Ministerul Educaţiei, al Culturii și Cercetării al Republicii Moldova

Universitatea Tehnică a Moldovei

Departamentul Informatică și Ingineria Sistemelor

**RAPORT**

Proiect de An

Analiza si Sinteza Dispozitivelor Numerice

A efectuat:

st. gr. C-171 D. Melniciuc

A verificat:

dr., conf.univ. V. Sudacevschi

Chişinău 2019

**CUPRINS**

**Capitolul I**

1.1. Introducere 3

1.2. Elemente de analiză a circuitelor logice combinaţionale 4

1.3. Sinteza circuitelor logice combinaţionale 12

1.4. Minimizarea funcţiilor logice combinaţionale 14

1.4.1. Metoda minimizarii pe baza axiomelor si teoremelor algebrei booleene 15

**Capitolul II**

2 **Partea I** 19

2.1 Conditiile Proiectului de An 20

2.2 Date initiale 20

2.3 Inmultirea 20

2.4 Tabel de adevar 21

2.5 Tabelele Karnaugh 21

3 **Partea II** 22

3.1 Date initiale 22

3.2 Tabel de tranzactie 23

4 **Concluzii**  24

5 **Bibliografie**  25

6 **Anexe**  24

# 1.1. Introducere

[1]**Registre**

În arhitectura calculatoarelor, un registru de procesor este o cantitate mică de spațiu de stocare disponibilă pe unitatea centrală de procesare, spațiu al cărui conținut poate fi accesat mai rapid decât datele aflate în altă parte (de exemplu, în memoria principală).

Setul de registre din cadrul UCP păstrează temporar operanzii unei operații aritmetice sau logice, rezultatele intermediare și finale, sau adresele acestora. Utilizarea registrelor crește viteza de prelucrare, eliminând necesitatea accesului repetat la memorie. Ele reprezintă deci o memorie internă temporară foarte rapidă.

Unele registre pot avea funcții dedicate, altele se pot utiliza pentru orice operații, fiind registre generale. O parte din registre nu sunt accesibile prin program, fiind registre de lucru (de exemplu, registrul de instrucțiuni, care păstrează instrucțiunea curentă (cea care se execută la un moment dat).

**NUMĂRĂTOARE**

Numărătoarele – sunt circuite logice secvenţiale utilizate pentru contorizarea (numărarea

şi memorarea) impulsurilor aplicate la intrările acestora. Numărătoarele nu au intrări de

date, tranziţiile se efectuează după o anumită regulă într-o anumită ordine, fixate prin

construcţia numărătorului, în ritmul unui semnal de tact.

Numărătoarele se realizează cu circuite basculante bistabile (celule de numărare) care

stabilesc capacitate de numărare şi porţi logice care stabilesc modul corect în care

numărătorul îşi schimbă stările în cadrul procesului de numărare.

Numărătoarele binare se clasifică după următoarele criterii:

• ***După modul de conectare a bistabilelor de comandă:***

- *numărătoare asincrone* – bistabilele sunt conectare în serie, intrarea de tact CLK a unui bistabil este conectată la ieşirea Q a bistabilului anterior, bascularea unui bistabil se face numai după bascularea bistabilului anterior.

- *numărătoare sincrone* – bistabilele sunt conectate în paralel, intrările de tact CLK a tuturor bistabilelor sunt conectate împreună, bascularea tuturor bistabililor se face în acelaşi moment.

***• După sensul numărării:***

- *numărătoare directe* – fiecare impuls prezent la intrarea numărătorului creşte conţinutul acestuia cu o unitate (numără în sens crescător)

*- numărătoare inverse* – fiecare impuls prezent la intrarea numărătorului scade conţinutul acestuia cu o unitate (numără în sens descrescător)

*- numărătoare reversibile* – efectuează numărarea în ambele sensuri în funcţie de comanda primită din exterior

***• După codul de numărare:***

*- numărătoare binare* – m=2n

- *numărătoare decadice* – m=10

# 1.2. [2]Elemente de analiză a circuitelor logice combinaţionale

Circuitele logice combinaţionale sunt, de regulă, reprezentate grafic, cu ajutorul schemelor logice combinaţionale cu porţi logice similare cu cea prezentată în figura 1.2. O astfel de schemă este utilă în analiza funcţionării circuitului dar, un astfel de circuit poate fi reprezentat în scheme mai complexe cu ajutorul schemei bloc echivalente din figura 1.1, atunci când funcţionarea circuitului este cunoscută.

În general un circuit logic combinaţional are ***n+1*** intrări notate cu x0, x1, … , xn şi ***m+1*** ieşiri y0, y1, … ,ym. Intrările x0, x1, … , xn se aplică unor elemente logice ale căror ieşiri pot fi ieşiri ale reşelei sau intrări pentru alte elemente logice din reţea. În figura 1.2. se prezintă un circuit logic cu nouă intrări şi trei ieşiri.

Fiecare element logic din reţea corespunde unei porţi logice din circuitul de comutare modelat. În reţelele logice nu se admite legarea ieşirilor elementelor logice decât prin intermediul altor elemente logice. Atunci când circuitul de comutare modelat conţine porţi care au proprietatea funcţiilor logice cablate, se reprezintă simbolic în reţea elementul logic prin care sunt legate din punct de vedere funcţional ieşirile porţilor respective.

Semnalele aplicate la intrarile unui circuit logic, parcurg, în general, mai multe porţi până se obţin semnalele de ieşire. Acest lucru se reflectă în reţea prin numărul elementelor logice interpuse între intrările şi ieşirile reţelei. Maximumul numarului de elemenete logice aflate între intrările şi ieşirile unei reţele logice dă numărul de niveluri logice al reţelei. Numerotarea nivelurilor se face, în mod convenţional, de la ieşire spre intrare (figura 1.3). În reţelele de comutare combinaţionale sunt admise legături inverse, adică legarea ieşirii unui element logic la intrările elementelor logice precedente acestuia (fig. 1.4), cu condiţia să fie respectată definiţia 1.1. Trebuie menţionat însă că orice reţea cu elemente logice fară legături inverse este combinaţională, în schimb numai anumite reţele cu legături inverse satisfac această condiţie. Reţelele cu elemente logice fară legatură inversă se mai numesc grafuri booleene.

Analiza unui circuit trebuie să înceapă cu stabilirea tipului acestuia. Aşa cum s-a arătat mai sus, dacă circuitul nu are legături inverse atunci el este un circuit logic combinaţional. În cazul în care circuitul are legături inverse (figura 1.4.), atunci, pentru a putea spune că acesta este un circuit logic combinaţional, va trebui să analizăm dependenţa semnalelor de ieşire de cele de intrare şi să arătăm că ecuaţia 1.1. este respectată. Nu intotdeauna este foarte uşor să detectăm existenţa unei legături inverse într-un circuit mai ales dacă acesta este complicat. Din acest motiv se va prezenta în continuare o regulă prin care se poate determina existenţa legăturilor inverse la scheme oricât de complicate.

Existenţa unei legături inverse într-o reţea de comutare cu elemente logice se poate determina folosind următoarea regulă de numerotare a elementelor.

**Regula 1.1.**

1. Elementele reţelei ale căror intrări fac toate parte din mulţimea intrărilor reţelei, X={x1 x2,...,xn}, se numerotează, într-o ordine arbitrară, cu numerele 1 pina la ***k***, unde ***k*** este numărul elementelor ce îndeplinesc această condiţie.

1. Elementele reţelei ale căror intrări sunt fie intrări din mulţimea X, fie ieşiri ale elementelor numerotate, la punctul ***a*** respectiv ***b***, se numerotează în continuare cu numerele ***k + 1*** pâna la ***m***, unde ***m - k*** este numărul elementelor ce îndeplinesc această condiţie.
2. Dacă procedând în acest fel s-au putut numerota toate elementele reţelei, adică ***m*** este numărul de elemente logice din reţea, rezultă că reţeaua nu are legături inverse. În caz contrar în reţea există cel puţin o legatură inversă.

Prin procedura de mai sus se face de fapt o ordonare parţială a elementelor logice din reţea. Această ordonare nu este posibilă decât atunci când un anumit element nu mai poate primi număr de ordine, deoarece unele din intrările sale provin de la elemente nenumerotate care succed elementul dat, ceea ce înseamnă că există o legatură inversă.

Referindu-ne la schema din figura 1.3, regula de ordonare se aplică astfel : porţile ***1*** şi ***2*** se numerotează primele deoarece intrările acestor porţi fac parte din mulţimea intrărilor circuitului x1şi x2 respectiv x4 şi x5. Urmează apoi elementul care se numerotează cu ***3*** deoarece acesta are intrarea conectată la un element care a fost deja numerotat (poarta numerotată cu ***1***), similar acestei situaţii este cea a elementului care a fost numerotat cu ***5***, iar elementele numerotate cu ***4***, ***6*** şi ***9*** sunt conectate la elemente deja numerotate şi intrări ale circuitului (poarta ***1*** şi intrarea x3, poarta ***2*** şi intrarea x6 şi, respectiv poarta ***7*** şi intrarea x0). Elementele numerotate cu ***7***, ***8***, ***10***, ***11*** şi respectiv ***12*** pot şi ele numerotate succesiv, în această ordine, deoarece intrările acestor elemente sunt conectate la elemente deja numerotate.

Pentru acest exemplu m = 12, toate elementele circuitului au putut fi numerotate şi deci nu există nici o legătură inversă iar circuitul este combinaţional, adică starea ieşirilor la un moment dat nu depinde decât de starea intrărilor la acel moment (relaţia 1.1).

Aplicind regula 1.1, definită mai sus, pentru reţeaua din figura 1.4 se constată că nu este posibilă o ordonare parţială a tuturor elementelor reţelei şi prin urmare reţeaua are o legatură inversă.

Pentru reţelele la care se poate face o ordonare parţială a elementelor, deci care nu au legaturi inverse, se poate scrie expresia ieşirii fiecarui element logic, în ordinea numerotării acestora, ca o funcţie de comutare de variabilele de intrare ale reţelei, de unde rezultă că şi în expresia ieşirii intră numai variabilele de intrare ale reţelei. Funcţia de comutare a unei asemenea reţele depinde deci numai de variabilele de intrare, prin urmare reţeaua este combinaţională.

Din relaţia (1.12) rezultă că y0 iar din relaţia (1.13) rezultă că y1 depind numai de variabilele de intrare ale reţelei şi deci reteaua din figura 1.3 este o reţea combinaţională.

La reţelele care au legături inverse, expresia ieşirii se poate scrie numai dacă se introduc variabile de intrare secundare datorate legăturilor inverse.

Pentru ca o astfel de reţea să fie combinaţională trebuie ca funcţiile de ieşire a reţelei să depindă numai de variabilele de intrare principale adică să fie respectată ecuaţia 1.1. şi deci variabilele de intrare secundare sunt neesenţiale. Demonstrarea adestui lucru se poate face prin construirea tabelului de adevăr a funcţiei date, în care se trec toate combinaţiile posibile ale variabilelor de intrare principale cât şi a celor secundare. Dacă din acest tabel rezultă faptul că valorile funcţiei de ieşire a circuitului depind numai de valorile variabilelor principale iar valoile variabilelor secundare nu au nici o influenţă asupra ieşirii, atunci funcţia analizată este combinaţională.

Să considerăm circuitul din figura 1.4. Se constată faptul că numerotarea tuturor elementelor circuitului eşuează şi deci există cel puţin o legătură inversă.

Funcţia logică a ieşirii y0 este combinaţională pentru că toate elementele aferente acestei ieşiri au putut fi numerotate. Funcţia de ieşire y0 rezultă imediat :

*y*0 = *x*0 ⋅ *x*1 ⋅ *x*2 (1.14)

Pentru ieşirea y1 se construieşte tabelul de adevăr 1.1. în care s-au introdus variabilele principale x0, x1, x2 şi variabila secundară ***Xa***. După realizarea tabelului, prin inspectarea acestuia, ne putem da seama de modul în care o variabilă influenţează ieşirea. Pentru a putea observa mai uşor influenţa variabilei secundare ***Xa*** asupra ieşirii y1, se aşază alăturat valorile de ieşire pentru situaţia când variabila ***Xa*** ia valoarea zero şi valorile de ieşire pentru variabila secundară egală cu unu. Dacă variabila secundară nu influenţează ieşirea, atunci cele două coloane trebuie să fie identice.

TABELUL 1.1.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **x2** | **x1** | **x0** | **Xa** | **A=x0x1** | **B=x1x2** | **C=x0x2** | **D=x0x1x2** | **E=x1x2Xa** | **y1** |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 |
| **x2** | **x1** | **x0** | **Xa** | **A=x0x1** | **B=x1x2** | **C=x0x2** | **D=x0x1x2** | **E=x1x2Xa** | **y1** |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 |
| 1 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |

Se observă că în tabelul 1.1 coloana y1 rămâne neschimbată atunci când variabila ***Xa*** ia valoarea zero sau valoarea unu şi deci această variabilă secundară este neesenţială. Putem să scriem acum şi funcţia combinaţională de ieşire y1:

*y*1 = *x*0 *x*1 + *x*1*x*2 + *x*0 *x*2 + *x*0 *x*1*x*2 = *x*0 *x*1(1+ *x*2 )+ *x*1*x*2 + *x*0 *x*2 =

(1.15)

= *x*0 *x*1 + *x*1*x*2 + *x*0 *x*2

Având în vedere faptul că reţelele de comutare cu elemente logice, fară legatură inversă sunt reprezentative pentru reţelele combinaţionale cu elemente logice, în cele ce urmează se trateaza numai acestea.

*Reţelele de comutare combinaţionale cu elemente logice cu o singură ieşire, la care fiecare dintre intrările reţelei se aplică la un singur element logic, iar ieşirea unui element logic poate fi aplicată ca intrare la un singur element logic, se numeşte arbore boolean*. Arborele boolean este un caz particular al grafului boolean. Un exemplu de arbore boolean este dat in figura 1.5

x1

x4

U1A

7408

1

2

3

**b**

U2A

7432

1

2

3

**c**

**a**

y0

x2

U4A

7408

1

2

3

U3A

7432

1

2

3

x0

x3

**Figura 1.5. Reţea combinaţională sub formă de arbore boolean**

Pentru analiza unui arbore boolean se poate folosi o metodă formală care constă în găsirea unei acoperiri a mulţimii n-uplelor funcţiei logice pentru care aceasta ia valoarea unu.

Se va exemplifica această metodă pe circuitul din figura 1.5. construindu-se tabelul 1.2. Tabelul construit va conţine toate variabilele principale ale funcţiei (x0, x1, x2, x3, x4,), variabilele secundare (a, b, c) şi ieşirea circuitului (y0).

TABELUL 1.2.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **x4** | **x3** | **x2** | **x1** | **x0** | **a** | **b** | **c** | **y0** | **Observaţii** |
| 1 |  |  |  |  |  |  |  |  | 1 |  |
| 2 |  |  |  |  |  | 1 | 1 |  |  | y0 = a b |
| 3 |  |  |  | x | 1 |  | 1 |  |  | a = x0 + x1 |
|  |  |  | 1 | x |  | 1 |  |  |
| 4 |  |  | 1 | x | 1 |  |  | x |  | b = x2 + c |
|  |  | x | x | 1 |  |  | 1 |  |
|  |  | 1 | 1 | x |  |  | x |  |
|  |  | x | 1 | x |  |  | 1 |  |
| 5 | x | x | 1 | x | 1 |  |  |  |  | c = x3 x4 |
| 1 | 1 | x | x | 1 |  |  |  |  |
| x | x | 1 | 1 | x |  |  |  |  |
| 1 | 1 | x | 1 | x |  |  |  |  |

Tabelul 1.2 se completează astfel: pe rândul unu se trece cifra unu pe coloana y0 simbolizându-se asfel faptul că se vor determina combinaţiile variabilelor de intrare pentru care funcţia ia valoarea unu. Pe rândul doi se trec valorile variabilelor de pe nivelul zero care determină valoarea unu la ieşire. Acestea vor fi: unu pentru variabila secundară ***a*** şi unu pentru varibila secundară ***b***. Deoarece numai această combinaţie a variabilelor ***a*** şi ***b*** conduc la valoarea unu a funcţiei de ieşire y0, se trece la rândul trei unde se vor determina combinaţiile variabilelor principale pentru care variabila secundară ***a*** ia valoarea unu. Aşa cum se vede, sunt posibile 2 combinaţii pentru variabilele principale x0 şi x1: variabila x0 ia valoarea unu şi atunci x1 poate avea orice valoare (se notează cu x – indiferent) sau, cind x1 ia valoarea unu şi atunci x0 poate avea orice valoare (x – indiferent). Se trece acum la rândul patru care determină combinaţile variabilelor care duc la variabila ***b*** egala cu unu. Sunt doua posibilităţi: x2 egal cu unu şi variabila ***c*** poate avea în acest caz orice valoare sau ***c*** egal cu unu şi atunci x2 poate avea orice valoare. Aceste două posibilităţi se aplică celor două combinaţii determinate la rândul trei şi rezultă patru combinaţii posibile.

La rândul cinci se explicitează variabila ***c*** pentru care nu există decât doua posibilităţi: variabila ***c*** poate lua orice valoare şi atunci şi x3 şi x4 pot lua orice valoare sau variabila ***c*** ia valoarea unu li atunci x3 = x4 = 1.

În momentul in care au fost explicitate toate variabilele secundare, completarea tabelului a fost terminată şi sa găsit o acoperire a funcţiei pentru care aceasta ia valoarea unu. Funcţia poate fi scrisă ca o sumă de produse a combinaţiilor găsite, din care se elimină variabilele principale notate cu ***x*** (care pot avea orice valoare) şi se notează cu variabila directă variabila principală egală cu unu şi cu variabila negată cea egală cu zero în tabel. Pentru exemplul dat se obţine (forma normală disjunctivă):

*y*0 = *x*2 *x*0 + *x*4 *x*3*x*0 + *x*2 *x*1 + *x*4 *x*3*x*1 (1.16)

Printr-o metodă similară – căutându-se combinaţiile variabilelor principale pentru care funcţia este egală cu zero – se poate determina funcţia de ieşire sub forma unui produs de sume.

Pentru a exemplifica acest lucru vom considera schema din figura 1.6. Vom căuta de data aceasta o acoperire a n-uplelor funcţiei logice pentru care aceasta ia valoarea zero. Se construieşte tabelul 1.2 similar tabelului 1.1. în care se trec variabilele principale şi cele secundare pe coloane iar pe rânduri combinaţiile acestora pentru care funcţia ia valoarea zero.

X1

U7A

14093

1

2

3

Y0

U5A

14081

1

2

3

X5

**b**

**a**

U3A

14081

1

2

3

X4

X0

X2

U8A

14011

1

2

3

X3

**d**

U6A

14071

1

2

3

**c**

**Figura 1.6. Arbore boolean**

TABELUL 1.3.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **x5** | **x4** | **x3** | **x2** | **x1** | **x0** | **a** | **b** | **c** | **d** | **y0** | **Observaţii** |
| 1 |  |  |  |  |  |  |  |  |  |  | 0 |  |
| 2 |  |  |  |  |  |  |  |  | 1 | 1 |  | *y*0 = *cd* |

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 3 | 0 | x |  |  |  |  |  |  |  | 1 |  | *c* = *x*4 *x*5 |
| x | 0 |  |  |  |  |  |  |  | 1 |  |
| 4 | 0 | x |  |  |  |  | 1 | x |  |  |  | *d* = *a* + *b* |
| x | 0 |  |  |  |  | 1 | x |  |  |  |
| 0 | x |  |  |  |  | x | 1 |  |  |  |
| x | 0 |  |  |  |  | x | 1 |  |  |  |
| 5 | 0 | x | x | x |  |  | 1 |  |  |  |  | *b* = *x*3*x*2 |
| x | 0 | x | x |  |  | 1 |  |  |  |  |
| 0 | x | 1 | 1 |  |  | x |  |  |  |  |
| x | 0 | 1 | 1 |  |  | x |  |  |  |  |
| 6 | 0 | x | x | x | 1 | 1 |  |  |  |  |  | *a* = *x*0 *x*1 |
| x | 0 | x | x | 1 | 1 |  |  |  |  |  |
| 0 | x | 1 | 1 | x | x |  |  |  |  |  |
| x | 0 | 1 | 1 | x | x |  |  |  |  |  |

În rândul 6 completarea tabelului se încheie deoarece combinaţiile tuturor variabilelor principale au fost precizate. În acest moment putem scrie ecuaţia funcţiei de ieşire ca un produs de sume în care variabilele de intrare care iau valoarea zero sunt scrise direct iar cele care iau valoarea unu sunt scrise negat (ecuaţia 1.18, forma normală conjunctivă):

*y*0 = (*x*5 + *x*1 + *x*0 )(*x*4 + *x*1 + *x*0 )(*x*5 + *x*3 + *x*2 )(*x*4 + *x*3 + *x*2 ) (1.18)

Verificarea corectitudinii rezultatului se poate face prin intermediul unui program care să genereze toate combinaţiile posibile ale variabilelor principale, să calculeze valorile ecuaţiei (1.18) şi să le compare cu valorile obţinute pentru funcţia de ieşire a circuitului din figura 1.6 a cărui ecuaţie poate fi scrisă imediat conform porţilor logice şi conexiunilor arătate în figură:

*y*0 = (*x*0 *x*1 + *x*2 *x*3 )*x*4 *x*5 (1.19)

În continuare se prezintă un program scris în limbaj de programare PASCAL pentru compararea ecuaţiei (1.18) cu ecuaţia (1.19). S-a ales limbjul PASCAL datorită faptului că este foarte asemănător cu limbajul pseudocod şi programele pot fi foarte uşor înţelese chiar şi fără cunoaşterea limbajului.

În cazul general al reţelelor sub formă de graf boolean se aplică aceeaşi procedură cu singura deosebire că intrările, respectiv ieşirile elementelor logice care se aplică la mai multe elemente logice se consideră de mai multe ori (notate cu indici sau cu prim, secund, terţ ş.a.m.d.), iar apoi se impune condiţia ca valorile obţinute să fie egale, respectiv se elimină subcuburile corespunzătoare liniilor pentru care variabilele cu acelaşi nume nu au aceeaşi valoare.

Vom exemplifica acest lucru pentru graful boolean din figura 1.7. care va fi analizat în tabelul 1.4.

U1A

2

3

1

**a**

**''**

**x**

**''**

**2**

**a**

Y0

**a**

**'**

U3A

1

2

3

X1

**c**

X2

X0

U2A

2

3

1

X3

**b**

**x2'**

U4A

1

2

13

12

**Figura 1.7. Reţea de comutare sub formă de graf boolean**

În tabelul 1.4, datorită faptului că variabila principală x2 se aplică la intrarea a două elemente logice, vom considera suplimentar variabilele x2’ şi x2’’ şi, similar, pentru variabila secundară ***a*** care la rândul ei se aplică la intările altor două elemente logice, vom considera, de asemenea variabilele secundare ***a’*** şi ***a”***.

TABELUL 1.4.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | x3 | x2 | x'2 | x"2 | x1 | x0 | a | a' | a" | b | c | y0 | Observaţii |
| 1 |  |  |  |  |  |  |  |  |  |  |  | 1 |  |
| 2 |  |  |  |  |  |  |  |  |  | 1 | x |  | y0 = b + c |
|  |  |  |  |  |  |  |  |  | x | 1 |  |
| 3 | x |  |  | x |  |  |  |  | x | 1 |  |  | c = a” x2” x3 |
| 1 |  |  | 1 |  |  |  |  | 1 | x |  |  |
| 4 | x |  |  | x |  | 1 |  | 1 | x |  |  |  | b = x0 a’ |
| 1 |  |  | 1 |  | x |  | x | 1 |  |  |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 5 | x |  |  | x |  | 1 | 1 |  |  |  |  |  | a = a’ = a” |
| 1 |  |  | 1 |  | x | 1 |  |  |  |  |  |
| 6 | x |  | x | x | 1 | 1 |  |  |  |  |  |  | a = x1 + x2’ |
| x |  | 1 | x | x | 1 |  |  |  |  |  |  |
| 1 |  | x | 1 | 1 | x |  |  |  |  |  |  |
| 1 |  | 1 | 1 | x | x |  |  |  |  |  |  |
| 7 | x | x |  |  | 1 | 1 |  |  |  |  |  |  | x2 = x2’ = x2” |
| x | 1 |  |  | x | 1 |  |  |  |  |  |  |
| 1 | 1 |  |  | 1 | x |  |  |  |  |  |  |
| 1 | 1 |  |  | x | x |  |  |  |  |  |  |

Până la rândul cinci completarea tabelului se face în modul discutat deja mai sus. În rândul cinci trebuie alese valorile pentru care a = a’ = a”. Sunt două situaţii: când a’ = 1 şi a” = x şi când a’ = x şi a” = 1 adică sunt posibile urmatoarele cazuri a’ = 1şi a” = 0 sau a’ = 1 şi a” = 1, respectiv a’ = 0 şi a” = 1 şi a’ = 1 şi a” = 1 (deoarece simbolul ***x*** semnifică „orice valoare” adică zero sau unu). Evident, singura soluţie posibila este a = a’ = a” = 1. Acelaşi raţionament este făcut pe rândul şapte când pentru x2’ = x şi x2” = x rezultă x2 = x în celelalte situaţii, ca şi pe rândul cinci, soluţia este x2 = x2’ = x2” = 0. Putem acum să scriem acum forma normală disjunctivă a funcţiei de ieşire:

*y*0 = *x*1*x*0 + *x*2 *x*0 + *x*3*x*2 *x*1 + *x*3*x*2 = *x*1*x*0 + *x*2 *x*0 + *x*3*x*2 (*x*1 +1)=

(1.20)

= *x*1*x*0 + *x*2 *x*0 + *x*3*x*2

# 1.3. Sinteza circuitelor logice combinaţionale

Problema sintezei circuitelor logice combinaţionale constă în realizarea fizică a unei funcţii logice combinaţionale dată sub forma unui tabel de adevăr sau a unei funcţii canonice.

Datorită faptului că există mai multe expresii echivalente pentru o funcţie logică, problema sintezei circuitelor logice va avea şi ea mai multe soluţii. În practică interesează în general acea soluţie care corespunde circuitului realizabil cu cost cât mai mic, dar pot exista şi alte criterii cum ar fi cele de protecţie a informaţiei, siguranţă în funcţionare etc.

Din acest punct de vedere sunt importante procedeele de minimizare a funcţiilor logice combinaţionale care duc la găsirea celei mai avantajoase expresii pentru funcţia combinaţională dată.

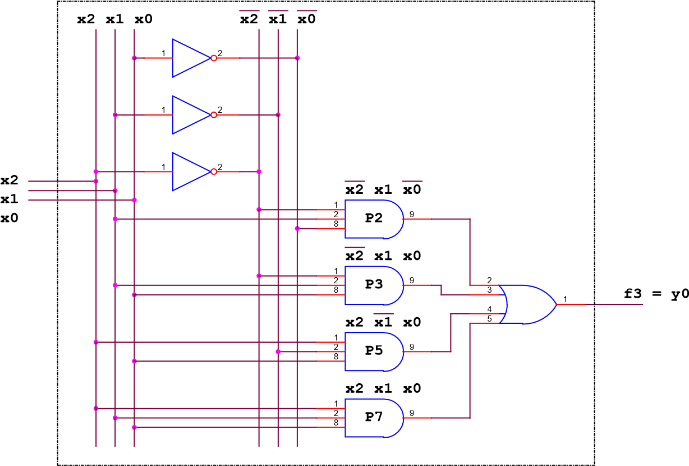
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | x2 | x1 | x0 | y0 |
| 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 |
| 2 | 0 | 1 | 0 | 1 |
| 3 | 0 | 1 | 1 | 1 |
| 4 | 1 | 0 | 0 | 0 |
| 5 | 1 | 0 | 1 | 1 |
| 6 | 1 | 1 | 0 | 0 |
| 7 | 1 | 1 | 1 | 1 |

TABELUL 1.5. De multe ori poate interesa realizarea funcţiei logice combinaţionale cu ajutorul unui singur tip de poartă logică. Aşa cum se ştie, există operaţii universale cu ajutorul cărora pot fi scrise in totalitate funcţiile logice. În continuarea se va arăta modul de realizare al unei funcţii logice al cărui tabel de adevăr este dat în tabelul 1.5. cu ajutorul porţii logice ŞI-NU (NAND) a cărei funcţie este o operaţie universală.

Pentru funcţia logică a cărei tabel de adevăr este dat în tabelul 1.5. putem scrie forma canonică normală disjunctivă:

|  |  |
| --- | --- |
| *f*3 = *P*2 + *P*3 + *P*5 + *P*7 = *x*2 *x*1 *x*0 + *x*2 *x*1*x*0 + *x*2 *x*1*x*0 + *x*2 *x*1*x*0 | (1.21) |

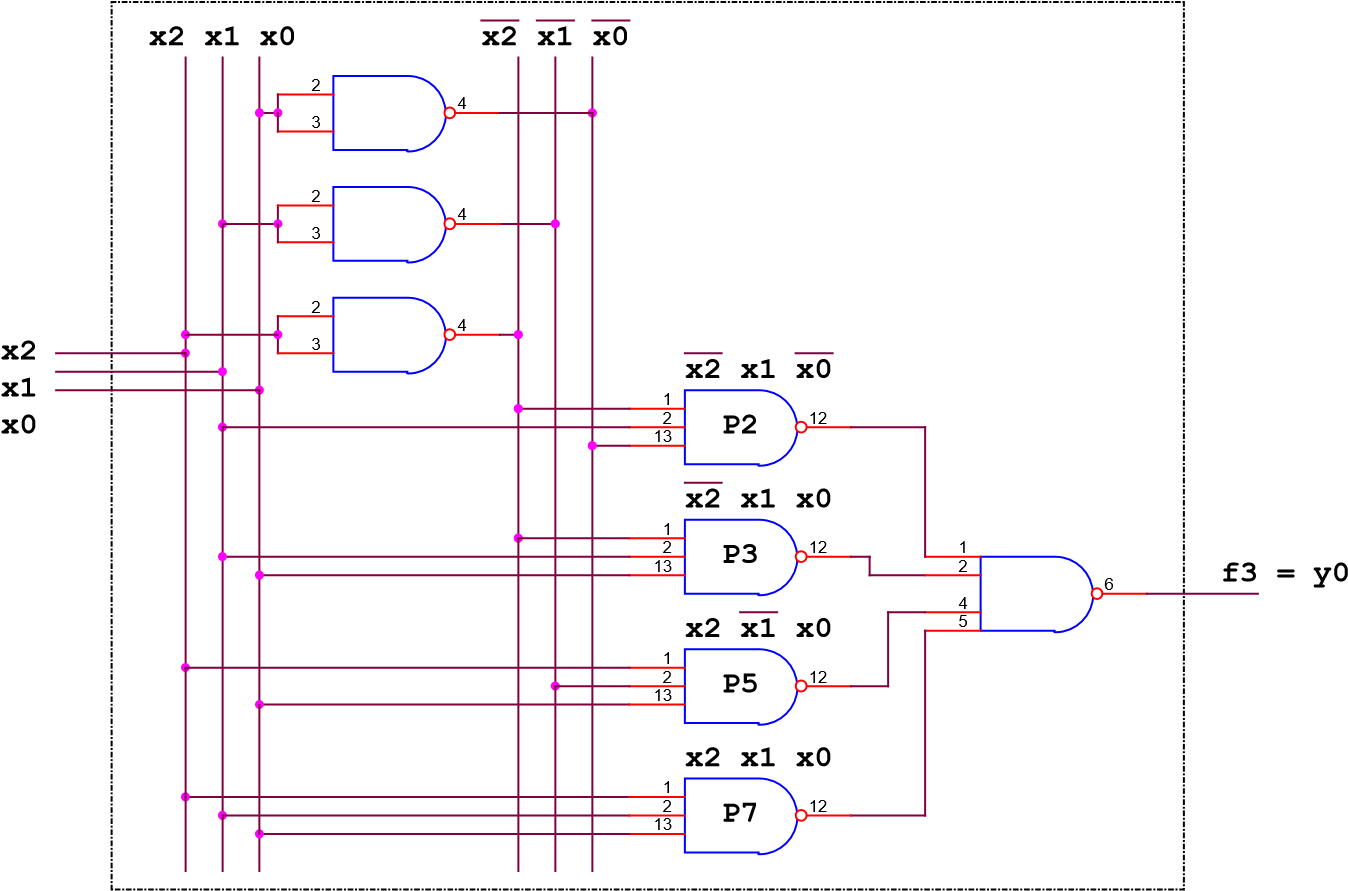
**Figura 1.8. Realizarea fizică a funcţiei (3.21) cu ajutorul porţilor logice.**



Ecuaţie, care, aşa cum se vede, poate fi realizată fizic numai cu porţi ŞI-NU. Pentru ca întreaga realizare fizică să conţină numai porţi logice ŞI-NU, inversoarele folosite în figura 1.8. pentru obţinerea variabilelor de intrare negate, se vor înlocui cu porţi ŞI-NU cu intrările scurtcircuitate. Conform tabelului de adevăr al acestei funcţii, dacă pe toate intrările porţii se aplică aceeaşi valoare atunci se obţine un circuit inversor.

Rezultatul acestor transformări este prezentat în figura 1.9.

Aşa cum rezultă din figura 1.9. pentru implementarea fizică a unei funcţii logice combinaţionale, pe baza funcţiilor canonice, este necesar un număr relativ ridicat de elemente logice. Din acest motiv este important să se găsească procedee de simplificare a ecuaţiilor ce descriu funcţiile de ieşire ale circuitelor logice combinaţionale în vederea reducerii numărului de porţi logice.



**Figura 1.9. Funcţie logică combinaţională realizată numai cu porţi ŞI-NU**

# 1.4. Minimizarea funcţiilor logice combinaţionale

Se vor prezenta în continuare principalele metode de minimizare a funcţiilor logice combinaţionale care, pe baza formelor canonice normale disjunctive sau normale conjunctive, permit obţinerea unor forme mai simple a funcţiei, realizabile la un cost mai scăzut (cu un număr mai mic de porţi logice mai simple).

## 1.4.1. Metoda minimizarii pe baza axiomelor si teoremelor algebrei booleene

Folosind axiomele şi teoremele algebrei booleene (paragraful 1.2), o funcţie booleană dată sub forma canonică disjunctivă sau sub forma canonică conjunctivă poate fi scrisă în cazul general sub o altă formă cu număr mai mic de termeni, respectiv factori elementari căreia îi corespunde o reţea cu cost mai redus. Această metodă de minimizare a funcţiei de comutare necesită din partea proiectantului multă îndemânare, ingeniozitate şi experienţă, motiv pentru care nu poate fi aplicată cu succes decât după o practică îndelungată în proiectarea circuitelor de comutare. Unul dintre principalele dezavantaje ale metodei îl constituie faptul că obţinându-se prin calcule o anumita formă a funcţiei nu se poate stabili cu uşurinţă dacă este forma minimă sau se mai poate simplifica.

Pentru exemplificare vom considera un exemplu foarte simplu:

*y*0 = *x*2 *x*0 (*x*0 + *x*1 )+ *x*2 *x*1*x*0 + *x*2 *x*1*x*0 (1.24)

aplicând teorema absorbţiei se obţine:

*y*0 = *x*2 *x*0 + *x*2 *x*1*x*0 + *x*2 *x*1*x*0 (1.25)

grupăm acum ultimii doi termeni:

*y*0 = *x*2 *x*0 + *x*2 *x*0 (*x*1 + *x*1 ) (1.26)

şi aplicăm axioma existenţei complementului:

0 2 0 2 *x*0 = *x*0 (*x*2 + *x*2 )=*x*0(1.27) *y* = *x x* + *x*

Acest exemplu simplu indică faptul că prelucrarea unei funcţii poate duce la simplificarea considerabilă a acesteia. În activitatea de simplificare a funcţiei logice unii termeni pot fi multiplicaţi, bazându-ne pe teorema idempotenţei, în scopul grupării convenabile a acestora şi reducerea unor variabile.

## 1.4.2. Metoda diagramelor Karnaugh

Folosirea unei forme speciale a diagramelor Venn, în scopul simplificarii cu ajutorul acestora a funcţiilor logice combinaţionale a fost sugerată pentru prima oară de către E. W. Veitch. La scurt timp M. Karnaugh propune de asemenea o formă modificată a diagramelor Venn cu acelaşi scop. Astfel au rezultat diagramele denumite

diagrame Karnaugh. Diagramele Karnaugh sunt folosite curent pentru reprezentarea funcţiilor booleene cu numar relativ mic de variabile.

Aceste diagrame sunt utile pentru minimizarea funcţiilor booleene deoarece permit evidenţierea cu uşurinţă a unor identităţi de forma:

*x* + *xy* = *x*

*xy* + *xy* = *x* (1.28) *x* + *xy* = *x* + *y*

În general, o diagramă Karnaugh pentru o funcţie booleana de ***n*** variabile se desenează sub forma unui pătrat sau dreptunghi, împărţit în 2n compartimente, fiecare compartiment fiind rezervat unui termen canonic al funcţiei, respectiv unuia dintre cele 2n n-uple ale funcţiei sau vârfuri ale cubului n-dimensional din reprezentarea geometrică a funcţiei.

În acest fel, o diagramă Karnaugh va fi reprezentată printr-un tabel cu ***m*** linii şi ***p*** coloane care îndeplinesc condiţia ***m*** x ***p*** = 2n iar ***m*** + ***p*** = ***n***. Capetele de tabel vor conţine combinaţiile posibile pentru variabilele funcţiei scrise în cod Gray.

În continuare vom exemplifica modul de realizare a diagramei Karnaugh pentru o funcţie de patru variabile.

Vom considera tabelul de adevăr al funcţiei tabelul 1.6 (în acest tabel valorile funcţiei sunt alese la întâmplare).

TABELUL 1.6.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | x3 | x2 | x1 | x0 | y0 |  | x3 | x2 | x1 | x0 | y0 |  | x3 | x2 | x1 | x0 | y0 |  | x3 | x2 | x1 | x0 | y0 |
| 0 | 0 | 0 | 0 | 0 | f(0)=0 | 4 | 0 | 1 | 0 | 0 | f(4)=1 | 8 | 1 | 0 | 0 | 0 | f(8)=0 | 12 | 1 | 1 | 0 | 0 | f(12)=1 |
| 1 | 0 | 0 | 0 | 1 | f(1)=1 | 5 | 0 | 1 | 0 | 1 | f(5)=0 | 9 | 1 | 0 | 0 | 1 | f(9)=0 | 13 | 1 | 1 | 0 | 1 | f(13)=0 |
| 2 | 0 | 0 | 1 | 0 | f(2)=1 | 6 | 0 | 1 | 1 | 0 | f(6)=1 | 10 | 1 | 0 | 1 | 0 | f(10)=1 | 14 | 1 | 1 | 1 | 0 | f(14)=1 |
| 3 | 0 | 0 | 1 | 1 | f(3)=0 | 7 | 0 | 1 | 1 | 1 | f(7)=0 | 11 | 1 | 0 | 1 | 1 | f(11)=1 | 15 | 1 | 1 | 1 | 1 | f(15)=0 |

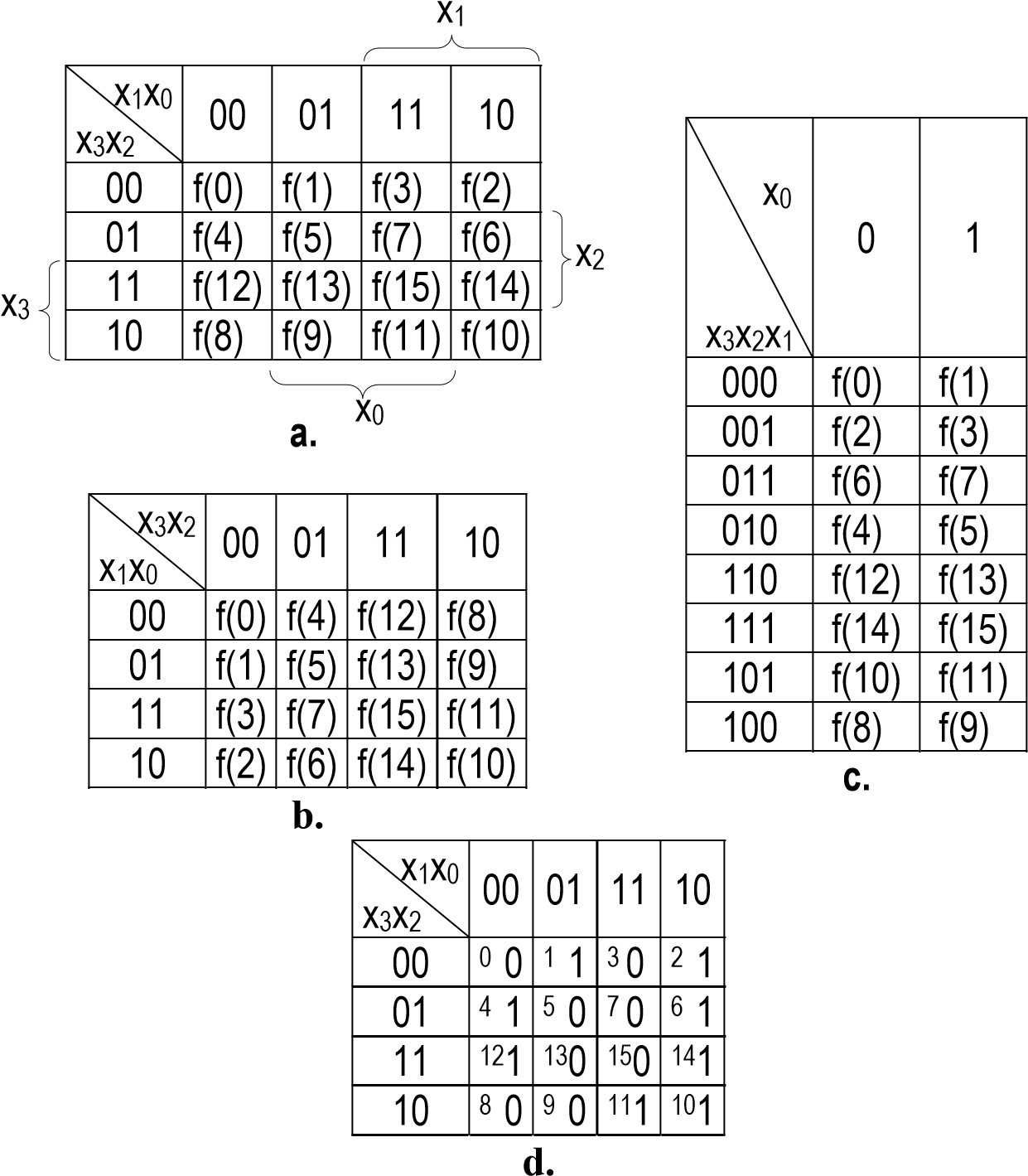
Conform acestui, tabel forma normală disjunctivă a funcţiei este:

f4 = P1 + P2 + P4 + P6 + P10 + P11 + P12 + P14 (1.29)

Tabelele Karnaugh posibil de construit pentru funcţia (1.29) sunt prezentete în figura 1.10.

Este evident faptul că există şi alte posibilităţi de aşezare a variabilelor pe liniile şi coloanele tabelului în afară de cele arătate în figura 1.10.

Tabelul se completează cu valorile funcţiei (zero sau unu) corespunzătoare combinatiei variabilelor de intrare de pe linia şi coloana respectivă (figura 1.10 d.).



**Figura 1.10. a. b. c. Posibilităţi de realizare a tabelului Karnaugh pentru o funcţie de patru variabile.**

**d. Tabelul Karnaugh pentru funcţia descrisă de ecuaţia (1.29).**

Codul Gray (codul binar reflectat), cod ce poartă numele celui care l-a imaginat, a fost construit pe principiul ca două secvenţe vecine să difere într-o singură poziţie binară. Secvenţele codului Gray pot fi deduse din cele ale codului binar pe baza următoarelor relaţii:

g0 = b0 ⊕ b1, g1 = b1 ⊕ b2,

(1.30)

g2 = b2 ⊕ b3,

g3 = b3 ,

unde g0, g1, g2, g3, sunt poziţiile unei secvenţe Gray scrise de la dreapta spre stânga, iar b0, b1, b2, b3, sunt poziţiile codului binar scrise în ordinea ponderilor.

Pentru exemplificare să considerăm cifra 6 scrisă în cod binar:

b3 b2 b1 b0

0 1 1 0

Pe baza relaţiilor (1.30) vom deduce succesiv valorile cifrelor binare din secvenţa de cod Gray:

g0 = b0 ⊕ b1 = 0 ⊕ 1 = 1, g1 = b1 ⊕ b2 = 1 ⊕ 1 = 0, g2 = b2 ⊕ b3 = 1 ⊕ 0 = 1, g3 = b3 = 0.

Prin urmare, secvenţa Gray corespunzătoare cifrei 6 va fi:

g3 g2 g1 g0

0 1 0 1

Putem deduce de asemenea şi relaţiile de transformare din cod Gray în cod binar. Acestea sunt:

b0 = g0 ⊕ g1⊕ g2 ⊕ g3, b1 = g1⊕ g2 ⊕ g3,

(1.31)

b2 = g2 ⊕ g3,

b3 = g3,

Valorile cifrelor binare sunt uşor de calculat dacă facem observaţia că 1 ⊕ 1 = 0.

Rezultă că este suficient să numărăm cifrele binare de unu din relaţia de calcul. Dacă acestea sunt în număr par, atunci rezultatul este zero, iar daca sunt în număr impar rezultatul este unu.

Pentru exemplificare vom considera numărul în cod Gray corespunzător cifrei 6 verificând dacă se obţine acelaşi rezultat:

g1 g2 g1 g0

0 1 0 1

Folosind relaţiile de mai sus se obţine:

b0 = g0 ⊕ g1⊕ g2 ⊕ g3 = 1 ⊕ 0⊕ 1 ⊕ 0 = 0, b1 = g1⊕ g2 ⊕ g3 = 0 ⊕ 1 ⊕ 0 = 1, b2 = g2 ⊕ g3 = 1 ⊕ 0 = 1, b3 = g3 = 0.

Deci numărul binar obţinut este:

b3 b2 b1 b0

0 1 1 0

adică cifra 6 de la care am pornit.

0 diagramă Karnaugh este astfel organizată încât două compartimente vecine, pe o linie sau pe o coloana, corespund la doi termeni canonici care diferă numai printr-o singură variabilă, care apare într-unul dintre ei negată, iar în celalălt directă, respectiv la două n-uple adiacente. Se consideră vecine şi compartimentele aflate la capetele opuse ale unei linii, respectiv coloane (marginile diagramei).

Pentru a putea reprezenta uşor funcţii date în mod convenţional prin indicii termenilor canonici, se poate nota fiecare compartiment cu indicele termenului canonic corespondent, ţinând cont de o anumită ordine a variabilelor.

De obicei, diagramele pentru mai mult de patru variabile se construiesc din diagrame de patru variabile, considerate diagrame elementare. Se pot construi însă diagrame Karnaugh pentru numar mai mare de variabile şi considerând ca diagrame elementare diagramele de trei variabile. În cazul diagramelor pentru mai mult de patru variabile, două compartimente se consideră vecine şi atunci când ocupă aceeaşi poziţie în două diagrame elementare vecine, adică în două diagrame elementare alăturate sau aflate la extremitaţi pe o aceeaşi linie sau coloană.

**Concluzie.** Folosind metoda diagrameolor Karnaugh se poate obţine forma minimă disjunctivă sau conjunctivă astfel:

1. se construieşte diagrama ;
2. se caută implicanţii primi ;
3. se determină implicanţii primi esenţiali ;
4. termenii canonici care nu sunt incluşi în implicanţii primi esenţiali se acoperă cu un număr cât mai mic de implicanţi primi ;
5. forma minimă normală disjunctivă sau conjunctivă va conţine toţi implicanţii primi esenţiali şi implicanţii primi neesenţiali ce conţin termeni canonici ce nu sunt incluşi în implicanţii primi esenţiali.

Aşa cum s-a arătat, metoda diagramelor Karnaugh, deşi este o metodă foarte simplă şi eficientă de minimizare a funcţiilor logice, ea nu poate fi aplicată decât funcţiilor cu un număr redus de variabile (cel mult şapte-opt).

Pentru minimizarea funcţiilor cu un număr mai mare de variabile se recurge la alte metode, algebrice sau tabelare, una dintre acestea fiind descrisă în paragraful următor.

##### **Partea II**

**2.1 Conditiile Proiectului de An**

1. Varianta 26
2. Inmultirea, metoda 2
3. Virgula mobila
4. Numar cu semn
5. 12 biti-mantisa
6. Registru-Numarator direct modulo 22
7. Deplasare reversibila

**2.2 Date initiale**

A = -225.72 A = 11100001.101 = 1.011100001101 \* 2^9

B = 361.78 B = 101101001.110 = 0.101101001110 \* 2^9

mA = 1.100011110011 |mA| = 011100001101

mB = 0.101101001110 |mB| = mB

**2.3 Inmultirea**

011100001101

101101001110

011100001101

011100001101

10001101000001

011100001101

100110110001111

011100001101

10011110101001001

011100001101

10011111000101010101

011100001101

100111110100110110111

011100001101

0.10011111010110111101001 = 81591.281

-225.72 \* 361.78 =-81660.9816

**2.4 Tabel de Adevar**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | T | T + 1 | J3K3 | J2K2 | J1K2 | J0K0 |
| x1x2 | Q3Q2Q1Q0 | Q3Q2Q1Q0 |
| dreapta | 0 0 | Q3Q2Q1Q0 |  | Q3 | Q3 | Q2 | Q1 |
| stinga | 0 1 | Q3Q2Q1Q0 |  | Q3 | Q2 | Q0 | 0 1 |
| CT | 11 | 0 0 0 0 | 0 0 0 1 | 0 \* | 0 \* | 0 \* | 1 \* |
| 0 0 0 1 | 0 0 1 0 | 0 \* | 0 \* | 1 \* | \* 1 |
| 0 0 1 0 | 0 0 1 1 | 0 \* | 0 \* | \* 0 | 1 \* |
| 0 0 1 1 | 0 1 0 0 | 0 \* | 1 \* | \* 1 | \* 1 |
| 0 1 0 0 | 0 1 0 1 | 0 \* | \* 0 | 0 \* | 1 \* |
| 0 1 0 1 | 0 1 1 0 | 0 \* | \* 0 | 1 \* | \* 1 |
| 0 1 1 0 | 0 1 1 1 | 0 \* | \* 0 | \* 0 | 1 \* |
| 0 1 1 1 | 1 0 0 0 | 1 \* | \* 1 | \* 1 | \* 1 |
| 1 0 0 0 | 1 0 0 1 | \* 0 | 0 \* | 0 \* | 1 \* |
| 1 0 0 1 | 1 0 1 0 | \* 0 | 0 \* | 1 \* | \* 1 |
| 1 0 1 0 | 1 0 1 1 | \* 0 | 0 \* | \* 0 | 1 \* |
| 1 0 1 1 | 1 1 0 0 | \* 0 | 1 \* | \* 1 | \* 1 |
| 1 1 0 0 | 1 1 0 1 | \* 0 | \* 0 | 0 \* | 1 \* |
| 1 1 0 1 | 0 0 0 0 | \* 1 | \* 1 | 0 \* | \* 1 |
| 1 1 1 0 | 0 0 0 0 | \* 1 | \* 1 | \* 1 | 0 \* |
| 1 1 1 1 | 0 0 0 0 | \* 1 | \* 1 | \* 1 | \* 1 |

**2.5 Tabelele Karnaugh**

***J3 K3***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 0 0 | 0 1 | 1 1 | 1 0 |
| 0 0 |  |  | \* | \* |
| 0 1 |  |  | \* | \* |
| 1 1 |  | 1 | \* | \* |
| 1 0 |  |  | \* | \* |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 0 0 | 0 1 | 1 1 | 1 0 |
| 0 0 | \* | \* |  |  |
| 0 1 | \* | \* | 1 |  |
| 1 1 | \* | \* | 1 |  |
| 1 0 | \* | \* | 1 |  |

J3 = Q1Q2Q0x1x2 + Q3 Q3 + Q2x2 K3 = (Q2Q0 + Q2Q1)x1x2 + + x2

***J2 K2***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 0 0 | 0 1 | 1 1 | 1 0 |
| 0 0 |  | \* | \* |  |
| 0 1 |  | \* | \* |  |
| 1 1 | 1 | \* | \* | 1 |
| 1 0 |  | \* | \* |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 0 0 | 0 1 | 1 1 | 1 0 |
| 0 0 | \* |  |  | \* |
| 0 1 | \* |  | 1 | \* |
| 1 1 | \* | 1 | 1 | \* |
| 1 0 | \* |  | 1 | \* |

J2 = Q1Q0x1x2 + Q3 + Q1 K2 = (Q3Q0 + Q1Q0 + Q3Q1) + + x2

***J1 K1***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 0 0 | 0 1 | 1 1 | 1 0 |
| 0 0 |  |  |  |  |
| 0 1 | 1 | 1 |  | 1 |
| 1 1 | \* | \* | \* | \* |
| 1 0 | \* | \* | \* | \* |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 0 0 | 0 1 | 1 1 | 1 0 |
| 0 0 | \* | \* | \* | \* |
| 0 1 | \* | \* | \* | \* |
| 1 1 | 1 | 1 | 1 | 1 |
| 1 0 |  |  | 1 |  |

J1 = (Q0 + Q0)x1x2 + Q2 + Q0 x2 K1 = (Q3Q2+Q0)x1x2 + + x2

***J0 K0***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 0 0 | 0 1 | 1 1 | 1 0 |
| 0 0 | 1 | 1 | 1 | 1 |
| 0 1 | \* | \* | \* | \* |
| 1 1 | \* | \* | \* | \* |
| 1 0 | 1 | 1 |  | 1 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 0 0 | 0 1 | 1 1 | 1 0 |
| 0 0 | \* | \* | \* | \* |
| 0 1 | 1 | 1 | 1 | 1 |
| 1 1 | 1 | 1 | 1 | 1 |
| 1 0 | \* | \* | \* | \* |

J0 = + + Q3)x1x2 + Q1 K0 = x1x2 + +

**Partea II**

**3.1 Date initiale**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | | Q2Q3 | | | |
| 0 0 | 0 1 | 1 1 | 1 0 |
| Q1Q0 | 0 0 | 0 | 7 | 8 |  |
| 0 1 | 1 | 6 | 9 |  |
| 1 1 | 2 | 5 |  |  |
| 1 0 | 3 | 4 |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Q3 | Q2 | Q1 | Q0 |
| S0 | 0 | 0 | 0 | 0 |
| S1 | 0 | 0 | 0 | 1 |
| S2 | 0 | 0 | 1 | 1 |
| S3 | 0 | 0 | 1 | 0 |
| S4 | 0 | 1 | 1 | 0 |
| S5 | 0 | 1 | 1 | 1 |
| S6 | 0 | 1 | 0 | 1 |
| S7 | 0 | 1 | 0 | 0 |
| S8 | 1 | 1 | 0 | 0 |
| S9 | 1 | 1 | 0 | 1 |

**3.2 Tabelu de tranzictii**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Si | Q3Q2Q1Q0 | Si | Q3Q2Q1Q0 | x | y | J3K2 | J2K2 | J1K1 | J0K0 |
| s0 | 0000 | s0 | 0000 | x1 | - |  |  |  |  |
| 0000 | s1 | 0001 | x1 | 1 -8 |  |  |  | J0 |
| s1 | 0001 | s2 | 0011 | 1 | 9 |  |  | J1 |  |
| s2 | 0011 | s3 | 0010 | x2 | 10 |  |  |  | K0 |
| 0011 | s4 | 0110 | x2x3 | 11 |  | J2 |  | K0 |
| 0011 | s5 | 0111 | x2x3x4 | 12 |  | J2 |  |  |
| s3 | 0010 | s4 | 0110 | x3 | 11 |  | J2 |  |  |
| 0010 | s5 | 0111 | x3x4 | 12 |  | J2 |  | J0 |
| 0010 | s6 | 0101 | x3x4 | 13-15 |  | J2 | K1 | J0 |
| s4 | 0110 | s5 | 0111 | x4 | 12 |  |  |  | J0 |
| 0110 | s6 | 0101 | x4 | 13-15 |  |  | K1 | J0 |
| s5 | 0111 | s6 | 0101 | 1 | 13-15 |  |  | K1 |  |
| s6 | 0101 | s5 | 0111 | x5x4 | 12 |  |  | J1 |  |
| 0101 | s7 | 0100 | x5x6 | 16 |  |  |  | K0 |
| 0101 | s8 | 1100 | x5x6x7 | 17, 18 | J3 |  |  | K0 |
| 0101 | s6 | 0101 | x5x4 | - |  |  |  |  |
| s7 | 0100 | s8 | 1100 | x7 | 17, 18 | J3 |  |  |  |
| 0100 | s9 | 1101 | x7 | 19 | J3 |  |  | J0 |
| s8 | 1100 | s9 | 1101 | 1 | 19 |  |  |  | J0 |
| s9 | 1100 | s0 | 1100 | 1 | - |  | K3 | K2 | K0 |

**Functii**

***J3*** = S6x5x6x7 + S7x7 + S7x7

***K3*** = S9

***J2*** = S2x2x3 + S2x2x3x4 + S3x3 + J3x3x4 + S3x3x4

***K2*** = S9

***J1*** = S1 + S6x5x4

***K1*** = S9x3x4 + S4x4 + S5

***J0*** = S0x1 + S3x3x4 + S3x3x4 + S4x4 + s4x4 + S7x7 + S8

***K0*** = S2x2 + S2x2x3 + S6x5x6 + S6x5x6x7 + S9

**4.Concluzii**

In urma efectuarii proiectului de am elaborat element multifunctional. Pentru aceasta am utilizat cunostinte din ASDN creaind tabelul de adevar si funtiile pentru ciruit. Deasemenea am folosit programul Logic Works in care am elaborate circuitul.

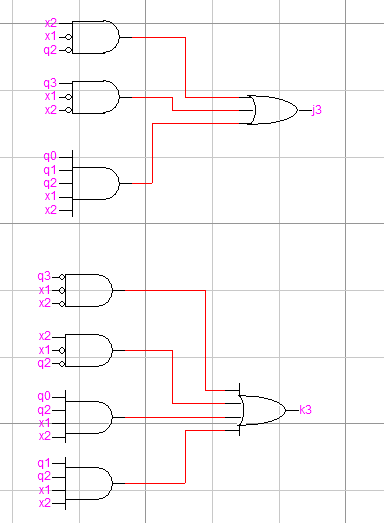
**5.Bibliografia**

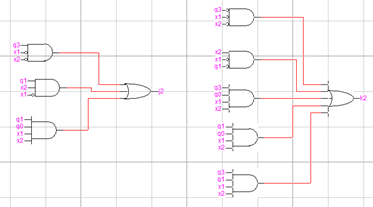
1) <https://www.wikipedia.org/>

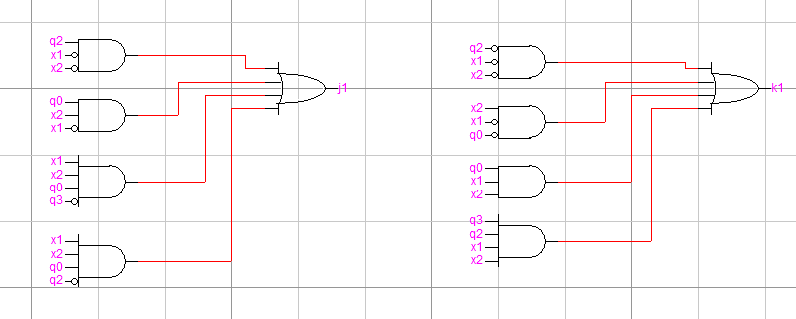
2)<https://www.academia.edu/12153630/PARTEA_A_II-a._CIRCUITE_LOGICE_SECVENTIALE_CLS_1._GENERALITATI>

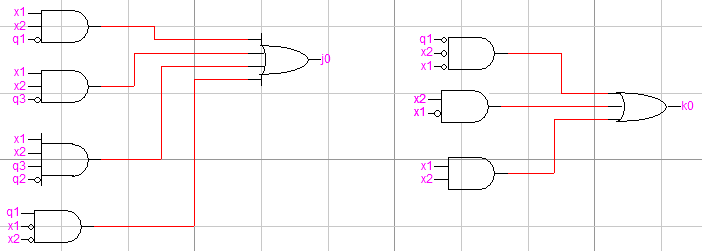
**6.Anexe**

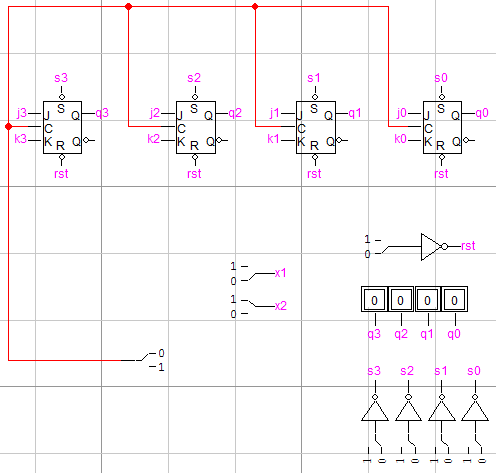
**Schema LogicWorks pentru parte I**

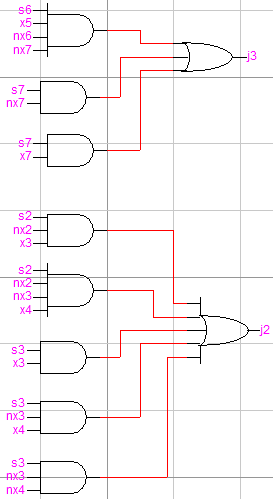
****

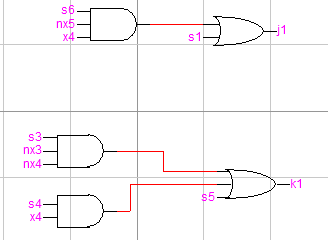
****

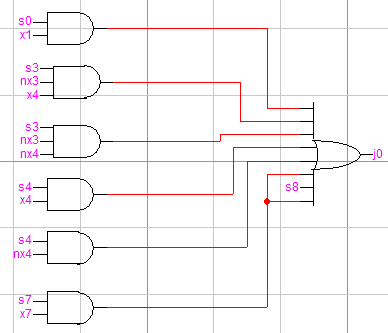
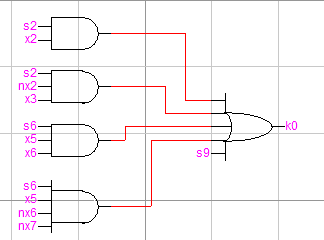
****

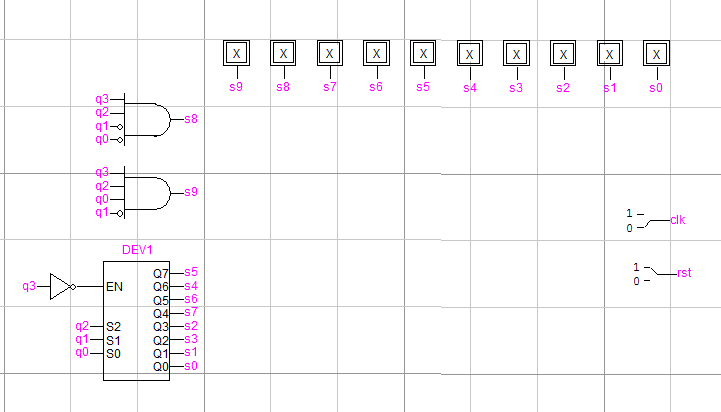
****

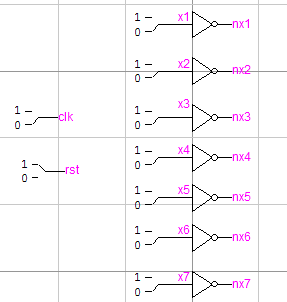
****

**Schema LogicWorks pentru parte II**

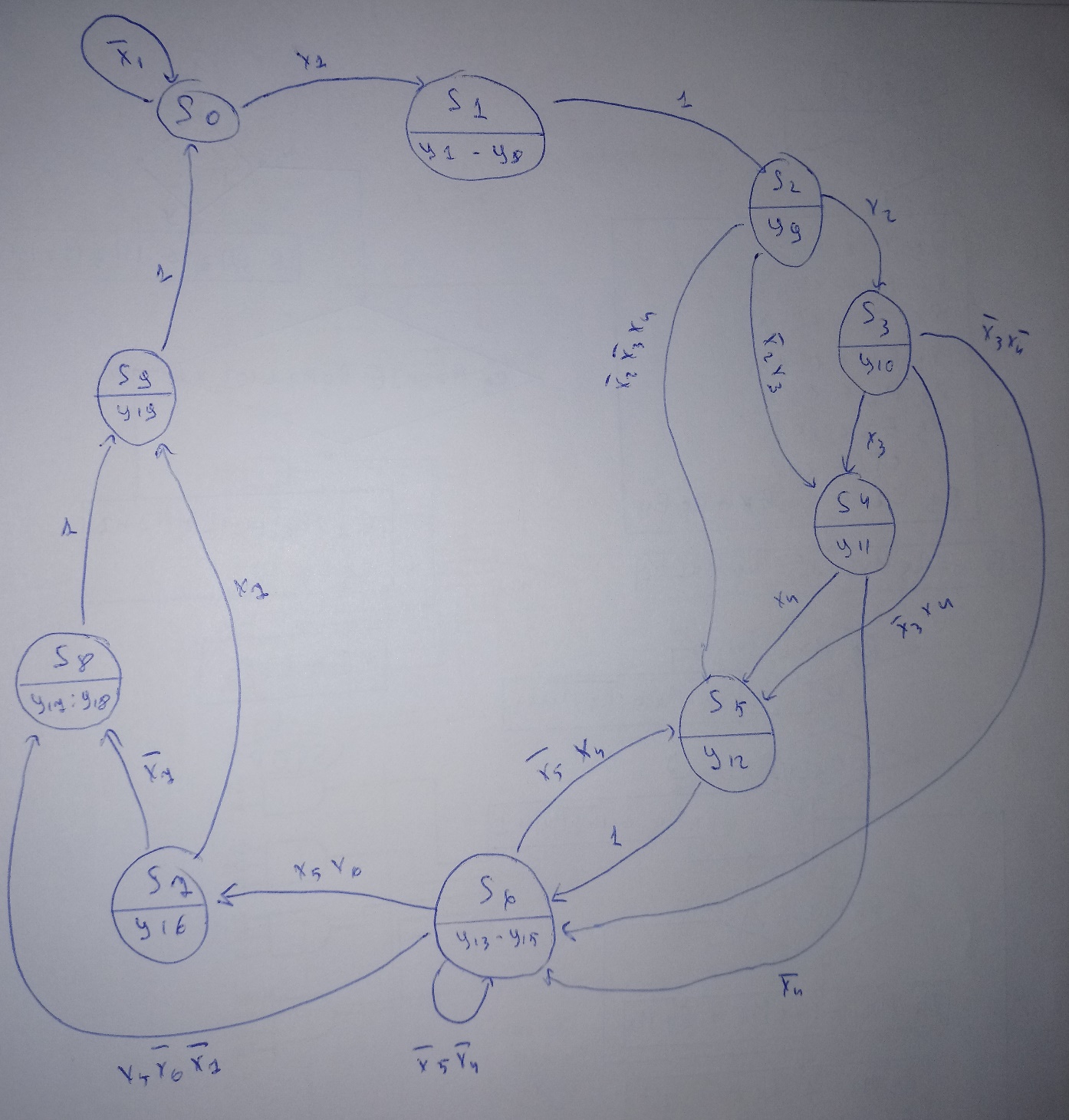








**Schema bloc marcata**

****

