Proiectarea subsistemelor VLSI

Problema principală:
- găsirea unei celule de bază care poate implementa funcțiile principale ale sistemului

Celula de bază dictează:
- un set de criterii/reguli de sincronizare

Geometria celulei și criteriile de sincronizare dictează proiectarea circuitului de comandă care înconjoară aria de prelucrare constituită din celule de bază

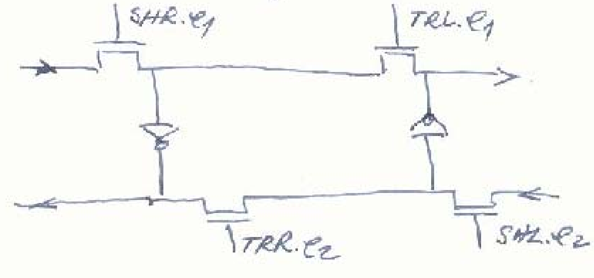
După amplasarea circuitelor de comandă rezultă un subsistem cu modul funcțional cu o interfață corespunzătoare privind:
- specificațiile funcționale;
- -"- geometrice
- -"- de I/E-date
- -"- de comandă și sincronizare.

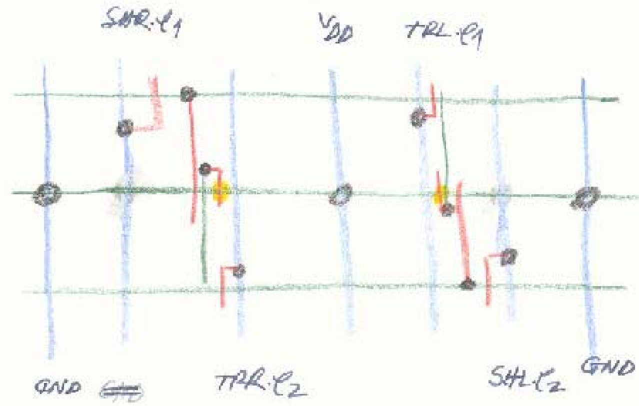
Setul de criterii de sincronizare are în vedere:
- intrările de date,
- ieșirile de date,
- intrările de comandă,

Exemplu: Proiectarea unei memorii de tip Stack LIFO

Celula de bază este capabilă să realizeze operațiile: Push-Pop-Nop, în condițiile unui ceas bifazic și al unui cod de operație.

Celula de bază





Rândul de sus va fi activat pe durata lui φ1, iar rândul de jos - pe durata lui φ2

Fluxul de comenzi este vertical, pe liniile de metalizare, iar fluxul de date este orizontal pe liniile de difuzie

STV3

Deplasare dreapta : PUSH

c_1 : SHR = 1 și c_2 : TRR = 1

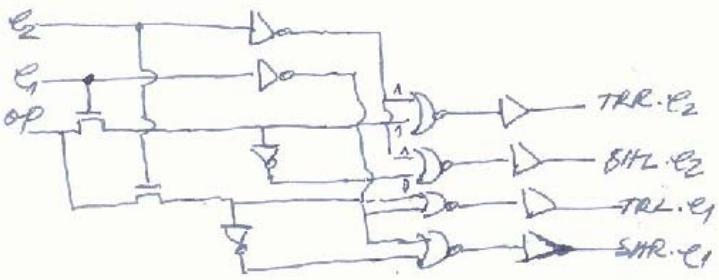
Deplasare stânga : POP

c_2 : SHL = 1 și c_1 : TRL = 1

Memorare : NOP

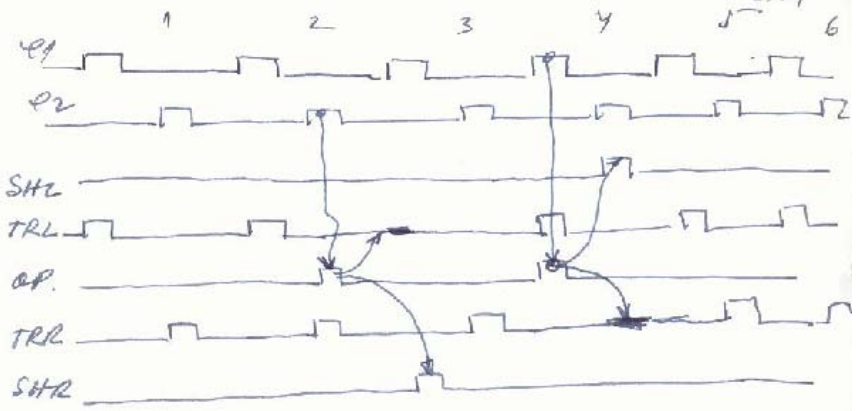
c_1 : TRR = 1 și c_2 : TRR = 1

Generarea semnalelor de comandă :
Schema este relativ simplă și poate fi generală pentru registre.
Se folosește un singur cod de operație de un bit, aplicat fie lui c_1 (op=1), fie lui c_2 (op=2).

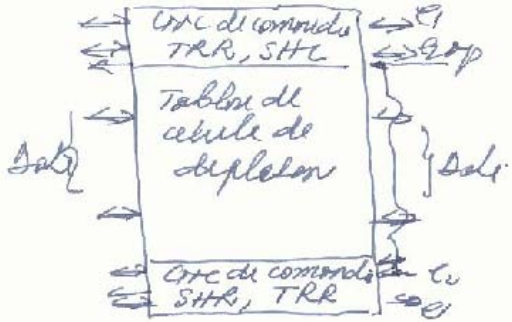


PUSH : op = 1, lui c_2
op = 0, lui c_1

STV4



Forma subsistemului



Proiectarea logicii combinaționale

Sunt trei categorii de probleme

1. Se cere o cantitate mică de logică

În cazul comenzii unui circuit format dintr-o singură celulă, care poate fi multiplicată.

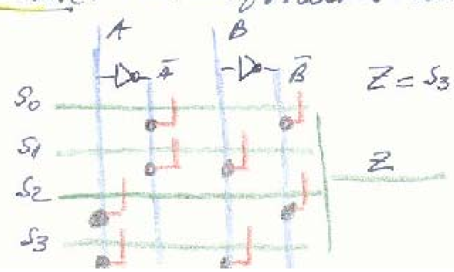
Se folosesc procedeele proiectării logice tradiționale cu circuite NAND, NOR

Prin inspecție, fără a mai folosi procedeele formale de minimizare.

Utilizarea porților statice nu este recomandată deoarece

- nu conduce la forme regulate
- nu asigură aria minimă
- puterea consumată mare
- nu realizează maximizarea numărului de funcții logice pe cantitatea de arie

Soluție: Rețele formate din tranzistoare de trecere



$Z = S_3 \cdot AB \vee S_2 A\bar{B} \vee S_1 \bar{A}B \vee S_0 \bar{A}\bar{B}$

2. A doua clasă de probleme.

Implementarea unei funcții logice complexe pentru care trebuie să se concepă metode de structurare topologică.

Ex: circuitul TALLY cu n intrări și n+1 ieșiri ($n = 3$)

$z_0 = \bar{x}_3 \bar{x}_2 \bar{x}_1$

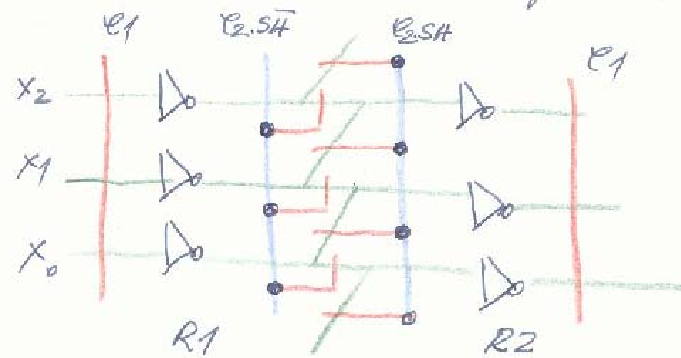
$z_1 = \bar{x}_3 \bar{x}_2 x_1 \vee \bar{x}_3 x_2 \bar{x}_1 \vee x_3 \bar{x}_2 \bar{x}_1$

$z_2 = \bar{x}_3 x_2 x_1 \vee x_3 \bar{x}_2 x_1 \vee x_3 x_2 \bar{x}_1$

$z_3 = x_3 x_2 x_1$

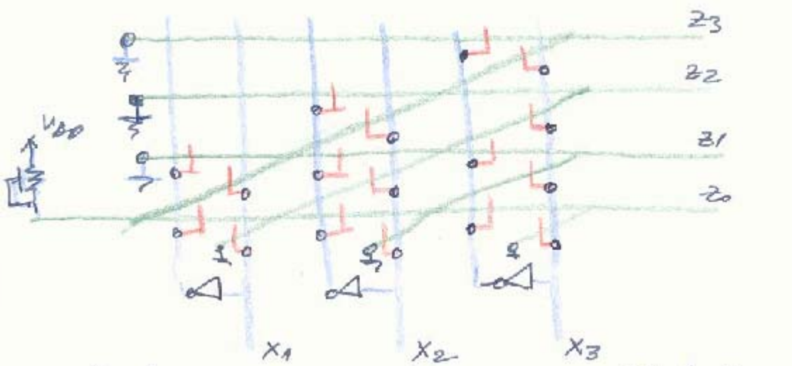
Implementarea conduce la structuri regulate sub aspect topologic.

Realizarea unui circuit de deplasare



$c_1 \mid R1 \leftarrow X$

$c_2 \mid R2 \leftarrow (R1 \,!\, SH(R1)) * (\overline{SH}, SH)$

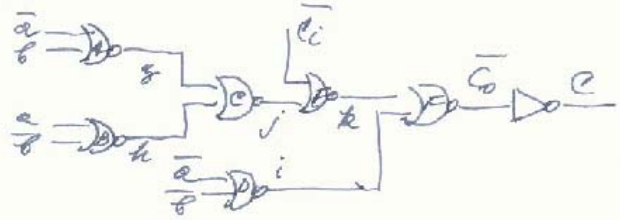


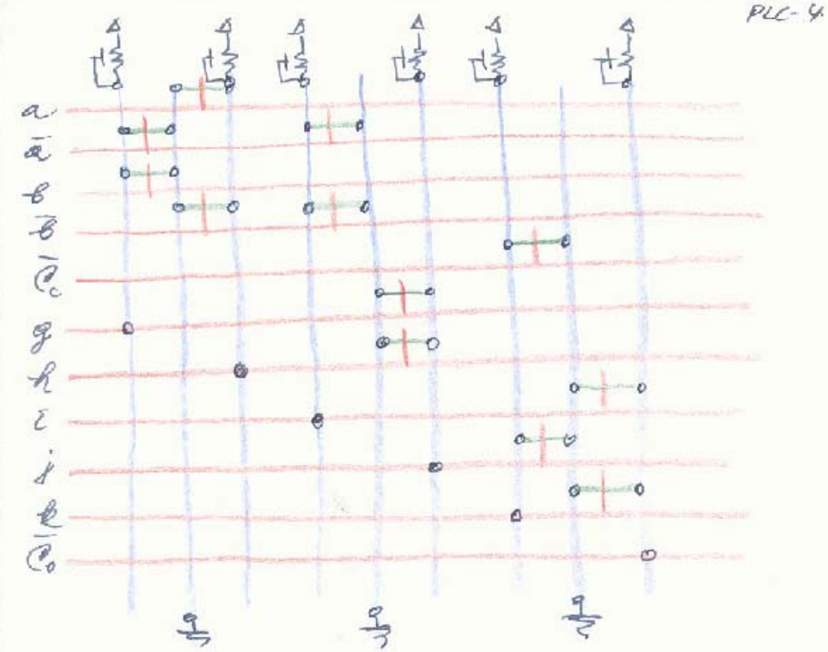
Pentru a manipula mai mult de 3 intrări se vor introduce buffere superetajate. Schema este generată direct din logica cu relee.

Retele Weinberger

Fie funcția

$C_o = ab \vee C_i(a\bar{b} \vee \bar{a}b)$ implementată cu circuite NOR





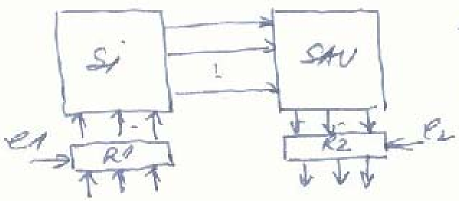
3. Retele logice programabile

O funcție logică complexă trebuie implementată fără a cunoaște aplicarea și dinainte într-o structură regulată. De exemplu logica rețelei registre.

Rețelele programabile permit aplicarea funcțiilor logice neregulate în structuri regulate. El. pot fi modificate substanțial fără a fi necesară modificarea necesară a modelului sau a matricei PLA

Utilizarea memoriilor pentru stocarea tabelelor de adevăr nu este convenabilă deoarece ocupă un volum mare.

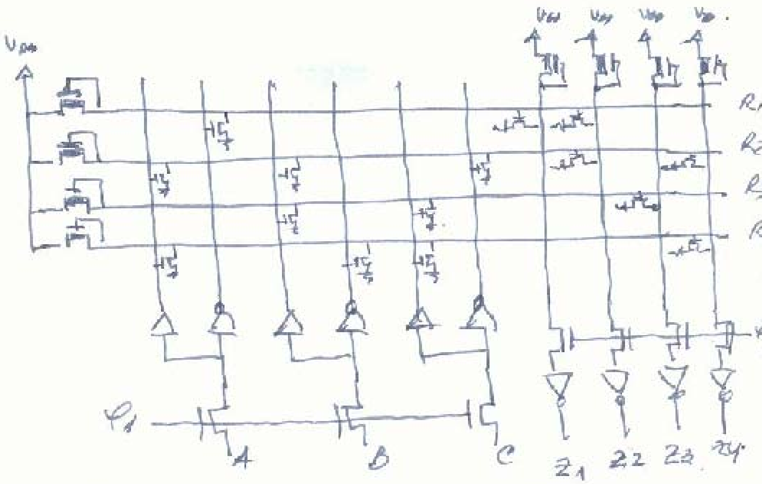
Schema bloc a unei PLA



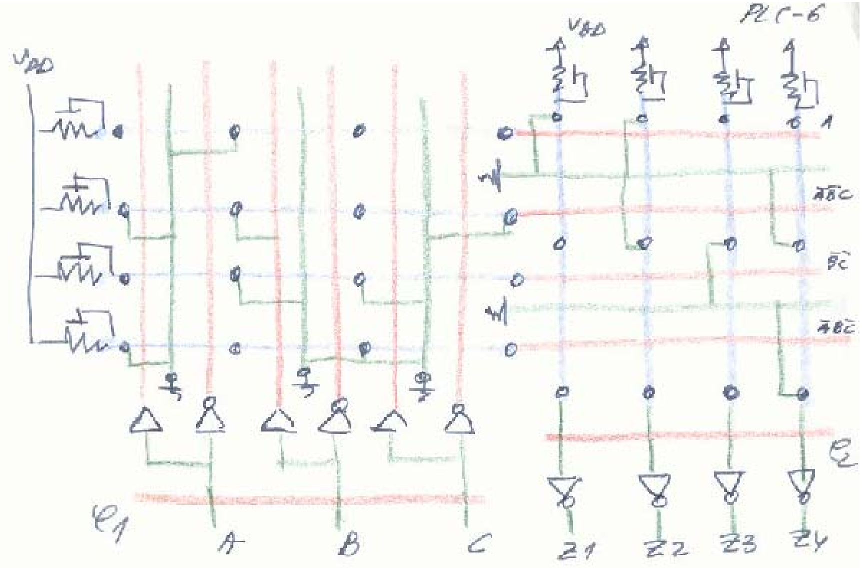
Ariile ȘI-SAU sunt realizate cu circuite NOR.

$Z1 = A$
$Z2 = A \vee \bar{A}\bar{B}C$
$Z3 = \bar{B}\bar{C}$
$Z4 = \bar{A}\bar{B}C \vee \bar{A}B\bar{C}$

reprezentați prin: $A, \bar{A}\bar{B}C, \bar{B}\bar{C}, \bar{A}B\bar{C}$



Se observă că
$R1 = A = \bar{\bar{A}}$
$R3 = \bar{B}\bar{C} = \overline{B \vee C}$
$R2 = \bar{A}\bar{B}C = \overline{A \vee B \vee \bar{C}}$
$R4 = \bar{A}B\bar{C} = \overline{A \vee \bar{B} \vee C}$



Aria logică programabilă necesită existența unei linii de elemente de circuit numai pentru cele produse care intră în forma canonică a funcției.

Dimensiunile și forma generală depind de următorii parametri:
- numărul de intrări,
- numărul termenilor de tip produs,
- numărul de ieșiri
- unitatea de lungime λ.

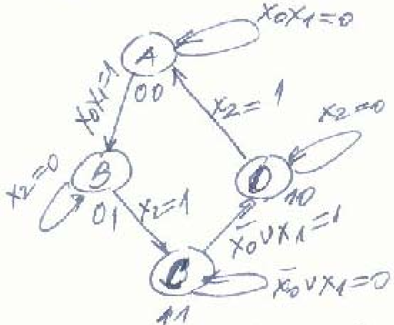
sunt suficient de scurte în raport cu perioada ceasului

Exemplu: Sinteza unui automat cu:

- trei intrări: x_0, x_1, x_2
- cinci ieșiri: $z_0, \dots, z_4$
- patru stări A, B, C, D codificate prin variabilele de stare

Y_0	Y_1	Stare
0	0	A
0	1	B
1	1	C
1	0	D

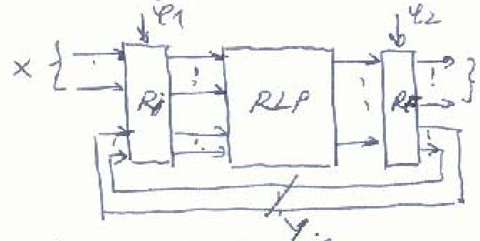
Automatul funcționează după următoarea organigramă / diagramă de tranziții



Realizarea tabelei de tranziție pentru stări și a tabelei pentru ieșiri

AUTOMATE FINITE (AF)

AF se realizează plasând o rețea logică programabilă între registre și asigurând o reacție de la ieșire spre intrare



Semnalele de intrare X împreună cu semnalele de stare Y sunt stocate în registrul de intrare Ri, pe durata lui φ_1. Semnalele se propagă prin RPL până la intrarea lui RE care le stochează pe durata lui φ_2

Trebuie avută în vedere, la calculul perioadei ceasului, întârzierea în RLP

Nr liniilor de reacție determină nr stărilor

Automatul este SINCRON

$$Y[(k+1)T] = f[Y(kT), X(kT)]$$
$$Z(kT) = g[Y(kT), X(kT)]$$

Dacă bucla de reacție ar fi permanent activă automatul s-ar comporta asincron

Mașinile asincrone funcționează corect dacă întârzierile prin circuitele de reacție

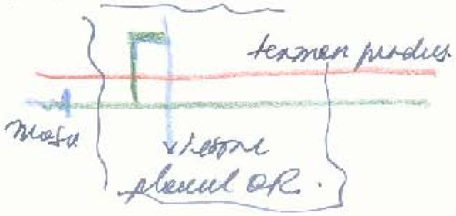
Semnale memorate cu Ri la C1 — Semnale memorate cu Re la C2

intrări	stări prezente	stări următoare	ieșiri	
$x_0 x_1 x_2$	$y_0 y_1$	$y_0 y_1$	$z_0 z_1 z_2 z_3 z_4$	
0 * *	0 0	0 0	0 0 0 1 0	R_1
* 0 *	0 0	0 0	0 0 0 1 0	R_2
1 1 *	0 0	0 1	1 0 0 1 0	R_3
* * 0	0 1	0 1	0 0 1 1 0	R_4
* * 1	0 1	1 1	1 0 1 1 0	R_5
1 0 *	1 1	1 1	0 1 0 0 0	R_6
0 * *	1 1	1 0	1 1 0 0 0	R_7
* 1 *	1 1	1 0	1 1 0 0 0	R_8
* * 0	1 0	1 0	0 1 0 0 1	R_9
* * 1	1 0	0 0	1 1 0 0 1	R_{10}

Reguli pentru programarea rețelei logice.

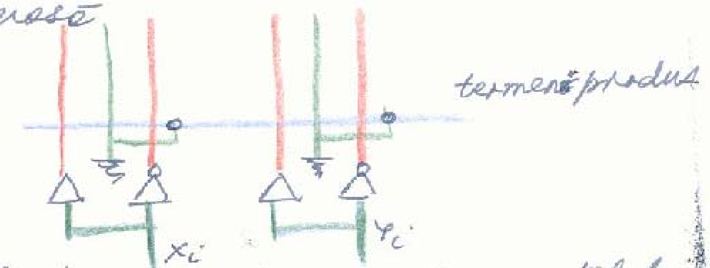
Aria OR

1. Pentru fiecare 1 logic din tabela pentru [illegible] starea următoare și pentru ieșiri se va trasa o cale de difuzie între linia corespunzătoare a ieșirii (verticală, metal), peste traseul de polisiliciu și traseul orizontal de difuzie conectat la masă

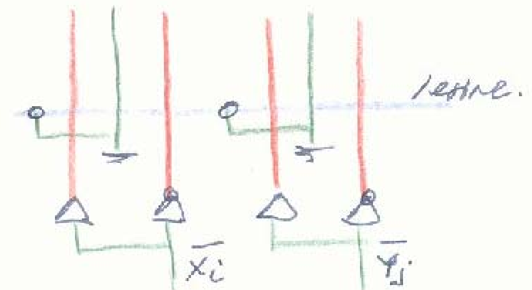


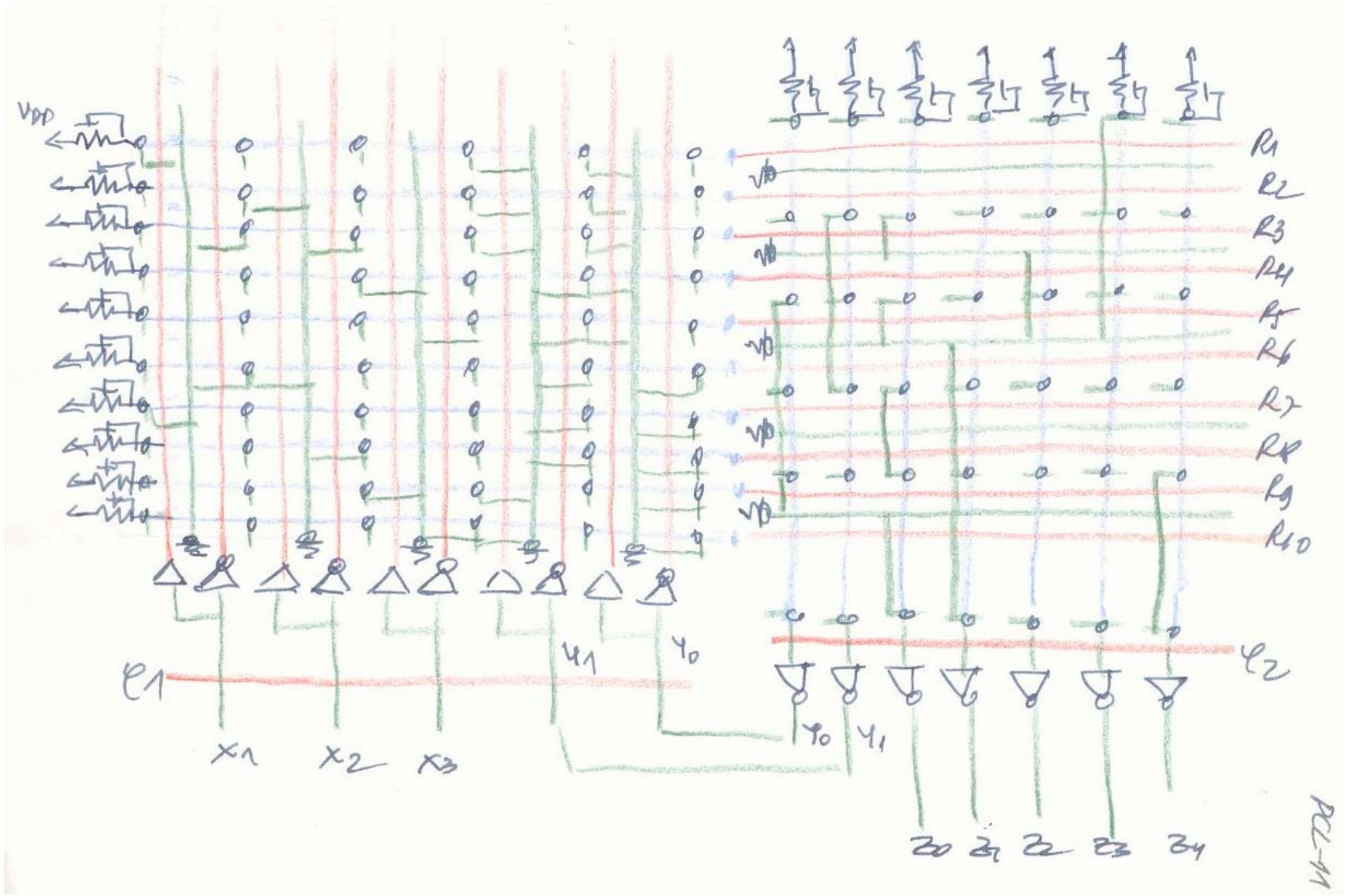
Pentru planul AND sunt două reguli

(2) Pentru fiecare 1 logic din tabela pentru intrări și starea prezentă se trasează o cale de difuzie de la linia corespunzătoare a termenului produs peste intrarea inversată la traseul vertical de difuzie conectat la masă



(3) Pentru fiecare termen egal cu 0 în tabela intrărilor și a stării prezente se va trasa o linie orizontală de difuzie de la termenul produs la masă intersectând traseul de polisiliciu corespunzător intrării directe




VDD
R1
R2
R3
R4
R5
R6
R7
R8
R9
R10
C1
C2
X1
X2
X3
Y0
Y1
Z0
Z1
Z2
Z3
Z4
PCL-11

Matricea secvențială se poate implementa PCL-12
pe o arie de $(150\lambda)^2$

Pentru $\lambda = 3\ \mu m$ (1978) (acum $\lambda \simeq 0{,}3\ \mu m$)
aria ar fi de $(450\ \mu m)^2 \simeq 0{,}002\ cm^2$

Matricea conține peste 150 tranzistori
și ocupă 1/25 din aria unui cip
integrat simplu – ($0{,}25\ cm^2$)

Se trece în prezent la 1/25.000 din
aria cipului simplu integrat