

## ARII LOGICE PROGRAMABILE

### STRUCTURĂ INTERNĂ ȘI FUNCȚIONARE

Ariile logice programabile , notate *PLA* (=Programmable Logic Array) , simplifică realizarea automatelor secvențiale .

Orice funcție logică  $f(x_0, x_1, \dots, x_n)$  poate fi exprimată analitic , fie în *forma canonică disjunctivă* prin combinațiile de intrare  $\alpha = (\alpha_0, \alpha_1, \dots, \alpha_n)$  în care ia valoarea logică „1” cu relația:

$$\begin{aligned} f(x_0, x_1, \dots, x_n) &= \sum_{\alpha=(\alpha_0, \alpha_1, \dots, \alpha_n)} \left( \prod_{i=0}^n x_i^{\alpha_i} \right) = \sum_{a=(a_0, a_1, \dots, a_n) \in B_2^n} [f(a) \cdot \left( \prod_{i=0}^n x_i^{a_i} \right)] = \\ &= \sum_{a=(a_0, a_1, \dots, a_n) \in B_2^n} [f(a) \cdot P_a(x_0, x_1, \dots, x_n)] , \end{aligned}$$

în care  $P_a(x_0, x_1, \dots, x_n) = \prod_{i=0}^n x_i^{a_i}$  se numește *termen produs* , fie în *forma canonică conjunctivă* prin combinațiile de intrare  $\beta = (\beta_0, \beta_1, \dots, \beta_n)$  în care ia valoarea logică „0” cu relația :

$$\begin{aligned} f(x_0, x_1, \dots, x_n) &= \prod_{\beta=(\beta_0, \beta_1, \dots, \beta_n)} \left( \sum_{i=0}^n \overline{x_i^{\beta_i}} \right) = \prod_{a=(a_0, a_1, \dots, a_n) \in B_2^n} [f(a) + \sum_{i=0}^n \overline{x_i^{a_i}}] = \\ &= \prod_{a=(a_0, a_1, \dots, a_n) \in B_2^n} [f(a) + S_a(x_0, x_1, \dots, x_n)] , \end{aligned}$$

în care  $S_a(x_0, x_1, \dots, x_n) = \sum_{i=0}^n \overline{x_i^{a_i}}$  se numește *factor sumă* .

Formele canonice disjunctivă și conjunctivă de exprimarea unei funcții logice , conținând doar operațiile *AND* logic și *OR* logic pot fi materializate cu diode . Astfel , prin folosirea diodelor , un termen produs poate fi materializat ca în fig.1-5 , în timp ce materializarea formei canonice disjunctive a unei funcții logice este arătată în fig.2-5 .

Realizarea mai multor funcții logice în formă canonică disjunctivă a condus la o *matrice cu diode* , a cărei structură este data în fig.3-5 , conținând o matrice *AND* având fiecare linie conectată la o variabilă de intrare și generând pe fiecare coloană câte un termen produs din funcțiile logice de materializat și o matrice *OR* având pe fiecare coloană conectată la câte un termen produs furnizat de matricea *AND* și generând pe fiecare linie a sa câte o funcție logică ca sumă a unor termeni produs .

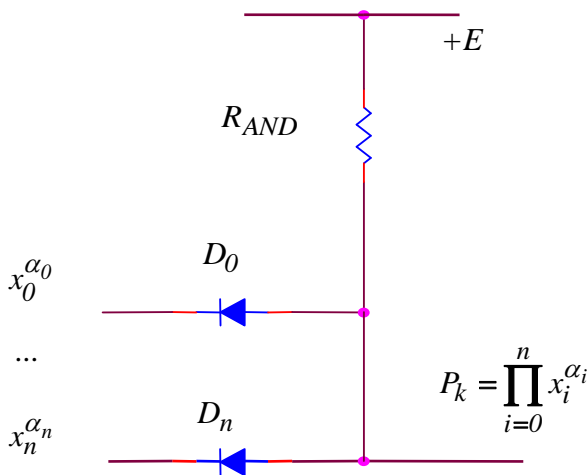


Fig.1-5 Materializare termen produs

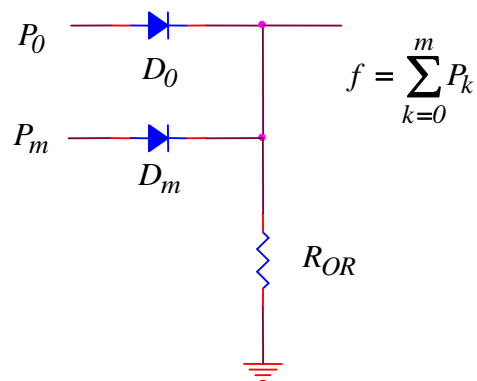


Fig.2-5 Materializare formă canonică disjunctivă

Într-o materializare cu diode , rezistoarele  $R_{AND}$  și  $R_{OR}$  se dimensionează pentru a obține tensiunile aferente stărilor logice „0” și „1” pe sarcinile conectate la ieșirile matricei . Pentru ca rezistoarele de sarcină să nu afecteze tensiunile stărilor , conectarea unei matrice cu diode la fiecare sarcină se va face prin câte un etaj adaptor ca de exemplu etajul Trigger Schmidt .

O *matrice cu diode* conținând și etaje separatoare pe ieșiri , realizată ca circuit integrat se numește *arie logică programabilă* , fiind notată *PLA* (= *Programmable Logic Array*) în care fiecare variabilă de intrare determină în matricea *AND* o linie pentru starea sa normală și o altă linie pentru starea sa complementară , iar conexiunile în matricile *AND* și *OR* fiind programate prin mască la fabricant .

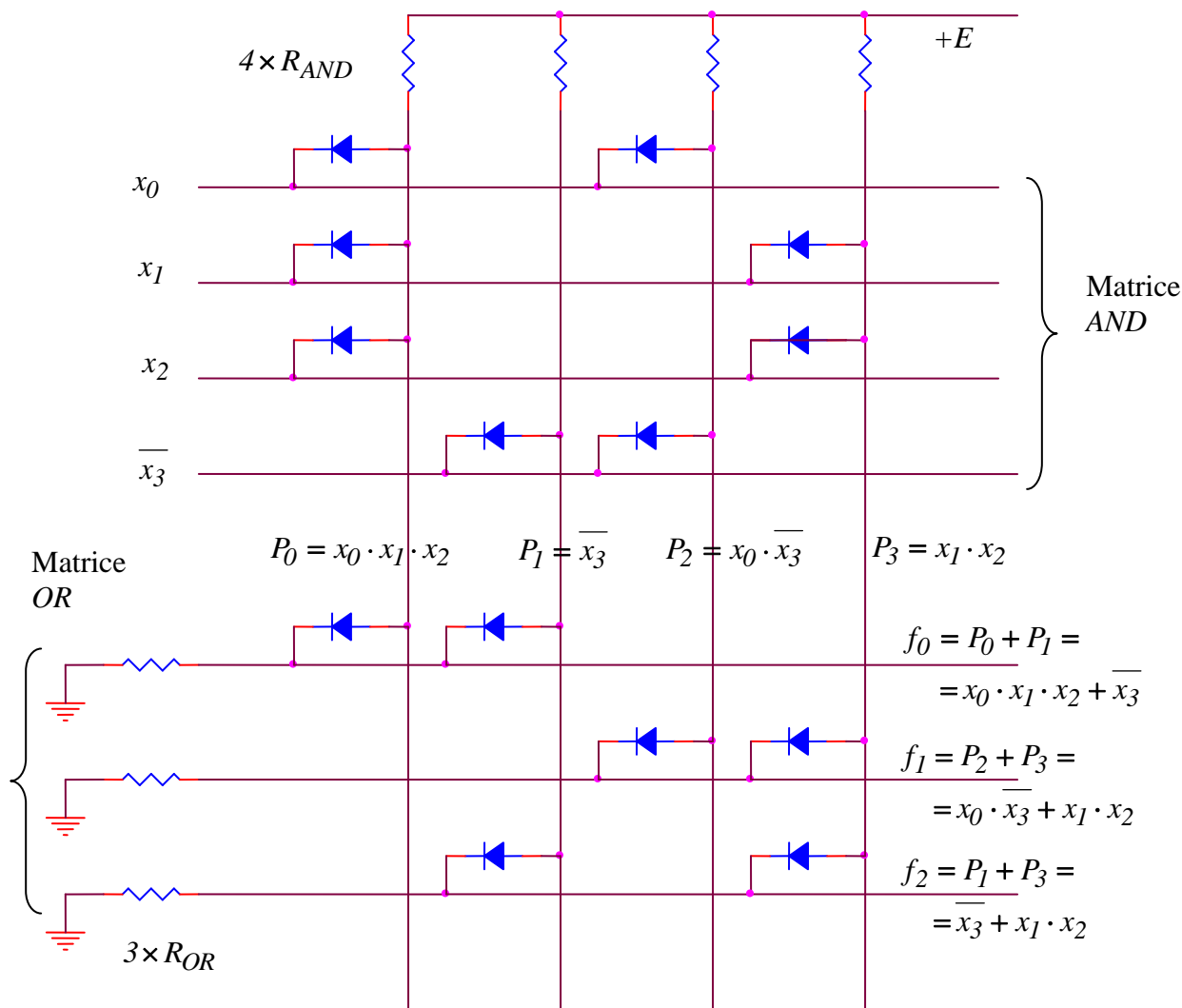


Fig.3-5 Materializare funcții logice prin *matrice cu diode*

O *matrice cu diode* , integrată și permițând programarea conexiunilor prin ardere la utilizator se notează *FPLA* (= *Field PLA*) .

Structura unui circuit *FPLA* programabil prin ardere de fuzibile este exemplificată în fig.4-5 prin circuitul *SINETICS 82S100* caracterizat prin 16 intrări , 8 ieșiri , și 48 termeni produs , iar reprezentarea simbolică a structurii acestui circuit este dată în fig.5-5 .

Fiecare ieșire a circuitului *SINETICS 82S100* are câte o poartă logica *XOR* ce prin ardere e transformată în poartă inversoare pentru generarea formei complementare a unei funcții logice . Selecția acestui circuit *FPLA* se realizează prin validarea tampoanelor cu trei stări de pe ieșiri cu semnalul  $\overline{CE} = 0$  , ieșirile fiind puse în înaltă impedanță cât timp  $\overline{CE} = 1$  .

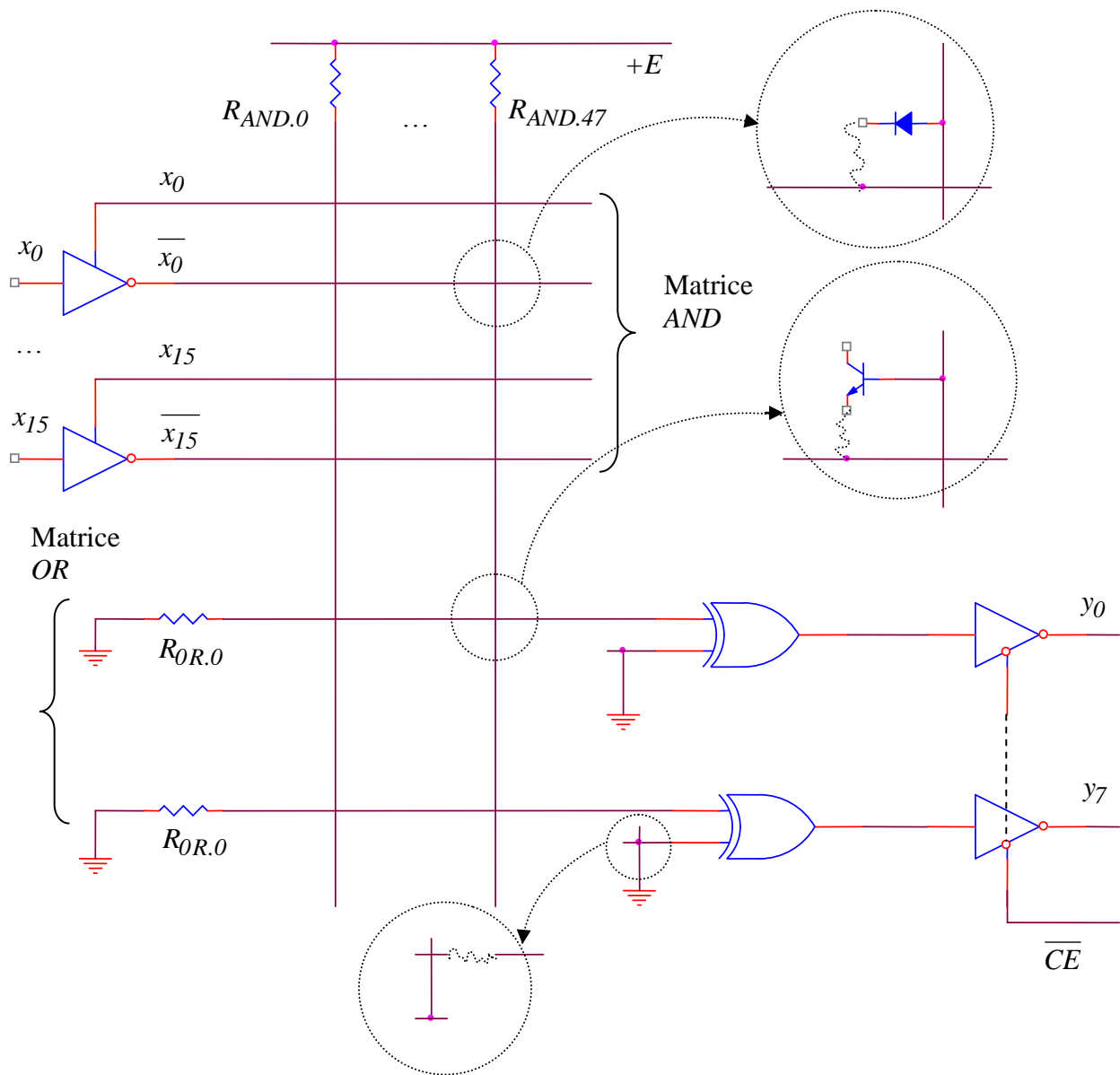


Fig.4-5 Structura internă a circuitului *SIGNETICS 82S100*

Circuitele *PLA* au o mare flexibilitate putând fi programate să realizeze diferite funcții logice complexe .

Pentru generarea cu un circuit *PLA* a unei funcții logice în formă canonică disjunctivă , numărul termenilor produs se reduce aducând funcția într-o formă disjunctivă normală prin metode analitice de minimizare folosind absorbția , alipirea și reducerea . Pentru că într-un *PLA* să nu se depășească numărul de termeni produs admis , funcțiile logice de ieșire vor fi exprimate în forme disjunctive normale și se vor grupa pe un același circuit funcții logice având în comun mai mulți termeni produs .

Într-un automat programabil algoritmic  $A=(X,Z,Y,f,g)$  cu structura de automat *Mealy sincron* dată în fig.6-5 , circuitul *PLA* materializează funcțiile logice de tranziție  $f$  și de ieșire  $g$  , registrul de stare *RS* memorează starea internă a automatului pe care o menține la intrarea *PLA* , iar registrul de ieșire *RE* memorează comenzile de ieșire ce rămân astfel stabile pe durata unei stări interne a automatului . Registrul de stare *RS* îndeplinește astfel funcția de memorie de stare , iar registrul de ieșire *RE* pe cea de memorie de ieșire (numită și memorie de comandă) .

În aplicațiile complexe un circuit *PLA* cu număr mare de intrări , ieșiri și/sau termeni produs necesari , este obținută cu mai multe circuite *PLA* în conexiuni de expandare adecvate , așa cum se arată în fig.7-5 .

Structura de automat *Mealy sincron* din fig.6-5 s-a realizat ca circuit integrat numit *secvențiator logic programabil* și notat *PLS* (= *Programmable Logic Sequencer*) . Structura internă a unui circuit *PLS* poate fi exprimată sintetic cu relația :

$$PLA = PLA + (RS + RE).$$

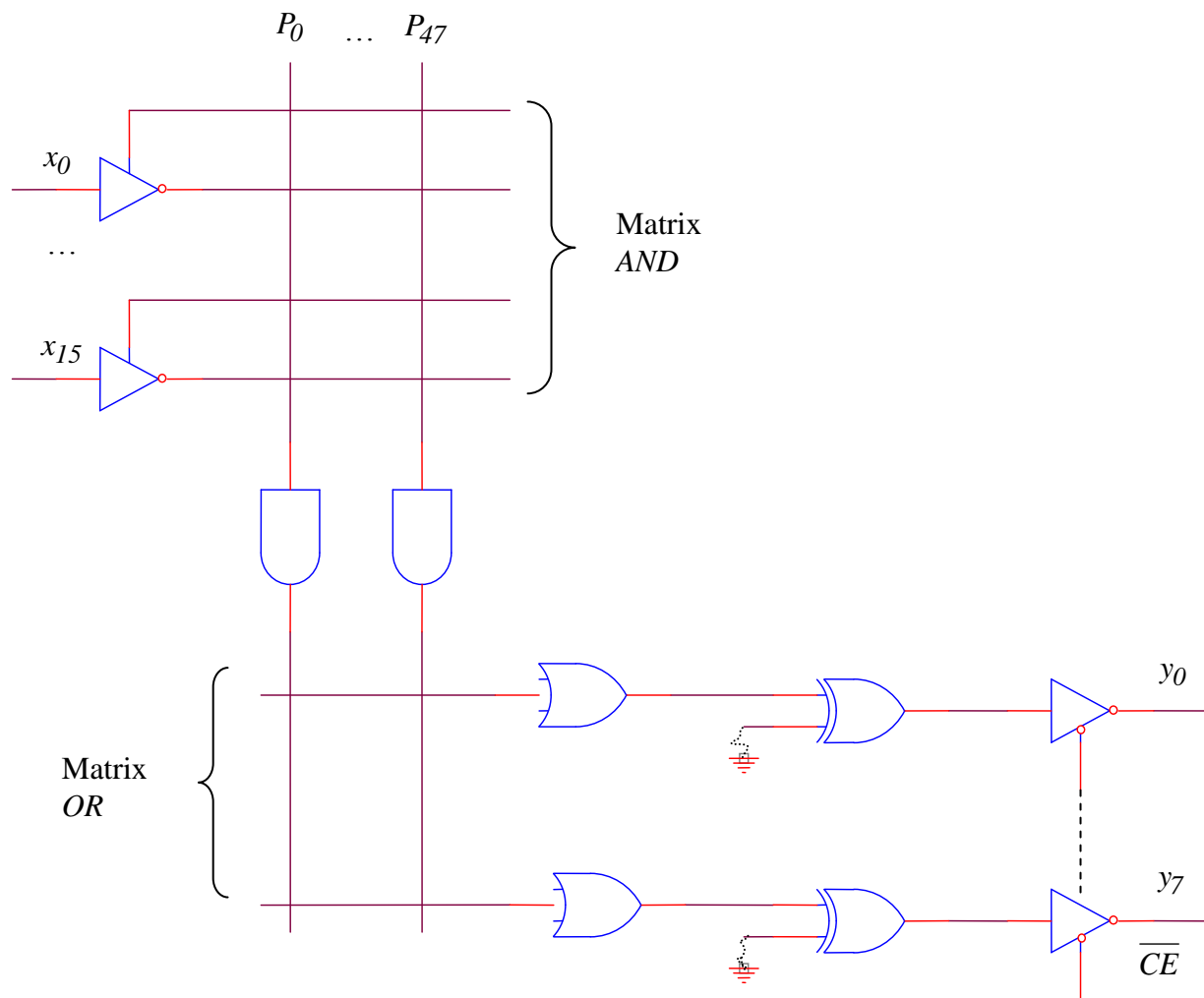


Fig.5-5 . Simbolizarea structurii interne a circuitului *SIGNETICS 82S100*

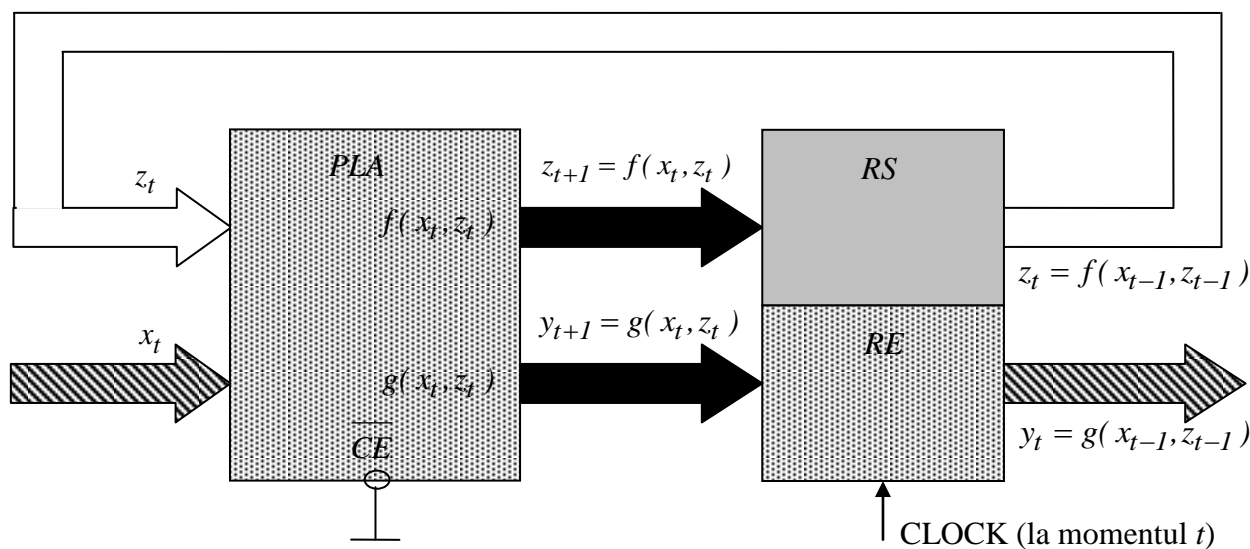


Fig.6-5 Automat *Mealy* sincron cu *PLA*

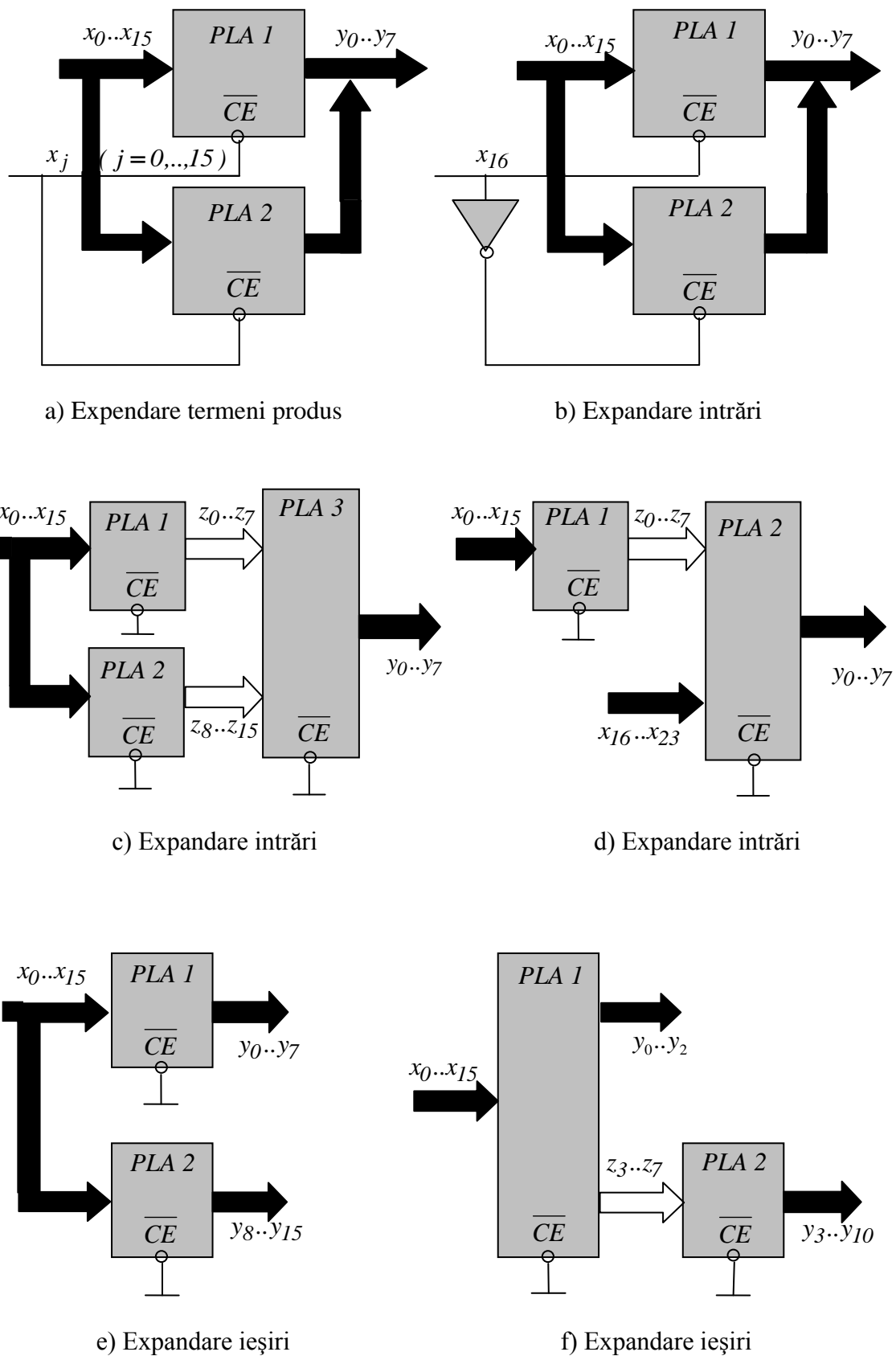


Fig.7-5 Conexiuni de expandare cu circuite PLA

Arhitectura internă a unui PLS este exemplificată în fig.8-5 unde se prezintă arhitectura internă a circuitului SIGNETICS 82S104 .

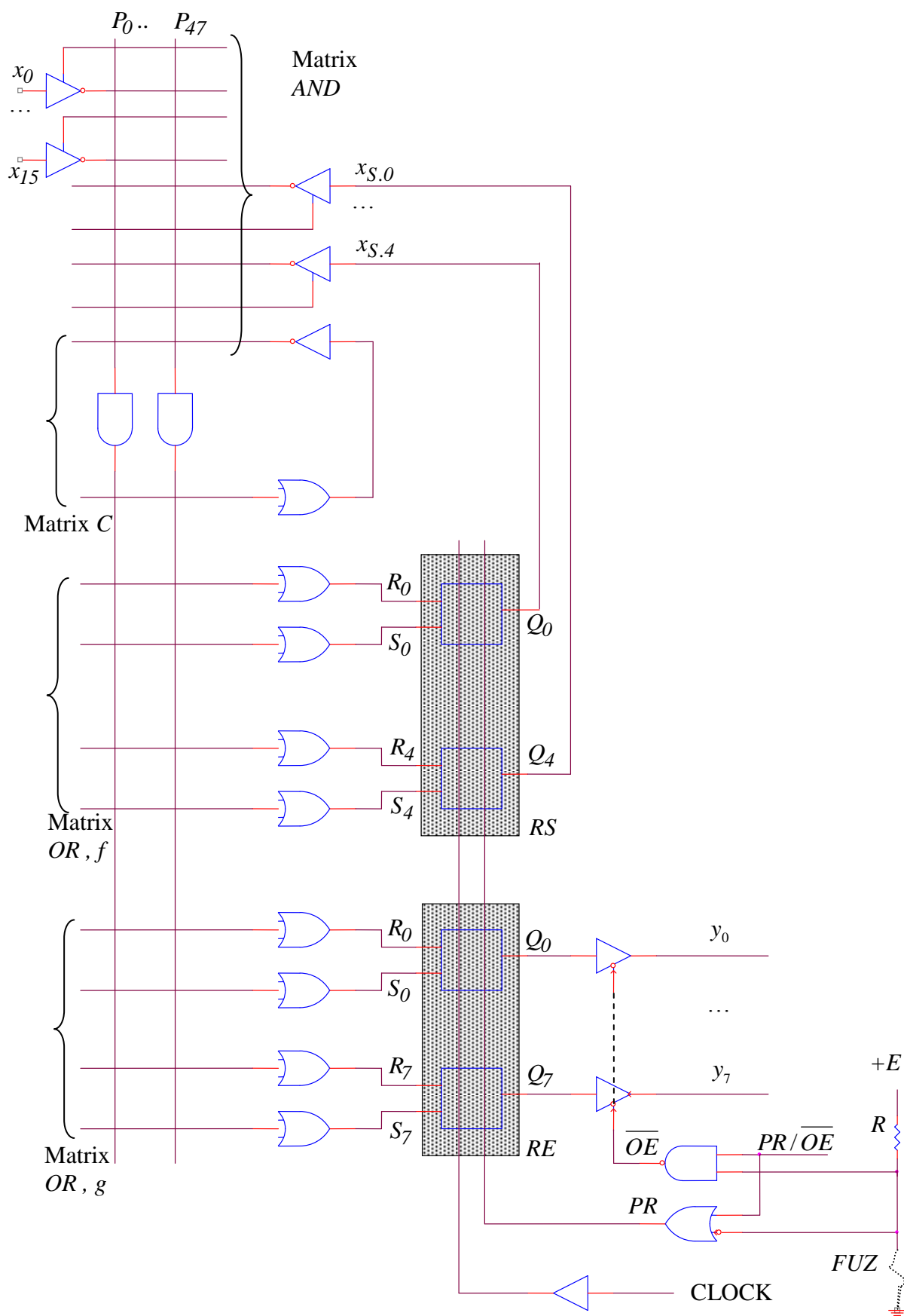


Fig.8-5 Structură circuit PLS SIGNETICS 82S104

La terminalul  $PR/\overline{OE}$  al acestui circuit , se poate realiza (prin două porți logice) , fie

comanda de presetare a registrelor  $RS$  și  $RE$  cu validarea permanentă a tampoanelor de ieșire cu trei stări , fie prin ardere, comanda de selectare a circuitului prin validarea tampoanelor de ieșire care la comanda  $\overline{OE} = 1$  trec în înaltă impedanță .

Circuitul este prevăzut cu o linie  $OR$  complementată care , fiind conectată la o linie a matricei  $AND$  permite realizarea unei reacții asincrone de la matricea  $OR$  la matricea  $AND$  , cele două linii constituind o matrice numită complementară .

Matricea complementară permite generarea de termeni produs complecși reprezentând reuniunea mai multor termeni produs și care se regăsesc în formele cononice disjunctive a unor funcții de materializat, astfel reducându-se numărul de termeni produs de programat în matricea  $AND$  a circuitului  $PLS$  .

Astfel, în scopul reducerii numărului de termeni produs , o funcție logică  $f$  poate fi materializată cu matrice complementară prin termenii produs ai funcției complementare  $\overline{f}$  , conform relației :

$$f = \overline{\overline{f}} = \sum_{\alpha=(\alpha_0, \alpha_1, \dots, \alpha_n) \in B_2^n} \left( \prod_{i=0}^n x_i^{\alpha_i} \right) = \sum_{a=(a_0, a_1, \dots, a_n) \in B_2^n} [ \overline{f(a)} \cdot \left( \prod_{i=0}^n x_i^{a_i} \right) ] ,$$

unde  $\alpha = (\alpha_0, \dots, \alpha_n)$  sunt combinații de intrare în care  $\overline{f(\alpha)} = 1$  ,  $B_2^n$  = mulțimea combinațiilor de intrare pe care  $\overline{f}$  este definită , iar :

$$x_i^{a_i} = \begin{cases} \overline{x_i} & \text{daca } a_i = 0 \\ x_i & \text{daca } a_i = 1 \end{cases}$$

### Exemplu

Să se materializeze cu un  $PLS$  funcțiile logice  $f(x_0, x_1, x_2)$  și  $g(x_0, x_1, x_2)$  cu următoarele tabele de adevăr :

$f(x_0, x_1, \dots, x_n)$	$\frac{x_0 \cdot x_1}{x_2}$	00	01	11	10	=
	0	1	1	0	1	
	1	0	1	1	1	

$g(x_0, x_1, \dots, x_n)$	$\frac{x_0 \cdot x_1}{x_2}$	00	01	11	10	=
	0	0	0	1	0	
	1	0	1	1	1	

### Rezolvare

Funcția complementară  $\overline{f}(x_0, x_1, x_2)$  , descrisă prin tabela de adevăr :

$\overline{f}(x_0, x_1, x_2)$	$\frac{x_0 \cdot x_1}{x_2}$	00	01	11	10	=
	0	0	0	1	0	
	1	1	0	0	0	

și având formă canonică disjunctivă :

$$\overline{f}(x_0, x_1, x_2) = x_0 \cdot x_1 \cdot \overline{x_2} + \overline{x_0} \cdot \overline{x_1} \cdot x_2 ,$$

permite generarea funcțiilor logice  $f(x_0, x_1, x_2)$  și  $g(x_0, x_1, x_2)$  cu relațiile :

$$f(x_0, x_1, x_2) = \overline{\overline{f(x_0, x_1, x_2)}} = \overline{x_0 \cdot x_1 \cdot x_2 + x_0 \cdot x_1 \cdot x_2} \stackrel{not}{=} \overline{P_0 + P_1}$$

$g(x_0, x_1, x_2) = x_0 \cdot x_1 \cdot x_2 + (\overline{x_0 \cdot x_1 \cdot x_2} + x_0 \cdot x_1 \cdot x_2 + x_0 \cdot \overline{x_1 \cdot x_2}) = P_0 + x_2 f \stackrel{not}{=} P_0 + P_2$  ,  
programarea PLS realizându-se astfel ca în figura de mai jos :

