o structură logică IF….THEN , descrisă într-o organigramă ASM , prin blocul ASM dat în figura 21 .

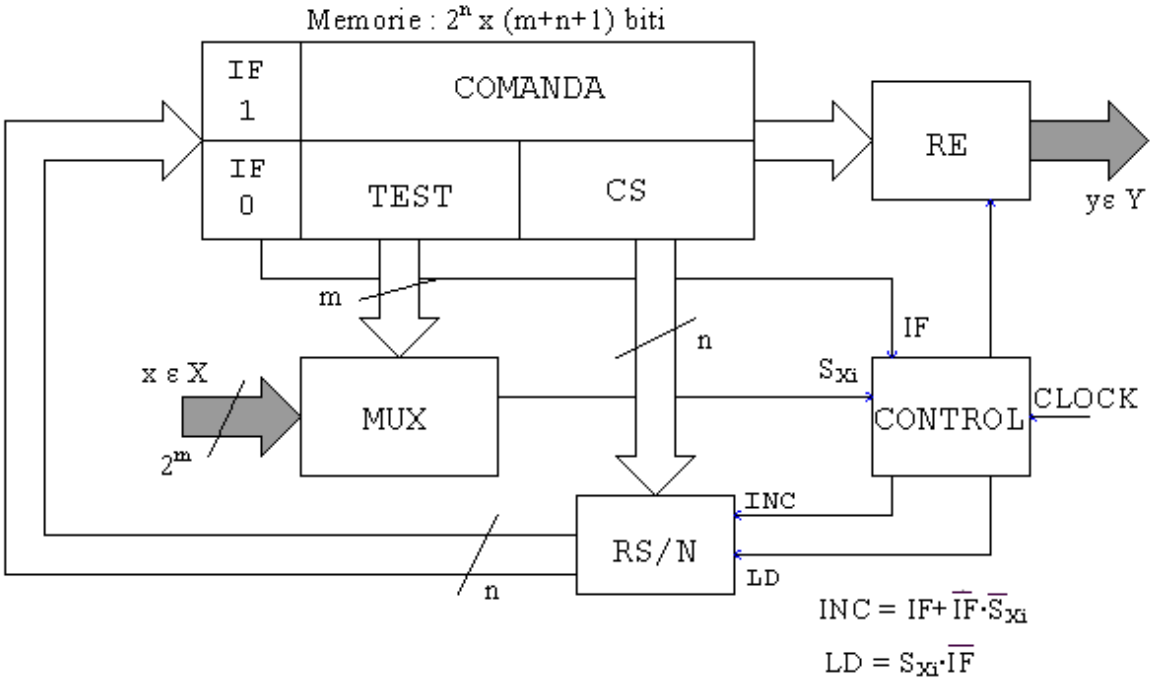


Fig. 20

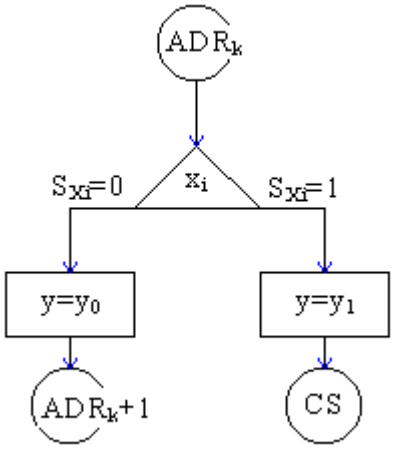


Fig. 21

Se observă din cele prezentate, că , spre deosebire de un APA cu memorie adresabilă în format fix , un APA cu memorie programată în format variabil mai conţine un registru de ieşire RE pentru menţinerea semnalelor de comandă pe durata unei stări a automatului şi o logică de control a registrului de stare cu numărare RS/N şi a celui ce conţine RE .

O conexiune de stare multiplă poate fi realizată cu structura de APA prezentată în figura 22 , în care o tranziţie este descrisă prin relaţiile :

$$ADR_{k+1} = \begin{cases} ADR_k + 1 & \text{, dacă IF=1} \\ CS_i & \text{, dacă IF=0} \end{cases}$$

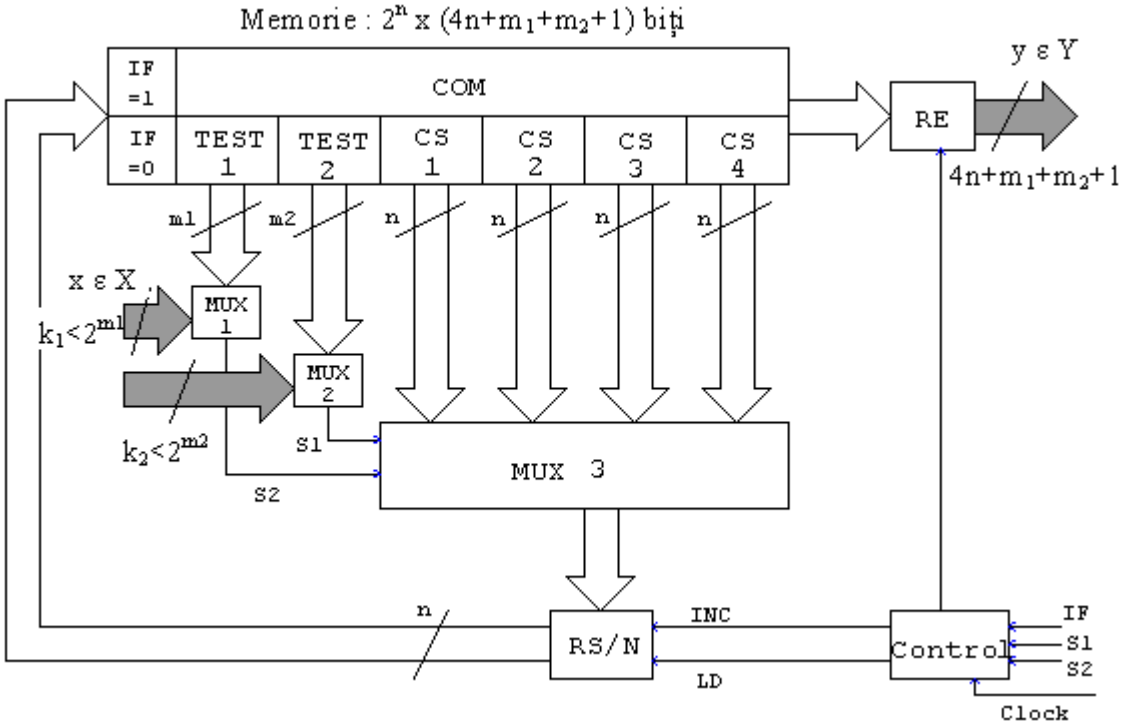


Fig. 22

În automatul cu structura din figura 22 , sub comanda câmpurilor Test 1 şi Test 2 se transferă la ieşirile multiplexoarelor MUX 1 şi MUX 2 două variabile de intrare, a căror combinaţie logică S1.S2 , identifică starea în care se va tranzita , prin selectarea câmpului conexiune stare cu numărare RS/N . O tranziţie este realizată de acest automat ca o structură logică IF…..THEN….ELSE .

b) APA organizată cu memorie în format variabil şi separare completă a câmpurilor

Într-un asemenea APA , fiecare din cuvintele ce descriu o stare a automatului se află în unul din câmpurile test , conexiune stare şi comandă folosite . Identificarea fiecăruia din cele N cuvinte ale unei stări a automatului se realizează prin câmpul *identificare format* conţinut alcătuit dintr-un număr de k biţi ce se determină cu relaţia :

$$2^k \geq N$$

Astfel pentru APA din figura 23, în care fiecare stare e formată din N = 4 cuvinte , câmpul *identificare format*, prezent în toate cuvintele , conţine k = 2 biţi notaţi IF1 şi IF2 .

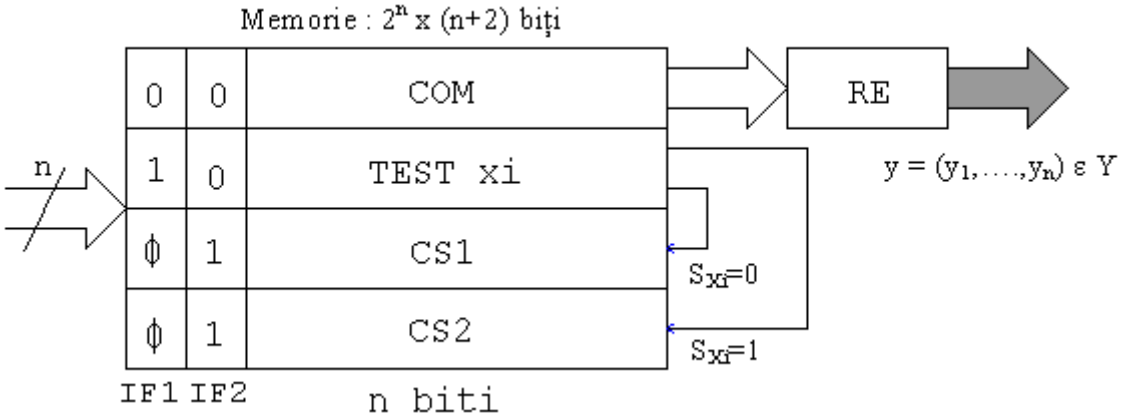


Fig. 23

Fiecare din acest APA va fi parcursă prin citirea a trei din cele patru cuvinte prin care este descrisă , întrucât prin câmpul test se alege doar un cuvânt conexiune stare , corespunzător stării logice a variabilei de intrare x_i selectate . O tranziţie se realizează acum conform relaţiilor :

$$ADR_{k+1} = \begin{cases} ADR_k + 1 & \text{, dacă } IF_2 \cdot (IF_1 + S) = 1 \\ ADR_k + 2 & \text{, dacă } \overline{IF_1} \cdot IF_2 \cdot S = 1 \\ CS_i & \text{, dacă } IF_2 = 1 \end{cases} ,$$

ca o structură logică IF….THEN….ELSE al cărei bloc ASM este reprezentat în figura 24 .

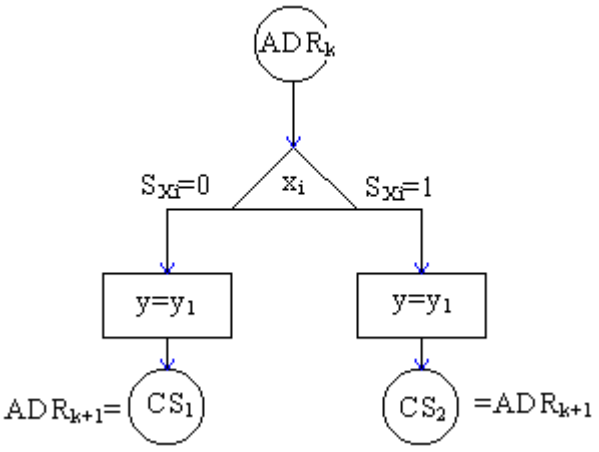


Fig. 24 – Bloc ASM pentru structura logică IF…THEN…ELSE

Reducerea memoriei într-un APA cu separare completă a câmpurilor în memorie , se obţine printr-o descriere a unei stări , sub forma structurii logice IF….THEN , aşa cum se exemplifica în figura 25 , o tranziţie fiind descrisă cu relaţiile :

$$ADR_{k+1} = \begin{cases} ADR_k+1, \text{dacă } IF \cdot S_{Xi} + IF \cdot S_{Xi} = 1 \\ ADR_k+2, \text{ dacă } IF \cdot S_{Xi} = 1 \\ CS \quad , \text{ dacă } IF \cdot \overline{S_{Xi}} = 1 \end{cases}$$

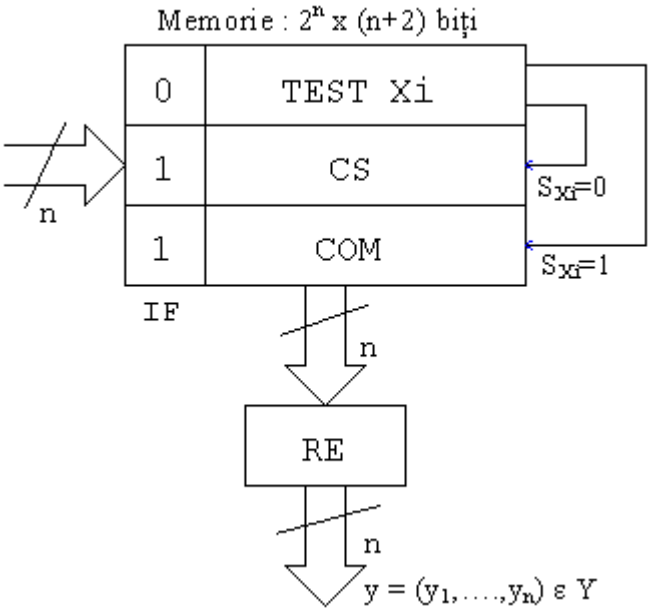


Fig. 25

Într-un astfel de APA , un bloc ASM se poate realiza şi ca o structură logică WHILE dată în figura 26 , o tranziţie fiind descrisă cu relaţiile :

$$ADR_{k+1} = \begin{cases} ADR_k+1 \text{ , dacă } IF \cdot S_{Xi} + IF \cdot S_{Xi} = 1 \\ ADR_k+2 \text{ , dacă } S_{Xi} \cdot \overline{IF} = 1 \\ ADR_{k-p} \text{ , dacă } S_{Xi} \cdot \overline{IF} = 1 \end{cases}$$

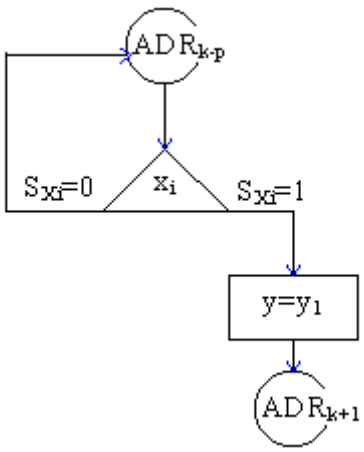
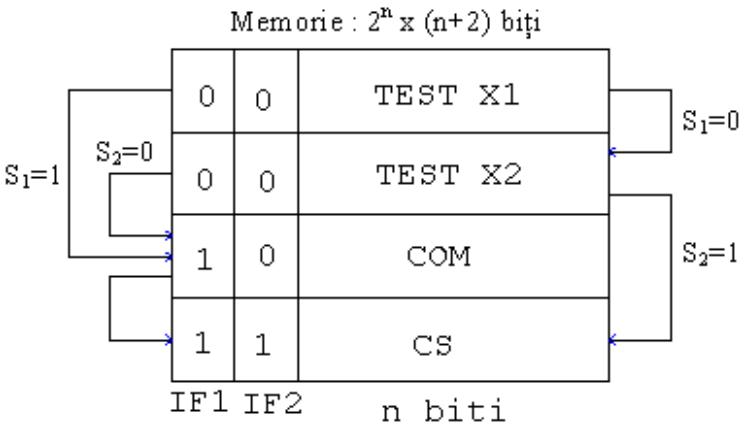


Fig. 26 – Realizarea unei structuri WHILE

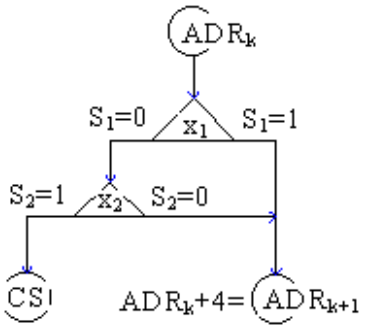
Într-un APA cu separare completă a câmpurilor, pentru realizarea , în tranziţii , de succesiuni de teste, fiecare stare ce conţine în descrierea sa câte un câmp test pentru fiecare testare de efectuat.

Această situaţie este exemplificată cu automatul din figura 27a în care o tranziţie este reprezentată prin blocul ASM din figura 27b şi descrisă prin relaţiile :

$$ADR_{k+1} = \begin{cases} ADR_k+1 \text{ , dacă } IF_2[\overline{IF_1}+IF_1(\overline{S_1}+\overline{S_2})]=1 \\ ADR_k+2 \text{ , dacă } IF_1\cdot\overline{IF_2}(S_1+S_2)=1 \\ CS \quad \text{ , dacă } IF_2=1 \end{cases}$$



a)



b)

Fig. 27 – APA cu multiple câmpuri TEST