

# Electronica digitală

## Table of Contents

- 1. Sisteme de numeratie
  - 1.1. Numere și simboluri
  - 1.2. Sisteme de numerație
- 2. Aritmetica binara
  - 2.1. Valori și sisteme de numerație
  - 2.2. Adunarea binară
  - 2.3. Numere binare negative
  - 2.4. Scăderea binară
  - 2.5. Depășirea binară
  - 2.6. Grupări de biți
- 3. Porti logice
  - 3.1. Semnale digitale și porți
  - 3.2. Porți logice cu două intrări
  - 3.3. Principiul universalității
  - 3.4. Modul de împachetare
- 4. Comutatoare
  - 4.1. Tipuri de comutatoare
  - 4.2. Poziția „normală” a contactelor
- 5. Relee electromecanice
  - 5.1. Construcția releelor
  - 5.2. Contactoare
  - 5.3. Relee temporizate
  - 5.4. Relee semiconductoare
- 6. Logica ladder
  - 6.1. Diagrame ladder
  - 6.2. Funcții logice digitale
  - 6.3. Circuite permissive și de blocare
  - 6.4. Circuite logice cu autoprotecție
  - 6.5. Automate programabile (PLC)
- 7. Algebra booleana
  - 7.1. Introducere și scurt istoric
  - 7.2. Aritmetica booleană
  - 7.3. Identități algebrice booleene
  - 7.4. Proprietăți algebrice booleene
  - 7.5. Reguli de simplificare booleană
  - 7.6. Simplificarea circuitelor logice
  - 7.7. Funcția SAU-exclusiv
  - 7.8. Teoremele lui DeMorgan
  - 7.9. Transformarea tabelelor de adevăr în expresii booleene
- 8. Harti karnaugh
  - 8.1. De ce hărți Karnaugh
  - 8.2. Diagrame Venn
  - 8.3. Relații booleene cu diagrame Venn
  - 8.4. Transformarea diagramelor Venn în hărți Karnaugh
  - 8.5. Hărți Karnaugh, tabele de adevăr și expresii booleene
  - 8.6. Simplificarea circuitelor logice cu hărți Karnaugh
  - 8.7. Hărți Karnaugh de patru variabile
  - 8.8. Mintermeni și maxtermeni
  - 8.9. Notăția  $\Sigma$  (sumă) și notația  $\Pi$  (produs)
  - 8.10. Hărți Karnaugh de 5 și 6 variabile
- 9. Circuite logice combinatoriale
  - 9.1. Circuite logice combinatoriale - introducere
  - 9.2. Half-Adder
  - 9.3. Full-Adder

## 1 Sisteme de numeratie

### 1.1 Numere și simboluri

Exprimarea cantităților sub formă numerică ni se pare un lucru natural. Această situație este atât în avantajul cât și în dezavantajul nostru atunci când studiem electronica. Pe de o parte, suntem obișnuiți să facem calcule atunci când analizăm circuitele electrice sau electronice, iar acesta este un lucru bun. Pe de altă parte, sistemul de notație utilizat zi de zi, încă din școala primară, nu este sistemul intern folosit și de echipamentele electronice moderne. Adoptarea unui nou sistem de notație și o re-examinare a ideilor și conceptelor deja învățate nu este tocmai un lucru peste care să putem trece cu ușurință.

În primul rând, trebuie să facem o diferențiere între numere și simbolurile utilizate pentru reprezentarea acestor numere. Un număr

este o cantitate matematică, corelată de obicei în cazul electronicii cu o cantitate fizică precum tensiune, curent sau rezistență. Există o multitudine de tipuri de numere. De exemplu: numere naturale (1, 2, 3, ...), numere întregi (... , -3, -2, -1, 0, 1, 2, 3, ...), numere iraționale ( $\pi$  - aproximativ 3,1415927,  $e$  - aproximativ 2,718281828), rădăcina pătrată a oricărui număr prim, etc.), numere reale (toate valorile numerice uni-dimensionale, negative și pozitive, incluzând zero, numerele naturale, întregi și iraționale) și complexe ( $3 - j4$ ,  $34$ ,  $5 \angle 20^\circ$ ).

În funcție de aplicația practică în cauză, se utilizează diferite tipuri de numere. Numerele naturale sunt perfecte și suficiente pentru „inventarierea” obiectelor discrete, precum numărul de rezistori dintr-un circuit. Numerele întregi sunt necesare atunci când avem nevoie și de echivalentul negativ al celor naturale. Numerele iraționale reprezintă acele numere ce nu pot fi exprimate exact ca și raport dintre două numere întregi; raportul dintre circumferința unui cerc și diametrul acestuia ( $\pi$ ) este un astfel de număr irațional. Valorile pentru tensiune, curent și rezistența ce le-am întâlnit în analiza circuitelor electrice de curent continuu pot fi exprimate sub forma numerelor reale, atât sub formă de fracții cât și sub formă decimală. Pentru analiza circuitelor de curent alternativ însă, numerele reale nu pot exprima esența duală a amplitudinii și a unghiului de fază, astfel încât am fost nevoiți să utilizăm numerele complexe, fie sub forma rectangulară, fie sub formă polară.

### 1.1.1 Forma analogică și forma digitală

În cazul în care utilizăm numere pentru înțelegerea proceselor fizice din lumea reală, realizarea predicțiilor științifice sau pentru calcule economice, avem nevoie de o simbolistică pentru reprezentarea acestora. Aceste notații pot fi sub două forme: analogică și digitală. În cazul reprezentării analogice, cantitatea simbolizată este divizibilă la infinit. În cazul reprezentării digitale, cantitatea simbolizată prezintă o diviziune discretă.



Figure 1: termometru analogic

De exemplu, un termometru „clasic” precum în figura alăturată reprezintă un aparat de măsură analogic. Practic putem măsura orice temperatură din intervalul  $0-50^\circ$ , rezoluția termometrului fiind practic infinită. De exemplu, putem spune că temperatura măsurată în acest caz este de  $35^\circ$ , dar, dacă avem ochi buni, putem fi mai preciși și spune că ea este de fapt  $35,7^\circ$ . Sau, dacă avem ochi foarte buni, sau un mijloc mult mai precis de citire a scalei, s-ar putea să vedem că temperatura reală este de fapt de  $35,72545^\circ$ .

Acest lucru nu este valabil și în cazul unui termometru digital. De exemplu, termometrul din figura alăturată nu poate măsura temperatura cu o precizie mai mare de  $0,1^\circ\text{C}$ . Astfel că putem citi o temperatură fie de  $33,0^\circ\text{C}$ , fie o temperatură de  $33,1^\circ\text{C}$ , dar în niciun caz nu putem citi o valoare între aceste două puncte (de exemplu,  $33,0125^\circ\text{C}$ ), așa cum am fi putut face cu un termometru analogic.

## 1.2 Sisteme de numerație

### 1.2.1 Sistemul de numerație roman

Romanii au pus la punct un sistem de numerație pe bază de simboluri (cifre) pentru reprezentarea cantităților, astfel:

$X = 10$   $L = 50$   $C = 100$   $D = 500$   $M = 1000$

Dacă o cifră este urmată de o altă cifră a cărei valoare este egală sau mai mică decât prima, niciuna dintre cifre nefiind mai mare decât cele din stânga sa, valoarea acestei cifre se adaugă la valoarea totală. Astfel,  $VIII$  reprezintă valoarea 8, iar  $CLVII$  reprezintă 157. Pe de altă parte, dacă o cifră este precedată la stânga sa de o altă cifră a cărei valoare este mai mică decât prima, valoare celei de a doua se scade din prima. Prin urmare,  $IV = 4$  ( $V$  minus  $I$ ), iar  $CM = 900$  ( $M$  minus  $C$ ). De exemplu, anul 1987 poate fi reprezentat în notația romană astfel:  $MCMLXXXVII$ . O analiză a acestei notații este bine-venită:

$M (1000) + CM (900) + L (50) + XXX (30) + V (5) + II (2) = 1987$

Numerele mari sunt dificil de reprezentat prin intermediul acestei notații. Adunarea și scăderea cifrelor ne poate și ea da bătăi de cap. O altă problemă majoră a acestui sistem este imposibilitatea reprezentării numerelor negative sau a valorii nule (zero), ambele fiind concepte foarte importante în matematică.

### 1.2.2 Sistemul de numerație zecimal

Una dintre cele mai importante idei ale sistemului zecimal de numerație se datorează babilonienilor. Aceștia au fost aparent prima civilizație ce s-a folosit de poziția cifrei pentru reprezentarea numerelor mari. În loc să inventeze cifre noi pentru reprezentarea cantităților mari, precum romanii, aceștia au refolosit aceleași cifre, dar plasate în poziții diferite de la dreapta spre stânga.

Sistemul zecimal actual utilizează acest concept, folosind doar 10 cifre (0, 1, 2, 3, 4, 5, 6, 7, 8 și 9) pentru reprezentarea valorilor în funcție de poziția acestora. Fiecare cifră reprezintă o valoare întreagă, iar fiecare poziție de la dreapta spre stânga reprezintă o constantă de multiplicare pentru fiecare dintre aceste valori întregi. De exemplu, notația zecimală „1206”, poate fi desfăcută în următorul produs:

$1206 = 1000 + 200 + 6$   $1206 = (1 \times 1000) + (2 \times 100) + (0 \times 10) + (6 \times 1)$

Fiecare simbol poartă numele de cifră, iar fiecare poziție este de zece ori mai mare decât poziție imediat următoare (din dreapta). Astfel că în cazul de mai sus avem poziția sau cifra unităților (6), cifra zecilor (0), cifra sutelor (2) și cifra miilor (1), de la dreapta spre stânga.

### 1.2.3 Sistemul de numerație binar

Ce s-ar întâmpla dacă am realiza un sistem de numerație cu aceleași principii de bază precum sistemul zecimal, dar cu mai puține sau mai multe cifre?

Sistemul binar este un astfel de sistem „modificat” ce utilizează doar două cifre, constanta de multiplicare a fiecărei cifre fiind în acest caz de două ori mai mare decât a cifrei precedente (de la dreapta la stânga). Cele două cifre sunt „0” și „1”. Poziția din dreapta este poziția unităților, la fel ca în cazul notației zecimale. Spre stânga, constantele de multiplicare sunt după cum urmează: 2, 4, 8, 16, etc. De exemplu, următorul număr binar poate fi exprimat, la fel ca și numărul zecimal 1206, ca și sumă dintre produsul fiecărei cifre cu, constanta de multiplicare (în funcție de poziție):

$$11010 = 2 + 8 + 16 = 26 \quad 11010 = (1 \times 16) + (1 \times 8) + (0 \times 4) + (1 \times 2) + (0 \times 1)$$

### 1.2.4 Specificarea bazei

Mai sus, am scris un număr sub formă binară (11010) și l-am transformat în formă zecimală ( $16 + 8 + 2 = 26$ ). Prin urmare, am amestecat două notații diferite în același loc. Pentru a nu crea confuzii, va trebui să explicităm tipul notației folosite. Acest lucru se realizează prin specificarea bazei numărului respectiv prin folosirea indicilor, 2 pentru notația binară, și 10 pentru cea zecimală, astfel:  $11010_2$  (baza doi) și  $26_{10}$  (baza zece).

Acești indicii nu sunt operatori matematici, precum exponenții (puteri). Tot ceea ce fac este să indice tipul de sistem de numerație utilizat pentru reprezentarea numărului respectiv. De obicei, atunci când nu este specificată nicio bază, se presupune că se lucrează în baza zece ( $_{10}$ ).

De remarcat că, în cazul notației binare, fiecare poziție poartă numele de bit

### 1.2.5 Scopul sistemului binar de numerație

De ce am vrea să folosim acest sistem de numerație binar? Sistemul decimal, cu cele zece cifre ale sale, este intuitiv și ușor de înțeles. Sistemul binar este folosit în principal de electronica digitală (folosită pentru calculatoare, de exemplu), datorită ușurinței de reprezentare electronică a celor două stări (0 și 1). Cu un circuit relativ simplu, putem efectua operații matematice asupra numerelor binare reprezentând fiecare bit printr-un circuit care este fie pornit (curent) fie oprit (curent zero). La fel ca în cazul unui abac, putem adăuga mai multe circuite pentru a reprezenta numere din ce în ce mai mare. Acest sistem este ideal pentru stocarea și redarea informației sub format numeric: benzi magnetice, CD-uri, hard-disk-uri, etc.

1. [Ce sunt sistemele de numerație și despre sistemul zecimal](#) (stefamedia.ro)

## 2 Aritmetica binară

### 2.1 Valori și sisteme de numerație

Este foarte important să înțelegem că sistemul de numerație ales pentru reprezentarea valorilor (numerelor) nu are absolut niciun impact asupra rezultatului aplicării operațiilor aritmetice de adunare, scădere, înmulțire, împărțire, rădăcini, puteri sau algoritmi. O valoare este tot timpul aceeași, indiferent de modul în care alegem să o simbolizăm. Fie că reprezentăm temperatura de  $35^\circ$  sub această formă (zecimală) sau sub forma 100011 (binară), aceasta nu schimbă valoarea reală a temperaturii ce o resimțim. Ea rămâne aceeași, ceea ce se modifică este modul de reprezentare a acesteia. Operațiile esențiale și legăturile matematice nu sunt afectate de modificarea sistemului de numerație pentru reprezentarea valorilor. Această distincție între valori și sisteme de numerație este foarte importantă și trebuie înțeleasă.

Această distincție esențială dintre cei doi termeni este asemănătoare distincției dintre cuvinte și obiectele asociate acestor cuvinte. O casă este tot o casă, indiferent dacă de limba pe care o folosim pentru desemnarea acesteia (română, engleză, germană, etc.). Obiectul este ceva real, palpabil, pe când cuvântul este doar un simbol pentru reprezentarea acelui obiect.

Acestea fiind spuse, o simplă operație de aritmetică sub formă binară, precum adunarea, pare ciudată pentru o persoană obișnuită să lucreze doar cu sistemul de numerație zecimal. În acest capitol vom analiza tehnicile folosite pentru efectuarea operațiilor aritmetice simple cu numere binare. Aceste tehnici vor fi folosite pentru proiectarea circuitelor electronice care să realizeze exact același lucru. Chiar dacă suntem obișnuiți cu operațiile de adunare și scădere folosind un calculator de mână, calculatorul însăși folosește numerele binare pentru obținerea rezultatului final.

### 2.2 Adunarea binară

Adunarea numerelor binare este relativ simplă, foarte asemănătoare cu adunarea numerelor zecimale. Adunarea se realizează adunând pe coloane fiecare bit, pe rând, de la dreapta la stânga, la fel ca în cazul adunării zecimale. Spre deosebire de aceasta însă, există puține reguli de memorat:

$$0 + 0 = 0 \quad 1 + 0 = 1 \quad 0 + 1 = 1 \quad 1 + 1 = 10 \quad 1 + 1 + 1 = 11$$

Când rezultatul adunării pe coloane este format din doi biți (de ex,  $1 + 1 = 10$ ), bitul din dreapta (0, în acest caz) se scrie iar cel din stânga se trece mai departe (1, în acest caz):

	11 1	11	<--- biți ce trec mai departe
1001101	1001001	1000111	
+ 0010010	+ 0011001	+ 0010110	
-----	-----	-----	
1011111	1100010	1011101	

Adunarea din stânga nu a dat niciunde doi biți, pe fiecare coloană suna fiind fie 1, fie 0. În celelalte două adunări, există sume care dau fie 10 fie 11, iar în acest caz bitul din stânga (0 sau 1) se trece mai departe la următoarea adunare.

După cum vom vedea mai târziu, se pot construi circuite electronice care să realizeze exact această operație aritmetică de adunare, prin reprezentarea fiecărui bit a fiecărui număr prin intermediul unui semnal de tensiune. Acest principiu reprezintă baza efectuării tuturor operațiilor aritmetice realizate de calculatoarele moderne.

### 2.3 Numere binare negative

#### 2.3.1 Bit-ul de semn

Operația de scădere se poate realiza asemănător cu cea de adunare prin considerarea unuia dintre numere ca fiind negativ. De

exemplu, operația de scădere „7 - 5” este aceeași cu cea de adunare „7 + (-5)”, adică, adunarea unui număr pozitiv cu un număr negativ. Din moment ce știm deja cum se realizează reprezentarea numerelor pozitive sub formă binară, tot ceea ce trebuie să facem este să reprezentăm și numerele negative sub formă binară. De aici va rezulta direct operația de scădere. Un număr zecimal negativ se reprezintă de obicei prin introducerea semnului minus(-) la stânga, la fel ca în exemplul de sus (-5). Totuși, scopul notație binare este realizarea circuitelor tip oprit/pornit pentru reprezentarea valorilor sub forma căderilor de tensiune (două valori alternative: „înalță” și „joasă”). În această situație, nu ne putem permite să introducem un al treilea simbol, precum semnul minus, din moment ce aceste circuite au doar două stări posibile, pornit sau oprit. O soluție o reprezintă utilizarea unui bit (circuit) doar pentru reprezentarea acestui semn matematic și pentru nimic mai mult:  $101_2 = 5_{10}$  (pozitiv) Utilizând un bit adițional (0 = pozitiv, 1 = negativ):  $0101_2 = 5_{10}$  (pozitiv)  $1101_2 = -5_{10}$  (negativ) Dar, în această situație în care folosim biți pentru alt scop decât pentru reprezentarea valorilor, trebuie să fim foarte atenți, altfel, riscăm ca numărul  $1101_2$  să fie interpretat ca fiind  $13_{10}$  în loc de  $-5_{10}$ . Pentru a nu face astfel de greșeli, trebuie să ne decidem în primul rând de câți biți avem nevoie pentru a reprezenta cel mai mare număr posibil cu care vom lucra în aplicația noastră. Ne putem apoi asigura că nu vom depăși această lungime (în biți) atunci când aplicăm operațiile aritmetice. În exemplul de mai sus, limita inferioară este -7 ( $1111_2$ ) iar cea superioară 7 ( $0111_2$ ), deoarece al patrulea bit este folosit pe post de semn. Doar prin stabilirea acestor numere putem fi siguri că nu vom amesteca un număr negativ cu un număr pozitiv, mai mare. Pe cât de simplă pare această abordare, ea nu este foarte practică din punct de vedere al aritmeticii. De exemplu, cum efectuăm adunarea unui număr negativ ( $1101_2$ ) cu un oricare alt număr, folosind tehnica standar al adunării binare? Ar trebui să inventăm o nouă metodă de realizare a adunării pentru ca această tehnică să fie practică. Dar, dacă realizăm acest lucru, nu vom mai avea avantajul utilizării numerelor negative pentru realizarea scăderii prin adunare obișnuită.

2.3.2 Reprezentarea în complement față de doi

Din fericire, există o altă metodă pentru reprezentarea numerelor negative ce este compatibilă cu operația de adunare obișnuită, și anume, complementarea. Cu această strategie, bit-ul din stânga primește un statut special, asemenea bit-ului de semn din exemplul precedent. Totuși, de această dată, bit-ul din stânga nu este doar un bit de semn, ci posedă și o valoare. De exemplu, -5 este reprezentat astfel:  $1011_2 = -5_{10}$  ( $1 \times -8_{10}$ ) + ( $0 \times 4_{10}$ ) + ( $1 \times 2_{10}$ ) + ( $1 \times 1_{10}$ ) =  $-5_{10}$  Utilizând cei trei biți din dreapta pentru reprezentarea valorilor de la zero la șapte, bit-ul din stânga reprezentând fie zero fie -8, putem reprezenta orice număr întreg de la -7 ( $1001_2 = -8_{10} + 1_{10} = -7_{10}$ ) la plus 7 ( $0111_2 = 010 + 710 = 710$ ). Reprezentarea numerelor pozitive utilizând această notație nu este diferită față de notația normală (bit-ul din stânga va fi tot timpul zero). Totuși, reprezentarea numerelor negative nu este chiar așa de intuitivă.

0 = 0000	
1 = 0001	-1 = 1111
2 = 0010	-2 = 1110
3 = 0011	-3 = 1101
4 = 0100	-4 = 1100
5 = 0101	-5 = 1011
6 = 0110	-6 = 1010
7 = 0111	-7 = 1001
	-8 = 1000

Observăm că numerele binare negative din coloana dreaptă, fiind suma celor trei biți din dreapta plus bitul negativ din dreapta, egal cu minus opt, nu se „numără” în aceeași ordine precum numerele binare pozitive din coloana stângă. În schimb, cei trei biți trebuie aduși la forma necesară, astfel încât, după adunarea cu minus opt să se obțină rezultatul negativ dorit. Spunem că cei trei biți din dreapta sunt o reprezentare în complement față de doi al numărului pozitiv corespunzător. Să facem o comparație:

număr pozitiv	complementul față de doi
-----	-----
001	111
010	110
011	101
100	100
101	011
110	010
111	001

În acest caz (bit-ul al patrulea are valoarea de minus opt), reprezentarea în complement față de doi a oricărui număr pozitiv este valoarea necesară însumări cu minus opt pentru a rezulta aceeași valoare, dar cu semn schimbat. Din fericire, există o cale ușoară de calcul al complementului, pentru oricare număr binar: inversăm toți biții celui număr, schimbând 1 cu 0 și invers. Ajungem astfel la reprezentarea în complement față de unu a numărului. Pentru soluția dorită de noi, mai trebuie să adăugăm un unu. De exemplu, pentru obținerea complementului lui cinci ( $101_2$ ), inversăm toți biții și obținem  $010_2$ (complement față de unu), apoi adăugăm un unu și obținem  $011_2$ , sau  $-5_{10}$  în complement față de doi. Este interesant de menționat faptul că, obținerea complementului față de doi a unui număr binar funcționează la fel de bine și dacă aplicăm inversarea tuturor biților, inclusiv bit-ului din stânga. Să luăm exemplul precedent, inversarea lui 5 în -5, dar aplicând operația de inversare tuturor biților. Trebuie să includem însă și bit-ul din stânga numărului inițial ( $0101_2$ ). După inversarea tuturor biților, obținem complementul față de unu ( $1010_2$ ). Apoi, adăugăm un unu pentru obținerea răspunsului final:  $1011_2$ , sau  $-5_{10}$  în complement față de doi, exprimat cu patru biți.

2.4 Scăderea binară

Putem realiza operația de scădere binară utilizând aceleași metode standard împrumutate de la scăderea zecimală. Totuși, dacă

putem utiliza metoda deja cunoscută (și mai ușoară) a adunării binare pentru efectuarea scăderii, ne va fi mai ușor. După cum am văzut, putem reprezenta numerele binare negative utilizând reprezentarea în complement a lui doi plus un bit adițional cu o valoare negativă. Să considerăm un exemplu:

$7_{10} - 5_{10}$  (scădere)  $7_{10} + (-5_{10})$  (adunare echivalentă)

Tot ce trebuie să facem este să reprezentăm numărul 7 și -5 sub formă binară:

$0111_2 = 7$   $1011_2 = -5$

Nu ne mai rămâne decât să efectuăm adunarea binară:

```
  1111    <--- biți ce trec mai departe
  0111
+ 1011
-----
 10010    <--- ignorăm bit-ul suplimentar

răspuns = 00102 = 210
```

Din moment ce am definit numărul nostru ca fiind compus din trei biți plus bitul cu valoare negativă, putem ignora al cincilea bit din răspuns (1), iar rezultatul final este astfel  $0010_2$ , sau plus doi, ceea ce reprezintă răspunsul corect.

O altă modalitate de a înțelege de ce înlăturăm al cincilea bit, este să ținem minte că bit-ul din stânga are o valoare negativă, egală cu minus opt în cazul de față. Atunci când adunăm aceste două numere binare, realizăm de fapt o scădere a biților. În cazul operației de scădere, cifrele nu sunt „duse” mai departe spre următoarea operație, ci sunt împrumutate.

Să considerăm un alt exemplu, cu numere mai mari de data aceasta. Dacă vrem să adunăm  $-25_{10}$  cu  $18_{10}$ , trebuie să stabilim în primul rând numărul de biți pe care numărul nostru îl va conține în reprezentarea binară. Pentru a putea reprezenta cea mai mare valoare absolută posibilă în acest caz, 25, avem nevoie de cel puțin cinci biți, plus un al șaselea bit pentru valoarea negativă. Să începem prin a reprezenta numărul 25 sub formă binară, și apoi sub forma complementului față de doi:

$+25_{10} = 011001_2$  (toți cei șase biți)  $11001_2 = 100110_2$  (complementul față de unu) complementul față de unu + 1 = complementul față de doi =  $100111_2 - 25_{10} = 100111_2$  (forma finală)

Mai exact, -25 sub formă binară este de fapt suma dintre bit-ul negativ de pe poziția a șasea cu valoarea de -32 și ceilalți cinci biți ( $00111_2 = 7_{10}$ ).

Să reprezentăm acum și numărul 17 sub formă binară, folosind toți cei șase biți:

$18_{10} = 010010_2$

Adunarea lor ne conduce la următorul rezultat:

```
   11
  100111
+ 010010
-----
 111001
```

În acest caz nu avem un „surplus” de biți după adunare, prin urmare, nu trebuie să „scăpăm” de niciunul din ei. Bitul din stânga este 1, ceea ce înseamnă că răspunsul, în complement față de doi, este negativ (ceea ce este corect). Pentru verificare, putem realiza conversia înapoi în forma zecimală prin însumarea produsului tuturor biților cu valorile lor respective, astfel:

$(1 \times -32_{10}) + (1 \times 16_{10}) + (1 \times 8_{10}) + (1 \times 1_{10}) = -7_{10}$

Răspunsul obținut este corect ( $18_{10} - 25_{10} = -7_{10}$ ).

## 2.5 Depășirea binară

Una din problemele numerelor binare cu semn, este bit-ului de depășire. Acesta apare ca în cazul în care rezultatul adunării sau scăderii efectuate între două numere binare este mai mare decât valoarea maximă ce poate fi reprezentată cu numărul de biți alocati. Țineți minte că poziția bit-ului de semn este fixată la începutul problemei. În exemplul precedent, am utilizat cinci biți pentru reprezentarea unui număr, iar bit-ul din stânga a fost utilizat pe post de bit de semn, cu pondere negativă. Cu cinci biți rămași pentru reprezentarea valorilor, cel mai mare număr ce-l putem scrie astfel este  $+31_{10}(01111_2)$ , iar cel mai mic  $-32_{10}(10000_2)$ . Dacă aplicăm o operație de adunare unor astfel de numere, iar rezultatul este mai mare decât  $31_{10}$  sau mai mic decât  $-32_{10}$ , răspunsul obținut nu va fi corect. Să luăm un exemplu:

$17_{10} = 10001_2$   $19_{10} = 10011_2$

Adăugând bit-ul de semn, adunarea celor două numere arată astfel:

```
  1  11    <--- biți ce se trec mai departe
  010001
+ 010011
-----
 100100
```

Răspunsul ( $100100_2$ ) este egal cu  $-28_{10}$ , nu cu  $+36_{10}$ , așa cum ar trebui să obținem adunând  $+17_{10}$  cu  $+19_{10}$ . Evident, acest răspuns nu este corect. Dar unde am greșit? Din moment ce valoarea reală a sumei ( $36_{10}$ ) depășește limita permisă de cei cinci biți (plus bit-ul de semn), ajungem la o eroare de depășire binară.

O eroare similară obținem și în cazul adunării a două numere negative a cărei sumă este mai mică decât  $-32_{10}$ :

$-17_{10} = 10111_2$   $-19_{10} = 10110_2$

Aplicând operația de adunare celor două numere, obținem:

```
  1 1111    <--- biți ce se trec mai departe
```

```

101111
+ 101101
-----
1011100

```

După înlăturarea bit-ului în plus, rezultatul final este:

$011100_2 = +28_{10}$

Acest răspuns este, desigur, incorect.

Să reconsiderăm cele două exemple de mai sus, dar de această dată utilizând șase biți pentru reprezentarea numărului. Al șaptelea bit va fi bit-ul de semn:

$17_{10} + 19_{10}$

$(-17_{10}) + (-19_{10})$

```

  1  11
0010001
+ 0010011
-----
01001002

```

```

  11 1111
1101111
+ 1101101
-----
110111002 <--- înlăturarea bit-ului suplimentar

```

Răspunsuri:  $01001002 = +36_{10}$

$10111002 = -36_{10}$

Utilizând un număr suficient de biți pentru reprezentarea rezultatelor, răspunsurile sunt corecte.

În aceste exemple am putut determina erorile de depășire prin realizarea „de mână” a operațiilor de adunare sub formă zecimală. Această metodă de verificare nu este însă foarte eficientă. Până la urmă, întregul scop al complementării este realizarea adunării sub formă binară. Acest lucru este valabil mai ales în cazul proiectării circuitelor electronice: circuitul trebuie să poată sesiza singur existența unei erori de depășire, fără ajutor uman.

Cea mai elegantă metodă de rezolvare a acestei situații constă în verificarea „semnului” sumei și compararea acestuia cu semnele numerelor însumate. Desigur, rezultatul sumei a două numere pozitive este un număr pozitiv, iar suma a două numere negative, este un număr negativ. Putem observa că, de fiecare dată când avem o situație de depășire, semnul sumei este invers față de semnul celor două numere adunate:  $+17_{10}$  plus  $+19_{10}$  ne-a dat  $-28_{10}$ , sau,  $-17_{10}$  plus  $-19_{10}$  ne-a dat  $+28_{10}$ . Prin simpla verificare a semnelor, putem să ne dăm seama că exista o eroare iar rezultatul este fals.

Ce se întâmplă în cazul în care unul din numere este pozitiv iar celălalt negativ? Care ar trebui să fie semnul sumei? Răspunsul este simplu: atunci când numerele însumate sunt de semne diferite, nu va exista niciodată o eroare de depășire. Motivul este la fel de simplu: depășirea are loc atunci când valoarea unui număr este mai mare decât cea permisă de numărul de biți utilizați. În cazul numerelor de semn contrar, valoarea rezultatului trebuie să fie între cele două numere, prin urmare, nu poate fi mai mare sau mai mică decât limita maximă permisă de numărul de biți.

Din fericire, această metodă de identificare a erorii de depășire este ușor de implementat într-un circuit electronic.

## 2.6 Grupări de biți

La reprezentarea numerelor binare cu ajutorul circuitelor electronice, suntem nevoiți să utilizăm un număr suficient de circuite cu tranzistori pentru reprezentarea tuturor biților. Cu alte cuvinte, la proiectarea unui circuit digital, trebuie să ne decidem câți biți vom utiliza (maxim) pentru reprezentarea valorilor, din moment ce fiecare bit necesită un circuit pornit/oprit separat.

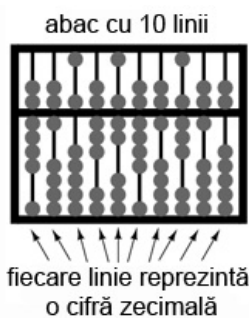


Figure 2: abac cu 10 linii; analogie

Acest principiu este analog proiectării unui abac pentru reprezentarea digitală a numerelor zecimale: trebuie mai întâi să ne decidem câte cifre dorim să reprezentăm cu acest dispozitiv primitiv. Fiecare cifră necesită o nouă linie cu mărgel.

Un abac cu 10 linii poate reprezenta un număr zecimal cu 10 cifre, sau o valoare maximă de 9.999.999.999. Dacă am dori reprezentarea unui număr mai mare decât atât, va trebui să mă adăugăm una sau mai multe linii.

Orice circuit digital se proiectează pe un anumit număr de biți: numărul maxim de biți alocați pentru reprezentarea valorilor numerice. Calculatoarele digitale inițiale erau proiectate pe patru sau pe opt biți. Sistemele moderne sunt proiectate pe 32 sau pe 64 de biți.

Pentru a vorbi mai ușor de numărul de biți dintr-o grupare, cele mai uzuale au primit și un nume, astfel:

Bit: unitatea fundamentală a notației binare; echivalentă cu o cifră zecimală Crumb, Tydbit sau Tayste: 2 biți Nibble sau Nybble: 4 biți Nickle: 5 biți Byte: 8 biți Deckle: 10 biți Playte: 16 biți Dynner: 32 biți Word: (în funcție de sistem)

Cel mai ambiguu termen este word-ul. Pentru un sistem pe 32 de biți, un word înseamnă 32 de biți. Dacă sistemul utilizează 16 biți, atunci word-ul este de 16 biți. Termenul playte și dynner se referă tot timpul la 16, respectiv 32 de biți, indiferent de contextul în care sunt folosiți.

Tot în funcție de sistem sunt folosiți și termenii de „double word” sau „longword” (ambii termenii desemnând o lungime dublă față de lungimea standard), „half-word” (jumătatea lungimii) sau quar (de patru ori lungimea standard).



## 3 Porti logice

### 3.1 Semnale digitale și porți

Deși sistemul de numerație binar reprezintă un concept matematic abstract interesant, încă nu am spus nimic despre aplicațiile practice în electronică. Acest capitol este dedicat prin urmare aplicării conceptelor binare circuitelor electronice. Importanța sistemului binar de numerație în electronica digitală este importantă datorită ușurinței cu care putem reprezenta biții sub formă fizică. Deoarece un bit poate avea doar două stări diferite, fie 0 fie 1, orice mediu fizic ce poate funcționa în două stări saturate diferite, poate fi folosit pentru reprezentarea unui bit. În consecință, orice sistem fizic ce este capabil să reprezinte biți sub formă binară, poate reprezenta de asemenea și valori numerice. Prin urmare, are potențialul de a manipula aceste numere. Acesta este principiul de bază al circuitelor digitale.

Circuitele electronice sunt perfecte pentru reprezentarea numerelor binare. Tranzistorii, atunci când funcționează la limită, se pot afla într-un din cele două stări: fie în stare blocată (curent de control zero), fie în stare de saturație (curent de control maxim). Dacă un circuit cu tranzistor este proiectat pentru maximizarea probabilității de funcționare într-una din cele două stări (evitarea funcționării tranzistorului în zona activă de funcționare), acesta poate fi folosit ca și reprezentare fizică a unui bit. Căderea de tensiune măsurată la ieșirea unui astfel de circuit poate fi folosită pentru reprezentarea unui singur bit. O tensiune joasă reprezentând „0”, și o tensiune (relativ) înaltă reprezentând „1”.

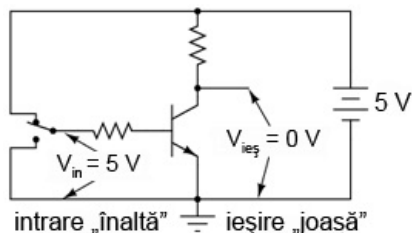


Figure 3: tranzistor aflat în saturație

În figura alăturată, tranzistorul este saturat datorită tensiunii de intrare de 5 V (prin intermediul comutatorului cu două poziții). Deoarece este saturat, căderea de tensiune dintre colector și emitor este foarte mică, rezultând o tensiune de ieșire de practic 0 V. Dacă am folosit acest circuit pentru reprezentarea numerelor binare, am spune că semnalul de intrare este „1” binar, iar semnalul de ieșire este „0” binar. Orice tensiune apropiată de tensiunea de alimentare (având ca referință masa), este considerată a fi „1”, iar o lipsă de tensiune este considerată a fi „0”. Alternativ, se folosesc termenii de „înalt” (1 binar) sau jos (0 binar). Termenul general pentru reprezentarea unui bit prin intermediul unei tensiuni poartă numele de „nivel logic”.

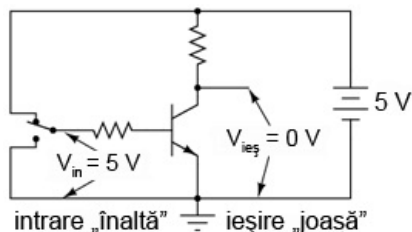


Figure 4: tranzistor aflat în saturație

Trecând comutatorul pe cealaltă poziție, aplicăm o intrare de 0 binar și obținem la ieșire 1 binar.

Ceea ce am creat mai sus poartă numele de poartă logică, sau simplu, poartă. O poartă nu este altceva decât un circuit amplificator special conceput pentru acceptarea și generarea semnalelor de tensiune. Aceste semnale corespund numerelor binare 0 și 1. Prin urmare, porțile nu sunt concepute pentru amplificarea semnalelor analogice (semnale de tensiune între 0 și tensiunea maximă). Mai multe porți conectate împreună se pot folosi pentru stocare (circuite de memorare) sau manipulare (circuite de calcul). Ieșirea fiecărei porți reprezintă în acest caz un singur bit dintr-un număr binar compus din mai mulți biți.

#### 3.1.1 Poarta inversoare (NU sau NOT)

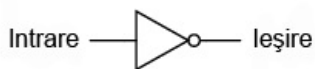


Figure 5: poartă inversoare; simbol

Alăturat este simbolul folosit pentru reprezentarea unei porți inversoare (NOT). Aceasta se comportă identic cu circuitul analizat mai sus, și anume: ieșirea porții este inversă față de intrare (intrare 0, ieșire 1 sau intrare 1, ieșire 0). Aceste porți sunt de fapt circuite cu tranzistoare de genul celui prezentat mai sus, dar, pentru simplificarea analizei circuitelor, se vor folosi aceste simboluri specifice fiecărei porți.

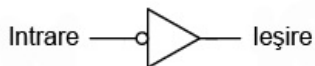


Figure 6: poartă inversoare; simbol alternativ

Un simbol alternativ pentru o poartă inversoare este cel din figura alăturată.

Forma triunghiulară este asemănătoare simbolului amplificatorului operațional. După cum am spus mai sus, porțile sunt de fapt amplificatoare. Metoda standard de reprezentare a unei funcții inversoare este prin intermediul celui mic cerc desenat pe terminalul de intrare sau de ieșire. Dacă îndepărtăm acest cerc din simbolul porții, lăsând doar triunghiul, acest simbol nu ar mai indica o inversare, ci o amplificare. Un astfel de simbol, și o astfel de poartă chiar există, și poartă numele de poarta ne-inversoare, sau buffer.

### 3.1.2 Poarta ne-inversoare (buffer)

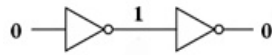


Figure 7: inversare dublă (două porți inversoare conectate cap-coadă)

Dacă ar fi să conectăm două porți inversoare, una în continuarea celeilalte, cele două funcții de inversare s-ar „anula” reciproc. În acest caz, semnalul de ieșire va fi același cu cel de intrare.

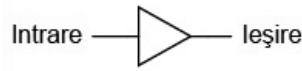


Figure 8: poartă ne-inversoare (buffer); simbol

Pentru acest scop, există o poartă logică separată, denumită buffer (sau poartă ne-inversoare). Simbolul este un triunghi simplu, precum în figura alăturată.

### 3.1.3 Reprezentarea conexiunilor

Asemănător simbolului amplificatorului operational, conexiunile de intrare și de ieșire sunt reprezentate printr-un singur fir, punct de referință implicit pentru fiecare cădere de tensiune fiind masa. În circuitele logice, masa este aproape tot timpul reprezentată de către conexiunea negativă a sursei de alimentare. Sursele de alimentare duale sunt rareori folosite în astfel de circuite. Datorită faptului că circuitele logice (cu porți) sunt de fapt niște amplificatoare, acestea necesită o sursă de putere pentru funcționare. La fel ca în cazul AO, conexiunile surselor de alimentare sunt omise pentru simplitate.

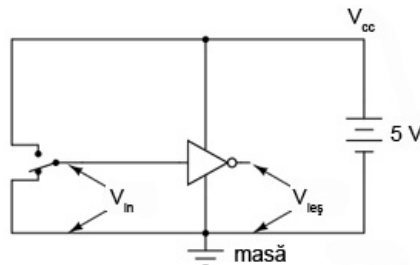


Figure 9: poartă inversoare; circuit de alimentare complet

Dacă ar fi să reprezentăm toate conexiunile necesare pentru utilizarea acestei porți, circuitul ar arăta precum cel din figura alăturată.

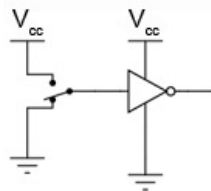


Figure 10: poartă inversoare; circuit de alimentare complet

Conductorii surselor de alimentare sunt rareori reprezentați în circuitele electronice, chiar și atunci când sunt reprezentate conexiunile pe fiecare poartă. Astfel, eliminând liniile ce nu sunt necesare, obținem rezultatul alăturat.

$V_{cc}$  reprezintă tensiunea constantă de alimentare a colectorului din circuitul cu tranzistor bipolar. Punctul de referință este, desigur, masa. Punctele marcate cu  $V_{cc}$  sunt toate conectate la același punct, iar acel punct este borna pozitivă a sursei de alimentare de curent continuu. Valoarea acesteia este de obicei de 5 V.

### 3.1.4 Tabelul de adevăr

Intrare	Ieșire
0	1
1	0

O modalitate de exprimare a funcției unei porți logice, poartă numele de tabel de adevăr. Aceste tabele descriu toate combinațiile posibile ale intrărilor și rezultatul ieșirilor. Pentru poarta inversoare, sau NOT, prezentată mai sus, tabelul de adevăr este cel alăturat.

Intrare	Ieșire
0	0
1	1

Pentru poarta ne-inversoare, tabelul de adevăr este puțin diferit.

Pentru porți mai complexe, tabelele de adevăr sunt mai mari decât acesta. Numărul liniilor unui astfel de tabel trebuie să fie egal cu  $2^n$ , unde  $n$  reprezintă numărul intrărilor porții logice considerate.

## 3.2 Porți logice cu două intrări



Având doar o intrare, singurele porți „disponibile” sunt cele inversoare și cele ne-inversoare. Pentru a lucra cu mai multe posibilități, trebuie să mărim numărul de intrări.

O poartă cu o singură intrare prezintă doar două posibilități: fie intrarea este „înaltă” (1), fie este „joasă” (0). În schimb, o poartă cu două intrări are patru posibilități (00, 01, 10, 11). O poartă cu trei intrări are opt combinații posibile (000, 001, 010, 011, 100, 101, 110 și 111). După cum am mai spus, numărul combinațiilor posibile este egal cu  $2^n$ , unde  $n$  este numărul de intrări.

Datorită faptului că există așa de multe posibilități folosind doar două terminale, există mai multe tipuri de porți logice cu două intrări. Vom prezenta mai jos fiecare tip.

### 3.2.1 Poarta logică ȘI (AND)

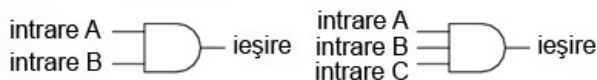


Figure 11: poarta logică ȘI cu două respectiv trei intrări; simbol

Una dintre cele mai ușor de înțeles porți este poarta ȘI. Denumirea vine de la faptul că ieșirea porții va fi 1 dacă și numai dacă toate intrările sunt 1. Asemenea, ieșirea va fi 0, dacă și numai dacă toate intrările sunt 0. Alăturat este prezentat simbolul porții ȘI cu două, respectiv trei intrări.

A	B	Ieșire
0	0	0
0	1	0
1	0	0
1	1	1

Tabelul de adevăr pentru poarta ȘI cu două intrări este conform tabelului alăturat.

Practic, ceea ce se înțelege din tabelul de adevăr de mai sus poate fi ilustrat în cele ce urmează. Poarta logică ȘI este supusă tuturor posibilităților de intrare. Pentru determinarea nivelului logic de ieșire, se folosește un LED:

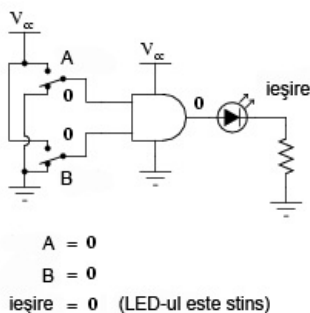


Figure 12: exemplificarea tabelului de adevăr a porții logice ȘI printr-un circuit practic

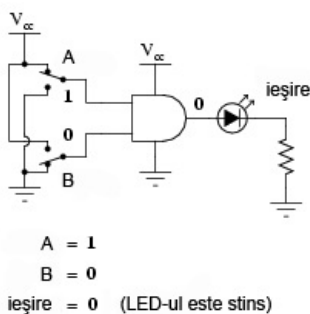


Figure 13: exemplificarea tabelului de adevăr a porții logice ȘI printr-un circuit practic

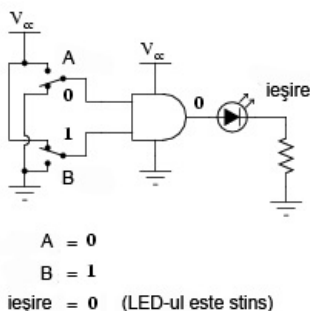


Figure 14: exemplificarea tabelului de adevăr a porții logice ȘI printr-un circuit practic

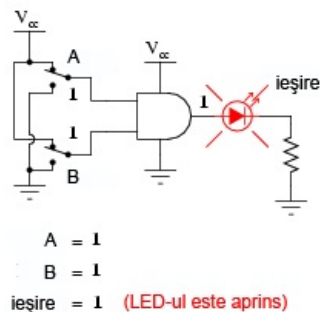


Figure 15: exemplificarea tabelului de adevăr a porții logice ȘI printr-un circuit practic

LED-ul este alimentat cu energie electrică doar atunci când ambele intrări logice sunt 1.

### 3.2.2 Poarta logică ȘI-negat (NAND)



Figure 16: circuitul echivalent al unei porți logice ȘI negat (NAND)

Poarta ȘI negat este o variație a porții ȘI. Practic, comportamentul porții este același ca al porții ȘI, doar că la ieșire este conectată o poartă NU (inversoare).

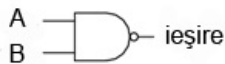


Figure 17: poarta logică ȘI negat (NAND); simbol

Pentru simbolizarea acestui lucru se trece un mic cerculeț pe terminalul de ieșire.

A	B	Ieșire
0	0	1
0	1	1
1	0	1
1	1	0

Tabelul de adevăr este exact invers față de cel prezentat anterior pentru poarta ȘI.

După câte se poate observa, principiul este asemănător: ieșirea este 1 dacă toate intrările sunt 0 și invers.

### 3.2.3 Poarta logică SAU (OR)

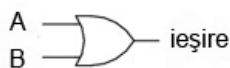


Figure 18: poarta logică SAU (OR); simbol

Ieșirea unei porți logice SAU este 1 dacă oricare dintre intrări este 1. Ieșirea este 0 doar dacă toate intrările sunt 0.

A	B	Ieșire
0	0	0
0	1	1
1	0	1
1	1	1

Tabelul de adevăr este cel alăturat.

Următoarele ilustrații redau modul de funcționare a porții SAU, atunci când cele două intrări formează toate combinațiile posibile.

Indicația vizuală a ieșirii este furnizată de un LED:

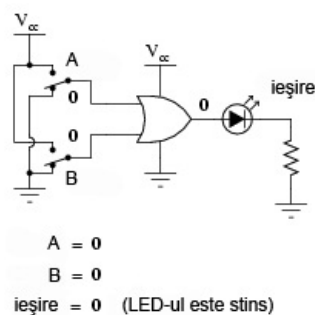


Figure 19: exemplificarea tabelului de adevăr a porții logice SAU printr-un circuit practic

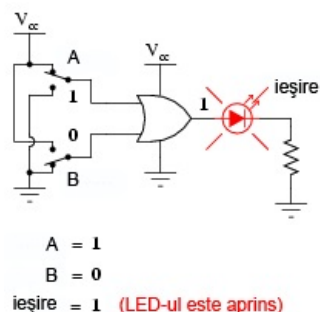


Figure 20: exemplificarea tabelului de adevăr a porții logice SAU printr-un circuit practic

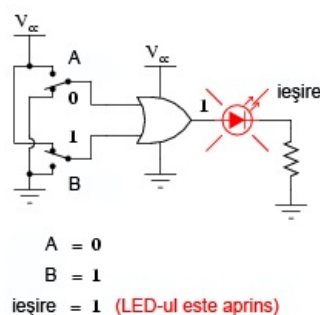


Figure 21: exemplificarea tabelului de adevăr a porții logice SAU printr-un circuit practic

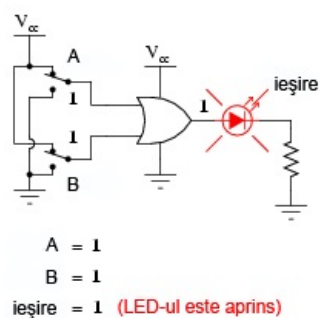


Figure 22: exemplificarea tabelului de adevăr a porții logice SAU printr-un circuit practic

Dacă oricare dintre intrări se află în poziția 1, LED-ul va fi alimentat cu energie electrică.

### 3.2.4 Poarta logică SAU negat (NOR)

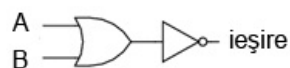


Figure 23: schema echivalentă a unei porți SAU negate (NOR)

După cum probabil v-ați dat seama, poarta SAU negată este o partă SAU cu valoarea de ieșire negată (0 negat este 1, iar 1 negat este 0). Schema echivalentă este cea din figura alăturată.

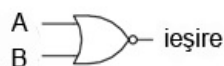


Figure 24: simbolul unei porți SAU negate (NOR)

Pentru simplificarea reprezentării însă, există desigur un simbol special, conform figurii alăturate.

A	B	Ieșire
0	0	1

0	1	0
1	0	0
1	1	0

Tabelul de adevăr este exact invers față de cel al porții SAU.

Principiul de bază este următorul: ieșirea este zero dacă cel puțin una dintre intrări este 1 și este 1 doar atunci când ambele intrări sunt 0.

### 3.2.5 Poarta logică ȘI negativă

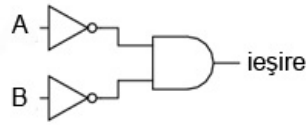


Figure 25: schema echivalentă a unei porți logice ȘI negative

O poartă logică ȘI negativă funcționează la fel ca o poartă ȘI având toate intrările inversate (conectate la porți NU).

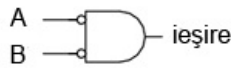


Figure 26: simbolul unei porți logice ȘI negative

Conform standardului de notare, aceste intrări sunt simbolizate cu ajutorul unor cerceulețe.

A	B	Ieșire
0	0	1
0	1	0
1	0	0
1	1	0

Contrar intuiției, comportamentul logic al unei porți ȘI negative nu este același cu al unei porți ȘI negate. De fapt, tabelul său de adevăr este identic cu al unei porți logice SAU negate.

### 3.2.6 Poarta logică SAU negativă

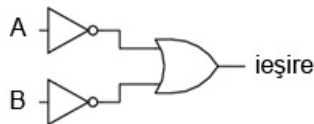


Figure 27: schema echivalentă a unei porți logice SAU negative

Conform aceluiași principiu, o poartă logică SAU negativă se comportă asemenea unei porți SAU cu toate intrările inversate.

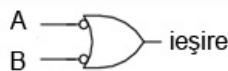


Figure 28: simbolul unei porți logice SAU negative

Conform standardului de notare, aceste intrări inversate sunt simbolizate prin cerceulețe.

A	B	Ieșire
0	0	1
0	1	1
1	0	1
1	1	0

Comportamentul logic și tabelul de adevăr este exact același cu al unei porți logice ȘI-negat.

### 3.2.7 Poarta logică SAU-exclusiv (XOR)



Figure 29: simbolul unei porți logice SAU-exclusiv

Ultimele șase variante de porți logice au fost variații directe ale celor trei funcții de bază: ȘI, SAU și NU. Poarta SAU-exclusiv este

însă diferită.

A	B	Ieșire
0	0	0
0	1	1
1	0	1
1	1	0

Ieșirea este 1 doar dacă intrările se află la nivele logice diferite, fie 0 și 1, fie 1 și 0. Altfel, ieșirea este 0 dacă toate intrările se află la același nivel logic.

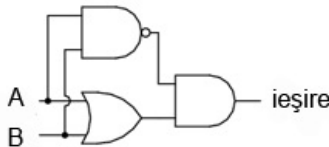


Figure 30: schema echivalentă a unei porți logice SAU-exclusiv formată din porți ȘI, SAU și NU

Circuitele echivalente pentru o poartă SAU-exclusiv sunt formate din porți ȘI, SAU și NU. O metodă directă de simularea a unei porți SAU-exclusiv este prin introducerea în circuit pentru început a unei porți SAU. Apoi adăugăm porți astfel încât să împiedicăm o valoare de 1 pe ieșire atunci când ambele intrări sunt 1.

Putem verifica faptul că tabelul de adevăr al circuitului echivalent de mai sus este același cu tabelul de adevăr prezentat inițial. În acest circuit, poarta ȘI de ieșire se comportă ca un repetor (memorie) pentru poarta SAU atunci când ieșirea porții ȘI-negat este 1. Acest lucru se întâmplă pentru primele trei combinații (00, 01 și 10). Totuși, atunci când ambele intrări sunt 1, ieșirea porții SAU-negat este 0, forțând o valoare de 0 pe ieșirea porții SAU.

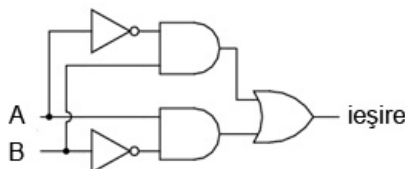


Figure 31: schema echivalentă a unei porți logice SAU-exclusiv formată din porți ȘI, SAU și NU

Un alt circuit echivalent pentru o poartă SAU-exclusiv este format din două porți ȘI negate (cu ajutorul unei porți NU). Acestea generează la ieșire o valoare de 1 dacă intrările sunt 01, respectiv 10. O poartă finală SAU permite o ieșire de 1 dacă cel puțin o poartă ȘI are o ieșire de 1.

Porțile SAU-exclusiv sunt utilizate în circuitele unde este necesară o comparație bit cu bine a două sau mai multe numere binare.

### 3.2.8 Poarta logică SAU-negat-exclusiv (XNOR)

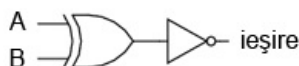


Figure 32: schema echivalentă a unei porți logice SAU-negat-exclusiv

Ultima poartă pe care o vom analiza este poarta SAU-negat-exclusiv. Aceasta este echivalentă cu poarta SAU-exclusiv, doar că ieșirea este inversată.

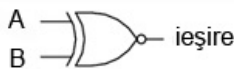


Figure 33: simbolul unei porți logice SAU-negat-exclusiv

Desigur, și pentru această poartă există un simbol special.

A	B	Ieșire
0	0	1
0	1	0
1	0	0
1	1	1

Și, în sfârșit, să vedem cum arată tabelul de adevăr pentru această poartă.

Așa cum reiese din acest tabel, scopul unei porți logice SAU-negat-exclusiv este de a genera un nivel logic 1 atunci când ambele intrări sunt la același nivel (fie 00, fie 11).

## 3.3 Principiul universalității

Porțile logice ȘI-negat și SAU-negat posedă o proprietate specială: sunt universale. Cu alte cuvinte, având un număr suficient de astfel de porți, fiecare din ele poate simula modul de funcționare al oricărei alte porți. De exemplu, putem construi un circuit care să

se comporte precum o poartă SAU, folosind trei porți ȘI-negat interconectate. Această abilitate este caracteristică doar acestor două tipuri de porți. Practic, multe sisteme de control digital sunt construite doar cu ajutorul porților ȘI-negat și SAU-negat, toate funcțiile logice necesare fiind derivate prin interconectarea acestor tipuri de porți. Vom lua mai jos câteva astfel de exemple.

### 3.3.1 Realizarea funcției NU

Să revedem prima dată simbolul și tabelul de adevăr pentru poarta NU:

Intrare	Ieșire
0	1
1	0



Figure 34: poartă logică NU



Figure 35: funcția logică NU realizată cu porți logice ȘI-negat și SAU-negat prin interconectarea intrărilor

În figura alăturată este prezentat modul de realizare a acestei funcții folosind porți logice ȘI-negat și SAU-negat.

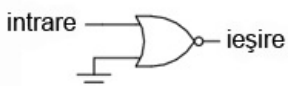


Figure 36: funcția logică NU realizată cu porți logice ȘI-negat și SAU-negat prin legarea uneia dintre intrări la masă

Această metodă de conectare împreună a intrărilor duce la creșterea curentului de intrare. Prin urmare, atât în cazul de față, cât și în exemplele ce urmează, se va folosi conectarea la masă a unuia dintre terminali (celălalt terminal de intrare va fi legat la sursa de alimentare). Funcțional, rezultatul este același.

### 3.3.2 Realizarea funcției ne-inversoare (buffer)

Să revedem prima dată simbolul și tabelul de adevăr pentru o poartă ne-inversoare:

Intrare	Ieșire
0	0
1	1



Figure 37: simbolul unei porți ne-inversoare

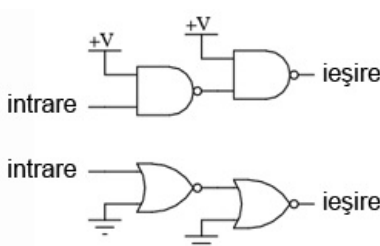


Figure 38: realizarea funcției ne-inversoare cu două etaje de porți ȘI-negat și SAU-negat conectate împreună

Conform celor spuse mai sus, realizarea acestei funcții folosind porți logice ȘI-negat și SAU-negat se realizează conectând două etaje împreună, conform figurii alăturate.

### 3.3.3 Realizarea funcției ȘI

Simbolul și tabelul de adevăr al porții logice ȘI:



A	B	Ieșire
0	0	0
0	1	0
1	0	0
1	1	1



Figure 39: simbolul porții logice ȘI

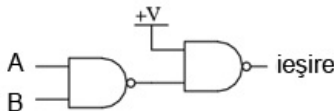


Figure 40: realizarea funcției ȘI prin intermediul a două porți logice ȘI-negat

Folosind porți logice ȘI-negat pentru realizarea funcției ȘI, avem nevoie de adăugarea unui etaj inversor (poartă NU) pe ieșirea porții ȘI-negat. Dar, am văzut mai sus cum se poate realiza o poartă NU folosind o poartă ȘI-negat. Prin urmare, schema finală este cea din figura alăturată.

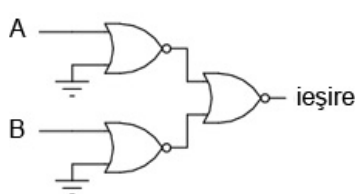


Figure 41: realizarea funcției ȘI prin intermediul a două porți logice ȘI-negat

Același lucru se poate realiza folosind porți logice SAU-negat, prin inversarea (poartă NU) tuturor intrărilor printr-o poartă SAU-negat. Din nou, am văzut mai sus cum se poate realiza o poartă NU dintr-o poartă SAU-negat.

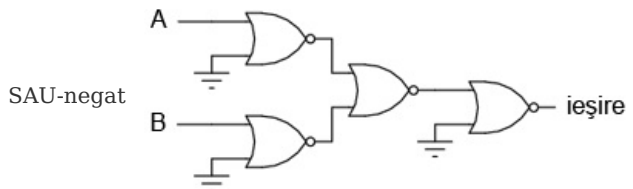
### 3.3.4 Realizarea funcției ȘI-negat

A	B	Ieșire
0	0	1
0	1	1
1	0	1
1	1	0



Figure 42: simbolul unei porți logice ȘI-negat

Desigur, nu avem ce „construi” la o funcție ȘI-negat cu ajutorul porților ȘI-negat, pentru că nu este nimic de făcut.



Cu ajutorul porților SAU-negat însă, va trebui să inversăm atât intrările cu o poartă SAU-negat, precum și ieșirea acestora din urmă (cu o poartă NU). Din nou, am văzut mai sus cum se poate realiza o poartă NU cu ajutorul porții SAU-negat.

### 3.3.5 Realizarea funcției SAU

A	B	Ieșire
0	0	0
0	1	1
1	0	1
1	1	1

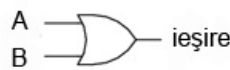


Figure 43: simbolul porții logice SAU

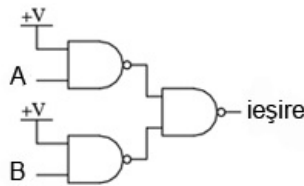


Figure 44: realizarea funcției logice SAU cu ajutorul porților logice SAU-negat

Inversarea ieșirii unei porți SAU-negat (cu ajutorul unei alte porți SAU-negat conectată ca și poartă NU) are ca rezultat funcția SAU.

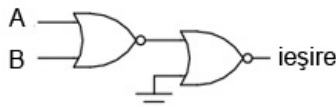


Figure 45: realizarea funcției logice SAU cu ajutorul porților logice SAU-negat

Folosind porți SAU-negat, trebuie să inversăm toate intrările pentru simularea funcției SAU, la fel cum a trebui să inversăm toate intrările unei porți SAU-negat pentru a obține funcție ȘI.

Țineți minte că inversarea tuturor intrărilor unei porți rezultă în schimbarea funcției esențiale ale acesteia. Astfel, poarta ȘI devine SAU, iar poarta SAU devine ȘI, plus o ieșire inversată. Astfel, cu toate intrările inversate, o poartă ȘI-negat se comportă precum o poartă SAU; o poartă SAU-negat se comportă precum o poartă ȘI; o poartă ȘI se comportă precum o poartă SAU-negat; și, în fine, o poartă SAU se comportă precum o poartă ȘI-negat. În cadrul algebrei booleene, aceste transformări sunt cunoscute sub numele de „teorema lui DeMorgan”.

### 3.3.6 Realizarea funcției SAU-negat

A	B	Ieșire
0	0	1
0	1	0
1	0	0
1	1	0

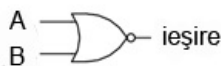


Figure 46: simbolul porții SAU-negat

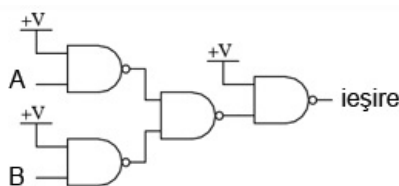


Figure 47: realizarea funcției SAU-negat prin intermediul porților ȘI-negat

Pentru realizarea acestei funcții folosind porți ȘI-negat, trebuie să inversăm toate intrările și ieșirea. Procedul este asemănător cu cel prin care am realizat funcția ȘI-negat folosind porți logice SAU-negat.

## 3.4 Modul de împachetare

Circuitele digitale cu porți logice sunt confecționate ca și circuite integrate: toți tranzistori și rezistorii ce intră în componența circuitului sunt construiți pe o singură bucată de material semiconductor. Prin urmare, dacă avem nevoie de un număr relativ de porți logice, putem folosi circuite integrate sub forma capsulelor DIP. Aceste circuite integrate sunt disponibile cu un număr par de pini, cele mai comune fiind cu 8, 14, 16, 18 sau 24 de pini

Numărul de catalog al acestor capsule indică numărul și tipul porților conținute în pachet. Aceste numere de catalog sunt standardizate, ceea ce înseamnă că un circuit „74LS02” produs de Motorola este identic ca și funcționalitate cu un circuit „74LS02” produs de Fairchild sau de oricare alt producător. Codul de litere ce precedă aceste numere de catalog sunt însă unice fiecărui producător în parte. De exemplu „SN74LS02” reprezintă o capsulă cu patru porți logice SAU-negat, produsă de Motorola. Un „DM74LS02” este același circuit din punct de vedere funcțional, dar produs de Fairchild.

Mai jos sunt date ca și referință câteva capsule DIP dintre cele mai utilizate:

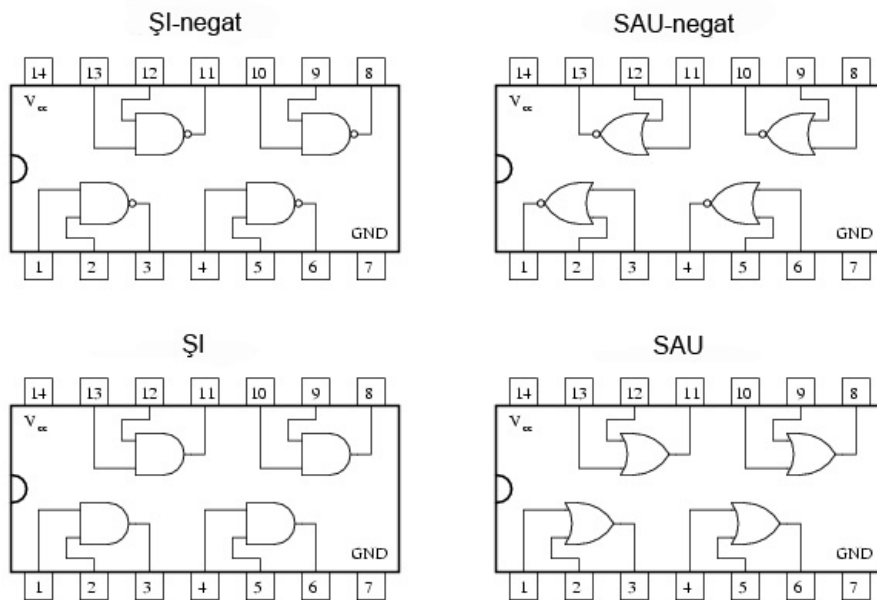


Figure 48: capsule DIP

## 4 Comutatoare

### 4.1 Tipuri de comutatoare

Un comutator electric este orice dispozitiv folosit pentru întreruperea deplasării electronilor prin circuit. Comutatoarele sunt practic dispozitive binare: fie sunt complet închise, fie complet deschise. Există o multitudine de tipuri de comutatoare. Vom prezenta câteva din ele mai jos.

Deși pare ciudat să prezentăm acest subiect elementar așa de târziu, în capitolele următoare vom explora un domeniu mai vechi al tehnologiei digitale. Aceasta din urmă se bazează pe contacte realizate cu ajutorul comutatoarelor mecanice și nu pe circuite digitale cu porți. Prezentarea ambelor metode conduce la o mai bună înțelegere a subiectului de față. Acest lucru ne va fi de folos atunci când vom învăța despre algebra booleană, matematica din spatele circuitelor logice digitale.

Cel mai simplu tip de comutator este acela în care contactul dintre doi conductori electrici se realizează prin acționarea unui mecanism de deplasare. Există și comutatoare mult mai complexe, comutatoare conținând circuite electronice capabile să închidă sau să deschidă circuitul în funcție de un stimul fizic (precum lumină sau câmp magnetic). Indiferent de caz, rezultatul final al unui comutator este o pereche de terminali ce vor fi conectați prin intermediul mecanismului intern al aparatului (comutator închis), fie vor fi separați (comutator deschis).

Orice comutator proiectat să fie folosit de către un operator uman, poartă numele de comutator manual. Există mai multe tipuri de astfel de comutatoare.

#### 4.1.1 Comutator basculant



Figure 49: comutator basculant; simbol

Aceste tipuri de comutatoare sunt acționate cu ajutorul unei manete. Această manetă se poate regăsi într-una dintre cele două sau mai multe poziții disponibile (în funcție de tip). Comutatorul obișnuit folosit pentru aprinderea și stingerea luminii în casă, este un bun exemplu de comutator basculant. Majoritatea comutatoarelor basculante se pot regăsi în oricare dintre poziții. Unele comutatoare sunt însă echipate cu un mecanism intern prin intermediul căruia maneta revine tot timpul într-o poziție normală, bine stabilită. Funcționarea (închiderea sau deschiderea circuitului, în funcție de caz) comutatorului se face doar pentru o perioadă scurtă de timp, după care acesta revine la poziția inițială.

#### 4.1.2 Comutator buton



Figure 50: comutator buton; simbol

Comutatoarele buton sunt dispozitive bi-poziționale acționate prin intermediul unui buton care apăsă iar apoi eliberat. Majoritatea butoanelor posedă un mecanism intern prin care butonul se reîntoarce la poziția sa inițială (ne-apăsă). Prin urmare, acest dispozitiv funcționează doar pe perioada în care butonul este apăsă, revenind apoi la poziția sa inițială. Un bun exemplu de astfel de comutator este butonul de pornire al calculatorului, sau de chemare al liftului. După apăsare, acestea revin la poziția inițială. Unele comutatoare pot rămâne în poziția apăsă până când acesta este tras înapoi. Aceste tipuri de comutatoare sunt prevăzute cu un buton de tip ciperă pentru ușurarea acțiunii.

#### 4.1.3 Comutator selector



Figure 51: comutator selector; simbol

Comutatoarele selectoare sunt acționate prin intermediul unui buton rotativ pentru selectarea uneia sau a mai multor poziții. La fel ca și comutatoarele basculante, acestea se pot regăsi în oricare dintre poziții, sau pot conține mecanisme pentru funcționarea de scurtă durată (revenirea la poziția normală).

#### 4.1.4 Comutator joystick

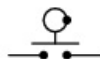


Figure 52: comutator joystick; simbol

Un comutator joystick este acționat prin intermediul unei manete cu un grad de libertate sporit. În funcție de direcția de deplasare a manetei, există unul sau mai multe mecanisme de contact ce intră în acțiune. Câteodată, acest lucru depinde și de distanța de deplasare a manetei. Cercul și punctul din simbolul comutatorului indică direcția de deplasare a manetei pentru acționarea contactului. Aceste tipuri de comutatoare sunt folosite de obicei pentru macarele și pentru control industrial.

#### 4.1.5 Comutatoare de limitare

Unele comutatoare sunt special concepute pentru acționarea lor nu de către un operator uman, ci de către mișcarea unui dispozitiv mecanic. Aceste comutatoare de mișcare poartă numele de comutatoare de limitare, datorită faptului că sunt folosite pentru limitarea deplasării unei mașini. Acest lucru se realizează prin întreruperea alimentării unui anumit component, dacă acesta se deplasează prea departe. La fel ca în cazul comutatoarelor manuale, există mai multe tipuri de comutatoare de limitare.

#### 4.1.6 Comutator de limitare cu manetă



Figure 53: Comutator de limitare cu manetă; simbol

Aceste limitatoare sunt asemănătoare comutatoarelor basculante sau selectoare. În cazul acestora însă, maneta este acționată de un dispozitiv mecanic, și nu de către un operator uman.

#### 4.1.7 Comutator de proximitate

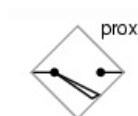


Figure 54: Comutator de proximitate; simbol

Comutatoarele de proximitate detectează apropierea unei părți metalice, fie prin intermediul unui câmp magnetic, fie prin intermediul unui câmp electromagnetic de frecvență înaltă. Comutatoarele de proximitate simple utilizează un magnet permanent pentru acționarea unui mecanism întrerupător atunci când componenta metalică se apropie prea mult (2-3 cm). Comutatoarele de proximitate mai complexe funcționează asemenea unui detector de metale, alimentând o bobină cu un curent de frecvență înaltă și măsurând electronic amplitudinea aceluia curent. Dacă o componentă metalică (nu neapărat magnetică) se apropie prea mult de bobină, curentul va crește și va acționa mecanismul de monitorizare a circuitului. Simbolul alăturat este al unui comutator de proximitate de tip electronic, indicat prin romb. Simbolul unui dispozitiv non-electric este același cu simbolul comutatorului de limitare cu manetă.

O altă variantă a comutatorului de proximitate o reprezintă comutatorul optic. Acesta este compus dintr-o sursă de lumină și un element fotosensibil. Poziția elementului mecanic (mașinii) este detectată fie prin întreruperea sau reflexia unei de lumină. Comutatoarele optice sunt folosite în aplicații de siguranță, unde o sursă de lumină poate fi folosită pentru detectarea intrării persoanelor neautorizate într-o zonă periculoasă.

#### 4.1.8 Comutatoare de proces

În multe aplicații industriale, este necesară o monitorizare a diferitelor mărimi fizice cu ajutorul comutatoarelor. Astfel de dispozitive pot fi folosite pentru pornirea unei alarme, indicând faptul că variabila de proces a depășit parametrii normali de funcționare. Sau pot fi folosite pentru oprirea proceselor sau a echipamentelor dacă acele variabile au atins un nivel periculos sau distructiv. Desigur, există mai multe variante de astfel de comutatoare de proces, prezentate mai jos.

#### 4.1.9 Comutator de viteză



Figure 55: Comutator de viteză; simbol

Aceste comutatoare pot detecta viteza de rotație a unui ax prin intermediul unui mecanism montat pe acesta. Desigur, acest lucru

este de preferat a se realiza fără un contact fizic între ax și comutator, caz în care detecția se realizează optic sau magnetic.

#### 4.1.10 Comutator de presiune



Figure 56: Comutator de presiune; simbol

Presiunea gazului sau a lichidului poate fi utilizată pentru acționarea unui mecanism de comutare. Această presiune trebuie să fie aplicată unui piston sau unei diafragme, care la rândul ei va converti presiunea în forță mecanică.

#### 4.1.11 Comutator de temperatură



Figure 57: Comutator de temperatură; simbol

Un mecanism relativ ieftin de detectare a temperaturii constă dintr-o bandă bimetalică: o bandă subțire formată din două metale diferite poziționate spate-în-spate. Fiecare metal posedă un coeficient de dilatare termică diferit. Această dilatare termică nu este altceva decât tendința corpurilor de a-și crește volumul în urma variației temperaturii (de obicei în urma creșterii acesteia, dar există și excepții). Fenomenul opus poartă numele de contracție termică.

Atunci când banda se răcește sau se încălzește, coeficienții de dilatare diferiți ale celor două metale cauzează curbarea acetei benzi. Curbarea benzii poate fi folosită apoi pentru acționarea unui mecanism de comutare.

Alte comutatoare de temperatură utilizează un bec de alamă umplut fie cu gaz fie cu lichid, și un tub ce conectează acest bec de un comutator de presiune. Pe măsură de becul este încălzit, volumul gazului sau al lichidului crește, generând o creștere de presiune care mai apoi acționează mecanismul de comutare.

#### 4.1.12 Comutator de nivel

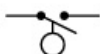


Figure 58: Comutator de nivel de lichid; simbol

Un obiect plutitor poate fi folosit pentru acționarea unui mecanism atunci când nivelul de lichid dintr-un bazin trece peste un anumit nivel. Dacă lichidul este conductor din punct de vedere electric, acesta poate fi folosit ca și conductor pentru închiderea circuitului dintre două sonde metalice. Acestea sunt plasate în bazin la adâncimea corespunzătoare. În majoritatea cazurilor însă, acest lucru nu este practic, ba mai mult, este chiar periculos.

Acest tip de comutatoare poate fi folosit și pentru detectarea nivelului materialelor solide, precum rumeguș, grâu, cărbune sau plante furajere. O metodă des întâlnită pentru această aplicație este utilizarea unei mici roți cu pale metalice sau din lemn, plasată în interiorul recipientului la înălțimea dorită. Aceasta roată este conectată la un motor electric ce o rotește cu o anumită viteză. Atunci recipientul este umplut cu material solid până la acel nivel, materialul blochează roata și împiedică rotirea ei. Răspunsul motorului este cel care acționează mecanismul de comutare.

O altă metodă utilizează un diapazon (instrument format dintr-o bară metalică în formă de U, ce vibrează la lovire). Acesta este introdus în recipient din exterior, la înălțimea dorită. Diapazonul este supus unei vibrații la frecvența de rezonanță prin intermediul unui circuit electronic și un magnet/electromagnet. Când materialul solid trece de înălțimea la care este montat diapazonul, acesta va atenua vibrațiile diapazonului. Această modificare a amplitudinii vibrațiilor și/sau frecvenței este detectată de circuitul electronic.

##### 1. Comutator nuclear

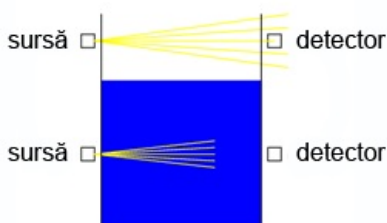


Figure 59: Comutator de nivel nuclear; simbol

O ultimă metodă de realizare a unui comutator de nivel pe care o luăm aici în considerare, îl reprezintă comutatorul nuclear. Acesta este compus dintr-un material radioactiv ca și sursă și un detector de radiație. Ambele elemente sunt montate în lungul diametrului recipientului pentru lichid sau pentru material solid.. Dacă înălțimea materialului trece de nivelul mecanismului sursă/detector, acesta va atenua puterea recepționată de detectorul de radiație. Această descreștere a radiației pe detector poate fi folosită pentru acționarea unui mecanism de comutare, fie pentru măsurarea nivelului, fie pentru declanșarea unei alarme sau chiar și pentru controlul nivelului din recipient.

Atât sursa cât și detectorul sunt montate în exteriorul recipientului, singurele elemente ce pătrung în interior sunt radiațiile. Sursele de radiație sunt extrem de slabe și nu prezintă niciun risc imediat operatorilor sau personalului de întreținere.

#### 4.1.13 Comutator de curgere



Figure 60: Comutator de curgere; simbol

Introdus într-o conductă, un comutator de curgere va detecta viteza de curgere a unui gaz sau a unui lichid. În momentul în care această viteză depășește o anumită limită, se va acționa mecanismul de comutare. De obicei se folosesc pale sau aripi ce sunt împinse de curgerea substanței respective. O metodă alternativă constă în detectarea căderii de presiune pe o anumită porțiune a conductei.

#### 4.1.14 Observație

Desigur, există tot timpul mai multe metode de implementare a unui comutator pentru monitorizarea sau controlul unui proces fizic. De obicei nu există un singur comutator „perfect” pentru nicio aplicație, deși unele prezintă câteva avantaje clare față de altele. Comutatoarele trebuie alese inteligent în funcție de aplicația în cauză. Acest lucru va determina funcționarea lor eficientă și sigură.

## 4.2 Poziția „normală” a contactelor

Orice tip de comutator poate fi proiectat astfel încât contactele sale să se închidă (stabilirea continuității circuitului), sau să se deschidă (întreruperea continuității), atunci când este acționat. Pentru comutatoarele prevăzute cu un mecanism de re-întoarcere la poziția inițială, direcția de re-întoarcere a comutatorului, atunci când nu este aplicată nicio forță externă, poartă numele de poziție normală. Prin urmare, contactele ce sunt deschise în poziția normală, poartă numele de „normal-deschise”. Contactele ce sunt închise în poziția normal, poartă numele de „normal-închise”.

Pentru comutatoarele de proces, poziția normală, este acea poziție în care nu există nicio influență de proces asupra comutatorului. O metodă simplă de determinare a poziției normale a unui comutator de proces, constă în determinarea poziției comutatorului atunci când acesta nu a fost încă instalat. Să luăm câteva exemple de poziții normale de proces. Comutator de viteză: axul este staționar; comutator de presiune: presiunea aplicată este zero; comutator de temperatură: temperatură ambientală (temperatura camerei); comutator de nivel: recipient gol; comutator de curgere: viteza de curgere a lichidului este zero.

Este important să facem diferența între poziția „normală” a comutatorului și funcționarea sa „normală” într-un proces. Să considerăm exemplul unui comutator de curgere ce este utilizat pentru semnalizarea (pe cale sonoră sau vizuală) scăderii debitului de apă dintr-un sistem de răcire. Funcționarea normală a sistemului de răcire constă într-un debit constant și suficient de lichid de răcire prin conducte. Să presupunem că urmărim închiderea contactelor comutatorului în cazul pierderii de lichid de răcire (pentru completarea unui circuit electric ce activează alarma vizuală sau auditivă, de exemplu). În acest caz, va trebui să folosim un comutator de curgere cu contacte în poziția normal-închis și nu în poziția normal-deschis. În momentul în care există un debit normal și suficient de lichid prin conducte, contactele comutatorului rămân în poziția deschis. Atunci când debitul lichidului scade sub un anumit nivel critic, contactele se re-întorc în poziția lor normală, și anume, normal-închis. Gândiți-vă tot timpul la starea normală a unui comutator ca la acea stare în care se regăsește dispozitivul când nu este introdus în sistem (este încă în magazin, de exemplu).

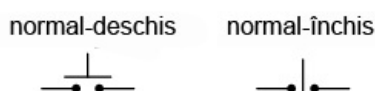


Figure 61: simbolul comutatorului tip buton în poziția normal-deschis, respectiv normal-închis

Simbolul comutatoarelor este diferit în funcție de scopul și de modul de acționare. Un comutator normal-deschis este reprezentat în așa fel încât să reprezinte un contact deschis, care în momentul acționării să se închidă. Invers, un comutator normal-închis este reprezentat ca și un contact închis ce se deschide la acționarea dispozitivului. Alăturat este reprezentat simbolul comutatorului tip buton în poziția normal-deschis, respectiv normal-închis.

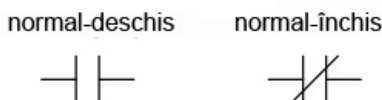


Figure 62: simbolul general al comutatorului în poziție normal-deschis, respectiv normal-închis

Există de asemenea un simbol generic pentru poziția oricărui tip de comutator, fie normal-deschis, fie normal-închis, asemeni figurii alăturate. Comutatorul din stânga se închide când este acționat și este deschis în poziția normală (atunci când nu este acționat din exterior). Comutatorul din dreapta se deschide când este acționat și este închis în poziția sa normală (când nu este acționat). Dacă se folosesc astfel de simboluri pentru reprezentarea comutatoarelor, tipul acestora este de obicei trecut în vecinătatea simbolului, în cuvinte.

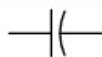


Figure 63: simbolul condensatorului folosit în circuitele logice digitale

Atenție, nu confundați simbolul stânga (în figura de mai sus) cu simbolul condensatorului. În cazul în care se dorește reprezentarea unui condensator într-un circuit logic, se va folosi simbolul alăturat. În electronica standard, acest simbol este rezervat condensatoarelor polarizate. În circuitele logice digitale, acest simbol este folosit pentru orice tip de condensator, chiar și în situația când condensatorul nu este un condensator polarizat.

#### 4.2.1 Secvența realizării contactelor

În cazul comutatoarelor cu mai multe poziții, trebuie luat în considerare și modul de deschidere și de închidere a contactelor, pe măsură ce selectorul se deplasează de pe o poziție pe alta.



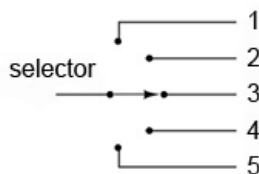


Figure 64: comutator cu întrerupere; realizarea contactelor

Selectorul comutatorului din figura alăturată realizează contactul (închiderea circuitului) într-una din cele cinci poziții diferite. Pozițiile sunt numerotate de la 1 la 5. Configurația cea mai des întâlnită a unui comutator cu pas reglabil, este aceea în care contactul cu o anumită poziție este deschis înainte de realizarea contactului (închiderea contactului) cu poziția următoare. Sub această configurație, comutatorul este cunoscut sub numele de comutator cu întrerupere. Ca și exemplu, să presupunem cazul în care comutatorul se află pe poziția 3. Dacă selectorul este întors în sensul acelor de ceasornic, acesta va deschide contactul 3, deschizând practic circuitul, și se va deplasa între poziția 3 și 4. În acest moment, ambele circuite (3 și 4) sunt deschise, ambele contacte fiind deschise. Dacă se continuă deplasarea selectorului în sensul acelor de ceasornic, se va ajunge în poziția 4, moment în care contactul se închide.

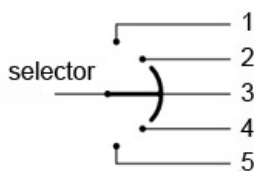


Figure 65: comutator cu suprapunere (comutator fără întrerupere); realizarea contactelor

Există însă situații în care nu este permisă în nicio clipă deschiderea completă a circuitului atașat selectorului. Pentru astfel de aplicații, există o altă variantă de comutator, denumită comutator cu suprapunere, sau comutator fără întrerupere. În acest caz, selectorul nu deschide circuitul precedent înainte de realizarea contactului cu circuitul următor (în sensul de rotire). În exemplu precedent, contactul 4 este realizat înaintea deschiderii contactului 3. Compromisul constă în faptul că circuitul trebuie să poată tolera asemenea contacte adiacente realizate simultan (1 cu 2, 2 cu 3, 3 cu 4 și 4 cu 5).

#### 4.2.2 Comutatoare multipolare

Când contactul(e) mobil poate fi adus pe unul dintre contactele fixe, acele poziții sunt denumite „direcții”. Numărul contactelor mobile poartă numele de poli. Ambele comutatoare prezentate mai sus cu un contact mobil și cinci contacte staționare pot fi desemnate ca și comutatoare monopolare cu cinci direcții.

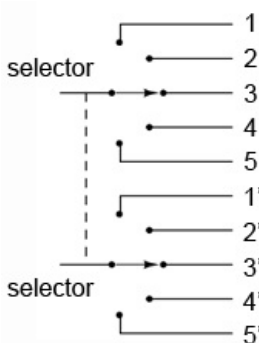


Figure 66: comutator bipolar cu cinci direcții

Să presupunem că două comutatoare bipolare cu cinci direcții sunt legate împreună, astfel încât ele să fie acționate prin intermediul aceluiași mecanism. Întregul dispozitiv astfel format poartă numele de comutator bipolar cu cinci direcții. Simbolul unui astfel de comutator este prezentat în figura alăturată. De menționat că linia întreruptă trasată între cele două selectoare, desemnează faptul că acestea sunt acționate simultan de același mecanism extern.

Să luăm și alte exemple de comutatoare:



Figure 67: comutator monopolar cu o direcție

Comutator monopolar cu o direcție

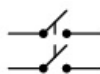


Figure 68: comutator bipolar cu o direcție

Comutator bipolar cu o direcție



Figure 69: comutator monopolar cu două direcții

Comutator monopolar cu două direcții

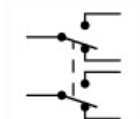


Figure 70: comutator bipolar cu două direcții

Comutator bipolar cu două direcții



Figure 71: comutator cvadripolar cu patru direcții

Comutator cvadripolar cu patru direcții

## 5 Relee electromecanice

### 5.1 Construcția releelor

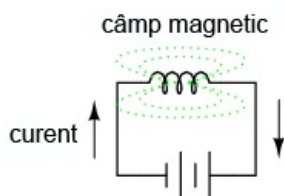


Figure 72: o bobină produce un câmp magnetic la trecerea curentului prin aceasta

La trecerea curentului electric printr-un conductor, va lua naștere un câmp magnetic în jurul acestuia. În cazul în care conductorul este construit sub forma unei bobine, câmpul magnetic produs se va orienta în lungimea bobine. Cu cât intensitatea curentului este mai mare, cu atât puterea câmpului magnetic este mai mare, toți ceilalți factori rămânând neschimbați.

Bobinele reacționează la variația curentului prin ele datorită energiei stocate sub forma acestui câmp magnetic. La construirea unui transformator din două bobine înfășurate în jurul unui miez magnetic comun, utilizăm de fapt acest câmp magnetic pentru a transfera energie electrică de la o înfășurare la alta. Totuși, există și alte metode mai simple și mai directe de utilizare a câmpurilor electromagnetice. Câmpul magnetic produs de o bobină poate fi folosit pentru exercitarea unei forțe mecanice asupra oricărui obiect magnetic. În același fel folosim și magneți permanenți pentru atragerea obiectelor magnetice. Diferența constă în faptul că acest electromagnet (format din bobină) poate fi pornit și oprit prin închiderea și deschiderea circuitului bobinei.

Dacă plasăm un obiect magnetic (un obiect metalic, de exemplu) în apropierea unei astfel de bobine, acest obiect se va deplasa atunci când prin bobină trece un curent electric. Obiectul magnetic deplasabil poartă numele de armătură, iar majoritatea lor pot fi deplasate fie prin intermediul curentului continuu, fie cu ajutorul curentului alternativ. Polaritatea câmpului magnetic este irelevantă din punct de vedere al atracției armăturii. Aceste dispozitive pot fi folosite pentru deschiderea pe cale electrică a încuietorilor, deschiderea sau închiderea valvelor, deplasarea brațelor mecanice, etc. Totuși, în situația în care aceste dispozitive sunt utilizate pentru acționarea unui comutator, ele sunt cunoscute sub denumirea de relee electromecanice.

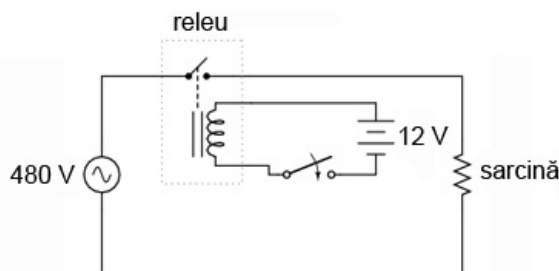


Figure 73: releu electromecanic într-un circuit electric

Releele sunt foarte practice pentru controlul unei cantități mari de curent sau tensiune prin intermediul unui semnal electric de putere mică. Bobina releului ce produce câmpul magnetic poate să consume o putere mai mică de 1 watt, de exemplu, pe când contactele acționate de acest câmp magnetic pot susține o putere de sute de ori mai mare. Funcțional, un releu electromecanic se comportă precum un amplificator cu două stări: pornit și oprit.

La fel ca în cazul tranzistorilor, abilitatea releelor de a controla un semnal electric prin intermediul unui alt semnal electric este utilizată pentru realizarea funcțiilor logice. Pentru moment însă, vom explora abilitatea de „amplificare” a releelor.

În figura de mai sus, bobina releului este energizată prin intermediul unei surse de tensiune de 12 V (c.c.). În schimb, comutatorul monopolar cu o singură direcție este conectat într-un circuit electric alimentat de la o sursă de tensiune de 480 V (c.a.). În acest caz, curentul necesar energizării bobinei este de sute de ori mai mic decât curentul nominal al contactului comutatorului.

Un singur dispozitiv bobină/armătură poate fi folosit pentru acționarea mai multor seturi de contacte. Aceste contacte pot fi normal-deschise, normal-închise, sau într-o combinație a celor două. Asemeni comutatoarelor, poziția „normal” a releelor reprezintă acea stare a contactelor atunci când bobina nu este energizată, sau mai bine spus, atunci când releul este încă „în cutie”.

Pe lângă abilitatea de control a unui curent mare prin intermediul unui curent mic, releele oferă și o izolație electrică între circuitul

bobine și circuitul contactelor. Acest lucru înseamnă că cele două elemente sunt izolate din punct de vedere electric una de cealaltă. Unul din circuite poate fi de c.c., iar celălalt de c.a., precum în exemplul anterior, sau chiar și la tensiuni diferite.

## 5.2 Contactoare

Atunci când releele sunt folosite pentru comutarea unor puteri mari prin contactele sale, acestea poartă numele de contactoare. Contactoarele au de obicei mai multe contacte, iar acestea sunt de obicei (deși nu neapărat) în poziția normal-deschis: circuitul este deschis atunci când bobina nu este energizată.

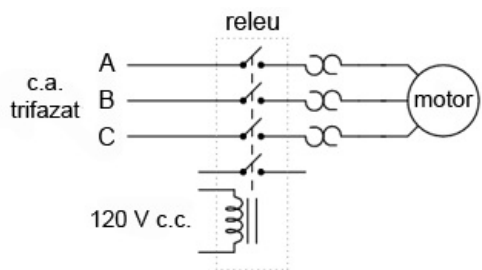


Figure 74: relee electromecanic într-un circuit trifazat pentru controlul unui motor electric

Probabil cea mai des întâlnită aplicație a contactoarelor constă în controlul motoarelor electrice.

## 5.3 Relee temporizate

Unele rele sunt prevăzute cu un fel de „amortizor”. Acest mecanism, atașat arăturii releului previne deplasarea imediată și completă a armăturii atunci când bobina este energizată sau de-energizată. Datorită acestui lucru, releul are proprietatea unei acționări temporizate (întârziate în timp). Astfel de rele temporizate pot fi construite pentru temporizarea deplasării armăturii în momentul energizării bobinei, în momentul de-energizării bobinei, sau pentru ambele situații.

Releele temporizate au ca și specificație nu doar poziția normal-închisă sau normal-deschisă a contactelor, ci și modul în care acționează temporizarea (la închiderea sau la deschiderea contactelor). Mai jos este o descriere a celor patru tipuri de rele temporizate.

### 5.3.1 Releu temporizat la închidere normal deschis

Acest tip de contact este normal deschis atunci când bobina nu este energizată. Contactul se închide doar după aplicarea unui curent electric prin bobina releului, și doar după o anumită perioadă de timp de la aplicarea acestuia. Cu alte cuvinte, direcția deplasării contactului este identică cu cea a unui contact normal deschis, dar există o întârziere (temporizare) la închiderea contactului. Datorită faptului că temporizarea are loc în direcția de energizare a bobinei, acest tip de contact mai poartă numele de normal-deschis cu acționare întârziată.

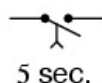


Figure 75: relee temporizat la închidere normal deschis

Releul din figura alăturată este un relee temporizat la închidere, normal-deschis. Acesta se închide după cinci secunde de la energizarea bobinei. Deschiderea se realizează imediat după de-energizarea bobinei.

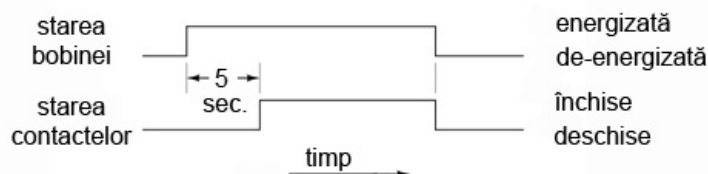


Figure 76: diagrama timp de funcționare a contactelor unui relee temporizat la închidere, normal deschis

Alăturat este o diagramă de timp ce prezintă modul de funcționare a contactelor acestui tip de relee temporizat.

### 5.3.2 Releu temporizat la deschidere, normal-deschis

Asemenea releului precedent, și acest dispozitiv este normal-deschis atunci când bobina este de-energizată. La aplicarea unui curent pe bobină, contactele releului se închid. Față de cazul precedent însă, temporizarea (întârzierea) are loc după de-energizarea bobinei și nu după energizarea ei. Datorită faptului că temporizarea are loc după de-energizarea bobinei, acest tip de contact mai poartă numele de normal deschis cu întârziere la revenire.

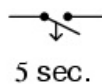


Figure 77: relee temporizat la deschidere normal deschis

Releul temporizat la deschidere, normal deschis, din figura alăturată se închide imediat după energizarea bobinei. Deschiderea contactelor are loc însă la cinci secunde după de-energizarea bobinei.

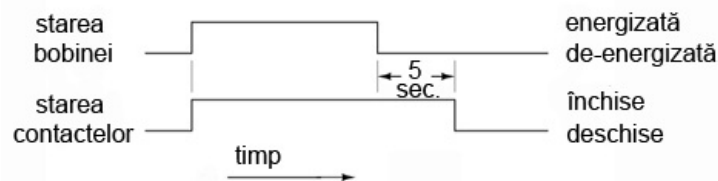


Figure 78: diagrama timp de funcționare a contactelor unui releu temporizat la deschidere, normal deschis

Diagrama de timp în acest caz, este cea din figura alăturată.

### 5.3.3 Releu temporizat la deschidere, normal-închis

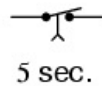


Figure 79: releu temporizat la deschidere normal închis

Acest tip de contact este normal închis atunci când bobina nu este energizată. Contactul se deschide la aplicarea unui curent prin bobină, dar doar după o anumită perioadă de timp. Cu alte cuvinte, direcția de deplasare a contactului este identică cu a unui contact normal-închis, doar că există o temporizare în direcția deschiderii acestuia. Datorită faptului că temporizarea are loc în direcția energizării bobinei, acest contact mai poartă numele de contact normal închis cu acționare întârziată.

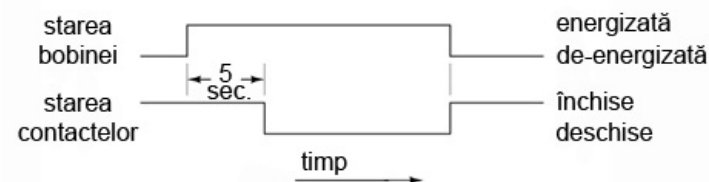


Figure 80: diagrama timp de funcționare a contactelor unui releu temporizat la deschidere, normal închis

### 5.3.4 Releu temporizat la închidere, normal-închis

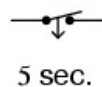


Figure 81: releu temporizat la închidere normal închis

Asemenea releului temporizat la deschidere normal-închis, acest tip de contact este normal-închis atunci când bobina nu este energizată. Deschiderea se realizează prin aplicarea unui curent prin bobina releului. Totuși, la de-energizarea bobinei, contactele se închid cu o anumită întârziere de timp. Acest tip de contact mai poartă numele de contact normal închis cu întârziere la revenire.

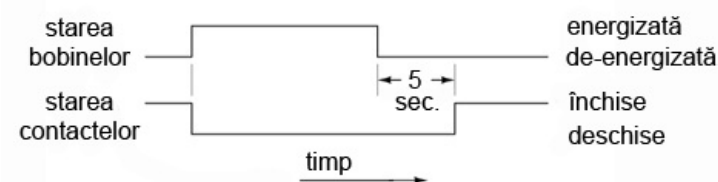


Figure 82: diagrama timp de funcționare a contactelor unui releu temporizat la închidere, normal închis

### 5.3.5 Alte tipuri de rele temporizate

Releele temporizate mecanice, mai vechi, utilizau amortizoare mecanice sau dispozitive cu pistoane/cilindrii umplute cu fluid pentru întârzierea deplasării armăturilor. Modelele mai nou utilizează circuite electronice cu rețele rezistor-condensator pentru generarea întârzierii. Energizarea (instantanee) a releului electromecanic se realizează cu ajutorul semnalului de ieșire al circuitului electronic. Aceste rele electronice sunt mai adaptabile decât variantele mecanice, și mult mai durabile. Multe modele sunt capabile de efectuarea unor operații de temporizare avansate:

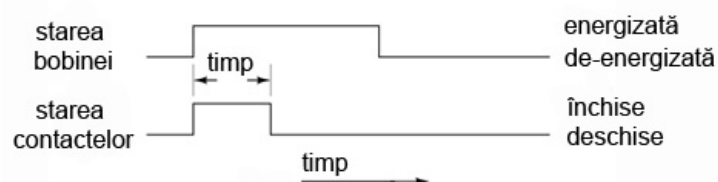


Figure 83: releu temporizat normal-deschis cu o singură închidere

Releu temporizat normal-deschis cu o singură închidere. Aceste rele se închid o singură dată, un anumit interval de timp și apoi se re-deschid, la o tranziție a intrării de la starea de-energizată la starea energizată.

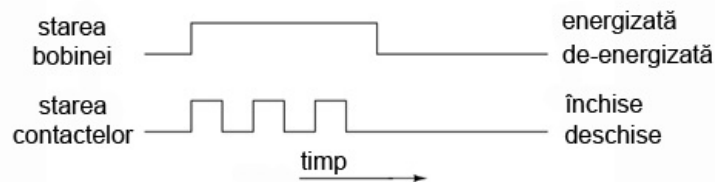


Figure 84: releu temporizat normal-deschis cu deschideri/închideri repetate

Releu temporizat normal-deschis cu deschideri/închideri repetate. Acest releu se închide și se deschide pentru un anumit interval de timp atâta timp cât bobina este energizată.

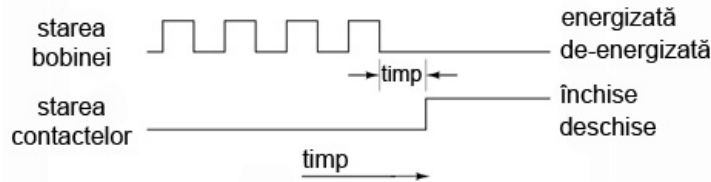


Figure 85: releu temporizat de supraveghere

Releu temporizat de supraveghere. Acest releu își schimbă starea în cazul în care semnalul de intrare nu variază continuu de la starea energizată la starea de-energizată.

Această ultimă metodă de temporizare este utilă pentru monitorizarea sistemelor bazate pe calculatoare. Dacă se folosește un calculator pentru controlul unui proces critic, este recomandată de obicei instalarea unui sistem automat de alarmare în cazul în care calculatorul se blochează din diferite motive. O metodă relativ simplă de instalare a unui astfel de sistem constă în energizarea și de-energizarea unei bobine prin intermediul unui semnal trimis de calculator. Dacă sistemul se blochează, semnalul de intrare pe releu nu se va mai modifica (se va bloca la ultima stare). La puțin timp după aceasta, contactul releului se va închide semnalizând o problemă.

## 5.4 Relee semiconductoare

Pe cât de utile sunt, releele electromecanice au totuși multe inconveniente. Acestea sunt relativ scumpe, au o durată de viață a contactelor limitată, ocupă mult loc, iar timpii de comutație sunt mari în comparație cu dispozitivele semiconductoare moderne. Aceste limitări se aplică în special releelor de putere. Pentru a întâmpina aceste neajunsuri, mulți producători oferă rele semiconductoare ce folosesc tiristori, triace sau tranzistori în loc de contactele mecanice.

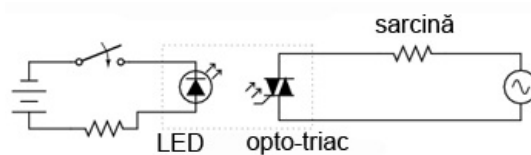


Figure 86: releu semiconductor

Aceste dispozitive de ieșire (tiristori, triace, tranzistori) sunt cuplate optic la o sursă de lumină (LED) în interiorul releului. Releul este pornit prin energizarea acestui LED, de obicei prin intermediul unei surse de tensiune de c.c. scăzute. Această izolare optică între intrare și ieșire se ridică la nivelul celor mai bune rele electromecanice posibile.

Fiind dispozitive semiconductoare, nu există părți mobile care să se deterioreze la uzură. Timpii de comutație sunt mult mai mici decât viteza maximă posibilă de deplasare a armăturilor în cazul releelor mecanice. Nu există pericolul apariției arcelor electrice între contacte și nici probleme corozive. Totuși, aceste dispozitive semiconductoare sunt mai scumpe, mai ales pentru curenți înalți. Un avantaj important al releelor cu tiristoare, este tendința naturală de deschidere a contactelor, într-un circuit de c.a., doar la trecerea curentului prin zero. Histereza „naturală” a tiristoarelor menține continuitatea circuitului chiar și după de-energizarea LED-ului, până în momentul în care c.a. scade sub un anumit prag (curentul de menținere). Practic, acest lucru înseamnă că circuitul nu se va întrerupe în mijlocul unei semi-perioade (atunci când valoarea curentului este maximă, de exemplu). O asemenea întrerupere într-un circuit cu o inductanță mare, va produce în mod normal o creștere mare și de scurtă durată tensiunii. Acest lucru se datorează „căderii” bruște a câmpului magnetic din jurul inductanței. Acest lucru nu are loc însă în cazul unui releu semiconductor echipat cu tiristori.

Un dezavantaj al releelor semiconductoare este tendința de scurt-circuitare în caz de defect. Releele electromecanice tind să se deschidă la defect. Dintre cele două stări, deschiderea la defect este considerată mai sigură față de scurt-circuitarea la defect. Din acest motiv, în anumite aplicații, releele electromecanice sunt încă folosite în dauna celor semiconductoare.

## 6 Logica ladder

### 6.1 Diagrame ladder

Diagramele ladder sunt diagrame speciale folosite de obicei în sistemele logice de control industrial. Denumirea de ladder (din engleză: scară) vine de la asemănarea acestora cu o scară: două linii verticale desemnând sursa de putere, și linii orizontale reprezentând circuitele de control.



Figure 87: diagrama ladder de control a unei lămpi prin intermediul unui comutator manual

Ca și exemplul de început, o diagramă ladder simplă reprezentând o lampă controlată de un comutator manual arată precum în figura alăturată. Notațiile  $L_1$  și  $L_2$  desemnează bornele unei surse de alimentare de 120 V c.a.  $L_1$  este faza iar  $L_2$  este conductorul neutru (legat la masă). Aceste notații nu au nicio legătură cu notația bobinelor.

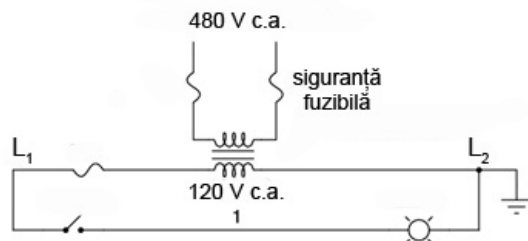


Figure 88: diagrama ladder de control a unei lămpi prin intermediul unui comutator manual; schema completă

Transformatorul sau generatorul ce alimentează acest circuit este omis pentru simplitate. În realitate, circuitul este cel alăturat.

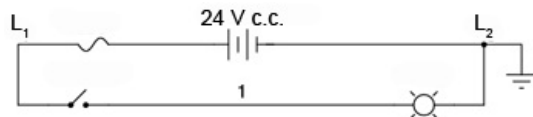


Figure 89: diagrama ladder de control a unei lămpi prin intermediul unui comutator manual; alimentare în c.c. la 24 V

Deși circuitele logice industriale utilizează o tensiune de 120 V în c.a., există și sisteme realizate la tensiuni mai mici în c.a. sau chiar și în c.c.

Atâta timp cât contactele comutatoarelor și bobinele releelor sunt dimensionate corespunzător, tensiunea de alimentare a sistemului este irelevantă.

Observați cifra „1” notată pe conductorul dintre comutator și lampă. În realitate acel conductor este notat cu cifra „1” folosind etichete adezive sau tuburi termocontractibile, în funcție de preferințe. Conductorii ce duc înspre comutator vor fi notați cu „ $L_1$ ”, respectiv „1”. Conductorii ce duc înspre lampă vor fi notați cu „1”, respectiv „ $L_2$ ”. Aceste numerotații sunt făcute pentru a ușura construirea și întreținerea întregului ansamblu. Fiecare conductor are propriul său număr unic. Numerotarea conductorilor nu se schimbă atunci când aceștia intră/ies dintr-un nod, indiferent dacă mărimea, culoarea sau lungimea lor se schimbă. Desigur, este de preferată utilizarea unei singure culori pentru desemnarea aceluiași conductor, dar acest lucru nu este tot timpul practic. Ceea ce contează este ca orice punct comun din punct de vedere electric dintr-un circuit de control să fie desemnat prin același număr de fir (conductor).

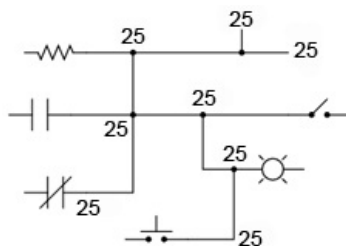


Figure 90: diagrama ladder

Să luăm de exemplu diagrama ladder alăturată. Conductorul notat cu „25” reprezintă de fapt același fir din punct de vedere electric, chiar dacă acesta este conectat la mai multe dispozitive.

În diagramele ladder, sarcina (lampă, releu, etc.) este aproape tot timpul conectată la dreapta „scării”. Deși din punct de vedere electric locația sarcinii nu are nicio importanță, contează totuși care capăt al „scării” este conectat la masă.

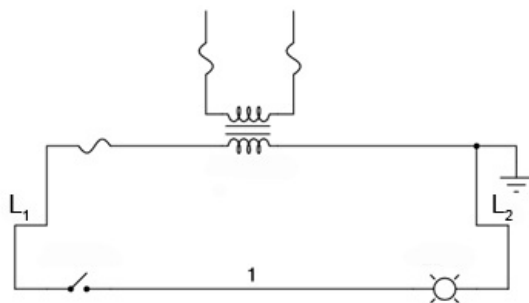


Figure 91: diagrama ladder

Să considerăm exemplul alăturat. În acest caz, lampa (sarcina) este conectată în dreapta circuitului, la fel și masa sursei de alimentare. Aceasta nu este o simplă coincidență.



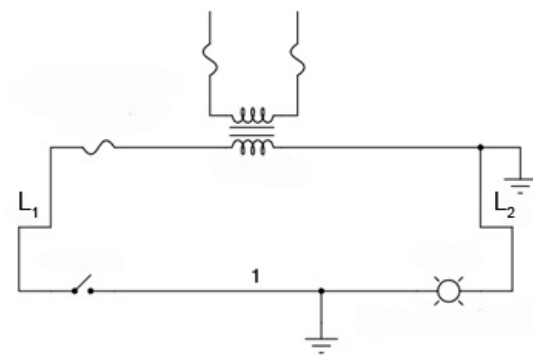


Figure 92: diagrama ladder; masă accidentală

Să presupunem că la un moment dat există o conexiune accidentală între conductorul 1 și masă. Această conexiune poate fi dată de uzura izolației și contactul conductorului cu un mediu conductor conectat la pământ. Cu ambele capete ale lămpii conectate la masă (același potențial, prin urmare, cădere de tensiune zero), lampa este practic scurt-circuitată și nu se poate aprinde. În cazul închiderii comutatorului, acest scurt-circuit va duce la arderea siguranței fuzibile.

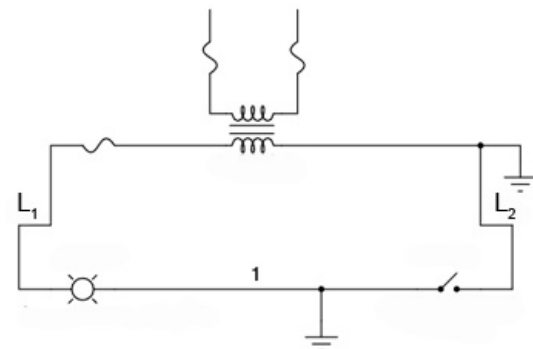


Figure 93: diagrama ladder; masă accidentală

Dar, să vedem ce s-ar întâmpla în cazul unei defecțiuni identice (contactul accidental dintre conductorul 1 și masă) în cazul în care poziția comutatorului este schimbată cu cea a lămpii. Și în acest caz,  $L_2$  este conectat la masă. Masa accidentală va forța aprinderea lămpii, iar comutatorul nu va avea niciun efect asupra funcționării acesteia.

Este mult mai bine și mai sigur din punct de vedere electric să avem un sistem a cărui siguranță fuzibilă se arde în cazul unui defect de împământare, decât un sistem a cărui componente (lămpi, relee, etc.) nu pot fi controlate în cazul aceluiași defect. Din această cauză, sarcina(ile) unei diagrame ladder trebuie tot timpul conectată lângă conductorul legat la masă (comun din punct de vedere electric cu acesta).

## 6.2 Funcții logice digitale

### 6.2.1 Funcția logică SAU

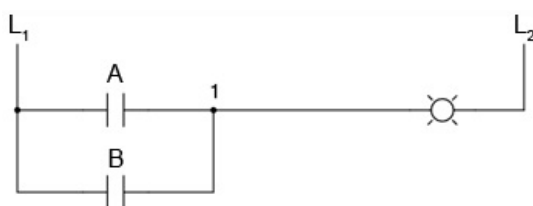


Figure 94: diagramă lader; funcția SAU

Putem construi funcții logice simple pentru circuitul cu lampă din secțiunea precedentă, folosind contacte multiple. Documentarea acestor circuite se face relativ simplu prin conectarea unor linii adiționale diagramei inițiale.

A	B	Ieșire
0	0	0
0	1	1
1	0	1
1	1	1

Dacă folosim notația binară standard pentru starea comutatoarelor și a lămpii (0 pentru ne-acționat sau de-energizat, 1 pentru acționat sau energizat), putem utiliza un tabel de adevăr pentru reprezentarea logicii circuitului.

După cum se poate observa din diagrama ladder, lampa se va aprinde (energiza) în cazul în care contactul A sau contactul B este acționat. Electronii nu au nevoie decât de o singură cale (de la  $L_1$  spre 1) pentru a ajunge spre lampă. Prin urmare, indiferent care contact se închide, A sau B, lampa se va aprinde.



Figure 95: simbolul porții logice SAU

Ceea ce am implementat de fapt în acest caz nu este altceva decât o poartă logică SAU, utilizând două contacte normal-deschise și o lampă.

### 6.2.2 Funcția logică ȘI



Figure 96: diagramă lader; funcția ȘI

Putem imita funcția unei porți logice ȘI prin conectarea celor două contacte normal-deschise în serie și nu în paralel.

A	B	Ieșire
0	0	0
0	1	0
1	0	0
1	1	1



Figure 97: simbolul porții logice ȘI

Putem verifica cu ajutorul tabelului de adevăr că acest lucru este într-adevăr corect.

În acest caz, lampa se va aprinde doar dacă ambele contacte sunt acționate simultan. Curentul va putea trece de la  $L_1$  la 2 doar dacă ambele contacte sunt închise.

### 6.2.3 Funcția logică NU



Figure 98: diagramă ladder; funcția logică NU

Funcția logică de inversare poate fi obținută prin simpla utilizare a unui contact normal-închis, față de un contact normal-deschis precum cele folosite mai sus.

A	Ieșire
0	1
1	0



Figure 99: simbolul funcției logice NU

Din nou, putem verifica prin intermediul tabelului de adevăr că acest lucru este corect.

### 6.2.4 Funcția logică ȘI-negat

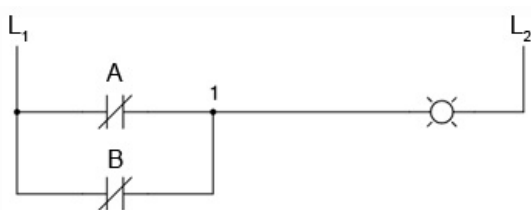


Figure 100: diagramă ladder; funcția logică ȘI-negat

Dacă luăm funcția SAU prezentată mai sus și inversăm fiecare intrare vom obține funcția ȘI-negat. Intrările se inversează prin utilizarea contactelor normal-închise în loc de contacte normal-deschise.

A	B	Ieșire
0	0	1

0	1	1
1	0	1
1	1	0

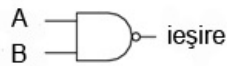


Figure 101: simbolul funcției logice ȘI-negat

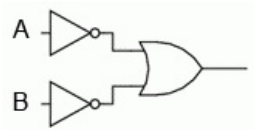


Figure 102: funcția logică ȘI-negat

Lampa va fi energizată dacă unul dintre contacte nu este acționat, și se va stinge doar dacă ambele contacte sunt acționate simultan.

### 6.2.5 Funcția logică SAU-negat



Figure 103: diagramă ladder; funcția logică SAU-negat

Asemănător, dacă luăm funcția ȘI implementată mai sus, și inversăm intrările, obținem funcția logică SAU-negat. Inversarea intrărilor se realizează și în acest caz prin utilizarea contactelor normal-închise în loc de contacte normal-deschise.

A	B	Ieșire
0	0	1
0	1	0
1	0	0
1	1	0

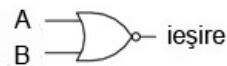


Figure 104: simbolul porții logice SAU-negat

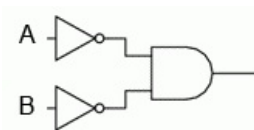


Figure 105: funcția logică SAU-negat

Din cele observate mai sus, putem trage unele concluzii, și anume: contactele paralele sunt echivalente cu o poartă logică SAU; contactele serie sunt echivalente cu o poartă ȘI; contactele normal-închise sunt echivalente cu o poartă NU (negare).

### 6.2.6 Funcția logică SAU-exclusiv

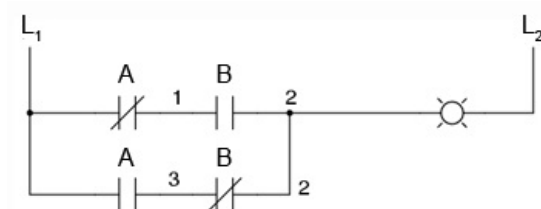


Figure 106: diagramă ladder; funcția logică SAU-exclusiv

Putem construi circuite logice combinaționale prin gruparea contactelor în aranjamente serie-paralel. În exemplul alăturat, funcția SAU-exclusiv este construită prin combinarea porților ȘI, SAU și NU.

A	B	Ieșire
0	0	0

0	1	1
1	0	1
1	1	0



Figure 107: simbolul porții logice SAU-exclusiv

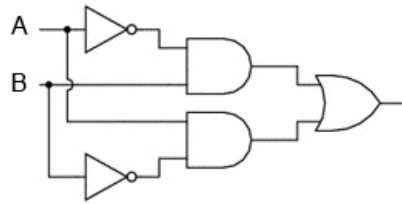


Figure 108: funcția logică SAU-exclusiv

Linia de sus (contactul normal-închis A în serie cu, contactul normal-deschis B) este echivalentă cu partea de sus a combinației de porți logice NU/ȘI. Linia de jos (contactul normal-deschis A în serie cu, contactul normal-închis B) este echivalentă cu partea de jos a combinației de porți NU/ȘI. Conexiunea în paralel a celor două linii în punctul 2, formează un circuit echivalent SAU. Acest lucru permite energizarea lămpii fie prin linia 1 fir prin linia 2.

Pentru realizarea funcției SAU-exclusiv a trebuit să folosim două contacte pe o singură intrare: un contact pentru intrarea directă, iar celălalt contact pentru intrarea inversată. Cele două contacte A din diagrama de mai sus sunt acționate fizic de același mecanism. Același lucru este valabil și pentru contactele B. Această legătură „fizică” dintre contacte este scoasă în evidență prin marcarea identică a contactelor. Nu există nicio limită a numărului de contacte ce pot fi reprezentate pe același releu. Fiecare nou contact adăugat unui releu sau unui comutator, fie că este contact normal-închis sau normal-deschis) este reprezentat prin același simbol.

### 6.2.7 Marcarea compusă

În unele situații, se folosește o marcă compusă de genul „A-1” și „A-2” în loc de „A” pentru ambele contacte ale aceluiași dispozitiv. Acest lucru este folositor mai ales în cazul în care dorim să scoatem în evidență care seturi de contacte, din fiecare dispozitiv, este utilizat pentru care parte a circuitului. Pentru simplitate însă, nu vom folosi o asemenea notație în cele ce urmează. Dacă vedeți mai multe contacte marcate identic (A, B, etc.), puteți să fiți siguri că acele contacte sunt acționate de același mecanism.

### 6.2.8 Inversarea ieșirii

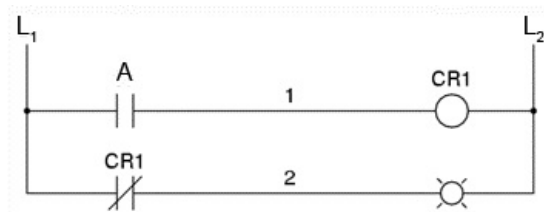


Figure 109: diagramă ladder; negarea ieșirii unei funcții logice

Dacă dorim să inversăm ieșirea unei funcții logice realizate cu ajutorul unui comutator, trebuie să folosim un releu cu un contact normal-închis. De exemplu, dacă vrem să energizăm o sarcină bazându-ne pe negarea (funcția NU) unui contact normal-deschis, putem realiza diagrama alăturată.

A	CR1	Ieșire
0	0	1
1	1	0



Figure 110: funcția logică NU

Releul este indicat pe figură prin notația CR1 (releu de control 1). Atunci când bobina releului, simbolizată printr-un cerc pe prima linie, este energizată, contactul de pe linia a doua se deschide. Deschiderea acestui contact de-energizează lampa. De la comutatorul la bobina CR1, funcția logică este ne-inversată. Contactul normal-închis este acționat de bobina releului CR1, asigurând o funcție logică de negare (NU) pe lampă, inversă față de starea de acționare a comutatorului (A).

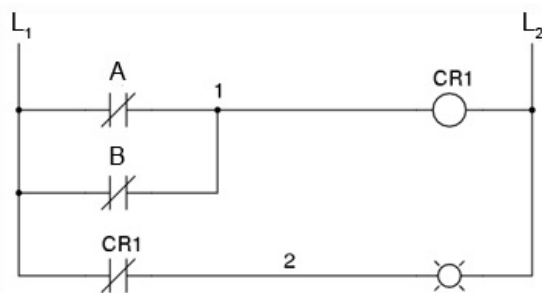


Figure 111: diagrama ladder; funcția logică ȘI-negat realizată prin inversarea ieșirii

Să aplicăm această strategie de inversare uneia dintre funcțiile cu intrare inversată realizate mai sus. Spre exemplu, funcția logică ȘI folosind diagrama funcției ȘI-negat de mai sus. Putem inversa ieșirea cu ajutorul unui releu pentru realizarea unei funcții ne-inversate.

A	B	Ieșire
0	0	0
0	1	0
1	0	0
1	1	1

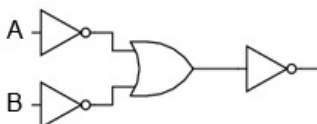


Figure 112: funcția logică ȘI



Figure 113: simbolul porții logice ȘI

De la comutator la bobina CR1, funcția logică realizată este cea a unei porți ȘI-negat. Contactele CR1 normal-închise inversează și transformă ieșirea funcției ȘI-negat într-o funcție ȘI.

## 6.3 Circuite permissive și de blocare

### 6.3.1 Controlul aprinderii furnalelor

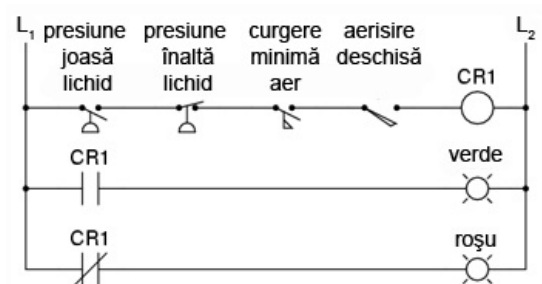


Figure 114: diagramă ladder; circuit de control al aprinderii furnalelor

O aplicație practică a logicii digitale, bazate pe relee și comutatoare, constă în controlul sistemelor în care există o serie de condiții ce trebuie îndeplinite, înainte ca un echipament să poate porni în siguranță. Un exemplu bun este cel al aprinderii furnalelor. Pentru pornirea în siguranță a acestora, sistemul de control trebuie să ceară „permisiunea” câtorva comutatoare de proces, printre care: comutatoare de presiune joasă și înaltă, comutatoare pentru verificarea funcționării ventilatorului, a poziției ușii de acces, etc. Fiecare condiție de proces se numește permisivă, iar fiecare contact permisiv este conectat în serie. Prin urmare, dacă oricare dintre contacte detectează o condiție de nesiguranță, circuitul va fi deschis.

Dacă toate condițiile sunt îndeplinite, CR1 se va energiza iar lampa verde se va aprinde. În realitate, nu doar lampa se energizează. De obicei există un releu de control, sau o valvă de fluid, ce este plasată pe acea linie a diagramei. Aceasta se va energiza cât toate contactele permissive sunt „în regulă”: adică, închise. Dacă oricare dintre condițiile permissive nu este îndeplinită, linia de sus a diagramei va rămâne întreruptă, CR1 se va de-energiza, iar lampa roșie se va aprinde.

Contactul pentru presiune înaltă a lichidului este un contact normal-închis. Acest lucru se datorează faptului că dorim deschiderea contactului doar în cazul în care presiunea lichidului devine prea mare. Din moment ce condiția „normală” a oricărui comutator de presiune este îndeplinită când presiunea aplicată asupra sa este zero, și dorim ca acest comutator să se deschidă în cazul unei presiuni excesive, trebuie să alegem un comutator ce este închis în starea sa normală.

### 6.3.2 Controlul pornirii motoarelor electrice

O altă aplicație practică a releelor constă în controlul sistemelor în care dorim ca două evenimente incompatibile să nu aibă loc în

același timp.

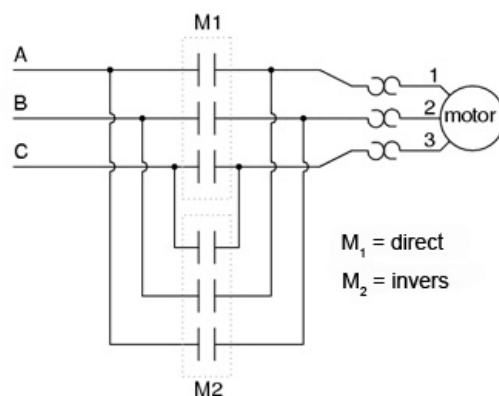


Figure 115: inversarea fazei unui motor electric

Un exemplu în acest sens constă controlul direcției de rotație a unui motor electric. Sunt utilizate contactoare pentru schimbarea polarității (sau secvenței fazelor) unui motor electric. Nu dorim însă ca atât contactorul de polarizare directă cât și cel de polarizare inversă să fie energizate în același timp.

Când contactorul  $M_1$  este energizat, sistemul trifazat de alimentare (A, B și C) este conectat direct la terminalii 1, 2 și 3 ai motorului. Totuși, când contactorul  $M_2$  este energizat, fazele A și B sunt inversate, A fiind conectată la terminalul 2 al motorului, iar B la terminalul 1. Inversarea fazei duce la inversarea direcției de rotație a motorului.

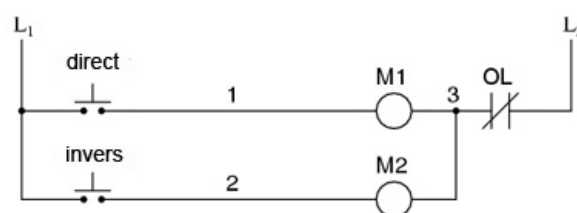


Figure 116: diagrama ladder; controlul pornirii unui motor electric

Să examinăm circuitul de control a acestor două contactoare. În dreapta avem un contact normal-închis (OL). Acesta este contactul termic de supra-încălzire ce este activat de elementele de „încălzire” conectate în serie cu fiecare fază a motorului de c.a. Dacă acestea se încălzesc prea tare, contactul va trece de la starea normală (închisă) la starea deschisă. Acest lucru nu va permite energizarea niciunui contactor.

Acest sistem de control este suficient, atâta timp cât nimeni nu apasă ambele butoane simultan. Dacă acest lucru se întâmplă însă, fazele A și B vor fi scurt-circuitate prin faptul că fazele A și B sunt conectate direct la motor prin intermediul contactorului  $M_1$ , iar contactorul  $M_2$  le inversează. Faza A se va afla în scurt-circuit cu faza B și invers. Evident, acesta nu este un sistem de control bun.

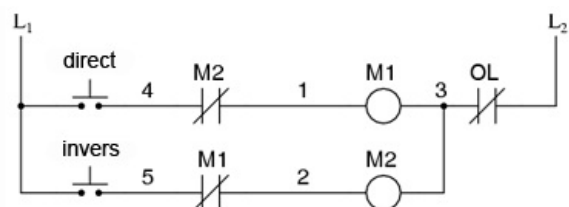


Figure 117: diagrama ladder; controlul pornirii unui motor electric

Pentru a preveni acest lucru, putem să proiectăm circuitul astfel încât energizarea unuia dintre contactoare săprevină energizarea celuilalt. Acest lucru se numește blocare, și se poate realiza prin utilizarea de contacte adiționale pe fiecare contactor.

Acum când  $M_1$  este energizat, contactul auxiliar normal-închis de pe linia a două se va deschide, prevenind astfel energizarea lui  $M_2$ , chiar dacă buton „invers” este apăsat. Asemănător, energizarea lui  $M_1$  nu este posibilă atunci când  $M_2$  este energizat. Observați că au fost adăugate numerotații suplimentare ale firelor (4 și 5) pentru a reflecta modificările.

Trebuie menționat faptul că aceasta nu este singur metodă de blocare a contactoarelor pentru prevenirea scurt-circuitului. Unele contactoare sunt echipate cu dispozitive de blocare mecanice. Pentru siguranța adițională însă, se pot folosi și metode de blocare electrice.

## 6.4 Circuite logice cu autoprotecție

Circuitele logic, fie că sunt compuse din relee electromecanice sau din porți logice semiconductoare, pot fi construite sub mai multe variante pentru realizarea aceleiași funcții. Nu există în general o metodă „corectă” de proiectare a circuitelor logice complexe, dar există unele metode ce sunt mai bune decât altele.

În sistemele de control, siguranța joacă un rol important (sau cel puțin ar trebui să o facă). Dacă exista mai multe metode de realizare a unui circuit digital care să realizeze aceeași funcție, iar una dintre metode este mai bună din punct de vedere al siguranței la funcționare, atunci acea metodă este mai bună decât celelalte.

Să luăm ca și exemplu un sistem simplu și să vedem cum îl putem implementa folosind relee logice. Să presupunem că un laborator mare sau o clădire industrială urmează să fie echipată cu un sistem de alarmă în caz de incendiu. Acest sistem urmează să fie activat de oricare dintre comutatoarele instalate în întreaga clădire. Sistemul ar trebui să funcționeze astfel încât sirena să se energizeze dacă oricare dintre comutatoare este acționat.

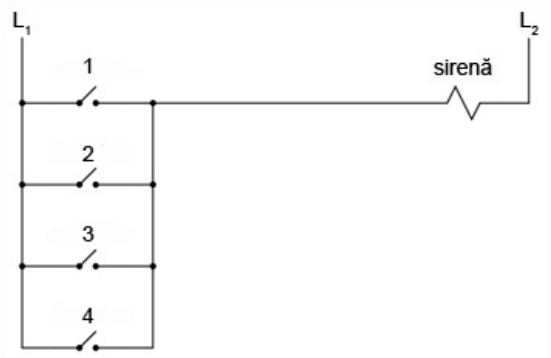


Figure 118: diagrama ladder

La o primă vedere, ar părea că circuitul logic este extrem de simplu: utilizăm contacte normal-deschise conectate în paralel, precum în figura alăturată. Practic, aceasta este o funcție logică SAU cu patru intrări. Putem extinde acest circuit pentru a include un număr oricât de mare de comutatoare, fiecare fiind adăugat în serie. Ne vom limita însă la patru în acest exemplu. În orice caz, acesta pare un sistem elementar și totul pare a fi în regulă.

Dar ce se întâmplă în cazul unui defect de circuit? Natura circuitelor electrice este astfel încât defectele de funcționare ce constau în deschiderea circuitului sunt mult mai frecvente decât oricare alt tip de defecte. Aceste deschideri ale circuitului se pot datora deschiderii contactelor releelor, întreruperea conductorilor, arderea siguranțelor fuzibile, etc. Luând acest lucru în considerare, pare normal să realizăm un circuit care să fie cât mai tolerant posibil la o astfel de defecțiune.

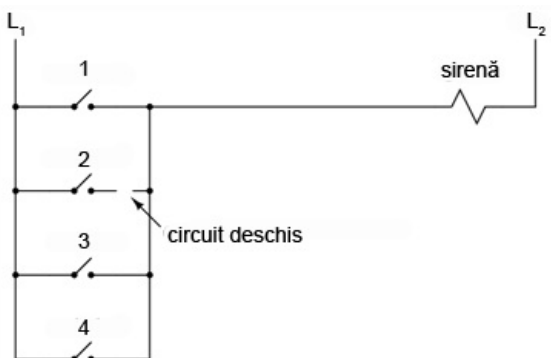


Figure 119: diagrama ladder

Să presupunem, de exemplu, ca firul conductor al comutatorului doi se distruge, ducând la deschiderea circuitului în acest punct. În cazul în care această defecțiune ar avea loc, comutatorul 2 nu ar mai putea energia sirena în cazul în care ar fi acționat (închis). Acest lucru, evident, nu este de dorit în cazul unui incendiu. Dacă sistemul nu este verificat periodic (o idee bună oricum), nimeni nu ar putea ști că există o problemă până când cineva nu ar încerca să utilizeze acel comutator în caz de urgență.

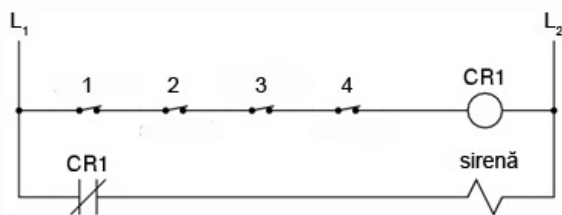


Figure 120: diagrama ladder

Dar dacă am re-proiecta sistemul astfel încât alarma să se declanșeze (și) în cazul unui astfel de defect? Astfel, defectul conductorului ar duce la o alarmă de incendiu falsă. Totuși, acest scenariu este de preferat celui în care comutatorul nu funcționează deloc în cazuri de urgență. Pentru atingerea acestui scop, va trebui să refacem circuit, astfel încât alarma să fie declanșată de un comutator deschis și nu de unul închis. Comutatoarele vor trebui să fie normal-închise și în serie, alimentând bobina unui releu care la rândul ei activează un contact normal-închis ce controlează sirena.

Atunci când toate comutatoarele sunt de-energizate (starea normală a sistemului), bobina releului CR1 va fi energizată (prima linie). Contactul CR1 (linia a doua) prin urmare, va fi deschis, iar sirena nu este alimentată. Totuși, în cazul în care oricare dintre comutatoare este acționat, bobina CR1 se de-energizează, iar contactul CR1 revine la starea sa normală, și anume, închis. Alarma este în acest caz alimentată și se va declanșa. Adicional, dacă există un defect al conductorilor din prima linie a circuitului, alarma se va declanșa. Dacă se descoperă că alarma este falsă, personalul clădirii va ști că sistemul de alarmă este defect și necesită reparații.

Cu siguranța, circuitul este mult mai complex decât era înainte introducerii releului de control CR1, iar sistemul poate și în acest caz să nu funcționeze corespunzător. Acest lucru se poate întâmpla dacă apare un defect în linia a doua a circuitului. Dar totuși, acest circuit este mai sigur și preferabil din acest punct de vedere.

## 6.5 Automate programabile (PLC)

### 6.5.1 Scurtă istorie

Înainte de apariția circuitelor logice cu semiconductori, sistemele logice de control erau proiectate și realizate exclusiv cu relee electromecanice. Sistemele și procesele ce necesită un control de tip „pornire/oprire” abundă în industria modernă, dar aceste sisteme sunt foarte rar realizate cu ajutorul releelor electromecanice sau a porților logice discrete. În schimb, sunt folosite

calculatoare digitale ce pot fi programate și pot realiza o varietate de funcții logice.

La sfârșitul anilor 1960, o companie americană pe nume Bedford Associates, a lansat un dispozitiv de calcul denumit MODICON. Ca și acronim, acesta s-ar traduce prin „controler digital modular”. Acesta mai târziu a deveni și numele diviziei care se ocupa cu proiectarea, realizarea și vânzarea acestor calculatoare de control speciale. Desigur, au existat mai apoi și alte companii care au dezvoltat propriile lor variante ale acestui dispozitiv. Până la urmă, acest dispozitiv a primit denumirea de PLC (Programmable Logic Controller), sau, în traducere, automat programabil. Scopul unui PLC a fost de a înlocui releele electromecanice ca și elemente de logică, locul lor urmând a fi luat de calculatoare digitale semiconductoare. Un program stocat în memoria calculatorului este capabil să simuleze funcții logice realizate înainte prin interconectarea unui număr mare de releele electromecanice.

Un automat programabil (PLC) are mai multe intrări, prin intermediul cărora interpretează stări logice „înalte”, respectiv „joase”, stări transmise de senzori și comutatoare. De asemenea, există mai mulți terminali de ieșire, prin intermediul cărora dispozitivul transmite semnale „înalte” sau „joase” către contactoare, motoare, lămpi, sau orice alte dispozitive ce pot fi controlate prin intermediul semnalelor de tip „închis/deschis”. În încercarea de simplificare a modului de programare a PLC-urilor, limbajul de programare a fost proiectat astfel încât să semene cu diagramele ladder. Astfel, un inginer sau electrician obișnuit cu citirea diagramelor ladder, se poate adapta relativ ușor mediului de programare a PLC-urilor pentru realizarea acelorași funcții de control. PLC-urile sunt „calculatoare industriale”, prin urmare, semnalele de intrare și de ieșire sunt de 120 V c.a, asemenea releelor electromecanice de control. Deși unele PLC-uri au intrări și ieșiri de c.c de amplitudini mai mici, aceasta este excepția și nu regula.

### 6.5.2 Programarea PLC-urilor

Modul de conectare și de programare diferă puțin în funcție de modelul de PLC ales, dar aceste caracteristici sunt destul de similar pentru a permite o introducere „generală” a programării PLC-urilor în acest capitol.

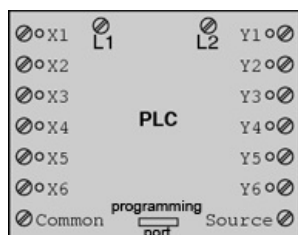


Figure 121: automat programabil

Figura alăturată este cea a unui PLC, văzut din față. Cei doi terminali  $L_1$  și  $L_2$  din partea superioară sunt pentru alimentarea circuitului intern al dispozitivului cu 120 V c.a. Cei șase terminali din partea stângă se folosesc pentru conectarea dispozitivelor de intrare, fiecare terminal reprezentând un „canal” diferit cu propria sa notație (X). Terminalul din stânga jos (common), reprezintă masa, ce se conectează la  $L_2$ .

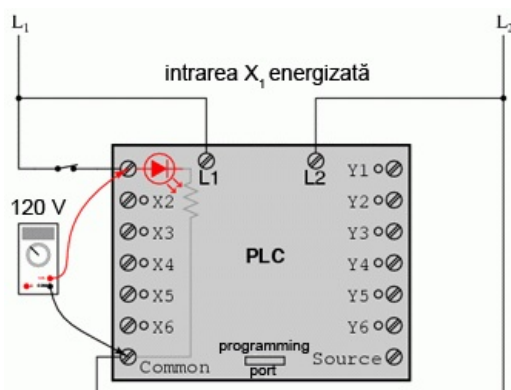


Figure 122: automat programabil; intrarea X1 energizată

În interiorul carcasei PLC-ului, între fiecare terminal de intrare și terminalul de masă, există conectat câte un dispozitiv optoizolator (LED). Acesta asigură o izolare electrică între semnalul logic „înalt” de la intrare și circuitul calculatorului, atunci când există o tensiune de 120 V c.a. aplicată între terminalul respectiv și masă. O intrare energizată poate fi „citită” prin intermediul unui LED aprins pe carcasa dispozitivului.

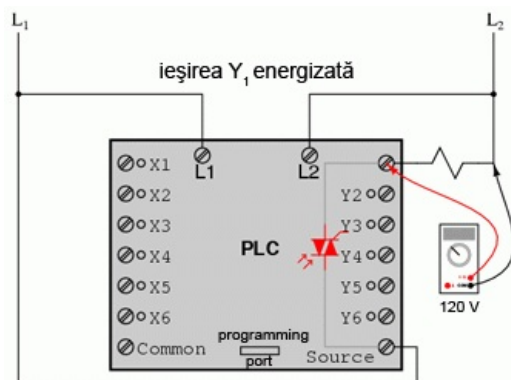


Figure 123: automat programabil; ieșirea Y1 energizată

Semnalele de ieșire sunt generate de circuitul intern al PLC-ului prin activarea unui dispozitiv de comutare (tranzistor, triac, sau chiar releu electromecanic), conectând terminalul „Source” cu oricare dintre terminalii de ieșire „Y”. Terminalul „Source” este la rândul său conectat de obicei la  $L_1$ . Din nou, o ieșire energizată poate fi citită de pe PLC prin intermediul unui LED.



În acest fel, PLC-urile sunt o interfață între dispozitivele reale precum comutatoare, lămpi, motoare, etc. Logica circuitului este stabilită în interiorul PLC-ului prin intermediul unui program software. Acest program decide care ieșiri sunt energizate și sub ce condiții de intrare. Chiar dacă programul însuși pare a fi o diagramă logică, cu simboluri pentru releee și comutatoare, în realitate nu există astfel de dispozitive în interiorul PLC-ului. Acestea sunt doar contacte și bobine imaginare. Programul este introdus și vizualizat prin intermediul unui PC conectat la portul PLC-ului (programming port).

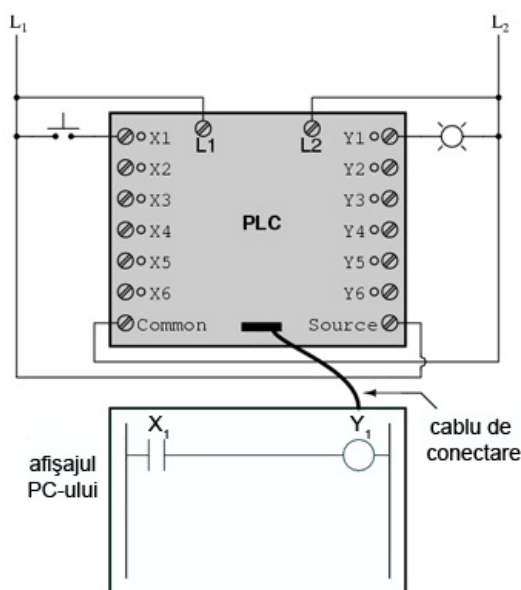


Figure 124: automat programabil și circuit logic

Să considerăm circuitul și programul alăturat. Atunci când comutatorul buton nu este apăsat, intrarea X<sub>1</sub> a PLC-ului nu este alimentată. Urmărind programul, putem vedea un contact X<sub>1</sub> normal-deschis în serie cu o bobină Y<sub>1</sub>. Puterea de pe bobina Y<sub>1</sub> este și în acest caz zero. Prin urmare, ieșirea Y<sub>1</sub> a PLC-ului rămâne de-energizată, iar lampa indicatoare conectată pe această ieșire nu se aprinde.

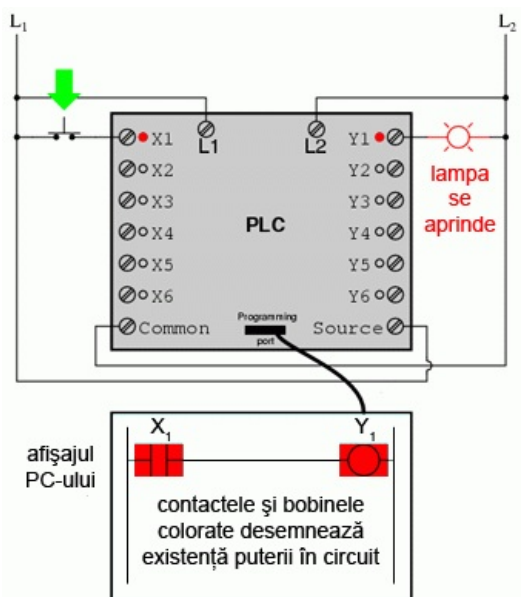


Figure 125: automat programabil și circuit logic

Apăsarea butonului însă face ca intrarea X<sub>1</sub> să fie alimentată. În acest caz, toate contactele X<sub>1</sub> ce apar în program vor fi acționate. Energizarea intrării X<sub>1</sub> va duce la închiderea contactului normal-deschis X<sub>1</sub> alimentând bobina Y<sub>1</sub>. Când bobina Y<sub>1</sub> a programului este energizată, ieșirea reală Y<sub>1</sub> va deveni energizată, iar lampa conectată pe ieșire se va aprinde. Trebuie înțeles faptul că atât contactul X<sub>1</sub> cât și bobina Y<sub>1</sub>, conductorii de legătură și „puterea” ce apar pe afișajul PC-ului, toate sunt elemente pur virtuale. Acestea nu există ca și componente reale. Ele există doar ca și comenzi în interiorul programului unui calculator.

PC-ul este utilizat doar pentru vizualizarea și editarea softului PLC-ului, și nu este necesară prezența acestuia pentru funcționarea dispozitivului. Odată ce programul a fost încărcat în PLC de pe PC, calculatorul poate fi deconectat de la acest, iar PLC-ul va continua să funcționeze conform instrucțiunilor programului. Afișajul (monitorul) calculatorului este redat în aceste figuri doar pentru a ajuta la înțelegerea principiilor de bază a funcționării PLC-urilor.

Adevărata utilitate a PLC-ului o putem vedea atunci când dorim modificarea comportamentului unui sistem de control. Din moment ce PLC-ul este un dispozitiv programabil, comportamentul acestuia poate fi modificat prin schimbarea comenzilor. Nu este nevoie de o reconfigurare a componentelor electrice conectate la intrarea și ieșirea acestuia.

De exemplu, să presupunem că dorim ca circuitul de mai sus să funcționeze exact invers: apăsarea butonului duce la închiderea lămpii, iar eliberarea acestuia la aprinderea ei. Soluția „hardware” ar consta în înlocuirea comutatorului buton normal-deschis cu un comutator buton normal-închis. Soluția software, aplicabilă cu ajutorul PLC-ului, constă în modificarea programului, astfel încât contactul X<sub>1</sub> să fie normal-închis în loc de normal-deschis.

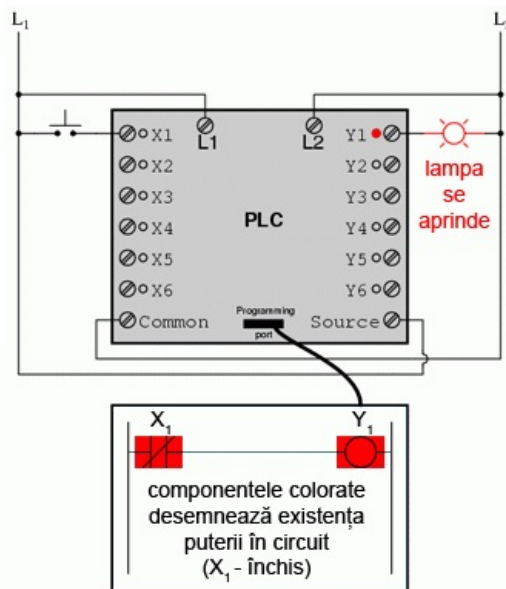


Figure 126: automat programabil și circuit logic

Sistemul modificat, în cazul în care comutatorul nu este acționat (nu este apăsat), este prezentat în figura alăturată.

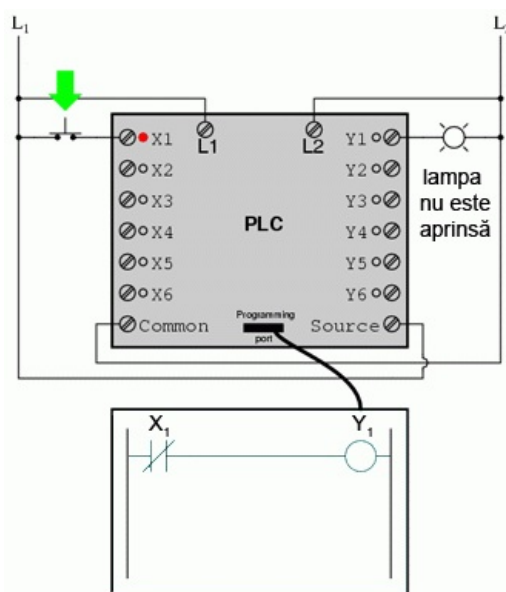


Figure 127: automat programabil și circuit logic

Când butonul este acționat, sistemul arată conform figurii alăturate.

### 6.5.3 Reutilizarea intrărilor

Un alt avantaj al implementării logicii de control în varianta software față de hardware, este că semnalele de intrare pot fi refolosite în interiorul programului ori de câte ori este necesar.

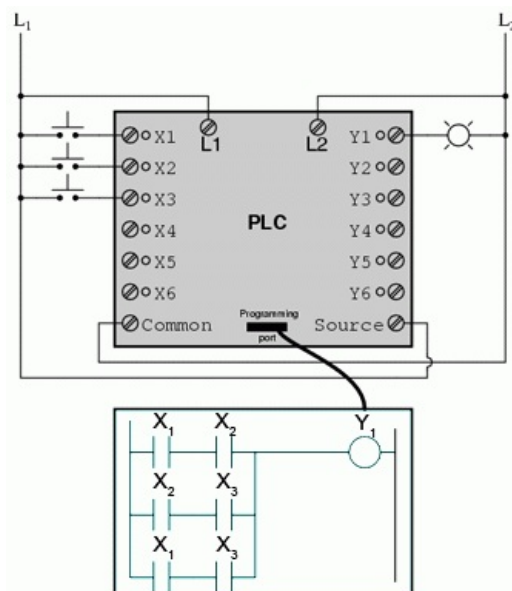


Figure 128: automat programabil și circuit logic

De exemplu, circuitul și programul alăturat sunt proiectate pentru a energiza lampa în cazul în care cel puțin două din cele trei comutatoare sunt acționate (închise) simultan.

Pentru realizarea unui circuit echivalent folosind relee electromecanice, ar fi trebui să folosim trei relee cu câte două contacte normal-deschise fiecare. În total, am fi avut nevoie de șase contacte, câte două pe fiecare intrare. Folosind un automat programabil în schimb, putem refolosi intrările X ori de câte ori dorim prin intermediul soft-ului. Nu este necesară adăugarea unor noi componente, deoarece fiecare intrare cât și ieșire a unui PLC nu este nimic mai mult decât un simplu bit (0 sau 1) stocat în memoria digitală a dispozitivului. Nu există o limită teoretică a numărului de reutilizări acestor biți.

Mai mult, din moment ce fiecare ieșire este, la fel, doar un bit stocat în memoria PLC-ului, putem adăuga contacte (virtuale) în interiorul programului. De exemplu, putem adăuga un contact acționat de ieșirea Y a PLC-ului.

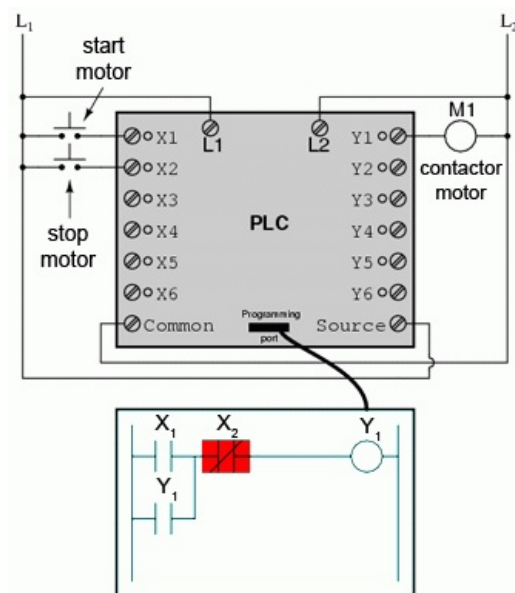


Figure 129: automat programabil și circuit logic; sistem de control al pornirii și opririi unui motor

Să considerăm exemplul alăturat. Acesta este un sistem de control al pornirii și opririi unui motor.

Comutatorul buton conectat la intrarea X<sub>1</sub> este utilizat pentru pornirea motorului, iar comutatorul conectat la intrarea X<sub>2</sub> pentru oprirea acestuia. Un contact adițional (virtual), adăugat în interiorul programului și denumit Y<sub>1</sub>, utilizează bobina de ieșire ca și contact de reținere. Contactorul motorului continuă să fie energizat chiar și după ce butonul „start” este eliberat. Contactul X<sub>2</sub> normal-închis este colorat, ceea ce înseamnă că este închis și conduce energie electrică.

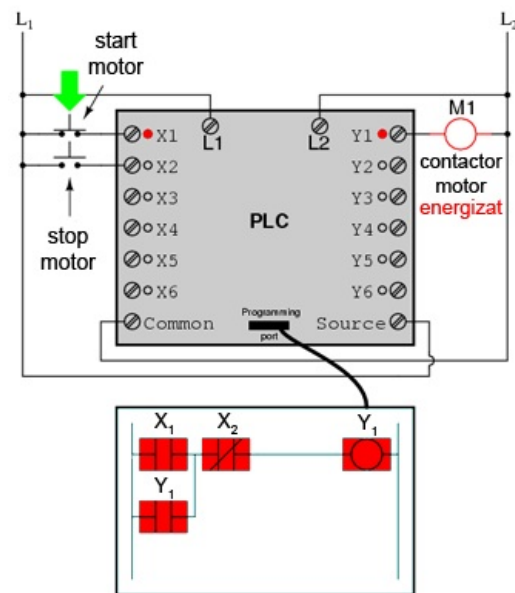


Figure 130: automat programabil și circuit logic; sistem de control al pornirii și opririi unui motor

Dacă apăsăm butonul de „start”, intrarea X<sub>1</sub> se va energiza, închizând contactul X<sub>1</sub> din program. Bobina Y<sub>1</sub> va fi energizată și se va aplica o tensiune de 120 V c.a. pe bobina contactorului motorului. Contactul paralel Y<sub>1</sub> se va închide și el, iar circuitul va rămâne într-o stare energizată.

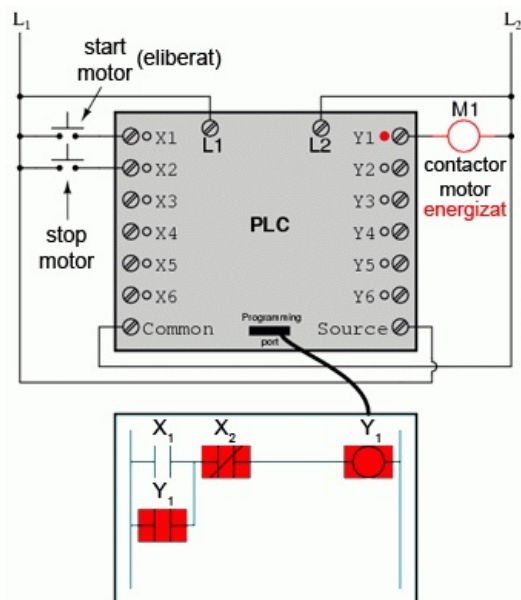


Figure 131: automat programabil și circuit logic; sistem de control al pornirii și opririi unui motor

Acum, dacă eliberăm contactorul „start”, contactul  $X_1$  normal-deschis se va reîntoarce la poziția sa normală (deschis). Motorul va continua însă să funcționeze, deoarece contactul de reținere intern  $Y_1$ , continuă să alimenteze bobina  $Y_1$ , care menține la rândul ei energizată ieșirea  $Y_1$ .

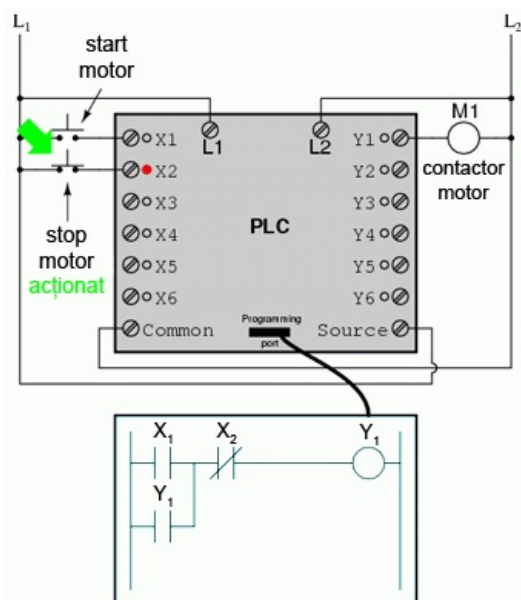


Figure 132: automat programabil și circuit logic; sistem de control al pornirii și opririi unui motor

Pentru a opri motorul, trebuie să apăsăm pentru o durată scurtă comutatorul „stop”. Acesta va energiza intrarea  $X_2$  și va deschide contactul (virtual) normal-închis. Continuitatea circuitului înspre bobina  $Y_1$  va fi întreruptă.

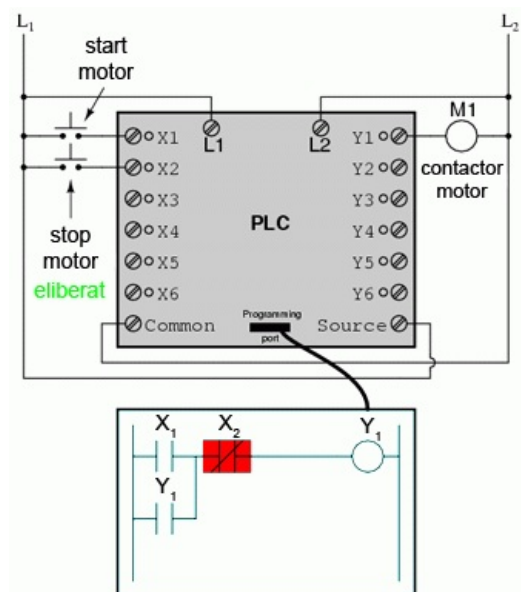


Figure 133: automat programabil și circuit logic; sistem de control al pornirii și opririi unui motor

Când butonul de „stop” este eliberat, intrarea  $X_2$  se de-energizează. Contactul  $X_2$  se reîntoarce în poziția sa normală (închis). Motorul nu va reporni însă până când comutatorul de „start” nu este acționat, datorită „pierderii” contactului de reținere  $Y_1$ .

#### 6.5.4 Autoprotecția

Desigur, proiectarea PLC-urilor astfel încât să conțină elemente de autoprotecție este la fel de importantă precum în cazul sistemelor cu relee electromecanice. Va trebui tot timpul să luăm în considerare efectele unui circuit deschis (distrugerea firelor conductoare, de exemplu) asupra dispozitivelor controlate. În exemplul de mai sus, avem o problemă: în cazul în care conductorul comutatorului de intrare  $X_2$  (butonul de stop) prezintă un defect (circuit deschis), nu vom putea opri motorul!

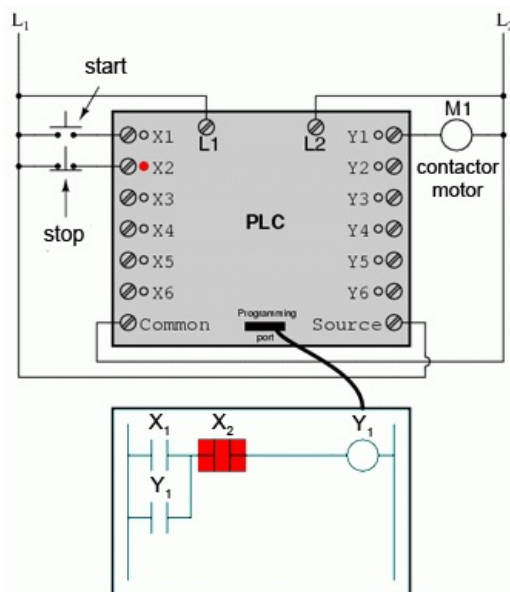


Figure 134: automat programabil și circuit logic; sistem de control al pornirii și opririi unui motor

Soluția acestei probleme constă în inversarea logicii între contactul  $X_2$  din interiorul programului PLC-ului și comutatorul „stop”. Când butonul „stop” nu este acționat, intrarea  $X_2$  este energizată, închizând contactul  $X_2$  din interiorul programului. Acest lucru permite pornirea motorului atunci când intrarea  $X_1$  este energizată, și permite funcționarea acestuia chiar și atunci când butonul „start” este eliberat. Când butonul „stop” este acționat, intrarea  $X_2$  se va de-energiza, deschizând contactul  $X_2$  din soft-ul PLC-ului și oprind motorul. Prin urmare, nu există nicio diferență din punct de vedere funcțional între această variantă și cea precedentă. Totuși, în caz de defect al conductorului pe intrarea  $X_2$  (circuit deschis), intrarea  $X_2$  va fi de-energizată. Efectul este similar acționării butonului de „stop”, rezultatul fiind oprirea imediată a motorului în caz de defect. Această variantă este mult mai sigură decât cea precedentă, în care, același tip de defect ar conduce la imposibilitatea opririi motorului.

#### 6.5.5 Relee de control

Pe lângă elementele de intrare (X) și de ieșire (Y), PLC-urile conțin bobine și contacte ce nu a legătură propriu-zisă cu exteriorul. Acestea sunt folosite asemenea releelor de control (CR1, CR2, etc.) pentru asigurarea unui semnal logic inversor în caz de nevoie.

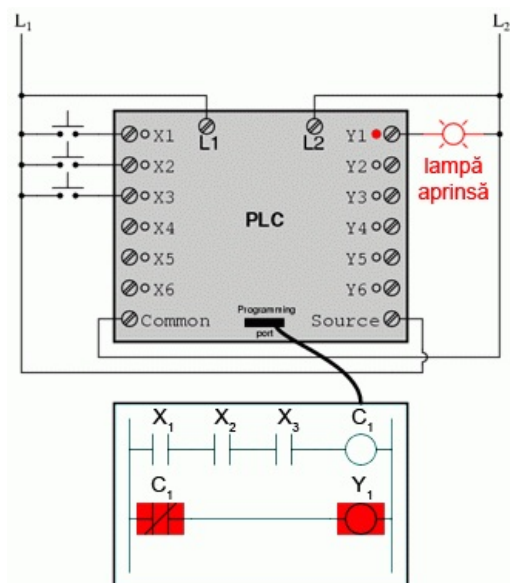


Figure 135: automat programabil și circuit logic; utilizarea releelor interne de control

Pentru demonstrarea funcționării unui asemenea tip de relee „intern”, să considerăm circuitul și programul alăturat. Acesta este proiectat pentru a simula o poartă ȘI-negat cu trei intrări. Din moment ce elementele unui program PLC sunt desemnate printr-o singură literă, vom nota releele de control intern cu C1 și nu cu CR1.

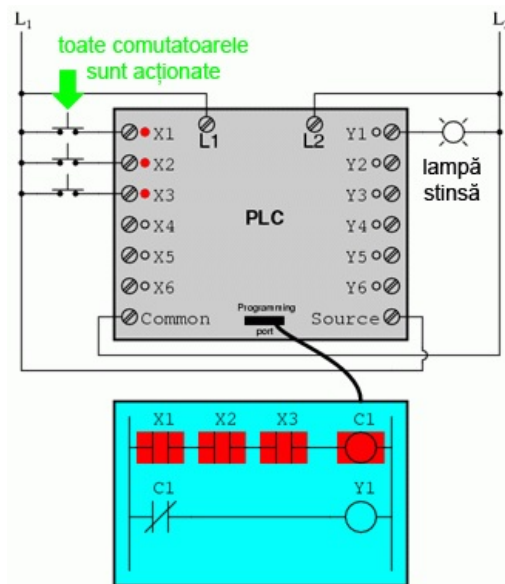


Figure 136: automat programabil și circuit logic; utilizarea releelor interne de control

În acest circuit, lampa va rămâne aprinsă atâta timp cât oricare dintre butoane rămâne ne-acționat (eliberat). Pentru a opri lampa, va trebui să acționăm (apasăm) toate cele trei butoane.

Un mare avantaj al automatelor programabile, avantaj ce nu poate fi duplicat cu ajutorul releelor electromecanice, constă în posibilitatea supravegherii și controlului la distanță a dispozitivelor prin intermediul rețelelor de calculatoare.

## 7 Algebra booleană

### 7.1 Introducere și scurt istoric

Regulile matematice sunt bazate pe limitele impuse asupra cantităților numerice particulare cu care avem de a face. Când spunem că  $1 + 1 = 2$ , sau  $3 + 4 = 7$ , presupunem din start că utilizăm un sistem de numerație zecimal. Regulile aritmetice pe care le considerăm de la sine înțelese - adevărate tot timpul și în orice situație - depind de fapt de ceea ce înțelegem printr-un număr.

### 7.2 Aritmetica booleană

#### 7.2.1 Numere binare și numere booleene

Trebuie înțeles încă de la început faptul că numerele booleene nu sunt tot una cu numerele binare. Numerele booleene reprezintă un sistem matematic total diferit de cel al numerelor reale, pe când notația binară este doar atât: o notație alternativă a numerelor reale. Cele două sunt adesea confundate datorită faptului că utilizează aceleași cifre: 0 și 1. Diferența constă în faptul că valorile booleene sunt limitate la un singur bit (fie 0, fie 1), pe când numerele binare pot fi compuse din mai mulți biți.

#### 7.2.2 Adunarea booleană

Să începem așadar capitolul de algebră booleană prin adunarea numerelor:

$$0 + 0 = 0 \quad 0 + 1 = 1 \quad 1 + 0 = 1 \quad 1 + 1 = 1$$

Primele trei sume nu sunt deloc ieșite din comun din punct de vedere al operației de adunare elementară. Ultima sumă în schimb, s-a dovedit a fi responsabilă de mai multă confuzie decât oricare alt element al electronicii digitale. Forma sa nu se supune principiilor de bază ale matematicii. Într-adevăr, aceasta contrazice principiile adunării numerelor reale, dar nu și a numerelor booleene. În cadrul matematicii booleene există doar două valori posibile pentru oricare valoare și pentru orice operație matematică: 0 sau 1. Nu există o valoare „2”. Din moment ce suma „1 + 1” nu poate fi 0, prin eliminare, această sumă trebuie să fie 1.

De asemenea, nu contează nici câți termeni conține suma. Să considerăm următoarele sume, de exemplu:

$$0 + 1 + 1 = 1 \quad 1 + 1 + 1 = 1 \quad 0 + 1 + 1 + 1 = 1 \quad 1 + 0 + 1 + 1 + 1 = 1$$

Revenind la primul set de ecuații, putem observa că aceste sume nu sunt altceva decât tabelul de adevăr al unei porți logice SAU. Cu alte cuvinte, adunarea booleană corespunde funcției logice a porții SAU, precum și comutatoarelor conectate în paralel:

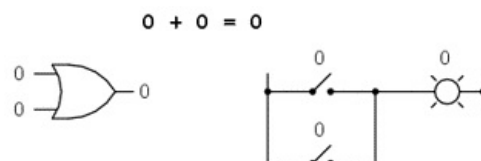


Figure 137: poartă logică SAU și comutatoare paralel (0 + 0)

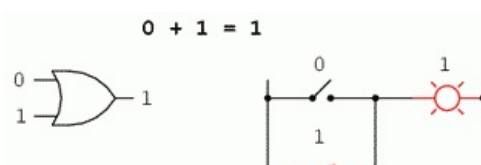


Figure 138: poartă logică SAU și comutatoare paralel (0 + 1)

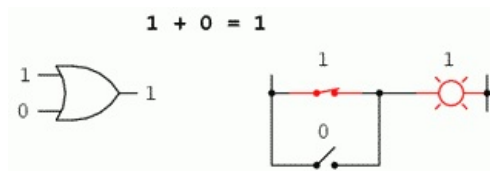


Figure 139: poartă logică SAU și comutatoare paralel ( $1 + 0$ )

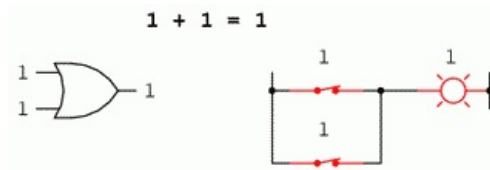


Figure 140: poartă logică SAU și comutatoare paralel ( $1 + 1$ )

### 7.2.3 Scăderea și împărțirea booleană

În cadrul matematicii booleene nu există noțiunea de scădere. Scăderea implică existența numerelor negative:  $5 - 3$  este identic cu  $5 + (-3)$ , de exemplu. Dar în algebra booleană, valorile negative nu există (doar 0 și 1).

De asemenea, nu există nici operație de împărțirea booleană. Împărțirea nu este altceva decât o scădere compusă, la fel cum înmulțirea nu este altceva decât adunare compusă.

### 7.2.4 Înmulțirea booleană

Înmulțirea booleană este permisă, iar regulile sunt aceleași cu înmulțirea numerelor reale: orice număr înmulțit cu 0 este 0, și orice număr înmulțit cu 1 rămâne neschimbat:

$$0 \times 0 = 0 \quad 0 \times 1 = 0 \quad 1 \times 0 = 0 \quad 1 \times 1 = 1$$

Setul de ecuații ar trebui să vă fie cunoscute: sunt aceleași reguli ce se regăsesc în tabelul de adevăr al porții ȘI. Cu alte cuvinte, înmulțirea booleană corespunde funcției logice a porții ȘI, precum și comutatoarelor conectate în serie:

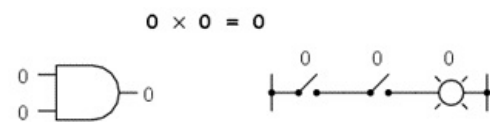


Figure 141: poartă logică ȘI și comutatoare serie ( $0 \times 0$ )

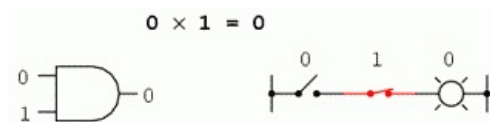


Figure 142: poartă logică ȘI și comutatoare serie ( $0 \times 1$ )

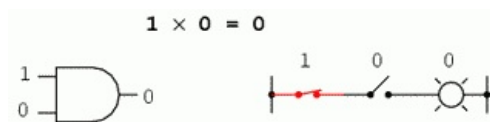


Figure 143: poartă logică ȘI și comutatoare serie ( $1 \times 0$ )

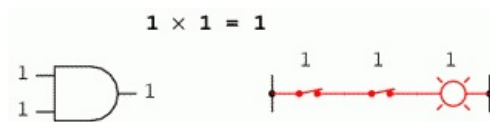


Figure 144: poartă logică ȘI și comutatoare serie ( $1 \times 1$ )

### 7.2.5 Variabile booleene și complementul lor

La fel ca algebra „normală”, algebra booleană utilizează litere pentru desemnarea variabilelor. Dar, fața de algebra „normală”, aceste variabile se trec tot timpul cu majuscule. Datorită faptului că există doar două stări posibile, fie 1, fie 0, fiecare variabilă posedă și un complement: valoarea opusă a acesteia. De exemplu, dacă variabila „A” este 0, atunci complementul ei este 1.

Complementul se specifică prin intermediul unei linii deasupra variabilei, astfel:

$$\begin{array}{ll} \text{Dacă: } A=0 & \text{Dacă: } A=1 \\ \text{Atunci: } \overline{A}=1 & \text{Atunci: } \overline{A}=0 \end{array}$$

Figure 145: complementul unei variabile booleene

Sub formă scrisă, complementul lui „A” este desemnat prin „A-negat”. Câteodată se utilizează simbolul „'” pentru reprezentarea complementului (A'). De obicei însă, simbolul cu linie este mai folosit decât simbolul „'”. Motivele le vom afla puțin mai încolo. Complementarea booleană este echivalentă cu o poartă logică NU, sau cu un contact normal-închis:



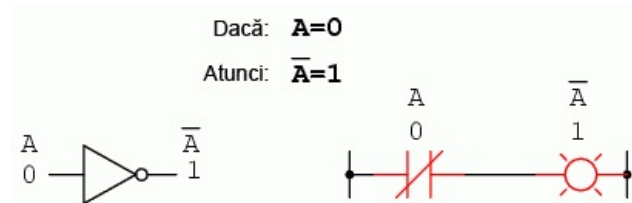


Figure 146: complementul unei variabile booleene; poartă logică SAU și contact normal-închis

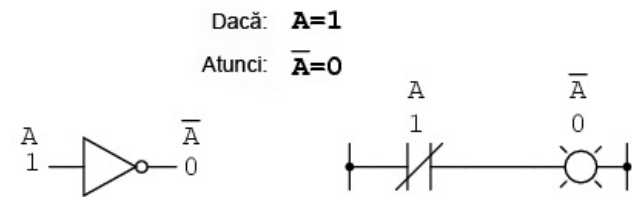


Figure 147: complementul unei variabile booleene; poartă logică SAU și contact normal-închis

## 7.3 Identități algebrice booleene

### 7.3.1 Ce este o identitate

În matematică, o identitate este o afirmație valabilă pentru toate valorile posibile ale variabilei sau variabilelor implicate. Identitatea algebrică  $x + 0 = x$ , ne spune că suma dintre oricare variabilă ( $x$ ) și zero este egală cu variabila inițială ( $x$ ), indiferent de valoarea acesteia. Asemenea algebrei obișnuite, există identități specifice algebrei booleene. Aceste identități sunt bazate pe cele două stări posibile ale variabilelor booleene (0 sau 1).

### 7.3.2 Identități aditive

Prima identitate booleană este suma unei variabile cu zero. Rezultatul este valoarea variabilei inițiale. Această identitate nu este cu nimic diferită față de echivalentul algebric al numerelor reale:

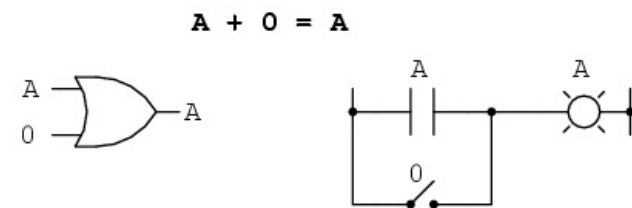


Figure 148: identitate booleană; suma unei variabile cu zero

Indiferent de valoare lui A, ieșirea va fi tot timpul aceeași. Când  $A = 1$ , ieșirea va fi 1; când  $A = 0$ , ieșirea va fi 0.

Următoarea identitate este cu siguranță diferită față de cele văzute în algebra obișnuită. Aici putem vedea că suma unei variabile cu 1 este 1:

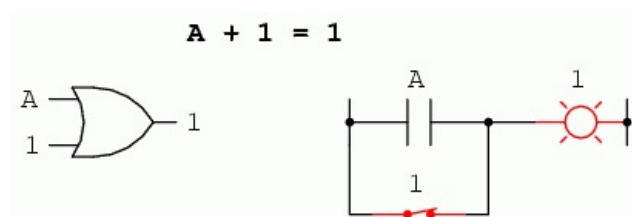


Figure 149: identitate booleană; suma unei variabile cu unu

Indiferent de valoarea lui A, suma lui A cu 1 va fi tot timpul 1. Practic, ieșirea circuitului nu ține cont de valoarea lui A, ci este fixată pe 1.

Următoare identitate este suma unei variabile cu ea însăși. Practic, acest lucru înseamnă conectarea intrărilor unei porți logice SAU și activarea lor cu același semnal:

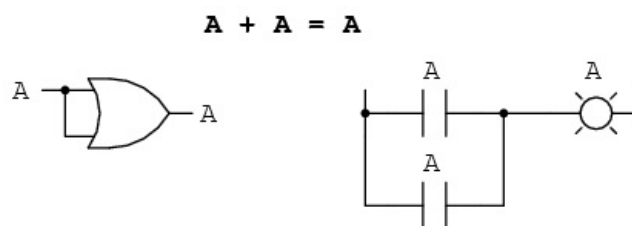


Figure 150: identitate booleană; suma unei variabile cu ea însăși

În algebra numerelor reale, suma a două variabile identice este dublul variabilei inițiale ( $x + x = 2x$ ). Dar în cadrul algebrei booleene nu există „2”, ci numai 0 și 1. Prin urmare, nu putem spune că  $A + A = 2A$ . Adunarea unei variabile cu ea însăși este egală cu suma originală:  $0 + 0 = 0$  și  $1 + 1 = 1$ .

Dacă introducem conceptul de complement într-o identitate aditivă, putem vedea un efect interesant. Din moment ce între orice variabilă și complementul acesteia trebuie să avem un 1, și din moment ce suma oricărei variabile booleene cu 1 este 1, suma dintre



o variabilă și complementul ei trebuie să fie 1:

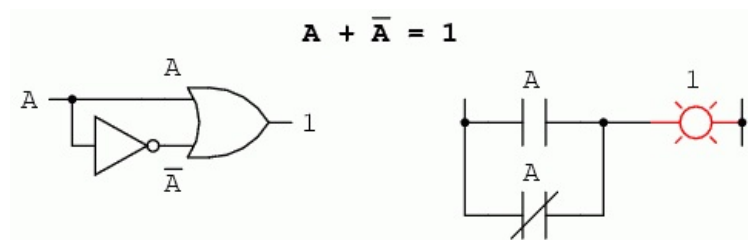


Figure 151: suma booleană dintre o variabilă și complementul acesteia

### 7.3.3 Identități multiplicative

La fel cum există patru identități booleene aditive ( $A + 0$ ,  $A + 1$ ,  $A + A$  și  $A + A'$ ), există și patru identități multiplicative:  $A \times 0$ ,  $A \times 1$ ,  $A \times A$  și  $A \times A'$ . Dintre acestea, primele două nu sunt deloc diferite de identitățile echivalente ale algebrei numerelor reale:

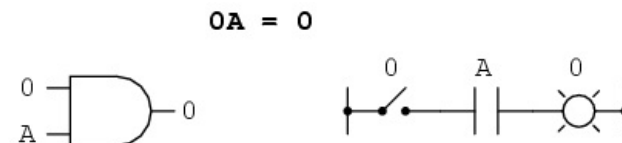


Figure 152: identități algebrice multiplicative: produsul dintre o variabilă și zero

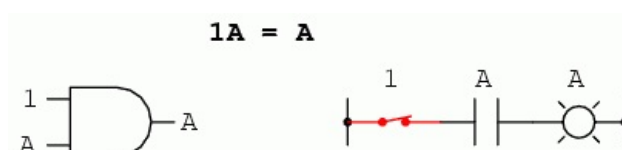


Figure 153: identități algebrice multiplicative: produsul dintre o variabilă și unu

Cea de a treia identitate multiplicativă exprimă rezultatul unei variabile booleene înmulțită cu ea însăși. În algebra numerelor reale, acest tip de produs reprezintă pătratul variabilei în cauză ( $3 \times 3 = 3^2 = 9$ ). Conceptul de „pătrat” implică existența valorii 2, valoare ce nu poate fi exprimată prin algebra booleană. Nu putem spune că  $A \times A = A^2$ . În schimb, produsul unei valori booleene cu ea însăși este valoarea inițială, din moment ce  $0 \times 0 = 0$  și  $1 \times 1 = 1$ :

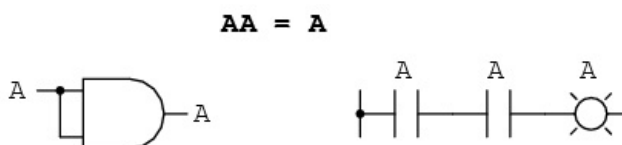


Figure 154: identități algebrice multiplicative: produsul dintre o variabilă și ea însăși

A patra identitate multiplicativă nu are echivalent în algebra numerelor reale, deoarece utilizează complementul variabilei. Acest concept este unic matematicii booleene. Din moment ce trebuie să avem o valoare de „0” între oricare variabilă și complementul acesteia, și din moment ce produsul oricărei valori booleene cu 0 este 0, produsul dintre o variabilă și complementul acesteia trebuie să fie 0:

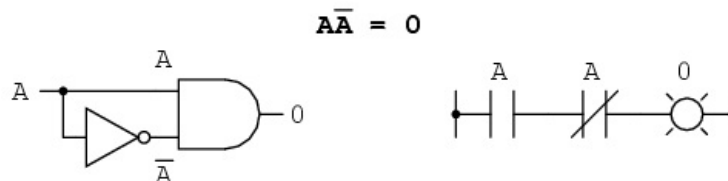


Figure 155: identități algebrice multiplicative: produsul dintre o variabilă și complementul ei

În concluzie, avem patru identități booleene de bază pentru adunare și patru pentru produs (multiplicative):

Identități aditive	Identități multiplicative
$A + 0 = A$	$0A = 0$
$A + 1 = 1$	$1A = A$
$A + A = A$	$AA = A$
$A + \bar{A} = 1$	$A\bar{A} = 0$

Figure 156: identități algebrice aditive și multiplicative

### 7.3.4 Identitatea complementului dublu

O altă identitate caracteristică complementului unei variabile este cea a complementului dublu: o variabilă inversată de două ori. Rezultatul complementării duble a unei variabile este valoarea booleană inițială a variabilei. Acest lucru este similar înmulțirii cu -1 în algebra numerelor reale: un număr par de astfel de înmulțiri se anulează, iar rezultatul final este valoarea inițială:

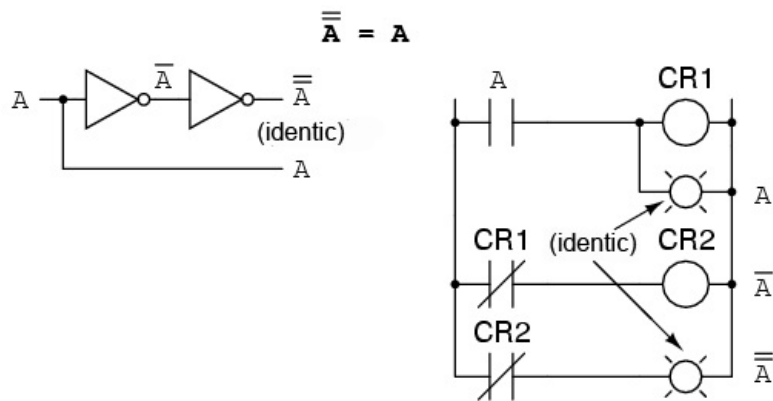


Figure 157: identitate algebrică booleană; complementul dublu

## 7.4 Proprietăți algebrice booleene

Un alt tip de identitate matematică, denumită „proprietate”, descrie relația dintre variabilele unui sistem de numere.

### 7.4.1 Comutativitatea

Una dintre aceste proprietăți poartă numele de comutativitate, și se aplică atât adunării cât și înmulțirii. Ceea ce ne spune comutativitatea este că, putem inversa ordinea variabilelor atât în cazul adunării, cât și în cazul înmulțirii. Rezultatul expresiei rămâne neschimbat în ambele cazuri. Comutativitatea adunării arată astfel:

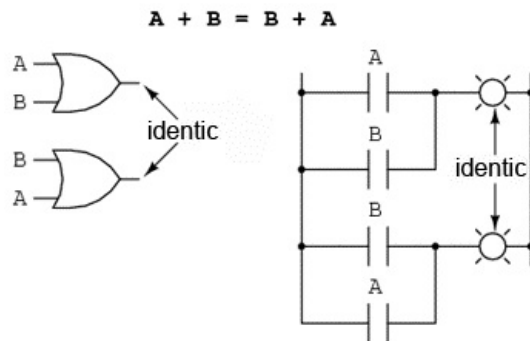


Figure 158: comutativitatea adunării booleene

Comutativitatea înmulțirii:

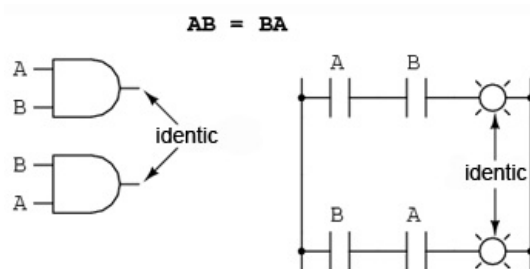


Figure 159: comutativitatea înmulțirii booleene

### 7.4.2 Asociativitatea

Această proprietate spune că putem asocia grupuri de sume sau înmulțiri, prin intermediul parantezelor, fără a modifica rezultatul ecuațiilor. Și în acest caz, asociativitatea se aplică atât adunării cât și înmulțirii. Asociativitatea adunării:

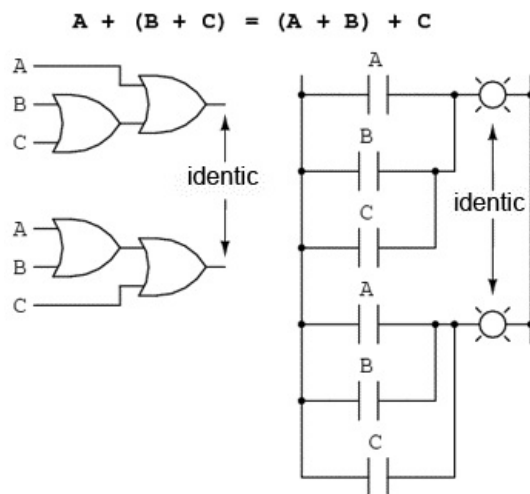


Figure 160: asociativitatea adunării booleene

Asociativitatea înmulțirii:

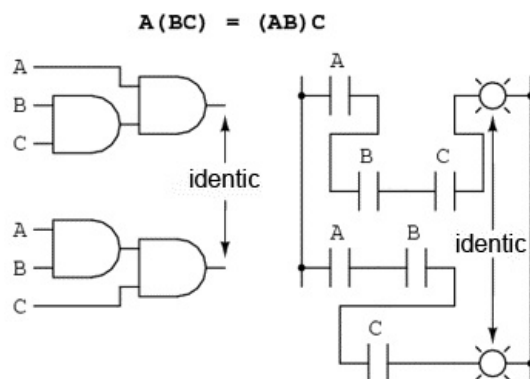


Figure 161: asociativitatea înmulțirii booleene

### 7.4.3 Distributivitatea

Proprietatea de distributivitate precizează modul de dezvoltare a unei expresii booleene formate din înmulțirea unei sume:

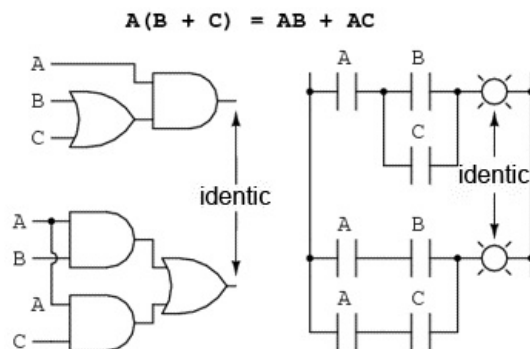


Figure 162: distributivitatea booleană

În concluzie, avem trei proprietăți booleene de bază: comutativitatea, asociativitatea și distributivitatea:

Proprietăți aditive	Proprietăți multiplicative
$A + B = B + A$	$AB = BA$
$A + (B + C) = (A + B) + C$	$A(BC) = (AB)C$
$A(B + C) = AB + AC$	

Figure 163: proprietăți booleene: comutativitatea, asociativitatea și distributivitatea

## 7.5 Reguli de simplificare booleană

Una dintre cele mai practice aplicații ale algebrei booleene constă în simplificarea circuitelor logice. Dacă transformăm funcția logică a unui circuit sub formă booleană, și aplicăm anumite reguli ecuației rezultate, putem reduce numărul termenilor sau operațiilor aritmetice necesare. Ecuația simplificată poate fi apoi transformată înapoi sub formă de circuit logic. Sub noua formă, circuitul logic realizează aceeași funcție, dar cu mai puține componente. Dacă un circuit echivalent poate fi realizat cu mai puține componente, costurile de realizare și de funcționare vor scădea.

Identitățile și proprietățile exprimate în acest secțiunile precedente sunt foarte folositoare simplificării booleene. Toate regulile

prezentate în această secțiune sunt specifice matematicii booleene.

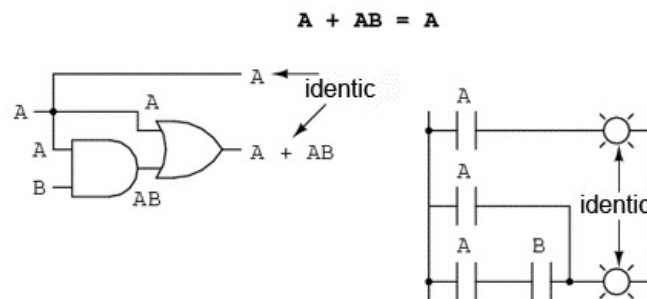


Figure 164: simplificare booleană

Această regulă poate fi demonstrată simbolic prin scoaterea termenului comun (A) în afara sumei. Aplicând apoi regulile  $A + 1 = 1$  și  $1A = A$ , ajungem la rezultatul final:

$$A + AB = A(1 + B) = A(1) = A$$

Observați cum a fost aplicată regula  $A + 1 = 1$  pentru reducerea termenului  $(B + 1)$  la 1. Când aplicăm o regulă precum „ $A + 1 = 1$ ”, exprimată prin intermediul literei „A”, nu înseamnă că aceasta se aplică doar expresiilor ce conțin „A”. A-ul din această expresie exprimă faptul că aceasta se aplică oricărei variabile sau colecții de variabile booleene.

De exemplu, expresia booleană  $ABC + 1$  se reduce tot la 1 prin intermediul aplicării identității  $A + 1 = 1$ . În acest caz, termenul standard „A” din definiția identității reprezintă întregul termen „ABC” al expresiei de mai sus.

Următoarea regulă este aproximativ similară cu prima. Practic, ea este destul de diferită, iar demonstrația este puțin mai dificilă:

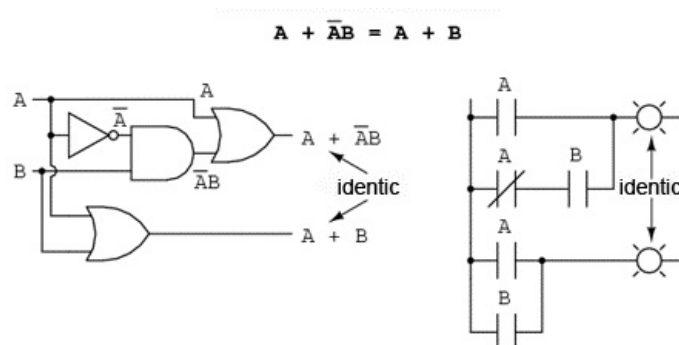


Figure 165: simplificare booleană

Pentru început, dezvoltăm termenul A, folosind regula precedentă ( $A + AB = A$ ). Scoatem termenul B în afara celei de a doua sume, și aplicăm apoi identitatea  $A + A' = 1$ . La sfârșit, nu ne mai rămne decât să aplicăm identitatea  $1A = A$  pentru obținerea rezultatului final:

$$A + A'B = A + AB + A'B = A + B(A + A') = A + B(1) = A + B$$

O altă regulă implică simplificarea expresiei unui produs de sume:

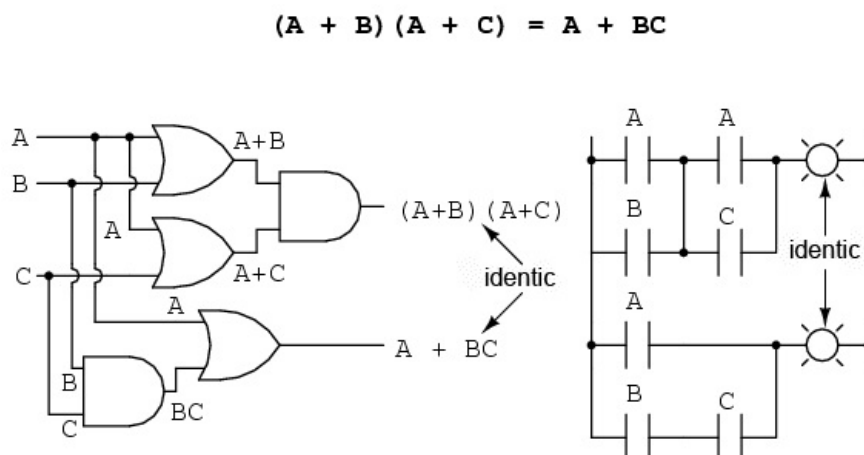


Figure 166: simplificare booleană

Pentru a demonstra această relație, realizăm pentru început înmulțirea celor două sume. Aplicăm apoi identitatea  $AA = A$ , apoi regula  $A + AB = A$  primilor doi termeni. Și, în sfârșit, aplicăm aceeași regulă,  $A + AB = A$  primilor doi termeni a expresiei rezultate. Rezultatul este conform expresiei de mai sus:

$$(A + B)(A + C) = AA + AC + AB + BC = A + AC + AB + BC = A + AB + BC = A$$

- BC

Pe scurt, acestea sunt cele trei reguli ale simplificării booleene:

$$A + AB = A$$

$$A + \bar{A}B = A + B$$

$$(A + B)(A + C) = A + BC$$

Figure 167: regulile simplificării booleene

## 7.6 Simplificarea circuitelor logice

### 7.6.1 Simplificarea circuitelor cu porți logice

Să începem cu un circuit logic cu porți ce necesită o simplificare. Presupunem că intrările A, B și C sunt asigurate de comutatoare, senzori sau alte porți logice. Originea acestor semnale nu este importantă din punct de vedere al simplificării.

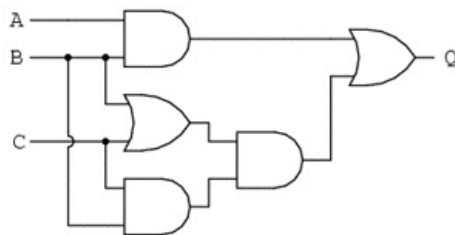


Figure 168: circuit cu porți logice

#### 1. Scrierea expresiei booleene

Primul pas al simplificării constă în scrierea expresiei booleene pentru acest circuit. Acest pas este cel mai ușor de realizat dacă scriem sub-expresii pentru ieșirea fiecărei porți, corespunzător semnalelor de intrare. Este bine să reamintim faptul că porțile SAU sunt echivalente adunării booleene, iar porțile ȘI sunt echivalente înmulțirii booleene. Să scriem așadar sub-expresii la ieșirea primelor trei porți:

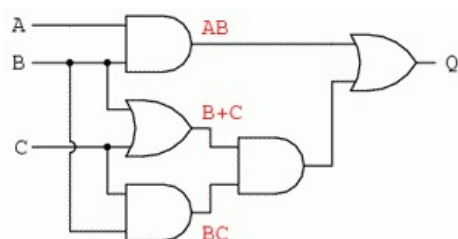


Figure 169: circuit cu porți logice; scrierea sub-expresiilor la ieșirea porților

Scriem apoi sub-expresiile următoarelor seturi de porți. În cazul de față, avem doar o singură poartă pe nivelul următor:

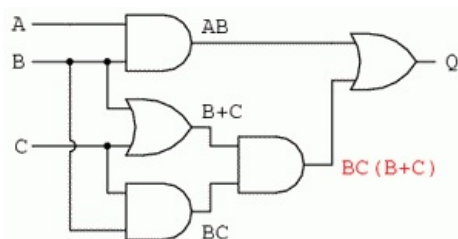


Figure 170: circuit cu porți logice; scrierea sub-expresiilor la ieșirea porților

Și, în sfârșit, ieșirea (Q) circuitului logic este egală cu următoarea expresie:

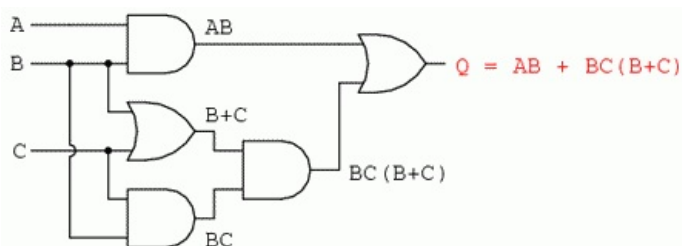


Figure 171: circuit cu porți logice; scrierea sub-expresiilor la ieșirea porților

#### 2. Simplificarea expresiei booleene

Acum că avem o expresie booleană, următorul pas este aplicarea regulilor algebrei booleene pentru reducerea expresiei de mai sus la forma ei cea mai simplă. Reamintim faptul că cea mai simplă formă este aceea care necesită cele mai puține porți logice pentru implementarea ei.

Prin urmare, expresia  $AB + BC(B + C)$  poate fi redusă astfel: la primul pas realizăm înmulțirea termenilor; aplicăm apoi identitatea  $AA = A$  termenilor doi și trei; aplicăm identitatea  $A + A = A$  termenilor doi și trei rezultați; scoatem termenul comun B în față:

$$AB + BC(B + C) = AB + BBC + BCC = AB + BC + BC = AB + BC = B(A + C)$$

Expresia rezultată,  $B(A + C)$ , este mult mai simplă decât cea originală. Ea realizează însă aceeași funcție. Dacă vrei să verificăți acest lucru, puteți construi un tabel de adevăr pentru ambele expresii, Determinați apoi rezultatul Q (ieșirea circuitului) pentru toate cele opt combinații posibile dintre A, B și C pentru ambele circuite. Cele două tabele trebuie să fie identice.

#### 3. Evaluarea expresiei booleene rezultate

Următorul pas constă în generarea unei scheme logice folosind această expresie booleană simplificată. Pentru realizarea acestui lucru, evaluăm expresia urmând ordinea matematică a operațiilor (înmulțirea înainte adunării, operațiile din interiorul parantezelor înaintea celorlalte). La fiecare pas vom adăuga o nouă poartă. Porțile sau sunt echivalente cu adunarea booleană, iar porțile ȘI sunt echivalente operației de înmulțirea booleană. În exemplul de față, începem construirea circuitului cu sub-expresia „A + C”, expresie ce nu este altceva decât o poartă SAU:

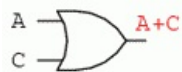


Figure 172: poartă logică SAU

Următorul pas în evaluarea expresiei  $B(A + C)$  constă în înmulțirea (poartă ȘI) semnalului B cu ieșirea porții precedente ( $A + C$ ):

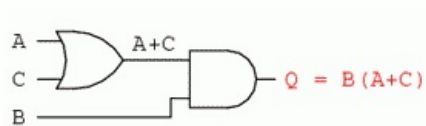


Figure 173: evaluarea expresiei booleene

Evident, acest circuit este mult mai simplu decât cel original, având doar două porți logice în loc de cinci. O astfel de reducere a numărului de componente duce la viteze de funcționare crescute (timpul de propagare a semnalului de la intrare la ieșire este mai scurt), consum de energie mai scăzută, cost mai mic și o fiabilitate crescută.

## 7.6.2 Simplificarea circuitelor cu relee electromecanice

Circuitele cu relee electromecanice pot profita foarte mult de pe urma simplificării booleene. De obicei, acestea sunt mai lente, consumă mult mai multă energie, costă mai mult, iar durata de viață medie este mai scurtă decât cea a porților logice semiconductoare. Să considerăm așadar exemplul de mai jos:

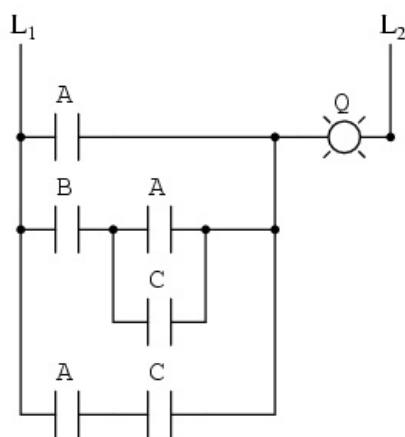


Figure 174: circuit logic cu relee electromecanice

### 1. Scrierea expresiei booleene

Primul pas al reducerii acestui circuit la forma cea mai simplă este, din nou, transformarea circuitului sub forma unei expresii booleene. Cea mai simplă metodă de realizare a acestui lucru este asemănătoare cu metoda reducerii unui circuit rezistiv serie-paralel la o singură rezistență. De exemplu, să considerăm circuitul rezistiv de mai jos, cu rezistorii aranjați asemeni contactelor circuitului precedent.

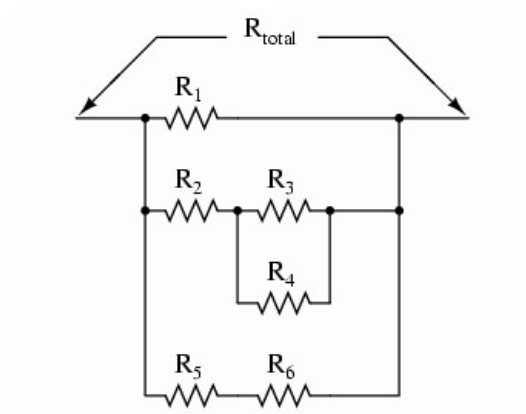


Figure 175: circuit rezistiv serie-paralel

Formula corespunzătoare reducerii acestui circuit la o rezistență echivalentă, este următoarea:

$$R_{\text{total}} = R_1 / [(R_3 / R_4) - R_2] // (R_5 - R_6)$$

Contactele paralele sunt echivalente cu adunarea booleeană, iar contactele serie cu înmulțirea booleeană. Expresia booleeană a circuitului cu relee de mai sus se scrie urmând aceleași reguli care se regăsesc în cazul reducerii circuitelor serie-paralel la o rezistență totală echivalentă. Simplificarea ne este ușurată dacă scriem sub-expresii booleene la stânga fiecărei linii în parte:

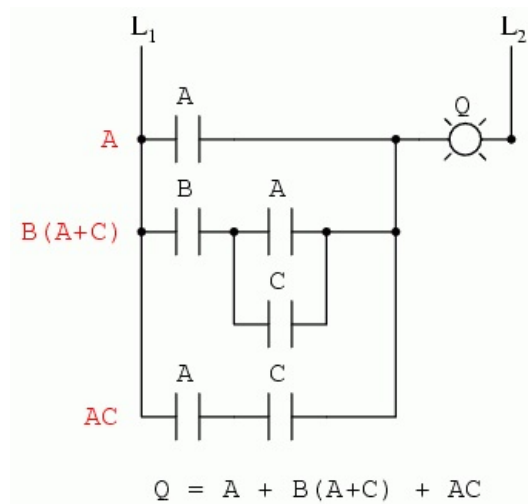


Figure 176: circuit logic cu relee electromecanice

Acum că avem o expresie booleană, tot ceea ce trebuie să facem este să aplicăm regulile de simplificare pentru a aduce expresia la forma ei cea mai simplă (formă ce necesită cele mai puține relee pentru implementarea fizică). Pașii sunt următorii: extindem termenul  $B(A + C)$ ; aplicăm regula  $A + AB = A$  primilor doi termeni; aplicăm regula  $A + AB = A$  primului termen și termenului al treilea:  
 $A + B(A + C) + AC = A + AB + BC + AC = A + BC + AC = A + BC$   
 După cum putem vedea, circuitul redus este mult mai simplu decât originalul, dar funcția logică pe care o îndeplinește este neschimbată:

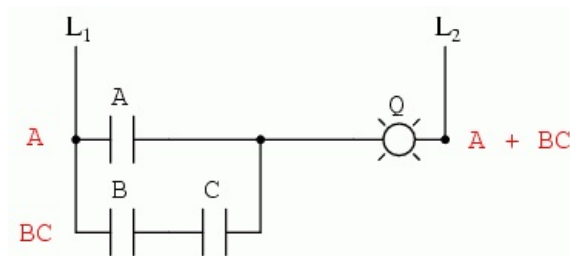


Figure 177: circuit logic cu relee electromecanice; forma simplificată

## 7.7 Funcția SAU-exclusiv

Un element ce nu l-am întâlnit până în acest moment în operațiile booleene este SAU-exclusiv. Deși funcția SAU este echivalentă cu o adunare booleană, funcția ȘI cu înmulțirea iar funcția NU cu complementarea, nu există un echivalent boolean pentru funcția SAU-exclusiv. Acest lucru nu ne împiedică să avem un simbol pentru reprezentarea ei:

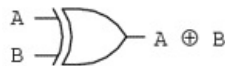
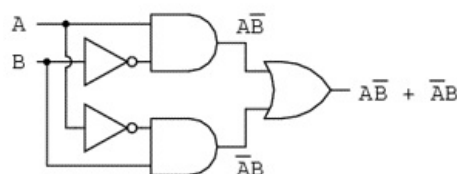


Figure 178: funcția SAU-exclusiv; simbol

Acest simbol este folosit foarte rar în expresiile booleene, deoarece identitățile, proprietățile și regulile de simplificare ce implică adunare, înmulțire și complementare nu se aplică și acestei expresii. Totuși, există o modalitate de reprezentare a funcției SAU-exclusiv cu ajutorul funcțiilor SAU și ȘI:



...este echivalent cu...



$$A \oplus B = A\bar{B} + \bar{A}B$$

Figure 179: funcția SAU-exclusiv realizată cu funcțiile SAU și ȘI

Ca și echivalență booleană, această regulă poate fi folosită în cazul simplificării anumitor expresii booleene. Orice expresie de forma  $AB' + A'B$  (două porți ȘI și o poartă SAU), poate fi înlocuită de o singură poartă SAU-exclusiv.

## 7.8 Teoremele lui DeMorgan

DeMorgan a dezvoltat o serie de reguli importante în algebra liniară cu privire la complementul de grup. Prin complementul de grup ne referim la complementul unui grup de termeni, și nu doar la o singură variabilă.

Țineți minte de la capitolul legat de porți logice, că inversând toate intrările unei porți, inversăm și funcția logică esențială a acesteia. O poartă SAU cu toate intrările inversate (o poartă SAU-negativă) se comportă precum o poartă ȘI-negat. O poartă ȘI cu toate intrările inversate (o poartă ȘI-negativă) se comportă precum o poartă SAU-negat. Teoremele lui DeMorgan exprimă aceeași echivalență în sens invers: inversând ieșirea unei porți, funcția rezultată este aceeași cu tipul opus de poartă cu intrările inversate:

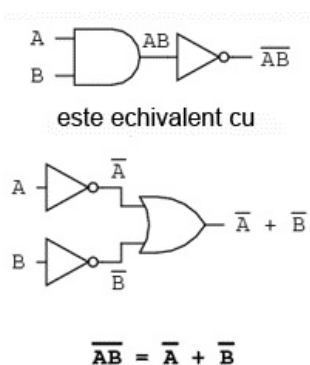


Figure 180: teorema lui DeMorgan

O bară deasupra termenului  $AB$  se comportă precum un simbol de grup. Acest lucru este total diferit față de produsul  $AB$  inversat separat ( $A'B'$ ). Cu alte cuvinte,  $(AB)'$  nu este egal cu  $A'B'$ . Acest lucru are un impact profund asupra modului de evaluare și de reducere a expresiilor booleene, după cum vom vedea.

Teorema lui DeMorgan poate fi gândită ca și „întreruperea” complementului (bara orizontală). Atunci când simbolul complementului este rupt în doua, operația de sub el se modifică din adunare în înmulțirea și invers. După aplicarea teoremei, fiecare variabilă are propriul ei complement. Ca și exemplu:

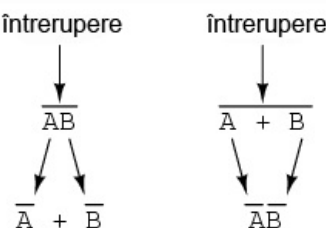


Figure 181: întreruperea complementului în aplicarea teoremei lui DeMorgan

Când există mai multe complemente deasupra aceleiași expresii, nu putem întrerupe decât un complement pe rând. Cel mai ușor este să începem cu cea mai lungă linie orizontală (cea de sus). Ca și exemplu, să considerăm expresia  $(A + (BC)')'$  redusă cu ajutorul teoremelor lui DeMorgan:

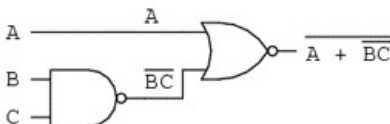


Figure 182: aplicarea teoremei lui DeMorgan

Urmând considerațiile exprimate mai sus, aplicăm următorii pași:

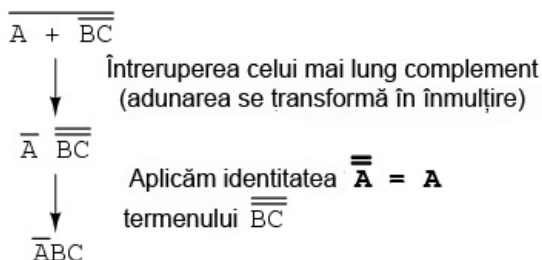


Figure 183: întreruperea complementului în aplicarea teoremei lui DeMorgan

Ca și rezultat, circuitul original este redus la un circuit format dintr-o poartă ȘI cu trei intrări, unde intrarea  $A$  este inversată printr-o poartă NU:

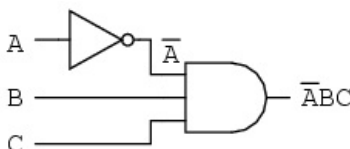


Figure 184: circuit logic simplificat cu ajutorul teoremei lui DeMorgan

Ca și contra-exemplu, nu întrerupeți niciodată mai mult de un complement la un singur pas:



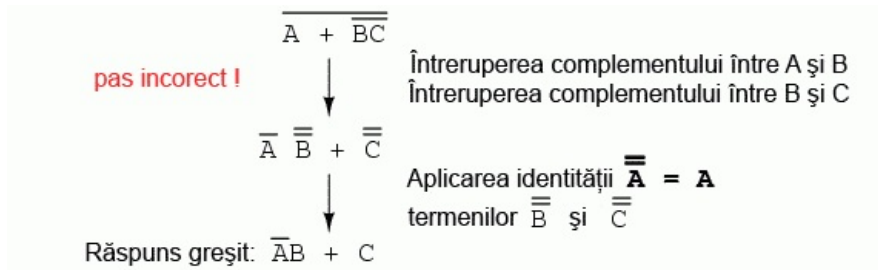


Figure 185: întreruperea greșită a complementului

Pe cât de tentant pare, pe atât de incorect este să scurtăm pași simplificării prin întreruperea mai multor complemente deodată. Prin urmare, nu faceți niciodată acest lucru!

Putem simplifica expresia de mai sus și prin întreruperea complementului scurt în primă instanță, și apoi a complementului lung:

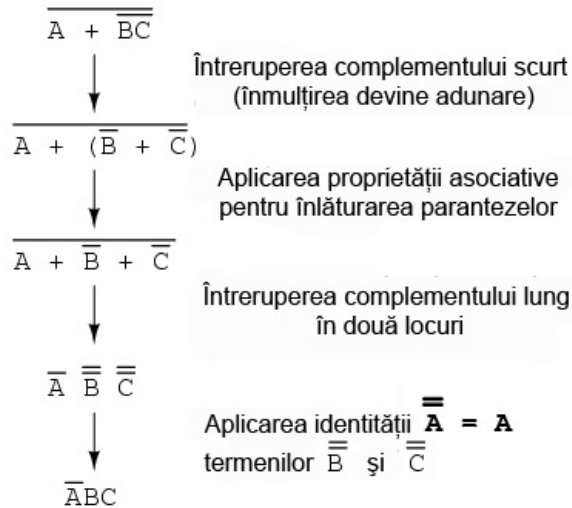


Figure 186: întreruperea complementului scurt

Desigur, rezultatul final este același și în acest caz. Pașii necesari pentru simplificare sunt însă mai numeroși față de exemplul precedent (întreruperea complementului lung la primul pas). La pasul al treilea, în exemplul de mai sus, întreruperea complementului lung se realizează în două locuri simultan. Această operație matematică este permisă, și nu este identică cu întreruperea a două complemente deodată! Interdicția întreruperii mai multor complemente deodată nu interzice întreruperea complementului în mai multe locuri.

Poate vă întrebați de ce am folosit paranteze în jurul sub-expresiei B'

• C', din moment ce oricum le-am îndepărtat la pasul următor. Am făcut acest lucru pentru a sublinia un aspect important dar neglijat al teoremei lui DeMorgan. Din moment ce o linie orizontală lungă funcționează ca și simbol de grup, variabilele incluse sub aceasta trebuie să rămână grupate. În caz contrar, ordinea operațiilor se pierde. În exemplul anterior, nu contează dacă am fi pus sau nu aceste paranteze, dar în alte cazuri s-ar putea să conteze. Să luăm un alt exemplu, menținând parantezele:

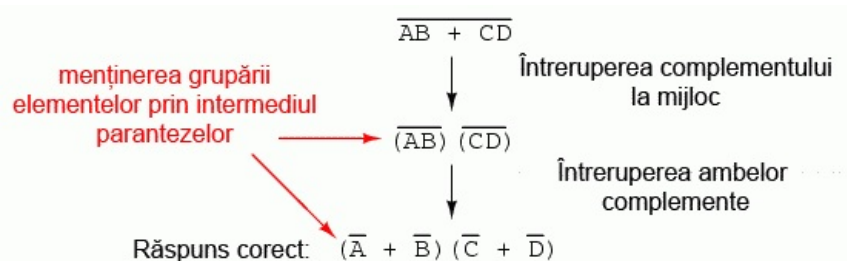


Figure 187: simplificarea expresiei booleene cu ajutorul teoremei lui DeMorgan

În cazul în care nu menținem parantezele, riscăm să obținem un răspuns greșit:

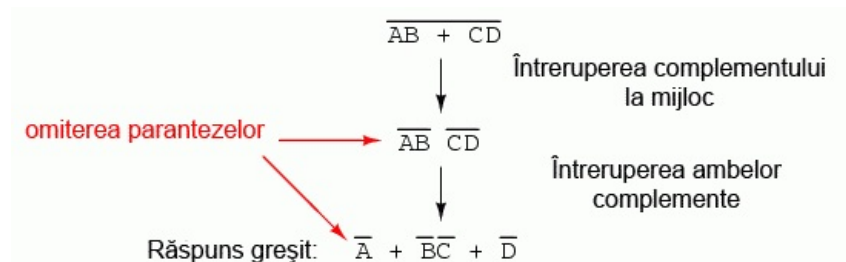


Figure 188: simplificarea expresiei booleene cu ajutorul teoremei lui DeMorgan

După cum se poate observa, menținerea grupării realizate implicit prin liniile de complementare, este crucială pentru obținerea răspunsului corect.

### 7.8.1 Simplificarea unui circuit logic - exemplu

Să aplicăm acum principiile teoremelor lui DeMorgan pentru simplificarea unui circuit cu porți logice:

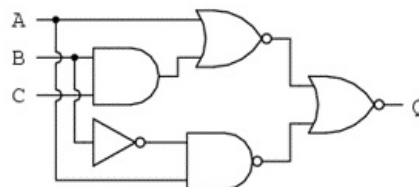


Figure 189: circuit cu porți logice

#### 1. Expresia booleană echivalentă

Ca de obicei, primul pas al simplificării circuitului constă în găsirea expresiei booleene echivalente. Putem face acest lucru prin notarea sub-expresiilor la ieșirea fiecărei porți, pe măsură ce intrările ne sunt cunoscute:

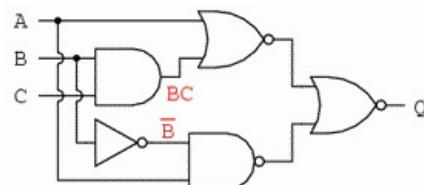


Figure 190: circuit cu porți logice; notarea sub-expresiilor la ieșirea porților

Apoi, notăm ieșirea primei porți SAU-negat și ieșirea porții ȘI-negat. Atunci când aveam de a face cu porți inversate pe ieșire, este mai ușor să scriem prima dată expresia fără inversarea finală. Observați și de pe figură faptul că săgeata indică ieșirea porții chiar înaintea inversării (cerculețul de la ieșire). Expresia finală, după inversare, este complementul expresiei precedente. Astfel, ne putem asigura că nu uităm introducerea complementului în cadrul expresiei:

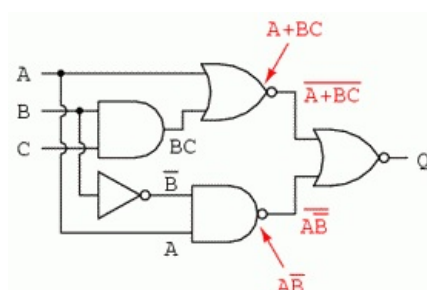


Figure 191: circuit cu porți logice; notarea sub-expresiilor la ieșirea porților

Și, în sfârșit, ultimul pas constă în scrierea expresiei pentru poarta SAU-negat finală:

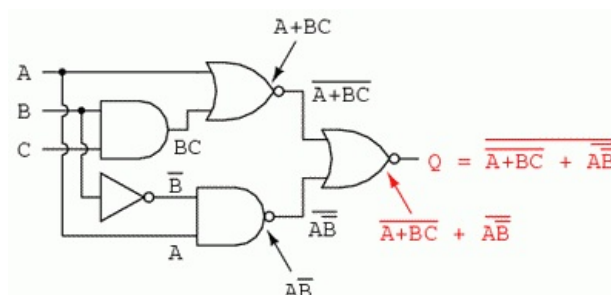


Figure 192: circuit cu porți logice; notarea sub-expresiilor la ieșirea porților

#### 2. Simplificare expresiei echivalente

Trecem apoi la reducerea acestei expresii folosind identitățile, proprietățile, regulile și teoremele (lui DeMorgan) algebrei booleene:

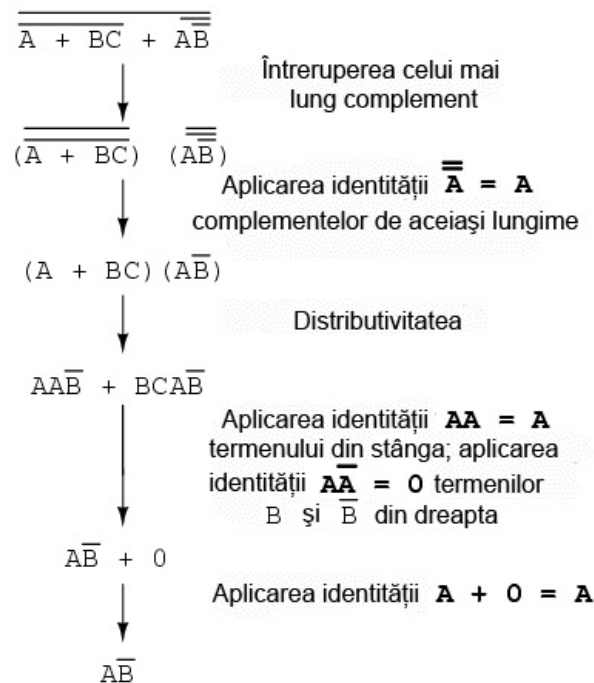


Figure 193: simplificarea expresiei booleene echivalente

### 3. Circuitul echivalent

Circuitul echivalent al expresiei mult simplificate:

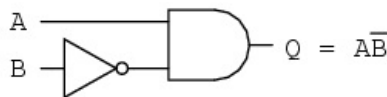


Figure 194: circuit cu porți logice echivalent (simplificat)

## 7.9 Transformarea tabelelor de adevăr în expresii booleene

Procesul de proiectare al circuitelor digitale începe adesea cu un tabel de adevăr. Acest tabel descrie modul de funcționare al circuitului, pe scurt, ce funcții trebuie aceasta să îndeplinească. Partea de proiectare constă în mare parte în determinarea tipului de circuit ce va realiza funcția propusă în acest tabel de adevăr. Deși există unii oameni care pot determina circuitul final prin simpla privire a tabelului de adevăr, pentru noi ceilalți există o serie metode foarte utile. Se va dovedi că algebra booleană este de un real folos în această situație.

Pentru ilustrarea acestor metode, cel mai indicat este să începem cu o problemă de proiectare practică. Să presupunem că trebuie să proiectăm un circuit de detectare a flăcării unui incinerator de deșeuri toxice. Astfel de tehnici de ardere sunt folosite de obicei pentru neutralizarea deșeurilor medicale, ce pot fi infectate cu viruși sau bacterii periculoase:

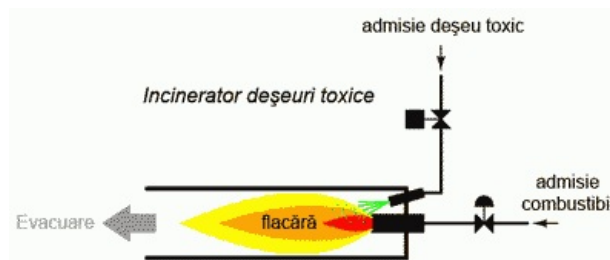


Figure 195: incinerator deșeuri toxice

Atâta timp cât flacăra este menținută în incinerator, injectarea deșeurilor toxice pentru neutralizare este sigură. Dacă în schimb flacăra se stinge, această alimentare a incineratorului se poate dovedi periculoasă. Evacuarea va conține deșeurile toxice ne-neutralizate, reprezentând un pericol de sănătate pentru persoanele aflate în apropiere. Avem nevoie prin urmare de un sistem de detectare a prezenței flăcării. Injectarea deșeurilor va fi permisă doar atunci când sistemul de detectare ne asigură de prezența flăcării.

Există mai multe metode de detectare a flăcării: optic (detectarea luminii), termic (detectarea temperaturii înalte) și conducție electrică (detectarea particulelor ionizate). Fiecare din aceste metode prezintă avantaje și dezavantaje. Să presupunem că, datorită pericolului ridicat al trecerii deșeurilor intacte prin evacuarea sistemului, s-a decis ca sistemul de detectare să fie redundant (senzori multipli). Astfel că, defectarea unuia dintre senzori să nu ducă la o situație nedorită. Fiecare senzor este echipat cu un contact normal-deschis (deschis - lipsă flacăra, închis - flacăra detectată) necesar activării intrărilor unui sistem logic:

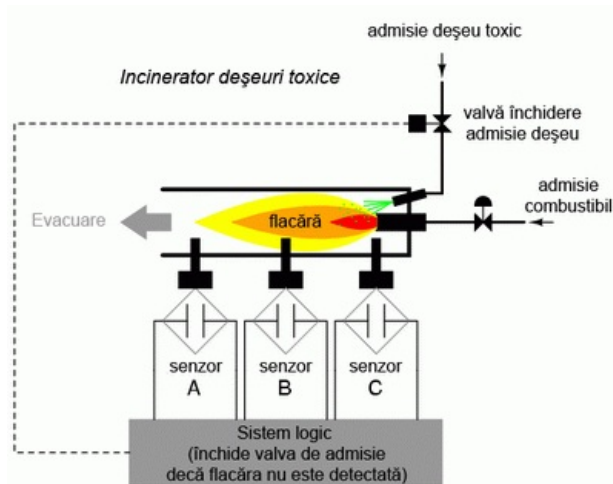


Figure 196: incinerator deșeuri toxice; utilizarea senzorilor și a circuitului logic pentru închiderea alimentării în cazul în care flacăra nu este detectată

Scopul nostru acum, este să proiectăm circuitul logic astfel încât acesta să deschidă valva de admisie doar dacă există flacăra (detectată de senzori). Prima dată trebuie să vedem comportamentul acestui sistem de control. Dorim ca valva să se deschidă în cazul în care doar unul din cei trei senzori detectează flacăra? Probabil că nu. Altfel, nu ar mai avea niciun rost să folosim trei senzori în loc de unul singur. Ceea ce ne dorim de la sistemul logic, este ca acesta să deschidă valva de admisie doar în cazul în care toți cei trei senzori detectează flacăra. În acest caz, tabelul de adevăr arată astfel:

senzori de intrare			ieșire
A	B	C	
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

ieșire = 0 (valvă închisă)

ieșire = 1 (valvă deschisă)

Figure 197: incinerator deșeuri toxice; tabelul de adevăr

Această funcționalitate poate fi asigurată folosind o poartă ȘI cu trei intrări: ieșirea circuitului este 1 doar dacă intrarea A ȘI intrarea B ȘI intrarea C este 1:

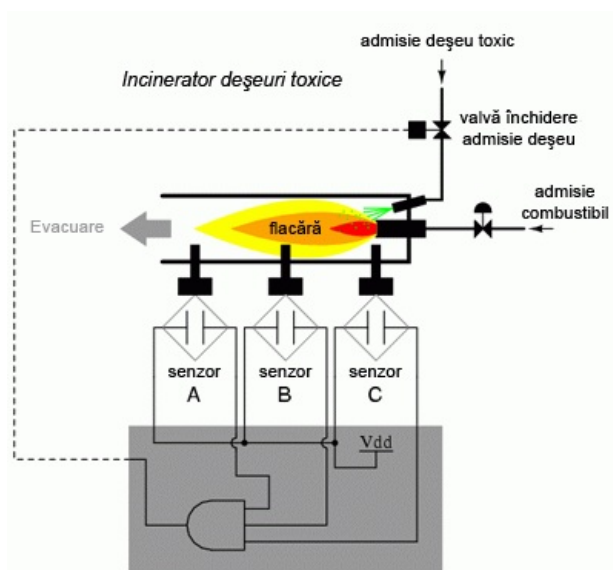


Figure 198: incinerator deșeuri toxice; adăugarea circuitului logic

Dacă folosim în schimb relee electromecanice, putem crea această funcție ȘI prin conectarea celor trei contacte în serie. Sau pur și simplu conectă cei trei senzori în serie, astfel încât, singura modalitate prin care se poate deschide valva de admisie, este dacă toți cei trei senzori indică prezența flăcării:

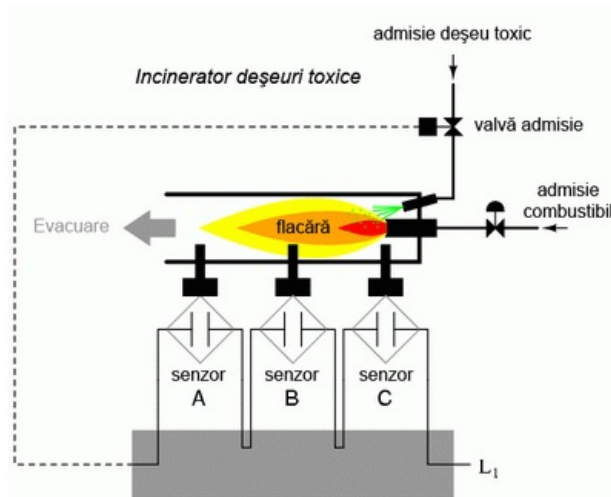


Figure 199: incinerator deșeuri toxice; utilizare relee electromecanice

Deși această strategie maximizează siguranța sistemului, este totuși foarte sensibilă la defect. În cazul în care unul din cei trei senzori se defectează, indicând lipsa flăcării din incinerator, întregul sistem se va opri. Asta chiar dacă ceilalți doi senzori funcționează și indică prezența flăcării. Această oprire „gratuită” a incineratorului duce la pierderi de producție și de combustibil (menținerea unei flăcări ce nu este folosită pentru incinerarea materialului toxic).

Va trebui să reproiectăm sistemul, astfel încât, un astfel de defect să nu ducă la închiderea întregului sistem. Bazându-ne pe doi senzori în detectarea prezenței flăcării, sistemul își păstrează și în acest caz redundanța. O astfel de strategie implică un circuit logic cu trei intrări, a cărui ieșire este 1 în cazul în care cel puțin două din cele trei intrări sunt 1. Tabelul de adevăr arată astfel:

senzori de intrare			ieșire	
A	B	C		
0	0	0	0	ieșire = 0 (valvă închisă)
0	0	1	0	
0	1	0	0	
0	1	1	1	
1	0	0	0	ieșire = 1 (valvă deschisă)
1	0	1	1	
1	1	0	1	
1	1	1	1	

Figure 200: incinerator deșeuri toxice; tabelul de adevăr

### 7.9.1 Suma-de-produse

În această situație nu este foarte clar ce tip de circuit logic ar satisface tabelul de adevăr. O metodă simplă de realizarea a unui astfel de circuit constă în utilizarea unei forme booleene standard, denumită sumă-de-produse. Ca și exemplu, o astfel de expresie ar putea arăta astfel:  $ABC + BC + DF$ , suma produselor  $ABC$ ,  $BC$  și  $DF$ .

Astfel de expresii sunt relativ ușor de realizat cu ajutorul tabelelor de adevăr. Trebuie doar să găsim acele rânduri din tabel unde ieșirea este 1, și să scriem apoi un produs boolean a cărui rezultat să fie 1, cunoscând condițiile de intrare. De exemplu, să luăm al patrulea rând din tabelul de adevăr de mai sus. Ieșirea acestuia este 1 (ceea ce căutăm), iar intrările sunt  $A = 0$ ,  $B = 1$  și  $C = 1$ . Produsul acestor trei variabile este unu dacă expresia arată astfel:  $A'BC$ .

senzori de intrare			ieșire	
A	B	C		
0	0	0	0	$\bar{A}BC = 1$
0	0	1	0	
0	1	0	0	
0	1	1	1	
1	0	0	0	
1	0	1	1	
1	1	0	1	
1	1	1	1	

Figure 201: incinerator deșeuri toxice; tabelul de adevăr

Să completăm și celelalte rânduri care au o ieșire de 1, cu produsul termenilor:

senzori de intrare			
A	B	C	ieșire
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

$$\overline{A}BC = 1$$

$$A\overline{B}C = 1$$

$$AB\overline{C} = 1$$

$$ABC = 1$$

Figure 202: incinerator deșeuri toxice; tabelul de adevăr

Însumăm toate aceste patru expresii, pentru a crea o singură expresie booleană ce descrie în întregime tabelul de adevăr:

senzori de intrare			
A	B	C	ieșire
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

$$\overline{A}BC = 1$$

$$A\overline{B}C = 1$$

$$AB\overline{C} = 1$$

$$ABC = 1$$

$$\text{ieșire} = \overline{A}BC + A\overline{B}C + AB\overline{C} + ABC$$

Figure 203: incinerator deșeuri toxice; tabelul de adevăr

### 7.9.2 Realizarea circuitului logic

După ce am obținut expresia booleană sub formă de sumă-de-produse, putem trece la realizarea circuitului logic bazat pe această expresie, fie cu porți logice:

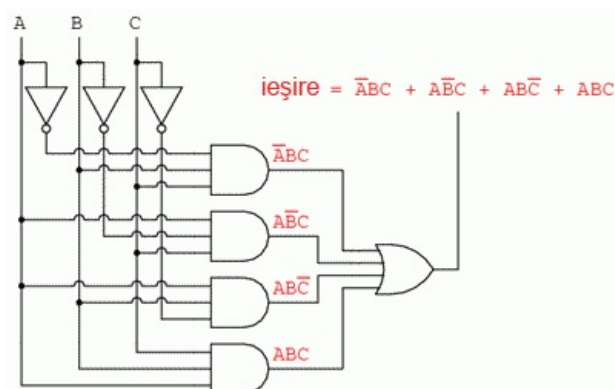


Figure 204: incinerator deșeuri toxice; circuitul logic (porți logice)

Fie cu relee electromecanice:

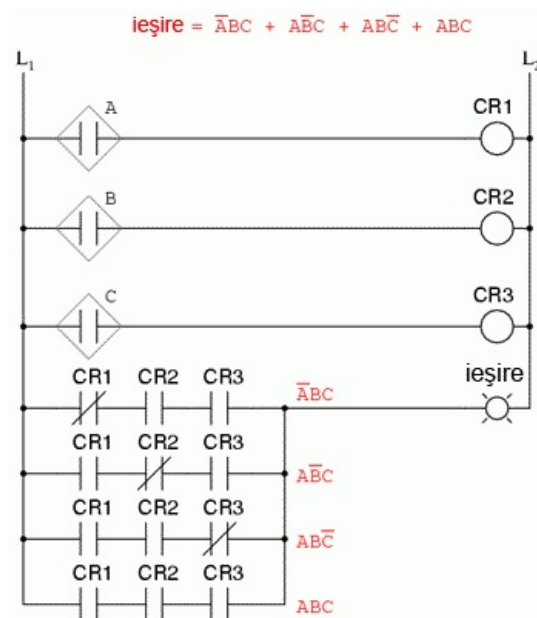


Figure 205: incinerator deșeuri toxice; circuitul logic (relee electromecanice)

### 7.9.3 Simplificarea expresiei booleene

Din păcate, ambele variante sunt destul de complexe. Din fericire însă, putem simplifica expresia inițială folosind regulile simplificării booleene:

$$\begin{aligned}
 & \overline{A}BC + A\overline{B}C + AB\overline{C} + ABC \\
 & \downarrow \text{BC comun termenilor 1 și 4} \\
 & BC(\overline{A} + A) + A\overline{B}C + AB\overline{C} \\
 & \downarrow \text{Aplicarea identității } A + \overline{A} = 1 \\
 & BC(1) + A\overline{B}C + AB\overline{C} \\
 & \downarrow \text{Aplicarea identității } 1A = A \\
 & BC + A\overline{B}C + AB\overline{C} \\
 & \downarrow \text{B comun termenilor 1 și 3} \\
 & B(C + A\overline{C}) + A\overline{B}C \\
 & \downarrow \text{Aplicarea regulii } A + \overline{A}B = A + B \text{ termenului } C + A\overline{C} \\
 & B(C + A) + A\overline{B}C \\
 & \downarrow \text{Distribuirea termenilor} \\
 & BC + AB + A\overline{B}C \\
 & \downarrow \text{A comun termenilor 2 și 3} \\
 & BC + A(B + \overline{B}C) \\
 & \downarrow \text{Aplicarea regulii } A + \overline{A}B = A + B \text{ termenului } B + \overline{B}C \\
 & BC + A(B + C) \\
 & \downarrow \text{Distribuirea termenilor} \\
 & BC + AB + AC \\
 & \text{sau} \\
 & AB + BC + AC
 \end{aligned}$$

**Rezultat simplificat**

Figure 206: incinerator deșeuri toxice; simplificarea expresiei booleene

Ca și rezultat al simplificării, putem acum construi un circuit logic mult simplificat, dar care îndeplinește exact aceeași funcție logică, fie cu porți logice:



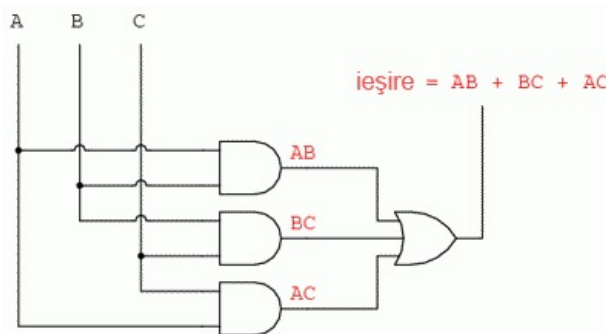


Figure 207: incinerator deșeuri toxice; circuitul logic (porți logice)

Fie cu relee electromecanice:

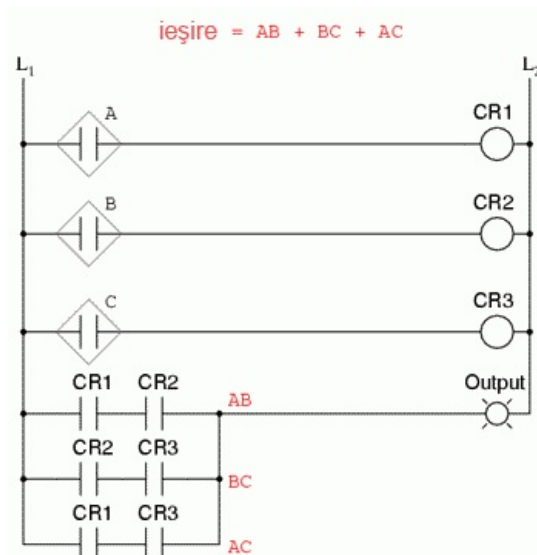


Figure 208: incinerator deșeuri toxice; circuitul logic (relee electromecanice)

## 8 Harti karnaugh

### 8.1 De ce hărți Karnaugh

La ce ne folosesc hărțile Karnaugh? Harta Karnaugh, asemenea algebrei booleene, este o metodă de simplificare a circuitelor logice digitale. Vedeți exemplul „incineratorului de deșeuri toxice” ca și exemplu de simplificare booleană a unui circuit logic. Harta Karnaugh va simplifica circuitul mult mai rapid și mai ușor în majoritatea cazurilor.

Simplificarea booleană este de fapt mai rapidă în cazul în care avem maxim două variabile booleene. Putem folosi această metodă chiar și în situația în care avem trei variabile, dar metoda booleană este mai greoaie în acest caz. Cu patru variabile de intrare, algebra booleană devine „imposibilă”. Hărțile Karnaugh sunt mai rapide și mai ușor de implementat. Acestea pot fi folosite cu succes în situațiile în care avem până la șase variabile de intrare. Între șase și opt, mai putem încă folosi aceste hărți. Peste această valoare, este indicat să folosim un program software specializat pentru realizarea simplificărilor logice.

### 8.2 Diagrame Venn

Matematicienii utilizează diagramele Venn pentru reprezentarea relațiilor logice dintre mulțimi (colecții de obiecte). Ne vom folosi de diagramele Venn pentru a face tranziția dintre algebra booleană și hărțile Karnaugh.

#### 8.2.1 Mulțimi și submulțimi

O mulțime este o colecție de obiecte dintr-un univers dat. Elementele mulțimii sunt obiecte ce aparțin mulțimii. Elementele unei mulțimi au de obicei ceva în comun, deși acest lucru nu este neapărat necesar. Din universul numerelor reale, de exemplu, mulțimea tuturor numerelor întregi pozitive  $\{1, 2, 3 \dots\}$ , este o mulțime. Mulțimea  $\{3, 4, 5\}$  este o mulțime mai mică, sau o submulțime a mulțimii numerelor întregi pozitive. Un alt exemplu este mulțimea tuturor băieților dintr-o clasă (reprezentând universul discuției). Vă puteți gândi și la alte mulțimi?

#### 8.2.2 Diagrame Venn - exemple

Diagrama Venn din figura de mai jos stânga, reprezintă mulțimea A (în interiorul cercului) din universul U (aria dreptunghiulară). Dacă tot ceea ce se află în interiorul cercului este A, atunci tot ceea ce se află în exteriorul cercului nu este A (A-negat). Prin urmare, în figura de mai jos centru, denumit aria dreptunghiulară din afara cercului A cu A-negat în loc de U. B și B-negat se reprezintă similar (figura de mai jos dreapta).



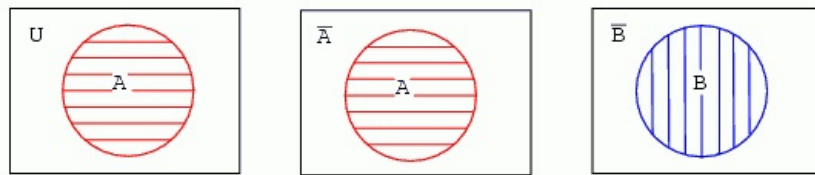


Figure 209: diagrame Venn

Ce se întâmplă dacă și A și B se află în același univers? Există patru posibilități:

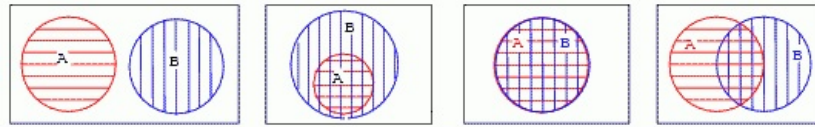


Figure 210: diagrame Venn

Să reluăm fiecare din cele patru posibilități în parte:

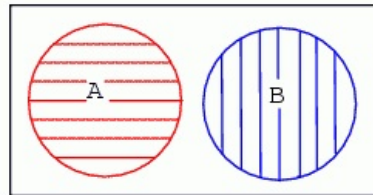


Figure 211: diagrame Venn

Primul exemplu indică faptul că mulțimile A și B nu au niciun element comun, conform diagramei Venn. Regiunile celor două mulțimi nu se suprapun în niciun punct. De exemplu, să presupunem că mulțimile A și B ar conține următoarele elemente:  $A = \{1, 2, 3, 4\}$ ,  $B = \{5, 6, 7, 8\}$ . Niciunul dintre elementele mulțimii A nu este inclus în mulțimea B și invers. Prin urmare, cele două cercuri nu se suprapun.

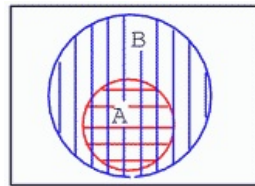


Figure 212: diagrame Venn

În cel de al doilea exemplu, mulțimea A este inclusă total în mulțimea B. Cum putem explica această situație? Să presupunem că mulțimile A și B conțin următoarele elemente:  $A = \{1, 2\}$ ,  $B = \{1, 2, 3, 4, 5, 6, 7, 8\}$ . Toate elementele din A se regăsesc și în B. Prin urmare, mulțimea A este o submulțime a mulțimii B, iar cercul A este inclus în cercul B.

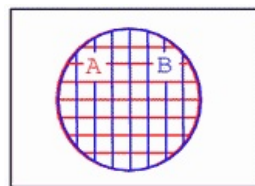


Figure 213: diagrame Venn

În cel de al treilea caz, mulțimile A și B se suprapun perfect. Din diagrama Venn, putem deduce că cele două mulțimi conțin exact aceleași elemente. Să presupunem că mulțimile arată astfel:  $A = \{1, 2, 3, 4\}$  și  $B = \{1, 2, 3, 4\}$ . Prin urmare  $A = B$ . Cele două mulțimi sunt identic egale deoarece conțin exact aceleași elemente.

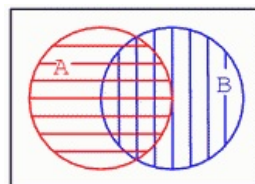


Figure 214: diagrame Venn

În ultimul caz, cele două mulțimi se suprapun, dar nu complet ci doar parțial. Acest lucru ne spune că există elemente comune celor două mulțimi, dar fiecare mulțime are și elemente unice. Să presupunem că cele două mulțimi ar arăta astfel:  $A = \{1, 2, 3, 4\}$  și  $B = \{3, 4, 5, 6\}$ . Ambele mulțimi conțin elementele 3 și 4. Acesta este și motivul pentru care cele două cercuri sunt suprapuse.

## 8.3 Relații booleene cu diagrame Venn

### 8.3.1 Funcția logică SAU (adunarea booleană)

În ultimul exemplu din secțiunea precedentă, mulțimile A și B s-au suprapus parțial. Inițial, ne vom concentra atenția asupra întregii regiuni hașurate de mai jos, abia apoi vom trece la analiza regiunii comune celor două mulțimi. Să utilizăm expresii booleene

pentru desemnarea regiunilor diagramelor Venn, conform figurii de mai jos:

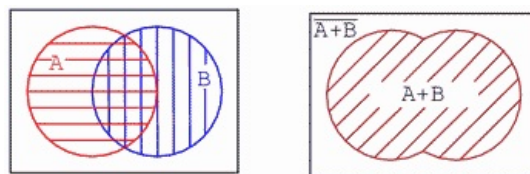


Figure 215: diagrame Venn

Aria mulțimii A este hașurată cu roșu, iar cea a mulțimii B cu albastru. Dacă analizăm întreaga aria hașurată (suma totală a tuturor ariilor hașurate), indiferent de culoare sau stil, obținem figura din dreapta sus. Aceasta corespunde funcției logice SAU, iar expresia booleană este  $A + B$ , aria fiind cea hașurată cu linii diagonale. Tot ceea ce se află în afară ariei hașurate reprezintă  $(A + B)$ -negat.

### 8.3.2 Funcția logică ȘI (înmulțirea booleană)

O altă metodă de interpretare a diagramei Venn cu regiuni suprapuse, este analizarea regiunii comune atât mulțimii A cât și mulțimii B, aria dublu hașurată de mai jos (stânga). Această arie corespunde funcției logice ȘI, iar expresia booleană este  $AB$  (jos dreapta). Tot ceea ce se află în afara ariei dublu hașurate  $AB$  reprezintă  $AB$ -negat:

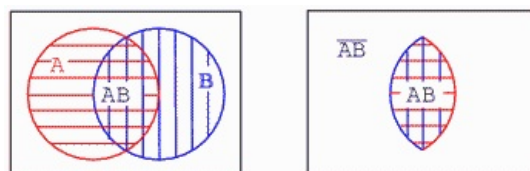


Figure 216: diagrame Venn

Observații că unele elemente ale mulțimilor A și B de sus, sunt elemente ale mulțimii  $(AB)$ -negat, dar niciunul dintre elementele mulțimii  $(AB)$ -negat nu se află în interiorul ariei dublu hașurate  $AB$ .

### 8.3.3 Expresii booleene

#### 1. Diagrama Venn pentru $A'B$

Vom trece acum la dezvoltarea unei expresii booleene. De exemplu, să presupunem că dorim reprezentarea prin diagrame Venn a expresiei booleene  $A'B$  ( $A$ -negat ȘI  $B$ ).

Pașii sunt următorii: hașurarea ariei  $A'$  ( $A$ -negat); hașurarea ariei  $B$ ; realizarea funcției ȘI ( $A'B$ ) prin suprapunerea celor două regiuni precedente. Am putea să ne oprim aici, dar, pentru claritate, putem păstra doar aria dublu hașurată:

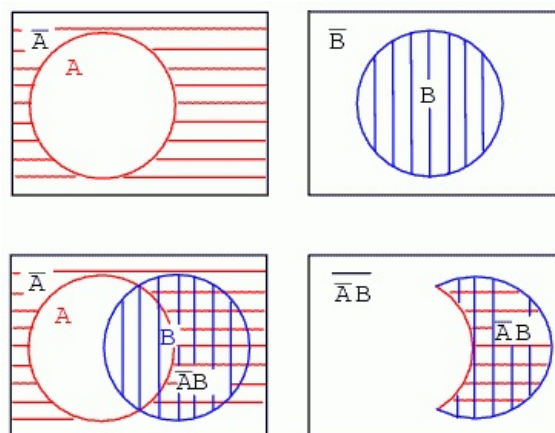


Figure 217: diagrame Venn

Expresia  $A'B$  reprezintă regiunea în care  $A'$  și  $B$  se suprapun. Regiunea nehașurată din afara ariei  $A'B$  este  $(A'B)'$ .

#### 2. Diagrama Venn pentru $B' + A$

Putem încerca același lucru cu expresia booleană SAU. De exemplu, să presupunem că dorim să reprezentăm prin diagrame Venn expresia  $B' + A$ .

Pașii sunt următorii: începem cu hașurarea lui  $B$ , și apoi a regiunii  $B'$ ; suprapunem  $A$  peste  $B'$ . Din moment ce suntem interesați de realizarea funcției SAU, vom căuta să reprezentăm întreaga arie formată de cele două mulțimi, indiferent de stilul hașurării. Prin urmare  $A + B'$  reprezintă întreaga arie hașurată:

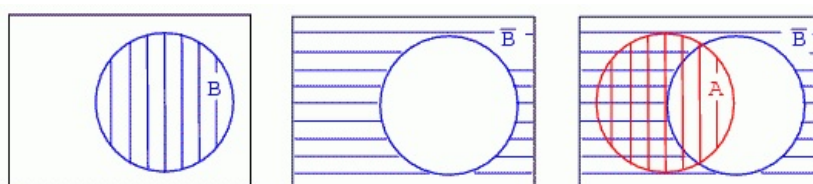


Figure 218: diagrame Venn

Pentru claritate, putem reprezenta întreaga regiune printr-o singură hașurare (jos stânga):

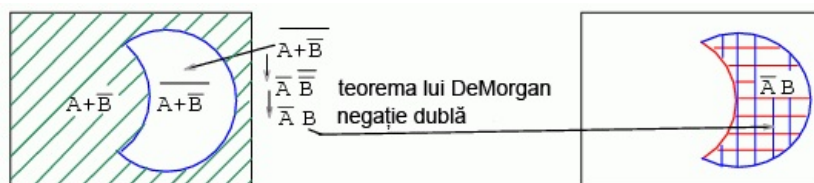


Figure 219: diagrame Venn

### 3. Diagrama Venn pentru $(A + B)'$

Aria hașurată cu verde de mai sus este rezultatul expresiei  $A + B'$ . Trecând la  $(A + B)'$ , căutam complementul expresiei  $A + B'$ , reprezentat prin aria nehașurată din figura de mai sus stânga. Aplicând teorema lui DeMorgan și negarea dublă ( $A'' = A$ ), ajungem la rezultatul  $(A + B)'' = AB$ . Prin urmare, cele două regiuni sunt identice.

Putem face acum observația că diagramele Venn nu demonstrează nimic. Avem nevoie de algebra booleană pentru acest lucru. Totuși, diagramele Venn pot fi utilizate pentru verificare și vizualizare. În exemplul de mai sus, am verificat și vizualizat teorema lui DeMorgan cu ajutorul unei diagrame Venn.

### 4. Diagrama Venn pentru $A' + B'$ și $(A' + B)'$

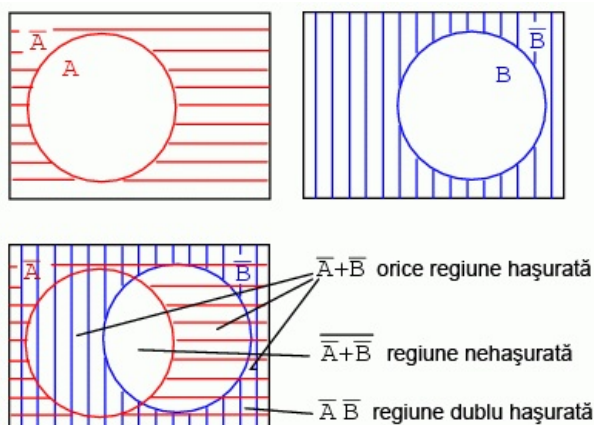


Figure 220: diagrame Venn

### 5. Arătați că $A' + B' = AB$

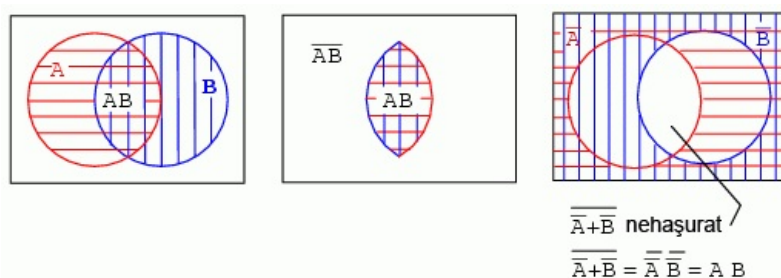


Figure 221: diagrame Venn

## 8.3.4 Diagrame Venn cu 3 variabile

Diagrama Venn de mai jos conține trei regiuni hașurate, A (roșu), B (albastru) și C (verde). Intersecția tuturor regiunilor în centru reprezintă expresia booleană  $ABC$ . Există o altă regiune unde A și B se intersectează, reprezentând expresia booleană  $AB$ . Similar, intersecția ariei A cu C și B cu C reprezintă expresia booleană  $AC$ , respectiv  $BC$ .

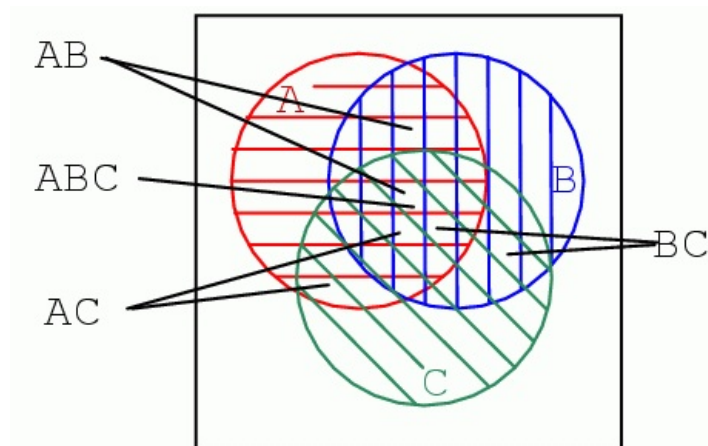


Figure 222: diagramă Venn cu trei variabile

Observând mărimea regiunilor descrise de funcția ȘI de mai sus, putem vedea că mărimea regiunii variază cu numărul variabilelor asociate expresiei ȘI.

## 8.4 Transformarea diagramelor Venn în hărți Karnaugh

### 8.4.1 Hărți Karnaugh cu două variabile

Începem transformarea unei diagrame Venn într-o hartă Karnaugh prin desenarea unei mulțimi A în universul A' (figura de mai jos, a):

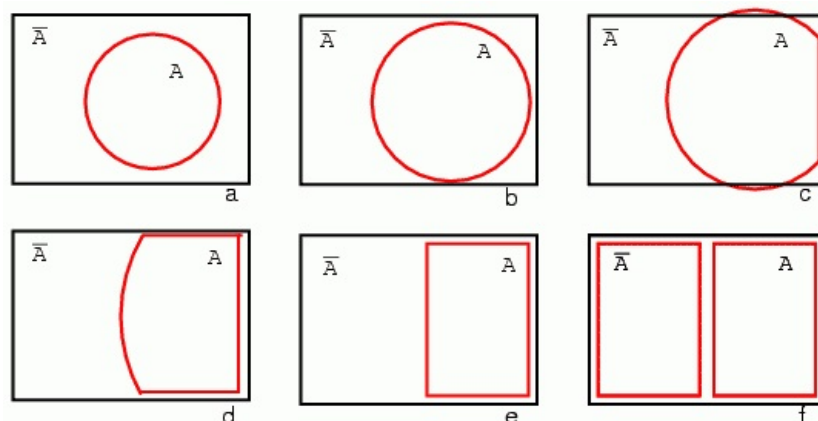


Figure 223: diagrame Venn

Extindem apoi cercul A (b și c), modificăm forma lui la punctul (d), și transformăm A într-un dreptunghi (e). Tot ceea ce nu se află în A este A'. Desenăm un dreptunghi și pentru A' la punctul (f). De asemenea, nu folosim hașuri pentru hărțile Karnaugh. Ceea ce avem până în acest moment este o hartă Karnaugh cu o singură variabilă. Acest lucru nu ne ajută însă. Avem nevoie de variabile multiple. Figura (a) de mai jos este identică diagramei Venn precedente, cu diferența că notațiile A și A' se afla deasupra diagramei și nu în interior. Urmând un proces similar, putem construi „o diagramă Venn dreptunghiulară” pentru B și B' (b). Vom trece acum la suprapunerea diagramelor de la (a) și (b) pentru obținerea rezultatului (c), la fel cum am făcut pentru diagramele Venn. Motivul pentru care realizăm acest lucru este pentru a observa ceea ce este comun celor două regiuni suprapuse - de exemplu, locul în care A se suprapune cu B. Pătratul din dreapta jos (c) corespunde relației AB, unde A se suprapune cu B:

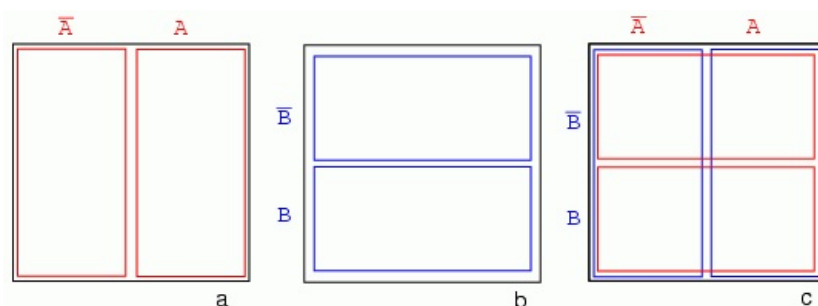


Figure 224: diagrame Venn

Totuși, nu vom pierde vremea desenând hărți Karnaugh precum cea de mai sus (c), ci vom folosi o versiune simplificată:

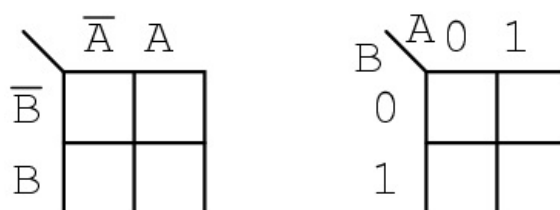


Figure 225: hărți Karnaugh

Coloana formată din cele două celule de sub A' este asociată mulțimii A' (stânga); similar pentru celelalte mulțimi. Pentru simplitate, regiunile nu vor fi delimitate atât de clar precum în cazul diagramelor Venn.

Harta Karnaugh din dreapta este o formă alternativă utilizată în majoritatea textelor. Numele variabilelor sunt trecute lângă linia diagonală. A-ul de deasupra diagonalei indică faptul că variabila A (și A') aparține coloanelor. 0 este folosit pentru A' iar 1 pentru A. B-ul de sub diagonală este asociat cu liniile: 0 pentru B' și 1 pentru B.

#### 1. Exemplu

Marcați căsuțele corespunzătoare expresiei booleene AB în diagrama Karnaugh de mai sus cu 1. Soluție: hașurăm sau încercuim regiunea corespunzătoare lui A; marcăm apoi regiunea corespunzătoare lui B. Intersecția celor două regiuni reprezintă AB; trecem un 1 în această căsuță. Nu este însă necesar să încercuim propriu-zis regiunile A și B:

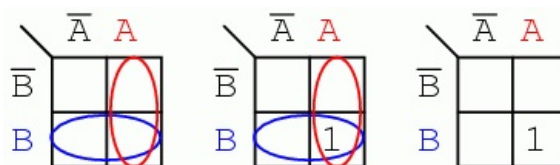


Figure 226: hărți Karnaugh

### 8.4.2 Hărți Karnaugh cu trei variabile

Trecem acum la dezvoltarea unei hărți Karnaugh pornind de la diagrame Venn. Universul (interiorul dreptunghiului negru) este împărțit în două regiuni înguste  $A'$  și  $A$ .  $B$  și  $B'$  împart universul în două regiuni pătrate.  $C$ -ul ocupă o regiune pătrată în mijlocul dreptunghiului, iar  $C'$  este împărțit în două dreptunghiuri verticale de fiecare parte a pătratului  $C$ :

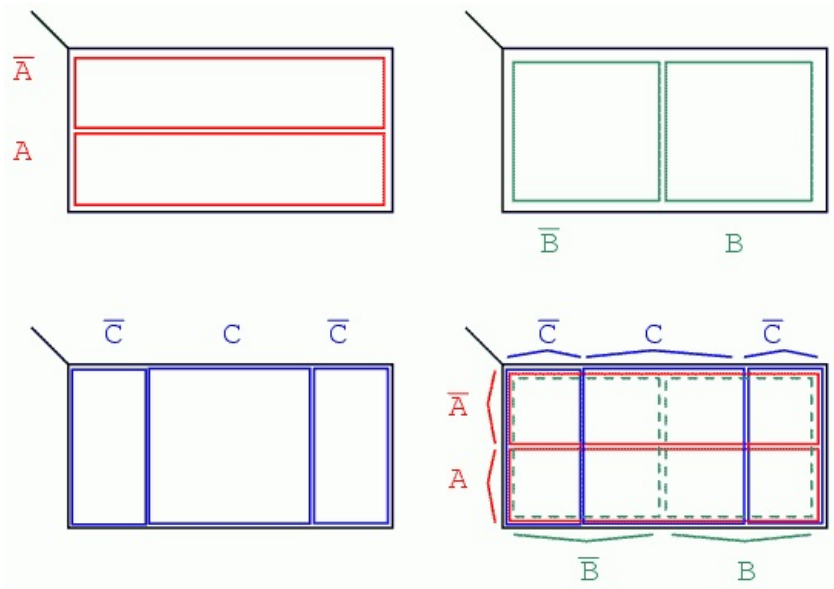


Figure 227: hărți Karnaugh cu trei variabile

În figura finală suprapunem toate cele trei variabile, încercând să delimităm clar fiecare regiune. Această hartă Karnaugh cu 3 variabile are  $2^3 = 8$  regiuni, căsuțele din interiorul hărții. Fiecare regiune este unic determinată prin intermediul celor trei variabile booleene ( $A$ ,  $B$  și  $C$ ). De exemplu  $ABC'$  reprezintă regiunea din dreapta jos (\*), iar  $A'B'C'$  reprezintă regiunea din stânga sus (x):

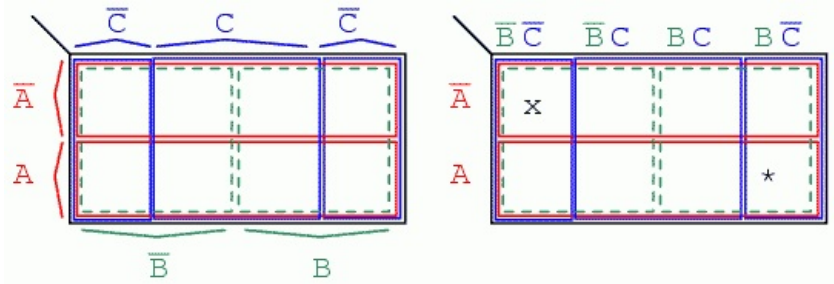


Figure 228: hărți Karnaugh cu trei variabile

Totuși, în mod normal nu vom nota o hartă Karnaugh conform figurii de mai sus stânga. Notarea hărților Karnaugh se va face conform figurii din dreapta. Fiecare regiune este unic determinată printr-un produs de 3 variabile, o expresie booleană ȘI. Cele două forme diferite de mai jos sunt echivalente, și reprezintă forma finală a acestora. Versiunea din dreapta este puțin mai ușor de folosit, din moment ce nu suntem nevoiți să scriem toate variabilele de fiecare dată, ci doar 1 și 0. Noația  $B'C'$ ,  $B'C$ ,  $BC$  și  $BC'$  din stânga este echivalentă cu 00, 01, 11 respectiv 10 din dreapta.  $A$  și  $A'$  sunt echivalente cu 0 respectiv 1.

### 8.5 Hărți Karnaugh, tabele de adevăr și expresii booleene

Hărțile Karnaugh simplifică funcțiile logice mult mai rapid și mai ușor în comparație cu algebra booleană. Dorim simplificarea circuitelor logic spre cel mai mic cost posibil prin eliminarea componentelor. Definim cel mai mic cost ca fiind cel mai mic număr de porți cu cel mai mic număr de intrări pe poarta.

Mai jos am reprezentat cinci metode diferite de reprezentare a aceluiași lucru: o funcție logică aleatoare cu două intrări. Metodele sunt: logica ladder, porți logice, tabel de adevăr, hartă Karnaugh și ecuație booleană. Ceea ce vrem să sublinem este că toate acestea sunt echivalente. Două intrări  $A$  și  $B$  pot lua valori de 0 sau 1, înalt sau jos, deschis sau închis, adevărat sau fals, în funcție de caz. Există  $2^2 = 4$  combinații pentru generarea unei ieșiri. Acest lucru se aplică tuturor celor cinci exemple.



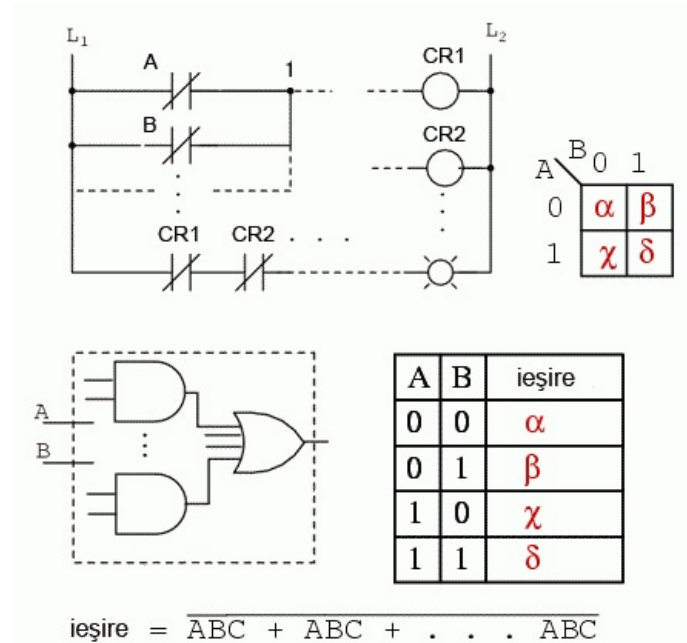


Figure 229: reprezentarea unei funcții logice prin metode diferite

Aceste patru ieșiri pot fi observate prin intermediul unei lampi la ieșirea circuitului ce utilizează logica ladder. Aceste ieșiri pot fi înregistrate într-un tabel de adevăr sau într-o hartă Karnaugh. Priviți harta Karnaugh ca și un tabel de adevăr „cosmetizat”. Ieșirea ecuației booleene poate fi obținută cu ajutorul legilor algebrei booleene și transferată tabelului de adevăr sau hărții Karnaugh. Care din cele cinci metode echivalente de reprezentare ar trebui să o folosim? Cea mai folositoare pentru situația în cauză. Ieșirile unui tabel de adevăr corespund unu-la-unu elementelor unei hărți Karnaugh. Începând cu partea de sus a tabelului de adevăr, intrările  $A = 0$  și  $B = 0$  produc ieșirea  $\alpha$ . Observă că aceeași ieșire,  $\alpha$ , se regăsește pe harta Karnaugh la adresa  $A = 0$ ,  $B = 0$ , în partea de sus stânga, la intersecția coloanei  $B = 0$  cu rândul  $A = 0$ . Celelalte ieșiri ale tabelului de adevăr,  $\beta$ ,  $\chi$  respectiv  $\delta$ , corespund intrărilor  $AB = 01$ ,  $10$  respectiv  $11$  au de asemenea corespundent pe harta Karnaugh:

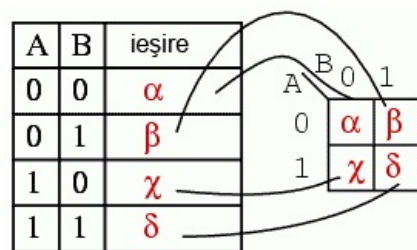


Figure 230: harta Karnaugh

Pentru ușurința expunerii, prezentăm mai jos regiunile adiacente ale hărții Karnaugh cu două variabile folosind metoda dreptunghiulară a diagramei Venn din secțiunea precedentă:

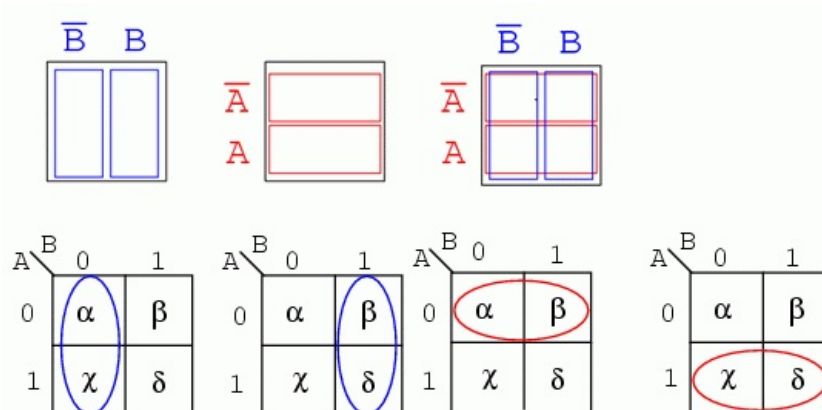


Figure 231: hărți Karnaugh și diagrame Venn

Regiunile  $\alpha$  și  $\chi$  sunt adiacente pe harta Karnaugh. Nu putem spune același lucru despre tabelul de adevăr precedent, întrucât există o altă valoare ( $\beta$ ) între ele. Acesta este și motivul organizării hărților Karnaugh sub formă de matrice pătrată. Regiunile cu variabile booleene comune trebuie să se afle una lângă cealaltă. Această structură este și trebuie să fie ușor de recunoscut când privim o astfel de hartă, din moment ce  $\alpha$  și  $\chi$  au variabila  $B'$  în comun. Știm acest lucru deoarece  $B$  este 0 (identic cu  $B'$ ) pentru coloana de deasupra celor două regiuni. Comparați acest lucru cu diagrama Venn de deasupra hărții Karnaugh.

În aceeași ordine de idei, putem observa că  $\beta$  și  $\delta$  au ca și variabilă comună  $B$  ( $B = 1$ ). Prin urmare,  $\alpha$  și  $\beta$  au în comun variabila booleană  $A'$  ( $A = 0$ ), iar  $\chi$  și  $\delta$  variabila  $A$  ( $A = 1$ ).

Pe scurt, am încercat să grupăm variabilele booleene pe regiuni astfel încât să reiasă elementele lor comune. Hărțile Karnaugh sunt organizate pentru a ne oferi exact această „imagine”.

### 8.5.1 Exemple de utilizare a hărților Karnaugh

### 1. Exemplul 1

Tabelul de adevăr de mai jos conține două valori de 1. Harta Karnaugh trebuie să conțină și ea tot două valori de 1. Luăm prima valoare de 1 din rândul al doilea al tabelului de adevăr: observați adresa AB a tabelului de adevăr; localizați regiunea hărții Karnaugh ce conține aceeași adresă; scrieți un 1 în acea regiune; repetați procesul pentru valoarea 1 din ultima linie a tabelului de adevăr.

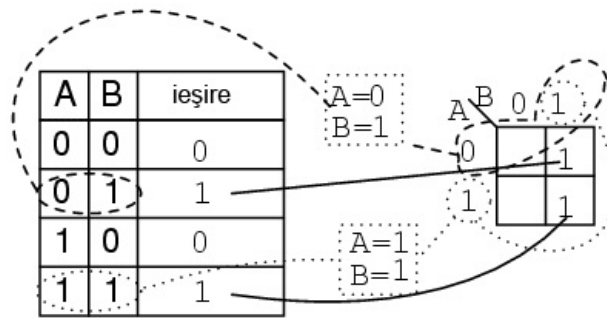


Figure 232: transformarea tabelului de adevăr în harta Karnaugh

Să încercăm să scriem acum pentru harta Karnaugh de mai sus și expresia booleană. Soluția este prezentată mai jos:

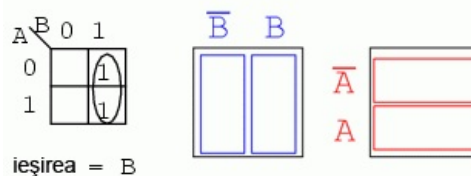


Figure 233: transformarea hărții Karnaugh în expresie booleană

Căutăm regiuni adiacente (regiunile diagonale nu sunt adiacente), întrucât acestea vor avea una sau mai multe variabile booleene în comun: grupăm cele două valori de 1 din coloană; căutăm acea sau acele variabile ce sunt comune pentru grup și scriem acest lucru ca și rezultat boolean (în cazul nostru acesta este B); ignorăm variabilele ce nu sunt identice pentru un grup de regiuni (în cazul nostru, A variază, este atât 1 cât și 0, prin urmare, ignorăm A); ignorăm de asemenea orice variabilă ce nu este asociată cu regiunile ce conțin 1 (B' nu conține niciun 1, prin urmare, ignorăm B'); rezultatul final și prin urmare expresia booleană asociată hărții Karnaugh precedente este B. Acest lucru poate fi observat mai ușor comparând diagramele Venn din dreapta, în mod special coloana B.

### 2. Exemplul 2

Scrieți expresia booleană asociată hărții Karnaugh de mai jos:

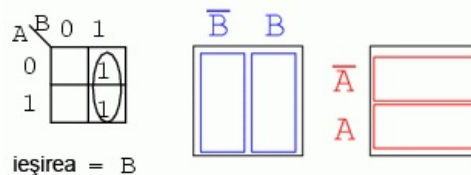


Figure 234: transformarea hărții Karnaugh în expresie booleană

Urmând o logică asemănătoare celei de mai sus, grupăm toate valorile de 1 și găsim variabila comună întregului grup astfel format; rezultatul este A'.

### 3. Exemplul 3

Pentru tabelul de adevăr de mai jos, găsiți harta Karnaugh corespunzătoare și scrieți apoi expresia booleană folosind rezultatul obținut:

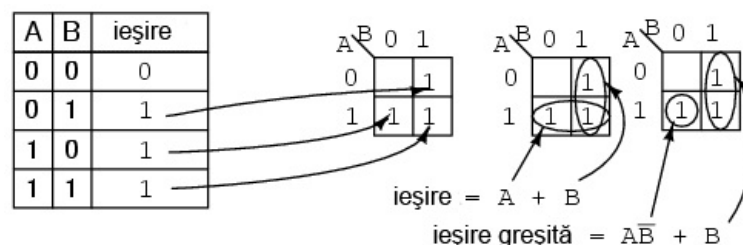


Figure 235: transformarea tabelului de adevăr în harta Karnaugh

Soluție: transferăm valorile de 1 din tabelul de adevăr în locațiile corespunzătoare pe harta Karnaugh; grupăm cele două valori de 1 pe coloana de sub B = 1; grupăm cele două valori de 1 de pe rândul A = 1; scriem rezultatul produsului primului grup (B); scriem rezultatul produsului celui de al doilea grup (A); scriem suma produselor celor doi termeni de mai sus (A + B). Soluția din mijloc este cea mai simplă și prezintă cel mai mic cost. O soluție mai puțin dorită este cea din dreapta. După gruparea valorilor 1, facem greșeala de a forma un grup cu o singură regiune. Motivul pentru care acest lucru nu este dorit este următorul: acest grup ce conține o singură regiune are termenul produsului egal cu AB'; soluția întregii hărți este în acest caz AB' + B, iar aceasta nu reprezintă cea mai simplă soluție.

Metoda corectă constă în gruparea acestui 1 singur cu regiunea din dreapta lui, regiune ce conține la rândul ei o valoare de 1, chiar dacă aceasta a fost deja inclusă într-un alt grup. (coloana B). Putem refolosi regiuni pentru a forma grupuri mai mari. De fapt, este chiar indicat să facem acest lucru întrucât conduce la rezultate mai simple.

Trebuie să facem observația că oricare dintre soluțiile de mai sus, atât cea corectă cât și cea „greșită” sunt de fapt corecte din punct de vedere logic. Ambele circuite vor genera aceeași ieșire. Pur și simplu, circuitul corect presupune un cost mai redus de

implementare fizică.

#### 4. Exemplul 4

Completați o hartă Karnaugh folosind expresia booleană de mai jos. Scrieți apoi expresia booleană a rezultatului:

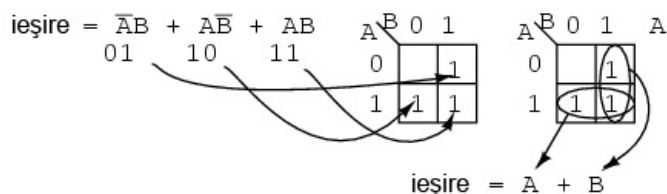


Figure 236: transformarea expresiei booleene în harta Karnaugh

Expresia booleană conține trei sume de produse. Va exista câte o valoare de 1 pe harta Karnaugh pentru fiecare produs. Deși, în general, numărul valorilor de 1 pe produs variază cu numărul variabilelor produsului în comparație cu mărimea hărții Karnaugh. Termenul produsului reprezintă adresa regiunii unde vom introduce valoare de 1. Primul termen este  $A'B$  și corespunde adresei 01 a hărții. Introducem un 1 în această regiune. Similar, introducem și ceilalți doi termeni de 1.

Trecem apoi la gruparea termenilor și simplificarea rezultatului conform exemplului precedent.

#### 5. Exemplul 5

Simplificați circuitul logic de mai jos:

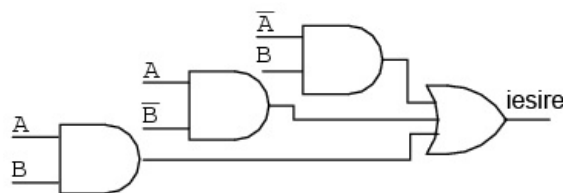


Figure 237: circuit logic

Scriem expresia booleană pentru circuitul logic inițial; transferăm expresia booleană rezultată într-o hartă Karnaugh; grupăm regiunile precum în exemplele precedente; scriem expresii booleene pentru fiecare grup, conform exemplurilor precedente; redesenăm circuitul logic simplificat:

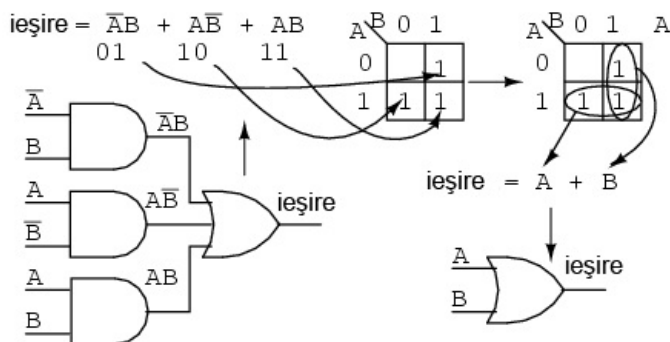


Figure 238: simplificarea unui circuit logic

#### 6. Exemplul 6

Simplificați circuitul logic de mai jos:

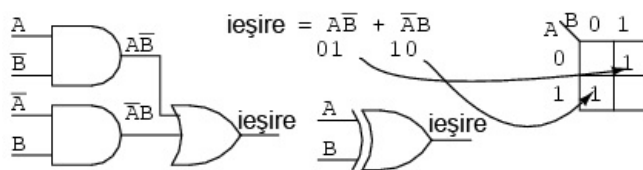


Figure 239: circuit logic

Scriem expresia booleană pentru circuitul logic inițial; completăm harta Karnaugh; observăm că nu putem forma niciun grup care să conțină mai mult de două regiuni 1; prin urmare, simplificarea nu este posibilă, iar expresia finală este identică cu cea inițială (SAU-exclusiv).

## 8.6 Simplificarea circuitelor logice cu hărți Karnaugh

Exemplele de simplificare a circuitelor logice de până acum puteau fi realizate la fel de bine și cu ajutorul algebrei booleene. Problemele de simplificare logică reale implică însă utilizarea unor hărți Karnaugh mai mari. În această secțiune vom concepe câteva exemple imaginare, lăsând aplicațiile practice pentru capitolul de logică combinațională. Aceste exemple sunt concepute doar pentru a ilustra tehnicile de simplificare.

Vom folosi harta Karnaugh dezvoltată anterior, mai exact forma din dreapta:



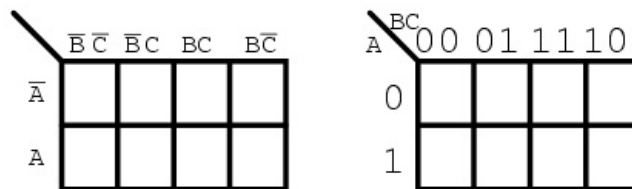


Figure 240: hartă Karnaugh

### 8.6.1 Codul Gray

Observați secvența numerelor din partea superioară a hărții. Aceasta nu este o secvență binară (00, 01, 10, 11), ci este o secvență de tipul 00, 01, 11, 10. Această secvență este cunoscută sub numele de cod Gray. Secvența de tip cod Gray modifică doar un singur bit pe măsură ce trecem de la un număr la următorul număr din secvență. Acest lucru nu este valabil într-o secvență binară. Regiunile adiacente diferă doar printr-un singur bit, sau variabilă booleană. Acest lucru este necesar dacă dorim organizarea ieșirilor unei funcții logice pentru observarea elementelor lor comune.

Mai mult, antetul coloanelor și rândurilor trebuie să fie în ordinea codului Gray, altfel, harta nu se va comporta precum o hartă Karnaugh. Regiunile ce au în comun variabile booleene nu vor mai fi adiacente și nu vom mai putea identifica caracteristicile specifice funcției pe cale vizuală. Regiunile adiacente variază cu un singur bit, deoarece secvența de cod Gray variază la rândul ei doar cu un singur bit.

### 8.6.2 Hărți Karnaugh cu 3 variabile - exemple de simplificare

Să folosim în continuare hărțile Karnaugh cu 3 variabile pentru simplificarea unor expresii booleene. Vom arăta cum să trecem termenii produs ai ecuației nesimplificate în harta Karnaugh. Vom ilustra și modul de identificare a grupurilor de regiuni adiacente ce duc la formarea sumei de produse simplificate a circuitului logic (expresiei booleene).

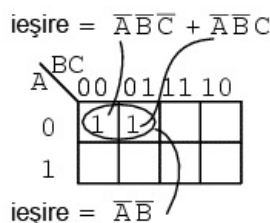


Figure 241: simplificarea unei expresii booleene prin intermediul hărții Karnaugh

Dându-se expresia  $(A'B'C' + A'B'C)$ , primul pas este introducerea valorilor de 1 pe harta Karnaugh corespunzător poziției fiecărui produs al sumei ( $A'B'C'$  este echivalent cu 000, iar  $A'B'C$  este echivalent cu 001). Identificăm apoi un grup de regiuni alăturate ce conțin valori de 1 (în cazul de față, avem doar două astfel de regiuni). Scriem apoi produsul de termeni pentru acest grup, ceea ce reprezintă rezultatul simplificat.

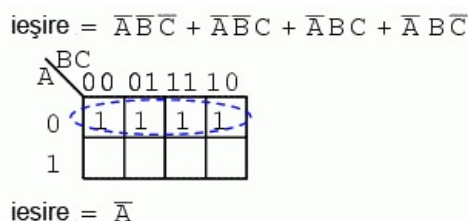


Figure 242: simplificarea unei expresii booleene prin intermediul hărții Karnaugh

Grupând cei patru termeni de 1 pe harta Karnaugh, rezultatul este asigurat de expresia  $A'$ .

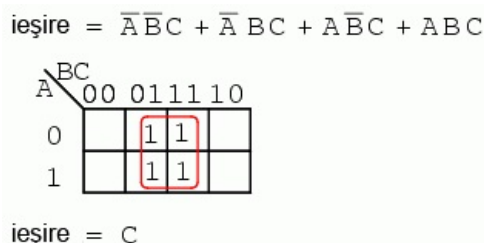


Figure 243: simplificarea unei expresii booleene prin intermediul hărții Karnaugh

Identic, grupând ce patru termeni de 1, putem foarte ușor observa că singura variabilă ce acoperă toate cele patru regiuni este  $C$ .

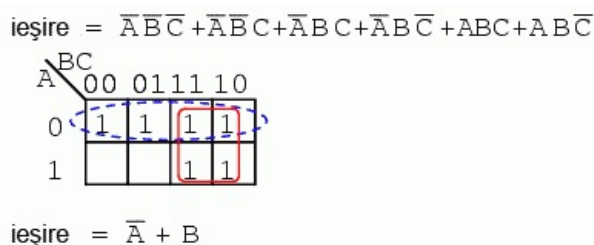


Figure 244: simplificarea unei expresii booleene prin intermediul hărții Karnaugh

Din moment ce avem două grupuri pe harta Karnaugh de mai sus, rezultatul va fi o sumă de produse, și anume,  $A' + B$ .

$$\text{ieșire} = \overline{A}BC + ABC$$

A \ BC	00 01 11 10			
	0	1	1	0
0			1	
1			1	

$$\text{ieșire} = BC$$

Figure 245: simplificarea unei expresii booleene prin intermediul hărții Karnaugh

Cele două produse de mai sus formează un grup de doi termeni ce se simplifică la  $BC$ .

$$\text{ieșire} = \overline{A}BC + \overline{A}B\overline{C} + ABC + AB\overline{C}$$

A \ BC	00 01 11 10			
	0	1	1	0
0			1	1
1			1	1

$$\text{ieșire} = B$$

Figure 246: simplificarea unei expresii booleene prin intermediul hărții Karnaugh

Variabila comună celor patru termeni grupați mai sus este  $B$

$$\text{ieșire} = \overline{A}\overline{B}\overline{C} + \overline{A}B\overline{C} + \overline{A}B\overline{C} + AB\overline{C}$$

A \ BC	00 01 11 10			
	0	1	1	0
0	1			1
1	1			1

$$\text{ieșire} = \overline{C}$$

Figure 247: simplificarea unei expresii booleene prin intermediul hărții Karnaugh

Cei patru termeni de mai sus formează un singur grup. Putem vizualiza acest grup dacă îndoim extremitățile hărții pentru a forma un cilindru. În acest caz, regiunile sunt adiacente. În mod normal, un astfel de grup se notează conform figurii din stânga. Din întregul set de variabile ( $A$ ,  $B$ ,  $C$ ), singura variabilă comună este  $C'$ .  $C'$  este zero în toate cele patru regiuni. Acesta este atunci rezultatul final al simplificării.

$$\text{ieșire} = \overline{A}\overline{B}\overline{C} + \overline{A}B\overline{C} + \overline{A}B\overline{C} + \overline{A}B\overline{C} + A\overline{B}\overline{C} + AB\overline{C}$$

A \ BC	00 01 11 10			
	0	1	1	0
0	1	1	1	1
1	1			1

$$\text{ieșire} = \overline{A} + \overline{C}$$

Figure 248: simplificarea unei expresii booleene prin intermediul hărții Karnaugh

Cele șase regiuni rezultate din ecuația nesimplificată pot fi organizate în două grupuri de câte patru. Aceste grupuri trebuie să rezulte într-o sumă de două produse, și anume  $A' + C'$ .

### 8.6.3 Incinerator deșeuri toxice - reconsiderare

Să reluăm mai jos exemplul incineratorului de deșeuri toxice studiat într-un capitol precedent. Vom încerca simplificarea circuitului logic folosind o hartă Karnaugh:

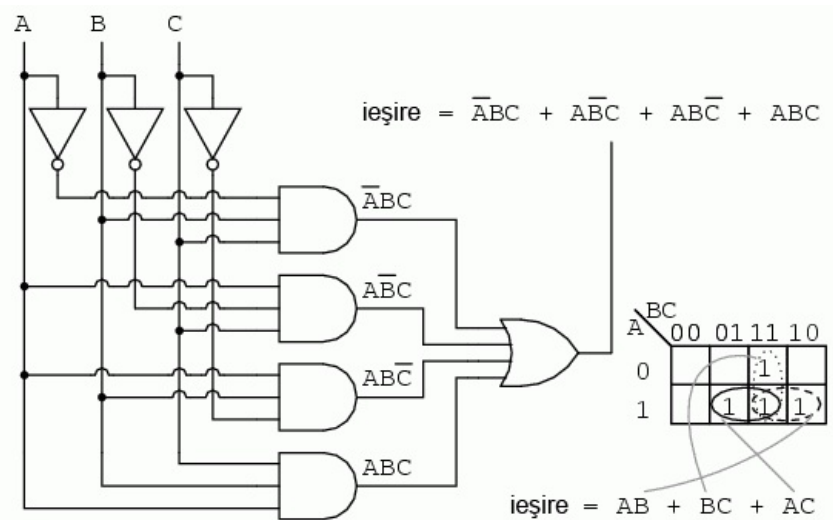


Figure 249: incinerator deşeurii toxice - simplificarea circuitului logic folosind hărţi Karnaugh

Ecuatia booleană de iesire este o sumă de patru produse. Prin urmare, vom avea patru regiuni de 1 pe harta Karnaugh. Grupând regiunile adiacente, avem trei grupuri de câte doi termeni. Vom avea prin urmare o sumă de trei produse, fiecare produs conţinând doi termeni. Circuitul logic simplificat, identic cu cel obţinut cu ajutorul regulilor de simlificare booleană, este redat mai jos:

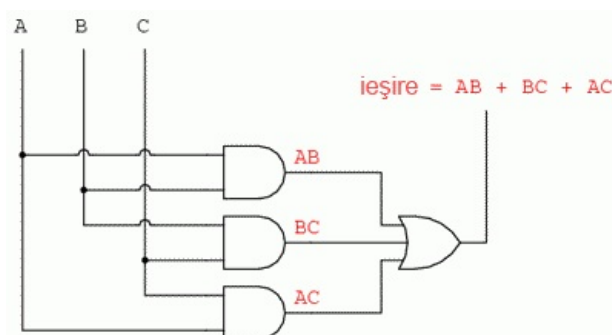


Figure 250: incinerator deşeurii toxice - circuitul logic simplificat

Făcând o comparaţie între regulile boolene folosite pentru simplificarea circuitului logic al incineratorului...

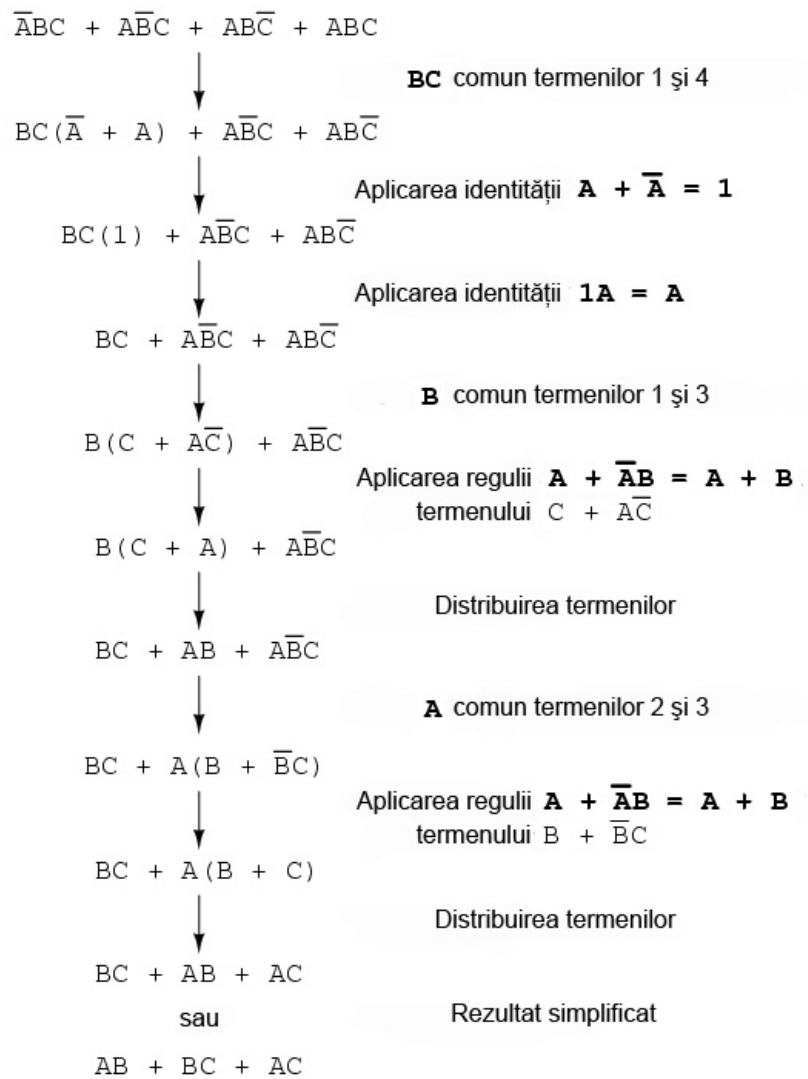


Figure 251: incinerator deșeuri toxice - simplificarea booleană

...și harta Karnaugh, care duce la exact același rezultat...



Figure 252: incinerator deșeuri toxice - simplificarea circuitului logic folosind hărți Karnaugh

Putem lesne vedea motivul pentru care hărțile Karnaugh sunt preferate pentru simplificarea circuitelor logice în detrimentul simplificării booleene.

## 8.7 Hărți Karnaugh de patru variabile

Folosindu-ne de codul Gray, putem construi hărți Karnaugh mai mari. O hartă Karnaugh cu patru variabile arată precum cea de mai jos:

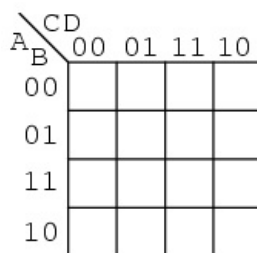
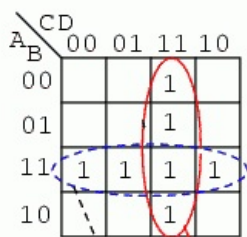


Figure 253: hartă Karnaugh cu patru variabile

Exemplele de mai jos ilustrează simplificarea expresiilor booleene ce sunt prea greu de realizat prin intermediul regulilor de simplificare booleană. Aceste expresii pot fi simplificate cu algebra booleană. Totuși, utilizarea hărților Karnaugh este un procedeu mult mai rapid și mai ușor, mai ales dacă există multe simplificări logice de realizat.

### 8.7.1 Exemple de simplificare logică cu hărți Karnaugh de patru variabile

$$\text{ieşire} = \overline{A}\overline{B}CD + \overline{A}B\overline{C}D + A\overline{B}CD + A\overline{B}\overline{C}D + A\overline{B}C\overline{D} + A\overline{B}CD + A\overline{B}C\overline{D}$$

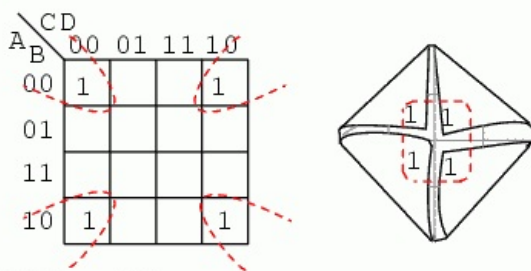


$$\text{ieşire} = \overline{A}B + CD$$

Figure 254: simplificarea expresiilor booleene folosind hărți Karnaugh de patru variabile

Expresia booleană de mai sus conține 7 produse. Acești termeni sunt grupați de sus în jos și de la stânga la dreapta pe harta Karnaugh de mai sus. De exemplu, primul termen,  $A'B'CD$ , se regăsește pe rândul 1, căsuța a 3-a, și corespunde locației  $A = 0$ ,  $B = 0$ ,  $C = 1$ ,  $D = 1$ . Ceilalți termeni sunt poziționați într-o manieră similară. Grupul orizontal (albastru) corespunde termenului  $AB$ , iar grupul vertical (roșu) corespunde expresiei booleene  $CD$ . Din moment ce avem două grupuri, rezultatul trebuie să fie o sumă de două produse, prin urmare,  $AB + CD$ .

$$\text{ieşire} = \overline{A}\overline{B}\overline{C}\overline{D} + \overline{A}\overline{B}C\overline{D} + \overline{A}B\overline{C}\overline{D} + \overline{A}B\overline{C}D$$

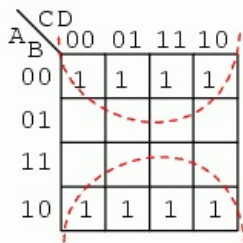


$$\text{ieşire} = \overline{B}\overline{D}$$

Figure 255: simplificarea expresiilor booleene folosind hărți Karnaugh de patru variabile

În cazul de mai sus, „împăturim” cele patru colțuri ale hărții Karnaugh, precum un șervețel, pentru a observa mai bine adiacența celor patru regiuni.  $B = 0$  și  $D = 0$  pentru toate regiunile. Celelalte variabile,  $A$  și  $C$ , sunt 0 în unele cazuri și 1 în altele. Prin urmare, aceste variabile nu se vor regăsi în rezultatul final al expresiei simplificate.

$$\text{ieşire} = \overline{A}\overline{B}\overline{C}\overline{D} + \overline{A}\overline{B}C\overline{D} + \overline{A}B\overline{C}\overline{D} + \overline{A}B\overline{C}D + A\overline{B}\overline{C}\overline{D} + A\overline{B}C\overline{D} + A\overline{B}\overline{C}D + A\overline{B}CD$$



$$\text{ieşire} = \overline{B}$$

Figure 256: simplificarea expresiilor booleene folosind hărți Karnaugh de patru variabile

Pentru o vizualizare mai bună, ne putem imagina că îndoim marginile de jos și de sus a hărții sub forma unui cilindru. În acest caz, ambele grupuri sunt adiacente și formează practic un singur grup. Acest lucru ne spune că rezultatul este un singur termen. Singura variabilă comună a acestui grup de 8 variabile este  $B = 0$ . Rezultatul simplificării este prin urmare  $B'$ .

$$\text{ieşire} = \overline{A}\overline{B}\overline{C}\overline{D} + \overline{A}\overline{B}C\overline{D} + \overline{A}B\overline{C}\overline{D} + \overline{A}B\overline{C}D + B\overline{C}\overline{D} + B\overline{C}D + A\overline{B}\overline{C}\overline{D} + A\overline{B}C\overline{D} + A\overline{B}\overline{C}D + A\overline{B}CD$$

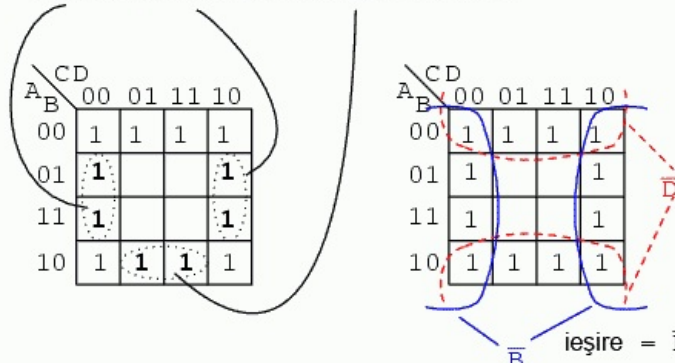


Figure 257: simplificarea expresiilor booleene folosind hărți Karnaugh de patru variabile

Expresia booleană de mai sus conține 9 termeni de produse, dintre care trei au doar trei variabile booleene în loc de patru.

Diferența constă în faptul că, deși termenii ce conțin patru variabile booleene acoperă o singură regiune, termenii cu trei variabile booleene acoperă o pereche de regiuni fiecare.

Trecând la simplificare, formăm două grupuri de câte opt termeni. Regiunile ce se regăsesc în colț sunt comune ambelor grupuri. Acest lucru este corect. De fapt, această strategie conduce la o soluție mai bună decât dacă am fi format un grup de opt și un grup de patru regiuni, fără nicio regiune comună celor două. Soluția finală este  $B' + D'$ .

$$\text{ieșire} = \overline{A}\overline{B}\overline{C}\overline{D} + \overline{A}\overline{B}C\overline{D} + \overline{A}B\overline{C}\overline{D} + ABCD$$

A \ B \ C \ D	00	01	11	10
00	1			1
01				
11			1	
10	1			

$$\text{ieșire} = \overline{B}\overline{C}\overline{D} + \overline{A}\overline{B}\overline{D} + ABCD$$

Figure 258: simplificarea expresiilor booleene folosind hărți Karnaugh de patru variabile

În exemplul de mai sus, trei regiuni formează două grupuri de câte două. O a patra regiune nu poate fi combinată cu nicio altă regiune, ceea ce se întâmplă frecvențe în situațiile reale. În acest caz, termenul  $ABCD$  rămâne neschimbat în cadrul procesului de simplificare a expresiei booleene inițiale. Rezultatul este  $B'C'D' + A'B'D' + ABCD$ .

Adeseori, există mai mult de o singură soluție cu cost minim pentru expresia nesimplificată. Un astfel de caz este cel de mai jos:

$$\begin{aligned} \text{ieșire} = & \overline{A}\overline{B}\overline{C}\overline{D} + \overline{A}\overline{B}C\overline{D} + \overline{A}B\overline{C}\overline{D} + \overline{A}BCD \\ & + ABCD + AB\overline{C}\overline{D} + A\overline{B}C\overline{D} + A\overline{B}C\overline{D} \end{aligned}$$

A \ B \ C \ D	00	01	11	10
00	1	1		
01		1	1	
11			1	1
10	1			1

A \ B \ C \ D	00	01	11	10
00	1	1		
01		1	1	
11			1	1
10	1			1

$$\text{ieșire} = \overline{B}\overline{C}\overline{D} + \overline{A}\overline{C}\overline{D} + BCD + AC\overline{D}$$

$$\text{ieșire} = \overline{A}\overline{B}\overline{C} + \overline{A}\overline{B}\overline{D} + ABC + A\overline{B}\overline{D}$$

Figure 259: simplificarea expresiilor booleene folosind hărți Karnaugh de patru variabile

Ambele rezultate de mai sus conțin patru termeni, cu trei variabile booleene fiecare. Ambele soluții sunt valide din punct de vedere al minimizării costurilor. Diferența dintre cele două soluții finale constă în modul de grupare al regiunilor. Reamintim faptul că o soluție cu cost minim este acea soluție ce permite o implementare fizică a circuitului logic cu un număr cât mai mic de porți logice și număr de intrări.

$$\begin{aligned} \text{ieșire} = & \overline{A}\overline{B}\overline{C}\overline{D} + \overline{A}\overline{B}C\overline{D} + \overline{A}B\overline{C}\overline{D} \\ & + \overline{A}B\overline{C}\overline{D} + \overline{A}B\overline{C}\overline{D} + \overline{A}B\overline{C}\overline{D} \\ & + AB\overline{C}\overline{D} + AB\overline{C}\overline{D} + ABCD \end{aligned}$$

A \ B \ C \ D	00	01	11	10
00	1	1	1	
01	1	1	1	
11	1	1	1	
10				

A \ B \ C \ D	00	01	11	10
00	1	1	1	
01	1	1	1	
11	1	1	1	
10				

A \ B \ C \ D	00	01	11	10
00	1	1	1	
01	1	1	1	
11	1	1	1	
10				

$$\text{ieșire} = \overline{A}\overline{C} + \overline{A}\overline{D} + B\overline{C} + BD$$

Figure 260: simplificarea expresiilor booleene folosind hărți Karnaugh de patru variabile

În următorul exemplu, cel de mai sus, după ce trecem toate valorile de 1 pe hartă Karnaugh, realizăm primul pas al simplificării, și anume, gruparea primelor patru regiuni (stânga). În acest punct, s-ar putea să nu fie foarte evident cum am putea grupa regiunile rămase.

La pasul al doilea (centru), grupăm încă patru regiuni. Mai rămân în acest moment încă două regiuni negrupate. Soluția cu cost minim este să grupăm aceste două regiuni, ca și grupuri de patru, conform figurii din dreapta.

Atenție, nu încercați să realizați grupuri de câte trei. Grupările trebuie să fie sub forma puterilor lui 2, și anume, 1, 2, 4, 8, etc.



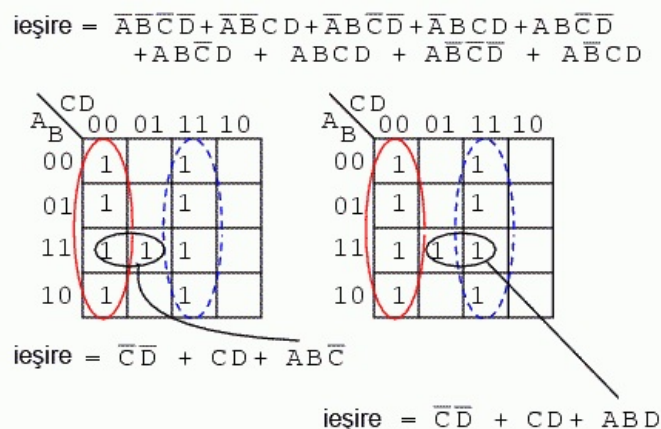


Figure 261: simplificarea expresiilor booleene folosind hărți Karnaugh de patru variabile

Avem din nou mai sus un exemplu ce suportă două soluții cu cost minim. Formăm inițial cele două grupuri de câte patru regiuni (roșu și albastru). Soluția finală depinde de modul în care grupăm regiunea rămasă liberă. Dacă o introducem în grupul din stânga (roșu), soluția este  $ABC'$ . Dacă o introducem în grupul din dreapta (albastru), soluția este  $ABD$ . Indiferent de alegerea făcută, ambele soluții sunt corecte din punct de vedere al minimizării costurilor de implementare.

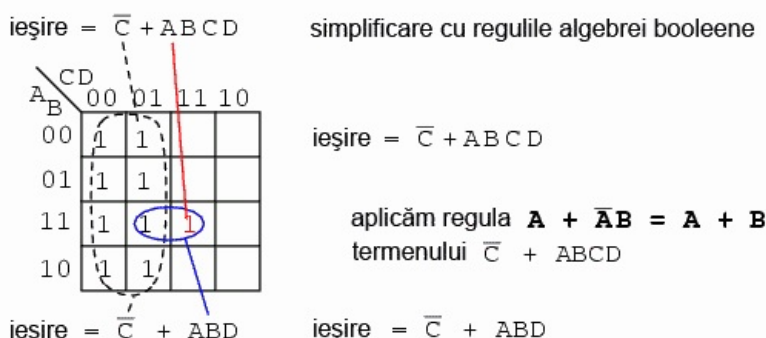


Figure 262: simplificarea expresiilor booleene folosind hărți Karnaugh de patru variabile

Mai sus este un exemplu de simplificare cu hărți Karnaugh (stânga) precum și cu regulile algebrei booleene (dreapta).  $C'$  ( $C = 0$ ) reprezintă aria formată de cele opt regiuni din stânga. Regiunea rămasă negrupată este echivalentă cu expresia  $ABCD$ . Grupând această regiune cu cea din stânga ei, simplifică termenul  $ABCD$  la  $ABD$ . Rezultatul final este prin urmare  $C' + ABD$ . Cazul de mai sus este un exemplu rar a unei probleme cu patru variabile ce poate fi redusă destul de ușor și cu algebra booleană. Asta în cazul în care vă amintiți teoremele de simplificare booleană.

## 8.8 Mintermeni și maxtermeni

Până în acest moment am căutat soluții sub forma unei sume de produse la problemele de simplificare booleană. Pentru fiecare dintre aceste soluții există o altă soluție sub forma unui produs de sume. Acest tip de soluție se poate dovedi a fi mai practică, în funcție de aplicație. Dar, înainte de a scrie soluțiile sub forma unui produs de sume, trebuie să introducem câteva concepte noi. Procedura de mai jos pentru extragerea termenilor sub formă de produs nu este nouă. Vrem doar să stabilim o procedură formală pentru mintermeni, ca mai apoi, să putem face o comparație cu noua procedură pentru maxtermeni.

### 8.8.1 Analiza regiunilor ce conțin valori de 1 - mintermeni

Un mintermen este o expresie booleană rezultând într-o valoare de 1 pentru ieșirea unei singure regiuni dintr-o hartă Karnaugh. Toate celelalte regiuni ale hărții Karnaugh sau ale tabelului de adevăr fiind 0 în acest caz. Dacă un mintermen conține un singur 1, iar regiunile rămase sunt toate 0, aria minimă pe care acest mintermen o acoperă este 1. Figura de mai jos (stânga) prezintă mintermenul  $ABC$ , un singur termen sub formă de produs, ca și o singură valoare de 1 pe o hartă Karnaugh unde toate celelalte regiuni sunt 0. Până în acest moment, nu am prezentat valorile de 0 pe hărțile Karnaugh considerate. Acestea se omit de obicei, excepție făcând cazurile speciale. Un alt mintermen,  $A'BC'$  este cel din dreapta. Ceea ce vrem să subliniem este faptul că adresa regiunii corespunde direct cu mintermenul extras de pe hartă. Regiunea 111 corespunde mintermenului  $ABC$  din stânga. Regiunea 010 corespunde la rândul ei mintermenului  $A'BC'$ . O expresie booleană sau o hartă poate avea mai mulți mintermeni.

Referindu-ne la figura de mai sus, putem scrie procedura introducerii unui mintermen pe harta Karnaugh:

- Identificăm mintermenul (produsul) ce vrem să-l introducem pe hartă
- Scriem valoarea numerică corespunzătoare
- Ne folosim de valoarea binară ca și adresă pe hartă
- Introducem un 1 la adresa respectivă
- Repetăm pașii de mai sus pentru un nou mintermen (termenii produs dintr-o sumă de produse)

O expresie booleană este formată de cele mai multe ori din mai mulți mintermeni, corespunzând mai multor regiuni pe o hartă Karnaugh, precum în exemplul de mai jos:

$$\begin{aligned} \text{ieșire} &= \bar{A} \bar{B} \bar{C} + ABC \\ \begin{array}{c|ccc} & \text{BC} & 00 & 01 & 11 & 10 \\ \hline \text{A} & & & & & \\ 0 & 0 & 0 & 0 & 1 & \\ 1 & 0 & 0 & 1 & 0 & \end{array} \\ \text{numeric} &= 010 \quad 111 \\ \text{mintermen} &= \bar{A} \bar{B} \bar{C} \quad ABC \\ \text{ieșire} &= \bar{A} \bar{B} \bar{C} + ABC \end{aligned}$$

Figure 263: simplificarea expresiei booleene prin intermediul hărții Karnaugh

Mintermenii multipli de pe această hartă sunt mintermenii individuali ce i-am analizat mai sus. Ceea ce vrem să reamintim este faptul că valorile de 1 sunt „traduse” de pe harta Karnaugh ca și o adresă binară transformată direct într-unul sau mai mulți termeni sub formă de produs. Prin direct, ne referim la faptul că 0 corespunde unei variabile negată, iar 1 corespunde unei variabile „pure”. De exemplu, 010 se transformă direct în  $A'BC'$ . În acest exemplu nu a existat nicio simplificare. Totuși, avem ca și rezultat o sumă de produse prin intermediul mintermenilor.

Referindu-ne la figura de mai sus, putem rezuma pe scurt procedura de urmat în cazul simplificării expresiei booleene sub forma unei sume de produse dintr-o hartă Karnaugh:

- Formăm grupuri de 1 cât mai mari posibile, acoperind toți mintermenii de pe hartă. Grupurile trebuie să conțină un număr de regiuni sub forma puterii lui 2 (1, 2, 4, 8, etc.)
- Scriem valori numerice binare pentru fiecare grup
- Transformăm valoarea binară sub forma unui produs
- Repetăm pașii de mai sus pentru toate grupurile formate. Din fiecare grup va rezulta un termen sub formă de produs
- Expresia simplificată reprezintă suma acestor termeni sub formă de produs

Nimic nou până în acest moment. Am scris doar pașii de urmat în cazul mintermenilor. Același lucru îl vom face și în cazul maxtermenilor.

### 8.8.2 Analiza regiunilor ce conțin valori de 0 - maxtermeni

Să considerăm acum o funcție booleană ce este 0 pentru o singură regiune și 1 în rest:

$$\begin{aligned} \text{ieșire} &= (A + B + C) \\ \text{maxtermen} &= A + B + C \\ \text{numeric} &= 1 \quad 1 \quad 1 \\ \text{complement} &= 0 \quad 0 \quad 0 \end{aligned}$$

		BC			
A		00	01	11	10
0		0	1	1	1
1		1	1	1	1

Figure 264: simplificarea expresiei booleene prin intermediul hărții Karnaugh

Un maxtermen este o expresie booleană a cărei valori este 0 pentru o singură regiune, toate celelalte regiunii ale hărții Karnaugh sau ale tabelului de adevăr fiind 0. Vedeti și explicația de la mintermen. Figura de sus stânga prezintă un maxtermen  $(A + B + C)$ , o sumă de trei termeni simplii. Pe hartă, această sumă este reprezentată printr-un singur 0, toate celelalte regiunii ale hărții fiind 1.

Dacă un maxtermen are un singur 0, iar celelalte regiuni sunt 1, aria maximă pe care o acoperă este 1. Există câteva diferențe acum că am introdus și maxtermenii. Maxtermenul este un 0, nu un 1 pe harta Karnaugh. Un maxterm este un termen sub formă de sumă,  $A + B + C$  în cazul nostru, și nu termen sub formă de produs ( $ABC$ , de exemplu).

Pare ciudat că locația expresiei (termenului)  $(A + B + C)$  pe hartă este

1. Pentru ecuația „ieșire =  $(A + B + C) = 0$ ”, toate cele trei variabile ( $A, B, C$ ) trebuie să fie egale cu 0. Doar expresia  $(0 + 0 + 0) = 0$  va fi egală cu 0. Prin urmare, trecem singurul nostru maxtermen  $(A + B + C)$  în regiunea ce se află la adresa  $A,B,C = 000$  pe harta Karnaugh, unde toate intrările sunt egale cu 0. Aceasta este singura posibilitate pentru a obține valoarea de 0 pentru maxtermen. Toate celelalte regiuni conțin valori de 1 pentru că orice alte valori de intrare diferite de  $(0, 0, 0)$ ,

1. pentru expresia  $(A + B + C)$  au ca și rezultat 1.

Luând în considerare figura de mai sus, pașii care trebuiesc urmați pentru introducerea unui maxtermen pe harta Karnaugh, sunt următorii:

- Identificăm termenul sub formă de sumă (maxtermenul) ce-l vom introduce pe hartă
- Scriem valoarea numerică binară corespunzătoare
- Formăm complementul
- Utilizăm complementul ca și adresă pentru introducerea valorii de 0 pe harta Karnaugh
- Repetăm pașii de mai sus pentru toți ceilalți maxtermeni (termeni-sumă dintr-o expresie sub forma de produs de sume)

Un alt maxtermen este prezentat în figura de mai jos. Valoarea numerică 000 corespunde termenului  $A' + B' + C'$ . Complementul este 111. Introducem o valoare de 0 pentru maxtermenul  $(A' + B' + C')$  la această adresă  $(1, 1, 1)$  a hărții Karnaugh de mai jos:



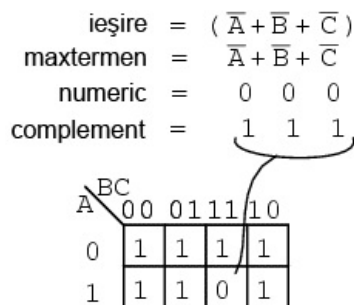


Figure 265: simplificarea expresiei booleene prin intermediul hărții Karnaugh

### 8.8.3 Scrierea expresiei booleene simplificate ca și produs de sume

O expresie booleană sub formă produsului de sume poate avea mai mulți maxtermeni, conform figurii de mai jos:

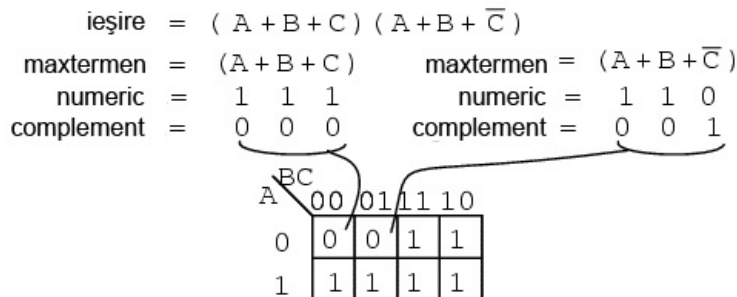


Figure 266: simplificarea expresiei booleene prin intermediul hărții Karnaugh

Maxtermenul  $(A + B + C)$  sub formă numerică este 111, iar complementat este 000. Plasăm prin urmare un 0 la adresa (0, 0, 0). Maxtermenul  $(A + B + \overline{C})$  sub formă numerică este 110, iar complementat este 001. Plasăm prin urmare un zero la adresa (0, 0, 1). Acum că am construit harta Karnaugh, suntem interesați de modul în care putem scrie o formă simplificată a expresiei booleene inițiale sub formă de produs de sume. Primul pas este gruparea termenilor de 0, precum grupul de mai jos:

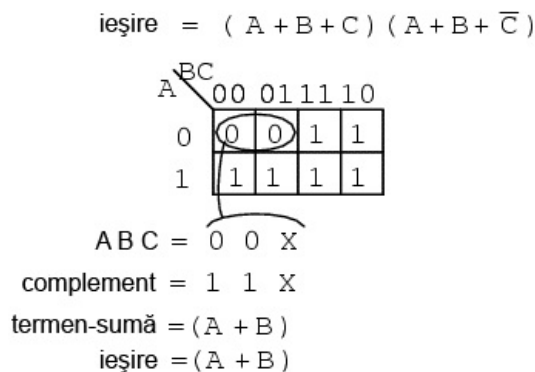


Figure 267: simplificarea expresiei booleene prin intermediul hărții Karnaugh

Scriem apoi valoarea binară corespunzătoare termenului-sumă, ce arată astfel: (0, 0, X). Pentru grupul format, atât A cât și B sunt 0. Dar C este atât 0 cât și 1. Prin urmare, scriem un X în locul valorii lui C. Formăm complementul: (1, 1, X). Scriem termenul sumă  $(A + B)$  ignorând C-ul și X-ul ce l-a înlocuit.

Să reluăm pașii necesari pentru reducerea unei expresii booleene la un produs de sume:

- Formăm grupuri de 0 cât mai mari posibile, incluzând toți maxtermenii. Numărul termenilor trebuie să fie puteri ale lui 2
- Scriem valoarea numerică a grupului
- Complementăm această valoare numerică a grupului
- Transformăm valoarea complementată într-un termen sub formă de sumă
- Repetăm pașii de mai sus pentru toate grupurile rămase pe hartă. Rezultatul fiecărui grup este un termen sub formă de sumă, iar rezultatul final este produsul acestor termeni-sumă

#### 1. Exemplul 1

Simplificați expresia booleană sub forma produsului de sume de mai jos. Scrieți rezultatul final sub forma unui produs de sume:

$$\begin{aligned} \text{ieșire} &= (A + B + C + \overline{D}) (A + B + \overline{C} + D) (A + \overline{B} + C + \overline{D}) (A + \overline{B} + \overline{C} + D) \\ &\quad (\overline{A} + \overline{B} + \overline{C} + D) (\overline{A} + B + C + \overline{D}) (\overline{A} + B + \overline{C} + D) \end{aligned}$$

Figure 268: expresie booleană sub formă de produs de sume

Soluție: completăm o hartă Karnaugh cu cei șapte maxtermeni de mai sus (introducem valori de 0). Rețineți să complementați variabile de intrare pentru găsirea adresei corespunzătoare:

$$\text{ieșire} = (A+B+C+\overline{D})(A+B+\overline{C}+D)(A+\overline{B}+C+\overline{D})(A+\overline{B}+\overline{C}+D)(\overline{A}+\overline{B}+\overline{C}+D)(\overline{A}+B+C+\overline{D})(\overline{A}+B+\overline{C}+D)(\overline{A}+B+C+D)$$

CD \ AB	00	01	11	10
00		0		0
01		0		0
11				0
10		0		0

Figure 269: simplificarea expresiei booleene prin intermediul hărții Karnaugh

După ce am introdus toți maxtermenii în tabel, trecem la gruparea regiunilor, precum în figura de mai jos. Grupurile mai mari se traduc printr-un termen-sumă cu mai puține intrări. Cu cât avem mai puține grupuri, cu atât vom avea mai puțin termeni-sumă în expresia finală:

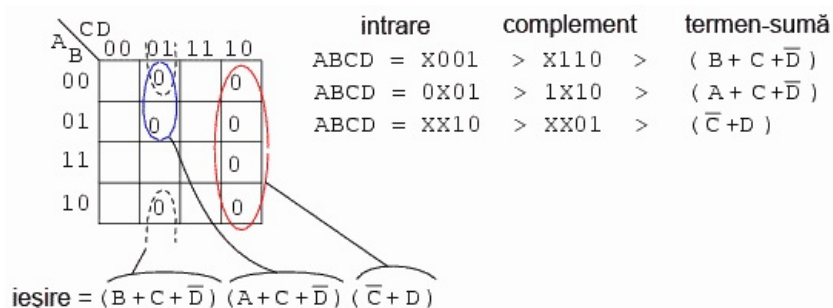


Figure 270: simplificarea expresiei booleene prin intermediul hărții Karnaugh

Avem trei grupuri, prin urmare, trebuie să avem trei termeni-sumă în rezultat final. Detaliile simplificării sunt prezentate în figura de mai sus. Pentru oricare grup, scriem mai întâi adresa de intrare, o complementăm și o transformăm într-un termen boolean sub formă de sumă. Rezultatul final este produsul acestor trei termeni-sumă.

## 2. Exemplul 2

Simplificați expresia booleană sub formă de produs de sume de mai jos, exprimând rezultatul sub forma unei sume de produse:

$$\text{ieșire} = (A+B+C+\overline{D})(A+B+\overline{C}+D)(A+\overline{B}+C+\overline{D})(A+\overline{B}+\overline{C}+D)(\overline{A}+\overline{B}+\overline{C}+D)(\overline{A}+B+C+\overline{D})(\overline{A}+B+\overline{C}+D)(\overline{A}+B+C+D)$$

Figure 271: expresie booleană sub formă de produs de sume

Această problemă este identică cu cea anterioară, cu diferența că expresia simplificată se cere sub formă de sumă de produse și nu sub formă de produs de sume.

Trecem maxtermenii (0) din expresia inițială pe harta Karnaugh de mai jos (stânga), exact ca în exemplul precedent:

$$\text{ieșire} = (A+B+C+\overline{D})(A+B+\overline{C}+D)(A+\overline{B}+C+\overline{D})(A+\overline{B}+\overline{C}+D)(\overline{A}+\overline{B}+\overline{C}+D)(\overline{A}+B+C+\overline{D})(\overline{A}+B+\overline{C}+D)(\overline{A}+B+C+D)$$

CD \ AB	00	01	11	10
00		0		0
01		0		0
11				0
10		0		0

CD \ AB	00	01	11	10
00	1	0	1	0
01	1	0	1	0
11	1	1	1	0
10	1	0	1	0

Figure 272: simplificarea expresiei booleene prin intermediul hărții Karnaugh

Completăm apoi toate celelalte regiuni rămase libere cu valori de 1 (dreapta sus).

Formăm grupuri de 1 pentru toate regiunile ce conțin valori de 1. Scriem apoi rezultatul simplificat sub forma sumei de produse, conform secțiunii precedente a acestui capitol. Acest lucru este identic problemei precedente:

CD \ AB	00	01	11	10
00	1	0	1	0
01	1	0	1	0
11	1	1	1	0
10	1	0	1	0

$$\text{ieșire} = \overline{C}\overline{D} + CD + ABD$$

Figure 273: simplificarea expresiei booleene prin intermediul hărții Karnaugh

### 8.8.4 Comparație între soluțiile cu mintermeni și maxtermeni

În figura de mai jos sunt ambele soluții ale exemplurilor de mai sus, pentru comparație:

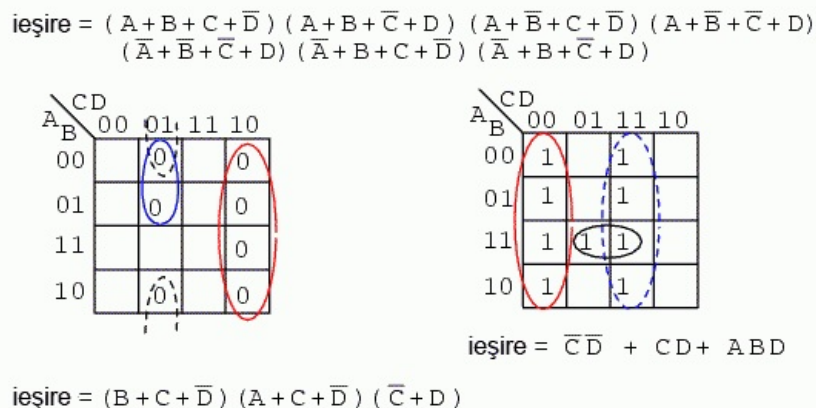


Figure 274: simplificarea expresiei booleene prin intermediul hărții Karnaugh

Care soluție este mai simplă? Dacă ar fi să implementăm fizic rezultatul sub formă de produs de sume, am avea nevoie de trei porți logice SAU și o poartă logică ȘI. Invers, dacă ar fi să implementăm rezultatul sub formă de sumă de produse, am avea nevoie de trei porți ȘI și o poartă SAU. În ambele situații am avea nevoie de patru porți. Să numărăm atunci și numărul de intrări ale porților. Prima variantă utilizează 8 intrări, iar a doua 7 intrări. Din definiția costului minim, soluția sub forma sumei de produse este mai simplă. Acesta este un exemplu tehnic corect, dar care nu ne este de prea mare folos în realitate.

Soluția „corectă” depinde însă de complexitate și de familia de porți logice folosite. Soluția sumei de produse este mai bună dacă folosim circuite TTL, a căror porți principale sunt porțile ȘI-negat. Acestea sunt foarte bune pentru implementări sub forma de sumă de produse. Pe de altă parte, soluția produsului de sume este acceptabilă dacă folosim circuite CMOS, deoarece avem astfel la dispoziție porți SAU-negat de toate mărimile.

### 8.8.5 Echivalența circuitelor ȘI-SAU cu circuitele ȘI-negat-ȘI-negat

Circuitele cu porți logice pentru ambele cazuri sunt prezentate mai jos, produsul de sume în stânga și suma de produse în dreapta:

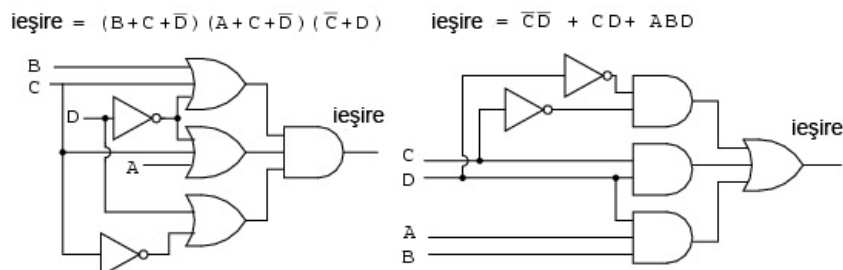


Figure 275: simplificarea expresiei booleene prin intermediul hărții Karnaugh

Reluăm mai jos (stânga) circuitul sub forma sumei de produse:

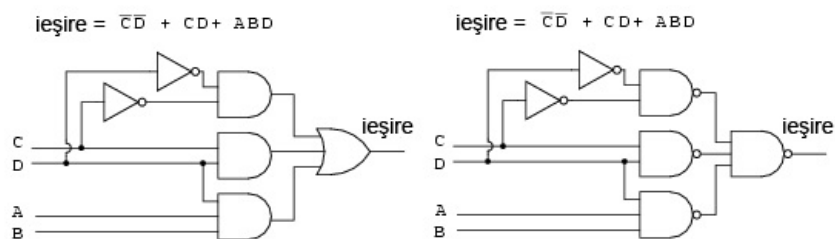


Figure 276: circuite cu porți logice

Dacă înlocuim toate porțile logice ȘI din stânga cu porți logice ȘI-negat, obținem rezultatul din dreapta sus. Poarta SAU de la intrare este înlocuită de asemenea cu o poartă ȘI-negat. Pentru a demonstra că logica ȘI-SAU este echivalentă cu logică ȘI-negat-ȘI-negat, este suficient să mutăm „cerculețele” inversoare de la ieșirea celor trei porți ȘI-negat la intrarea porții finale ȘI-negat, conform figurii de mai jos:

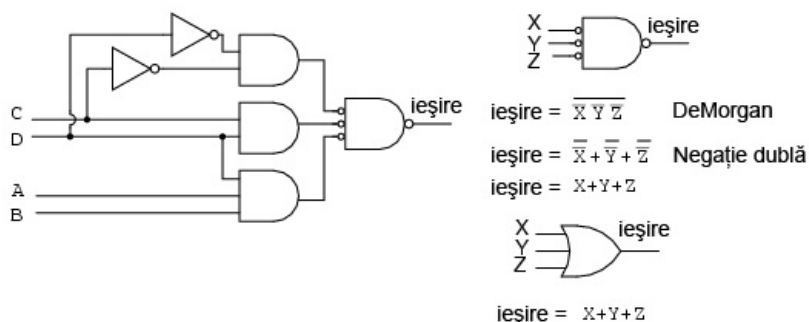


Figure 277: circuite cu porți logice

În figura de mai sus (dreapta), putem observa că ieșirea unei porți SI-negat cu intrări inversate este echivalentă din punct de vedere logic cu o poartă SAU, conform teoremei lui DeMorgan și a negației duble. Această informație ne este de ajutor în implementarea fizică a circuitelor digitale atunci când dispunem de circuite logice TTL cu porți ȘI-negat.

Pașii necesari construirii logicii ȘI-negat-ȘI-negat în locul logicii ȘI-SAU, sunt următorii:

- Realizăm un circuit logic (teoretic) sub formă de sumă de produse
- Când desenăm diagrama logică, înlocuim toate porțile logice (ȘI și SAU) cu porți logice ȘI-negat
- Intrările nefolosite trebuie legate la valoarea logică „înalt”
- În caz de defect, nodurile interne de la primul nivel de ieșire al porților ȘI-negat nu sunt identice cu valorile diagramei ȘI-SAU, ci sunt inversate. Folosim diagrama logică ȘI-negat-ȘI-negat. Intrările și ieșirile finale sunt identice, totuși
- Notăm fiecare capsulă (circuit integrat) cu U<sub>1</sub>, U<sub>2</sub>, etc.
- Folosim catalogul producătorului pentru conectarea pinilor circuitului integrat la intrările și ieșirile porților din circuit

#### 1. Exemplul 1

Să reluăm o problemă precedentă ce implică o simplificare sub formă sumei de produse. Vom realiza o simplificare sub forma unui produs de sume de această dată. Putem compara cele două soluții la final.

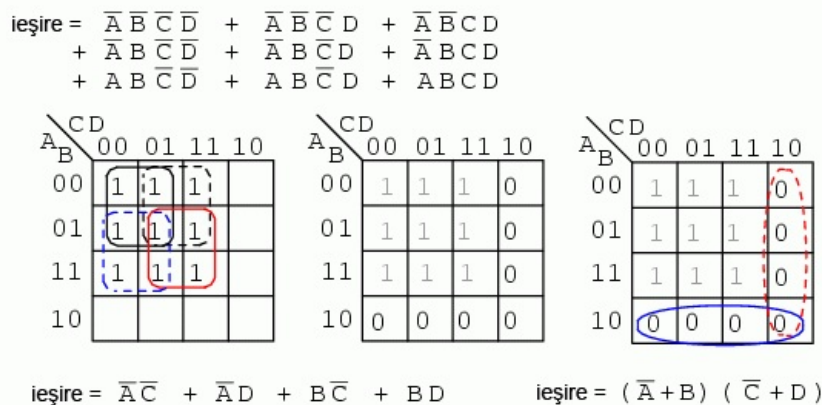


Figure 278: simplificarea expresiei booleene cu ajutorul hărți Karnaugh

Soluție: în figura de sus stânga avem problema inițială, o expresie booleană cu 9 mintermeni nesimplificată. Recapitulând, am format patru grupuri de câte patru regiuni fiecare. Rezultatul a fost o sumă de patru produse (partea din stânga, jos).

În figura din mijloc, completăm regiunile rămase libere cu valori de 0. Formăm două grupuri de câte patru regiuni. Grupul de jos (albastru) este  $A' + B$ , iar grupul din dreapta (roșu) este  $C' + D$ . Rezultatul este prin urmare un produs de două sume,  $(A' + B)(C' + D)$ .

Comparând cele două soluții de mai sus, putem observa că soluția produsului de sume reprezintă soluția cu cel mai mic cost. Pentru implementarea primei soluții am avea nevoie de 5 porți, iar pentru soluția produsului de sume am avea nevoie doar de 3. Folosind circuite logice TTL, aceasta din urmă este și atractivă datorită simplității rezultatului. Putem găsi porți logice ȘI și SAU cu 2 intrări. Mai jos sunt prezentate circuitele logice pentru ambele soluții

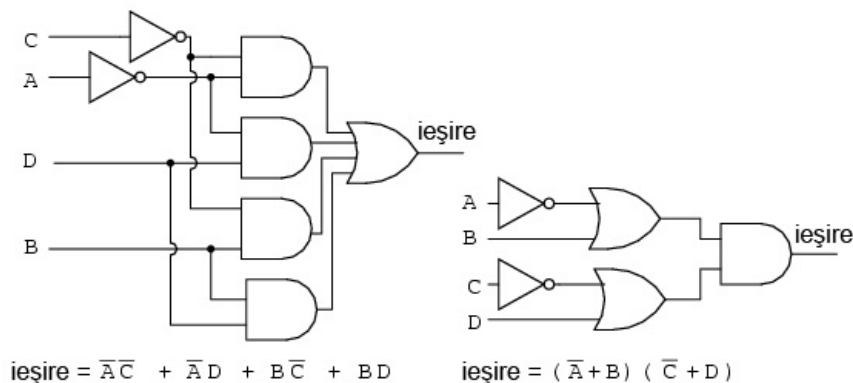


Figure 279: circuite cu porți logice

Să presupunem că avem la dispoziție circuitele logice TTL de mai jos. În acest caz, cunoaștem și poziționarea porților logice în interiorul acestora, precum în figura de mai jos:

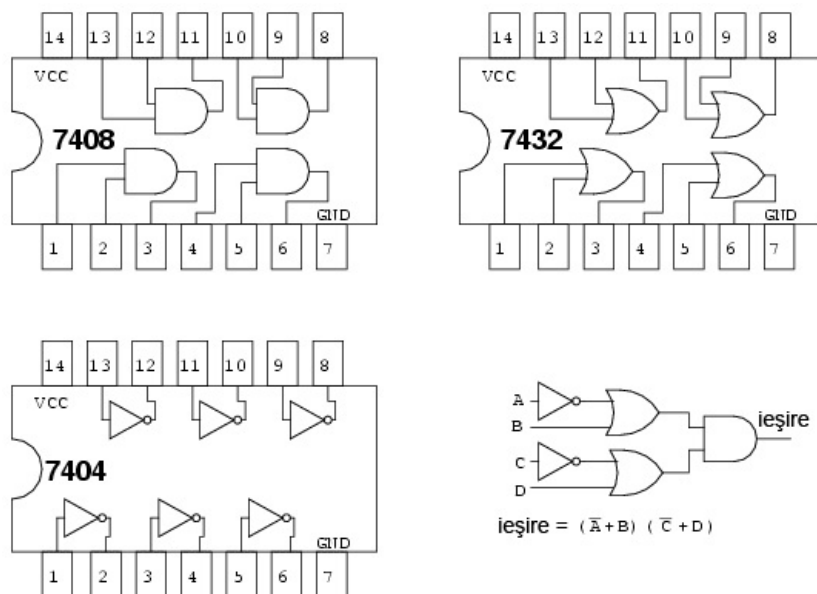


Figure 280: circuite logice TTL

Circuitele integrate folosite (trei la număr) vor fi identificate prin notația U1, U2 respectiv U3. Pentru a face distincție între porțile individuale din fiecare capsulă, acestea vor fi identificate prin a, b, c, d, etc. Circuitul inversor 7404 va fi U1. Porțile inversoare individuale sunt U1-a, U1-b, U1-c, etc. Circuitul SAU 7432 va fi notat cu U2, iar U3 este notație folosită pentru circuitul ȘI 7408.

Luând în considerare pinningul circuitelor logice folosite mai sus, vom desemna toate intrările și ieșirile circuitului logic ce vrem să-l construim, conform figurii de mai jos (intrările porților nefolosite se vor lega la masă):

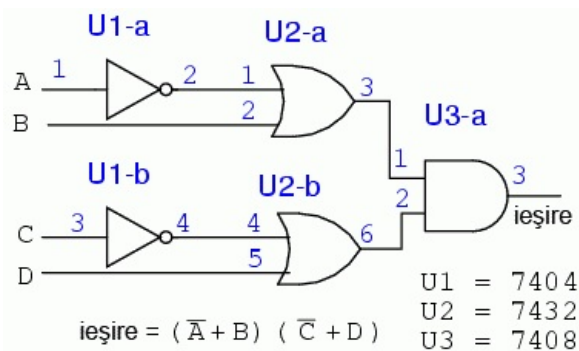


Figure 281: circuit logic

Putem găsi cu ușurință porți logice ȘI cu două intrări (7408, stânga). Totuși, este mai greu să găsim o poartă logică SAU cu patru intrări. Singurul tip de poartă cu patru intrări este un circuit TTL 7420 cu porți ȘI-negat (dreapta):

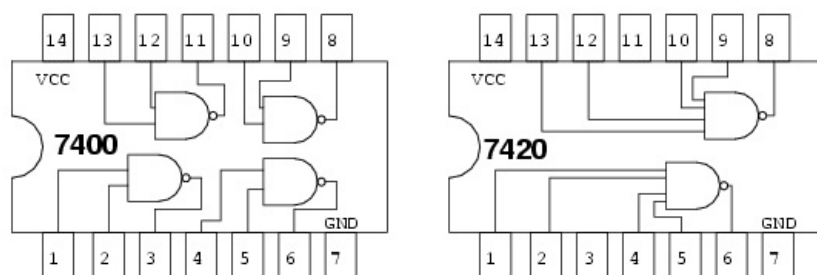


Figure 282: circuite logice TTL

Putem transforma poarta logică ȘI-negat cu patru intrări într-o poartă logică SAU cu patru intrări prin inversarea intrărilor acesteia:

$$\begin{aligned} \bar{Y} &= \bar{A} \bar{B} = \overline{A+B} & \text{DeMorgan} \\ Y &= A+B & \text{Negare dublă} \end{aligned}$$

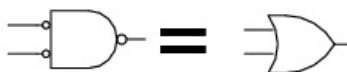


Figure 283: transformarea funcției logice ȘI-negat în SAU

Putem prin urmare folosi circuitul 7420 cu porți logice ȘI-negat cu patru intrări ca și poartă SAU prin negarea (inversarea) intrărilor.

Nu vom folosi porți logice inversoare discrete pentru inversarea intrărilor circuitului 7420. Vom folosi în schimb porți logice ȘI-negat cu două intrări în locul porților ȘI din soluția booleană cu mintermeni (sumă de produse). Inversarea ieșirii porților ȘI-negat cu două intrări este suficientă pentru inversarea necesară realizării porții logice SAU cu patru intrări:

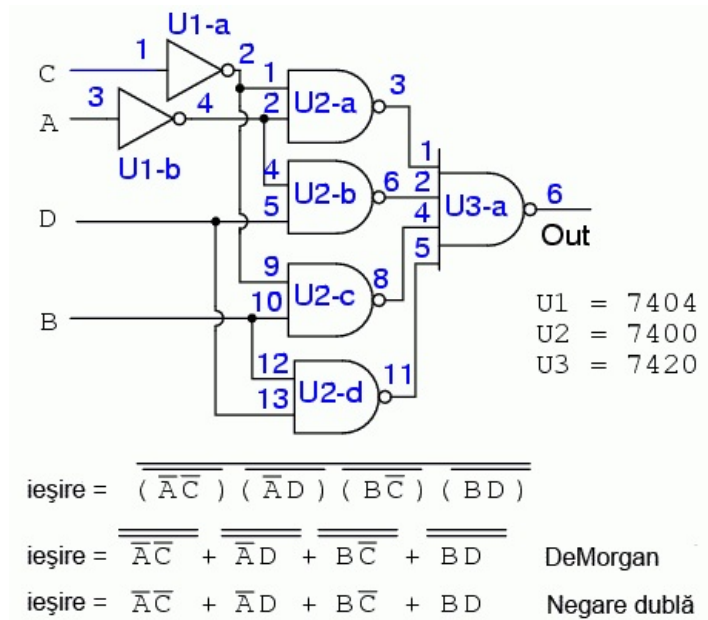


Figure 284: circuit logic - notarea intrărilor şi ieşirilor

Rezultatul de mai sus este singura modalitate practică de realizarea a circuitului folosind TTL cu porţi logice ŞI-negat-ŞI-negat în locul porţilor ŞI-SAU.

## 8.9 Notăția Σ (sumă) și notația Π (produs)

Ca și referința, această secțiune introduce terminologia folosită în unele texte pentru descrierea mintermenilor și maxtermenilor aparținând hărților Karnaugh. Mai departe de atât, această secțiune nu conține nimic nou.

### 8.9.1 Notăția Σ (sumă) pentru mintermeni

Simbolul Σ (sigma) indică o sumă iar litera „m” indică mintermenii. Prin urmare, Σm reprezintă o sumă de mintermeni. Următorul exemplu ilustrează afirmația de mai sus. În loc de ecuație booleană, am ales să enumerăm mintermenii:

$f(A,B,C,D) = \Sigma m(1, 2, 3, 4, 5, 7, 8, 9, 11, 12, 13, 15)$  sau  $f(A,B,C,D) = \Sigma(m1,m2,m3,m4,m5,m7,m8,m9,m11,m12,m13,m15)$

Indicii termenilor indică locația regiunii, sau adresa, dintr-o hartă Karnaugh. Acesta este cu siguranța un mod mult mai compact pentru descrierea mintermenilor sau regiunilor unei hărți Karnaugh.

$$\begin{aligned} \text{ieşire} = & \overline{A} \overline{B} \overline{C} \overline{D} + \overline{A} \overline{B} \overline{C} D + \overline{A} \overline{B} C D \\ & + \overline{A} B \overline{C} \overline{D} + \overline{A} B \overline{C} D + \overline{A} B C D \\ & + A B \overline{C} \overline{D} + A B \overline{C} D + A B C D \end{aligned}$$

$$f(A, B, C, D) = \Sigma m(0, 1, 3, 4, 5, 7, 12, 13, 15)$$

A \ B	C D			
	00	01	11	10
00	0	1	3	2
01	4	5	7	6
11	12	13	15	14
10	8	9	11	10

A \ B	C D			
	00	01	11	10
00	1	1	1	0
01	1	1	1	0
11	1	1	1	0
10	0	0	0	0

A \ B	C D			
	00	01	11	10
00	1	1	1	0
01	1	1	1	0
11	1	1	1	0
10	0	0	0	0

$$f(A, B, C, D) = \overline{A} \overline{C} + \overline{A} D + B \overline{C} + B D$$

Figure 285: harta Karnaugh

Soluția exprimată sub formă sumei de produse nu este afectată prin utilizarea acestei terminologii. Mintermenii de pe harta (valorile de 1) sunt grupați ca de obicei, iar mai apoi putem scrie o soluție sub forma sumei de produse.

### 8.9.2 Notăția Π (produs) pentru maxtemeni

Mai jos luăm în considerare și terminologia folosită pentru descrierea unei liste de maxtermeni. Produsul este indicat prin litera Π (pi), iar „M” indică maxtermenii. Prin urmare, ΠP indică un produs de maxtermeni. Putem folosi același exemplu pentru ilustrarea celor spuse mai sus. Ecuația logică booleană nesimplificată este înlocuită cu o listă de maxtermeni:

$f(A,B,C,D) = \Pi M(2, 6, 8, 9, 10, 11, 14)$  sau  $f(A,B,C,D) = \Pi(M2, M6, M8, M9, M10, M11, M14)$

Din nou, numerele indică adresa sau locația pe harta Karnaugh. Pentru maxtermeni, acestea reprezintă locațiile valorilor de 0. Soluția sub forma produsului de sume se scrie ca de obicei.



$$\text{ieşire} = \overline{(\overline{A} + \overline{B} + \overline{C} + \overline{D})} (\overline{A} + \overline{B} + \overline{C} + D) (\overline{A} + \overline{B} + \overline{C} + D) (\overline{A} + B + C + D) \\ (\overline{A} + B + \overline{C} + \overline{D}) (\overline{A} + B + \overline{C} + \overline{D}) (\overline{A} + B + \overline{C} + D)$$

$$f(A, B, C, D) = \prod M(2, 6, 8, 9, 10, 11, 14)$$

$\begin{array}{c} \text{CD} \\ \text{A} \backslash \text{B} \end{array}$	00	01	11	10
00	0	1	3	2
01	4	5	7	6
11	12	13	15	14
10	8	9	11	10

$\begin{array}{c} \text{CD} \\ \text{A} \backslash \text{B} \end{array}$	00	01	11	10
00	1	1	1	0
01	1	1	1	0
11	1	1	1	0
10	0	0	0	0

$\begin{array}{c} \text{CD} \\ \text{A} \backslash \text{B} \end{array}$	00	01	11	10
00	1	1	1	0
01	1	1	1	0
11	1	1	1	0
10	0	0	0	0

$$f(A, B, C, D) = \overline{(\overline{A} + B)} (\overline{C} + D)$$

Figure 286: harta Karnaugh

## 8.10 Hărți Karnaugh de 5 și 6 variabile

Pentru reducerea circuitelor logice mai mari se folosesc, evident, hărți Karnaugh mai mare. Dar care este mărimea maximă (practică) a unei hărți Karnaugh? Acest lucru depinde de numărul de intrări a circuitului logic considerat. Practic, se poate constata că această limită este de 6 intrări. Prezentăm mai jos așadar hărțile Karnaugh de 5 și 6 variabile.

### 8.10.1 Harta Karnaugh de 5 variabile

Prima variantă a hărții Karnaugh de 5 variabile este modelul în oglindă. Desigur, numerotarea se realizează în cod Gray (partea de sus). Acesta se reflectă aproximativ la mijlocul hărții. Acest stil este folosit de textele mai vechi:

$\begin{array}{c} \text{CDE} \\ \text{A} \backslash \text{B} \end{array}$	000	001	011	010	110	111	101	100
00								
01								
11								
10								

Figure 287: hartă Karnaugh de 5 variabile (stil vechi, în oglindă)

Variantă preferată, cea cu suprapunere, este prezentată mai jos:

$\begin{array}{c} \text{CDE} \\ \text{A} \backslash \text{B} \end{array}$	000	001	011	010	100	101	111	110
00								
01								
11								
10								

Figure 288: hartă Karnaugh de 5 variabile (stil nou, cu suprapunere)

Această variantă constă pur și simplu din două (patru pentru o hartă Karnaugh de 6 variabile) hărți identice, cu excepția bitului cel mai semnificativ din adresa de 3 biți din partea superioară. Dacă ne uităm în partea de sus a hărții, observăm că numerotația este diferită față de harta precedentă (în cod Gray). Dacă ignorăm bitul cel mai semnificativ, precum am spus mai sus, secvența 00, 01, 11, 10 se regăsește în partea superioară a ambelor sub-hărți. Secvența formată din cele opt numere de 3 biți nu este cod Gray.

#### 1. Harta Karnaugh cu 5 variabile - exemplu

Să proiectăm un circuit cu 5 intrări binare (A, B, C, D, E), A fiind bit-ul cel mai semnificativ. Circuitul va trebui să producă o ieșire „înaltă” pentru orice număr prim detectat la intrare:

Prezentăm mai jos o soluție sub forma hărții Karnaugh de 5 variabile în oglindă, folosind cod Gray:

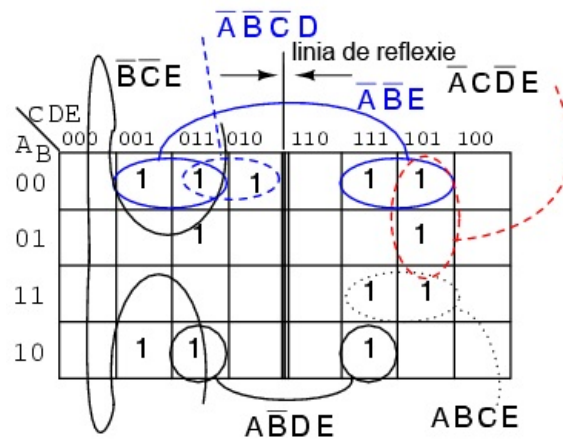


Figure 289: hartă Karnaugh de 5 variabile (stil vechi, în oglindă)

Numerele prime sunt (1,2,3,5,7,11,13,17,19,23,29,31). Introducem o valoare de 1 în fiecare regiune corespunzătoare. Trecem apoi la gruparea regiunilor și scrierea rezultatului simplificat. Observați că grupul de patru regiuni A'B'E conține două perechi de câte două regiuni aflate de fiecare parte a liniei de reflexie. Același lucru este valabil și pentru grupul format din două regiuni AB'DE. Aceste grupuri se formează prin reflexie. Atunci când folosim acest stil de hartă Karnaugh, va trebui să căutăm astfel de grupuri reflectate. Expresia booleană simplificată mai sus este următoarea:

$$\text{ieșire} = A'B'E + B'C'E + A'C'DE + A'CD'E + ABCE + AB'DE + A'B'C'D$$

Să considerăm și varianta hărții Karnaugh cu 5 variabile, cu suprapunere:

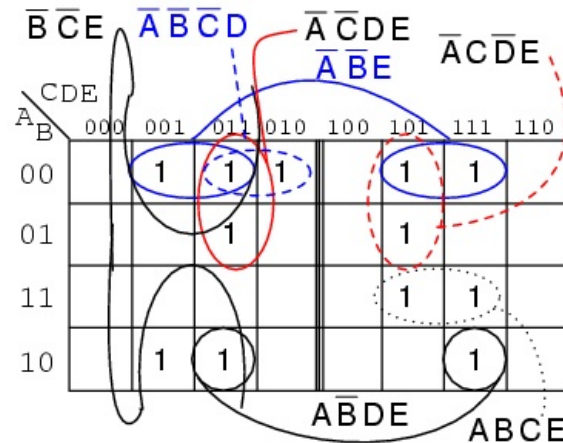


Figure 290: hartă Karnaugh de 5 variabile (stil nou, cu suprapunere)

Dacă facem o comparație între cele două variante de sus, anumite regiuni din partea dreaptă a hărții își modifică locație, din moment ce adresele din partea de sus a hărții s-au modificat. Trebuie de asemenea să găsim o altă modalitate de grupare a termenilor din cele două jumătăți ale hărții. Soluția constă în suprapunerea (imaginară) a celor două jumătăți. Orice suprapunere a hărții de deasupra cu harta de dedesupt prezintă o posibilă grupare. Figura de mai jos indică faptul că grupul AB'DE este compus din două regiuni suprapuse. Grupul A'B'E este format din două perechi de regiuni suprapuse:

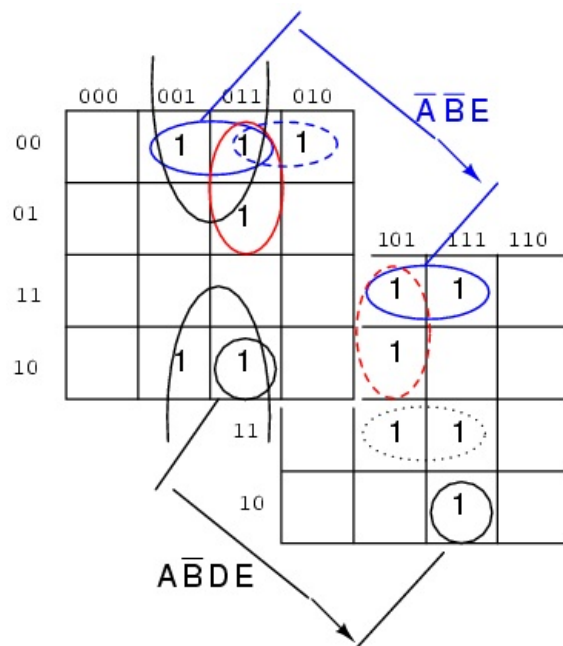


Figure 291: hartă Karnaugh de 5 variabile (stil nou, cu suprapunere)

Pentru grupul A'B'E de patru regiuni, ABCDE = 00xx1. Cu alte cuvinte, variabilele A, B și E sunt aceleași (001) pentru grup. Pe



de altă parte,  $CD = xx$  (acesta variabile nu sunt identice pentru grup). Din moment ce  $ABCDE = 00xx1$ , grupul de patru regiuni este acoperit de  $A'B'XE = A'B'E$ .

### 8.10.2 Hartă Karnaugh de 6 variabile

Luăm acum un exemplu de utilizare a unei hărți Karnaugh de 6 variabile. Am suprapus (imaginar) cele patru sub-hărți pentru a putea vizualiza gruparea de patru regiuni corespunzătoare ieșirii C'F':

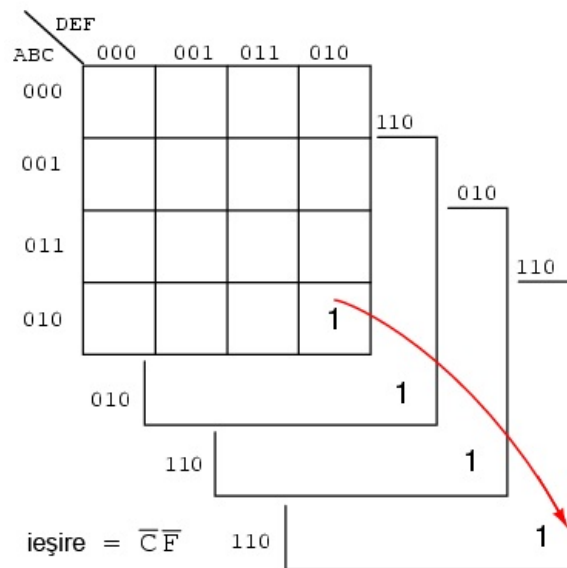


Figure 292: hartă Karnaugh de 6 variabile (suprapunere)

Un comparator de amplitudine (utilizat pentru ilustrarea utilizării hărții Karnaugh de 6 variabile) compară două numere binare. Acesta indică dacă cele două numere sunt egale, mai mici sau mai mari unul față de celălalt. Un astfel de comparator are trei ieșiri:

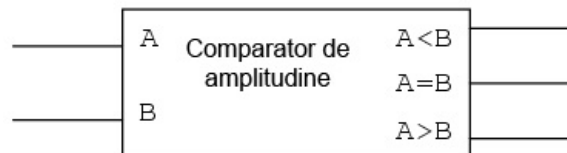


Figure 293: comparator digital de amplitudine

Un comparator de amplitudine pe trei biți are două intrări:  $A_2A_1A_0$  și  $B_2B_1B_0$ . Un comparator de amplitudine sub forma unui circuit integrat (7485) are practic patru intrări. Totuși, harta Karnaugh de mai jos trebuie menținută la o mărime rezonabilă. Vom rezolva problema doar pentru ieșirea  $A > B$ .

Pentru simplificarea logicii comparatorului de amplitudine pe 3 biți, folosim harta Karnaugh cu 6 variabile de mai jos. Această variantă este cea cu suprapunere. Codul binar folosit nu este cod Gray. Găsim expresiile redundante prin suprapunerea celor patru sub-hărți, precum am arătat mai sus. Am putea găsi regiuni comune tuturor celor patru hărți, deși, în exemplul de mai jos nu este cazul. Putem observa totuși că există regiuni comune sub-hărților:

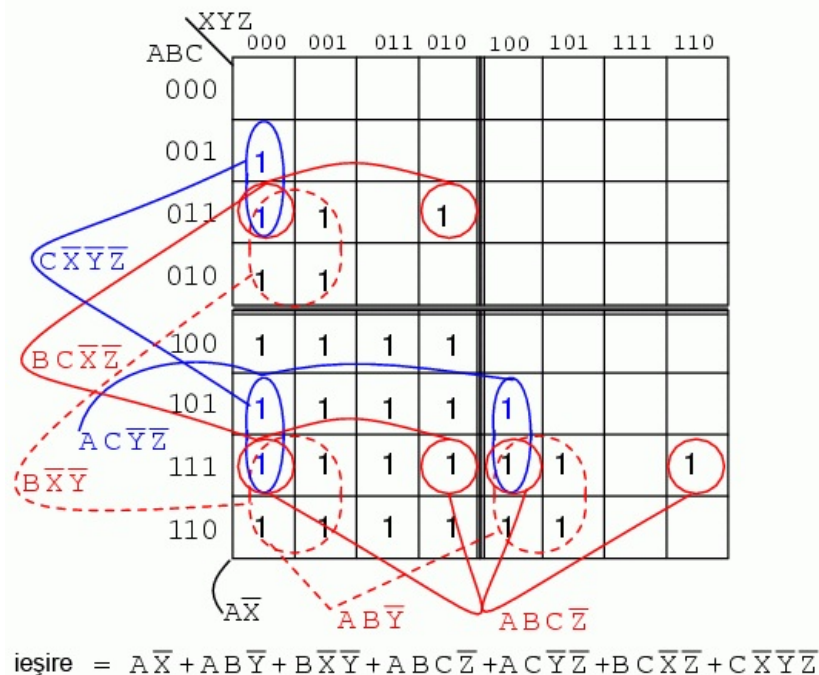


Figure 294: hartă Karnaugh de 6 variabile (suprapunere)

Ieșirea  $A > B$  este reprezentată de  $ABC > XYZ$  pe harta de mai sus. Ori de câte ori  $ABC$  este mai mare decât  $XYZ$ , avem o valoare de 1 pe hartă. Pe prima linie,  $ABC = 000$  nu poate fi mai mare decât nicio valoare a lui  $XYZ$ . Nu avem nici o valoare de 1 pe această linie.

Pe linia a doua,  $ABC = 001$ , și doar în prima regiune,  $ABCXYZ = 001000$ ,  $ABC$  este mai mare decât  $XYZ$ . Avem un singur 1 în prima regiune a celei de a doua linii. Pe linia a patra,  $ABC = 010$ , există o pereche de 1. Pe linia a treia,  $ABC = 011$  și avem trei valori de 1. Prin urmare, harta este completată cu valori de unu ori de câte ori  $ABC$  este mai mare decât  $XYZ$ . Pentru gruparea regiunilor, acolo unde este posibil, încercăm să formăm grupuri cu sub-hărțile adiacente. Toate grupurile în afară de un grup de 16 regiuni sunt formate cu regiuni aparținând sub-hărților adiacente. Rezultatul este: 1 grup de 16 regiuni; 2 grupuri de 8 regiuni; 4 grupuri de 4 regiuni. Grupul de 16 regiuni,  $AX'$ , ocupă toată sub-harta din partea de jos-stânga a hărții Karnaugh, deși, în figura de mai sus, aceasta nu este încercuită. Numărând valorile de 1 de pe hartă, ajungem la un total de  $16 + 6 + 6 =$

1. Înainte de reducerea logică folosind harta Karnaugh de mai sus, soluția logică sub formă de sumă de produse ar fi avut 28 de termeni, fiecare cu 6 intrări. Simplificarea logică cu ajutorul hărții Karnaugh de mai sus, a redus numărul termenilor la șapte, fiecare cu un număr de patru sau mai puțin de patru intrări. Acesta este de fapt scopul hărților Karnaugh!

## 9 Circuite logice combinationale

### 9.1 Circuite logice combinaționale - introducere

Termenul „combi-național” provine din matematică. În matematică, o combinație reprezintă o mulțime neordonată. Pe scurt, este un mod de a spune că nimănui nu-i pasă ordinea elementelor mulțimii respective. Majoritatea jocurilor funcționează exact în acest fel: dacă aruncăm zarurile, pe rând, nu contează dacă am dat un 2 urmat de 3 sau un 3 urmat de 2; rezultatul este același. Același lucru este valabil și pentru circuitele logice combinaționale: ieșirea circuitului este identică indiferent de ordinea intrărilor. Desigur, există și circuite a căror ieșire depinde de ordinea intrărilor, iar aceste circuite poartă numele de circuite logice secvențiale. Deși nu există un capitol cu acest titlul, următoarele capitole din volumul electronicii digitale se vor ocupa cu aceste circuite secvențiale. Circuitele practice sunt realizate dintr-o combinație de circuite logice combinaționale și secvențiale. Cele secvențiale asigură faptul că totul se întâmplă într-o anumită ordine. Cele combinaționale realizează funcții aritmetice, logice sau de conversie. Am folosit deja circuite logice combinaționale. Fiecare dintre porțile logice discute este un astfel de circuit. Să urmărim comportamentul a două porți logice ȘI-negat, atunci când intrările acestora sunt alimentate în diferite combinații. Când ambele intrări sunt 0:

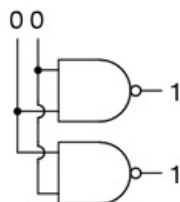


Figure 295: circuit logic combinațional format din două porți logice ȘI-negat

Când una dintre intrări este 1:

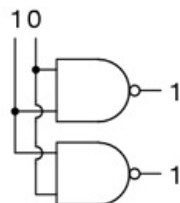


Figure 296: circuit logic combinațional format din două porți logice ȘI-negat

Când și cealaltă intrare este 1:

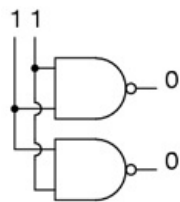


Figure 297: circuit logic combinațional format din două porți logice ȘI-negat

Prin urmare, porțile ȘI-negat sunt indiferente la ordinea de alimentare a intrărilor. Același lucru este valabil și pentru celelalte porți logice discutate până în acest moment (ȘI, SAU-exclusiv, SAU, SAU-negat, SAU-negat-exclusiv și NU).

### 9.2 Half-Adder

Ca și prim exemplu a unui circuit logic combinațional util, să realizăm un dispozitiv ce realizează adunarea a două numere binare. Putem calcula rapid care ar trebui să fie răspunsul unui astfel de dispozitiv:

$$0 + 0 = 0 \quad 0 + 1 = 1 \quad 1 + 0 = 1 \quad 1 + 1 = 10_2$$

Avem nevoie prin urmare de două intrări (a și b) și de două ieșiri. Prima ieșire o denumim  $\Sigma$ , deoarece reprezintă suma. A doua ieșire o numim  $C_{out}$  și reprezintă bitul de depășire. Tabelul de adevăr corespunzător este reprezentat mai jos:

A	B	$\Sigma$	$C_{out}$
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Figure 298: tabel de adevăr

Simplificând ecuațiile booleene, sau realizând o hartă Karnaugh, vom obține circuitul de mai jos:

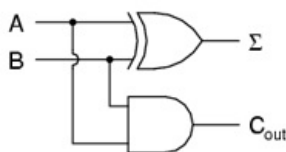


Figure 299: circuit logic combinațional

Dar să ne uităm mai atent la rezultate. Coloana  $\Sigma$  nu este altceva decât o poartă logică SAU-exclusiv. Coloana  $C_{out}$  reprezintă o poartă logică ȘI. Acest dispozitiv poartă numele de half-adder (semisumator), din motive pe care le vom vedea în secțiunea următoare.

Sub forma unei diagrame ladder, circuitul de mai sus arată astfel:

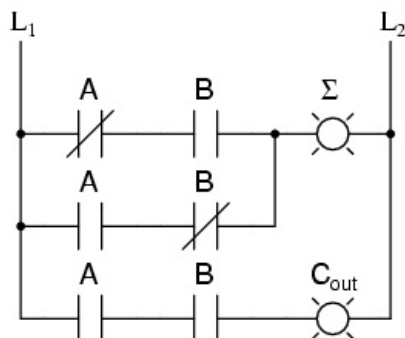


Figure 300: diagramă ladder

### 9.3 Full-Adder

Circuitul half-adder (semi-sumator) este extrem de util până în momentul în care dorim adăugarea unor valori ce nu pot fi reprezentate cu un singur bit. Cea mai lentă metodă de realizare a sumatoarelor pe doi biți constă în realizarea unui tabel de adevăr (și reducerea acestuia). Dacă avem nevoie de un sumator pe trei biți, patru biți, etc., repetăm acest proces. Circuitele vor fi rapide, dar timpul de realizare al lor va fi destul de lung.

Să ne uităm puțin la o sumă dintre două numere pe doi biți. Putem să ne facem astfel o idee a circuitului ce dorim să-l implementăm:

11  
11  
11  
---  
110

Putem observa numărul de intrări necesar coloanei din mijloc. Circuitul nostru sumator are nevoie de trei intrări: a, b și bit-ul de depășire. Putem folosi sumatorul cu două intrări pentru construirea unui sumator cu trei intrări.

Termenul  $\Sigma$  este destul de ușor de obținut. Aritmetica ne spune că în cazul în care  $\Sigma = a + b + C_{in}$  și  $\Sigma_1 = a + b$ , atunci  $\Sigma = \Sigma_1 + C_{in}$ :

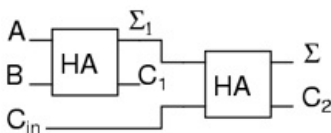


Figure 301: full-adder

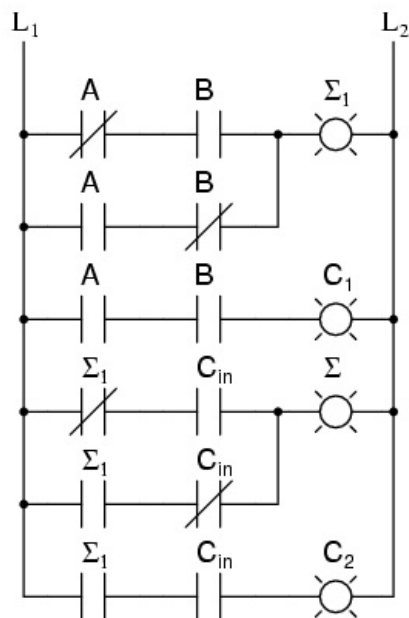


Figure 302: diagramă ladder

La ce ne ajută însă  $C_1$  și  $C_2$ ? Să analizăm rezultatul adunării celor trei intrări:

$$C_{in} + a + b = ?$$

$$0 + 0 + 0 = 0$$

$$0 + 0 + 1 = 1$$

$$0 + 1 + 0 = 1$$

$$0 + 1 + 1 = 10$$

$$1 + 0 + 0 = 1$$

$$1 + 0 + 1 = 10$$

$$1 + 1 + 0 = 10$$

$$1 + 1 + 1 = 11$$

Dacă aveți nelămuriri legate de bitul de rang inferior, puteți verifica dacă circuitul și diagrama ladder l-au calculat corect.

Pentru a calcula bitul de rang superior, putem observa că valoarea acestuia este 1 în ambele cazuri în care  $a + b$  produce un  $C_1$ . De asemenea, bitul de rang superior este 1 când  $a + b$  produce un  $\Sigma_1$ , iar  $C_{in}$  este 1. Prin urmare, vom avea un bit de depășire ori de câte ori avem  $C_1$  sau ( $\Sigma_1$  și  $C_{in}$ ). Sumatorul nostru complet (full-adder) cu trei intrări, arată astfel:

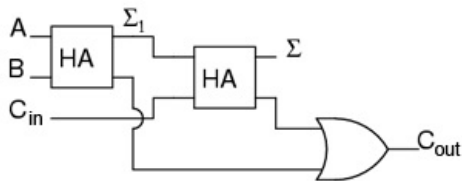


Figure 303: full-adder

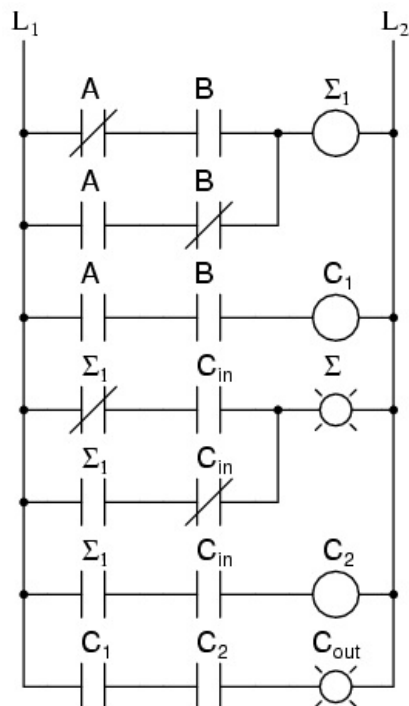


Figure 304: diagramă ladder

Pentru unele circuite, eliminarea uneia sau mai multor tipuri de porți poate fi importantă. Putem înlocui poarta SAU finală cu o poartă SAU-exclusiv fără a modifica rezultatele. Putem acum conecta două sumatoare pentru realizarea adunării numerelor pe 2 biți:

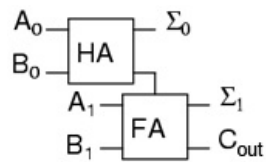


Figure 305: full-adder

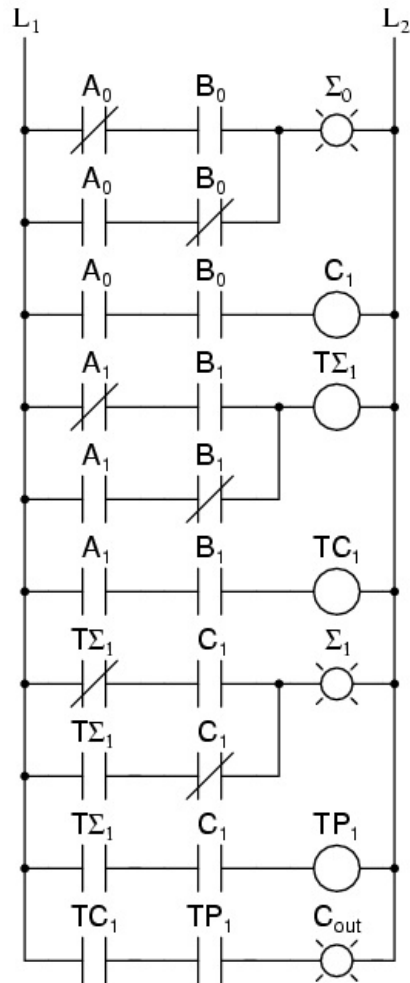


Figure 306: diagramă ladder

$A_0$  este bitul de rang inferior a lui A iar  $A_1$  este bitul de rang superior a lui A. Același lucru este valabil și pentru B.  $\Sigma_0$  este bitul de rang inferior al sumei iar  $\Sigma_1$  este bitul de rang superior al sumei.  $C_{out}$  este bitul de depășire.

### 9.3.1 Sume de numere mai mari de doi biți

Un sumator pe doi biți nu va fi realizat niciodată în acest fel. În schimb, biți de rang inferior vor trece și ei printr-un sumator complet (full-adder):

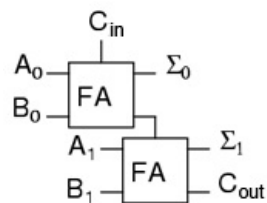


Figure 307: full-adder

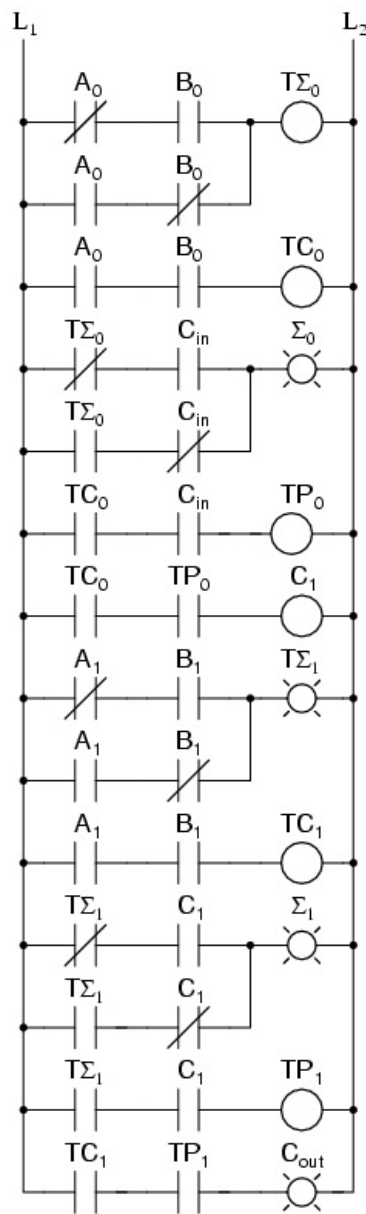


Figure 308: diagramă ladder

Există câteva motive pentru această alegere. Unul dintre ele este că în acest caz, permitem circuitului să determine dacă bitul de depășire de rang inferior este inclus în sumă. Acest lucru permite însumarea unor numere mai mari. Să considerăm două moduri diferite de analiză a unei sume de numere pe patru biți:

111	1<-+	11<+-
0110	01   10	
1011	10   11	
-----	-   -   ---	
10001	1 +-100 +-101	

Dacă permitem programului însumarea numerelor pe doi biți, și reținem bitul de depășire, putem folosi acest bit de depășire în următoarea sumă. În acest fel, programul poate însuma orice număr de biți, chiar dacă folosim un sumator pe doi biți.

### 9.3.2 Conectarea sumatoarelor între ele

Aceste sumatoare complete pot fi extinse pe un număr de biți oricât de mare. Ca și exemplu, un sumator pe 8 biți poate fi realizat astfel:

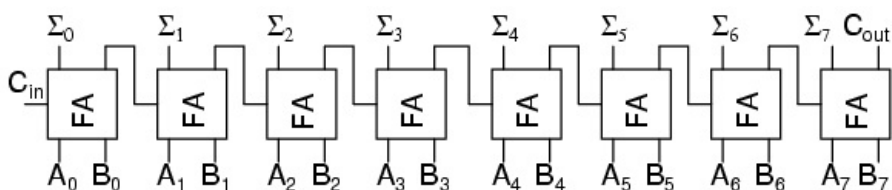


Figure 309: conectarea sumatoarelor

Acest rezultat este identic utilizării sumatoarelor pe doi biți pentru realizarea unui sumator pe 4 biți, și utilizării a două astfel de sumatoare pe 4 biți pentru realizarea unui sumator pe 8 biți:

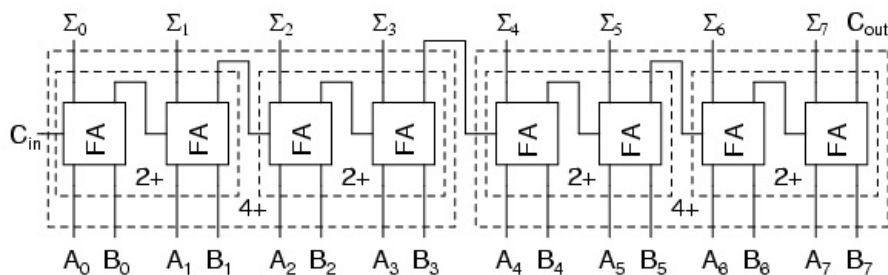


Figure 310: conectarea sumatoarelor

Fiecare „2+” este un sumator pe 2 biți și realizat din două sumatoare complete (full-adder). Fiecare „4+” este un sumator pe 4 biți realizat din două sumatoare pe 2 biți. Iar rezultatul celor două sumatoare pe 4 biți este un sumator pe 8 biți.

### 9.3.3 Multiplicarea circuitelor simple sau construirea integrală a

dispozitivului

Există două metode principale de realizare a oricărui circuit logic combinațional mare: putem folosi circuite simple, multiplicându-le; sau putem proiecta întregul circuit complex ca și un singur dispozitiv. Utilizând circuite simple pentru realizarea circuitelor complexe, timpul petrecut pentru proiectarea lor scade foarte mult. Dezavantajul este că semnalele necesită un timp mai lung de propagare prin tranzistori. Sumatorul pe 8 biți de mai sus trebuie să aștepte ca toate semnalele C<sub>xout</sub> să treacă de la A<sub>0</sub> + B<sub>0</sub> spre intrările A<sub>7</sub> + B<sub>7</sub>.

Dacă în schimb, proiectăm sumatorul pe 8 biți ca și dispozitiv complet, simplificat la o sumă de produse, atunci fiecare semnal trece printr-o singură poartă logică NU, o poartă logică ȘI și o poartă logică SAU. Un dispozitiv cu 17 intrări are un tabel de adevăr cu 131.072 de intrări, iar reducerea acestor intrări la o sumă de produse va lua ceva timp.

Atunci când proiectăm circuite pentru sisteme ce au un timp de răspuns maxim pentru obținerea rezultatului final, putem începe partea de proiectare prin utilizarea circuitelor simple. Putem încerca apoi înlocuirea porțiunilor de circuit ce sunt prea „lente”. În acest fel, ne putem concentra pe porțiunile de circuit care contează cel mai mult.

Author: Mihai Olteanu

Created: 2019-05-18 Sat 13:22

[Validate](#)