Cuprins

PREFAȚĂ

Manualul de faţă se adresează studenţilor din anul II de la specializarea „Telecomenzi și Electronică în Transporturi”, domeniul Inginerie Electronică și Telecomunicații, din Facultatea Transporturi, Universitatea POLITEHNICA din București. De asemenea, manualul prezintă interes și pentru studenții de la celelalte specializări din domeniul menționat, precum și pentru specialiști din domeniu sau cu pregătiri în domenii conexe (inclusiv pentru proiectarea circuitelor și/sau sistemelor digitale).

În lucrare sunt prezentate definiții și concepte de bază necesare pentru studiul, proiectarea și realizarea unor circuite digitale frecvent utilizate, cu exemple de documentații și de proiectare.

Autorii sunt cadre didactice din Catedra Telecomenzi și electronică în transporturi și au o bogată experiență în activitatea didactică (curs, seminar, laborator, proiect), atât în ceea ce privește disciplinele din trunchiul comun al domeniului, cât și în cadrul disciplinelor care definesc specializarea. Se mai impune și mențiunea că pentru aplicațiile specifice domeniilor de transport, indiferent de mod, sunt esențiale rezolvările care asigură siguranța și securitatea proceselor și a tehnicilor aferente, fiabilitatea, protecția mediului înconjurător, mentenanța ș.a.

De altfel, elaborarea manualului se înscrie într-un context mai larg, care include și proiecte și contracte de cercetare, de consultanță, publucarea unor lucrări științifice, participări la manifestări științifice interne și internaționale (congrese, conferințe, simpozioane).

Îmi exprim convingerea că și acest manual va fi foarte util și va contribui la o creștere a nivelului de pregătire teoretică și practică a studenților noștri.

Îi felicit pe autori și apreciez contribuția pe care o aduc în ceea ce privește preocuparea pentru elaborarea unor manuale didactice, asigurând astfel continuitatea într-o activitate care a început încă din anii '60 și a fost continuată după 1975 (an în care specialitatea a fost transferată de la Facultatea de Electronică și telecomunicații - Electronică, telecomunicații și tehnologia informației în prezent - la Facultatea Transporturi).

Prof.dr.ing. Corneliu Mihail ALEXANDRESCU
Decan al Facultății TRANSPORTURI
Universitatea Politehnica din București
membru corespondent al ASTR

1. Introducere

1.1 Circuite digitale

Sunt denumite „circuite logice”. Cursul îşi propune să trateze principiile şi implementarea lor. Majoritatea principiilor expuse aici îşi vor păstra valabilitatea încă mulţi ani; unele vor avea aplicaţii care la ora actuală nici nu au fost descoperite. În ceea ce priveşte realizarea practică, s-ar putea ca ea să fie puţin diferită de cele expuse în paginile de faţă şi, cu siguranţă, se va modifica permanent de-a lungul timpului.

Cursul îşi propune să prezinte principiile de bază ale circuitelor digitale, în suficientă măsură pentru a putea fi înţelese când se desfăşoară o anumită activitate folosind tehnica de calcul. Aceleaşi principii pot arăta unde este eroarea atunci când nu totul merge cum trebuie.

Mai jos sunt menţionate câteva reguli care trebuie însuşite din studiul curent. Probabil că multe dintre ele nu au nici o semnificaţie, deocamdată, însă este bine să le revedeţi ulterior.

Proiectarea circuitelor digitale înseamnă inginerie, iar inginerie înseamnă rezolvarea unor probleme. Doar 5%-10% din munca de proiectare reprezintă partea plăcută, de creaţie, scânteia interioară, imaginarea unui nou mod de abordare. Cam tot restul este muncă de rutină. Desigur, acest rest se face astăzi mult mai uşor şi plăcut decât acum 20 sau chiar 10 ani.

1.1.1 Reguli de bază pentru proiectarea circuitelor digitale

- Nişte instrumente de lucru bune nu garantează reuşita proiectării dar oferă certitudinea unui lucru bine executat.
- Circuitele digitale au caracteristici analogice.
- Vă puteţi da seama când ceva nu este în regulă după aspectul analogic al conceptului digital.
- Asociaţi niveluri active denumirilor de semnale şi folosiţi proiectarea cu cerculeţe.
- Este bine să cunoaşteţi şi să folosiţi blocuri structurale funcţionale standardizate.

- Realizaţi proiecte ce implică un minimum de cost la nivelul sistemului, considerând şi munca dumneavoastră de proiectare ca parte a acestui cost.
- Proiectarea unui automat de stări seamănă cu scrierea unui program, abordaţi-o din acest punct de vedere.
- Utilizaţi circuite logice programabile pentru a simplifica proiectele, a reduce costurile şi a opera modificări de ultim moment.
- Evitaţi proiectarea cu circuite asincrone. Până la apariţia unei metodologii mai bune, folosiţi circuitele sincrone.
- În cazurile în care nu puteţi evita folosirea unor circuite asincrone ca interfeţe între diverse module şi echipamente exterioare, acestea trebuie prevăzute cu circuite de sincronizare adecvate.
- O greşeală sesizată la timp vă scuteşte de multă muncă inutilă.

În afară de partea plăcută, de creaţie, a muncii de proiectare şi de munca de rutină mai există şi alte domenii în care un bun proiectant trebuie să fie competent, şi anume:

- *Depanarea.* Este aproape imposibil să fii un bun proiectant fără a fi şi un bun depanator. O depanare reuşită necesită un plan, o abordare sistematică, răbdare şi logică: dacă nu puteţi descoperi unde *se află* defectul, aflaţi unde *nu este*.
- *Activităţi conexe muncii de proiectare.* Munca unui proiectant de circuite digitale include o serie de activităţi care nu au o legătură directă cu ingineria, cum ar fi cunoaşterea standardelor de întocmire a documentaţiei, a posibilităţilor de procurare a componentelor, stabilirea cerinţelor beneficiarilor, elaborarea temelor de proiectare, fixarea termenelor de execuţie, politica de personal şi invitarea furnizorilor la masa de prânz.
- *Evaluarea riscurilor.* Când începeţi un proiect trebuie să apreciaţi atent care sunt riscurile în comparaţie cu eventualele recompense şi consecinţe, din mai multe puncte de vedere, ca, de exemplu, cel al alegerii unor componente nou lansate (vor fi oare pe piaţă la termenul de realizare a prototipului?) sau al fixării termenelor (dacă nu realizez prototipul la termen, voi mai avea oare această slujbă?).
- *Comunicarea.* La un moment dat va trebui să predaţi rezultatul muncii dumneavoastră plină de succes altor ingineri, altor departamente sau beneficiarului. Fără o bună capacitate de comunicare nu veţi putea repurta succese în această etapă. Amintiţi-vă permanent că o bună

comunicare nu este constituită doar din transmiterea de informaţii, ci şi din recepţionarea acestora; învăţaţi să fiţi un bun ascultător!

1.2 Opoziţia analogic – digital

Dispozitivele şi aparatele analogice lucrează cu semnale variabile în timp, care pot lua orice valoare cuprinsă într-un domeniu continuu de tensiune, curent sau altă mărime fizică. Acelaşi lucru fac şi circuitele digitale, însă, spre deosebire de primele, în cazul lor putem pretinde că nu este aşa! Se consideră că un semnal digital poate lua, în orice moment, numai una dintre cele două valori discrete pe care le numim 0 şi 1 (sau L (de la Low=jos) şi H (de la High=sus), sau ADEVĂRAT (TRUE) şi FALS (FALSE), sau negare şi confirmare, sau Sam şi Fred).

Tehnica de calcul digitală a apărut în anii 1940 şi este comercializată pe scară largă de prin 1960. Însă doar în ultimii 10 sau 20 de ani „revoluţia digitală" a pătruns în multe alte domenii. Iată câteva exemple de aparate odinioară analogice, iar în prezent digitale.

1.2.1 Fotografia

În majoritatea aparatelor de fotografiat, imaginile încă se înregistrează pe pelicule acoperite cu halogenuri de argint. Dar creşterea densităţii de date digitale ce pot fi înregistrate pe un cip a condus la realizarea aparatelor de fotografiat digitale, care înregistrează o imagine sub forma de matrice de pixeli de dimensiuni 640x480 sau mai mari (4368x2912 la aparatele de 12,8MegaPixeli), în care fiecare pixel este memorat ca rezultat al compunerii intensităţilor culorilor roşu, verde şi albastru, fiecăreia dintre acestea fiindu-i atribuită o valoare exprimată pe 8 biţi. O asemenea cantitate de informaţie – de peste o sută milioane de biţi, în exemplul nostru – poate fi prelucrată şi comprimată, într-un format JPEG, până la 5% din capacitatea de stocare necesitată iniţial, în funcţie de abundenţa detaliilor din imagine. Prin urmare, aparatele digitale utilizează atât stocarea digitală, cât şi prelucrarea digitală.

1.2.2 Înregistrările video

Pe discurile digitale versatile (DVD) se înregistrează video într-un format digital cu înaltă compresie, denumit MPEG-2. Prin acest procedeu, se comprimă o secţiune din fiecare cadru video de sine stătător într-un mod similar celui utilizat la JPEG, iar fiecare dintre cadrele următoare apare ca diferenţă faţă de cadrul ce îl precede. Capacitatea de stocare a unui DVD cu un singur strat, pe o singură faţă, este de aproximativ 35 de miliarde de biţi, suficientă pentru a oferi cam două ore de înregistrare video de cea mai bună calitate, iar un disc cu două

straturi şi cu două feţe are capacitate de patru ori mai mare. În prezent există şi alte modalităţi de compresie (DivX, Xvid, Mpeg-4) şi stocare a formatelor video (Blue-Ray, HDVD).

1.2.3 Înregistrări audio

Realizate odinioară exclusiv prin imprimarea semnalelor analogice pe discuri de vinil sau pe bandă magnetică, înregistrările audio au în prezent ca suport discurile compacte digitale (CD-uri). Pe un CD, muzica se înregistrează ca o succesiune de numere de 16 biţi ce reprezintă valori ale eşantioanelor obţinute din semnalul analogic original, pentru un canal stereo extrăgându-se câte un eşantion la fiecare 22,7 microsecunde. Un CD înregistrat la capacitatea sa maximă (73 de minute) conţine informaţie de peste şase milioane de biţi. Prin comprimarea acestor informaţii (în format Mpeg-3, WMA) se pot înregistra pe suport DVD circa 64 ore de înregistrări audio la o calitate excepţională (pierderi mai mici de 3%).

1.2.4 Automobile

Anterior, motoarele automobilelor erau comandate în exclusivitate de angrenaje mecanice (dintre care unele ingenioase, „analogice”, care funcţionau ca traductoare de presiune, temperatură etc.), însă astăzi ele au încorporate microprocesoare. Diferiţi senzori electrici şi electro-mecanici convertesc valorile parametrilor caracteristici motorului în valori numerice pe care microprocesorul le examinează şi stabileşte cum trebuie reglate debitele de combustibil şi oxigen ce alimentează motorul, modul în care se face repartiţia cuplului motor pe diversele roţi de tracţiune, posibilitatea schimbării angrenajelor din cutia de viteză, sisteme de avertizare în cazul depăşirii vitezei, a apropierii de antemergător, etc. Semnalul de ieşire al microprocesorului reprezintă o succesiune de valori numerice variabile în timp, care activează nişte subansamble de acţionare electro-mecanice care, la rândul lor, comandă motorul, frânele şi alte dispozitive.

1.2.5 Sistemul de telefonie

A debutat, acum o sută de ani, cu microfoane şi receptoare analogice, legate printr-o pereche de conectoare de cupru. Chiar şi în prezent, în majoritatea locuinţelor se găsesc tot telefoane analogice, care transmit semnale analogice către o centrală telefonică. Însă, în cele mai multe dintre aceste centrale telefonice, semnalele analogice sunt convertite în semnale digitale înainte de a fi transmise către destinaţie, indiferent dacă aparatul care urmează să recepţioneze semnalul se află conectat la aceeaşi centrală telefonică sau în orice alt loc de pe Glob. O perioadă îndelungată, centralele telefonice de

interior, folosite pretutindeni, retransmiteau semnalul digital până la postul de destinaţie. În momentul actual, numeroase centrale de interior, centrale telefonice publice şi companii furnizoare de servicii de telefonie adoptă un sistem integrat, care combină codarea digitală a vocii cu transmisia de date utilizând o singură reţea și un protocolul pentru internet VoIP.

1.2.6 Semafoarele

Semafoarele erau comandate de temporizatoare electro-mecanice, care permiteau aprinderea lămpii verzi şi fiecare indicator de direcţie pentru un anumit interval de timp. Apoi s-au folosit relee conectate în circuite de comandă care aprindeau lămpile semaforului în funcţie de tipul de trafic, determinat cu senzori încorporaţi în carosabil. Circuitele de comandă sunt realizate cu microprocesoare, aprinzând lămpile semafoarelor în aşa fel încât să se obţină un maxim de fluenţă a traficului. În prezent se folosesc sisteme integrate de comandă a semafoarelor, cu post dispecer (unde se realizează şi monitorizarea traficului), acesta furnizând fiecărei intersecţii datele de care are nevoie pentru a realiza fluidizarea traficului (sistemele *undă verde*, prioritizarea mijloacelor de transport public, a vehiculelor de intervenţie, a vehiculelor cu gabarit depăşit sau cu materiale periculoase).

1.2.7 Efecte speciale pentru filmări

Efectele speciale erau realizate mai demult numai folosind miniaturi de argilă, stop-cadre, trucaje fotografice şi numeroase suprapuneri pe peliculă cadru cu cadru. Acum, navele spaţiale, insectele, scenele din alte lumi sunt sintetizate în întregime cu tehnică de calcul digitală.

Revoluţia din electronică are, de acum, o oarecare vechime, iar revoluţia din domeniul aparaturii a început cu dispozitivele analogice şi aparatele realizate cu acestea, de pildă tranzistoarele şi radioreceptoarele tranzistorizate. Dar de ce are loc şi o revoluţie *digitală*? Deoarece circuitele digitale prezintă, într-adevăr, o mulţime de avantaje faţă de cele analogice:

- *Reproductibilitatea rezultatelor*. Pentru o aceeaşi combinaţie de semnale de intrare, (atât ca valori cât şi ca succesiune în timp), un circuit digital bine conceput va furniza întotdeauna acelaşi rezultat. Semnalele de ieşire ale circuitelor analogice variază în funcţie de temperatură, de variaţiile tensiunilor de alimentare, de îmbătrânirea componentelor şi de alţi factori.
- *Simplitatea proiectării.* Proiectarea cu circuite integrate – numite adesea „circuite logice” – este logică. Nu necesită cunoştinţe vaste în domeniul matematicilor, iar comportarea circuitelor logice de bază poate fi urmărită mental, fără efectuarea unor calcule care să descrie

funcţionarea condensatoarelor, a tranzistoarelor sau a altor componente.

- *Flexibilitatea şi aplicabilitatea.* După aducerea în formă digitală, orice problemă poate fi rezolvată printr-o succesiune spaţio-temporală de etape logice. De exemplu, se poate realiza un circuit digital care să distorsioneze o voce înregistrată în aşa măsură încât acesta să fie neinteligibilă pentru orice persoană care nu deţine o parolă, dar să se audă nedistorsionat când se introduce acea parolă.
- *Programabilitatea.* Proiectarea a numeroase circuite digitale se face acum scriind programe în *limbaje descriptoare de echipamente* – HDL (*hardware description languages*) sau în software-uri cu o interfaţă care prezintă în clar structura componentelor folosite. Astfel de limbaje şi software-uri permit descrierea sau *modelarea* atât a configuraţiei, cât şi a funcţiei unui circuit digital. Pe lângă compilator, acestea includ programe de simulare şi de sintetizare. Aceste instrumente virtuale servesc la testarea modelului de echipament înainte de realizarea practică a acestuia, urmată de sintetizarea modelului într-un circuit conform unei anumite tehnologii de fabricaţie a componentelor.
- *Viteza.* Dispozitivele digitale actuale sunt foarte rapide. Fiecare tranzistor din cele mai rapide integrate poate comuta în mai puţin de 4 picosecunde, iar un întreg dispozitiv complex, realizat cu asemenea tranzistoare, poate să exploreze intrările şi să furnizeze un semnal de ieşire într-un interval de timp mai scurt de 0,3 nanosecunde. Înseamnă că astfel de dispozitive pot genera 3 miliarde sau chiar mai multe rezultate pe secundă.
- *Costul scăzut.* Circuitele digitale efectuează numeroase operaţii şi ocupă un spaţiu redus. Circuitele repetate se pot „integra” pe un singur „cip” şi se pot fabrica în masă cu costuri foarte scăzute, fapt pentru care obiecte cum sunt calculatoarele, ceasurile digitale şi felicitările muzicale se aruncă fără regrete.
- *Progresul tehnologic continuu*. În proiectarea unui aparat digital există puţine limitări. Peste câţiva ani va apărea unul mai rapid, mai ieftin sau superior din alt punct de vedere. Un proiectant bun ţine cont de aceasta la realizarea aparatului iniţial, luând măsuri de prevenire a demodării lui şi făcându-l astfel mai apreciat de utilizatori. De exemplu, calculatoarele de birou sunt prevăzute adesea cu „socluri de expansiune”, pentru ca în ele să poată fi introduse procesoare mai rapide sau circuite de memorie de capacitate mai mare decât cele disponibile în momentul realizării calculatorului.

1.3 Dispozitive digitale

Cele mai simple dispozitive digitale se numesc porţi (în limba engleză – gates). Ele au fost denumite astfel după funcţia pe care o îndeplinesc, de a permite sau a întârzia transmiterea informaţiei digitale. În general, o poartă are una sau mai multe intrări şi furnizează la ieşire un semnal care este funcţie de valorile curente ale intrărilor. Intrările şi ieşirile pot fi diverse mărimi fizice ca, de exemplu, tensiunea, intensitatea curentului, chiar şi presiunea hidraulică, însă reprezentarea acestora poate lua două valori discrete, 0 şi 1.

În Figura 1-1 sunt prezentate simbolurile celor trei tipuri principale de porţi. Poarta AND (ŞI) cu două intrări, notată (a), are ieşirea 1 când ambele intrări sunt 1; în toate celelalte cazuri, ieşirea sa este 0. În figură apare aceeaşi poartă de patru ori, cu cele patru combinaţii de intrare posibile şi ieşirile respective. Porţile se mai numesc *circuite combinaţionale* deoarece ieşirea depinde numai de combinaţia de intrare curentă.

O poartă OR cu două intrări, ca în Figura 1-1 (b), are ieşirea 1 dacă o intrare sau ambele intrări sunt 1; ieşirea sa este 0 numai dacă ambele intrări sunt 0. Şi în acest caz sunt posibile patru combinaţii de intrare, ieşirile corespunzătoare fiecăreia fiind cele din figură.

O poartă NOT, denumită de cele mai multe ori *inversor*, are la ieşire valoarea opusă celei de la intrare, cum se observă în Figura 1-1 (c).

Aceste trei tipuri de porţi sunt cele mai importante din mai multe motive. Orice funcţie digitală poate fi realizată utilizând exclusiv aceste trei porţi. În capitolul 3 veţi vedea cum se realizează porţile din circuite cu tranzistoare. Trebuie să ştiţi că au fost construite porţi sau s-a propus construirea acestora şi cu alte tehnologii, de exemplu cu relee, cu tuburi cu vid, pe principii ale hidraulicii şi cu structură moleculară.

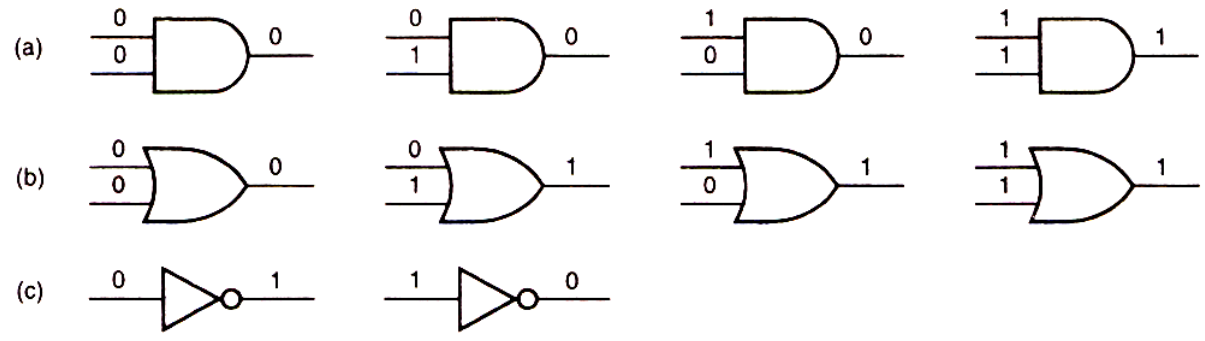


Figura 1-1 Dispozitive digitale: (a) poartă AND; (b) poartă OR; (c) poartă NOT sau inversoare

Un *bistabil* este un dispozitiv care poate să aibă ieşirea fie în 0, fie în 1. *Starea* unui bistabil este valoarea curentă stocată. Aceasta poate fi modificată numai la anumite momente de timp impuse de o intrare de tact sau de „ceas”,

noua valoare depinzând de starea curentă a bistabilului şi de intrările sale de „comandă”. Un bistabil se poate realiza din mai multe porţi conectate judicios.

Un circuit digital ce conţine bistabile se numeşte *circuit secvenţial*, deoarece, în orice moment, semnalul de la ieşirea sa depinde nu numai de valorile intrărilor, dar şi de succesiunea sau secvenţa de semnale de intrare existentă anterior. Cu alte cuvinte, un circuit secvenţial *memorează* evenimentele anterioare.

1.4 Electronica aferentă proiectării circuitelor digitale

Aşa cum vom vedea în capitolul 3, circuitele digitale lucrează cu tensiuni şi curenţi de natură analogică şi sunt realizate din componente analogice. „Abstractizarea digitală” permite ca natura lor analogică să fie ignorată în majoritatea cazurilor, deci modelele de circuite pot fi considerate ca lucrând practic cu valori de 0 şi 1.

Unul dintre aspectele importante ale abstractizării digitale este asocierea unui *domeniu* de valori analogice fiecărei valori logice (0 sau 1). Aşa cum arată Figura 1-2, o poartă obişnuită nu prezintă un nivel de tensiune garantat cu mare precizie, corespunzător valorii logice 0. Practic, ea poate genera o tensiune cuprinsă într-un *subdomeniu* al domeniului de recunoaştere ca nivel 0 la intrările altor porţi, care este garantată. Intervalul dintre extremităţile domeniilor se numeşte *margine de zgomot*; în cazul unui circuit real, aceasta reprezintă limita până la care se poate suprapune zgomotul fără a afecta interpretarea corectă a semnalului de ieşire ca nivel logic la intrările altor porţi.

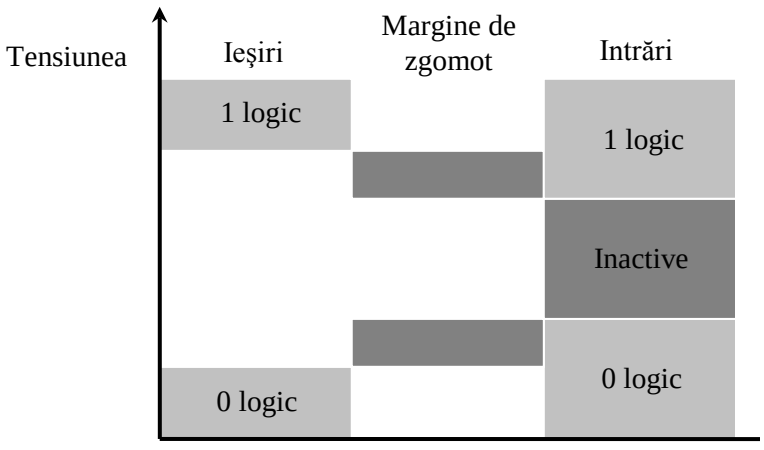


Figura 1-2 Valorile logice si marginea de zgomot

Comportarea este similară în cazul ieşirilor în starea logică 1. În figură se observă că între domeniile de intrare corespunzătoare valorilor logice 0 şi 1 apare o regiune „inactivă”. Deşi, la un dispozitiv digital oarecare, în condiţii precizate de tensiune şi temperatură, diferenţa (sau pragul) dintre cele două

domenii prezintă o delimitare destul de netă, la dispozitive diferite, limitele pot fi diferite. Cu toate acestea, la dispozitivele care funcţionează corect, limitele respective se regăsesc *undeva* în interiorul domeniului de „inactivitate”. În concluzie, orice semnal aflat în interiorul domeniilor de definiţie a valorilor logice 0, respectiv 1, va fi interpretat în mod identic de dispozitive diferite. Această caracteristică este esenţială din punct de vedere al reproductibilităţii rezultatelor.

Garantarea faptului că porţile logice generează şi recunosc semnale logice cuprinse în domeniile corespunzătoare este una dintre atribuţiile proiectantului de circuite *electronice*. Aceasta este o problemă de proiectare cu circuite analogice; vom menţiona câteva aspecte legate de ea în capitolul 3. Este imposibil de conceput un circuit care să se comporte aşa cum dorim în orice condiţii de alimentare cu tensiune, temperatură, sarcină şi alţi factori. De aceea, proiectantul de circuite electronice sau producătorul de circuite specifică valorile limită ce definesc condiţiile în care se garantează funcţionarea corectă.

Un proiectant de circuite *digitale*, nu trebuie să pătrundă prea adânc în detaliile comportării analogice a dispozitivelor digitale ca să fie sigur că acestea funcţionează corect. Tot ceea ce are de făcut este să verifice dacă dispozitivul funcţionează în condiţiile impuse de valorile limită. Desigur, sunt necesare unele cunoştinţe de electronică analogică pentru a face verificarea, dar nu în asemenea măsură ca şi cum s-ar apuca să proiecteze un circuit digital pornind de la zero.

1.5 Circuite integrate

Un ansamblu realizat pe acelaşi cip de siliciu poartă numele de *circuit integrat* (CI). CI de dimensiuni mari, care conţin zeci de milioane de tranzistoare, pot avea una dintre laturi puţin mai mare decât 1 cm, dar poate fi sub 2,5 mm.

Indiferent de mărime, orice CI se construieşte iniţial pe o *plachetă* circulară de dimesiuni mult mai mari, cu diametrul de 25 cm, care conţine de la câteva zeci la câteva sute de circuite de acelaşi tip. Toate cipurile de CI de pe plachetă se construiesc simultan şi fiecare bucată (cip de CI) se numeşte *pastilă*. După încheierea procesului tehnologic aplicat plachetei, pastilele sunt testate fără a fi detaşate din aceasta, iar cele defecte se marchează. Apoi se trece la decuparea plachetei în pastile separate, iar cele marcate se îndepărtează. Fiecare pastilă nemarcată se montează în câte o capsulă, se realizează conexiunile între terminalele pastilei şi pinii capsulei, apoi CI încapsulat este supus unei testări finale.

Denumirile date structurilor prezentate mai sus nu sunt utilizate totdeauna cu consecvenţă. Uneori „CI” desemnează pastila de siliciu,. Alteori, aceasta este denumită cip. Câteodată, prin „CI” sau „cip” vorbitorul înţelege ansamblul pastilă de siliciu – capsulă. Proiectanţii de circuite digitale folosesc alternativ

ambii termeni şi nu îşi fac deloc probleme din această cauză. Nu simt nevoia unei definiţii riguroase deoarece sunt interesaţi numai de funcţionarea şi caracteristicile electrice ale obiectelor în discuţie. Pentru consecvenţa exprimării, pe parcursul cursului vom folosi denumirea CI pentru a desemna pastila încapsulată.

Prima clasificare a CI s-a făcut după dimensiuni - mici, medii şi mari -, în funcţie de numărul de porţi conţinute. Cele mai simple tipuri de CI care se găsesc pe piaţă sunt încă denumite *cu integrare la scară mică* (SSI – small-scale integration) şi conţin echivalentul a 1 până la 20 de porţi. CI de tip SSI conţin de obicei, un număr mic de porţi sau bistabile – elemente structurale de bază ale circuitelor digitale.

CI de tip SSI se prezintă în capsule DIP (*dual in-line pin* – perechi de pini aliniaţi) cu 14 pini. După cum observaţi în Figura 1-3, capsulele DIP mai mari sunt adaptate pentru funcţii ce necesită un număr mai mare de pini. *Diagrama pinilor* prezintă corespondenţa dintre semnalele proprii dispozitivului şi pinii capsulei. În Figura 1-4 sunt prezentate diagramele pinilor pentru câteva CI SSI uzuale. Asemenea diagrame sunt utile numai pentru asamblarea mecanică, în cazul în care proiectantul doreşte să cunoască numerotarea pinilor unui anumit CI. În schema de principiu a unui circuit nu apare numerotarea pinilor, porţile fiind însă grupate după funcţia pe care o îndeplinesc.

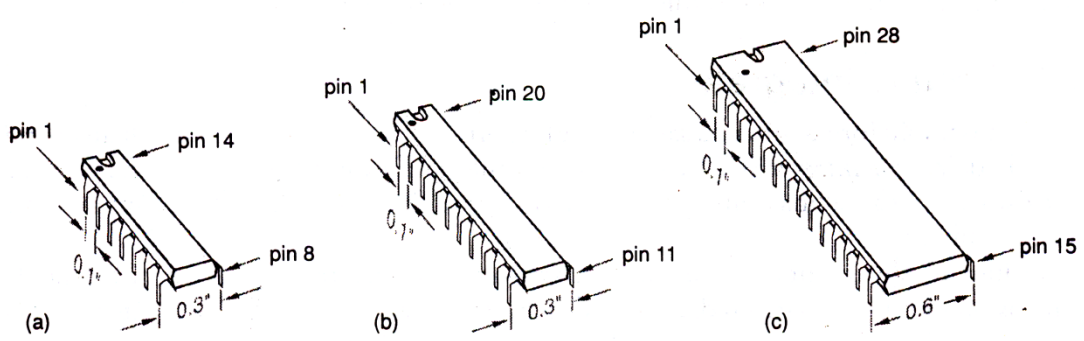


Figura 1-3 Capsule DIP (dual in-line pin): (a) cu 14 pini; (b) cu 20 de pini; (c) cu 28 de pini

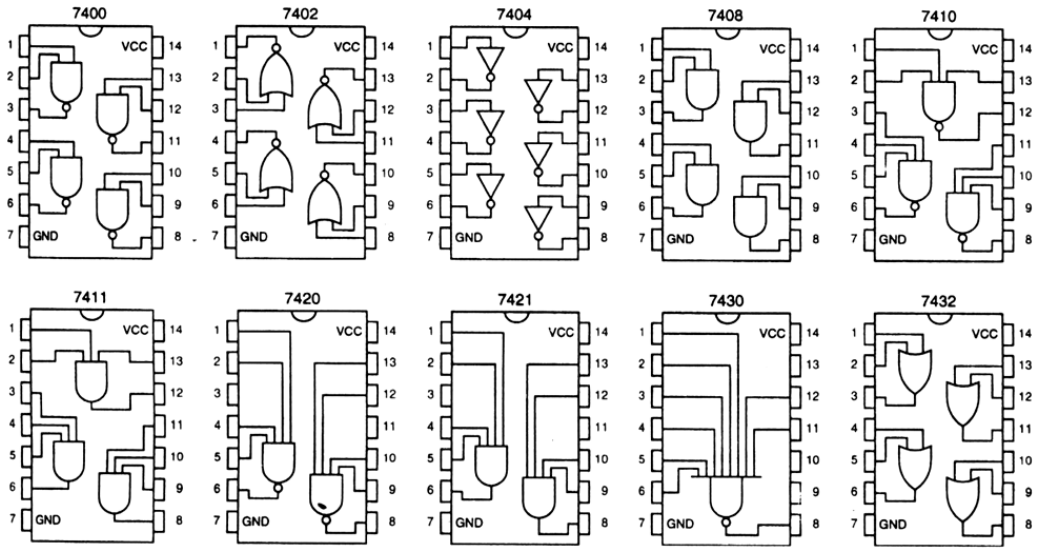


Figura 1-4 Diagrame de pini pentru CI SSI din seria 7400

CI SSI sunt încă folosite ca elemente de legătură, asigurând conectarea elementelor integrate la scară largă, însă ele au fost detronate într-o mare măsură de dispozitivele logice programabile (PLD).

Cea de-a doua categorie de CI de pe piaţă este numită *integrare la scară medie* (MSI – *medium-scale integration*), conţinând, cu aproximaţie, echivalentul a 20 până la 200 de porţi. Un CI MSI conţine, de obicei, un bloc funcţional structural; de exemplu un decodor, un registru de memorie, un numărător etc. Deşi CI MSI discrete sunt din ce în ce mai puţin întrebuinţate, echivalentele blocurilor structurale amintite sunt larg utilizate la proiectarea unor CI de dimensiuni mari.

CI *cu integrare la scară largă* (LSI – *large-scale integration*) sunt şi mai cuprinzătoare, conţinând echivalentul a 200 până la 200.000 de porţi sau chiar mai mult. Componente LSI sunt circuitele de memorie de capacitate mică, microprocesoarele, dispozitivele logice programabile şi cele cu destinaţie specială.

Graniţa dintre LSI şi integrarea la scară foarte largă (VLSI – very large-scale integration) este confuză şi se stabileşte în funcţie de numărul de tranzistoare, nu după numărul de porţi. Orice CI cu peste 1.000.000 de tranzistoare este considerat, fără discuţie, VLSI. Printre acestea se numără majoritatea microprocesoarelor şi circuitelor de memorie actuale, dispozitivele logice programabile de mari dimensiuni şi cele cu destinaţie specială. Au fost realizate CI VLSI cu 20 de miliarde de tranzistoare (vezi circuitele de memorie flash de 2GByte), iar numărul acestora a crescut.

1.5.1 Dispozitive logice programabile

Există o gamă largă de CI ale căror funcţii logice pot fi *programate* în interiorul lor după încheierea procesului de fabricare. Majoritatea acestor

dispozitive folosesc tehnologii ce permit reprogramarea funcţiilor, ceea ce înseamnă că, dacă descoperiţi o eroare de proiectare o puteţi corecta fără a înlocui dispozitivul sau fără a modifica fizic conexiunile.

Cronologic, *matricele logice programabile (PLA – programmable logic array)* au fost primele dispozitive logice programabile. Structura PLA constă în două nivele de porţi AND şi OR, ale căror conexiuni se programează de către utilizator. Folosind această structură, proiectantul poate realiza orice funcţie logică până la un anumit nivel de complexitate aplicând teoria sintezei şi minimizării circuitelor logice.

Structura PLA a fost îmbunătăţită şi costurile ei au fost reduse în urma apariţiei *dispozitivelor logice matriceale programabile (PAL – programmable array logic device)*. La ora actuală, ele sunt denumite generic *dispozitive logice programbile (PLD – programmable logic device)* şi constituie varianta MSI a producţiei de circuite logice programabile.

Capacitatea în continuă creştere a circuitelor integrate a dus la conceperea unor arhitecturi cu PLD *complexe (CPLD)*. Un CPLD uzual este un ansamblu format din mai multe PLD de sine stătătoare şi o structură de interconectare realizate pe acelaşi cip. În plus faţă de PLD de sine stătătoare, structura de interconectare realizată pe cip este şi ea programabilă, oferind o gamă largă de variante de circuite. CPLD pot ajunge la dimensiunile dorite prin mărirea numărului de PLD de sine stătătoare şi prin dezvoltarea structurii de interconectare de pe cipul de CPLD.

Cam în aceeaşi perioadă în care au fost inventate CPLD, alţi producători de CI au abordat în mod diferit problema extinderii dimensiunilor cipurilor logice programabile. Comparativ cu CPLD, *matricele de porţi programabile în câmp (FPGA – field-programmable gate array)* conţin un număr mult mai mare de mici blocuri logice de sine stătătoare şi dispun de o structură extinsă, ramificată, de interconectare, predominantă la nivelul întregului cip. Figura 1-5 ilustrează deosebirea dintre cele două concepţii de realizare.

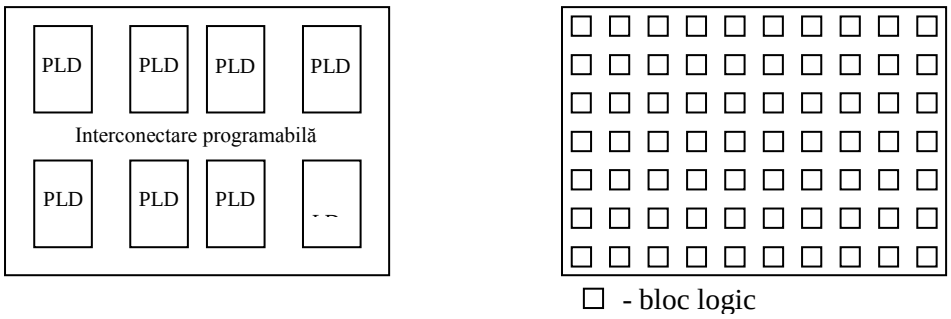


Figura 1-5 Concepţii de realizare a dispozitivelor logice de mari dimensiuni: (a) CPLD; (b) FPGA

1.5.2 CI dedicate unor aplicații

Cipurile realizate special pentru un anumit produs, cu aplicabilitate limitată, sunt denumite *CI semidedicate* sau *CI specifice unei aplicaţii (ASIC – application-specific IC)*. Prin folosirea cipurilor ASIC se reduce, în general, costul total aferent componentelor şi manoperei, pentru un anumit produs, prin reducerea numărului de cipuri, a dimensiunilor constructive şi a consumului de energie, toate acestea contribuind adesea şi la obţinerea unor performanţe superioare.

Costul proiectării de unicat pentru un ASIC îl poate depăşi pe cel aferent proiectării cu componente discrete cu 5.000 până la 250.000 de dolari sau chiar cu mai mult.

Exemplu : microcontrolerele (pentru prelucrarea semnalelor audio, video, radio sau radar, pentru prelucrarea unor informaţii în cadrul fluxurilor de date).

1.5.3 Cartele de circuit imprimat (cablaje)

CI se montează, de obicei, pe cartele de circuit imprimat (sau cablaj), prin care se realizează conectarea lor cu celelalte CI din sistem. Cartelele multistratificate, utilizate, în general, în sistemele cu circuite digitale, sunt prevăzute cu trasee de cupru obţinute prin corodare, dispuse pe mai multe straturi subţiri din fibră de sticlă, care se lipesc apoi, formând o placă cu grosimea de aproximativ 1,5 mm.

La ora actuală, pentru majoritatea componentelor se utilizează tehnologia de montare pe suprafaţă (SMT- surface-mount technology).

Tehnologia de montare a componentelor pe suprafaţă, asociată cu tehnologia de realizare a traseelor subţiri, permite obţinerea unei densităţi extrem de mari la montarea circuitelor integrate şi a altor componente pe cartelele de circuit imprimat. Astfel se obţine mai mult decât o economie de spaţiu. În cazul circuitelor de mare viteză, prin dispunerea densă a componentelor se înlătură, în mare măsură, efectele nedorite de natură analogică, cum sunt cele caracteristice liniilor de transmisie şi limitările la viteza luminii.

Pentru a satisface cele mai stricte cerinţe în ceea ce priveşte viteza şi densitatea au fost realizate *module multicip (MCM – multichip module)*. Cu această tehnologie, pastilele de CI nu se mai montează în capsule individuale de ceramică sau plastic, ci, în cazul subsistemelor de mare viteză (să zicem, un procesor şi memoria cache aferentă lui), ele se montează direct pe un substrat ce conţine interconexiunile necesare, stratificate. MCM se etanşează apoi şi se prevede cu pini de alimentare şi de masă şi pini ce corespund exclusiv semnalelor prin care subsistemul în cauză interacţionează cu sistemul din care face parte.

Obiectivele CI sunt, aşadar:

- reducerea costurilor de realizare;
- reducerea numărului de CI;
- reducerea suprafeţei cablajului;
- reducerea timpului de proiectare;
- posibilitatea de producţie în masă fără abateri foarte mari de la produsul original.

2. Sisteme de numerație și coduri numerice

Aparatele digitale sunt construite din circuite care lucrează cu cifre binare (0 şi 1). De aceea, proiectantul de aparatură digitală trebuie să stabilească un fel de corespondenţă între numerele digitale – care pot fi prelucrate de circuitele digitale – şi numerele, evenimentele şi condiţiile din viaţa reală.

Sistemul tradiţional de numeraţie studiat în şcoală şi de uz curent în contabilitate este numit *sistem de numeraţie poziţional*. Într-un astfel de sistem, un număr se reprezintă printr-un şir de cifre în care fiecare dintre poziţiile cifrelor are o anumită *pondere*. Valoarea unui număr este suma ponderată a cifrelor sale.

De exemplu:

$$1743=1\cdot1000 + 7\cdot100 + 4\cdot10 + 3\cdot1$$

Ponderea fiecărei poziţii este egală cu 10 la puterea dată de numărul de ordine al poziţiei respective. Cifra cea mai din stânga este *cifra de cel mai mare ordin* sau *cifra cea mai semnificativă*; cifra cea mai din dreapta este *cifra de cel mai mic ordin* sau *cifra cea mai puţin semnificativă*.

În circuitele digitale, semnalele pot avea, în mod normal, una din singurele două stări posibile: de jos sau de sus, cu sarcină sau fără sarcină, oprit sau pornit. Astfel de semnale sunt interpretate ca reprezentând cifre binare (sau biţi), ale căror valori posibile sunt 0 şi 1. Aceasta este cauza pentru care s-a ales *baza de numeraţie 2* (sau *binară*) pentru reprezentarea numerelor în sistemele digitale.

2.1 Conversia unui număr din baza zece într-o bază B

Fie numărul N în baza 10 şi baza B, oarecare, în care dorim să transformăm numărul N.

Numărul N în baza zece se poate scrie:

$$N = a_n\cdot10^n + a_{n-1}\cdot10^{n-1} + \ldots.. + a\cdot10^1 + a_0\cdot10^0$$

Coeficienţii a_n, a_{n-1},….., a_1, a_0 reprezintă cifrele reprezentării în baza 10, în această ordine.

În baza B, numărul N va fi:

$$N = b_n\cdot B^n + b_{n-1}\cdot B^{n-1} + \ldots.. + b_1\cdot B^1 + b_0\cdot B^0$$

Coeficienţii b_n, b_{n-1},….., b_1, b_0 reprezintă cifrele reprezentării în baza oarecare B, în această ordine.

Conversia unui număr din baza zece într-o bază oarecare, B se face după următorul algoritm:

- se efectuează împărţirea lui N la B, obţinându-se restul b_0;
- câtul acestei împărţiri se împarte şi el la B, obţinându-se un nou rest, b_1;
- se repetă pasul al doilea până la obţinerea câtului 0.
- Resturile obţinute, scrise în ordine inversă (începând cu ultimul) reprezintă numărul N scris în baza B.

Exemplu:
Numărul zecimal 13 scris în baza 3 se va obţine astfel:

13 : 3 = 4 rest 1
4 : 3 = 1 rest 1
1 : 3 = 0 rest 1

deci: $13_3 = 1 \cdot 3^2 + 1 \cdot 3^1 + 1 \cdot 3^0$

2.2 Numere binare, octale şi hexazecimale

Baza de numeraţie 10 prezintă importanţă pentru că este cea de uz curent în activitatea cotidiană, iar importanţa bazei 2 se datorează faptului că numerele binare pot fi prelucrate direct de circuitele digitale.

Numerele din alte baze pot fi rareori prelucrate direct, însă au o anumită importanţă în scop documentar şi în diverse alte scopuri. În particular, bazele 8 şi 16 constituie sisteme convenabile de reprezentare rapidă a numerelor de mai mulţi biţi într-un sistem digital.

Sistemul de numeraţie octal are baza 8, iar sistemul de numeraţie hexazecimal are baza 16.

Tabelul 2-1 prezintă numerele zecimale de la 0 la 15 şi echivalentele lor în sistem binar, octal şi hexazecimal. Sistemul octal utilizează 8 cifre (0, 1, …, 7) din sistemul zecimal, iar cel hexazecimal necesită 16 cifre, deci cifrelor zecimale 0, …, 9 li se adaugă literele A, …, F.

Sistemele de numeraţie octal şi hexazecimal sunt adecvate reprezentării numerelor de mai mulţi biţi deoarece bazele lor sunt puteri ale lui 2. Întrucât cu un şir de 3 biţi se pot realiza opt combinaţii diferite, înseamnă că fiecare şir de trei biţi are o reprezentare unică printr-o cifră octală, aşa cum arată coloana a

treia şi a patra din tabel. Similar, un şir de 4 biţi poate fi reprezentat printr-o cifră hexazecimală, conform coloanelor a cincea şi a şasea din tabel.
Exemplu:

$100011001110_2 = 100\ 011\ 001\ 110_2 = 4316_8$
$11101101110101001_2 = 011\ 101\ 101\ 110\ 101\ 001_2 = 355651_8$

$100011001110_2 = 1000\ 1100\ 1110_2 = 8CE_{16}$
$11101101110101001_2 = 0001\ 1101\ 1011\ 1010\ 1001_2 = 1DBA9_{16}$

În exemplele de mai sus au fost adăugate în stânga numerelor atâtea zerouri câte au fost necesare pentru ca numărul total de biţi să fie multiplu de 3 sau 4, după caz.

În cazul numerelor scrise în baza 2, bitul cel mai din stânga este numit bitul de cel mai mare ordin sau bitul cel mai semnificativ (MSB – most significant bit); bitul cel mai din dreapta este bitul de cel mai mic ordin sau bitul cel mai puţin semnificativ (LSB – least significant bit).

Tabel 2-1 Numere zecimale, binare, octale şi hexazecimale.

Zecimal	Binar	Octal	Şir de 3 biţi	Hexazecimal	Şir de 4 biţi
0	0	0	000	0	0000
1	1	1	001	1	0001
2	10	2	010	2	0010
3	11	3	011	3	0011
4	100	4	100	4	0100
5	101	5	101	5	0101
6	110	6	110	6	0110
7	111	7	111	7	0111
8	1000	10	-	8	1000
9	1001	11	-	9	1001
10	1010	12	-	A	1010
11	1011	13	-	B	1011
12	1100	14	-	C	1100
13	1101	15	-	D	1101
14	1110	16	-	E	1110
15	1111	17	-	F	1111

2.3 Adunarea şi scăderea numerelor nezecimale

Pentru a aduna două numere binare, X şi Y, se adună biţii cei mai puţin semnificativi cu transportul iniţial (c_{in}) de 0, rezultând biţii de transport (c_{out}) şi sumă (s). Se aplică acelaşi procedeu tuturor biţilor pe rând, pornind de la dreapta şi adăugând transportul provenit de pe fiecare coloană la suma coloanei următoare.
Exemplu:

X = 190
Y = 141
X + Y = 331

C 1 0 1 1 1 1 0 0 0
X 1 0 1 1 1 1 1 0 +
Y 1 0 0 0 1 1 0 1
X+Y 1 0 1 0 0 1 0 1 1

X = 173
Y = 44
X + Y = 217

C 0 0 1 0 1 1 0 0 0
X 1 0 1 0 1 1 0 1 +
Y 0 0 1 0 1 1 0 0
X+Y 1 1 0 1 1 0 0 1

Scăderea se efectuează în binar în mod similar, transportul între coloane fiind înlocuit de împrumut (b_{in} şi b_{out}), iar rezultatul este un bit diferenţă, d.
Exemplu:

X = 229
Y = 46
X - Y = 183

B 0 0 1 1 1 1 1 0 0
X 1 1 1 0 0 1 0 1 –
Y 0 0 1 0 1 1 1 0
X-Y 1 0 1 1 0 1 1 1

X = 210
Y = 109
X - Y = 101

B 0 1 1 0 1 1 0 1 0
X 1 1 0 1 0 0 1 0 –
Y 0 1 1 0 1 1 0 1
X-Y 0 1 1 0 0 1 0 1

Adunarea şi scăderea în octal şi hexazecimal se face fie direct, fie prin transformarea operanzilor în binar.
Exemplu:

X = 9FAH
Y = 7B2H
(sufixul H se adaugă la numerele în hexazecimal în locul indicelui 16)

$X = 1001\ 1111\ 1010_2$
$Y = 0111\ 1011\ 0010_2$

1 0 0 1 1 1 1 1 1 0 1 0 +
0 1 1 1 1 0 1 1 0 0 1 0
1 0 0 0 1 1 0 1 0 1 1 0 0 = 1 0001 1010 1100_2 = 11ACH

şi direct:

1 1 (propagarea transportului)
↖↖
9 F A +
7 B 2
1 1 A C

Se procedează asemănător şi la scădere:

X = B21H
Y = 14H

B 2 1 –
1 4
B 0 D

X = 767_8
Y = 665_8

1 1 1
↖↖↖
$7\,6\,7_8$ +
$6\,6\,5_8$
$1\,6\,5\,4_8$

2.4 Reprezentarea numerelor fracţionare

Reprezentarea are două părţi: partea întreagă şi partea fracţionară.
Pentru partea fracţionară ponderea va fi negativă.

$$\mathbf{D} = \sum_{i=-n}^{p-1} d_i \cdot r^i$$

$\mathbf{D = d_{p-1} d_{p-2} \ldots d_1 d_0, d_{-1} d_{-2} \ldots d_{-n}}$

Exemplu:

$100,111_2 = 1\cdot2^2 + 0\cdot2^1 + 0\cdot2^0 + 1\cdot2^{-1} + 1\cdot2^{-2} + 1\cdot2^{-3} = 4 + 0,5 + 0,25 + 0,125 = 4,875_{10}$

$632,24_{10} = ?_2$

632 = 1001111000

$0,24 = 0\cdot2^{-1} + 0\cdot2^{-2} + 1\cdot2^{-3} + 1\cdot2^{-4} + 1\cdot2^{-5} + 1\cdot2^{-6}$

Rezultă: $632,24_{10} = 1001111000,001111_2$

2.5 Reprezentarea numerelor negative

În activitatea cotidiană se foloseşte sistemul de reprezentare a numerelor prin modul şi semn. Cu toate acestea, în majoritatea calculatoarelor se utilizează unul dintre sistemele de reprezentare prin complement a unui număr.

2.5.1 Reprezentarea prin modul şi semn

În sistemul de reprezentare prin modul şi semn, un număr apare sub forma unei valori precedate de un semn care arată dacă acea valoare este pozitivă sau negativă. Prin urmare, ştim cum să interpretăm numerele zecimale +98, -57, +123,5 şi -13 şi mai ştim că dacă un număr nu este precedat de nici un semn, este un număr pozitiv. Pentru zero sunt posibile două reprezentări: +0 şi -0, însă ambele au aceeaşi valoare.

Sistemul de reprezentare prin modul şi semn se aplică numerelor binare utilizând un bit excedentar, numit *bit de semn*. După tradiţie, bitul cel mai din stânga (MSB) dintr-un şir de biţi este considerat bit de semn (0 = plus, 1 = minus), iar biţii următori reprezintă modulul.

Exemplu:

$01010101_2 = +85_{10}$ $\quad$ $11010101_2 = -85_{10}$

$01111111_2 = +127_{10}$ $\quad$ $11111111_2 = -127_{10}$

$00000000_2 = +0_{10}$ $\quad$ $10000000_2 = -0_{10}$

Un întreg de *n* biţi, reprezentat prin modul şi semn, aparţine domeniul cuprins între $-(2^{n-1}-1)$ şi $(2^{n-1}-1)$ cu cele două reprezentări posibile pentru zero.

Să presupunem că se vrea realizarea un circuit logic digital care să efectueze adunarea unor numere reprezentate prin modul şi semn. Circuitul trebuie să examineze semnele termenilor pentru a putea decide cum să prelucreze modulele. Dacă semnele sunt identice, modulele trebuie adunate, iar rezultatul primeşte acelaşi semn. Dacă semnele sunt diferite, modulele trebuie comparate, cel mai mic se scade din cel mai mare, iar rezultatul primeşte semnul termenului cu modulul mai mare. Este mult mai simplu ca adunarea să se efectueze folosind reprezentarea numerelor prin complement. Poate că singurul avantaj al sistemului de reprezentare prin modul şi semn este acela că, o dată ce am construit un circuit de adunare, este cât se poate de simplu să construim şi

unul de scădere; este nevoie doar de schimbarea semnului scăzătorului şi de adunarea lui la descăzut folosind circuitul de adunare.

2.5.2 Sisteme de reprezentare a unui număr prin complementul lui

În timp ce, în sistemul de reprezentare prin modul şi semn, un număr se neagă prin schimbarea semnului, în sistemul de reprezentare a unui număr prin complementul lui, numărul se neagă prin calcularea complementului său conform sistemului care defineşte complementul. Calcularea complementului este o operaţie mai dificilă decât schimbarea semnului, însă două numere reprezentate prin complement pot fi adunate sau scăzute direct, fără a mai fi nevoie de explorarea semnelor şi modulelor, ca în sistemul de reprezentare respectiv.

Reprezentarea prin complement faţă de bază

În sistemul de reprezentare prin complement faţă de bază, complementul unui număr de n cifre se obţine scăzând numărul din r^n. Dacă numărul (fie el D = $d_{n-1}d_{n-2}\ldots d_1d_0$) aparţine intervalului dintre 1 şi r^n-1, prin scădere rezultă un alt număr cuprins între 1 şi r^n-1. Dacă D este 0, rezultatul scăderii este r^n, care are forma 100…00, numărul total de cifre fiind n+1. Dacă înlăturăm cifra de rang superior prin care se depăşeşte numărul n de cifre, obţinem 0. Prin urmare, reprezentarea lui zero într-un sistem de reprezentare prin complement faţă de bază este unică.

Din definiţie, se pare că pentru calcularea complementului numărului D faţă de bază este necesară efectuarea unei scăderi. Scăderea poate fi însă evitată dacă scriem r^n sub forma (r^n-1)+1, iar r^n-D sub forma ((r^n-1)-D)+1. De exemplu, 10000 este egal cu 9999+1. Dacă definim complementul cifrei d ca fiind r-1-d, atunci (r^n-1)-D se obţine prin calcularea complementelor cifrelor numărului D. Prin urmare, complementul faţă de bază al unui număr D se obţine prin calcularea tuturor complementelor cifrelor numărului D, la care se adaugă 1. De exemplu, complementul faţă de 10 al numărului 1849 este 8150 + 1 sau 8151.

Tabelul 2-2 prezintă complementele faţă de bază ale cifrelor din sistemele de numeraţie binar, octal, zecimal şi hexazecimal.

Tabel 2-2 Complementele cifrelor

Cifra	Binar	Octal	Zecimal	Hexazecimal
0	1	7	9	F
1	0	6	8	E
2	-	5	7	D
3	-	4	6	C
4	-	3	5	B
5	-	2	4	A

6	-	1	3	9
7	-	0	2	8
8	-	-	1	7
9	-	-	0	6
A	-	-	-	5
B	-	-	-	4
C	-	-	-	3
D	-	-	-	2
E	-	-	-	1
F	-	-	-	0

Reprezentarea prin complement faţă de 2

În cazul numerelor binare, complementul faţă de bază este complementul faţă de 2. În sistemul binar, MSB al unui număr serveşte ca bit de semn; un număr este negativ dacă şi numai dacă are MSB egal cu 1. Echivalentul zecimal al complementului faţă de 2 al unui număr binar se calculează la fel ca pentru numerele care nu sunt precedate de semn, cu excepţia faptului că ponderea MSB este aici de -2^{n-1}, în loc de $+2^{n-1}$. Domeniul căruia îi aparţin reprezentările numerelor este cuprins între $-(-2^{n-1})$ şi $+(-2^{n-1}-1)$. Iată, în continuare, câteva exemple pentru numere de 8 biţi:

$17_{10} = 00010001_2 \rightarrow 11101110 + 1 = 11101111_2 = -17_{10}$

$-99_{10} = 10011101_2 \rightarrow 01100010 + 1 = 01100011_2 = 99_{10}$

$119_{10} = 01110111_2 \rightarrow 10001000 + 1 = 10001001_2 = -119_{10}$

$-127_{10} = 10000001_2 \rightarrow 01111110 + 1 = 01111111_2 = 127_{10}$

$0_{10} = 00000000_2 \rightarrow 11111111 + 1 = \underline{1}\,00000000_2 = 0_{10}$

$-128_{10} = 10000000_2 \rightarrow 01111111 + 1 = 10000000_2 = -128_{10}$

Un transport provenit din poziţia MSB apare într-un singur caz, marcat mai sus printr-o cifră subliniată. Ca în toate operaţiile cu complement faţă de 2, bitul acela se ignoră, folosindu-se doar cei *n* biţi de ordin inferior ai rezultatului.

În sistemul de reprezentare a numerelor prin complement faţă de 2, zero este considerat pozitiv deoarece bitul său de semn este 0. Întrucât în reprezentarea prin complementul faţă de 2, zero are o reprezentare unică, apare un număr negativ suplimentar, -2^{n-1}, care nu are un simetric pozitiv.

Putem converti un număr X, de *n* biţi, reprezentat prin complementul său faţă de 2, într-un număr de *m* biţi, însă trebuie procedat cu mare atenţie. Dacă $m>n$, trebuie să adăugăm în stânga lui X un număr de $m-n$ biţi identici cu bitul

de semn al lui X. Cu alte cuvinte, un număr pozitiv se completează cu zerouri, iar unul negativ cu cifre de 1; această operaţie se numeşte *extensie de semn*. Dacă $m<n$, îndepărtăm $n-m$ biţi din poziţiile din stânga ale lui X; rezultatul obţinut este corect numai dacă biţii înlăturaţi sunt identici cu bitul de semn al rezultatului.

Majoritatea calculatoarelor şi a aparatelor digitale folosesc pentru numerele negative sistemul de reprezentare prin complementul faţă de 2.

Reprezentarea prin complementul redus faţă de bază

În sistemul de reprezentare prin complementul redus faţă de bază, complementul unui număr D, de n cifre, se obţine scăzând numărul din r^n-1. Acest lucru se poate realiza prin calcularea complementului fiecărei cifre a numărului D fără a aduna 1, cum se proceda la calcularea complementului faţă de bază.

Reprezentarea prin complementul faţă de şir de 1

În sistemul binar, reprezentarea prin complementul redus faţă de bază înseamnă *complementarea faţă de un şir de 1*. Ca şi în cazul complementului faţă de 2, bitul cel mai semnificativ reprezintă semnul: 0 pentru numere pozitive şi 1 pentru numere negative. Există, prin urmare două reprezentări ale numărului zero, una pozitivă (00…00) şi una negativă (11…11). Numerele pozitive sunt reprezentate identic atât în complementarea faţă de 2, cât şi în complementarea faţă de şir de 1. Reprezentările numerelor negative diferă însă printr-un 1. Ponderea bitului celui cel mai semnificativ nu mai este, în complementarea faţă de şir de 1, -2^{n-1}, ci $-(2^{n-1}-1)$. Domeniul de reprezentare a numerelor este cuprins între $-(2^{n-1}-1)$ şi $+(2^{n-1}-1)$.

Exemple:

$17_{10} = 00010001_2 \rightarrow 11101110_2 = -17_{10}$
$119_{10} = 01110111_2 \rightarrow 10001000_2 = -119_{10}$

$-99_{10} = 10011100_2 \rightarrow 01100011_2 = 99_{10}$
$-127_{10} = 10000000_2 \rightarrow 01111111_2 = 127_{10}$

Principalele avantaje ale sistemului de complementare faţă de şir de 1 sunt simetria şi simplitatea calculelor. Circuitul sumator pentru reprezentate prin complementare faţă de şir de 1 este însă mai complicat decât în cazul folosirii complementului faţă de 2. De asemenea, circuitele detectoare de zero care lucrează cu numere complementate faţă de un şir de 1 trebuie să verifice ambele reprezentări ale lui zero, fie să efectueze conversia 11…11 în 00…00.

2.5.3 Reprezentarea cu exces

Într-adevăr, numărul de sisteme de reprezentare a numerelor negative este excesiv de mare. În reprezentarea cu exces de B, un şir de *m* biţi, a cărui valoare întreagă, neprecedată de semn, este M ($0 \leq M \leq 2^m$), reprezintă întregul precedat de semn M–B, unde B poartă denumirea de *pol* al sistemului de numeraţie.

De exemplu, într-un sistem de reprezentare cu exces de 2^{m-1} se poate reprezenta orice număr X din domeniul cuprins între -2^{m-1} şi $2^{m-1}-1$ prin reprezentarea binară, cu *m* biţi, a numărului $X+2^{m-1}$ (care este întotdeauna negativ şi mai mic decât 2^m). În acest caz, domeniul de reprezentare este acelaşi ca pentru numerele reprezentate prin complementul faţă de 2, cu *m* biţi. De fapt, reprezentarea oricărui număr în ambele sisteme este identică, excepţie făcând biţii de semn, care sunt întotdeauna diferiţi. (Afirmaţia este valabilă doar pentru polul 2^{m-1}).

Reprezentările cu exces se folosesc cel mai frecvent în sistemele de numeraţie cu virgulă mobilă.

2.6 Adunarea şi scăderea complementelor faţă de 2

2.6.1 Reguli de adunare

Tabelul 2-3, ce conţine cifrele zecimale şi echivalentele acestora în alte sisteme de numeraţie, pune în evidenţă motivul pentru care complementul faţă de 2 este preferabil pentru efectuarea operaţiilor aritmetice. Pornind de la 1000_2 (-8_{10}) şi parcurgând crescător şirul de numere, observăm că fiecare complement faţă de 2 al unui număr, până la 0111_2 ($+7_{10}$), se obţine din cel precedent prin adăugarea unui 1, ignorând transportul dincolo de bitul din poziţia a patra. Nu putem spune acelaşi lucru despre reprezentările prin modul precedat de semn şi prin complementare faţă de şir de 1. Întrucât adunarea obişnuită este doar o extensie a numărării, numerele reprezentate prin complementul faţă de 2 se pot aduna conform metodei obişnuite din binar, ignorând eventualul transport dincolo de MSB. Rezultatul adunării va fi întotdeauna corect atâta timp cât nu este depăşit domeniul sistemului de numeraţie. Afirmaţiile de mai sus sunt susţinute de exemplele următoare de adunare în zecimal şi de corespondenţele lor de 4 biţi în complement faţă de 2.

Tabel 2-3 Cifre zecimale şi echivalentele lor de 4 biţi

Zecimal	Complement faţă de 2	Complementare faţă de şir de 1	Modul şi semn	Cu exces de 2m-1
-8	1000	-	-	0000
-7	1001	1000	1111	0001
-6	1010	1001	1110	0010

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

2.6.2 Depăşirea

Dacă rezultatul unei adunări depăşeşte domeniul sistemului de numeraţie, se spune că are loc o depăşire (*overflow*). Adunarea a două numere de semne diferite nu va produce niciodată o depăşire, spre deosebire de adunarea a două numere cu acelaşi semn, aşa cum dovedesc şi exemplele următoare:

- 3 +	1101 +	+5 +	0101 +
- 6	1010	+6	0110
- 9	10111 = +7	+11	1011 = -5

- 8 +	1000 +	+7 +	0111 +
- 8	1000	+7	0111
- 16	10000 = +0	+14	1110 = -2

Din fericire există o regulă simplă prin care se poate verifica dacă adunarea a generat o depăşire: „În urma adunării are loc o depăşire dacă termenii sunt de acelaşi semn, iar suma are semnul opus.”

Acest enunţ este exprimat uneori în funcţie de transporturile rezultate în operaţia de adunare: „Depăşirea are loc dacă biţii c_{in}, de transport către bitul de semn, şi c_{out}, de transport de la bitul de semn, sunt diferiţi.”

2.6.3 Reguli de scădere

Numerele reprezentate prin complement față de 2 se pot scădea la fel ca orice numere binare neprecedate de semn, și în acest caz putând fi formulate niște reguli referitoare la depășire. Oricum, majoritatea circuitelor de efectuare a scăderii ce lucrează cu numere reprezentate în complement față de 2 nu aplică o metodă directă de scădere. De cele mai multe ori, scăzătorul se neagă prin calcularea complementului lui față de 2, apoi se adună la descăzut conform regulilor de adunare.

Negarea scăzătorului și adunarea lui la descăzut se pot realiza printr-o singură operație, astfel: se calculează complementul bit cu bit al scăzătorului și rezultatul se adună la descăzut cu un transport inițial (c_{in}) egal cu 1, în loc de 0. Iată câteva exemple:

		$1 - c_{in}$			$1 - c_{in}$
+4 –	0100 –	0100 +	+3 –	0011 –	0011 +
+3	0011	1100	+4	0100	1011
+1		1 0001	- 1		1111

		$1 - c_{in}$			$1 - c_{in}$
+3 –	0011 –	0011 +	-3 –	1101 –	1101 +
- 4	1100	0011	-4	1100	0011
+7		0111	+1		1 0001

În cazul scăderii, depășirea se detectează prin examinarea semnelor descăzutului și *complementului* scăzătorului, aplicând aceleași reguli ca pentru adunare. Sau, dacă folosim metoda din exemplele precedente, se examinează transporturile către și de la bitul de semn, neconcordanța semnelor lor traducându-se prin producerea unei depășiri, tot ca la adunare.

Încercarea de a nega numărul negativ pentru care nu există un simetric pozitiv are ca efect generarea unei depășiri, după regulile de mai sus, când adăugăm un 1 pentru a obține complementul:

```
-(-8) = -1000 = 0111 +
                0001
                1000 = -8
```

Acest număr poate fi însă folosit în adunări și scăderi atâta timp cât rezultatul final nu depășește domeniul de reprezentare a operanzilor:

				$1 - c_{in}$
+4 +	0100 +	- 3 –	1101 –	1101 +
- 8	1000	- 8	1000	0111
- 4	1100	+5		1 0101

În concluzie, la adunarea numerelor neprecedate de semn, apariţia unui transport sau a unui împrumut relative la poziţia bitului celui mai semnificativ indică faptul că rezultatul se situează în afara domeniului. În cazul adunării complementelor faţă de 2, condiţia de depăşire este cea care arată dacă rezultatul nu se încadrează în domeniu. La adunarea numerelor precedate de semn, transportul dintre poziţia bitului celui mai semnificativ nu constituie un indiciu, în sensul că depăşirea se poate produce sau nu, indiferent dacă apare sau nu un transport.

2.7 Înmulţirea în binar

Am învăţat să efectuăm înmulţirea prin adunarea decalată a produselor dintre deînmulţit şi fiecare dintre cifrele înmulţitorului. Aceeaşi metodă poate fi utilizată pentru calcularea produsului a două numere binare neprecedate de semn. Calcularea produselor dintre deînmulţit şi fiecare cifră a înmulţitorului este banală, întrucât cifrele înmulţitorului sunt doar 0 şi 1.
Exemplu:

11 ×	1011 ×	deînmulţit
13	1101	înmulţitor
33	1011	
11	0000	
143	1011	
	1011	produse parţiale decalate
	10001111	produs

În loc de a calcula toate produsele decalate şi de a le aduna ulterior, în sistemele digitale este mai convenabil ca fiecare dintre acestea să fie adăugat, pe măsură ce este calculat, la *valoarea curentă a produsului*.

Aplicând metoda aceasta exemplului anterior, pentru numerele de 4 biţi sunt necesare patru adunări la valoarea curentă a produsului:

11 ×	1011 ×	deînmulţit
13	1101	înmulţitor
	0000	valoare curentă
	1011	produs parţial decalat
	01011	valoare curentă
	0000↓	produs parţial decalat
	001011	valoare curentă
	1011↓↓	produs parţial decalat
	0110111	valoare curentă
	1011↓↓↓	produs parţial decalat

10001111 produs

În general, la înmulţirea unui număr de n biţi cu unul de m biţi, pentru reprezentarea produsului sunt necesari maximum n + m biţi.

Înmulţirea numerelor precedate de semn se poate efectua înmulţind numerele neprecedate de semn şi respectând regula semnelor: se înmulţesc modulele fără a lua în consideraţie semnele lor şi se atribuie produsului semnul plus dacă factorii au acelaşi semn şi minus dacă au semne diferite. Metoda aceasta este foarte convenabilă în reprezentarea prin modul şi semn, întrucât cele două entităţi sunt exprimate separat.

În sistemul de reprezentare prin complement faţă de 2, înmulţirea se poate efectua printr-o succesiune de adunări, în acest sistem, a produselor parţiale decalate, cu excepţia ultimei etape, când produsul parţial rezultat din înmulţirea cu MSB trebuie negat înainte de a fi adunat la valoarea curentă a produsului. Reluăm, în continuare exemplul precedent, de data aceasta considerând că deînmulţitul şi înmulţitorul sunt numere reprezentate prin complement faţă de 2:

- 5 ×	1011 ×
- 3	1101
	*0*0000
	*1*1011
	*1*11011
	*0*0000↓
	*1*111011
	*1*1011↓↓
	*1*1100111
	*0*0101↓↓↓
	00001111

La determinarea MSB este necesară multă atenţie deoarece în fiecare etapă se adaugă câte un bit semnificativ, iar numerele cu care se lucrează sunt precedate de semn. Din această cauză, înainte de a aduna un nou produs parţial decalat la valoarea curentă, de k biţi, a produsului, trebuie să le exprimăm pe ambele prin k + 1 biţi semnificativi prin extensie de semn, aşa cum arată cifrele cursive din exemplu. Fiecare dintre sumele rezultate are k + 1 biţi; orice transport dinspre poziţia MSB al sumei de k + 1 biţi se ignoră.

2.8 Împărţirea în binar

Cel mai simplu algoritm de împărţire în binar are la bază metoda de scădere decalată învăţată în clasele primare.

Metodele de împărţire în binar sunt complementare metodelor de înmulţire în binar. Pentru un algoritm de împărţire obişnuit sunt necesare un deîmpărţit de n + m biţi şi un împărţitor de n biţi, obţinându-se un cât de m biţi şi un rest de n biţi. La împărţire se consideră că se produce o *depăşire* dacă împărţitorul este 0 sau dacă pentru exprimarea câtului sunt necesari mai mult de m biţi. Pentru majoritatea circuitelor de împărţire din calculatoare, n = m.

Împărţirea numerelor precedate de semn se realizează efectuând împărţirea numerelor neprecedate de semn şi aplicând regula semnelor. Restul trebuie să aibă acelaşi semn ca deîmpărţitul. Ca şi în cazul înmulţirii, există metode speciale de a efectua împărţirea unor numere exprimate prin complement faţă de 2; asemenea metode sunt frecvent utilizate de circuitele de împărţire din calculatoare.

2.9 Codarea binară a numerelor zecimale

Deşi numerele binare sunt cele mai adecvate pentru calculele ce trebuie efectuate intern de către sistemele digitale, majoritatea oamenilor preferă să lucreze cu numere zecimale. Din această cauză, interfeţele cu exteriorul ale sistemelor digitale trebuie să preia şi să afişeze valori zecimale.

Folosirea sistemului zecimal de către utilizatorii umani nu poate afecta principiul de bază al circuitelor digitale, care prelucrează semnale ce pot avea una din cele două stări desemnate prin 0 sau 1. Prin urmare un număr zecimal se reprezintă în sistem digital printr-un şir de biţi.

O mulţime formată din şiruri de *n* biţi, în care fiecare şir de biţi reprezintă un element sau număr, se numeşte *cod.* O combinaţie determinată de valori a *n* biţi se numeşte *cuvânt de cod*. Între valorile biţilor unui cuvânt de cod şi elementul pe care acesta îl reprezintă poate exista o relaţie matematică sau nu. Mai mult, nu e obligatoriu ca un cod constituit din şirul de *n* biţi să conţină 2^n cuvinte de cod diferite.

Pentru reprezentarea celor zece cifre ale sistemului zecimal sunt necesari minim 4 biţi. Există milioane de moduri în care să se exprime cele zece cuvinte de cod de câte 4 biţi, unele dintre cele mai utilizate coduri zecimale fiind prezentate în tabelul 2-4.

Cel mai ”natural” cod zecimal este, probabil codul BCD (binary-coded decimal; sau ZCB – zecimal codat binar), care reprezintă cifrele de la 0 la 9 prin cuvinte de 4 biţi, neprecedate de semn, de la 0000 la 1001.

Tabel 2-4 Coduri de numere zecimale

Cifra zecimală	BCD (8421)	2421	Cu exces de 3	Bicvinar	1 di 10
0	0000	0000	0011	0100001	1000000000
1	0001	0001	0100	0100010	0100000000
2	0010	0010	0101	0100100	0010000000

3	0011	0011	0110	0101000	0001000000
4	0100	0100	0111	0110000	0000100000
5	0101	1011	1000	1000001	0000010000
6	0110	1100	1001	1000010	0000001000
7	0111	1101	1010	1000100	0000000100
8	1000	1110	1011	1001000	0000000010
9	1001	1111	1100	1010000	0000000001

Conversiile între reprezentările în BCD şi în zecimal se efectuează banal, înlocuindu-se direct fiecare cifră zecimalã cu câte un cuvânt de 4 biţi.

Unele programe de calculator lucrează cu grupe de câte două cifre zecimale, asociindu-le cuvinte de 8 biţi, în reprezentare BCD compactată. În acest mod, un octet acoperă valorile 0...99, spre deosebire de domeniul 0...255, acoperit de un cod binar obişnuit. Numerele pot fi reprezentate în BCD prin cuvinte de orice lungime, folosindu-se cîte un octet pentru fiecare combinaţie de două cifre.

Ca şi în binar, în BCD există mai multe moduri de reprezentare a numerelor negative. Numerele cu semn prezintă în BCD un bit suplimentar, pentru semn. Sunt larg utilizate reprezentarea prin modul şi semn (codarea semnelor se face arbitrar), cât şi cea prin complement faţă de 10 (0000 = minus, 1001 = plus).

Adunarea cifrelor în BCD este similară cu adunarea numerelor binare pe 4 biţi, fără semn, cu excepţia faptului că este necesară o corecţie dacă rezultatul depăşeşte 1001. Corecţia se face prin adunarea cifrei 6 (0110).

Exemple:

5	0101	8	1000	4	0100
+9	+1001	+8	+1000	+5	+0101
14	1110	16	10000	9	1001
	+0110 - corecţie		+0110 - corecţie		
10+4	10100	10+6	10110		

BCD este un *cod ponderat*, deoarece fiecare cifră zecimală se poate calcula pornind de la cuvântul de cod şi atribuind câte o pondere fixă fiecărui bit al cuvântului de cod. În codul BCD, ponderile biţilor sunt, respectiv, 8, 4, 2, şi 1, codul mai fiind denumit şi *cod 8421*.

Codul 2421 din tabelul 2-4, atribuie biţilor alte ponderi. Acest cod are avantajul că este *autocomplementar*, adică un cuvânt de cod atribuit oricărei cifre reprezentate prin complement faţă de şir de 9 se poate obţine prin complementarea faţă de şir de 1 a cuvântului de cod ce desemnează acea cifră.

Codul exces de 3 este tot un cod autocomplementar, dar nu este un cod ponderat. Un cuvânt de cod se obţine prin adunarea cuvântului corespunzător din BCD cu $0011_{2.}$

Codul bicvinar prezentat în tabelul 2-4 este un cod care utilizează 7 biţi. Primii 2 biţi ai cuvântului de cod indică în care dintre intervalele 0...4 şi 5...9 se încadrează numărul, iar următorii 5 biţi indică poziţia pe care se afle numărul, în intervalul selectat. Unul din avantajele acestui cod este proprietatea de detectare a erorilor.

Codul 1 din 10 este modalitatea de codare a cifrelor zecimale cea mai puţin compactă, utilizând 10 din cele 1024 de cuvinte de cod pe 10 biţi posibile.

2.10 Codul Gray

Acest cod prezintă avantajul modificarii unui singur bit între două cuvinte de cod succesive.

În tabelul 2-5 este prezentat un cod Gray pe 3 biţi şi modul de formare al acestuia, între cuvintele de cod succesive existând diferenţă de un singur bit.

Tabel 2-5 Reprezentarea numerelor în cod Gray

Nr. zecimal	Cod Gray	Cod binar
0	000	000
1	001	001
2	011	010
3	010	011
4	110	100
5	111	101
6	101	110
7	100	111

Există două moduri de a constitui un cod Gray. Prima metodă se bazează pe faptul că acest cod este *reflectant*, el putând fi definit recursiv după următoarele reguli:

- Un cod Gray de un bit are două cuvinte de cod: 0 şi 1.
- Primele 2^n cuvinte de cod ale unui cod Gray de n+1 biţi se formează din cuvintele de cod ale unui cod Gray de n biţi, în aceeaşi ordine, prin adăugarea în faţă a unui 0.
- Ultimele 2^n cuvinte ale unui cod Gray de n+1 biţi se formează din cuvintele de cod ale unui cod gray de n biţi, ordonate invers, prin adăugarea în faţă a unui 1.

Pentru a construi un cod Gray de n biţi prin această metodă, este necesar să construim toate codurile Gray cu un număr de biţi mai mic ca n.

Prin cea de-a doua metodă, cuvintele de cod ale unui cod Gray de *n* biţi se pot deduce direct din cuvintele de cod corespunzătoare codului binar pe *n* biţi:

- Biţii cuvintelor de cod ale codurilor binar sau Gray de *n* biţi se numerotează de la 0 la *n*-1, de la dreapta la stânga.
- Bitul *i* al cuvântului de cod Gray are valoarea 0 dacă biţii *i* şi *i*+1 ai cuvântului de cod binar corespunzător sunt identici, în caz contrar valoarea bitului *i* este 1; (dacă *i*+1=*n*, atunci bitul *n* al cuvântului de cod binar se consideră 0).

2.11 Coduri de caractere

Un şir de biţi nu trebuie să reprezinte neapărat un număr, majoritatea informaţiilor prelucrate de un calculator fiind de natură numerică. Cel mai comun tip de date numerice este tipul *text*, adică şiruri de caractere aparţinând unui set de caractere. Fiecare caracter este reprezentat în calculator ca un şir de biţi, conform unei convenţii de reprezentare.

Codul de caractere cel mai cunoscut şi utilizat este codul ASCII (American Standard Code for Information Interchange). Fiecare caracter este reprezentat pe 7 biţi obţinându-se în total 128 de caractere diferite.

2.12 Coduri pentru transmisia si stocarea datelor seriale

Există două moduri de stocare şi de transmisie a datelor numerice: în format paralel sau în format serial.

În transmisia paralelă de date, pentru fiecare poziţie de bit este necesară o linie de transmisie separată a semnalului. În cazul stocării paralele a semnalului toţi biţii unui cuvânt pot fi citiţi sau scrişi simultan. Utilizarea formatelor paralele nu este rentabilă în anumite aplicaţii. Spre exemplu, stocarea paralelă a octeţilor de date pe un disc magnetic ar necesita opt capete de citire/scriere, iar în reţelele telefonice, transmisia paralelă a octeţilor de date ar implica instalarea a opt linii telefonice.

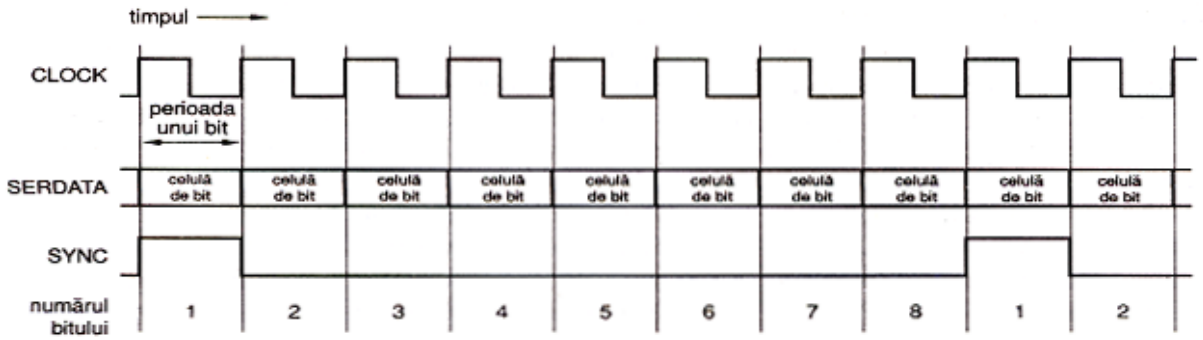


Figura 2-1 Concepte fundamentale ale transmisiei de date seriale

Formatele seriale permit transmiterea şi stocarea datelor bit după bit, reducând preţul sistemelor pentru numeroase aplicaţii. Așa cum se poate observă în figura 2-1, pentru transmisia şi refacerea unui flux de date seriale sunt necesare minimum trei semnale: un semnal de tact pentru definirea celulelor de bit, un semnal de sincronizare pentru definirea limitelor unui cuvânt şi datele seriale propriu-zise.

În numeroase aplicaţii, nu este totuşi rentabilă transmiterea a trei semnale separate (exemplu: liniile telefonice sau capetele de citire/scriere). Aceste sisteme folosesc de obicei un singur flux de date seriale format dintr-o combinaţie a celor trei semnale, iar pentru refacerea la recepţie a informaţiilor de tact şi de sincronizare se utilizează circuite analogice şi digitale foarte sofisticate.

În figura 2-2 sunt prezentate câteva dintre cele mai folosite codurile de transmisie serială, cel mai mult folosite.

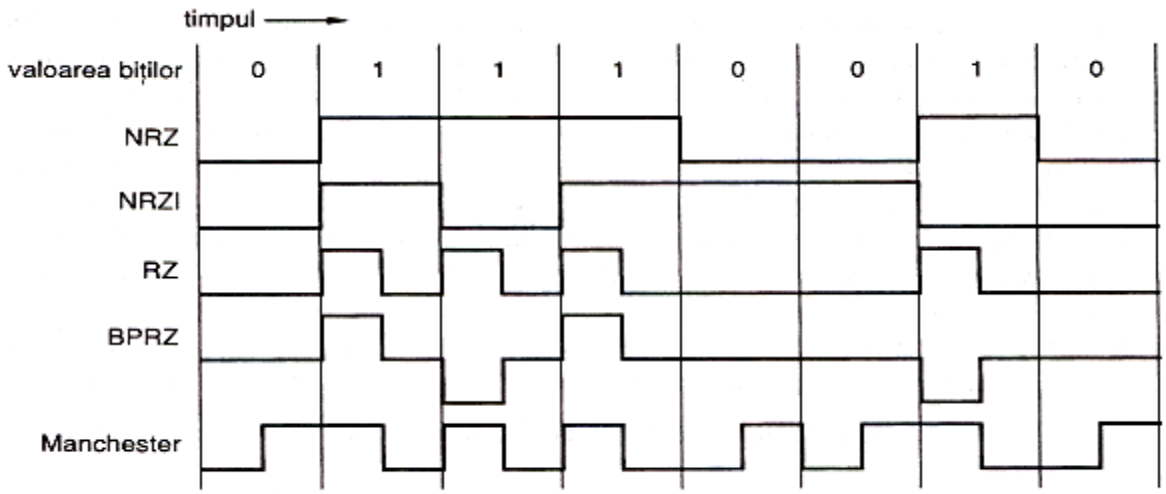


Figura 2-2 Coduri pentru date seriale larg utilizate

Codul *Non-Returnto-Zero* (NRZ – fără revenire la zero) este cea mai simplă şi mai fiabilă metodă de codare pentru transmisiile la mică distanţă. Fiecare valoare de bit este transmisă pe linie pe intreaga celulă de bit. Dezavantajul este că semnalul de date trebuie să fie insoţit şi de un semnal de tact, pentru definirea celulelor de bit şi pentru a se putea reface corect informaţia la recepţie.

Codul *Non-Return-to-Zero Invert-at-1s* (NRZI – fără revenire la zero cu inversare la 1) elimină această deficienţă, semnalul de tact putând fi refăcut la recepţie de un circuit specializat DPLL – *Digital Phase-Locked Loop* (buclă digitală de urmărire a fazei).

Codul *Return-to-Zero* (RZ – cu revenire la zero) este asemănător codului NRZ cu excepţia faptului că nivelul corespunzător unui bit 1 se transmite numai pe o fracţiune din perioada bitului (de obicei 1/2).

Codul *Bipolar Return-to-Zero* (BPRZ – bipolar cu revenire la zero) este asemănător cu codul RZ, însă transmisia se face pe trei niveluri de semnal, biţii de 1 fiind transmişi alternativ ca +1 şi -1.

În ceea ce priveşte codul *Manchester* (*bifazic*), fiecare bit (0 sau 1) este caracterizat de o tranziţie de fază: bitul de 0 este codat ca o tranziţie de la 0 la 1 la mijlocul unei celule de bit, iar bitul de 1 este codat ca o tranziţie de la 1 la 0 la mijlocul celulei de bit.

În continuare mai amintim câteva tipuri de coduri nu mai puţin importante, dar asupra cărora nu ne vom opri pentru o discuţie amănunţită:

- **Coduri de activare, de condiţionare şi de stare** (folosite în cazuri în care se comandă o acţiune, se semnalizează realizarea sau nerealizarea unei condiţii sau se reprezintă starea curentă a echipamentelor).
- Cuburi n-dimensionale şi distanţe.
- Coduri pentru detectarea si corectarea erorilor:
  - coduri detectoare de erori (ex.: coduri de paritate);
  - coduri corectoare de erori şi detectoare de erori multiple;
  - coduri Hamming;
  - coduri cu control de redundanţă ciclică (CRC);
  - coduri bidimensionale;
  - coduri cu sumă de control;
  - coduri m din n.

3. Tehnologii de realizare a circuitelor digitale

În ciuda avalanşei publicitare, lumea în care trăim este analogică, nu digitală. Tensiunile, curenţii şi alte mărimi fizice din circuitele în funcţiune iau un număr infinit de valori, în funcţie de dispozitivele concrete ce conţin acele circuite. Deoarece valorile reale variază continuu, putem utiliza o mărime fizică, de pildă, tensiunea semnalului dintr-un circuit, pentru a reprezenta un număr real (de exemplu, 3,14159265358979 volţi reprezintă constanta matematică π cu precizia de 14 cifre zecimale).

Din păcate, mărimile fizice din circuitele reale nu sunt caracterizate de o bună stabilitate şi precizie. Ele sunt influenţate, printre altele, de toleranţele de fabricaţie, de temperatură, de tensiunea de alimentare, de radiaţiile cosmice şi de zgomotul generat de alte circuite. Dacă am folosi o tensiune analogică pentru a reprezenta numărul π, am observa că, în loc să fie o constantă matematică absolută, π variază cu 10% şi chiar mai mult.

De asemenea, multe operaţii matematice şi logice ar fi dificil şi chiar imposibil de efectuat folosind mărimi analogice. Cu oarecare abilitate, este posibilă realizarea unui circuit analogic a cărui tensiune de ieşire să fie egală cu rădăcina pătrata a tensiunii de intrare, însă nimeni nu a construit vreodată un circuit analogic cu 100 de intrări şi 100 de ieşiri, ale cărui tensiuni de ieşire să aibă aceleaşi valori ca tensiunile de intrare, însă ordonate crescător.

3.1 Semnale şi porţi logice

Logica digitală disimulează capcanele lumii analogice prin transformarea mulţimii infinite de valori reale şi pot fi luate de mărimile fizice în două submulţimi cărora le corespund doar două valori numerice şi valori logice posibile - 0 şi 1. Ca urmare, circuitele care utilizează logica digitală pot fi analizate şi proiectate pragmatic cu ajutorul algebrei şi tabelelor de comutaţie şi prin alte mijloace abstracte de descriere a comportării într-un circuit a cifrelor 0 şi 1.

O valoare logică, 0 sau 1, este denumită frecvent cifră binară sau bit. Dacă o aplicaţie necesită mai mult de două valori discrete şi pot folosi biţi suplimentari, cu o mulţime de n biţi putând fi reprezentate 2^n valori diferite.

În tabelul 3-1 găsiţi exemple de fenomene fizice cu ajutorul cărora se pot reprezenta biţi în unele tehnologii digitale moderne (şi nu prea). Cele mai multe fenomene prezintă o regiune de nedeterminare între stările de 0 şi de 1 (cum ar fi

o tensiune de 1,8 V, o lumină slabă, un condensator încărcat puţin etc.). Această regiune de nedeterminare este necesară pentru ca stările de 0 şi 1 să poată fi definite riguros şi detectate totdeauna corect. Rezultatele pot fi alterate cu mai mare uşurinţă de către zgomot dacă limitele ce separă stările de 0 şi de 1 sunt prea apropiate.

Când se referă la circuitele logice electronice, cum sunt CMOS şi TTL, proiectanţii de circuite digitale utilizează adesea, în loc de „0” şi „1”, indicativele „L”, de la LOW (jos), şi „H” de la HIGH (sus), fiindcă ştiu că este vorba de circuite reale, nu de mărimi abstracte:

- L (LOW) – Semnal cu valori cuprinse în domeniul valorilor algebrice mici, care sunt interpretate ca „0” logic.
- H (HIGH) – Semnal cu valori cuprinse în domeniul valorilor algebrice mari, care sunt interpretate ca „1” logic.

Remarcaţi că atribuirea valorilor 0 şi 1 stărilor LOW şi HIGH se face întrucâtva arbitrar. Atribuirea valorii 0 stării LOW şi a valorii 1 stării HIGH pare ceva firesc şi se numeşte logică pozitivă. Celălalt mod de atribuire, 1 pentru LOW şi 0 pentru HIGH, este utilizat mai rar şi se cheamă logică negativă.

Deoarece o singură valoare binară se reprezintă printr-o gamă largă de valori ale unei mărimi fizice, circuitele logice digitale sunt extrem de imune la variaţiile parametrilor componentelor, la cele ale tensiunilor de alimentare şi la zgomot. În plus, semnalele mai slabe pot fi regenerate cu ajutorul unor circuite de amplificare tampon, astfel că semnalele digitale pot fi transmise la orice distanţă fără ca informaţia conţinută de ele să se piardă. De exemplu, un amplificator tampon în logica CMOS transformă orice tensiune de intrare HIGH într-o tensiune de ieşire de aproximativ 5,0 V şi orice tensiune de intrare LOW într-o tensiune de ieşire de aproximativ 0,0 V.

Tabel 3-1 Stări fizice care pot reprezenta biţi în diverse logici de calcul şi tehnologii de memorare

Starea şi reprezintă un bit		
Tehnologia	**„0’’**	**„1”**
Logica pneumatică	Fluid la presiune joasă	Fluid la presiune inaltă
Logica de relee	Circuit deschis	Circuit închis
Logica metal-oxid-semiconductor	0 – 1,5 V	3,5 – 5,0 V
Logica tranzistor-tranzistor (TTL)	0 – 0,8V	2,0 – 5,0 V
Fibre optice	Întuneric	Lumină
Memorie dinamică	Condensator descărcat	Condensator încărcat
Memorie nevolatilă anulabilă	Electroni blocaţi	Electroni eliberaţi
Memorie bipolară cu acces numai pentru citire	Siguranţă arsă	Siguranţă intacta
Memorie cu bule magnetite	Absenţa bulelor magnetite	Prezenta bulelor magnetite
Benzi şi discuri magnetite	Flux orientat către „nord”	Flux orientat către „sud”
Memorie polimerică	Moleculă în starea A	Moleculă în starea B

Disc compact (CD) cu acces numai pentru citire	Pit-uri absente	Pit-uri prezente
Disc compact (CD) reînregistrabil	Emulsie în stare cristalină	Emulsie în stare necristalină

Un circuit logic poate fi reprezentat, cu minimum de detalii, ca o simplă „cutie neagră" cu un anumit număr de intrări şi ieşiri. De exemplu, în fig. 3-1 apare un circuit logic cu trei intrări şi o ieşire. Dar această reprezentare nu descrie modul în care circuitul răspunde la semnalele de intrare.

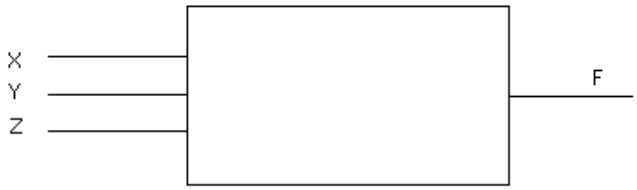


Figura 3-1 Reprezentarea printr-o cutie neagră a unui circuit cu trei intrări şi o ieşire

Pentru a proiecta circuite electronice sunt însă necesare numeroase informaţii care să descrie cu precizie comportarea unui circuit din punct de vedere electric. Dar, întrucât se poate considera că semnalele de la intrările unui circuit logic iau doar valorile discrete 0 şi 1, funcţionarea „logică" a circuitului poate fi descrisă printr-un tabel care nu ţine seama de aspectele de natură electrică, menţionând numai valorile discrete 0 şi 1.

Tabel 3-2 Tabelul de adevăr pentru un circuit logic combinational

X	Y	Z	F
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

Un circuit logic ale cărui semnale de ieşire depind numai de semnalele de intrare curente se numeste ***circuit combinaţional***. Funcţionarea unui asemenea circuit se descrie printr-un ***tabel de adevăr,*** în care apar toate combinaţiile de semnale de intrare şi valoarea semnalului (semnalelor) de ieşire şi rezultă pentru fiecare combinaţie de intrare. Tabelul 3-2 este tabelul de adevăr al unui circuit logic cu trei semnale de intrare, X, Y şi Z, şi o singură ieşire, F.

Un circuit cu capacitate de memorare, ale cărui semnale de ieşire depind de semnalul de intrare curent şi de valorile anterioare ale semnalului de intrare, se numeşte ***circuit secvenţial.*** Comportarea unui asemenea circuit poate fi

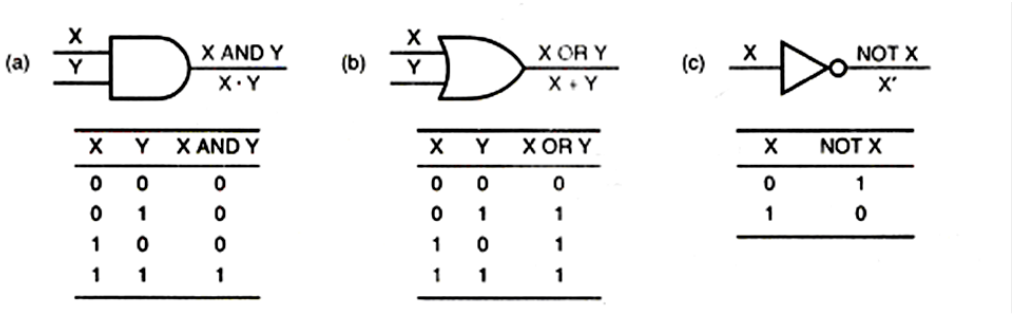
descrisă printr-un ***tabel de stări***, în care se precizează valoarea semnalului de ieşire şi starea următoare în funcţie de starea curentă şi de semnalul de intrare curent.

Pentru realizarea oricărui circuit digital logic combinaţional se pot folosi numai trei funcţii logice de bază: **AND** (ŞI), **OR** (ŞI) şi **NOT** (NU). În fig. 3-2 găsiţi tabelele de adevăr şi simbolurile „porţilor" logice care realizează aceste funcţii. Simbolurile şi tabelele de adevăr ale funcţiilor **AND** şi **OR** pot fi adaptate pentru porţi cu orice număr de intrări. Funcţiile porţilor pot fi descrise cu uşurinţă în cuvinte:

- *poartă **AND*** are ieşirea 1 dacă şi numai dacă toate intrările sale sunt 1.
- *poară **OR*** are ieşirea 1 dacă şi numai dacă cel puţin una dintre intrările sale este 1.
- *poartă **NOT**,* numită, de obicei, *inversor,* produce la ieşire valoarea opusă valorii de la intrare.

Cerculeţul de la ieşirea simbolului inversorului mai este numit *cerculeţ inversor* şi se foloseşte atât împreună cu acest simbol, cât şi cu alte simboluri de porţi, pentru a indica funcţia de inversare.

Remarcaţi că, atunci când am definit funcţiile **AND** şi **OR**, am precizat numai condiţiile de intrare pentru care ieşirea este 1, deoarece, dacă ieşirea nu este 1, nu poate fi decât 0.



X	Y	X AND Y
0	0	0
0	1	0
1	0	0
1	1	1

X	Y	X OR Y
0	0	0
0	1	1
1	0	1
1	1	1

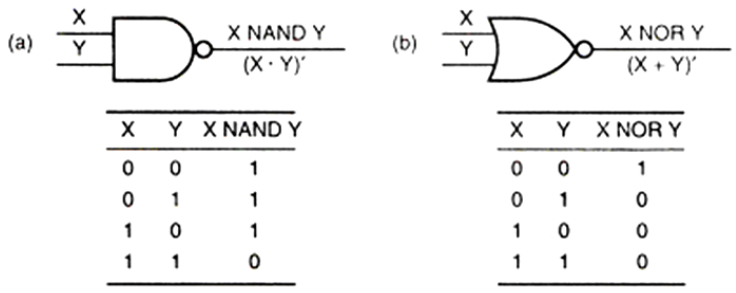
X	NOT X
0	1
1	0

Figura 3-2 Elemente logice de bază: a) AND; b) OR; c) NOT (inversor)

Prin combinarea funcţiei logice **NOT** cu **AND**, respectiv cu **OR**, se obţin alte două funcţii logice: **NAND** (ŞI-NU) şi **NOR** (ŞI-NU). În fig. 3-3 sunt date tabelele de adevăr şi simbolurile acestor porţi; funcţiile lor sunt, de asemenea, uşor de descris în cuvinte:

- *O poartă **NAND*** are ieşirea invers decât o poartă **AND**, adică 0 dacă şi numai dacă toate intrările sale sunt 1.
- *O poartă **NOR*** are ieşirea invers decât o poartă **OR**, adică 0 dacă şi numai dacă cel puţin una dintre intrările sale este 1.

Ca şi în cazul porţilor **AND** şi **OR**, simbolurile şi tabelele de adevăr ale porţilor **NAND** şi **NOR** pot fi adaptate pentru orice număr de intrări.



X	Y	X NAND Y
0	0	1
0	1	1
1	0	1
1	1	0

X	Y	X NOR Y
0	0	1
0	1	0
1	0	0
1	1	0

Figura 3-3 Porţi inversoare: a) NAND; b) NOR

În fig. 3-4 este reprezentat un circuit logic format din porţi **AND**, **OR** şi **NOT**, care funcţionează conform tabelului de adevăr 3-2.

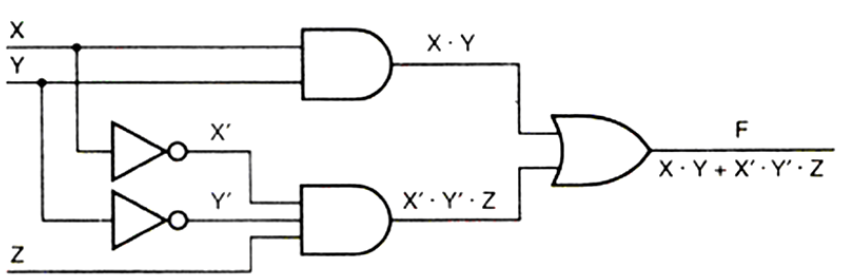


Figura 3-4 Circuit logic corespunzător tabelului de adevăr 3-2

Circuitele logice reale funcţionează, de asemenea, în altă coordonată analogică - timpul. De exemplu, figura 3-5 prezintă o *diagramă temporală* care arată cum ar putea răspunde circuitul din fig. 3-4 când la intrarea lui se aplică anumite forme de undă variabile în timp. Diagrama temporală indică faptul că semnalele logice nu trec instantaneu din 0 în 1 şi, de asemenea, că există o întârziere între o tranziţie de la intrare şi tranziţia corespunzătoare de la ieşire.

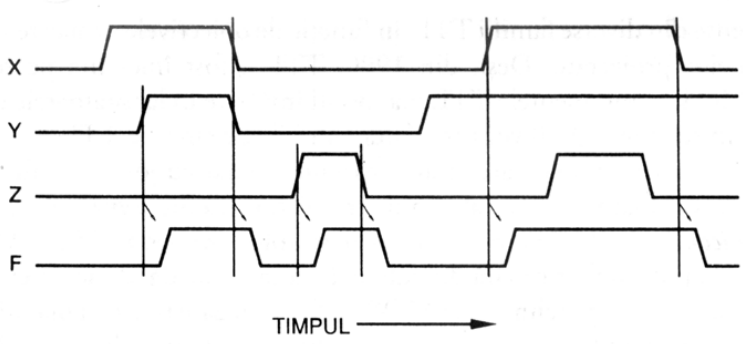


Figura 3-5 Diagrama temporală a unui circuit logic

3.2 Familii de circuite logice

Există nenumărate moduri de a proiecta un circuit logic. Primele circuite logice cu comandă electrică, realizate de Bell Laboratories în anii 1930, aveau la bază releele. Pe la mijlocul deceniului al cincilea, primul calculator electronic digital - numit Eniac - folosea circuite logice construite cu tuburi cu vid.

Inventarea *diodei semiconductoare* şi *a tranzistorului bipolar cu joncţiuni* a permis realizarea, spre sfârşitul anilor '50, a unor calculatoare mai mici, mai rapide şi cu capacitate mai mare. În anii '60, inventarea *circuitului integrat (CI)* a permis producerea pe un singur cip a mai multor diode, tranzistoare şi a altor componente, iar calculatoarele au avut de câstigat din aceasta.

Tot anii 1960 au fost martorii apariţiei primelor familii de circuite logice integrate. O ***familie de circuite logice*** este constituită din diverse cipuri de circuite integrate care prezintă intrări, ieşiri şi caracteristici interne similare, dar care îndeplinesc funcţii logice diferite. Cipuri aparţinând aceleiaşi familii pot fi interconectate în vederea obţinerii oricărei funcţii logice dorite. Pe de altă parte, este posibil ca unele cipuri din familii diferite să nu fie compatibile; ele pot folosi tensiuni de alimentare diferite şi pot reprezenta valorile logice prin condiţii de intrare şi de ieşire diferite.

De cel mai mare succes s-a bucurat *familia de circuite logice cu tranzistoare bipolare* numită ***logica tranzistor-tranzistor (TTL*** *- transistor-transistor logic).* Aparută în anii '60, TTL este, la ora actuală, o familie de familii de circuite logice compatibile între ele şi deosebindu-se numai prin viteză, putere consumată şi preţ.

Cu zece ani *înainte* de inventarea tranzistorului bipolar cu joncţiuni fuseseră stabilite principiile de funcţionare ale unui alt tip de tranzistor, numit *tranzistor cu efect de câmp metal-oxid-semiconductor (MOSFET)* şi, mai simplu, *tranzistor MOS.*

Începând de la jumătatea deceniului '80-'90, progresele realizate în proiectarea circuitelor cu tehnologie MOS, în special a celor cu tehnologie ***MOS complementară,*** au îmbunătăţit foarte mult performanţele şi popularitatea acestora. Majoritatea covârşitoare a noilor circuite integrate la scară largă, cum sunt microprocesoarele şi memoriile, folosesc tehnologia ***CMOS***.

Logica CMOS este cea mai puternică tehnologie logică digitală de pe piaţă, dar şi cea mai usor de înţeles. Drept urmare a perioadei îndelungate de transformări ale industriei pentru a realiza trecerea de la TTL la CMOS, multe familii CMOS au fost concepute pentru a fi compatibile, într-o oarecare măsură, cu TTL.

3.3 Logica CMOS

3.3.1 Nivelurile logice CMOS

Elementele logice abstracte lucrează cu cifrele binare, 0 şi 1, însă circuitele logice reale prelucrează semnale electrice, cum ar fi nivelurile de tensiune. Oricărui circuit logic (fie el electric sau neelectric) caracterizat de un parametru de circuit îi este caracteristic un domeniu de tensiuni şi se interpretează ca 0 logic una dintre stările acestuia şi cealaltă stare, care nu se suprapune peste prima, interpretată ca 1 logic.

Circuitele logice CMOS se alimentează, de obicei, cu tensiunea de 5 V. Asemenea circuite interpretează orice tensiune din domeniul 0 ... 1,5 V ca 0 logic, iar 1 logic fiind orice tensiune din domeniul 3,5 ... 5,0 V. Prin urmare, definirea nivelurilor LOW şi HIGH pentru circuitele logice CMOS alimentate cu 5 V sunt cele ilustrate de fig. 3-6.

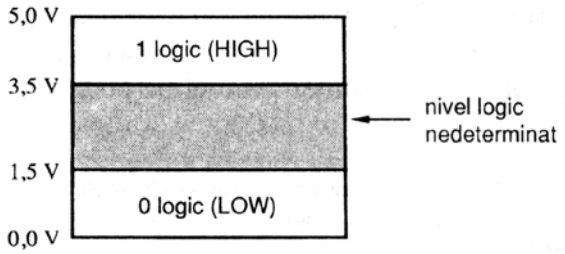


Figura 3-6 Nivelurile logice pentru circuitele CMOS uzuale

Tensiunile cuprinse între cele două domenii pot apărea numai în timpul tranziţiilor semnalelor şi sunt interpretate ca valori logice nedeterminate (adică pot fi interpretate de circuite fie ca 0, fie ca 1). Circuitele CMOS alimentate cu tensiuni de alte valori, de exemplu 3,3 V şi 2,7 V, au domeniile de valori împărţite în mod asemănător.

3.3.2 Tranzistoarele MOS

Modelul unui tranzistor MOS este cel al unui dispozitiv cu 3 terminale, care se comportă ca o rezistenţă comandată în tensiune. Aşa cum sugerează figura 3-7, o tensiune de intrare aplicată unuia dintre terminale comandă rezistenţa dintre celelalte două terminale. În aplicaţiile de logică digitală, tranzistoarele MOS sunt comandate astfel încât rezistenţa lor să fie ori foarte mare (şi atunci tranzistorul este închis), ori foarte mică (tranzistorul fiind, în acest caz, deschis).

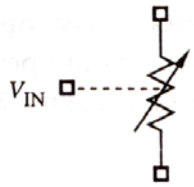
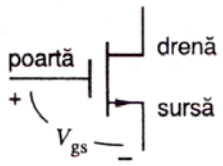


Figura 3-7 Tranzistorul MOS ca rezistenţă comandată în tensiune

Există două tipuri de tranzistoare MOS: cu canal *n* şi cu canal *p;* denumirile desemnează tipul de material semiconductor dintre terminalele rezistenţei comandate. Simbolul utilizat în scheme pentru *tranzistoarele MOS cu canal n (NMOS)* este cel din fig. 3-8. Terminalele tranzistorului se numesc *poartă, sursă* şi *drenă.* Drena se află, în mod normal, la o tensiune mai mare decât sursa.



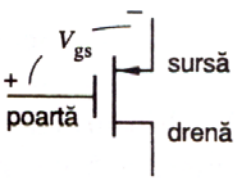
Rezistenţă comandată în tensiune:
V_{gs} creşte → R_{ds} scade

Notă: în mod normal $V_{gs} \geq 0$

Figura 3-8 Simbolul utilizat în scheme pentru un tranzistor MOS cu canal n (NMOS)

Tensiunea dintre poarta şi sursa (V_{gs},) unui tranzistor NMOS este, în mod normal, foarte aproape de zero şi pozitivă. Dacă $V_{gs} = 0$, rezistenţa drenă-sursă *(R_d,)* este foarte mare, de cel puţin 1 MΩ (10^6 Ω). Pe măsură ce mărim V_{gs}, (adică mărim tensiunea aplicată pe poartă), R_d, scade până la valori foarte mici, în jur de 10 Ω şi chiar mai mici, la unele dispozitive.

Simbolul utilizat în scheme pentru un *tranzistor MOS cu canal p (PMOS)* este cel din fig. 3-9. Funcţionarea acestuia este similară cu cea a unui tranzistor NMOS, cu excepţia faptului că, în mod normal, sursa se află la o tensiune mai mare decât drena, iar V_{gs}, este aproape zero şi negativă. Când V_{gs}, este zero, rezistenţa dintre sursă şi drenă (R_d,) este foarte mare. Dacă scădem algebric V_{gs}, (adică micşorăm tensiunea aplicată pe poartă), *Rd*, scade până la o valoare foarte mică.



Rezistenţă comandată în tensiune:
V_{gs} scade → R_d scade

Notă: în mod normal $V_{gs} \leq 0$

Figura 3-9 Simbolul utilizat în scheme pentru un tranzistor MOS cu canal p (PMOS)

Poarta unui tranzistor MOS are impedanţă foarte mare. Aceasta înseamnă că poarta este separată de sursă şi drenă printr-un material izolator cu rezistenţă foarte mare. Cu toate acestea, tensiunea aplicată pe poartă creează un câmp electric capabil să amplifice şi să diminueze intensitatea curentului dintre sursă şi drenă. Acesta este „**efectul de câmp**" din denumirea „MOSFET".

În ce priveşte curenţii dintre poartă şi sursă şi dintre poartă şi drenă, ei sunt aproape inexistenţi, indiferent de valoarea tensiunii de poartă. Rezistenţele dintre poartă şi celelalte terminate ale dispozitivului au valori extrem de mari, cu mult peste 1 MΩ. Printr-o astfel de rezistenţă circulă un curent foarte slab, cu valoarea tipică mai mică de 1 μA (10^{-6} A), numit ***curent rezidual***.

3.3.3 Circuitul CMOS inversor de bază

Tranzistoarele NMOS şi PMOS se utilizează împreună, în mod complementar, formând *logica CMOS*. Cel mai simplu circuit CMOS, inversorul logic, necesită câte un tranzistor din fiecare tip, conectate ca în fig. 3-10(a). Tensiunea sursei de alimentare, V_{DD}, se înscrie, tipic, în domeniul 2 ... 6 V şi este fixată, în cele mai multe cazuri, la 5,0 V, pentru compatibilitate cu circuitele TTL.

Teoretic, funcţionarea circuitului inversor CMOS poate fi caracterizată numai prin cele două stări specificate în tabelul din fig. 3-10(b):

1. V_{IN} este de 0,0 V. În acest caz, tranzistorul cu canal *n* din partea de jos, *T1,* este închis, întrucât are V_{gs}, egală cu 0, iar tranzistorul cu canal *p* din partea de sus, *T2,* este deschis, întrucât V_{gs}, aferentă lui are o valoare negativă mare (-5,0 V). Prin urmare, *T2* prezintă doar o rezistenţă mică între terminalul conectat la sursa de alimentare (V_{DD}, +5,0 V) şi terminalul de ieşire (V_{OUT}), tensiunea de ieşire fiind de 5,0 V.

2. V_{IN} este de 5,0 V. Acum, *TI* este deschis, întrucât V_{gs}, aferentă lui are o valoare pozitivă mare (+5,0 V), iar *T2* este închis, întrucât are V_{gs}, egală cu 0. În acest caz, *T1* prezintă o rezistenţă mică între terminalul de ieşire şi masă, tensiunea de ieşire fiind de 0V.

Modul de funcţionare descris mai sus arată că acest circuit are un comportament clar de inversor logic, deoarece un semnal de intrare de 0 V produce la ieşire un semnal de 5 V şi invers.

O altă metodă de a descrie funcţionarea unui CMOS este utilizarea modelului cu întrerupătoare. După cum se observă în fig. 3-11(a), tranzistorul cu canal *n* (cel din partea de jos) este echivalat cu modelul unui întrerupător normal deschis, iar tranzistorul cu canal p (cel din partea de sus) - cu un întrerupător normal închis. Aplicarea unei tensiuni HIGH are ca efect comutarea fiecărui întrerupător în starea opusă celei normale, cum se observă în fig-11(b).

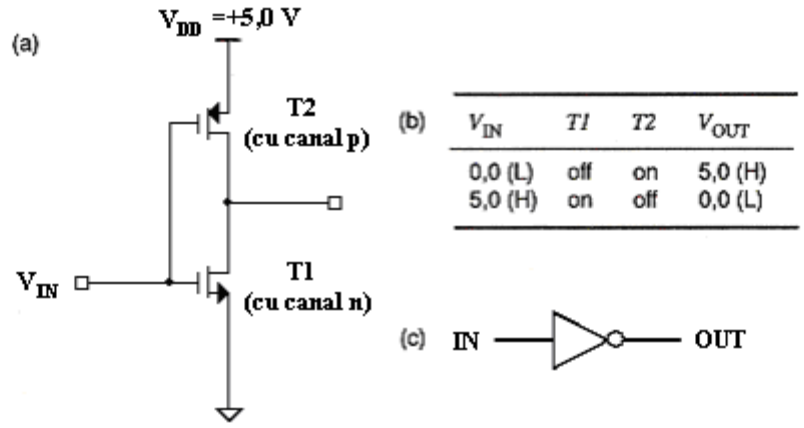


Figura 3-10 Inversor CMOS: (a) schema circuitului; (b) mobul de funcţionare; (c) simbolul logic

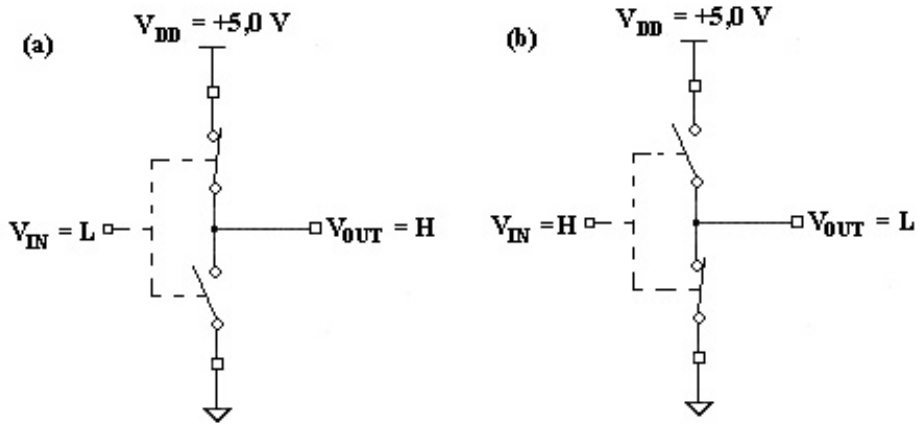


Figura 3-11 Modelul cu întrerupătoare pentru inversorul CMOS: (a) intrare LOW; (b) intrare HIGH

Modelul cu întrerupătoare face posibilă desenarea circuitelor CMOS într-un mod în care funcţia lor logică este pusă în evidenţă mai clar. După cum arată fig. 3-12, pentru tranzistoarele cu canal *n* şi cu canal *p* se folosesc diferite simboluri, care scot în evidenţă comportarea lor logică. Tranzistorul cu canal *n (T1)* este deschis şi curentul circulă între sursă şi drenă dacă pe poartă se aplică o tensiune HIGH; acest lucru pare firesc. Tranzistorul cu canal *p (T2)* se comportă invers. El este deschis cand tensiunea aplicată este LOW; cerculeţul inversor reprezentat pe poartă indică funcţionarea în sens invers a acestui tranzistor. În circuitele care au şi funcţii analogice (sau doar analogice) nu se folosesc aceste simboluri.

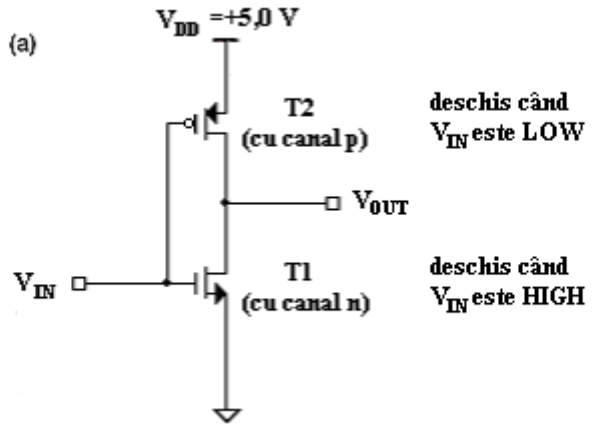
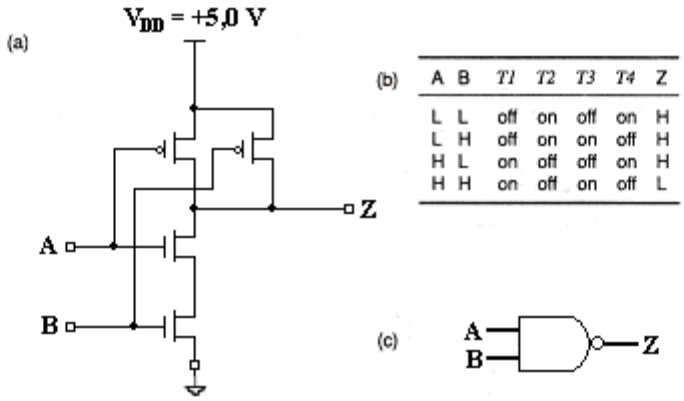


Figura 3-12 Funcţionarea logică a inversorului CMOS

3.3.4 Porţi CMOS NAND şi NOR

Cu CMOS se pot construi atât porţi **NAND**, cât şi porţi **NOR.** O poartă cu *k* intrări necesită *k* tranzistoare cu canal *n şi k* tranzistoare cu canal *p*. *În* fig. 3-13 este reprezentată o poartă CMOS **NAND** cu două intrări. Dacă oricare dintre intrări este LOW, ieşirea porţii NAND este conectată la V_{DD} printr-o impedanţă mică, prin tranzistorul cu canal p corespunzător, care este deschis, iar calea către masă este blocată de către tranzistorul pereche, cu canal *n,* care este închis. Dacă ambele intrări sunt HIGH, calea catre V_{DD} este blocată, iar ieşirea circuitului este conectată la masă printr-o impedanţă mică. În fig. 3-14 apare modelul cu întrerupătoare care descrie funcţionarea porţii **NAND**.



A	B	T1	T2	T3	T4	Z
L	L	off	on	off	on	H
L	H	off	on	on	off	H
H	L	on	off	off	on	H
H	H	on	off	on	off	L

Figura 3-13 Poartă CMOS NAND cu două intrări: (a) schema circuitului; (b) tabelul de adevăr; (c) simbolul logic

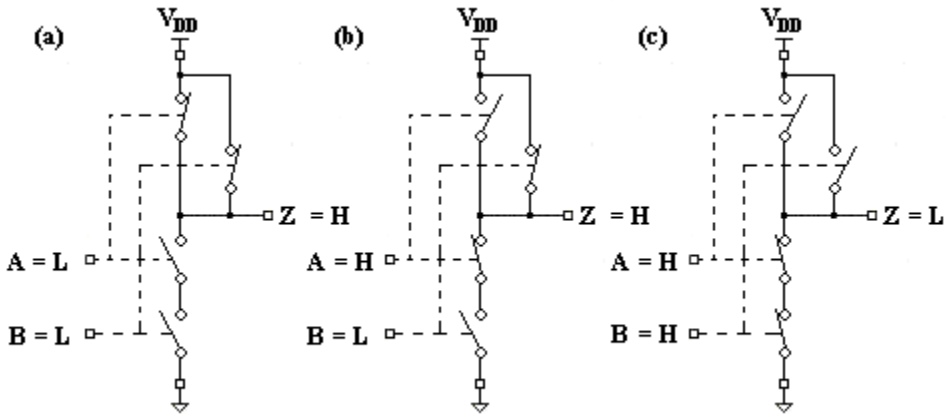


Figura 3-14 Modelul cu întrerupătoare pentru o poartă CMOS NAND cu două intrări: (a) ambele intrări LOW; (b) o intrare HIGH; (c) ambele intrări HIGH

În fig. 3-15 este reprezentată o poartă CMOS **NOR.** Dacă ambele intrări sunt LOW, ieşirea porţii este conectată la V_{DD} printr-o impedanţă mică, prin tranzistoarele cu canal p, care sunt „deschise", iar calea către masă este blocată de către tranzistoarele cu canal *n,* care sunt „închise". Dacă oricare dintre intrări este HIGH, calea către V_{DD} este blocată, iar ieșirea este conectată la masă printr-o impedanţă mică.

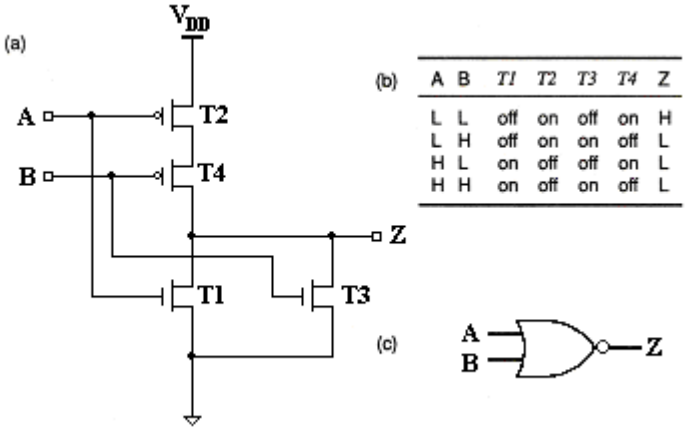
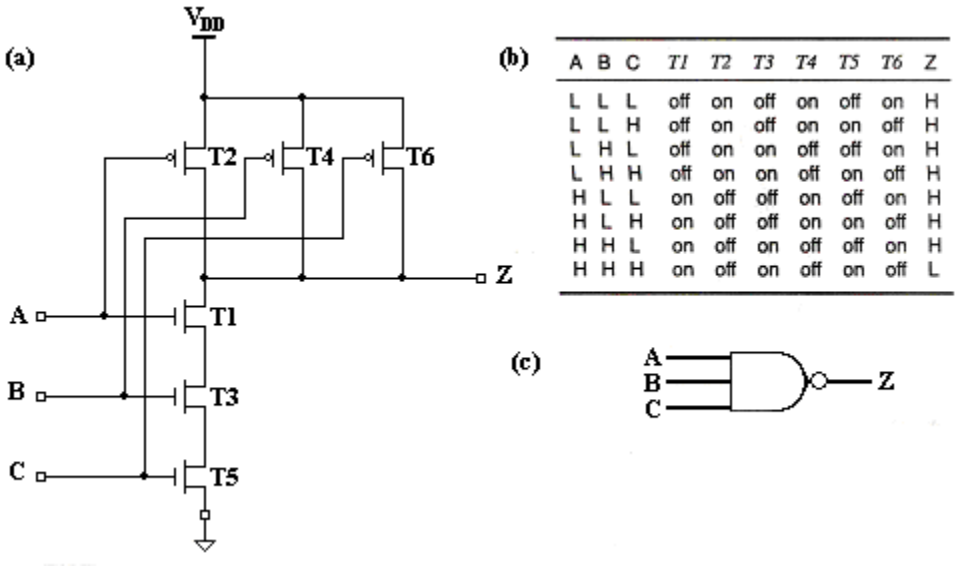


Figura 3-15 Poartă CMOS NOR cu două intrări: (a) schema circuitului; (b) tabelul de adevăr; (c) simbolul logic

3.3.5 Fan-in

Numărul de intrări pe care le poate avea o poartă aparţinând unei anumite familii de circuite logice este cunoscut sub denumirea, preluată din limba engleză, de ***fan-in***. Porţile CMOS cu mai mult de două intrări se pot obţine prin extinderea serie-paralel a schemelor din fig. 3-13 şi 3-15 într-un mod corespunzător. De exemplu, în fig. 3-16 este prezentată o poartă CMOS **NAND** cu trei intrări.



A	B	C	T1	T2	T3	T4	T5	T6	Z
L	L	L	off	on	off	on	off	on	H
L	L	H	off	on	off	on	on	off	H
L	H	L	off	on	on	off	off	on	H
L	H	H	off	on	on	off	on	off	H
H	L	L	on	off	off	on	off	on	H
H	L	H	on	off	off	on	on	off	H
H	H	L	on	off	on	off	off	on	H
H	H	H	on	off	on	off	on	off	L

Figura 3-16 Poartă CMOS NAND cu trei intrări: (a) schema circuitului; (b) tabelul de adevăr; (c) simbolul logic

În principiu, se pot proiecta porţi CMOS **NAND** şi **NOR** cu un număr foarte mare de intrări. Practic însă, rezistenţele de conducţie ale tranzistoarelor conectate în serie, care şi adună, limitează numărul de intrări al porţilor CMOS, de obicei la 4 pentru porţile **NOR** şi la 6 pentru porţile **NAND**.

Pentru un număr mai mare de intrări, este posibil ca proiectantul circuitului să mărească dimensiunile fizice ale tranzistoarelor înseriate pentru a micşora rezistenţele acestora şi întârzierea la comutaţie corespunzătoare. În fig. 3-17 puteţi vedea structura logică a unei porţi CMOS **NAND** cu 8 intrări. Întârzierea totală introdusă de un **NAND** cu 4 intrări, un **NOR** cu 2 intrări şi un inversor este, în mod normal, mai mică decât întârzierea introdusă de un circuit NAND cu 8 intrări la acelaşi nivel.

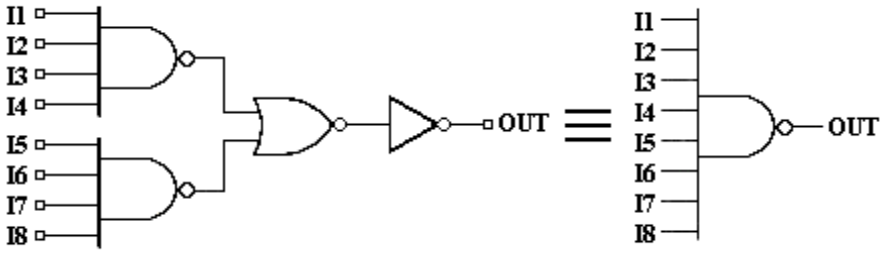


Figura 3-17 Schema logică echivalentă pentru o structură internă de poartă CMOS NAND cu 8 intrări

3.3.6 Porţi neinversoare

În familia de circuite logice CMOS, precum şi în alte familii, cele mai simple porţi sunt inversoarele, urmate, ca nivel de complexitate, de porţile

NAND şi **NOR**. Inversarea logică apare „de la sine” şi, în mod normal, este imposibil de proiectat o poartă neinversoare cu mai puţine tranzistoare decât una inversoare.

Etajele tampon neinversoare CMOS şi porţile **AND** şi **OR** se obţin prin conectarea unui inversor la ieşirea porţilor inversoare respective. De pildă, fig. 3-18 prezintă un etaj tampon neinversor, iar fig. 3-19, o poartă **AND**. Prin combinarea schemei din fig. 3-15(a) cu un inversor se obţine o poartă **OR**.

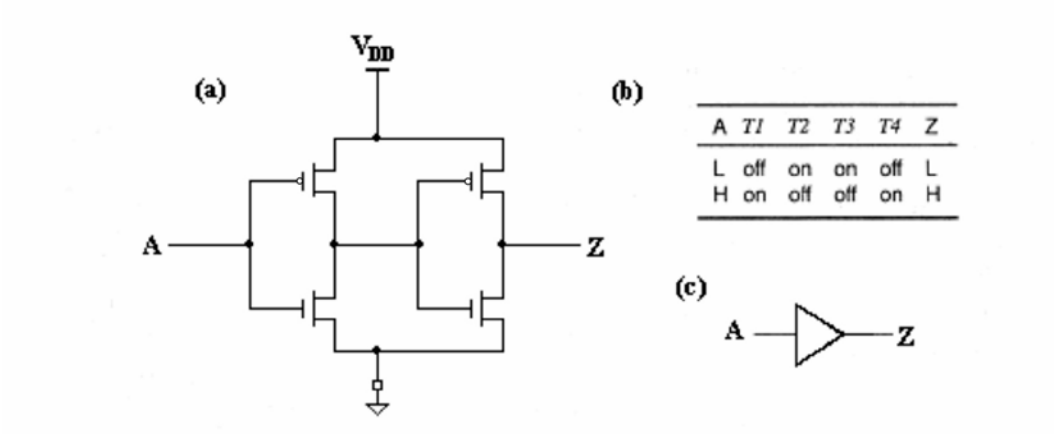


Figura 3-18 Circuit tampon neinversor CMOS: (a) schema circuitului; (b) tabelul de adevăr; (c) simbolul logic

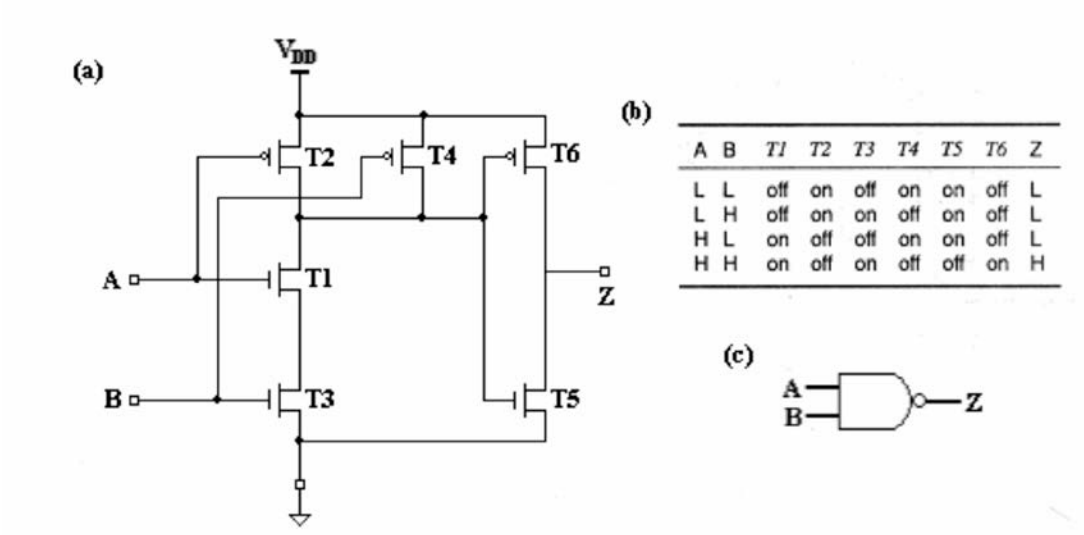


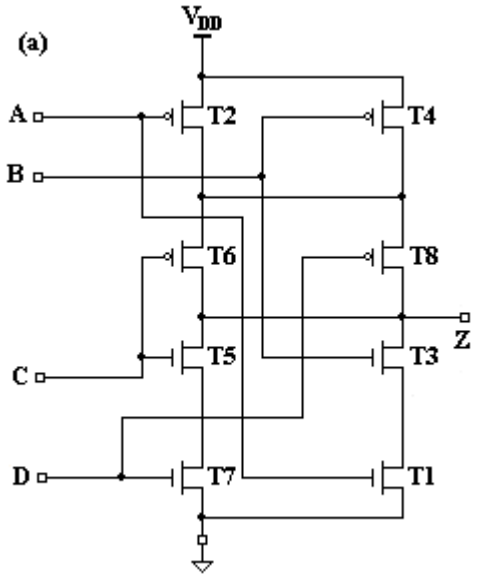
Figura 3-19 Poartă CMOS AND cu două intrări: (a) schema circuitului; (b) tabelul de adevăr; (c) simbolul logic

3.3.7 Porţi CMOS AND-OR-Inversor şi OR-AND-Inversor

Circuitele CMOS pot realiza funcţii logice cu două niveluri folosind un singur „nivel” de tranzistoare. De exemplu, circuitul CMOS din fig. 3-20(a) este *o poartă* ***AND-OR-Inversor (AOI)*** *cu* 2 cu 2 intrări. Tabelul funcţiei pentru acest circuit este cel din fig. (b), iar schema logică a funcţiei, implementată cu

porţi **AND** şi **NOR**, este dată în fig. 3-21. Circuitului i se pot adăuga şi elimina tranzistoare pentru a obţine o funcţie **AOI** cu un alt număr de porţi **AND** şi cu un alt număr de intrări pe poartă **AND**.

Conţinutul coloanelor *T1 ... T8* din fig. 3-20(b) depinde numai de semnalul de intrare aplicat pe poarta tranzistorului respectiv. Valorile din ultima coloană au fost obţinute prin examinarea fiecărei combinaţii de intrare şi stabilind dacă ŞI este conectată la V_{DD} şi la masă prin tranzistoarele aflate în conducţie pentru combinaţia de intrare respectivă. Nici un tranzistor din cele care realizează funcţia ŞI nu este niciodată conectat atât la V_{DD}, cât şi la masă, indiferent de combinaţia de intrare; o asemenea situaţie ar da la ieşire o valoare logică nedeterminată, aflată între LOW şi HIGH, iar circuitul de ieşire ar consuma excesiv putere din cauza conexiunii de impedanţă mică dintre V_{DD} şi masă.



(b)

A	B	C	D	T1	T2	T3	T4	T5	T6	T7	T8	Z
L	L	L	L	off	on	off	on	off	on	off	on	H
L	L	L	H	off	on	off	on	off	on	on	off	H
L	L	H	L	off	on	off	on	on	off	off	on	H
L	L	H	H	off	on	off	on	on	off	on	off	L
L	H	L	L	off	on	on	off	off	on	off	on	H
L	H	L	H	off	on	on	off	off	on	on	off	H
L	H	H	L	off	on	on	off	on	off	off	on	H
L	H	H	H	off	on	on	off	on	off	on	off	L
H	L	L	L	on	off	off	on	off	on	off	on	H
H	L	L	H	on	off	off	on	off	on	on	off	H
H	L	H	L	on	off	off	on	on	off	off	on	H
H	L	H	H	on	off	off	on	on	off	on	off	L
H	H	L	L	on	off	on	off	off	on	off	on	L
H	H	L	H	on	off	on	off	off	on	on	off	L
H	H	H	L	on	off	on	off	on	off	off	on	L
H	H	H	H	on	off	on	off	on	off	on	off	L

Figura 3-20 Poartă CMOS AND-OR-inversor: (a) schema circuitului; (b) tabelul de adevăr

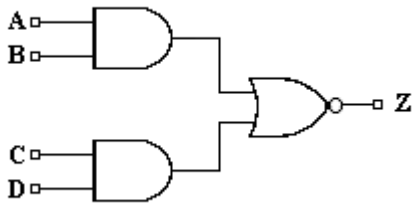
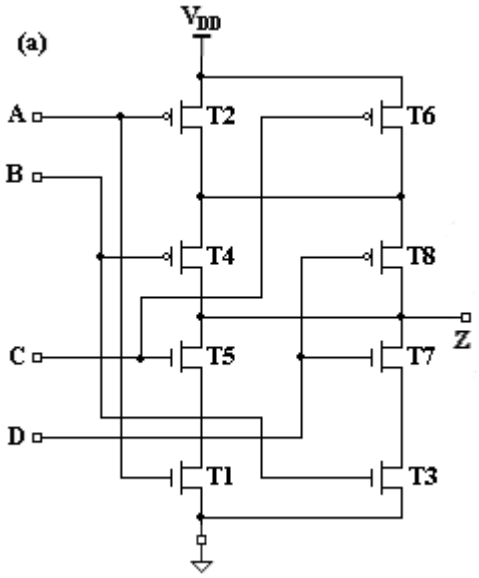


Figura 3-21 Schema logică a unei porţi CMOS AND-OR-inversor

Se poate proiecta, de asemenea, un circuit care să realizeze funcţia OR-AND-Inversor. De exemplu, circuitul CMOS din fig. 3-22(a) este o poartă ***OR-AND-inversor (OAI)*** *cu 2 x* 2 intrări. Tabelul funcţiei pentru acest circuit este cel din fig. (b); valorile din fiecare coloană au fost stabilite prin acelaşi procedeu

ca pentru poarta CMOS **AOI**. Schema logică a funcţiei **OAI**, implementată cu porţi **OR** şi **NAND**, este dată în fig. 3-23.



(b)

A	B	C	D	T1	T2	T3	T4	T5	T6	T7	T8	Z
L	L	L	L	off	on	off	on	off	on	off	on	H
L	L	L	H	off	on	off	on	off	on	on	off	H
L	L	H	L	off	on	off	on	on	off	off	on	H
L	L	H	H	off	on	off	on	on	off	on	off	H
L	H	L	L	off	on	on	off	off	on	off	on	H
L	H	L	H	off	on	on	off	off	on	on	off	L
L	H	H	L	off	on	on	off	on	off	off	on	L
L	H	H	H	off	on	on	off	on	off	on	off	L
H	L	L	L	on	off	off	on	off	on	off	on	H
H	L	L	H	on	off	off	on	off	on	on	off	L
H	L	H	L	on	off	off	on	on	off	off	on	L
H	L	H	H	on	off	off	on	on	off	on	off	L
H	H	L	L	on	off	on	off	off	on	off	on	H
H	H	L	H	on	off	on	off	off	on	on	off	L
H	H	H	L	on	off	on	off	on	off	off	on	L
H	H	H	H	on	off	on	off	on	off	on	off	L

Figura 3-22 Poartă CMOS OR-AND-inversor: (a) schema circuitului; (b) tabelul de adevăr

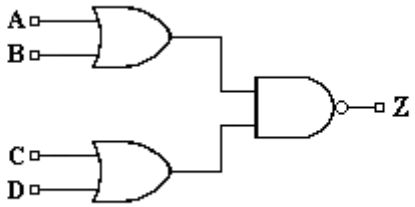


Figura 3-23 Schema logică a unei porţi CMOS OR-AND-inversor

Viteza şi alte caracteristici electrice ale porţilor CMOS **AOI** şi **OAI** sunt comparabile aproximativ cu cele ale unei singure porţi CMOS **NAND** sau **NOR**. În consecinţă, aceste porţi sunt preferabile, deoarece realizează funcţii logice cu două niveluri (**AND-OR** şi **OR-AND**) introducând o întârziere corespunzătoare unui singur nivel.

3.4 Alte structuri CMOS de intrare şi de ieşire

Proiectanţii de circuite au adus nenumărate modificări circuitului CMOS de bază, pentru a realiza porţi dedicate anumitor aplicaţii. Secţiunea de faţă descrie câteva dintre cele mai cunoscute variante de structuri CMOS de intrare şi de ieşire.

3.4.1 Porţi de transmisie

O pereche de tranzistoare, unul cu canal *p* şi celalalt cu canal *n,* se pot conecta împreună pentru a forma un comutator comandat logic. Un asemenea circuit se numeşte ***poarta de transmisie* CMOS** şi este prezentat în fig. 3-24.

O poartă de transmisie este concepută să funcţioneze astfel încât semnalele de la intrarea sa, EN şi EN_L, să aibă întotdeauna niveluri opuse. Când EN este HIGH şi EN_L este LOW, între punctele A şi B apare o conexiune de impedanţă mică (cam de 2 ... 5 Ω). Când EN este LOW şi EN_L este HIGH, conexiunea dintre punctele A şi B se întrerupe.

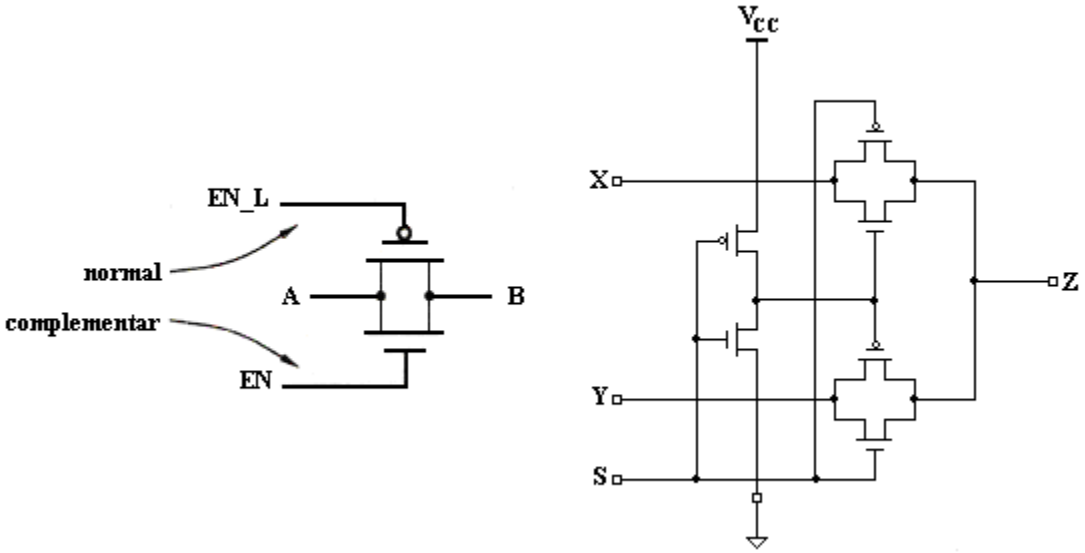


Figura 3-24 Poartă de transmisie CMOS; Figura 3-25 Multiplexor cu două intrări realizat cu porţi de transmisie CMOS

Când poarta de transmisie este activată, timpul de propagare de la A la B (sau invers) este foarte scurt. Datorită întârzierii reduse pe care o introduc şi simplităţii conceptuale, porţile de transmisie sunt utilizate frecvent în structura internă a dispozitivelor CMOS integrate la scară largă, ca multiplexoarele şi circuitele basculante. De exemplu, fig. 3-25 arată cum se pot folosi porţile de transmisie la realizarea unui multiplexor cu două intrări. Când S este LOW, intrarea X se conectează la ieşirea Z. Când S este HIGH, Y se conectează la Z.

3.4.2 Intrări cu trigger Schmitt

Un trigger Schmitt este un tip de circuit care utilizează o reacţie internă pentru a deplasa pragul de comutaţie în funcţie de sensul comutării de la intrare, de la LOW la HIGH sau de la HIGH la LOW.

De exemplu, în fig. 3-26, intrarea cu trigger Schmitt a unui inversor se află iniţial la 0 V, adică într-o stare bine definita că LOW. Atunci, ieşirea se află în starea HIGH, la aproape 5,0 V. Când tensiunea de intrare creşte progresiv,

ieşirea nu va trece în LOW până când tensiunea de intrare nu va ajunge la aproximativ 2,9 V. Dar, după ce ieşirea a trecut în LOW, ea nu va mai reveni în HIGH până când tensiunea de intrare nu va scădea la aproximativ 2,1 V. Prin urmare, pragul de comutaţie pentru tranziţii ascendente la intrare, notat cu V_{T+} este de aproximativ 2,9 V, iar pragul de comutaţie pentru tranziţii de intrare descendente, notat cu V_{T-}, este de aproximativ 2,1 V.

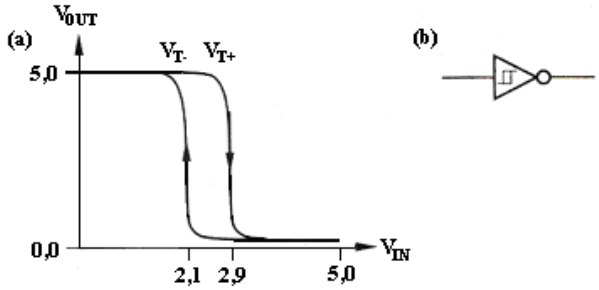


Figura 3-26 Inversor cu trigger Schmitt: (a) caracteristica de transfer intrare-ieşire; (b) simbolul logic

Diferenţa dintre cele două praguri se cheamă ***histerezis***. Inversorul cu trigger Schmitt prezintă un histerezis de aproximativ 0,8 V.

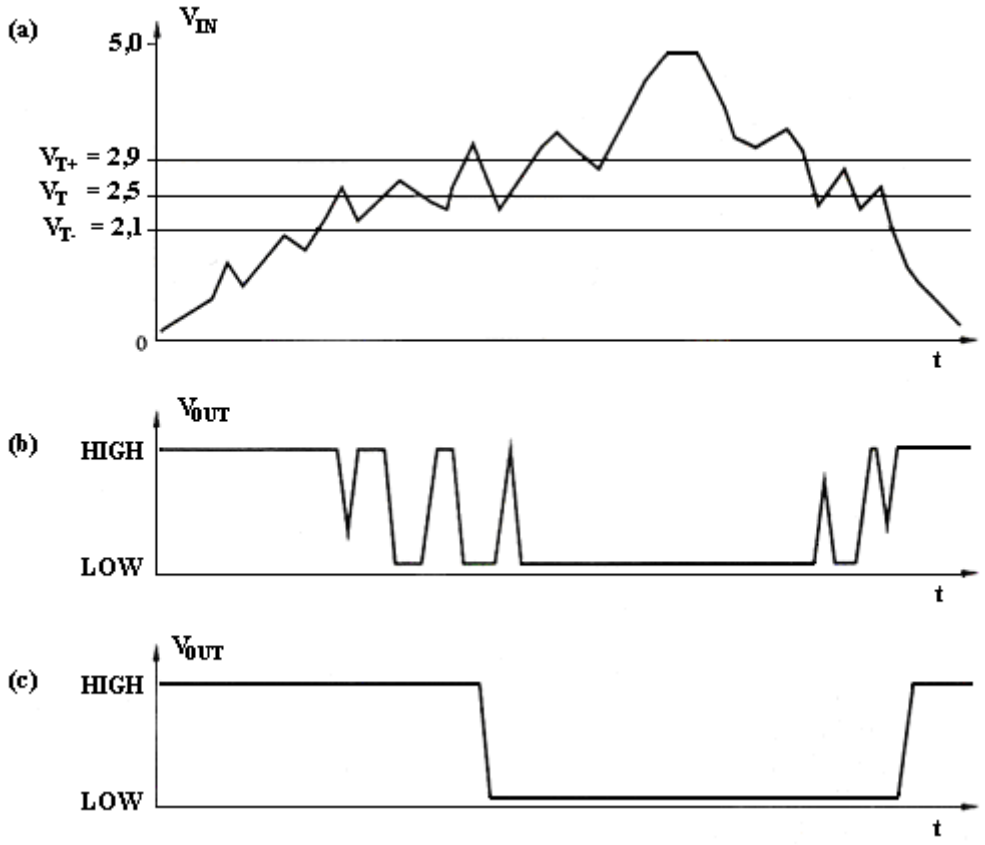


Figura 3-27 Funcţionarea dispozitivului cu semnale de intrare cu tranziţii lente: (a) semnal cu tranziţii lente şi zgomot suprapus; (b)semnal de ieşire al unui inversor obişnuit; (c) semnal de ieşire al unui inversor cu histerezis de 0,8 V

Comparativ, în fig. 3-27(a) avem un semnal de intrare cu timpi de creştere şi de scădere de valoare mare, peste care se suprapune un zgomot de aproximativ 0,5 V. Răspunsul la zgomot al inversorului obişnuit este cel prezentat în fig. (b), cu multe comutări la ieşire, corespunzătoare tuturor depăşirilor de prag de la intrare din cauza suprapunerii zgomotului. Însă, aşa cum se vede în fig. (c), un inversor cu trigger Schmitt nu reacţionează la zgomot deoarece histerezisul sau este mai mare decât amplitudinea zgomotului.

3.4.3 Ieşiri cu trei stări

Ieşirile logice au două stări normale, LOW şi HIGH, corespunzătoare valorilor logice 0 şi 1. Unele ieşiri prezintă însă şi o a treia stare, care nu este câtuşi de puţin o stare logică, numită ***stare de înaltă impedanţă***, ***Hi-Z*** sau ***flotantă***. Într-o asemenea stare, ieşirea se comportă de parcă nici n-ar fi conectată cu restul circuitului, cu excepţia prezenţei unui curent rezidual slab care poate circula către sau dinspre borna de ieşire. Prin urmare, o ieşire se poate afla într-una dintre cele trei stări: 0 logic, 1 logic şi Hi-Z.

Dispozitivele prevăzute cu astfel de ieşiri au o intrare suplimentară, numită, de obicei, „activarea ieşirii” (output enable) sau, dezactivarea „ieşirii” (output disable), pentru trecerea ieşirii (ieşirilor) dispozitivului în starea de înaltă impedanţă.

În fig. 3-28(a) apare schema unui *circuit tampon CMOS cu trei stări.* Pentru simplificarea schemei, funcţiile interne **NAND**, **NOR** şi inversoare au fost reprezentate prin simbolurile funcţionale, nu prin circuitele cu tranzistoare; în realitate, ele utilizează în total 10 tranzistoare. Aşa cum reiese din tabelul funcţiei, din fig. (b), când intrarea de activare (EN) este LOW, ambele tranzistoare de ieşire sunt închise, iar ieşirea se află în starea Hi-Z. În caz contrar, ieşirea se află într-una din stările HIGH sau LOW, după cum comandă intrarea „de date” A. În simbolurile logice ale circuitelor tampon şi ale porţilor cu trei stări, intrarea de activare se reprezintă, de obicei, în partea superioară, ca în fig. (c).

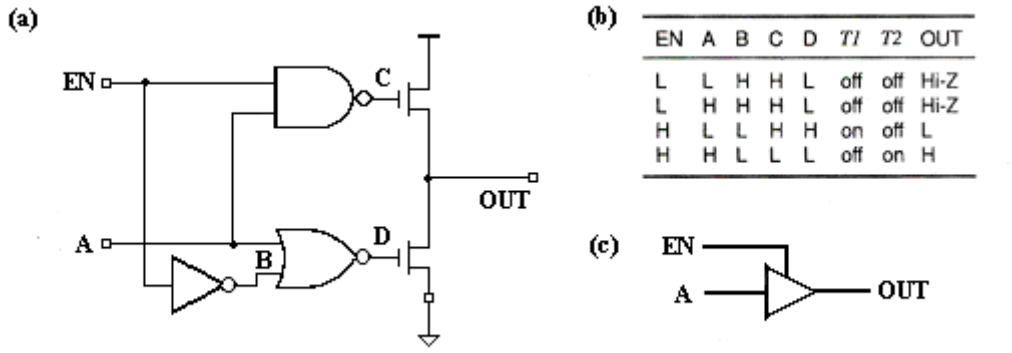


Figura 3-28 Circuit tampon CMOS cu trei stări: (a) schema circuitului; (b) tabelul de adevăr; (c) simbolul logic

În practică, circuitul de comandă pentru cele trei stări poate fi diferit faţă de cel prezentat aici, pentru a asigura comportarea dinamică dorită tranzistoarelor de ieşire în timpul tranziţiilor către şi din starea Hi-Z.

3.4.4 Circuitele CMOS cu drenă în gol

Se spune că tranzistoarele cu canal *p* din structurile de ieşire CMOS efectuează o *comutare activă,* deoarece, practic, ele sunt cele ce determină creşterea tensiunii de ieşire în tranziţiile LOW-HIGH. Aceste tranzistoare nu se includ în porţile cu *ieşiri cu drena în gol,* cum este poarta NAND din fig. 3-29(a). Drena tranzistorului cu canal *n* de deasupra nu este conectată intern, deci când ieşirea nu este în starea LOW, este „în gol", cum arată fig. (b). Rombul subliniat din simbolul din fig. (c) se foloseşte uneori pentru a indica o ieşire cu drena în gol. O configuraţie similară, numită „ieşire cu colectorul în gol", se regăseşte la familiile de circuite logice TTL.

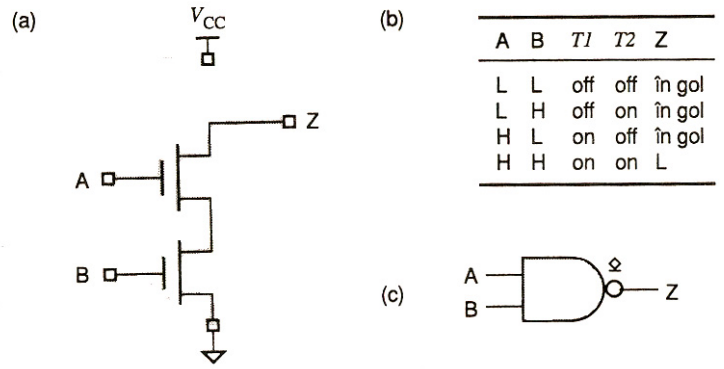


Figura 3-29 Poartă CMOS NAND cu drena în gol: (a) schema circuitului; (b) tabelul funcţiei; (c) simbolul logic

O ieşire cu drena în gol necesită un *rezistor de forţare în HIGH* exterior pentru a asigura *comutarea pasivă* către nivelul HIGH. De exemplu, fig. 3-30 prezintă o poartă CMOS NAND cu drena în gol, cu rezistorul de forţare în HIGH şi cu sarcina comandată.

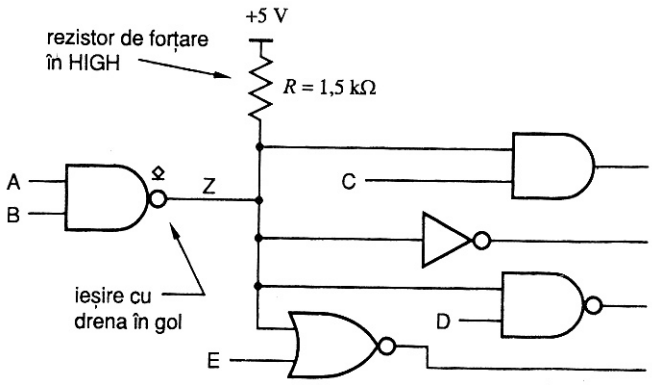


Figura 3-30 Poartă CMOS NAND cu drena în gol şi sarcina comandată de ea

Pentru realizarea celei mai mari viteze posibile, rezistorul de forţare în HIGH aferent unei ieşiri cu drena în gol trebuie să aibă o valoare cât mai mică posibil; în acest mod, constanta de timp *RC* are valoarea minimă pentru tranziţiile LOW-HIGH (timpul de creştere). Însă valoarea rezistorului de forţare în HIGH nu poate fi aleasă arbitrar mică; rezistenţa minimă este impusă de curentul maxim absorbit de ieşirea cu drena în gol, I_{OLmax}. De exemplu, pentru seriile CMOS HC şi HCT, I_{OLmax} este de 4 mA, iar rezistorul de forţare în HIGH nu poate avea o valoare mai mică de 5,0 V/4 mA, adică 1,25 kΩ. Având în vedere că ordinul de mărime al acestei rezistenţe este mai mare decât cel corespunzător rezistenţei de conducţie a tranzistoarelor cu canal *p* dintr-o poartă CMOS standard, înseamnă că tranziţiile LOW-HIGH ale semnalului de ieşire sunt mult mai lente în cazul unei porţi cu drena în gol decât în cazul unei porţi standard, cu comutare activă. În pofida timpului mare de creştere, ele şi-au dovedit utilitatea în cel puţin trei aplicaţii: comanda diodelor luminescente (LED-uri) şi a altor dispozitive; realizarea circuitelor logice cablate; comanda magistralelor cu mai multe surse.

3.5 Familii de circuite logice CMOS

Prima familie CMOS cu succes comercial a fost *seria CMOS 4000*. Deşi circuitele din această serie se bucurau de avantajul unei disipări de putere reduse, erau destul de lente şi greu de interconectat cu circuite bipolare, TTL. De aceea, seria 4000 a fost detronată, în majoritatea aplicaţiilor, de următoarele familii CMOS, care prezentau mai multe avantaje.

Toate dispozitivele CMOS la care vom face referire sunt desemnate prin coduri de forma „74FAMxx", „FAM" reprezentând denumirea - formată din litere - a familiei, iar xx - un indicativ numeric al funcţiei. Dispozitivele aparţinând unor familii diferite, dar cu aceeaşi valoare xx realizează aceeaşi funcţie. De exemplu, 74HC30, 74HCT30, 74AC30, 74ACT30 şi 74AHC30 sunt porţi **NAND** cu 8 intrări.

Prefixul „74" este doar un număr folosit de unul dintre primii şi cei mai cunoscuţi producători de dispozitive TTL, Texas Instruments. Când un cod este însoţit de prefixul „54", înseamnă că dispozitivul desemnat este conceput pentru a lucra într-o gamă mai largă de temperaturi sau tensiuni de alimentare, pentru aplicaţii militare.

3.5.1 Familiile HC şi HCT

Primele două familii CMOS din seria 74 sunt *HC (High-speed CMOS* – CMOS de mare viteză) şi *HCT (High-speed CMOS, TTL compatible* – CMOS de mare viteză, compatibil cu TTL). În comparaţie cu prima familie, 4000, atât HC, cât şi HCT au viteză mai mare şi pot absorbi şi furniza curenţi mai mari.

Dispozitivele din familia HCT se alimentează de la o sursa V_{CC} de 5 V, putând fi interconectate cu dispozitive TTL, care se alimentează tot cu 5 V.

Familia HC a fost realizată în vederea utilizării optime în sisteme formate exclusiv cu circuite logice CMOS, tensiunea de alimentare putând lua orice valoare cuprinsă între 2 V şi 6 V. Pentru obţinerea unor viteze mai mari se folosesc tensiuni de alimentare mai ridicate, iar la tensiuni mai scăzute se disipă mai puţină putere. Alimentarea la tensiuni mici duce la obţinerea unor randamente foarte bune, deoarece majoritatea puterii disipate de un dispozitiv CMOS este proporţională cu pătratul tensiunii.

Dispozitivele HC nu prea sunt compatibile cu TTL chiar când sunt alimentate la 5 V. Mai precis, circuitele HC sunt concepute pentru a recunoaşte nivelurile de intrare CMOS HC. În fig. 3-31(a) sunt reprezentate nivelurile de intrare şi de ieşire caracteristice dispozitivelor CMOS HC, pentru o tensiune de alimentare de 5,0 V. Nivelurile de ieşire ale dispozitivelor TTL nu corespund cu exactitate aceloraşi domenii, de aceea pentru dispozitivele HCT s-au ales niveluri de intrare diferite - cele din fig. (b).

Aceste niveluri sunt impuse prin procesul de fabricaţie, tranzistoarele fiind realizate cu alte praguri de comutare, ceea ce duce la obţinerea unor caracteristici de transfer diferite, pe care le puteţi observa în fig. 3-32.

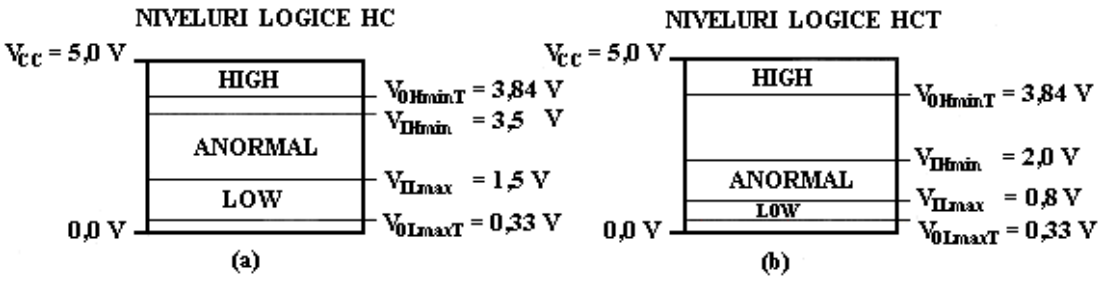


Figura 3-31 Nivelurile de intrare şi de ieşire la dispozitivele CMOS alimentate cu 5 V: (a) la familia HC; (b) la familia HCT

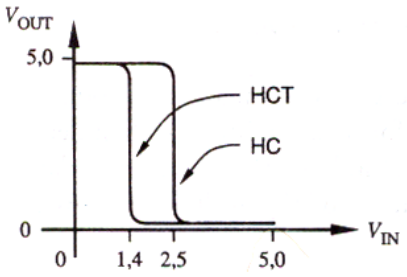


Figura 3-32 Caracteristicile de transfer ale circuitelor HC si HCT, în condiţii de funcţionare tipice

3.5.2 VHC şi VHCT

Începând cu anii 1980, apoi 1990, au fost realizate câteva noi familii CMOS. Două dintre cele mai recente şi, probabil, cele mai versatile sunt *VHC (Very High-Speed CMOS* - CMOS de viteză foarte mare) şi *VHCT (Very High-Speed CMOS, TTL compatible* - CMOS de viteză foarte mare, compatibile cu TTL). Dispozitivele din aceste familii lucrează la viteze aproape duble faţă de HC/HCT, menţinându-şi însă compatibilitatca cu predecesoarele lor. Asemenea familiilor HC şi HCT, familiile VHC şi VHCT se deosebesc doar prin nivelurile de intrare pe care le recunosc; caracteristicile lor de ieşire sunt identice.

Tot asemenea familiilor HC/HCT, VHC/VHCT prezintă o *simetrie de comandă la ieşire.* Aceasta înseamnă că ieşirea poate absorbi sau furniza acelaşi curent, fiind la fel de „eficace" în ambele stări. Alte familii de circuite logice, cum sunt FCT şi TTL mai recente, prezintă *asimetrie de comandă la ieşire*; ele pot absorbi mai mult curent în starea LOW decât pot furniza în starea HIGH.

3.5.3 Caracteristicile electrice ale familiilor HC, HCT, VHC şi VHCT

Valorile parametrilor sunt cele corespunzătoare unei tensiuni de alimentare nominale de 5 V, însă dispozitivele pot funcţiona (cu unele corecţii ale valorilor) cu orice tensiune de alimentare cuprinsă între 2 V şi 5,5 V (6 V pentru HC/HCT).

Dispozitivele de uz general (seria 74) sunt destinate funcţionării la temperaturi din gama 0°C...70°C, iar cele de uz militar (seria 54), în gama -55°C... 125°C. Specificaţiile din tabelul 3-3 se referă la funcţionarea la 25°C.

Majoritatea dispozitivelor aparţinând aceleiaşi familii de circuite logice se caracterizează prin parametri de intrare şi de ieşire identici, deosebindu-se, de obicei, numai prin consumul de putere şi prin timpul de propagare. În tabelul 3-3 sunt prezentaţi parametrii aferenţi unei porţi **NAND** cu două intrări 74x00 şi unui decodor cu 3 intrări şi 8 ieşiri 74x138, din familiile HC, HCT, VHC şi VHCT. Poarta **NAND** cu două intrări 00 este cel mai mic bloc logic structural din fiecare familie, iar decodorul '138 este un dispozitiv de complexitate medie, conţinând echivalentul a aproximativ 15 porţi **NAND**.

Tabel 3-3 Caracteristici de viteză şi de putere ale familiilor CMOS, pentru tensiunea de alimentare de 5 V

				Familia			
Parametrul	**Cod**	**Simbol**	**Conditii**	**HC**	**HCT**	**VHC**	**VHCT**
Timp de propagare	'00	t_{PD}		9	10	5,2	5,5
tipic (ns)	'138			18	20	7,2	8,1
Curent static de	'00	I_{CC}	V_{IN} = 0 sau V_{CC}	2,5	2,5	5,0	5,0
alimentare (μA)	'138		V_{IN} = 0 sau V_{CC}	40	40	40	40
Putere statica disipata	'00		V_{IN} = 0 sau V_{CC}	0,0125	0,0125	0,025	0,025
(mw)	138		V_{IN} = 0 sau V_{CC}	0,2	0,2	0,2	0,2
Capacitate de disipare	'00	C_{PD}		22	15	19	17

a puterii (pF)	'138	C_{PD}		55	51	34	49
Putere dinamică disipată (mW/MHz)	'00			0,55	0,38	0,48	0,43
	138			1,38	1,28	0,85	1,23
	'00		f = 100 kHz	0,068	0,050	0,073	0,068
	'00		f = 1 MHz	0,56	0,39	0,50	0,45
Putere disipată totală (mW)	'00		f = 10 MHz	5,5	3,8	4,8	4,3
	138		f = 100 kHz	0,338	0,328	0,285	0,323
	138		f = 1 MHz	1,58	1,48	1,05	1,43
	'138		f = 10 MHz	14,0	13,0	8,7	12,5
	'00		f = 100 kHz	0,61	0,50	0,38	0,37
	'00		f = 1 MHz	5,1	3,9	2,6	2,5
Produs viteză-putere (pJ)	'00		f = 10 MHz	50	38	25	24
	'138		f = 100 kHz	6,08	6,55	2,05	2,61
	'138		f = 1 MHz	28,4	29,5	7,56	11,5
	'138		f = 10 MHz	251	259	63	101

Prima linie a tabelului 3-3 prezintă valorile timpului de propagare. Se observă că HC şi HCT au aproape aceeasi viteză ca şi LS-TTL, iar VHC şi VHCT sunt aproape tot atât de rapide ca ALS TTL. La '138, timpul de propagare este ceva mai lung decât la '00, deoarece semnalele trebuie să parcurgă cele trei sau patru niveluri de porţi din interior.

Liniile a doua şi a treia ale tabelului arată că puterea statică disipată este practic nulă, mult sub un miliwatt (mW).

Ultima linie a tabelului, *produsul viteză putere,* indică, pur şi simplu, produsul dintre timpul de propagare şi consumul de putere caracteristic unei porţi obişnuite, unitatea de măsură fiind pJ (picojoule). Un joule este o unitate de energie, deci produsul viteză-putere este o evaluare a randamentului, măsurând câtă energie consumă o poartă logică pentru a-şi schimba starea de ieşire. Evident că este de preferat un consum de energie cât mai mic.

Tabel 3-4 Parametrii de intrare ai familiilor CMOS pentru VCC de 4,5V … 5,6 V

Parametrul	**Simbol**	**Conditii**	**Familia**			
			HC	**HCT**	**VHC**	**VHCT**
Curent rezidual de intrare (µA)	I_{Imax}	V_i = oricare	± 1	± 1	± 1	± 1
Capacitate maximă de intrare (pF)	C_{INmax}		10	10	10	10
Tensiune de intrare la nivelul LOW (V)	V_{ILmax}		1,35	0,8	1,35	0,8
Tensiune de intrare la nivelul HIGH (V)	V_{IHmin}		3,85	2,0	3,85	2,0

Tabelul 3-4 prezintă valorile parametrilor de intrare tipice pentru dispozitivele CMOS din fiecare familie. Unele valori sunt specificate în ipoteza că tensiunea de alimentare de 5 V are o toleranţă de ± 10%, adică V_{CC} poate avea orice valoare cuprinsă între 4,5 V şi 5,5 V.

I_{Imax} Curentul maxim de intrare, oricare ar fi valoarea tensiunii de intrare. Din tabel reiese că spre sau dinspre o intrare CMOS circulă un curent de maximum 1µA, oricare ar fi valoarea tensiunii de intrare. Cu alte cuvinte, intrările CMOS constituie o sarcină de c.c. neglijabilă pentru circuitele care le comandă.

C_{INmax} Capacitatea maximă a unei intrări. Această valoare poate fi luată în consideraţie la calcularea sarcinii de c.a. văzută la o ieşire ce comandă, pe lângă alte intrări, şi intrarea respectivă. Majoritatea producătorilor mai menţionează încă o capacitate tipică de intrare, de valoare mai mică, aproximativ 5 pF, cu care, se poate estima satisfacător valoarea sarcinii de c.a.

V_{ILmax} Tensiunea maximă recunoscută garantat de o intrare ca nivel LOW. Valorile diferă la HC/VHC faţă de cele de la HCT/VHCT. La dispozitivele destinate a fi interconectate cu CMOS, valoarea de 1,35 V reprezintă 30% din tensiunea minimă de alimentare, în timp ce valoarea caracteristică dispozitivelor ce pot fi interconectate cu TTL este de 0,8 V, pentru asigurarea compatibilităţii.

V_{IHmin} Tensiunea minimă recunoscută garantat de o intrare ca nivel HIGH. La dispozitivele destinate a fi interconectate cu CMOS, valoarea de 3,85 V reprezintă 70% din tensiunea maximă de alimentare, în timp ce valoarea caracteristică dispozitivelor ce pot fi interconectate cu TTL, este de 2,0 V, pentru asigurarea compatibilităţii.

Pentru ieşirile CMOS compatibile cu TTL se specifică, de obicei, două seturi de parametri de ieşire; trebuie luat în consideraţie unul dintre cele două seturi, în funcţie de sarcina conectată la ieşire. În cazul unei *sarcini CMOS,* curentul continuu pe care trebuie să-l absoarbă şi să-l furnizeze ieşirea este foarte mic, de 20µA pentru HC/HCT şi de 50µA pentru VHC/VHCT. Desigur, situaţia aceasta se întâlneste atunci când ieşirile CMOS comandă intrări exclusiv CMOS. Cu sarcină CMOS, tensiunea de ieşire a unei ieşiri CMOS se menţine la o diferenţă de maximum 0,1 V faţă de barele de alimentare, 0 şi Vcc. (Valorile din tabel corespund cazului celui mai defavorabil, V_{CC} = 4,5 V; în consecinţă, V_{OHminC} = 4,4 V).

O sarcină TTL necesită un consum mai mare de curent absorbit şi furnizat, de până la 4 mA pentru ieşirile de HC/HCT şi până la 8 mA pentru ieşirile de VHC/VHCT. În acest caz, pe tranzistoarele în conductie din circuitul de ieşire apare o cădere de tensiune mai mare, însă tensiunea de ieşire se menţine garantat în domeniul normal al nivelurilor de ieşire TTL.

Tabel 3-5 Parametrii de ieşire ai familiilor CMOS pentru VCC de 4,5V … 5,6 V

			Familia			
Parametrul	**Simbol**	**Condiţii**	**HC**	**HCT**	**VHC**	**VHCT**
Curent de iesire la nivelul LOW (mA)	I_{OlmaxC}	Sarcina CMOS	0,02	0,02	0,05	0,05
	I_{OLmaxT}	Sarcina TTL	4,0	4,0	8,0	8,0

Tensiune de iesire la nivelul LOW (V)	V_{OLmaxC}	$I_{out} \leq I_{OlmaxC}$	0,1	0,1	0,1	0, l
	V_{OLmaxT}	$I_{out} \leq I_{OlmaxT}$	0,33	0,33	0,44	0,44
Curent de iesire la nivelul HIGH (mA)	I_{OHmaxC}	Sarcina CMOS	-0,02	-0,02	-0,05	-0,05
	I_{OHmaxT}	Sarcina TTL	-4,0	-4,0	-8,0	-8,0
Tensiune de iesire la nivelul HIGH (V)	V_{OHminC}	$I_{out} \leq I_{OlmaxC}$	4,4	4,4	4,4	4,4
	V_{OHminT}	$\|I_{out}\| \leq \|I_{OlmaxT}\|$	3,84	3,84	3,80	3,80

Tabelul 3-5 prezintă parametrii de ieşire al familiilor CMOS pentru sarcini atât CMOS, cât şi TTL. Semnificaţiile acestor parametri sunt următoarele:

I_{OLmaxC} Curentul maxim pe care îl poate asigura o ieşire în starea LOW, având conectată o sarcină CMOS. Deoarece valoarea dată este pozitivă, curentul circulă *către* borna de ieşire.

I_{OLmaxT} Curentul maxim pe care îl poate asigura o ieşire în starea LOW, având conectată o sarcină TTL.

V_{OLmaxC} Tensiunea maximă pe care o asigură garantat, în starea LOW, o ieşire având conectată o sarcină CMOS.

V_{OLmaxT} Tensiunea maximă pe care o asigură garantat, în starea LOW, o ieşire având conectată o sarcină TTL, adică fără a se depăsi I_{OLmaxT}.

I_{OHmaxC} Curentul maxim pe care îl poate asigura o ieşire în starea HIGH, având conectată o sarcina CMOS. Deoarece valoarea dată este negativă, curentul circulă *dinspre* borna de ieşire.

I_{OHmaxT} Curentul maxim pe care îl poate asigura o ieşire în starea HIGH, având conectată o sarcină TTL.

V_{OHminC} Tensiunea minimă pe care o asigură garantat, în starea HIGH, o ieşire având conectată o sarcina CMOS, adică fără a se depăşi I_{OHmaxC}.

V_{OHminT} Tensiunea minimă pe care o asigură garantat, în starea HIGH, o ieşire având conectată o sarcină TTL, adică fără a se depăşi I_{OHmaxT}.

Valorile tensiunilor de mai sus determină marginile de zgomot în c.c. În starea LOW, marginea de zgomot în c.c. este diferenţa dintre V_{OLmax} şi V_{ILmax}. Ea depinde atât de caracteristicile ieşirii de comandă, cât şi de cele ale intrărilor comandate de aceasta. De exemplu, în starea LOW, marginea de zgomot în c.c. pentru o ieşire de HCT care comandă câteva intrări de HCT (sarcină CMOS) este de 0,8 - 0,1 = 0,7 V. Analog, în starea HIGH, marginea de zgomot în c.c. este diferenţa dintre V_{OHmin} şi V_{IHmin}. În general, la interconectarea unor dispozitive din familii diferite trebuie comparate V_{OLmax} şi V_{OHmin} corespunzătoare porţilor de comandă cu V_{ILmax} şi V_{IHmin} corespunzătoare tuturor porţilor comandate, în vederea determinării marginilor de zgomot în cazul cel mai defavorabil.

3.5.4 Fanout-ul circuitelor CMOS

Parametrii I_{OLmax} şi I_{OHmax} din tabel determină posibilitatile de fanout şi sunt extrem de importanţi în situaţia în care o poartă comandă mai multe intrări din una sau mai multe familii diferite. Pentru a stabili dacă o ieşire se încadrează în limita de fanout specificată, trebuie calculaţi doi parametri:

Fanout în starea HIGH - Se adună valorile I_{IHmax} corespunzătoare tuturor intrărilor comandate. Suma obţinută trebuie să fie mai mică decât I_{OHmax} corespunzător ieşirii de comandă.

Fanout în starea LOW - Se adună valorile I_{ILmax} corespunzătoare tuturor intrărilor comandate. Suma obţinută trebuie să fie mai mică decât I_{OLmax} corespunzător ieşirii de comandă.

3.6 Circuite logice bipolare

Familiile logice bipolare folosesc, ca blocuri structurale fundamentale ale circuitelor logice, *diode semiconductoare* şi *tranzistoare bipolare cu joncţiuni*. Cele mai simple elemente logice bipolare realizează funcţii logice numai cu diode şi rezistoare; denumirea lor generică este „circuite logice cu diode”. Majoritatea porţilor cu circuite logice TTL au structura internă realizată pe baza circuitelor logice cu diode, iar caracteristicile de comandă de la ieşire se obţin prin amplificarea în circuite cu tranzistoare.

3.6.1 Diode

*O **diodă semiconductoare*** se fabrică din două tipuri de materiale semiconductoare, numite *n* şi *p*, care sunt puse în contact ca în fig. 3-33(a). În principiu, materialele sunt tot cele folosite la realizarea tranzistoarelor MOS cu canal *n* şi cu canal *p*. Punctul de contact dintre materialele de tip *n* şi de tip *p* se numeşte *joncţiune pn*; (în realitate, diodele se fabrică, în mod normal, dintr-un singur cristal monolitic de material semiconductor, fiecare jumătate a acestuia fiind dopată cu impurităţi în mod diferit, pentru obţinerea caracteristicilor de material de tip *p*, respectiv de tip *n.)*

Proprietăţile fizice ale joncţiunii *pn* determină circulaţia fără dificultate a curentului, în sensul pozitiv, dinspre materialul de tip p către cel de tip *n*. Prin urmare, dacă realizăm practic circuitul din fig. 3-33(b), joncţiunea *pn* se comportă aproape ca un scurtcircuit. Dar, datorită aceloraşi proprietăţi fizice, circulaţia unui curent pozitiv în sensul invers, de la *n* către *p*, întâmpină o puternică rezistenţă. Astfel, în circuitul din fig. 3-33(c), joncţiunea *pn* se comportă aproape ca o întrerupere. Fenomenele descrise constituie ***caracteristica de diodă***.

Deşi există posibilitatea construirii unor tuburi cu vid sau a altor dispozitive cu caracteristică de diodă, în sistemele moderne se utilizează joncţiunile *pn* - diodele semiconductoare- pe care de aici înainte le vom numi simplu *diode.* Figura 3-34(a) prezintă simbolul utilizat în scheme pentru diodă. După cum am mai spus, în funcţionare normală este permisă circulaţia unui curent semnificativ numai în sensul indicat de cele două săgeti, dinspre *anod* spre *catod.* Practic, dioda se comportă ca un scurtcircuit atâta timp cât căderea de tensiune pe joncţiunea anod-catod este pozitivă. Dacă tensiunea anod-catod este negativă, dioda se comportă ca o întrerupere a circuitului, nepermiţând trecerea curentului.

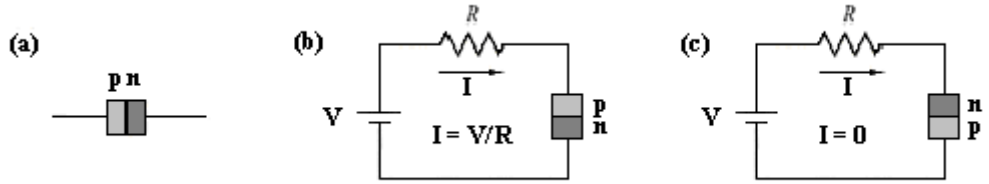


Figura 3-33 Diode semiconductoare: (a) joncţiunea pn; (b) joncţiune polarizată direct, permiţând circulaţia curentului; (c) joncţiune polarizată invers, blocând circulaţia curentului

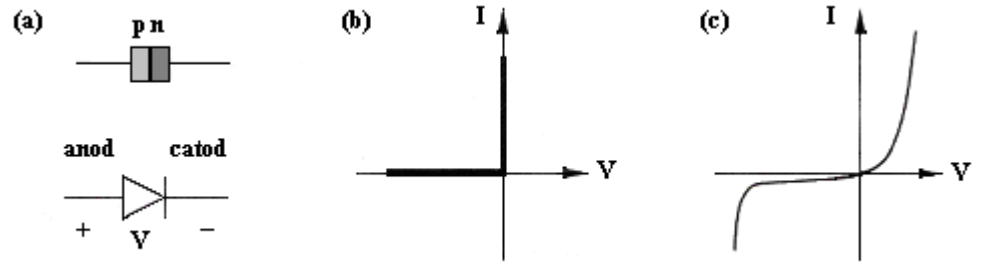


Figura 3-34 Diode: (a) simbolul; (b) caracteristica de transfer a unei diode ideale; (c) caracteristica de transfer a unei diode reale

Caracteristica de transfer a unei diode ideale, prezentată în fig. 3-34(b), ilustrează mai bine această comportare. Dacă tensiunea anod-catod, notată cu V, este negativă, se spune că dioda este *polarizată invers,* iar curentul I, prin diodă, este zero. Dacă *V* este pozitivă, se spune că dioda este *polarizată direct,* iar I poate avea o valoare pozitivă, arbitrar mare. De fapt, V nu poate fi niciodată mai mare ca zero, deoarece o diodă ideală se comportă, în polarizare directă, ca un scurtcircuit cu rezistenţă zero.

O diodă reală, neideală, prezintă o rezistenţă mai mică decât infinit în polarizare inversă şi o rezistenţă mai mare ca zero în polarizare directă, deci caracteristica ei de transfer arată ca aceea din fig. 3-34(c). În polarizare directă, dioda se comportă ca o rezistenţă neliniară de mică valoare; în polarizare inversă, prin diodă circulă un *curent rezidual* slab, negativ. Dacă tensiunea negativă aplicată depăşeşte în modul o anumită valoare, dioda *se străpunge,*

permiţând circulaţia unui curent negativ de intensitate mare; în majoritatea aplicaţiilor se evită acest mod de funcţionare.

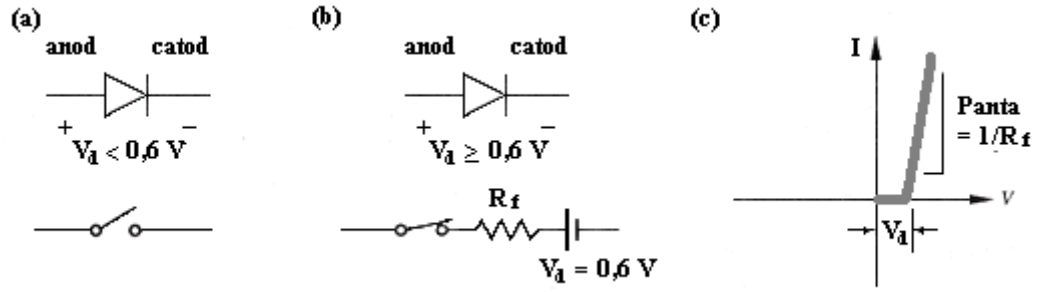


Figura 3-35 Modelul diodei reale: (a) în polarizare inversă (b) în polarizare directă; (c) caracteristica de transfer a diodei polarizate direct

O diodă reală poate fi reprezentată prin modelul simplu din fig. 3-35(a) şi (b). În polarizare inversă, dioda se comportă că o întrerupere a circuitului; curentul rezidual se ignoră. În polarizare directă, dioda se comportă ca o rezistenţă R_f de mică valoare, în serie cu o sursă de tensiune de mică valoare, V_d. R_f se numeşte *rezistenţă directă* a diodei, iar *Vd* este *căderea de tensiune pe diodă.*

Diode Zener

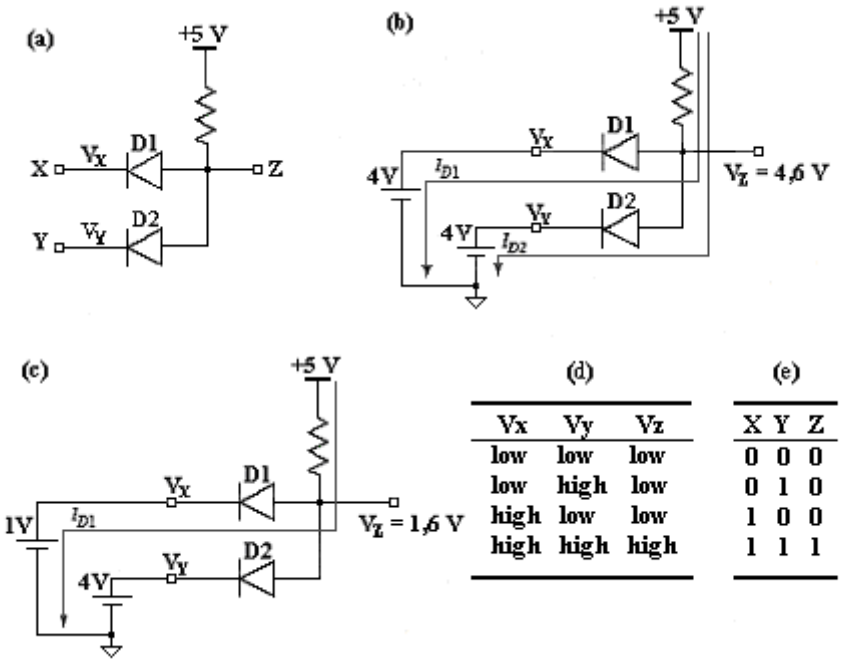
Diodele Zener valorifică fenomenul de străpungere a diodelor, mai precis panta foarte mare a caracteristicii V-I în regiunea de străpungere. O diodă Zener poate funcţiona ca stabilizator de tensiune, Dacă i se adaugă un rezistor pentru limitarea curentului de străpungere. Există o gamă largă de diode Zener, cu diferite tensiuni de străpungere, fabricate pentru a fi încorporate în stabilizatoare de tensiune.

3.6.2 Circuite logice cu diode

Tabel 3-6 Nivelurile logice într-un sistem logic simplu cu diode

Nivel de semnal	Semnificaţie	Valoare logică binară
0 … 2 V	LOW	0
2 … 3 V	margine de zgomot	nedeterminată
3 … 5 V	HIGH	1

Caracteristica de diodă poate fi utilizată pentru efectuarea unor operaţii logice. Se consideră un sistem logic alimentat cu 5 V şi având caracteristicile din tabelul 3-6. Domeniul de 5 V este împărţit în două subdomenii de tensiuni de semnal, LOW şi HIGH, şi o margine de zgomot de 1 V, între ele. O tensiune cuprinsă în domeniul LOW este interpretată ca 0 logic, iar o tensiune din domeniul HIGH este considerată 1 logic.



(d)

Vx	Vy	Vz
low	low	low
low	high	low
high	low	low
high	high	high

(e)

X	Y	Z
0	0	0
0	1	0
1	0	0
1	1	1

Figura 3-36 Poartă AND cu diode: (a) schema electrică; (b) ambele intrări HIGH; (c) o intrare HIGH şi cealaltă LOW; (d) tabelul funcţiei; (e) tabelul de adevăr

Ţinând cont de aceste definiţii, se poate construi *o poartă AND cu diode* ca în fig. 3-36(a). În circuitul prezentat, să considerăm că ambele intrări, X şi Y, sunt conectate la surse de tensiune HIGH, spre exemplu, de 4 V, deci V_X *şi* V_Y sunt, fiecare, de câte 4 V, ca în fig.(b). Prin urmare, ambele diode sunt polarizate direct, tensiunea de ieşire, V_Z, depăşind 4 V cu căderea de tensiune de pe o diodă, fiind deci de aproximativ 4,6 V. Un curent de valoare mică, determinat de rezistorul R, circulă dinspre sursa de 5 V, prin cele două diode, către sursele de 4 V. Săgeţile din figură arată traseele pe care circulă curentul.

Presupunem acum că V_X scade la 1V, ca în fig. 3-36(c). În poarta AND cu diode, tensiunea de ieşire devine egală cu cea mai mică dintre cele două tensiuni de intrare plus căderea de tensiune pe o diodă. În consecinţă, V_Z scade la 1,6 V, iar dioda *D2* devine polarizată invers (anodul se află la 1,6 V, în timp ce pe catod se menţin 4 V). Singura intrare LOW „forţează" o valoare LOW la ieşirea porţii AND cu diode. Evident, două intrări LOW vor genera la ieşire tot starea LOW. Funcţia realizată este prezentată sistematizat în fig. (d) şi reluată în (c), exprimată în valori logice binare.

3.6.3 Tranzistoare bipolare cu joncţiuni

Un ***tranzistor bipolar cu joncţiuni*** este un dispozitiv cu trei terminale care se comportă, în majoritatea circuitelor logice, ca un întrerupător comandat în curent. Dacă aplicăm un curent slab unuia dintre terminale, numit ***bază,*** întrerupătorul se închide, permiţând circulaţia curentului între celelalte două

terminale, numite ***emitor*** şi ***colector***. Dacă în bază nu se aplică nici un curent, întrerupătorul rămâne deschis şi între emitor şi colector nu circulă nici un curent.

Pentru a studia funcţionarea unui tranzistor vom cerceta mai întâi funcţionarea unei perechi de diode conectate ca în fig. 3-37(a). În acel circuit, curentul poate circula dinspre nodul B către nodurile C sau E, în funcţie de polarizarea directă a uneia sau a alteia dintre diode. În orice caz, între C şi E şi în sensul opus nu poate circula nici un curent deoarece, orice set de tensiuni am alege pentru nodurile B, C şi E, una sau ambele diode ar fi întotdeauna în polarizare inversă. Joncţiunile *pn* corespunzătoare celor două diode care formează circuitul sunt prezentate în fig. (b).

Să presupunem acum că ansamblul format din cele două diode conectate în modul descris se fabrică aşa încât diodele să aibă regiunea de tip *p* comună, ca în fig. 3-37(c). Structura astfel obţinută se numeşte *tranzistor npn şi* are o proprietate uimitoare. Dacă prin joncţiunea *pn* bază-emitor facem să circule un curent, atunci poate circula curent şi prin joncţiunea *np* colector-bază (ceea ce, în mod normal, nu pare posibil) şi de acolo către emitor.

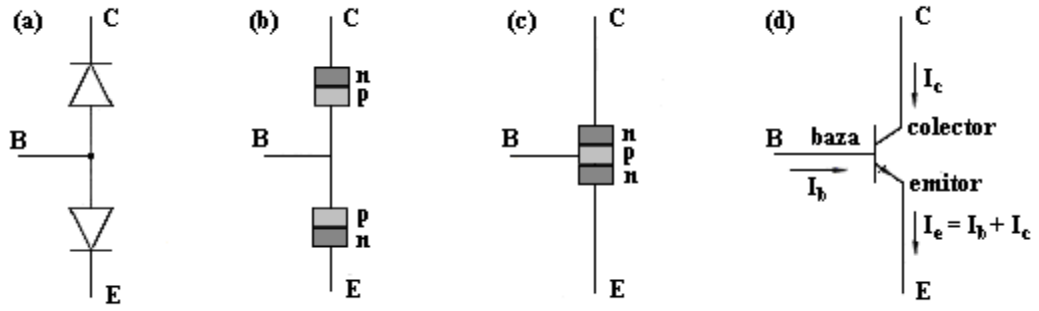


Figura 3-37 Alcătuirea unui tranzistor npn: (a) diode cu terminalele de aceeaşi polaritate conectate împreună; (b) Joncţiunile pn corespunzătoare; (c) structura unui tranzistor npn; (d) simbolul tranzistorului npn

Simbolul utilizat în scheme pentru tranzistorul *npn* este cel din fig. 3-37(d). Simbolul conţine o mică săgeată ce indică sensul pozitiv al curentului. Săgeata ne aminteşte, de asemenea, că joncţiunea bază-emitor este o joncţiune *pn*, ca a unei diode, al cărei simbol conţine o săgeată orientată în acelaşi sens.

Se pot realiza şi *tranzistoare pnp*. Tranzistoarele *pnp* sunt însă rar utilizate în circuitele digitale, deci nu le vom mai acorda atenţie în continuare.

3.6.4 Inversor logic realizat cu tranzistor

Figura 3-38 arată cum se poate realiza un inversor logic utilizând un tranzistor *npn în* configuraţia cu emitorul comun. Când tensiunea de intrare este LOW, tensiunea de ieşire este HIGH şi reciproc.

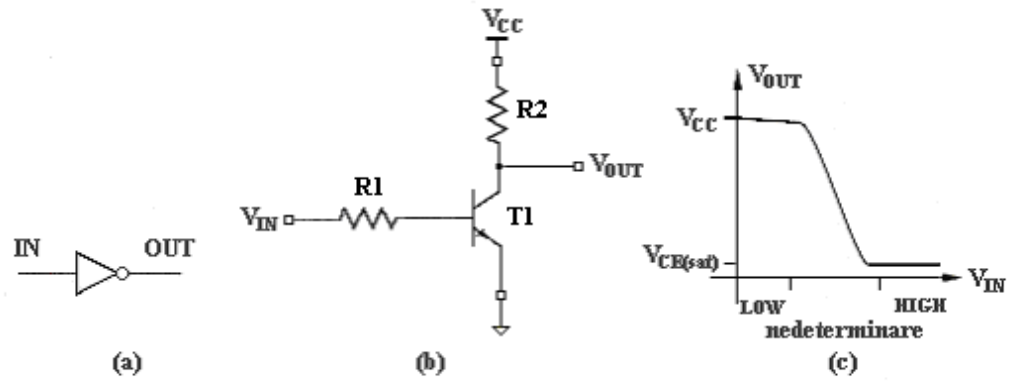


Figura 3-38 Inversor realizat cu tranzistor: (a) simbolul logic; (b) schema electrică; (c) caracteristica de transfer

În aplicaţiile de comutaţie digitală, tranzistoarelor bipolare li se impune adesea un regim de funcţionare în care ele sunt fie blocate, fie saturate. Prin urmare, circuitele digitale de genul inversorului din fig. 3-38 sunt astfel concepute încât tranzistoarele lor să se afle (aproape) întotdeauna într-una dintre stările descrise de fig. 3-39. Când tensiunea de intrare V_{IN} este LOW, ea are o valoare suficient de mică pentru că I_b să fie zero, iar tranzistorul să fie blocat; atunci, joncţiunea colector-emitor constituie aproape o întrerupere de circuit. Când V_{IN} este HIGH, valoarea ei este suficient de mare (şi R_I - rezistența internă, este suficient de mică) pentru ca tranzistorul să fie saturat. Dacă *R2* are o valoare rezonabilă; atunci, joncţiunea colector-emitor constituie aproape un scurtcircuit. Tensiunile de intrare cu valori cuprinse în domeniul dintre cele definite ca LOW şi HIGH nu sunt acceptate, cu excepţia intervalelor de timp în care au loc tranziţiile. Această regiune de nedeterminare corespunde marginii de zgomot.

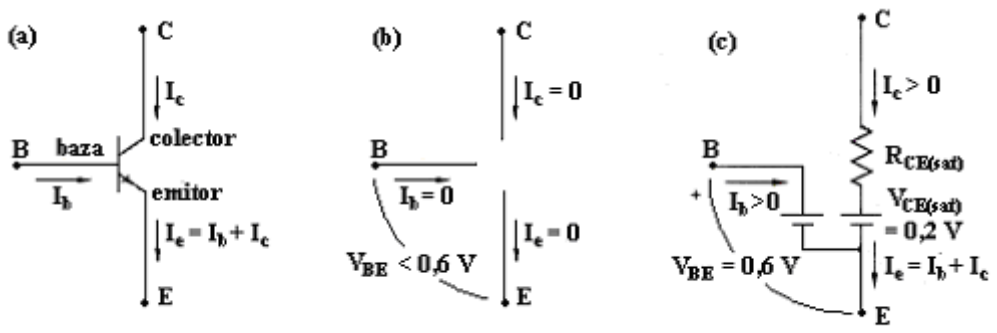


Figura 3-39 Stările normale ale unui tranzistor npn în circuitele de comutaţie digitale: (a) simbolul tranzistorului şi curenţii; (b) circuitul echivalent al tranzistorului blocat (OFF); (c) circuitul echivalent al tranzistorului saturat (ON)

3.6.5 Tranzistoare Schottky

Când semnalul de la intrarea unui tranzistor saturat se modifică, semnalul de ieşire nu se modifică imediat; este nevoie de un timp suplimentar, numit *timp de stocare,* pentru ca tranzistorul să iasă din saturaţie.

Timpul de stocare poate fi eliminat, iar timpul de propagare poate fi redus dacă se creează condiţii ca tranzistoarele să nu se satureze în funcţionare normală. Familiile de circuite logice TTL din generatiile actuale realizează acest lucru prin montarea unei ***diode Schottky*** între baza şi colectorul fiecărui tranzistor susceptibil de a intra în saturaţie, ca în fig. 3-40. Tranzistoarele obţinute, care nu se pot satura, sunt numite *tranzistoare cu limitare Schottky* sau, mai scurt, ***tranzistoare Schottky***.

În polarizare directă, căderea de tensiune pe o diodă Schottky este mult mai mică decât pe o diodă obişnuită, respectiv 0,25 V faţă de 0,6 V. La un tranzistor obişnuit saturat, tensiunea bază-colector este de 0,4 V, cum arată fig. 3-41(a). La un tranzistor Schottky, dioda Schottky constituie o cale prin care curentul începe să circule dinspre bază spre colector înainte ca tranzistorul să între în saturaţie, ca în fig. (b). În fig. 3-42 este prezentată schema electrică a unui inversor simplu realizat cu tranzistor Schottky.

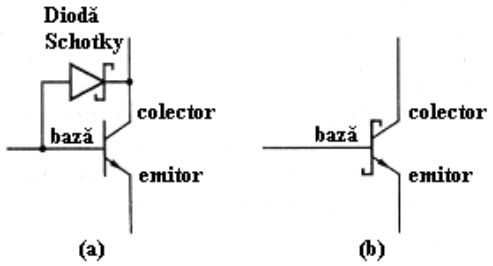


Figura 3-40 Tranzistor cu limitare Schotky: (a) schema electrică; (b) simbolul

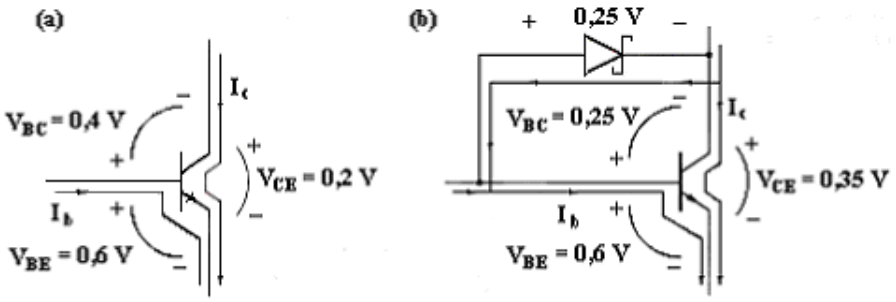


Figura 3-41 Funcţionarea unui tranzistor cu curent mare de bază: (a) tranzistor obişnuit saturat; (b) tranzistor cu diodă Schotky pentru prevenirea saturării

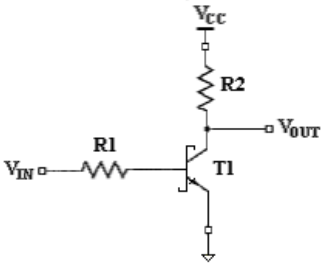


Figura 3-42 Inversor realizat cu tranzistor Schotky

3.7 Logica tranzistor-tranzistor

Familia de circuite logice cu tranzistoare bipolare cu cea mai largă răspândire este cea care utilizează logica tranzistor-tranzistor. De fapt, există mai multe familii TTL diferite, care acoperă o întreagă gamă de viteze, consumuri de putere şi alte caracteristici.

Familiile TTL lucrează, în principiu, cu aceleaşi niveluri logice ca şi familiile CMOS compatibile cu TTL, prezentate anterior. Pentru studierea comportării circuitelor TTL vom folosi următoarele definiţii ale nivelurilor LOW şi HIGH:

LOW 0 ... 0,8 V
HIGH 2,0 ... 5,0 V

3.7.1 Poarta TTL NAND de bază

În fig. 3-44 este prezentată schema electrică a unei porţi LS-TTL **NAND** cu două intrări. Funcţia **NAND** este obţinută prin combinarea unei porţi **AND** cu diode cu un amplificator tampon inversor. Veţi înţelege mai bine cum funcţionează circuitul dacă îl considerăm ca fiind format din cele trei părţi puse în evidenţă în fig. 3-43, şi anume:

- Poarta **AND** cu diode şi protecţia intrărilor.
- Separatorul de fază.
- Etajul de ieşire.

Diodele ***D1X*** şi ***D1Y*** şi rezistorul ***R1*** din fig. 3-44 formează o poarta **AND** cu diode. Diodele de limitare ***D2X*** şi ***D2Y*** nu au nici un rol în cazul unei funcţionări normale, însă limitează excursiile negative nedorite, apărute la intrări, la valoarea căderii de tensiune pe o singură diodă. Asemenea excursii

negative pot apărea în tranziţiile HIGH-LOW de la intrări, ca rezultat al fenomenelor caracteristice liniilor de transmisie.

Tranzistorul ***T2*** şi rezistoarele conectate la acesta formează un separator de fază care comandă etajul de ieşire. În funcţie de tensiunea, LOW sau HIGH, de la ieşirea porţii **AND** cu diode, ***T2*** este fie blocat, fie în conducţie.

(a)

X	Y	V_A	T2	T3	T4	T5	T6	V_Z	Z
L	L	<1,05	off	on	on	off	off	>2,7	H
L	H	<1,05	off	on	on	off	off	<0,35	H
H	L	<1,05	off	on	on	off	off	<0,35	H
H	H	1,2	on	off	off	on	on	<0,35	H

(b)

X	Y	Z
0	0	1
0	1	1
1	0	1
1	1	0

(c)

Figura 3-43 Funcţia realizată de o poartă TTL NAND cu două intrări: (a) tabelul funcţiei; (b) tabelul de adevăr; (c) simbolul logic

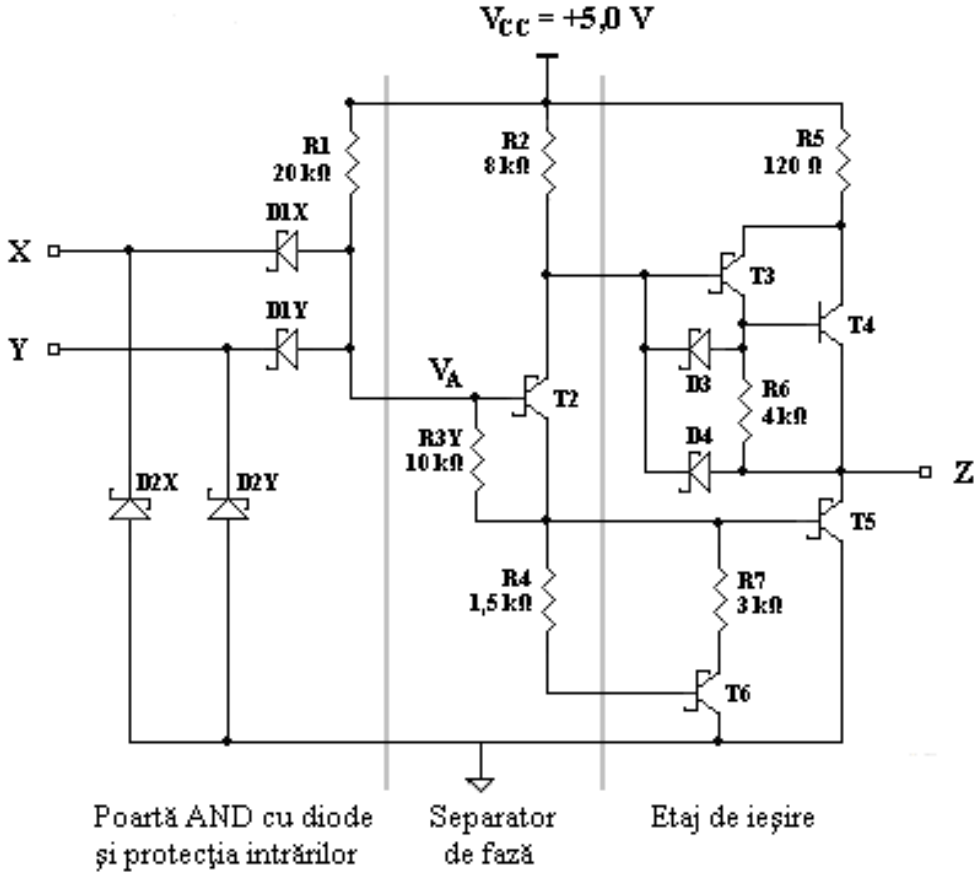


Figura 3-44 Schema electrică a porţii **NAND** cu două intrări LS-TTL

Etajul de ieşire conţine două tranzistoare, ***T4*** şi ***T5***, numai unul dintre acestea fiind deschis la un moment dat. Etajul de ieşire TTL este cunoscut şi sub denumirile ***ieşire în contratimp*** sau ***totem pole***. Asemenea tranzistoarelor cu canal *n şi* cu canal *p* din circuitele CMOS, ***T4*** şi ***T5*** efectuează Funcţia realizată de poarta TTL **NAND** este descrisă de tabelul din fig. 3-43(a). Poarta efectuează într-adevăr funcţia **NAND**, în fig. (b) şi (c) fiind prezentate tabelul de adevăr şi simbolul logic care îi corespund. Porţile TTL **NAND** comercializate au până la 13 intrări.

3.7.2 Nivelurile logice şi marginile de zgomot

La începutul capitolului am precizat că vom considera LOW semnalele cuprinse între 0 şi 0,8 V şi HIGH pe cele cuprinse între 2,0 şi 5,0 V. De fapt, se pot preciza cu mai multă rigurozitate nivelurile TTL de intrare şi de ieşire, la fel cum am procedat la circuitele CMOS:

V_{OHmin} Tensiunea de ieşire minimă în starea HIGH; la majoritatea familiilor TTL, 2,7V.

V_{IHmin} Tensiunea de intrare minimă recunoscută garantat ca HIGH; 2,0 V pentru toate familiile TTL.

V_{ILmax} Tensiunea de intrare maximă recunoscută garantat ca LOW; la majoritatea familiilor TTL, 0,8 V.

V_{OLmax} Tensiunea de ieşire maximă în starea LOW; la majoritatea familiilor TTL, 0,5V

Marginile de zgomot sunt reprezentate în fig. 3-45:

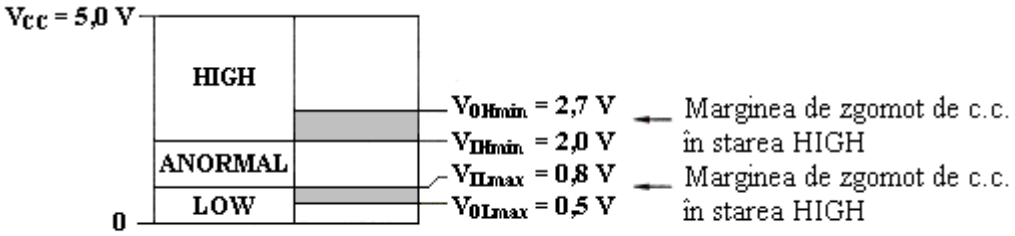


Figura 3-45 Marginile de zgomot caracteristice familiilor de circuite logice TTL de uz larg (74LS, 74S, 74ALS, 74AS, 74F)

La majoritatea familiilor TTL, pentru starea HIGH, valoarea de catalog V_{OHmin}, depăşeşte cu 0,7 V valoarea V_{IHmin}, deci circuitele TTL prezintă în starea HIGH *o margine de zgomot de c.c.* de 0,7 V. Înseamnă că numai un zgomot de minimum 0,7 V poate modifica, în cazul cel mai defavorabil, un semnal de ieşire HIGH astfel că acesta să nu poată fi recunoscut garantat ca semnal de intrare HIGH. În starea LOW, V_{ILmax}. depăşeşte valoarea V_{OLmax} cu numai 0,3 V, deci marginea de zgomot de c.c. în starea LOW este de numai 0,3

V. În general, circuitele TTL şi cele compatibile cu ele au tendinţa de a fi mai sensibile la zgomot în stare LOW, decât în HIGH.

3.7.3 Fanout-ul circuitelor TTL

Conform definiţiei prezentate deja în secţiunea 3.5.4, *fanout* este un parametru ce arată câte intrări de porţi pot fi conectate la (şi comandate de) o singură ieşire a unei porţi. Fanout în c.c. aferent ieşirilor CMOS ce comandă intrări CMOS este practic nelimitat deoarece intrările CMOS aproape că nu necesită curent în nici una dintre stări, HIGH sau LOW. Cu intrările TTL, lucrurile stau altfel. Din această cauză există limite bine definite impuse parametrului fanout aferent ieşirilor TTL şi CMOS care comandă intrări TTL

Ca şi la CMOS, sensul curentului prin bornele de intrare sau de ieşire ale unui circuit TTL se consideră pozitiv când curentul *intră în* borne din exterior şi negativ când curentul *iese din* borne către exteriorul circuitului. Ca urmare, pentru o ieşire la care sunt conectate una sau mai multe intrări, suma algebrică a tuturor curenţilor de intrare şi de ieşire este 0.

Curentul necesar unei intrări TTL depinde de starea acesteia, HIGH sau LOW, şi se stabileşte prin intermediul a doi parametri:

I_{ILmax} Curentul maxim necesar unei intrări pentru a trece în starea LOW. Deoarece curentul iese dintr-o bornă de intrare a circuitului TTL în starea LOW, I_{ILmax} este negativ. Pentru majoritatea intrărilor LS-TTL, I_{ILmax} = -0,4 mA, valoare denumită uneori *sarcină unitară în starea LOW* pentru LS-TTL.

I_{IHmax} Curentul maxim necesar unei intrări pentru a trece în starea HIGH. Deoarece curentul *intră* într-o bornă de intrare a circuitului TTL în starea HIGH, I_{IHmax} este pozitiv. Pentru majoritatea intrărilor LS-TTL, I_{IHmax} = 20μA, valoare denumită uneori *sarcină unitară în starea HIGH* pentru LS-TTL.

Analog ieşirilor CMOS, ieşirile TTL pot furniza sau absorbi curent de o anumită intensitate, în funcţie de starea lor, HIGH sau LOW:

I_{OLmax} Curentul maxim ce poate fi absorbit de o ieşire în starea LOW, pentru care tensiunea de ieşire se mai menţine la o valoare ce nu depăşeşte V_{OLmax}. Deoarece curentul intră în borna de ieşire, I_{OLmax} este pozitiv, pentru majoritatea ieşirilor LS-TTL find de 8 mA.

I_{OHmax} Curentul maxim ce poate fi furnizat de o ieşire în starea HIGH, pentru care tensiunea de ieşire se mai menţine la o valoare de minimum $V_{OHmin.}$ Deoarece curentul iese din borna de ieşire, I_{OHmax} este negativ, pentru majoritatea ieşirilor LS-TTL fiind de -400 μA.

Valoarea I_{OLmax} pentru ieşirile LS-TTL tipice este de exact 20 de ori mai mare decât valoarea absolută a I_{ILmax}. Din acest motiv se spune că LS-TTL au *fanout în starea LOW* de 20, deoarece o ieşire poate comanda 20 de intrări în starea LOW. Similar, valoarea absolută a I_{OHmax} este de exact 20 de ori mai mare decât $I_{IHmax,}$ deci LS-TTL au *fanout în starea HIGH* tot de 20. Valoarea de *fanout global* este cea mai mică dintre valorile de fanout corespunzătoare stărilor HIGH şi LOW.

Conectarea la o ieşire TTL a unei sarcini ce depăşeşte valoarea de fanout prescrisă are aceleaşi efecte distructive, ca la dispozitivele CMOS. Este vorba despre reducerea sau dispariţia marginilor de zgomot de c.c., o eventuală creştere a timpilor de tranziţie şi posibilitatea ca dispozitivele să se supraîncălzească.

În general, pentru a verifica dacă unei ieşiri nu i-a fost conectată o suprasarcină trebuie efectuate două calcule:

în starea HIGH: Se adună valorile I_{IHmax} corespunzătoare tuturor intrărilor comandate. Suma obţinută trebuie să fie mai mică sau egală cu valoarea absolută a I_{OHmax} corespunzătoare ieşirii de comandă.

în starea LOW: Se adună valorile I_{ILmax} corespunzătoare tuturor intrărilor comandate. Valoarea absolută a sumei obţinute trebuie să fie mai mică sau egală cu I_{OLmax} corespunzător ieşirii de comandă.

3.7.4 Poarta TTL NOR

Deşi poarta **NAND** constituie „calul de bătaie" al familiei TTL, se mai pot construi şi alte tipuri de porţi cu aceeaşi structură generală de circuit.

În fig. 3-46 este prezentată schema electrică a unei porţi LS-TTL **NOR.** Dacă oricare dintre intrările X sau Y este conectată la HIGH, intră în conducţie, corespunzător intrării respective, unul dintre tranzistoarele ***T2X*** sau ***T2Y,*** care au rolul de separator de fază. Ca urmare, ***T3*** şi **T4** se blochează, iar ***T5*** şi ***T6*** intră în conducţie, rezultând la ieşire un semnal LOW. Dacă ambele intrări sunt LOW, cele două tranzistoare separatoare de fază sunt blocate, iar la ieşire apare un semnal HIGH. Modul de funcţionare este descris şi de fig. 3-47.

Poarta LS-TTL **NOR** are circuitele de intrare, separatorul de fază şi etajul de ieşire aproape identice cu cele ale porţii **NAND**. Deosebirea constă în faptul că poarta LS-TTL **NAND** utilizează diode pentru realizarea funcţiei **AND**, pe când poarta LS-TTL **NOR** utilizează tranzistoare conectate în paralel în separatorul de fază pentru realizarea funcţiei **OR**.

Viteza, caracteristicile de intrare şi cele de ieşire ale porţii TTL **NOR** sunt comparabile cu cele ale porţii TTL **NAND.** Cu toate acestea, o poartă **NOR** cu n intrări conţine mai multe tranzistoare şi rezistoare, ocupând o suprafaţă mai mare de siliciu decât o poartă **NAND** cu *n* intrări. De asemenea, curentul rezidual intern limitează numărul de tranzistoare ***T2*** ce se pot conecta în paralel,

deci porţile **NOR** au un fan-in mai scăzut. (Cel mai mare număr de intrări al unei porţi **NOR** individuale este 5, faţă de cele 13 intrări ale unei porţi **NAND**). În consecinţă, porţile **NOR** sunt mai puţin întrebuinţate în proiectarea cu TTL decât porţile **NAND.**

Porţile TTL cele mai „fireşti" sunt porţile inversoare, cum sunt **NAND** şi **NOR**. Porţile TTL neinversoare conţin un etaj inversor suplimentar, de obicei între etajul de intrare şi separatorul de fază. Din această cauză, ele sunt, în mod normal, mai mari şi mai lente decât porţile inversoare corespunzătoare.

Asemenea porţilor CMOS, porţile TTL pot fi prevăzute cu ieşiri cu trei stări. Astfel de porţi au o intrare de „activare a ieşirii" *(output enable)* sau „dezactivare a ieşirii" *(output disable)*, care comandă trecerea ieşirii în starea de impedanţă mare, în care nici unul dintre tranzistoarele de ieşire nu se află în conducţie.

Unele porţi TTL se fabrică şi în varianta cu *ieşiri cu colectorul în gol*. În astfel de circuite, întreaga parte de sus a figurii 3-44 este omisă, astfel că este posibilă numai o aducere pasivă în HIGH, cu ajutorul unui rezistor exterior. Aplicaţiile porţilor TTL cu colectorul în gol şi calculele necesare sunt similare celor prezentate pentru porţile CMOS cu drenă în gol.

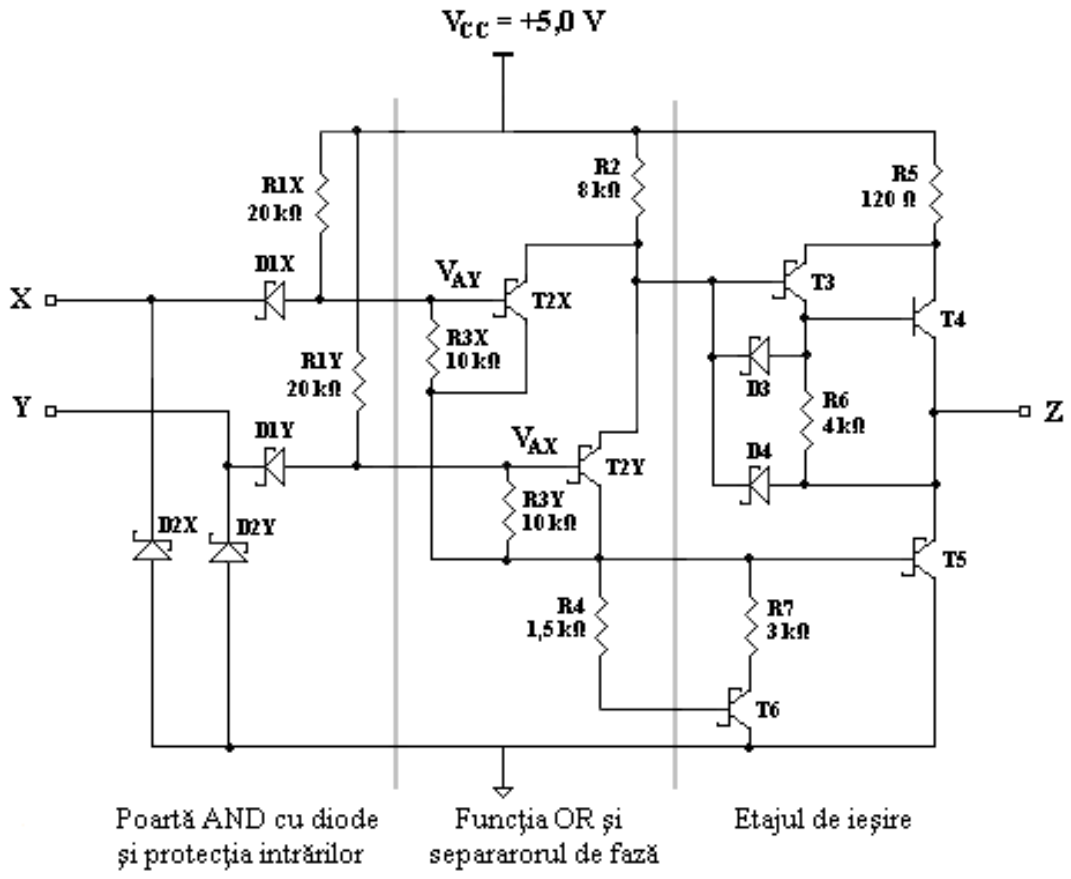


Figura 3-46 Schema electrică a unei porţi LS-TTL NOR cu două intrări

(a)

X	Y	V_{AX}	T2X	V_{AY}	T2Y	T3	T4	T5	T6	V_Z	Z
L	L	<1,05	off	<1,05	off	on	on	off	off	>2,7	H
L	H	<1,05	off	1,2	on	off	off	on	on	<0,35	L
H	L	1,2	on	<1.05	off	off	off	on	on	<0,35	L
H	H	1,2	on	1,2	on	off	off	on	on	<0,35	L

(b)

X	Y	Z
0	0	1
0	1	0
1	0	0
1	1	0

(c)

Figura 3-47 Poartă LS-TTL NOR cu două intrări: (a) tabelul funcţiei; (b) tabelul de adevăr; (c) simbolul logic

3.8 Familii TTL

Familiile TTL au evoluat de-a lungul anilor ca urmare a dorinţei proiectanţilor de circuite digitale de a obţine performanţe superioare. Consecinţa a fost dispariţia a trei familii TTL, proiectanţii din zilele noastre putând alege din cele cinci familii rămase la dispoziţia lor. Toate familille TTL sunt compatibile, în sensul că utilizează aceeaşi tensiune de alimentare şi aceleaşi niveluri logice, însă fiecare familie prezintă anumite avantaje în ceea ce priveste viteza, consumul de putere şi preţul.

3.8.1 Primele familii TTL

Familia iniţială de porţi logice TTL a fost lansată de Sylvania în 1963. Ea a devenit cunoscută datorită companiei Texas Instruments, ale cărei coduri din seria 7400, adaptate pentru porţi şi alte componente TTL au ajuns rapid să constituie un standard industrial.

Ca şi în cazul seriei CMOS 7400, dispozitivele aparţinând unei familii TTL sunt desemnate prin coduri de forma 74FAMxx, unde „FAM" este denumirea familiei, alcătuita din litere, iar xx este indicativul numeric al funcţiei. Dispozitivele aparţinând unor familii diferite, dar ale căror coduri conţin aceeaşi valoare xx, realizează aceeaşi funcţie. Din denumirea familiei TTL iniţiale lipseste grupul de litere „FAM", familia fiind desemnată ca *seria TTL 74.*

Valorile rezistoarelor din circuitul TTL original au fost modificate, obţinându-se două noi familii, cu caracteristici diferite. Familia *74H (High-speed TTL* - TTL de viteză mare*)* conţinea rezistoare de valori mai mici, realizând un timp de propagare redus în detrimentul consumului de putere.

Familia *74L (Low power TTL* - TTL de mică putere*)* conţinea rezistoare de valori mai mari, realizând un consum de putere redus în detrimentul timpului de propagare.

Având la dispoziţie cele trei familii TTL prezentate mai sus, proiectanţii de circuite digitale ai anilor '70 puteau opta între circuite de mare viteză şi circuite cu consum mic de putere. Apariţia tranzistoarelor Schottky le-a oferit aceste posibilităţi şi a detronat seriile TTL 74, 74H şi 74L. În continuarea secţiunii de faţă vom prezenta caracteristicile familiilor TTL actuale, cu performanţe superioare.

3.8.2 Familii TTL Schottky

Cronologic, prima familie în care au fost folosite tranzistoare Schottky a fost *74S (TTL Schottky)*. Folosind tranzistoare Schottky şi rezistoare de valori mici, această familie beneficiază de viteză mult mai mare, dar şi de un consum de putere mai mare decât seria TTL 74 originală.

Probabil că familia TTL cea mai larg folosită şi cea mai puţin costisitoare este *74LS (Low power Schottky TTL* - TTL Schottky de mică putere), lansată la scurt timp după 74S. Cu o combinaţie de tranzistoare Schottky şi rezistoare de valori mai mari, TTL 74LS atinge aceeaşi viteză că seria TTL 74, dar consumă aproximativ o cincime din puterea consumată de aceasta. Astfel, 74LS a devenit una dintre familiile de circuite logice preferate pentru noile aplicaţii cu TTL.

Inovaţiile în tehnologia circuitelor integrate şi în domeniul configuraţiilor, care au urmat, au dus la apariţia a încă două noi familii de circuite logice Schottky. Familia *74AS (Advanced Schottky TTL* - TTL Schottky avansată) are viteză aproape dublă faţă de 74S, la aproximativ acelaşi consum de putere. Familia *74ALS (Advanced Low-power Schottky TT)* - TTL Schottky avansată), de mică putere prezintă atât un consum mic de putere, cât şi viteze mai mari decât 74LS, cu care rivalizează în popularitate în privinţa satisfacerii parametrilor majori impuşi noilor aplicaţii cu TTL. Familia *74F (Fast TTL* - TTL rapid) se situează între 74AS şi 74ALS în ceea ce priveşte compromisul viteză/putere şi constituie, probabil, opţiunea cea mai frecventă pentru atingerea obiectivelor de rapiditate în aplicaţii cu TTL de ultimă oră.

Tabel 3-7 Caracteristicile porţilor din familiile TTL

		Familia				
Parametrul	**Simbol**	**74S**	**74LS**	**74AS**	**74ALS**	**74F**
Timp de propagare maxim (ns)		3	9	1,7	4	3
Puterea consumata de o poartă (mW)		19	2	8	1,2	4
Produs viteză-putere (pJ)		57	18	13,6	4,8	12
Tensiune de intrare la nivelul LOW (V)	$\mathbf{V_{ILmax}}$	0,8	0,8	0,8	0,8	0,8
Tensiune de ieşire la nivelul LOW(V)	$\mathbf{V_{0Lmax}}$	0,5	0,5	0,5	0,5	0,5
Tensiune de ieşire la nivelul HIGH (V)	$\mathbf{V_{OHmax}}$	2,7	2,7	2,7	2,7	2,7

Curent de intrare la nivelul LOW (mA)	$\mathbf{I_{ILmax}}$	-2,0	-0,4	-0,5	-0,2	-0,6
Curent de ieşire la nivelul LOW (mA)	$\mathbf{I_{OHmax}}$	20	8	20	8	20
Curent de intrare la nivelul HIGH (μA)	$\mathbf{I_{ILmax}}$	50	20	20	20	20
Curent de ieşire la nivel HIGH(μA)	$\mathbf{I_{OHmax}}$	-1000	-400	-2000	-400	-1000

3.8.3 Caracteristicile familiilor TTL

Primele două linii ale tabelului 3-7 specifică timpii de propagare (în nanosecunde) şi consumurile de putere (în mW) caracteristice unei porţi **NAND** tipice cu două intrări, din fiecare familie.

Un parametru ce exprimă eficienţa unei familii de circuite logice este *produsul viteză putere,* din linia a treia a tabelului. Aşa cum am mai arătat, acesta este, pur şi simplu, produsul dintre timpul de propagare şi consumul de putere caracteristice unei porţi tipice. Produsul viteză-putere constituie o estimare a randamentului, arătând câtă energie consumă o poartă logică pentru a comuta semnalul de ieşire.

Ultimele linii ale tabelului 3-7 specifică parametrii de intrare şi de ieşire ai unor porţi TTL tipice din fiecare familie. Folosind aceste informaţii se poate analiza funcţionarea porţilor T'TL privite din exterior, fără a cunoaşte detalii interne ale circuitelor.

3.8.4 Foaie de catalog pentru un dispozitiv TTL

Tabelul 3-8 prezintă un fragment dintr-o foaie de catalog pentru dispozitivul 74LS00. Dispozitivul 54LS00, ale cărui date apar, de asemenea, în tabel este identic cu cel dintâi, însă poate funcţiona în întreaga gamă de temperatură şi de tensiune „de uz militar" şi costă mai mult. Majoritatea componentelor TTL se produc şi în versiunea serie 54 de uz militar). În tabel apar trei secţiuni ale foii de catalog:

- *Condiţiile de funcţionare recomandate* precizează tensiunea de alimentare, domeniile tensiunilor de intrare, sarcina admisă în c.c. şi temperatura la care dispozitivul funcţionează normal.
- *Caracteristicile electrice* menţionează valorile tensiunilor şi curenţilor care mai apar la intrările şi la ieşirea dispozitivului în condiţiile de funcţionare recomandate:

I_I Curentul maxim de intrare la tensiune de intrare HIGH foarte mare.

I_{OS} Curentul de ieşire cu ieşirea în HIGH scurtcircuitată la masă.

I_{CCH} Curentul absorbit de la sursa de alimentare când toate ieşirile (ale tuturor celor patru porţi **NAND**) sunt HIGH. (Valoarea specificată se referă la întreaga capsulă, care conţine patru porţi **NAND**, deci curentul corespunzător unei singură porţi este un sfert din valoarca din tabel.)

I_{CCL} Curentul absorbit de la sursa de alimentare când toate ieşirile (ale tuturor celor patru porţi **NAND**) sunt LOW.

t_{LH} Timp de comutaţie directă (timpul de comutaţie din LOW în HIGH)

t_{HL} Timp de comutaţie inversă (timpul de comutaţie din HIGH în LOW)

Tabel 3-8 Foaie de catalog obişnuită, furnizată de producător, pentru 74LS00

CONDIŢII DE FUNCŢIONARE RECOMANDATE									
			SN54LS00			**SN74LS00**			
Param	**Descriere**		**Min.**	**Nom**	**Max.**	**Min.**	**Nom**	**Max.**	**Unit.**
V_{CC}	Tensiune de alimentare		4,5	5,0	5,5	4,75	5,0	5,25	V
V_{IH}	Tensiune de intrare HIGH		2,0			2,0			V
V_{IL}	Tensiune de intrare LOW				0,7			0,8	V
I_{OH}	Curent de ieşire HIGH				-0,4			-0,4	mA
I_{OL}	Curent de ieşire LOW				4			8	mA
T_A	Temp. aerului la funcţ.		-55		125	0		70	°C
CARACTERISTICI ELECTRICE LA TEMPERATURILE RECOMANDATE PENTRU AER									
			SN54LS00			**SN74LS00**			
Param	**Condiţii de testare**		**Min.**	**Nom**	**Max.**	**Min.**	**Nom**	**Max.**	**Unit.**
V_{IK}	V_{CC} = Min. ; I_N= -18 mA				-1,5			-1,5	V
V_{OH}	V_{CC} = Min.; V_{IH} = 2,0 V ; I_{OH} = -0,4 mA		2,5	3,4		2,5	3,4		V
V_{OL}	V_{CC} = Min. ; V_{IH} = 2,0 V; I_{OL} = 4 mA			0,25	0,4		0,25	0,4	V
	V_{CC} = Min. ; V_{IH} = 2,0 V; I_{OL} = 8 mA						0,35	0,5	V
I_I	V_{CC} = Max. ; V_I = 7,0 V;				0,1			0,1	mA
I_{IH}	V_{CC} = Max. ; V_I = 2,7 V;				20			20	µA
I_{IL}	V_{CC} = Max. ; V_I = 0,4 V;				-0,4			-0,4	mA
I_{OS}	V_{CC} = Max.		-20		-100	-20		-100	mA
I_{CCH}	V_{CC} = Max. ; V_I = 0 V;			0,8	1,6		0,8	1,6	mA
I_{CCL}	V_{CC} = Max. ; V_I = 4,5 V;			2,4	4,4		2,4	4,4	mA
CARACTERISTICI DE COMUTAŢIE, V_{CC} = 5,0 V, T_A = 25°C									
Param.	**De la (intrarea)**	**La (ieşirea)**	**Cond. de testare**			**Min.**	**Tip.**	**Max.**	**Unit.**
t_{PLH}	A sau B	Y	R_s = 2 kΩ; C_s = 15 pF				9	15	ns
t_{PHL}							10	15	ns

În catalogul producătorului, uneori, mai apare o a patra secţiune:

- Valorile maxime absolute precizează condiţiile cele mai defavorabile în care dispozitivul poate fi exploatat sau stocat fără a se deteriora.

Un catalog complet prezintă, de asemenea, circuitele de testare folosite la măsurarea parametrilor la producător şi diagrame care ilustrează modul în care parametrii tipici variază în anumite condiţii de funcţionare cum sunt variaţiile tensiunii de alimentare (V_{CC}), ale temperaturii mediului ambiant (T_A) şi ale sarcinii (R_S, C_S).

3.9 Realizarea interfeţelor CMOS/TTL

Proiectanţii de circuite digitale aleg o familie de circuite logice „prestabilită”, pe baza căreia construiesc un întreg sistem, alegerea fiind dictată de considerente generale ca viteză, putere, precum şi altele. Totuşi, în unele cazuri se impune utilizarea unor dispozitive aparţinând altor familii, fie din cauza faptului că sunt singurele disponibile, fie datorită unor cerinţe speciale. (De exemplu, nu toate componentele din familia *74LS* se produc şi în *74 HCT, şi* reciproc.) Prin urmare, este foarte important ca proiectantul să cunoască bine implicaţiile conectării unor ieşiri TTL la intrări CMOS şi invers.

La realizarea unei interfeţe *TTL/CMOS* trebuie luaţi în consideraţie mai mulţi factori, primul dintre acestia fiind marginea de zgomot. În starea LOW, marginea de zgomot de c.c. depinde de V_{OLmax} a ieşirii de comandă şi de V_{ILmax} a intrării comandate, fiind egală cu $V_{ILmax} - V_{OLmax}$. Similar, în starea HIGH, marginea de zgomot dc c.c. este egală cu $V_{OHmin} - V_{IHmin}$. Figura 3-48 prezintă valorile semnificative pentru familiile *TTL* şi *CMOS.*

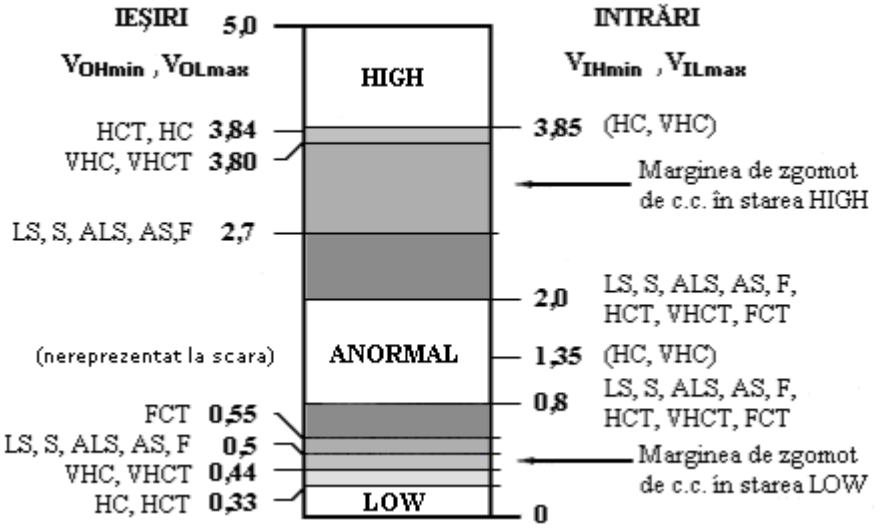


Figura 3-48 Nivelurile de ieşire şi de intrare de care să se ţină seama la realizarea interfeţelor dintre familiile TTL şi CMOS. (Remarcaţi că intrările familiilor HC şi HCT nu sunt compatibile cu TTL)

Fanout-ul este următorul factor ce trebuie avut în vedere. La fel ca în cazul sistemelor ce includ numai dispozitive TTL, proiectantul trebuie să calculeze suma curenţilor de intrare necesari dispozitivelor comandate de o anumită ieşire şi să compare această sumă cu posibilităţile de comandă ale ieşirii în ambele stări. Când un dispozitiv TTL comandă dispozitive CMOS, fanout-ul nu constituie o problemă, deoarece intrările CMOS aproape că nu necesită curent în nici una dintre stări. Însă intrările TTL, în special în starea LOW, necesită un curent substanţial, mai ales în comparaţie cu posibilităţile ieşirilor HC şi HCT.

Ultimul factor de care trebuie să se ţină seama este sarcina capacitivă. Capacitatea de sarcină a unui circuit logic măreste atât timpul de propagare, cât şi puterea disipată. Timpii de propagare măriţi se observă în special la ieşirile HC şi HCT, ai căror timpi de tranziţie cresc aproximativ cu câte 1 ns pentru fiecare 5 pF din sarcină. Tranzistoarele de ieşire ale circuitelor FCT prezintă rezistenţe de conducţie foarte scăzute, deci timpii lor de tranziţie cresc, cu aproximaţie, cu numai 0,1 ns pentru fiecare 5 pF din sarcină.

Pentru aceeaşi capacitate de sarcină, aceeaşi tensiune de alimentare şi aceeaşi configuraţie a circuitului în care sunt incluse, dispozitivele CMOS din toate familiile prezintă aceeaşi disipare de putere dinamică. Dispozitivele TTL disipă însă ceva mai puţină putere dinamică deoarece excursia de tensiune între nivelurile TTL HIGH şi LOW este mai mică.

3.10 Circuitele logice CMOS de tensiune scăzută

Industria producătoare de circuite integrate s-a orientat către dispozitive CMOS alimentate cu tesiuni mai scăzute din următoarele două motive:

- În majoritatea aplicaţiilor, excursia de tensiune de la o ieşire CMOS este egală cu diferenţa dintre barele de alimentare. Scăderea tensiunii de alimentare reduce mai mult decât proporţional puterea dinamică disipată.
- Pe măsură ce miniaturizarea tranzistoarelor este mai pronunţată, stratul izolator de oxid dintre poarta unui tranzistor CMOS, pe de o parte, şi drena şi sursa acestuia, de cealaltă parte, devine tot mai subţire, deci incapabil să constituie o bună izolaţie între diferenţele „ridicate” de potenţial, de 5 V.

Ca urmare, grupul de standarde industriale în domeniul circuitelor integrate JEDEC a stabilit tensiunile de ***3,3 V ± 0,3 V***; ***2,5 V ± 0,2 V*** şi ***1,8 V ± 0,15 V*** ca noi tensiuni standardizate pentru alimentarea circuitelor logice. Standardele JEDEC precizează şi nivelurile de tensiune ale semnalelor logice de intrare şi de ieşire corespunzătoare dispozitivelor alimentate cu aceste tensiuni.

Trecerea la tensiuni de alimentare mai scăzute s-a produs în etape şi se va desfăşura la fel în continuare. Pentru familiile de circuite logice discrete se urmăreşte producerea unor dispozitive caracterizate prin tensiuni de alimentare şi de ieşire scăzute, dar care pot prelua şi semnale de intrare de nivel mai mare. În acest mod, familiile CMOS alimentate cu 3,3 V pot interacţiona cu familii CMOS şi TTL alimentate cu 5 V.

Asemenea dispozitive au dimensiuni suficient de mari ca să justifice utilizarea a două tensiuni de alimentare. O tensiune scăzută, ca aceea de 2,5 V,

alimentează porţile din interiorul cipurilor, care alcătuiesc *circuitul logic central.* Pentru alimentarea circuitelor din exterior, de intrare şi de ieşire, care alcătuiesc *inelul exterior,* se utilizează o tensiune mai ridicată, de pildă 3,3 V, pentru asigurarea compatibilităţii cu dispozitivele din generaţiile anterioare ce fac parte din acelaşi sistem. În interior există şi circuite tampon speciale, care efectuează sigur şi rapid conversia între nivelurile logice de tensiune din circuitul logic central şi cele din inelul exterior.

3.10.1 Circuite logice LVTTL şi LVCMOS, alimentate cu 3,3 V

Corespondenţa dintre nivelurile de semnal caracteristice familiilor TTL standard şi cele ale dispozitivelor CMOS de tensiune scăzută, care funcţionează la valoarea nominală a tensiuni de alimentare este ilustrată sugestiv în fig. 3-49. În fig. (a) apar nivelurile de semnal originale, simetrice, ale familiilor CMOS alimentate exclusiv cu 5 V, ca HC şi VHC. La familiile CMOS compatibile cu TTL, cum sunt HCT, VHCT şi FCT, nivelurile de tensiune sunt deplasate în jos pentru asigurarea compatibilităţii, cum puteţi observa în fig. (b).

Într-o primă etapă de evoluţie a familiilor CMOS de tensiune scăzută s-a ales tensiunea de alimentare de 3,3 V. Standardul JEDEC pentru circuitele logice de 3,3 V defineşte, de fapt, două seturi de niveluri. Nivelurile *LVCMOS (low-voltage CMOS* – CMOS de tensiune scăzută*)* se utilizează în aplicaţiile realizate exclusiv cu componente CMOS, în care ieşirile comandă sarcini de c.c. reduse (mai mici de 100 μA), deci V_{OL} şi V_{OH} se menţin la o diferenţă de maximum 0,2 V faţă de barele de alimentare. Nivelurile *LVTTL (low-voltage TTL* – TTL de tensiune scăzută*),* din fig. (c), se utilizează în aplicaţii în care sarcinile de c.c. ale ieşirilor sunt semnificative, V_{OL} putând ajunge la 0,4 V, iar V_{OH}, la 2,4 V.

Alegerea nivelurilor logice TTL de la capătul inferior al domeniului de 5V s-a făcut relativ necorelat. După cum se vede în fig. (b) şi (c), nivelurile LVTTL au putut fi alese astfel încât să corespundă exact nivelurilor TTL. Astfel, o ieşire LVTTL poate comanda fără probleme o intrare TTL atâta timp cât curentul de ieşire se încadrează în domeniul specificat (I_{OLmax}, I_{OHmax}). Similar, o ieşire TTL poate comanda o intrare LVTTL, fără a depăşi însă tensiunea $V_{CC,}$ de 3,3 V, caracteristică dispozitivelor LVTTL.

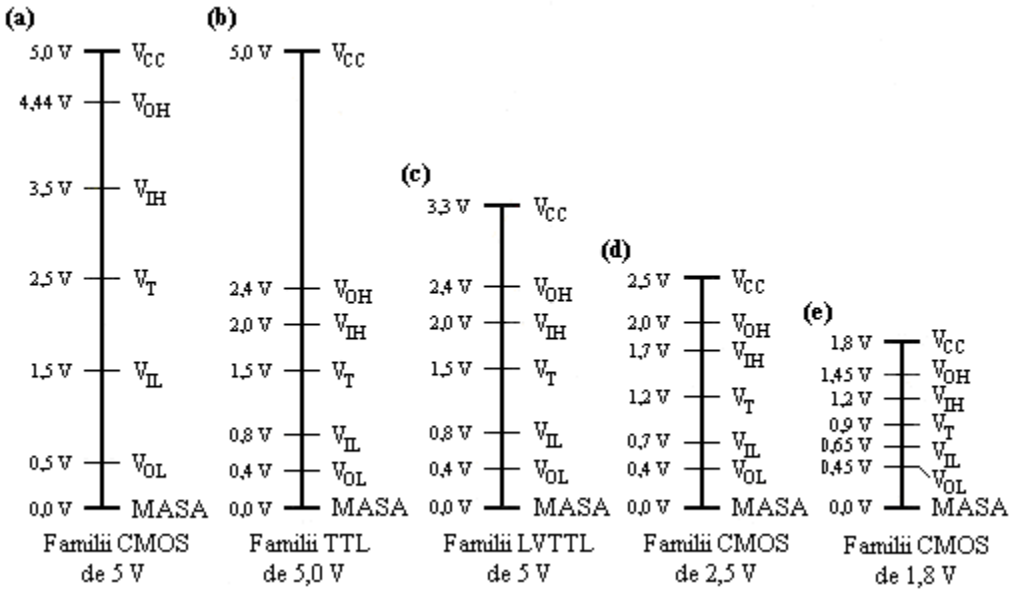


Figura 3-49 Comparaţie între nivelurile logice: (a) CMOS de 5V; (b) TTL de 5V şi CMOS de 5V compatibile cu TTL; (c) LVTTL de 3,3V; (d) CMOS de 2,5V; (e) CMOS de 1,8V

Interfeţe TTL/LVTTL – rezumat:

În conformitate cu cele expuse în subsecţiunile precedente, în acelaşi sistem pot fi utilizate atât dispozitive TTL (de 5V), cât şi LVTTL (de 3,3V), cu condiţia respectării următoarelor trei reguli:

- Ieşirile LVTTL pot comanda direct intrări TTL, ţinându-se seama doar de limitările impuse de obicei curentului de la ieşirea dispozitivelor de comandă (I_{OLmax}, I_{OHmax}).
- Ieşirile TTL pot comanda intrări LVTTL dacă intrările acceptă tensiuni de 5V.
- Ieşirile dispozitivelor TTL şi LVTTL cu trei stări pot comanda aceeaşi magistrală dacă ieşirile LVTTL suportă tensiuni de 5V.

4. Algebra circuitelor digitale

Circuitele logice sunt de două categorii: combinaţionale şi secvenţiale. *Circuite logice* combinaţionale sunt acele circuite ale căror ieşiri sunt funcţii exclusiv de starea curentă a intrărilor.

Ieşirile unui *circuit logic secvenţial* depind nu numai de starea curentă a intrărilor, ci şi de starea anterioară a acestora, care se poate să fi existat o perioadă de timp arbitrar îndelungată.

Circuitele combinaţionale pot fi formate dintr-un număr arbitrar de porţi logice şi inversoare, însă nu conţin nici o buclă de reacţie. *Bucla de reacţie* este o cale de semnal dintr-un circuit, prin care semnalul de la ieşirea unei porţi se poate întoarce la intrarea aceleiaşi porţi; în general, funcţionarea circuitelor secvenţiale se bazează pe acest tip de buclă.

În *analiza* circuitelor combinaţionale se porneşte de la o schemă logică din care se obţine o reprezentare formală a funcţiei realizate de acel circuit, de pildă un tabel de adevăr sau o expresie logică. În sinteză se procedează invers, adică se pleacă de la o reprezentare formală şi se obţine o schemă logică.

4.1 Algebra de comutaţie

La originea formalismului aplicat în metodele de analiză a circuitelor digitale se află opera unui matematician englez pe nume George Boole. În 1854, el a inventat un sistem algebric bazat pe două valori, numit astăzi *algebră booleană*.

Mult timp după apariţia operei lui Boole, în 1938, cercetătorul Claude E. Shannon, de la Bell Laboratories, a arătat cum poate fi adaptată algebra booleană în scopul descrierii şi analizării funcţionării circuitelor cu relee - elementele logice digitale cele mai larg utilizate în acea epocă. În *algebra de comutaţie* a lui Shannon, starea unui contact de releu - deschis sau închis - este reprezentată prin variabila X, care poate lua una dintre cele două valori posibile, 0 şi 1. În tehnologiile logice actuale, aceste valori sunt atribuite unei game largi de stări fizice – tensiune HIGH sau LOW, lumină aprinsă sau stinsă, condensator încărcat sau descărcat, siguranţă arsă sau intactă.

4.1.1 Axiomele algebrei booleene

Prin *convenţia de logică pozitivă* asociem valorii „0” o tensiune LOW şi valorii „1” o tensiune HIGH. Prin *convenţia de logică negativă* se face asocierea inversă: 0=HIGH şi 1=LOW. Însă utilizarea fie a logicii pozitive, fie a celei negative nu afectează posibilităţile de a reprezenta algebric fără echivoc funcţionarea unui circuit; sunt afectate doar detaliile formalismului de reprezentare algebrică a fenomenelor fizice.

Axiomele (sau *postulatele*) unui sistem matematic alcătuiesc un set minim de definiţii fundamentale pe care le acceptăm ca adevărate şi din care se deduc toate celelalte informaţii referitoare la acel sistem.

Vom nota cu X’ inversul (opusul sau complementul) lui X.

(A1): $X = 0$ dacă $X \neq 1$ (A1’): $X = 1$ dacă $X \neq 0$

(A2): Dacă $X = 0$, atunci $X' = 1$ (A2’): Dacă $X = 1$, atunci $X' = 0$

(A3): $0 \cdot 0 = 0$ (A3’): $1 + 1 = 1$

(A4): $1 \cdot 1 = 1$ (A4’): $0 + 0 = 0$

(A5): $0 \cdot 1 = 1 \cdot 0 = 0$ (A5’): $1 + 0 = 0 + 1 = 1$

Cele cinci perechi de axiome definesc complet algebra de comutaţie. Pornind de la ele se demonstrează toate celelalte propoziţii referitoare la acest sistem.

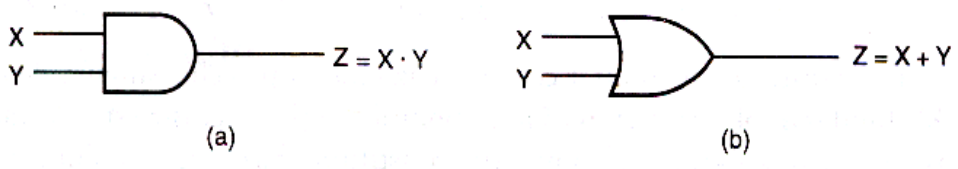


Figura 4-1 Denumirile semnalelor şi notaţia algebrică pentru: (a) poartă AND, (b) poartă OR

Funcţia realizată de o poartă **AND** cu două intrări este numită uneori *multiplicare logică* sau *produs logic* şi este simbolizată algebric printr-un *punct*

de multiplicare. Deci semnalul de ieşire al unei porţi **AND** cu intrările X şi Y are valoarea X • Y, cum arată fig. 4-1 (a).

Funcţia realizată de o poartă **OR** cu două intrări este numită uneori *sumă logică* şi este simbolizată algebric prin semnul plus (+). Semnalul de ieşire al unei porţi **OR** cu intrările X şi Y are valoarea X + Y, ca în fig. 4-1 (b).

Prin convenţie, într-o expresie logică în care apar atât operaţii de multiplicare, cât şi de însumare, multiplicarea are *prioritate, la* fel ca în expresiile cu întregi din limbajele de programare convenţionale. În consecinţă, expresia W•X+Y•Z este echivalentă cu (W•X)+(Y•Z).

Ultimele trei perechi de axiome definesc formal operaţiile **AND** şi **OR** arătând ce valoare apare la ieşirea fiecărui tip de poartă pentru fiecare combinaţie de intrare posibilă.

4.1.2 Teoreme pentru o singură variabilă

În desfăşurarea analizei sau sintezei circuitelor logice scriem frecvent expresii algebrice care caracterizează funcţionarea reală sau propusă a unui circuit. În algebra de comutaţie, *teoremele* sunt enunţuri demonstrate ca adevărate, care ne permit să efectuăm o analiză mai simplă sau o sinteză mai eficientă a circuitelor respective. De exemplu, teorema X + 0 = X ne permite să înlocuim prin X, într-o expresie, orice apariţie a sumei X + 0.

Tabel 4-1 Teoreme pentru o variabilă în algebra de comutaţie

(T1)	X + 0 = X	(T1')	X · 1 = X	(Identităţi)
(T2)	X + 1 = 1	(T2')	X ·0 = 0	(Elemente nule)
(T3)	X + X = X	(T3')	X · X =X	(Idempotenţă)
(T4)	(X')' = X	(T4')		(Involuţie)
(T5)	X + X' = 1	(T5')	X · X' = 0	(Complemente)

Tabelul 4-1 prezintă teoremele algebrei de comutaţie referitoare la o singură variabilă, X. Majoritatea teoremelor din algebra de comutaţie se demonstrează extrem de simplu printr-o metodă numită *inducţie perfectă.* Axioma de bază a acestei metode este Al: întrucât o variabilă de comutaţie poate lua doar două valori diferite, teoremele pentru o singură variabilă se demonstrează verificându-le atât pentru X = 0, cât şi pentru X = 1. De exemplu, pentru demonstrarea teoremei T1 se efectuează două substituţii:

[X = 0] 0 + 0 = 0 adevărat, conform axiomei A4'
[X = 1] 1 + 0 = 1 adevărat, conform axiomei A5'

4.1.3 Teoreme pentru două şi trei variabile

Teoremele algebrei de comutaţie referitoare la două şi trei variabile sunt prezentate în tabelul 4-2. Fiecare dintre ele se demonstrează simplu prin inducţie perfectă, verificând relaţiile corespunzătoare pentru cele patru combinaţii posibile de X şi Y sau pentru cele opt combinaţii de X, Y şi Z.

Tabel 4-2 Teoremele algebrei de comutaţie pentru două şi trei variabile

(T6)	X + Y = Y + X	(T6')	X · Y = Y ·X	(Comutativitate)
(T7)	(X + Y) + Z = X + (Y + Z)	(T7')	(X · Y) · Z = X · (Y · Z)	(Asociativitate)
(T8)	X · Y + X · Z = X · (Y + Z)	(T8')	(X + Y) · (X + Z) = X + Y · Z	(Distributivitate)
(T9)	X + X · Y = X	(T9')	X · (X + Y) = X	(Acoperire)
(T10)	X · Y + X · Y' = X	(T10')	(X + Y) · (X + Y') = X	(Combinare)
(T11)	X · Y + X' · Z + Y · Z = X · Y + X' · Z			(Consens)
(T11')	(X + Y) · (X' + Z) · (Y + Z) = (X Y) · (X' + Z)			

Primele două perechi de teoreme se referă la comutativitatea şi asociativitatea sumei logice şi produsului logic şi sunt identice cu legile de comutativitate şi asociativitate aferente numerelor întregi şi reale. În ansamblu, ele arată că nu este necesar să se ţină cont de includerea între paranteze sau de scrierea într-o anumită ordine a termenilor unei sume logice sau ai unui produs logic. De exemplu, din punct de vedere strict algebric, o expresie de genul W • X • Y • Z este ambiguă; sunt de preferat variantele (W • (X • (Y • Z))) sau (((W • X) Y) • Z) sau (W • X) • (Y • Z). Teoremele arată însă că forma ambiguă a expresiei este corectă, întrucât în oricare dintre cazuri se obţine acelaşi rezultat. Putem chiar să schimbăm ordinea variabilelor (de exemplu, X • Z • Y • W) şi rezultatul va fi acelaşi.

Oricât de banală ar părea această discuţie, ea are mare importanţă, deoarece constituie baza teoretică a utilizării porţilor logice cu mai mult de două intrări. S-au definit operatorii • şi + ca *operatori binari* - operatori ce intervin între *două* variabile. În practică însă utilizăm porţi **AND** şi **OR** cu 3, 4 sau mai multe intrări. Teoremele ne arată că putem conecta intrările porţilor în orice ordine; de fapt, multe programe de dispunere a componentelor pe cartelele cu circuit imprimat şi în interiorul ASIC exploatează această posibilitate. Astfel, se pot folosi fie o poartă cu *n* intrări, fie *(n* - 1) porţi cu două intrări, deşi, probabil, timpul de propagare şi preţul sunt mai mari în cazul folosirii mai multor porţi cu două intrări.

Teorema T8 este identică legii de distributivitate pentru numere întregi şi reale, adică produsul logic este distributiv faţă de suma logică. În consecinţă, putem „multiplica" o expresie punând-o sub forma unei sume de produse, ca în exemplul de mai jos:

$$V•(W+X)•(Y+Z)=V•W•Y+V•W•Z+V•X•Y+V•X•Z$$

Algebra de comutaţie prezintă însă şi o proprietate neobişnuită: inversa acestei teoreme este, de asemenea, adevărată – suma logică este distributivă faţă de produsul logic – aşa cum reiese din teorema T8'. Deci, putem extinde o expresie şi sub formă de produs de sume:

$$(V•W•X)+(Y•Z)=(V+Y)•(V+Z)•(W+Y)•(W+Z)•(X+Y)•(X+Z)$$

Teoremele T9 şi T10 sunt mult utilizate la minimizarea funcţiilor logice. De exemplu, dacă subexpresia X + X • Y este inclusă într-o expresie logică, *teorema de acoperire* T9 arată că este suficientă includerea în acea expresie a variabilei X; se spune ca *X acoperă X • Y. Teorema de combinare T10* arată că dacă într-o expresie este inclusă subexpresia X • Y + X • Y', aceasta poate fi înlocuită prin X. Întrucât Y poate fi ori 0, ori 1, în oricare caz, subexpresia originală poate lua valoarea 1 dacă şi numai dacă X este 1.

Deşi T9 poate fi demonstrată cu uşurinţă prin inducţie perfectă, corectitudinea ei reiese mai evident dacă vom folosi în demonstraţie celelalte teoreme deja demonstrate:

$$\begin{aligned} X + X • Y &= X • 1 + X • Y \text{ (în conformitate cu T1')} \\ &= X • (1 + Y) \text{ (în conformitate cu T8)} \\ &= X • 1 \text{ (în conformitate cu T2)} \\ &= X \text{ (în conformitate cu T1')} \end{aligned}$$

În mod asemănător, celelalte teoreme pot fi folosite şi pentru a demonstra T10, principalul artificiu fiind aici rescrierea membrului stâng ca X • (Y + Y'), conform teoremei T8.

Teorema T11 este cunoscută ca *teorema de consens.* Termenul Y • Z este numit *consens* între X • Y şi X' • Z. Se pleacă de la ideea că dacă Y • Z este 1, atunci fie X • Y, fie X' • Z trebuie să fie tot 1, întrucât atât Y, cât şi Z sunt 1 şi fie X, fie X' trebuie să fie 1. Prin urmare, termenul Y • Z este redundant şi poate fi eliminat din membrul drept al relaţiei T11. Teorema de consens are două aplicaţii importante. Ea poate fi utilizată pentru eliminarea anumitor incertitudini de temporizare caracteristice circuitelor logice combinaţionale. De asemenea, teorema de consens constituie baza metodei consensului iterativ de aflare a implicanţilor primi.

În toate teoremele este posibilă înlocuirea oricărei variabile cu o expresie logică arbitrară. Una dintre înlocuirile simple constă în complementarea uneia sau a mai multor variabile:

$$(X +Y') + Z' = X + (Y' + Z') \text{ (pe baza teoremei T7)}$$

Se pot înlocui însă şi expresii mai complicate:

(V'+X)•(W•(Y'+Z))+(V'+X)•(W•(Y'+Z))'=V'+X (pe baza teoremei T10)

4.1.4 Teoreme pentru n variabile

Câteva teoreme importante, prezentate în tabelul 4-3, se verifică pentru orice număr *n* de variabile. Majoritatea lor se demonstrează printr-o metodă – denumită *inducţie finită* – ce implică două etape: într-o primă etapă se arată că teorema se verifică pentru *n = 2 (etapa de bază),* iar apoi se arată că dacă teorema se verifică pentru *n* = i, atunci ea se verifică şi pentru *n* = i + 1 *(etapa de inducţie).* De exemplu, să considerăm teorema de idempotenţă generalizată, *T12.* Pentru *n* = 2, T12 este echivalentă cu T3, deci se verifică. Dacă se verifică pentru o sumă logică de i variabile X, atunci se verifică şi pentru suma a i + 1 variabile X, conform raţionamentului următor:

X + X + ... +X = X + (X + ... +X) (i + 1 variabile X în ambii membri)
= X + (X) (dacă T12 se verifică pentru n = i)
= X (în conformitate cu T3)

Deci teorema se verifică pentru orice valoare finită *n.*

Teoremele lui DeMorgan (T13 şi T13') sunt, probabil, teoremele cele mai frecvent utilizate din întreaga algebră de comutaţie. Teorema T13 afirmă că o poartă **AND** cu n intrări şi ieşirea complementată este echivalentă cu o poartă ***OR*** *cu n* intrări complementate. Prin urmare, circuitele din fig. 4-2 (a) şi (b) sunt echivalente.

Tabel 4-3 Teoremele algebrei de comutaţie pentru n variabile

(T12)	$X + X + \ldots + X = X$	(Idempotenţă generalizată)
(T12')	$X \cdot X \cdot \ldots \cdot X = X$	
T(13)	$(X_1 \cdot X_2 \cdot \ldots \cdot X_n)' = X_1' + X_2' + \ldots + X_n'$	(Teoremele lui DeMorgan)
(T13')	$(X_1 + X_2 + \ldots + X_n)' = X_1' \cdot X_2' \cdot \ldots \cdot X_n'$	
(T14)	$[F(X_1, X_2, \ldots, X_n, +, \cdot)]' = F(X_1', X_2', \ldots, X_n', \cdot, +)$	(Teorema lui DeMorgan generalizată)
(T15)	$F(X_1, X_2, \ldots, X_n) = X_1 \cdot F(1, X_2, \ldots, X_n) + X_1' \cdot F(0, X_2, \ldots, X_n)$	(Teoremele de expansiune ale lui Shannon)
(T15')	$F(X_1, X_2, \ldots, X_n) = [X_1 + F(0, X_2, \ldots, X_n)] + [X_1' + F(1, X_2, \ldots, X_n)]$	

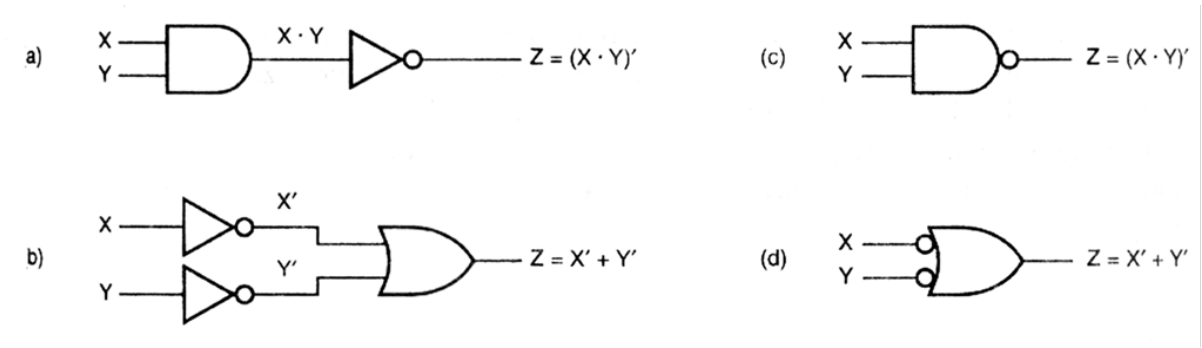


Figura 4-2 Circuite echivalente conform teoremei T13, a lui DeMorgan: (a) AND-NOT; (b) NOT-OR; (c) simbolul logic al unei porţi NAND; (d) simbol echivalent al unei porţi NAND

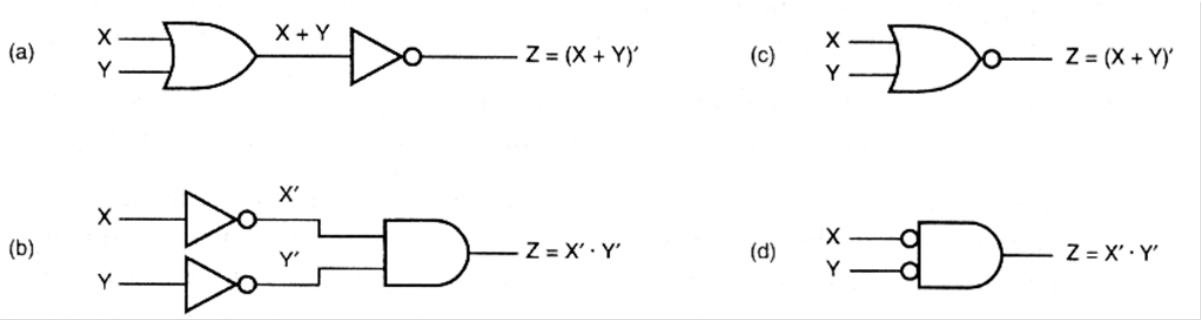


Figura 4-3 Circuite echivalente conform teoremei T13’, a lui DeMorgan: (a) OR-NOT; (b) NOT-AND; (c) simbolul logic al unei porţi NOR; (d) simbol echivalent al unei porţi NOR

Teoremele T13 şi T13' sunt cazuri particulare ale *teoremei generalizate a lui DeMorgan,* T14, aplicabilă unei expresii logice oarecare, F. Prin definiţie, *complementul unei expresii logice,* notat (F)', este o expresie a cărei valoare este opusă valorii expresiei F pentru orice combinaţie de intrare posibilă. Teorema T14 este foarte importantă deoarece ne oferă o cale de prelucrare şi simplificare a complementului unei expresii.

Teorema T14 afirmă că, dată fiind o expresie logică de n variabile, complementul acesteia se poate obţine înlocuind + cu • şi invers şi complementând toate variabilele. De exemplu, să considerăm expresia:

$$F(W, X, Y, Z) = (W' \bullet X) + (X \bullet Y) + (W \bullet (X' + Z'))$$
$$= ((W)' \bullet X) + (X \bullet Y) + (W \bullet ((X)' + (Z)'))$$

În rândul al doilea am introdus între paranteze variabilele complementate pentru a vă aminti că ' nu face parte din denumirea variabilei, ci este un operator. Aplicând teorema T14 obţinem:

$$[F(W,X,Y,Z)]' = ((W')' + X') \bullet (X' + Y') \bullet (W' + ((X')' \bullet (Z')'))$$

Cu teorema T4, expresia se simplifică la:

$$[F(W,X,Y,Z)]' = (W + X') \bullet (X' + Y') \bullet (W' + (X \bullet Z))$$

4.1.5 Dualitatea

Toate axiomele algebrei de comutaţie au fost enunţate în perechi. Varianta notată cu prim a fiecărei axiome (de exemplu A5') se obţine din varianta fără notaţia prim (de exemplu A5) prin simpla schimbare între 0 şi 1 şi între • şi +, atunci când aceşti operatori apar. În consecinţă, putem enunţa următoarea *metateoremă – o* teoremă despre teoreme:
Principiul dualităţii:

"**Orice teoremă sau identitate din algebra de comutaţie îşi păstrează valabilitatea dacă se înlocuiesc reciproc, peste tot, 0 şi 1 şi • şi +.**"

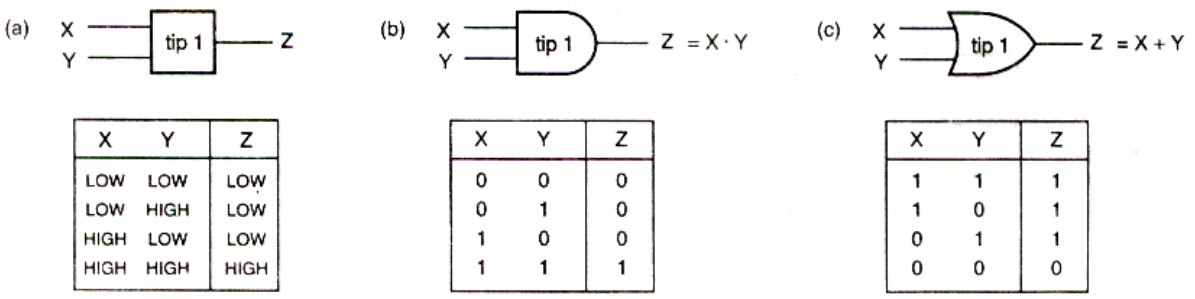
Dualitatea prezintă importanţă deoarece tot ceea ce a fost descris despre algebra de comutaţie şi despre tratarea funcţiilor de comutaţie are o dublă utilitate. De exemplu, dacă s-a arătat cum se sintetizează circuitele logice **AND-OR** cu două etaje pornind de la o expresie formată dintr-o sumă de produse, se cunoaşte automat şi metoda duală de sintetizare a circuitelor **OR-AND** pornind de la o expresie formată dintr-un produs de sume.

În algebra de comutaţie există un singur caz în care • şi + sunt tratate diferit, deci dualitatea nu este totdeauna valabilă. Să considerăm următorul enunţ al teoremei T9 şi „duala" sa, în mod clar absurdă:

$$X + X \bullet Y = X \qquad \text{(teorema T9)}$$
$$X \bullet X + Y = X \qquad \text{(după aplicarea principiului dualităţii)}$$
$$X + Y = X \qquad \text{(după aplicarea teoremei T3')}$$

Evident, ultimul rând de mai sus este un enunţ fals; unde este eroarea? Problema constă în respectarea ordinei operatorilor. Membrul stâng al relaţiei din primul rând a fost scris corect fără paranteze deoarece, prin convenţie, operatorul • are prioritate. Însă, după aplicarea principiului dualităţii, ar fi trebuit să acordăm prioritate operatorului + sau să scriem cel de-al doilea rând sub forma $X \bullet (X + Y) = X$. Cea mai bună modalitate de a evita erorile de acest gen este scrierea cu toate parantezele a expresiei înainte de a trece la duala acesteia.

În fig. 4-4 (a) apare tabelul de adevăr al semnalelor electrice caracteristice funcţiei unui element logic pe care îl vom numi simplu „poartă de tipul 1". În convenţia de logică pozitivă (LOW = 0 şi HIGH = 1), aceasta este o poartă **AND**, însă în convenţia de logică negativă (LOW = 1 şi HIGH = 0), este o poartă **OR**, cum arată fig. (b) şi (c). Ne putem imagina şi o poartă „de tipul 2", ca aceea din fig. 4-5, care realizează funcţia **OR** în logică pozitivă şi **AND** în logică negativă. Asemenea tabele se pot scrie şi pentru porţi cu mai mult de două intrări.

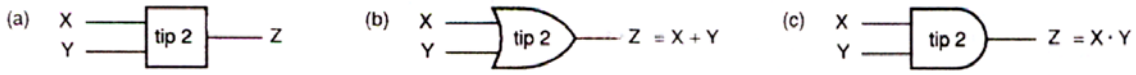


X	Y	Z
LOW	LOW	LOW
LOW	HIGH	LOW
HIGH	LOW	LOW
HIGH	HIGH	HIGH

X	Y	Z
0	0	0
0	1	0
1	0	0
1	1	1

X	Y	Z
1	1	1
1	0	1
0	1	1
0	0	0

Figura 4-4 Poartă logică de „tipul 1”: (a) tabelul logic al semnalelor electrice; (b) tabelul funcţiei logice şi simbolul în logică pozitivă; (c) tabelul funcţiei logice şi simbolul în logică negativă



X	Y	Z
LOW	LOW	LOW
LOW	HIGH	HIGH
HIGH	LOW	HIGH
HIGH	HIGH	HIGH

X	Y	Z
0	0	0
0	1	1
1	0	1
1	1	1

X	Y	Z
1	1	1
1	0	0
0	1	0
0	0	0

Figura 4-5 Poartă logică de „tipul 2”: (a) tabelul logic al semnalelor electrice; (b) tabelul funcţiei logice şi simbolul în logică pozitivă; (c) tabelul funcţiei logice şi simbolul în logică negativă

Să considerăm o expresie logică oarecare, $F(X_1, X_2, ..., X_n)$. Respectând convenţia de logică pozitivă, putem construi un circuit care să realizeze această funcţie folosind inversoare pentru operaţia de negare, porţi de tipul 1 pentru **AND** şi porţi de tipul 2 pentru **OR**, ca în fig. 4-6. Acum să presupunem că nu schimbăm nimic în circuit, însă trecem de la logica pozitivă la cea negativă. Atunci, circuitul trebuie refăcut ca în fig. 4-7. Este clar că pentru toate combinaţiile posibile de semnale de intrare (HIGH şi LOW), circuitul generează aceeaşi tensiune de ieşire. Dar, din punctul de vedere al algebrei de comutaţie, valoarea de ieşire - 0 sau 1 - este opusă celei obţinute în convenţia de logică pozitivă. Analog, fiecare valoare de intrare reprezintă acum opusul celei anterioare. Prin urmare, pentru orice combinaţie posibilă de semnale de la intrarea circuitului din fig. 4-6, la ieşire se obţine opusul valorii rezultate din aplicarea combinaţiei de valori opuse la intrările circuitului din fig. 4-7:

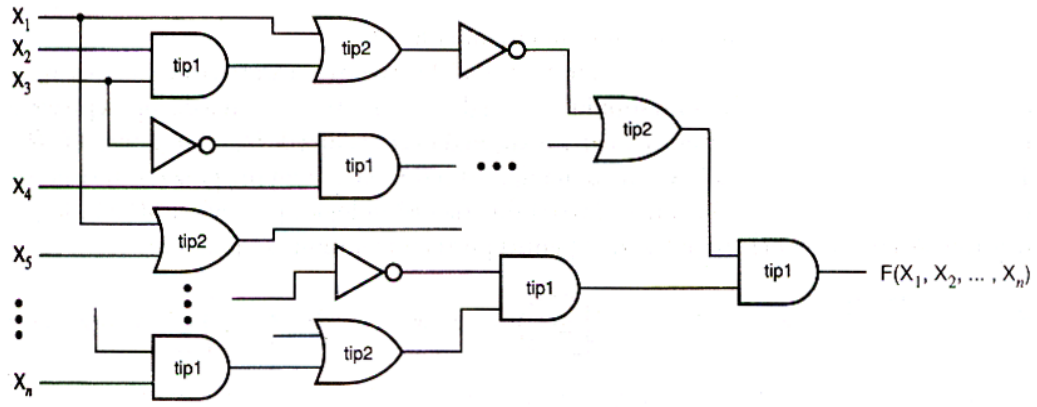


Figura 4-6 Circuit ce realizează o funcţie logică folosind inversoare şi porţi de tipurile 1 şi 2, în convenţia de logică pozitivă

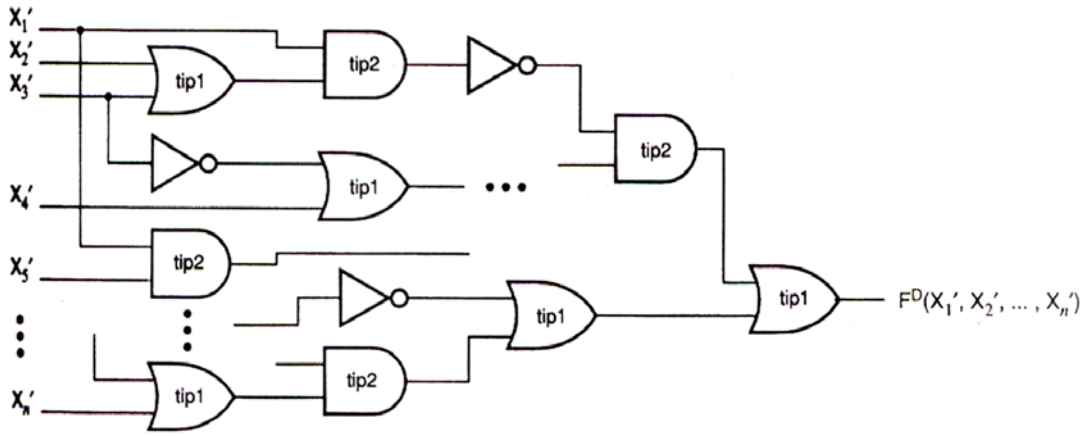


Figura 4-7 Interpretarea în logică negativă a circuitului prezentat anterior.

Prin complementarea ambilor membri se obţine teorema lui DeMorgan generalizată:

$$[F(X_1,X_2,...,X_n)]' = F^D(X1',X_2',...,X_n')$$

Tabel 4-4 Forma generală a tabelului de adevăr al unei funcţii logie de trei variabile F(X, Y, Z)

Rândul	X	Y	Z	F
0	0	0	0	F (0, 0, 0)
1	0	0	1	F (0, 0, 1)
2	0	1	0	F (0, 1, 0)
3	0	1	1	F (0, 1, 1)
4	1	0	0	F (1, 0, 0)
5	1	0	1	F (1, 0, 1)
6	1	1	0	F (1, 1, 0)
7	1	1	1	F (1, 1, 1)

Reprezentarea de bază a unei funcţii logice este tabelul de adevăr. Bazându-se pe o concepţie similară metodei de demonstrare prin inducţie perfectă, această reprezentare grosieră este, pur şi simplu, o listă a valorilor obţinute la ieşirea unui circuit pentru toate combinaţiile de intrare posibile. Tradiţional, combinaţiile de intrare sunt aranjate pe o coloană în ordinea crescătoare a valorilor binare, iar valorile de ieşire corespunzătoare apar în coloana alăturată. Forma generală a unui tabel de adevăr pentru 3 variabile este prezentată în tabelul 4-4.

Rândurile sunt numerotate de la 0 la 7, corespunzător combinaţiilor binare de intrare, dar această numerotare nu este esenţială pentru conţinutul tabelului de adevăr. În tabelul 4-5 este prezentat tabelul de adevăr pentru un caz particular de funcţie logică de trei variabile. Fiecare combinaţie de 0 şi 1 din coloana valorilor de ieşire realizează o funcţie logică diferită de celelalte; există 2^8 asemenea combinaţii. Deci funcţia logică din tabelul 4-5 este una dintre cele 2^8 funcţii logice diferite, de trei variabile.

Tabelul de adevăr corespunzător unei funcţii logice de n variabile conţine 2^n rânduri.

Tabel 4-5 Tabelul de adevăr pentru un caz particular de funcţie logică de trei variabile F(X, Y, Z)

Rândul	X	Y	Z	F
0	0	0	0	1
1	0	0	1	0
2	0	1	0	0
3	0	1	1	-1
4	1	0	0	1
5	1	0	1	0
6	1	1	0	1
7	1	1	1	1

Informaţiile conţinute de un tabel de adevăr pot fi interpretate şi algebric. În acest scop ne sunt necesare câteva definiţii:

- Variabilele sau complementele variabilelor reprezentate printr-o singură literă se numesc *variabile literale.* Exemple: **X, Y, X', Y.**
- Un *termen produs* este o variabilă literală sau produsul logic a două sau mai multe variabile literale. Exemple: **Z, W • X • Y, X • Y' • Z, W' • Y' • Z**.
- *O expresie sumă de produse* este suma logică a unor termeni produs. Exemple:

Z' + W • X • Y + *X* • Y' • Z + W' • Y' • *Z*.

- Un *termen sumă* este o variabilă literală sau suma logică a două sau mai multe variabile literale. Exemple: **Z', W + X + Y, X + Y' + Z, W' + Y' + Z**.

- *O expresie produs de sume* este produsul logic al unor termeni sumă. Exemple:

Z' • (W + X+ + Y)• (X + Y' + Z)• (W' +Y' + Z).

- Un *termen normal* este un produs sau o sumă în care nici o variabilă nu se repetă. Un termen non-normal poate fi adus totdeauna la forma unei constante sau a unui termen normal prin aplicarea uneia dintre teoremele T3, T3', T5 sau T5'. Exemple de termeni non-normali:

W • X • X • Y', W + W + X'+ Y, X • X' • Y.

Exemple de termeni normali: **W • X • Y', W + X' + Y**.

- Un *mintermen* de *n* variabile este un termen normal, produs a *n* variabile literale. Există 2^n asemenea produse. Exemple de mintermeni de 4 variabile:

W' • X' • Y' • Z', W • X • Y • Z, W' • X' • Y • Z'.

- Un *maxtermen* de *n* variabile este un termen normal, sumă a n variabile literale. Există 2^n asemenea sume. Exemple de maxtermeni de 4 variabile:

W' + X' + Y' + Z', W + X' + Y' + Z, W' + X' + Y + Z'.

Există o corespondenţă strânsă între tabelul de adevăr, mintermeni şi maxtermeni.

Un mintermen poate fi definit ca un termen produs care are valoarea 1 într-un singur rând al tabelului de adevăr. Analog, un maxtermen poate fi definit ca un termen sumă care are valoarea 0 într-un singur rând al tabelului de adevăr. Tabelul 4-6 înfăţişează această corespondenţă într-un tabel de adevăr pentru trei variabile.

Tabel 4-6 Mintermenii şi maxtermenii unei funcţii logice de trei variabile, F(X, Y, Z)

Rândul	X	Y	Z	F	Mintermeni	Maxtermeni
0	0	0	0	F (0, 0, 0)	X’ · Y’ · Z’	X + Y + Z
1	0	0	1	F (0, 0, 1)	X’ · Y’ · Z	X + Y + Z’
2	0	1	0	F (0, 1, 0)	X’ · Y · Z’	X + Y’ + Z
3	0	1	1	F (0, 1, 1)	X’ · Y · Z	X + Y’ + Z’
4	1	0	0	F (1, 0, 0)	X · Y’ · Z’	X’ + Y + Z
5	1	0	1	F (1, 0, 1)	X · Y’ · Z	X’ + Y + Z’
6	1	1	0	F (1, 1, 0)	X · Y · Z’	X’ + Y’ + Z
7	1	1	1	F (1, 1, 1)	X · Y · Z	X’ + Y’ + Z’

Un mintermen de *n* variabile poate fi reprezentat printr-un întreg de *n* biţi, denumit *numărul mintermenului.* Mintermenul corespunzător rândului *i* din tabelul de adevăr va fi numit *mintermenul i.* În acest mintermen, o variabilă dată apare complementată dacă bitul corespunzător ei în reprezentarea binară a numărului i este 0; în caz contrar, variabila apare necomplementată. De exemplu, pentru rândul 5, reprezentarea binară este 101, iar mintermenul

corespunzător este X • Y' • Z. Aşa cum, probabil, vă aşteptaţi, în cazul maxtermenilor, totul este invers: în maxtermenul i, o variabilă este complementată dacă bitul corespunzător ei în reprezentarea binară este 1. Prin urmare, maxtermenul 5 (101) este X' + Y + Z'. Remarcaţi că toate cele expuse până acum sunt aplicabile dacă ştim numărul de variabile din tabelul de adevăr - trei, în exemple.

Plecând de la corespondenţa dintre tabelul de adevăr şi mintermeni, putem obţine cu uşurinţă o reprezentare algebrică a funcţiei logice, dedusă din tabelul de adevăr. Suma canonică a unei funcţii logice este suma mintermenilor corespunzători rândurilor (combinaţiilor de intrare) din tabelul de adevăr pentru care valoarea de ieşire a funcţiei este 1. De exemplu, suma canonică a funcţiei logice din tabelul 4-5 este:

$$F = \Sigma_{X, Y, Z} (0, 3, 4, 6, 7)$$
$$= X' \bullet Y' \bullet Z' + X' \bullet Y \bullet Z + X \bullet Y' \bullet Z' + X \bullet Y \bullet Z' + X \bullet Y \bullet Z$$

Aici, notaţia reprezintă o listă a mintermenilor cu semnificaţia „suma mintermenilor 0, 3, 4, 6 şi 7 cu variabilele X, Y şi Z”. Lista mintermenilor mai este denumită şi *deschiderea* funcţiei logice (on-set). Vă puteţi da seama că fiecare mintermen „deschide” ieşirea exact pentru o singură combinaţie de intrare. Orice funcţie logică poate fi scrisă sub forma unei sume canonice.

Produsul canonic al unei funcţii logice este produsul maxtermenilor corespunzători combinaţiilor de intrare pentru care valoarea de ieşire a funcţiei este 0. De exemplu, produsul canonic al funcţiei logice din tabelul 4-5 este:

$$F = \Sigma_{X, Y, Z} (1, 2, 5)$$
$$= (X + Y + Z') \bullet (X + Y' + Z) \bullet (X' + Y + Z')$$

Aici, notaţia reprezintă o listă a maxtermenilor cu semnificaţia „produsul maxtermenilor 1, 2 şi 5 cu variabilele X, Y şi Z”. Lista maxtermenilor mai este denumită şi *închiderea* funcţiei logice (off-set). Vă puteţi da seama că fiecare maxtermen „închide” ieşirea exact pentru o singură combinaţie de intrare. Orice funcţie logică poate fi scrisă sub forma unui produs canonic.

Conversia între lista de mintermeni şi cea de maxtermeni se efectuează uşor. În cazul unei funcţii de n variabile, mintermenii şi maxtermenii pot avea numere din mulţimea 0, 1, 2^n - 1; o listă de mintermeni sau de maxtermeni este constituită dintr-o submulţime a acestei mulţimi.

De exemplu:

$$\Sigma_{A, B, C} (0, 1, 2, 3) = \Pi_{A, B, C} (4, 5, 6, 7)$$
$$\Sigma_{X, Y} (l) = \Pi_{X, Y} (0, 2, 3)$$
$$\Sigma_{W, X, Y, Z} (0, 1, 2, 3, 5, 7, 11, 13) = \Pi_{W, X, Y, Z} (4, 6, 8, 9, 10, 12, 14, 15)$$

Cunoaştem acum cinci moduri de reprezentare a unei funcţii logice combinaţionale:

1. Tabelul de adevăr.
2. Suma algebrică a mintermenilor, adică suma canonică.
3. Lista mintermenilor, cu notaţia Σ.
4. Produsul algebric al maxtermenilor, adică produsul canonic.
5. Lista maxtermenilor, cu notaţia Π.

Fiecare dintre aceste reprezentări oferă exact aceleaşi informaţii; cunoscând pe oricare dintre ele, le putem deduce pe celelalte patru printr-un mecanism de calcul simplu.

4.2 Transformări de configuraţii

Metodele de proiectare prezentate până acum folosesc porţi **AND**, **OR** şi **NOT**. Dar se poate întâmpla să dorim să utilizăm şi porţi **NAND** şi **NOR**, acestea fiind, în majoritatea tehnologiilor, mai rapide decât porţile **AND** şi **OR**. Însă, de obicei, oamenii nu construiesc propoziţii logice cu ajutorul conjuncţiilor **NAND** şi **NOR**. Prin urmare, dacă se dă o expresie logică "firească", aceasta trebuie transpusă, prin anumite metode, în alte forme.

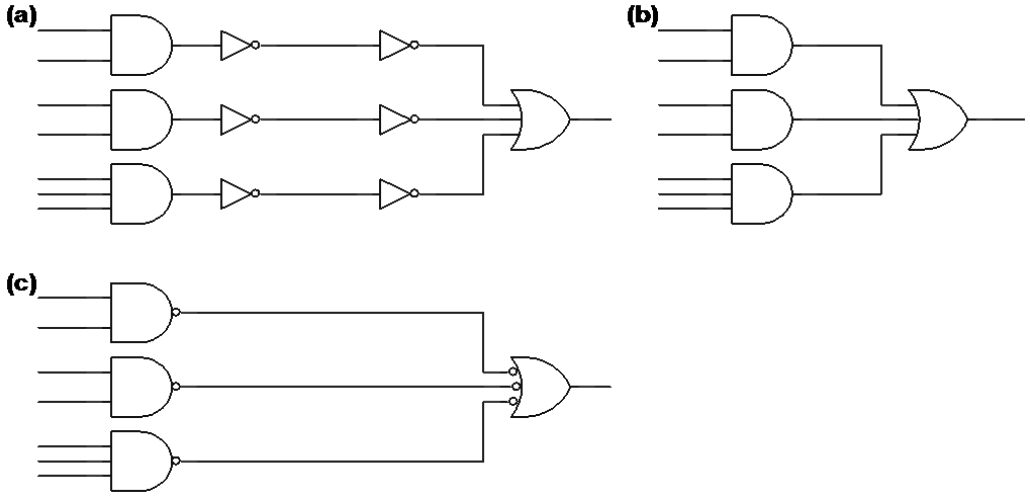


Figura 4-8 Implementări alternative pornind de la sume de produse: (a) cu porţi AND-OR; (b) cu porţi AND-OR şi perechi de inversoare suplimentare; (c) cu porţi NAND-NAND

Putem transforma orice expresie logică într-o sumă de produse echivalentă, prin simpla deschidere a parantezelor. Aşa cum arată fig. 4-8 (a), o asemenea expresie poate fi implementată direct cu porţi **AND** şi **OR**. Inversoarele necesare pentru complementarea intrărilor nu sunt reprezentate în desen.

După cum observaţi în fig. 4-8 (b), se poate introduce câte o pereche de inversoare între fiecare ieşire a unei porţi **AND** şi intrarea porţii **OR** corespunzătoare dintr-un circuit **AND-OR** cu două niveluri. Conform teoremei T4, aceste inversoare nu modifică funcţia de ieşire a circuitului. Am reprezentat intenţionat cerculeţul inversor *la intrarea* celui de-al doilea inversor din fiecare pereche, pentru a vă aminti şi prin imagine că inversările se anulează reciproc. Dacă încorporăm aceste inversoare în porţile **AND** şi **OR**, obţinem porţi **AND-NOT** pe primul nivel şi **NOT-OR** pe nivelul al doilea. Acestea nu sunt decât două variante de reprezentare a aceluiaşi tip de poartă – NAND. Prin urmare, un *circuit* cu două niveluri **AND-OR** poate fi transformat într-un *circuit* cu două niveluri **NAND-NAND** prin simpla substituire a porţilor.

Dacă oricare dintre produsele din expresia sumă de produse este format dintr-o singură variabilă literală, este posibil ca, în cursul transformării din **AND-OR** în **NAND-NAND,** să pierdem sau să câştigăm inversoare. Astfel, în exemplul din fig. 4-9 nu mai este necesară prezenţa unui inversor la intrarea W, însă trebuie adăugat unul la intrarea Z.

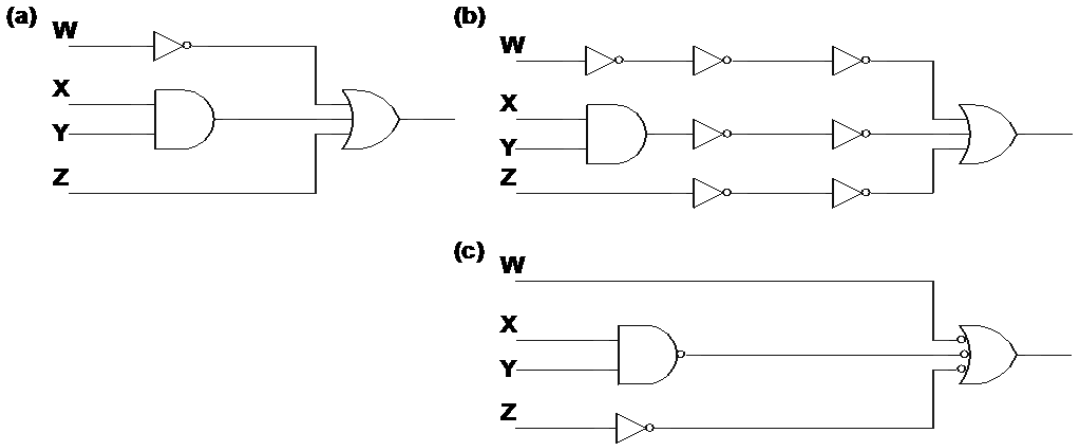


Figura 4-9 Alte circuite cu două niveluri obţinute pe baza unor sume de produse: (a) AND-OR; (b) AND-OR cu perechi de inversoare suplimentare; (c) NAND-NAND

Am arătat că orice expresie sumă de produse poate fi implementată în două moduri, fie ca un circuit cu porţi **AND-OR**, fie ca unul **NAND-NAND**. Duala acestei afirmaţii este, de asemenea, valabilă: orice expresie produs de sume poate fi implementată ca un circuit **OR-AND** sau ca un circuit **NOR-NOR**. Un exemplu este cel din fig. 4-10. Orice expresie logică poate fi transformată într-o expresie produs de sume echivalentă, prin gruparea corespunzătoare a termenilor, şi deci poate fi implementată atât printr-un circuit **OR-AND**, cât şi printr-unul **NOR-NOR**.

Aceeaşi modalitate de transformare poate fi aplicată oricărui circuit logic. De exemplu, în fig. 4-11 (a) este prezentat un circuit construit din porţi **AND** şi **OR**. După adăugarea perechilor de inversoare se obţine circuitul din (b). Observaţi însă că una dintre porţi - poarta **AND** cu două intrări, dintre care o

singură intrare este inversoare - nu este o poartă standard. Putem folosi un inversor separat, ca în fig. (c), pentru a obţine un circuit format exclusiv din porţi standard, **AND**, **NAND** şi inversoare. Însă inversorul este mai bine folosit în fig. (d); se elimină astfel întârzierea introdusă de un nivel de porţi, iar ultima poartă devine **NOR**, în loc de **AND**. În majoritatea tehnologiilor de circuite logice, porţile inversoare, ca **NAND** şi **NOR**, sunt mai rapide decât cele neinversoare, ca **AND** şi **OR**.

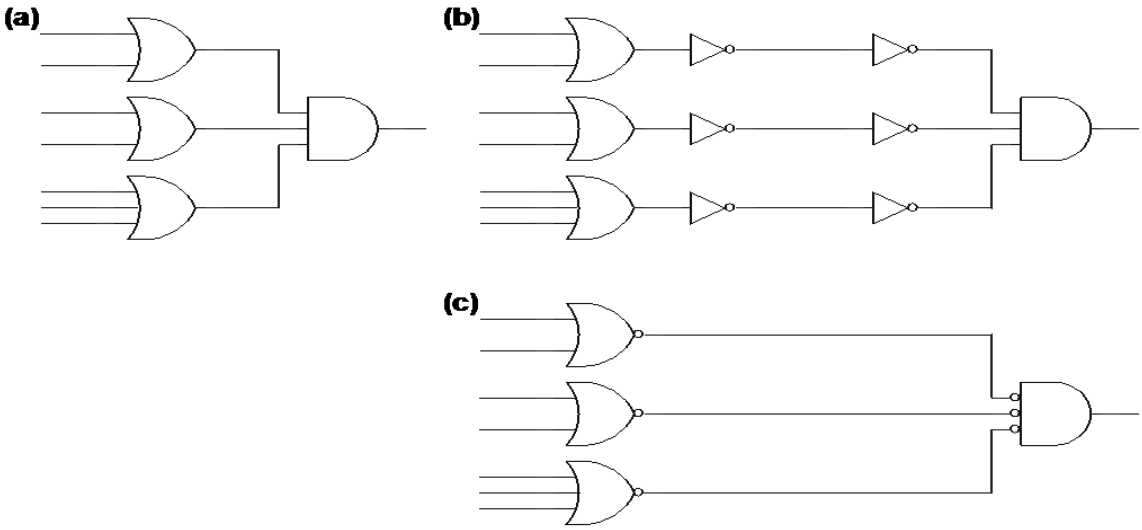


Figura 4-10 Implementarea unei expresii produs de sume: (a) OR-AND; (b) OR-AND cu perechi de inversoare suplimentare; (c) NOR-NOR

4.3 Minimizarea circuitelor combinaţionale

Adesea este neeconomic să implementaţi un circuit logic direct, pornind de la prima expresie. În special expresiile canonice – sumă şi produs – duc la soluţii costisitoare, deoarece numărul de mintermeni şi maxtermeni posibili (şi deci numărul de porţi) creşte exponenţial cu numărul de variabile. Circuitele combinaţionale se *minimizează* prin reducerea numărului şi a dimensiunilor porţilor necesare.

Metodele tradiţionale de minimizare a circuitelor combinaţionale pe care le vom studia au ca punct de plecare un tabel de adevăr sau – echivalent – o listă de mintermeni sau de maxtermeni. Dacă funcţia logică pe care trebuie să o prelucrăm nu este scrisă în această formă, trebuie să o transpunem într-o formă adecvată înainte de aplicarea unei metode de minimizare. De exemplu, dată fiind o expresie logică oarecare, putem să calculăm valoarea ei pentru fiecare combinaţie de intrare şi apoi să întocmim tabelul de adevăr.

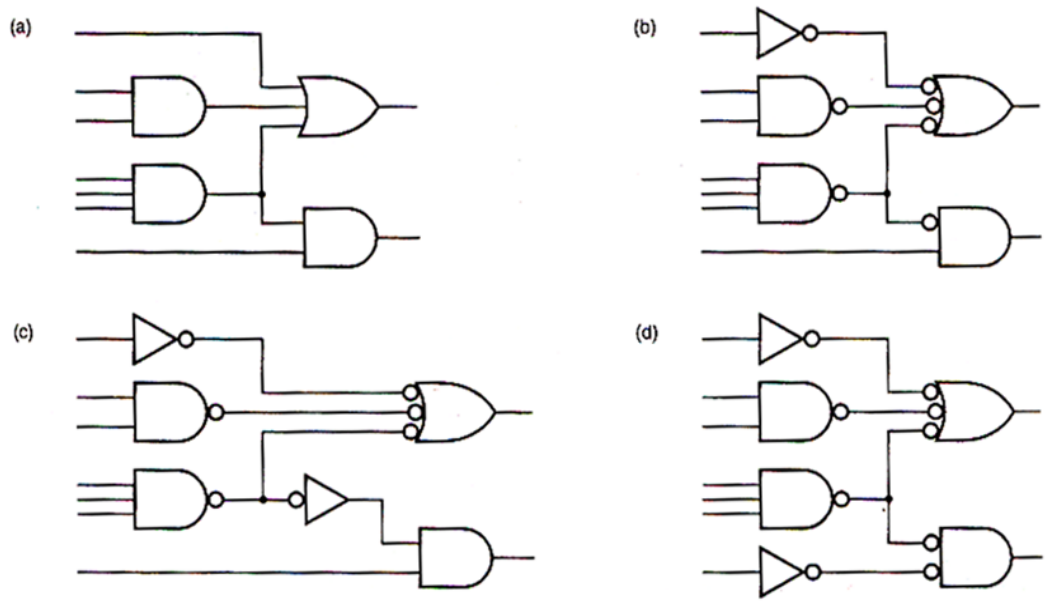


Figura 4-11 Transformări în reprezentare simbolică: (a) circuitul iniţial; (b) variantă cu o poartă nestandardizată; (c) folosirea unui inversor pentru eliminarea porţii nestandardizate; (d) amplasarea preferabilă pentru inversor

Metodele de minimizare reduc costurile circuitelor cu două niveluri **AND-OR**, **OR-AND**, **NAND-NAND** şi **NOR-NOR** în trei moduri:

1. Prin minimizarea numărului de porţi din primul nivel.
2. Prin minimizarea numărului de intrări al fiecărei porţi din primul nivel.
3. Prin minimizarea numărului de intrări al porţii din nivelul al doilea. De fapt, aceasta este o consecinţă a reducerii de la primul nivel.

Metodele de minimizare nu iau însă în calcul costurile inversoarelor de intrare: ele se aplică în ipoteza că există acces la toate variabilele, atât directe, cât şi complementate. Această ipoteză nu este întotdeauna adevărată la proiectarea unui nivel de porţi sau a unui ASIC, însă este adecvată proiectării cu PLD; la PLD există acces la „discreţie” la toate variabilele, atât directe, cât şi complementate.

Majoritatea metodelor de minimizare se bazează pe o generalizare a teoremelor de combinare, T10 şi T10':

$$\text{factor} \cdot Y + \text{factor} \cdot Y' = \text{factor}$$
$$(\text{termen} + Y) \cdot (\text{termen} + Y') = \text{termen}$$

Aceasta înseamnă că dacă doi termeni ai unui produs sau ai unei sume se deosebesc numai prin faptul că o variabilă este complementată în unul şi necomplementată în celălalt, ei pot fi grupaţi într-un singur termen, în care

variabila respectivă nu mai apare. În acest mod se economiseşte o poartă, iar poarta rămasă are cu o intrare mai puţin.

4.3.1 Diagrame Karnaugh

O *diagramă Karnaugh* este o reprezentare grafică a tabelului de adevăr al unei funcţii logice. În fig. 4-12 apar diagramele Karnaugh aferente unor funcţii logice de 2, 3 şi 4 variabile. Diagrama unei funcţii logice cu *n* intrări este un tablou cu 2^n celule, câte una pentru fiecare combinaţie de intrare posibilă sau pentru fiecare mintermen posibil.

Liniile şi coloanele unei diagrame Karnaugh sunt etichetate astfel încât combinaţia de intrare a oricărei celule să poată fi aflată cu uşurinţă din denumirile liniei şi coloanei corespunzătoare acelei celule. Numărul de dimensiuni mai mici înscris în fiecare celulă este numărul mintermenului corespunzător din tabelul de adevăr, considerând că intrările sunt înscrise în tabelul de adevăr în ordine alfabetică, de la stânga la dreapta (de exemplu, X, Y, Z), iar liniile sunt numerotate binar, în ordine crescătoare. De exemplu, celula 13 din diagrama pentru 4 variabile corespunde rândului din tabelul de adevăr pentru care WXYZ = 1101.

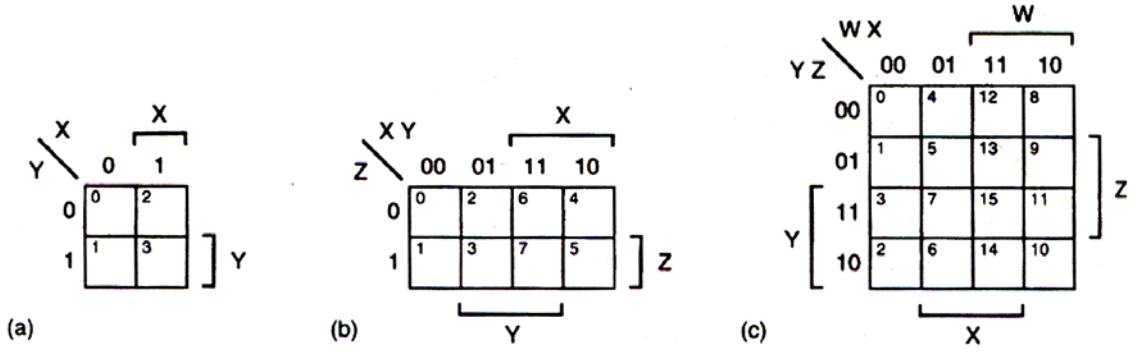


Figura 4-12 Diagrame Karnaugh: (a) pentru două variabile; (b) pentru trei variabile; (c) pentru patru variabile

Când desenăm diagrama Karnaugh a unei funcţii date, fiecare celulă a diagramei conţine data ce se găseşte pe rândul din tabelul de adevăr al funcţiei ce poartă acelaşi număr ca şi celula, şi anume: 0 dacă funcţia are valoarea 0 pentru acea combinaţie de intrare şi 1 în caz contrar.

Utilizăm două denumiri redundante pentru desemnarea liniilor şi a coloanelor din diagrame. De exemplu, să privim diagrama cu 4 variabile din fig. 4-12 (c). Pe coloane apar cele patru combinaţii posibile de W şi X, WX = 00, 01, 11 şi 10. În mod analog, pe linii apar combinaţiile de Y şi Z. Aceste denumiri ne oferă toate informaţiile necesare. Folosim însă şi paranteze pentru a asocia patru regiuni ale diagramei cu cele patru variabile. Fiecare regiune marcată cu o

paranteză este o zonă din diagramă în care variabila respectivă este 1. Evident, parantezele dau aceleaşi informaţii ca şi denumirile liniilor şi coloanelor.

Când desenăm manual o diagramă, este mai uşor să trasăm parantezele decât să scriem toate denumirile liniilor şi coloanelor. În textul lucrării, pe diagramele Karnaugh apar însă şi denumirile, pentru o mai bună înţelegere. În orice caz, trebuie să aveţi grijă să scrieţi denumirile liniilor şi coloanelor în ordinea corectă, pentru a păstra corespondenţa dintre conţinuturile celulelor şi datele din rândurile tabelului de adevăr, ca în fig. 4-12.

Pentru a reprezenta o funcţie logică printr-o diagramă Karnaugh, se copiază, pur şi simplu, cifrele de 1 şi 0 din tabelul de adevăr sau din echivalentele acestuia în celulele corespunzătoare ale diagramei. În figura 4-13 (a) şi (b) apar tabelul de adevăr şi diagrama Karnaugh corespunzătoare funcţiei logice:

$$F = ((X + Y') \cdot Z) + (X' \cdot Y \cdot Z')$$

4.3.2 Minimizarea sumelor de produse

Fiecare combinaţie de intrare pentru care tabelul de adevăr dă valoarea 1 corespunde câte unui mintermen din suma canonică a funcţiei logice. Întrucât perechile de celule adiacente ale diagramei Karnaugh ce conţin valoarea 1 corespund unor mintermeni care se deosebesc printr-o singură variabilă, orice pereche de mintermeni poate fi combinată într-un singur produs cu ajutorul generalizării teoremei T10, factor • Y + factor • Y' = factor. În acest mod, diagrama Karnaugh serveşte la simplificarea sumei canonice a unei funcţii logice.

Pentru exemplificare, să considerăm celulele 5 şi 7 din fig. 4-13 (b) şi contribuţia lor la suma canonică a funcţiei respective:

$$F = ... + X \cdot Y' \cdot Z + X \cdot Y \cdot Z = ... + (X \cdot Z) \cdot Y' + (X \cdot Z) \cdot Y = ... + X \cdot Z$$

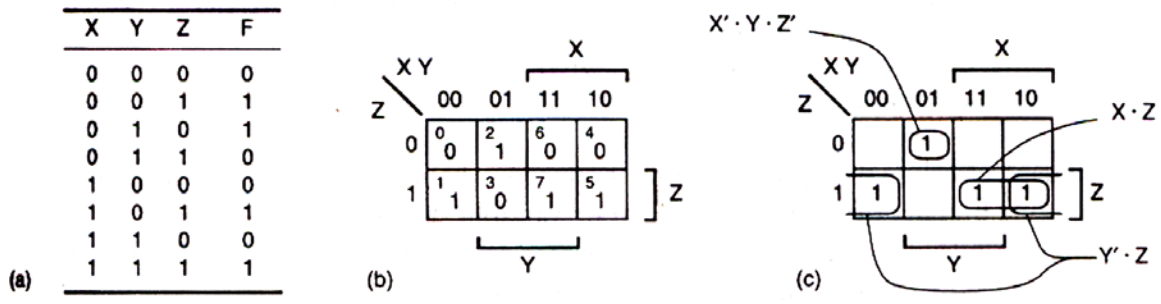


Figura 4-13 $F = \Sigma_{X, Y, Z} (1, 2, 5, 7)$: (a) tabelul de adevăr; (b) diagrama Karnaugh; (c) combinarea celulelor de 1 adiacente

Amintindu-ne de modul în care se ordonează automat cuvintele pe rânduri pentru a completa o suprafaţă dată, observăm că, în fig. 4-13 (b), celulele 1 şi 5 sunt, de asemenea, adiacente şi se pot combina:

$$F = X' \cdot Y' \cdot Z + X \cdot Y' \cdot Z + ... = X' \cdot (Y' \cdot Z) + X \cdot (Y' \cdot Z) + ... = Y' \cdot Z + ...$$

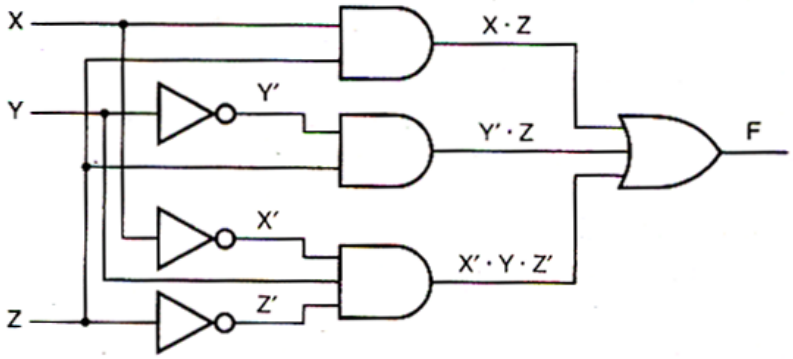


Figura 4-14 Circuit AND-OR minimizat

În general, o funcţie logică poate fi simplificată prin combinarea perechilor de celule adiacente ce conţin 1 (mintermeni) ori de câte ori acest lucru este posibil, scriind apoi o sumă de produse care include toate celulele de 1. În fig. 4-13(c) este prezentat rezultatul corespunzător funcţiei logice din exemplul nostru. O pereche de 1 încercuită indică faptul că mintermenii corespunzători se combină într-un singur termen produs. Circuitul **AND-OR** corespunzător este cel din fig. 4-14.

În cazul multor funcţii logice, procedeul de combinare a celulelor poate fi extins, astfel încât să se combine într-un singur termen produs mai mult de două celule de 1. De exemplu, să considerăm suma canonică a funcţiei logice $F = \Sigma_{X, Y, Z} (0, 1, 4, 5, 6)$. Putem utiliza iterativ prelucrările algebrice din exemplele precedente pentru a combina patru dintre cei cinci mintermeni:

$$F = X' \cdot Y' \cdot Z' + X' \cdot Y' \cdot Z + X \cdot Y' \cdot Z' + X \cdot Y' \cdot Z + X \cdot Y \cdot Z' = [(X' \cdot Y') \cdot Z' + (X' \cdot Y') \cdot Z] + [(X \cdot Y') \cdot Z' + (X \cdot Y') \cdot Z] + X \cdot Y \cdot Z' = X' \cdot Y' + X \cdot Y' + X \cdot Y \cdot Z'$$
$$= [X' \cdot (Y') + X \cdot (Y')] + X \cdot Y \cdot Z'$$
$$= Y' + X \cdot Y \cdot Z'$$

În general, 2^i celule de 1 pot f combinate pentru a forma un termen produs conţinând $n - i$ variabile literale, unde n este numărul de variabile al funcţiei.

Există o regulă matematică prin care se poate afla cu precizie câte celule de 1 pot fi combinate şi ce formă are termenul produs corespunzător:

Celulele unei mulţimi de 2^i celule de 1 pot fi combinate dacă există i variabile ale funcţiei logice care formează toate cele 2^i combinaţii posibile în cadrul acelei mulţimi, iar restul de $n - i$ variabile îşi păstrează aceeaşi valoare în toată mulţimea. Termenul produs corespunzător are $n - i$ variabile literale, o

variabilă fiind complementată dacă apare ca 0 în toate celulele de 1 şi necomplementată dacă apare ca 1.

Grafic, această regulă se traduce prin faptul că putem încercui mulţimi *dreptunghiulare* de 2^i celule de 1, extinzând atât textual, cât şi figurativ, definirea unui dreptunghi astfel încât acesta să cuprindă şi celulele adiacente la margini. Putem determina direct din diagramă variabilele literale incluse în termenii produs corespunzători; pentru fiecare variabilă procedam astfel:

- Dacă pe diagramă se încercuiesc numai zone în care variabila este 0, atunci variabila apare complementată în termenul produs.
- Dacă pe diagramă se încercuiesc numai zone în care variabila este 1, atunci variabila apare necomplementată în termenul produs.
- Dacă pe diagramă se încercuiesc zone în care variabila este atât 0, cât şi 1, atunci variabila nu apare deloc în termenul produs.

O expresie sumă de produse aferentă unei funcţii trebuie să conţină termeni produs (mulţimi încercuite de celule de 1) care includ toate valorile de 1 din diagramă şi nici un 0.

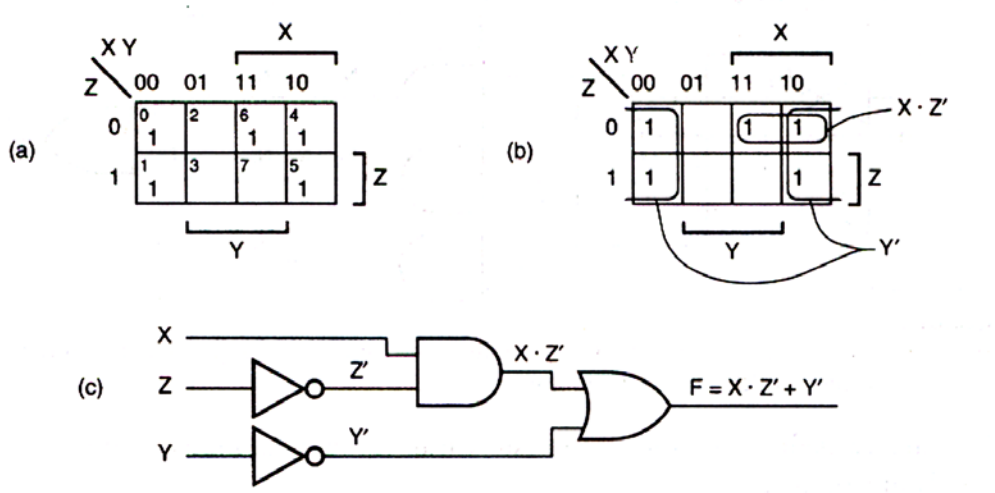


Figura 4-15 $F = \Sigma_{X, Y, Z}(0, 1, 4, 5, 6)$: (a) diagrama Karnaugh iniţială; (b) diagrama Karnaugh cu termenii produs încercuiţi; (c) circuitul AND/OR

Diagrama Karnaugh pentru cel mai recent exemplu, $F = \Sigma_{X, Y, Z}(0, 1, 4, 5, 6)$, apare în fig. 4-15 (a) şi (b). Am încercuit o mulţime de patru celule de 1, corespunzând termenului produs Y', şi o mulţime de două celule de 1, corespunzând termenului produs X • Z'. Se observă că al doilea termen produs are cu o variabilă literală mai puţin decât termenul produs corespunzător rezultatului din prelucrarea algebrică (X • Y • Z'). Prin încercuirea celei mai mari mulţimi posibile de 1 care include celula 6 am găsit o implementare mai puţin costisitoare a funcţiei logice, întrucât o poartă **AND** cu două intrări este mai ieftină decât una cu trei intrări. Faptul că doi termeni produs diferiţi includ o

aceeaşi celulă de 1 (cu numărul 4) nu afectează cu nimic funcţia logică, întrucât, în adunarea logică, 1 + 1 = 1, nu 2! Circuitul corespunzător **AND/OR** cu două niveluri este prezentat în fig. (c).

Câteva definiţii:

- *Suma minimală a* unei funcţii logice $F(X_1, X_2, ..., X_n)$ este o expresie sumă de produse a funcţiei F cu proprietatea că pentru F nu există o altă expresie sumă de produse care să conţină mai puţini termeni produs, iar orice altă expresie sumă de produse cu acelaşi număr de termeni produs conţine cel puţin tot atâtea variabile literale. Cu alte cuvinte, suma minimală are cel mai mic număr posibil de termeni produs (cele mai puţine porţi pe primul nivel şi cele mai puţine intrări de porţi pe al doilea nivel) şi, în aceste condiţii restrictive, cel mai mic număr de variabile literale (cele mai puţine intrări de porţi pe primul nivel).
- O funcţie logică $P(X_1, X_2, ..., X_n)$ *implică* o funcţie logică $F(X_1, X_2, ..., X_n)$ dacă F=1 pentru orice combinaţie de intrare pentru care P=1. Aceasta înseamnă că, dacă P implică F, atunci F este 1 pentru toate combinaţiile de intrare pentru care P = 1, dar posibil şi pentru altele. Notaţia simbolică este P => F. Se mai spune că *"F include P"* sau *"F acoperă P"*.
- *Implicant prim* al unei funcţii logice $F(X_1, X_2, ..., X_n)$ este un termen produs normal $P(X_1, X_2, ..., X_n)$ care implică F astfel încât, orice variabilă ar fi eliminată din P, termenul produs obţinut nu mai implică F. Pe o diagramă Karnaugh, un implicant prim al lui F este o mulţime de celule de 1 încercuite, care respectă regula de combinare; mulţimea aceasta este formată în aşa fel încât, dacă am încerca să o extindem (pentru a cuprinde un număr dublu de celule), ar include unul sau mai multe zerouri.

4.3.3 Simplificarea produselor de sume

Folosind principiul dualităţii, putem minimiza expresiile produs de sume căutând zerourile pe diagrama Karnaugh. Fiecare 0 din diagramă corespunde unui maxtermen din produsul canonic al funcţiei logice. Întregul procedeu din secţiunea precedentă poate fi reformulat dual, inclusiv regulile de scriere a termenilor sumă corespunzători mulţimilor de zerouri încercuite, pentru a găsi un *produs minimal*.

O dată ce am aflat sumele minimale, este mai uşor să găsim produsul minimal aferent unei funcţii F date. Într-o primă etapă se complementează F, pentru a obţine F'. Complementarea se face cu uşurinţă dacă F este exprimată ca listă de mintermeni sau ca tabel de adevăr; valorile de 1 din F' sunt tocmai

zerourile din F. Apoi aflăm suma minimală pentru F' şi complementăm rezultatul cu ajutorul teoremei generalizate a lui DeMorgan, obţinând un produs minimal pentru (F')' = F.

În general, pentru a obţine cea mai ieftină implementare cu două niveluri a unei funcţii logice trebuie să aflăm atât suma minimală, cât şi produsul minimal şi să le comparăm. Dacă o sumă minimală a unei funcţii logice are mulţi termeni, este posibil ca un produs minimal al aceleiaşi funcţii să conţină mai puţini termeni. Compromisul ce trebuie realizat între aceste expresii poate fi exemplificat cu uşurinţă considerând funcţia **OR** cu 4 intrări:

$$F = (W) + (X) + (Y) + (Z) \text{ (o sumă de patru termeni produs banali)}$$
$$= (W + X + Y + Z) \text{ (un produs format dintr-un termen sumă)}$$

Şi situaţia inversă poate fi întâlnită uneori, ca în cazul unei banale funcţii **AND** cu 4 intrări:

$$F = (W) \bullet (X) \bullet (Y) \bullet (Z) \text{ (un produs de patru termeni sumă obişnuiţi)}$$
$$= (W \bullet X \bullet Y \bullet Z) \text{ (o sumă formată dintr-un termen produs)}$$

Pentru unele funcţii logice, ambele forme minimale se implementează cu aceleaşi costuri. De exemplu, să considerăm o funcţie, „**OR** exclusiv" cu trei intrări; ambele expresii minimale au câte patru termeni şi fiecare termen conţine câte trei variabile literale:

$$F = \Sigma_{X, Y, Z}(1, 2, 4, 7) = (X' \bullet Y' \bullet Z) + (X' \bullet Y \bullet Z') + (X \bullet Y' \bullet Z') + (X \bullet Y \bullet Z)$$
$$= (X + Y + Z) \bullet (X + Y' + Z') \bullet (X' + Y + Z') \bullet (X' + Y' + Z)$$

Cu toate acestea, în majoritatea cazurilor, una dintre forme este mai avantajoasă decât cealaltă. Compararea ambelor forme se dovedeşte utilă în special în proiectarea cu PLD.

4.3.4 Combinaţii de intrare "indiferente"

Uneori, pentru unele circuite combinaţionale se specifică faptul că ieşirea acestora nu prezintă importanţă în cazul anumitor combinaţii de intrare, denumite *indiferente.* Asemenea situaţii pot exista, pentru că este posibil ca valorile de ieşire să nu conteze, într-adevăr, când la intrare apar acele combinaţii sau ca respectivele combinaţii de intrare să nu apară niciodată în timpul funcţionării normale. De exemplu, să presupunem că dorim să construim un detector de numere prime a cărui intrare de 4 biţi, $N = N_3N_2N_1N_0$, este întotdeauna o cifră BCD; în acest caz, mintermenii 10 ... 15 nu ar trebui să apară niciodată. Prin urmare, funcţia caracteristică unui detector de cifre BCD prime ar putea fi scrisă astfel:

$$F = \Sigma N_{3,} N_{2,} N_{1,} N_o (1, 2, 3, 5, 7) + d(10, 11, 12, 13, 14, 15)$$

Lista d(...) precizează combinaţiile de intrare indiferente ale funcţiei, fiind denumită uneori *mulţimea d.* În cazul de faţă, F trebuie să ia valoarea 1 pentru combinaţiile de intrare din mulţimea activă (1, 2, 3, 5, 7), poate lua orice valoare pentru intrările din mulţimea d(10, 11, 12, 13, 14, 15) şi trebuie să fie 0 pentru toate celelalte combinaţii de intrare (din mulţimea corespunzătoare zerourilor).

Figura 4-16 arată cum se obţine implementarea unei sume de produse minimale, aferentă detectorului de cifre BCD prime, incluzând şi combinaţiile indiferente. Literele d din diagramă reprezintă combinaţiile de intrare indiferente. Vom folosi un alt mod de încercuire a mulţimilor de 1 (implicanţii primi), astfel:

- Este permisă includerea literelor d în grupurile de 1 încercuite, pentru ca aceste grupuri să fie cât mai extinse cu putinţă. Astfel se reduce numărul de variabile din implicanţii primi corespunzători. În exemplu apar doi asemenea implicanţi ($N_2 \bullet N_0$ şi N2' • N1).
- Nu se încercuiesc grupuri formate exclusiv din d. Adăugarea în expresia funcţiei a termenului produs corespunzător unui asemenea grup ar creşte inutil costurile. În exemplu sunt încercuiţi doi astfel de termeni produs ($N_3 \bullet N_2$ şi N3 • N_0).
- Reamintim: Ca de obicei, nu încercuiţi nici un 0.

De aici înainte, procedeul este acelaşi. Atragem atenţia că trebuie să căutam celulele distincte de 1, nu de d, şi luăm în consideraţie numai implicanţii primi esenţiali corespunzători, precum şi orice alţi implicanţi primi mai sunt necesari pentru a acoperi toate celulele de 1 din diagramă. În fig. 4-16, cei doi implicanţi primi esenţiali acoperă în totalitate celulele de 1 din diagramă. Întâmplător sunt acoperite şi două celule de d, deci F va fi 1 pentru combinaţiile de intrare indiferente 10 şi 11 şi 0 pentru celelalte combinaţii de intrare indiferente.

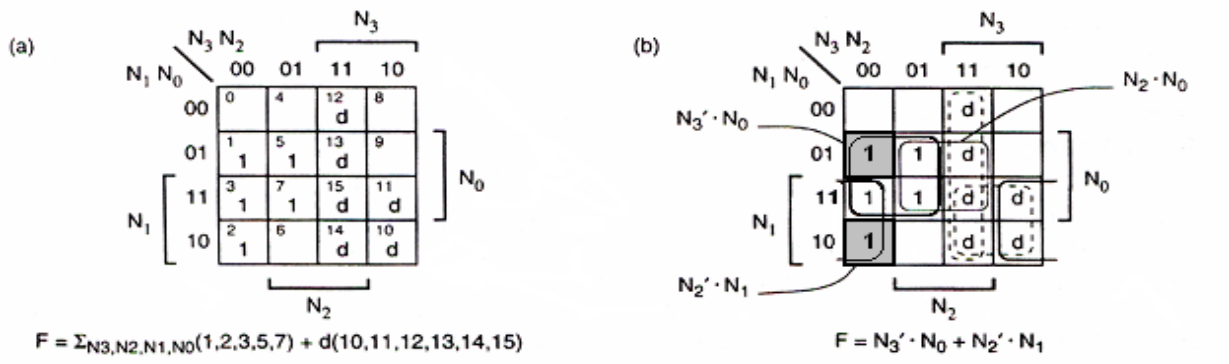


Figura 4-16 Detector de cifre BCD prime: (a) diagrama Karnaugh iniţială; (b) diagrama Karnaugh cu implicanţii primi şi celulele de 1 distincte

4.3.5 Minimizarea circuitelor cu mai multe ieşiri

Majoritatea circuitelor combinaţionale utilizate în practică necesită mai mult decât o singură ieşire. Întotdeauna putem lucra cu un circuit cu *n* ieşiri considerându-l, din punctul de vedere al proiectării, ca *n* circuite independente cu câte o singură ieşire. Însă, în acest caz, putem pierde din vedere câteva posibilităţi de optimizare. Să considerăm, pentru exemplificare, următoarele două funcţii logice:

$$F = \Sigma_{X, Y, Z}(3, 6, 7)$$
$$G = \Sigma_{X, Y, Z}(0, 1, 3)$$

În fig. 4-17, F şi G sunt prelucrate ca două funcţii independente, cu câte o singură ieşire. Însă, aşa cum se vede în fig. 4-18, mai putem găsi o pereche de expresii sumă de produse care conţin un acelaşi termen produs, astfel încât circuitul rezultat are o poartă mai puţin decât varianta iniţială.

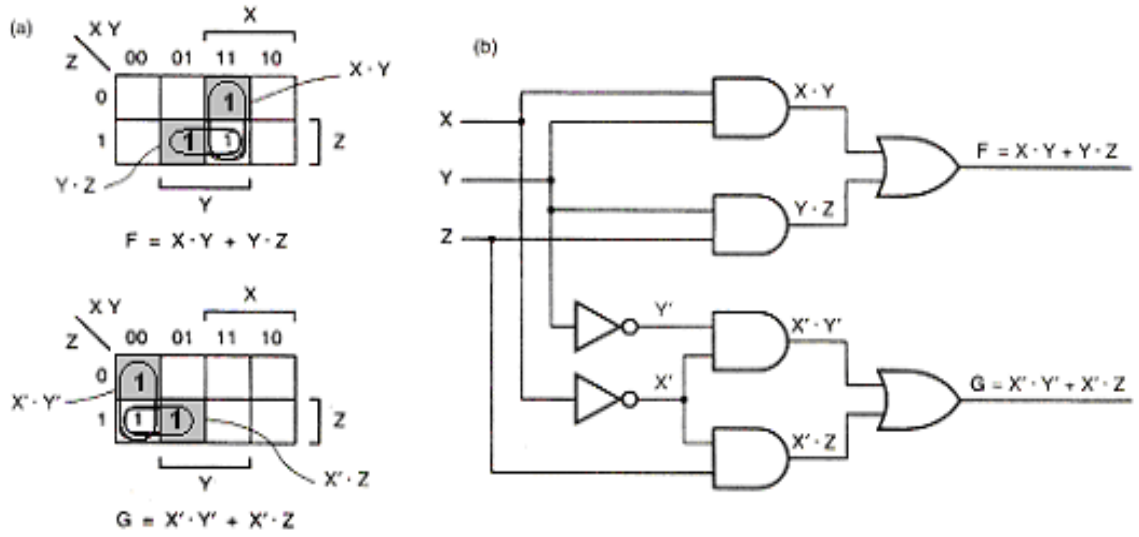


Figura 4-17 Tratarea unui circuit cu două ieşiri ca două circuite independente cu câte o singură ieşire: (a) diagramele Karnaugh; (b) circuitul "minimal"

Dacă proiectăm circuite combinaţionale cu mai multe ieşiri folosind porţi discrete, ca într-un ASIC, este evident că găsirea unor termeni produs comuni duce la scăderea dimensiunilor circuitului şi a costurilor. PLD conţin, în plus, mai multe structuri identice sumă de produse, câte una pentru fiecare ieşire, iar unele PLD permit utilizarea în comun, de către mai multe ieşiri, a unor termeni produs comuni. Ca urmare, principiile prezentate în subsecţiunea de faţă au fost preluate în multe programe de minimizare a circuitelor logice.

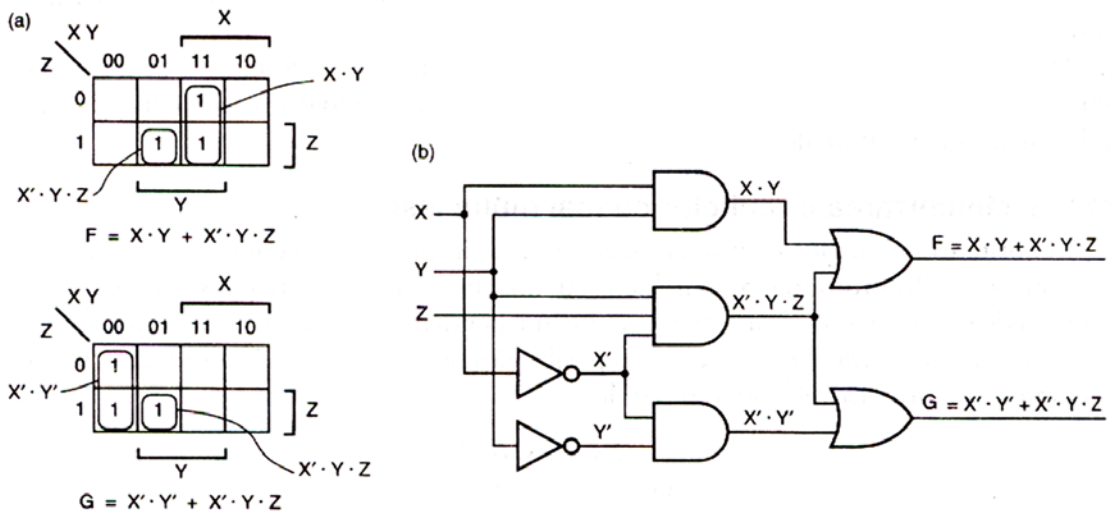


Figura 4-18 Minimizarea unui circuit cu mai multe ieşiri, în cazul particular a două ieşiri: (a) diagramele minimizate, care prezintă un acelaşi termen; (b) circuitul minimal cu mai multe ieşiri

Circuitele de dimensiuni mai mari pot fi minimizate numai cu ajutorul unui algoritm - bine determinat - de minimizare a circuitelor cu mai multe ieşiri. Aici vom face o trecere în revistă succintă a principiilor acestui algoritm.

Pentru a efectua cu succes o minimizare de circuit cu mai multe ieşiri, pornind de la un set de *n* funcţii, trebuie să ţinem seama nu doar de cele *n* funcţii iniţiale cu câte o ieşire, ci şi de "funcţiile produs". *O funcţie produs de m* dintr-o mulţime de *n* funcţii este produsul a m dintre acele funcţii, unde $2 \leq m \leq n$.. Există $2^n - n - 1$ asemenea funcţii. Din fericire, în exemplul nostru, $n = 2$, trebuind luată în consideraţie o singură funcţie produs, F • G. Diagramele Karnaugh pentru F, G şi F • G sunt prezentate în fig. 4-19; în general, diagrama unei funcţii produs de *m* se obţine prin aplicarea funcţiei **AND** diagramelor ce conţin cele *m* componente ale funcţiei produs.

Un *implicant prim pentru mai multe ieşiri* corespunzător unui set de *n* funcţii este unul dintre implicanţii primi ai uneia dintre cele *n* funcţii sau dintre funcţiile produs. Prima etapă în minimizarea circuitelor cu mai multe ieşiri este găsirea tuturor implicanţilor primi pentru mai multe ieşiri. Fiecare implicant prim al unei funcţii produs de *m* poate constitui un termen inclus în cele *m* ieşiri ale circuitului. Dacă dorim să minimizăm un set de 8 funcţii, trebuie să găsim implicanţii primi corespunzători atât celor $2^8 - 8 - 1$ funcţii produs, cât şi celor 8 funcţii date.

După găsirea implicanţilor primi aferenţi circuitului cu mai multe ieşiri, vom încerca să simplificăm problema prin identificarea celor esenţiali. O *celulă de 1 distinctă* a unei funcţii F cu o singură ieşire este o celulă de 1 acoperită de exact un singur implicant prim al lui F sau al funcţiilor produs formate cu F. În fig. 4-19, celulele de 1 distincte sunt haşurate. *Implicant prim esenţial* al unei funcţii cu o singură ieşire este acel implicant prim care conţine o celulă de 1

distinctă. Ca şi în cazul minimizării circuitelor cu o singură ieşire, implicanţii primi esenţiali trebuie să facă parte din soluţia cea mai puţin costisitoare. În continuarea algoritmului se iau în consideraţie numai acele celule de 1 care nu au fost acoperite de implicanţii primi esenţiali.

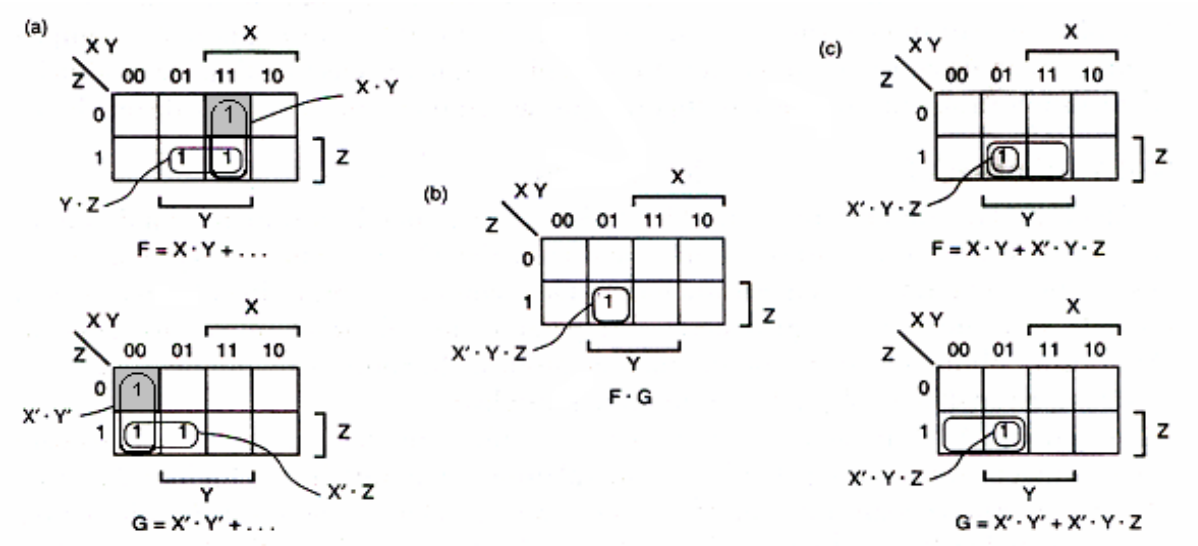


Figura 4-19 Diagramele Karnaugh aferente unui set de două funcţii: (a) diagramele pentru F şi G; (b) diagrama produsului de 2 pentru F • G; (c) diagramele reduse pentru F şi G, după îndepărtarea implicanţilor primi esenţiali şi a celulelor de 1 acoperite de aceştia

Ultima etapă este alegerea unui set minimal de implicanţi primi care să acopere celulele de 1 rămase. Acum trebuie să examinăm cele n funcţii simultan, luând în consideraţie posibilitatea ca ele sa conţină termeni comuni. În exemplul din fig. 4-19 (c), observăm că există un singur termen produs comun ce acoperă celula de 1 rămasă atât în F, cât şi în G.

5. Metode de proiectare a circuitelor logice combinaționale

Un circuit combinaţional real poate avea zeci de intrări şi ieşiri, iar pentru descrierea lui ar putea fi necesare sute, mii sau chiar milioane de termeni produs ai unei sume, precum şi tabele de adevăr conţinând miliarde de rânduri. Din această cauză, majoritatea problemelor de proiectare a circuitelor logice combinaţionale reale sunt de dimensiuni mult prea mari pentru a putea fi rezolvate prin aplicarea „în forţă" a metodelor teoretice.

Secretul este concepţia structurală. Un circuit complex sau un sistem este considerat ca un ansamblu de subsisteme de dimensiuni mai mici, fiecare dintre acestea putând fi descris mai simplu.

În proiectarea circuitelor logice combinaţionale se lucrează cu câteva structuri de bază – decodoare, multiplexoare, comparatoare şi altele asemenea – care apar în mod regulat ca blocuri structurale ale sistemelor de dimensiuni mari. Cele mai importante dintre aceste structuri vor fi prezentate în capitolul de faţă.

Înainte de a face cunoştinţă cu aceste blocuri structurale trebuie să discutăm un alt aspect important, care se referă la standardele de documentaţie folosite de proiectanţii de circuite digitale pentru a se asigura că rezultatele muncii lor sunt corecte şi posibil de fabricat şi de întreţinut.

5.1 Standarde pentru documentaţie

O documentaţie de calitate este esenţială pentru proiectarea corectă şi întreţinerea eficientă a sistemelor digitale. Pe lângă faptul că trebuie să fie precisă şi completă, documentaţia trebuie să furnizeze şi unele informaţii practice, astfel încât inginerul ce testează sistemul, tehnicianul care se ocupă de întreţinere şi însuşi proiectantul să îşi poată da seama, după simplă citire a documentaţiei, cum funcţionează acel sistem.

Cu toate că stilul unei documentaţii depinde de complexitatea sistemului şi de mediile de proiectare şi de fabricaţie unde a fost elaborată, un pachet de documentaţie ar trebui să cuprindă cel puţin următoarele şase elemente:

1. *O descriere tehnică,* prin care să se arate cu precizie cum trebuie să funcţioneze circuitul sau sistemul respectiv, inclusiv o prezentare a tuturor semnalelor de intrare şi de ieşire („interfeţele") şi a funcţiilor ce trebuie realizate. Reţineţi că în această descriere nu trebuie să se arate şi în ce mod sistemul îşi îndeplineşte funcţiile, ci numai parametrii care trebuie să se

obţină. Multe companii obişnuiesc să ataşeze descrieri tehnice şi unul sau mai multe documente în care să arate cum funcţionează sistemul.

2. *O schemă bloc,* adică un desen sugestiv care prezintă principalele module funcţionale ale sistemului şi principalele interconexiuni.

3. *O schemă electrică* incluzând, într-o formă standardizată, valorile componentelor electrice ale sistemului, interconexiunile lor şi toate detaliile necesare pentru construirea acelui sistem, printre care tipurile de CI, simbolurile şi notaţiile folosite şi numerotarea pinilor. Denumirea de *schemă logică,* folosită până acum, desemnează un desen sugestiv care nu conţine informaţii atât de detaliate ca *schema electrică*.

4. *O diagramă temporală,* care indică valorile diferitelor semnale logice ca funcţii de timp, inclusiv decalajele cauză-efect între principalele semnale.

5. *O descriere a dispozitivelor logice structurate,* care prezintă funcţionarea internă a dispozitivelor logice programabile (PLD), a matricelor de porţi programabile în câmp (FPGA) sau a circuitelor integrate specifice unei aplicaţii (ASIC). Aceasta este scrisă, în mod normal, într-un limbaj descriptor de echipamente (HDL), dar poate avea şi forma unor ecuaţii logice, tabele de stări sau diagrame de stări. În unele cazuri, pentru modelarea funcţionării unui circuit sau pentru descrierea funcţionării lui poate fi folosit şi un limbaj de programare convenţional, cum este C.

6. *O descriere a circuitului,* sub forma unui document text narativ care, alături de celelalte documente, explică funcţionarea internă a circuitului. Aici trebuie menţionate toate condiţiile şi defectele de concepţie şi de funcţionare posibile şi trebuie atrasă atenţia asupra utilizării unor „artificii" de proiectare neevidente. O descriere bine realizată a circuitului conţine definiţiile acronimelor şi ale altor termeni de specialitate, precum şi trimiteri la documente cu aceeaşi tematică.

Ultimul domeniu din documentaţie, descrierea circuitului, este de mare importanţă în practică. Aşa cum un programator cu experienţă redactează un document de concepţie a programului înainte de a începe să scrie codul, un proiectant de circuite logice cu experienţă va începe să elaboreze un document ce descrie circuitul înainte de a desena schema. Din nefericire, uneori, descrierea circuitului este ultimul document ce se elaborează, iar alteori nu se întocmeşte deloc. În absenţa descrierii, circuitele sunt dificil de depanat, de testat, de întreţinut, de fabricat şi de îmbunătăţit.

5.1.1 Schema bloc

Într-o *schemă bloc* se arată semnalele de intrare şi de ieşire, modulele funcţionale, căile interne de date şi semnalele de comandă importante dintr-un sistem. În general, schema bloc nu trebuie să fie atât de detaliată încât să ocupe mai mult de o pagină, însă trebuie să fie suficient de clară. O schemă bloc de dimensiuni reduse poate cuprinde trei până la şase blocuri, pe când una de dimensiuni mari poate avea de la 10 la 15 blocuri, în funcţie de complexitatea sistemului. În oricare caz, ea trebuie să prezinte cele mai importante elemente ale sistemului şi modul în care funcţionează ansamblul acestora. Pentru sisteme foarte mari este posibil să fie necesară anexarea unor scheme bloc ale unor subsisteme separate, însă întotdeauna trebuie să existe o schemă „de ansamblu" în care apare întregul sistem.

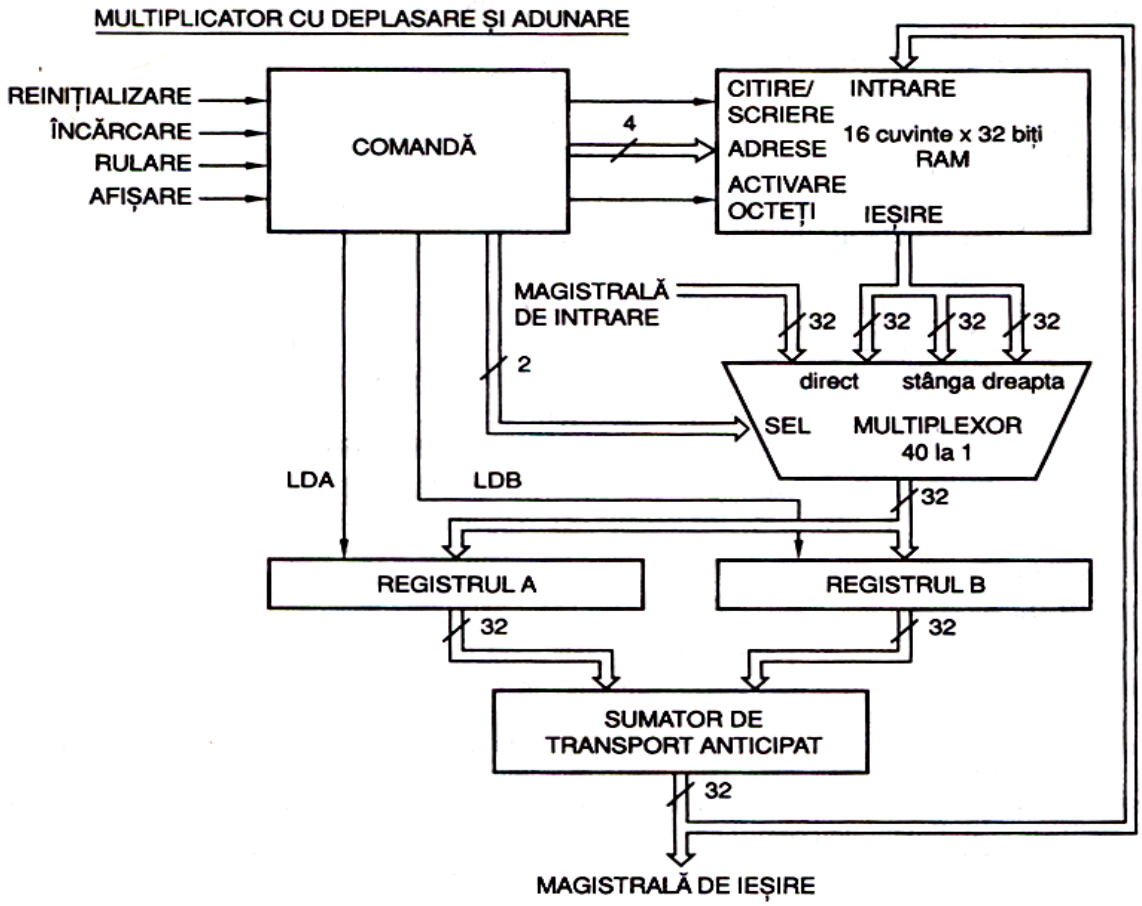


Figura 5-1 Exemplu de schemă bloc ce constituie tema unui proiect de circuite digitale

În figura 5-1 este prezentat un fragment de schemă bloc. Pe fiecare bloc apare o etichetă cu numele funcţiei realizate de blocul respectiv, şi nu cu tipul de cipuri pe care este realizat blocul.

O *magistrală* este un ansamblu de două sau mai multe linii de semnal asociate. În schemele bloc, magistralele se reprezintă cu linie dublă sau

îngroşată. O bară oblică („/”) şi un număr pot arăta câte linii de semnal separate conţine o magistrală.

Fluxurile de comenzi şi de date dintr-o schemă bloc trebuie indicate clar. În schemele logice, semnalele se reprezintă, în general, ca având sensul de la stânga spre dreapta, însă acest lucru este mai greu de realizat într-o schemă bloc. Semnalele de intrare şi de ieşire pot fi reprezentate pe oricare dintre laturile unui bloc, iar sensul de circulaţie al semnalelor se poate alege arbitrar. Pentru eliminarea ambiguităţilor, pe magistrale şi pe liniile de semnal obişnuite se desenează săgeţi.

5.1.2 Simboluri de porţi

Formele simbolurilor pentru porţile **AND** şi **OR** şi pentru circuitele tampon sunt cele din figura 5-2(a). Pentru a reprezenta porţi logice cu un număr foarte mare de intrări, simbolurile porţilor **AND** şi **OR** suferă o expandare, ca în figura 5-2(b). Un mic cerc, numit *cerculeţ inversor,* semnifică inversarea logică sau complementarea şi se utilizează în simbolurile porţilor **NAND** şi **NOR** şi ale inversoarelor, ca în figura 5-2(c).

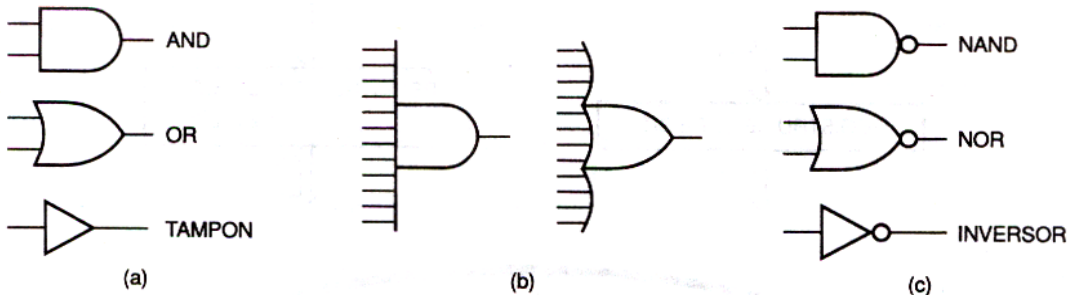


Figura 5-2 Simbolurile porţilor logice de bază: (a) AND, OR, TAMPON; (b) porţi cu intrări expandate; (c) porţi cu cerculeţe inversoare (NAND, NOR, INVERSOR)

Aplicând teorema lui DeMorgan generalizată, putem aduce expresiile logice într-o formă adecvată porţilor cu ieşiri complementate. De exemplu, dacă X şi Y sunt semnalele de la intrările unei porţi **NAND** al cărei semnal de ieşire este Z, putem scrie:

$$Z = (X \bullet Y)' = X' + Y'$$

De aici rezultă două simboluri diferite, dar la fel de corecte, pentru o poartă **NAND**. De fapt, asemenea prelucrări pot fi aplicate la fel de bine şi porţilor cu semnale de ieşire necomplementate. Să considerăm, ca exemplu, următoarele ecuaţii, în cazul unei porţi **AND**:

$$Z = X \bullet Y = ((X \cdot Y)')' = (X' + Y')'$$

Ca urmare, o poartă **AND** poate fi reprezentată simbolic ca o poartă **OR** cu cerculeţe inversoare la intrări şi la ieşire.

Simbolurile echivalente pentru porţile standard, care se pot obţine prin astfel de transformări sunt prezentate sistematizat în figura 5-3. Deşi ambele simboluri dintr-o pereche reprezintă aceeaşi funcţie logică, folosirea unuia sau a altuia dintre ele într-o schemă logică nu este arbitrară, cel puţin în cazul în care dorim să ne conformăm unor standarde de calitate pentru documentaţie. Aşa cum vom arăta, prin alegerea judicioasă a denumirilor semnalelor şi a simbolurilor pentru porţi, schemele logice devin mult mai uşor de utilizat şi de înţeles.

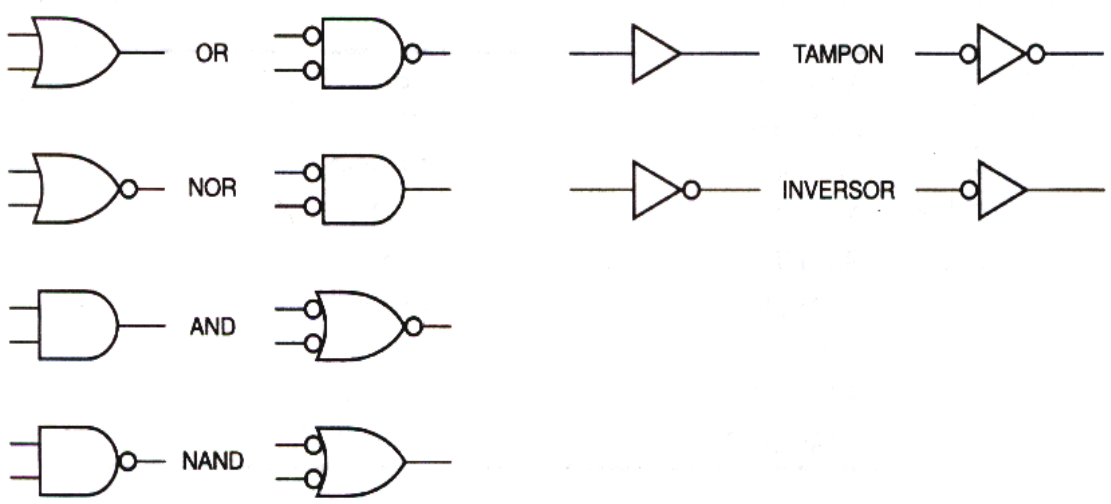


Figura 5-3 Simbolurile echivalente pentru porţi, obţinute prin aplicarea teoremei lui DeMorgan

5.1.3 Denumiri de semnale şi niveluri active

Fiecare semnal de intrare şi de ieşire dintr-un circuit logic trebuie să poarte o etichetă alfa-numerică descriptivă – denumirea semnalului. Nişte denumiri de semnale bine alese furnizează informaţii celor ce consultă schema.

Fiecare denumire de semnal trebuie să aibă un *nivel activ* asociat. Un semnal este *activ la nivelul HIGH* dacă efectuează acţiunea numită sau semnalizează condiţia numită când are valoarea **HIGH** sau 1. Un semnal este *activ la nivelul LOW* dacă efectuează acţiunea numită sau semnalizează condiţia numită când are valoarea **LOW** sau 0. Se spune că un semnal este *confirmat* când se află la nivelul său activ. Când nu se află la nivelul activ, semnalul se numeşte *negat.* Nivelul activ al fiecărui semnal dintr-un circuit este, în mod normal, precizat în denumirea semnalului, conform unei convenţii. Întrucât desemnarea nivelului activ face parte din denumirea semnalului, convenţia de denumire trebuie să fie compatibilă cu parametrii de intrare impuşi de instrumentul de proiectare asistată de calculator care urmează să lucreze cu

denumirile semnalelor, ca, de exemplu, editoarele de scheme, compilatoarele HDL şi simulatoarele.

Este foarte important să înţelegeţi care este diferenţa dintre denumirile de semnale, expresii şi ecuaţii. *O denumire de semnal* este doar un nume, o etichetă alfa-numerică. O *expresie logică* operează cu denumiri de semnale folosind operatorii din algebra de comutaţie - **AND**, **OR** şi **NOT**. *O ecuaţie logică* este atribuirea unei expresii logice unei denumiri de semnal - ea descrie funcţiile realizate de un semnal în funcţie de alte semnale.

5.1.4 Proiectarea cu cerculeţe

Proiectarea cu cerculeţe este o modalitate de a alege simbolurile logice şi denumirile semnalelor în aşa fel încât funcţia unui circuit logic să devină mai uşor de înţeles. De obicei, aceasta se face prin alegerea denumirilor semnalelor şi a tipurilor de porţi astfel încât majoritatea cerculeţelor inversoare să se anuleze reciproc, iar schema logică să poată fi analizată ca şi cum toate semnalele ar fi active în High.

De exemplu, să presupunem că trebuie să obţinem un semnal care determină un dispozitiv să intre în „**FUNCŢIONARE**” când suntem „**GATA**” şi când primim o „**CERERE**”. Din enunţul problemei reiese clar că este necesară o funcţie **AND** (în algebra de comutaţie vom scrie **FUNCŢIONARE** = **GATA·CERERE**). Dar pentru realizarea funcţiei **AND** putem folosi diverse porţi, în funcţie de nivelul activ necesar pentru semnalul **FUNCŢIONARE** şi de nivelurile active ale semnalelor de intrare disponibile.

În figura 5-4(a) este prezentat cazul cel mai simplu, în care **FUNCTIONARE** trebuie să fie activ în **HIGH**, iar semnalele de intrare disponibile sunt, de asemenea, active în **HIGH** - folosim o poartă **AND**. Dacă însă dispozitivul comandat necesită un semnal activ în **LOW**, **FUNCTIONARE_L**, putem folosi o poartă **NAND**, ca în 5-4(b). Dacă semnalele de intrare disponibile sunt active în **LOW**, putem folosi o poartă **NOR** sau una **OR**, ca în 5-4(c) şi 5-4(d).

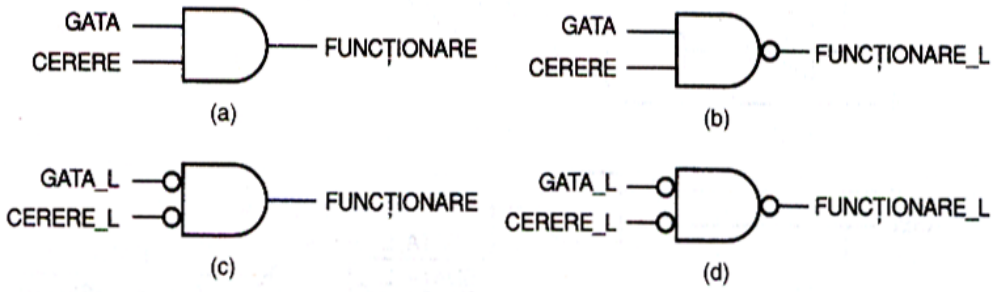


Figura 5-4 Diverse căi de a intra în FUNCŢIONARE: (a) intrări şi ieşire active în HIGH; (b)intrări active în HIGH şi ieşire activă în LOW; (c) intrări active în LOW şi ieşire activă în HIGH; (d) intrări şi ieşire active în LOW

Nivelurile active ale semnalelor disponibile nu coincid întotdeauna cu nivelurile active ale porţilor disponibile. De exemplu, să presupunem că se dau semnalele de intrare **GATA_L** (activ în **LOW**) şi **CERERE** (activ în **HIGH**). Figura 5-5 prezintă două modalităţi de a obţine semnalul **FUNCŢIONARE**, folosind un inversor pentru a genera nivelul activ necesar realizării funcţiei **AND**. În general se preferă a doua modalitate, deoarece porţile inversoare, cum este **NOR**, sunt mai rapide decât cele neinversoare, ca **AND**. Am reprezentat inversorul altfel în fiecare caz pentru ca nivelul activ al semnalului de ieşire să corespundă denumirii semnalului.

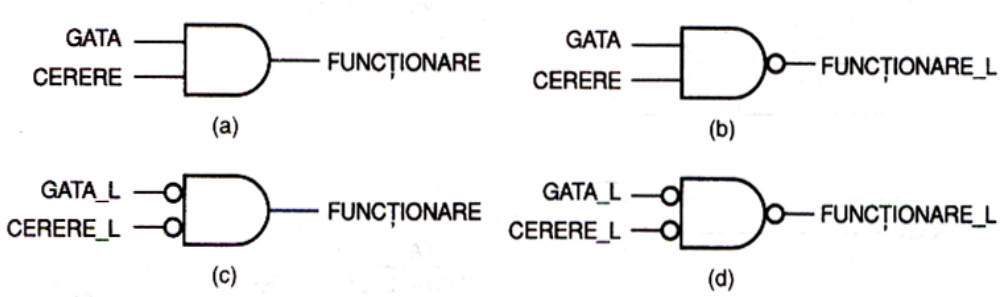


Figura 5-5 Alte două moduri de a intra în FUNCŢIONARE, pentru niveluri diferite ale semnalelor de intrare: (a) cu o poartă AND; (b) cu o poartă NOR

Iată câteva reguli utile pentru proiectarea schemelor logice cu cerculeţe:

- Numele semnalului de ieşire al unui dispozitiv trebuie să aibă acelaşi nivel activ ca şi pinul de ieşire al dispozitivului, adică să fie activ în **LOW** dacă simbolul dispozitivului are cerculeţ inversor la pinul de ieşire şi activ în **HIGH** dacă nu apare cerculeţul.
- Dacă nivelul activ al unui semnal de intrare coincide cu cel al pinului de intrare la care se aplică, atunci funcţia logică realizată în interiorul conturului simbolului este activată când semnalul este confirmat. Acesta este cazul cel mai frecvent întâlnit în schemele logice.
- Dacă nivelul activ al unui semnal de intrare este opus celui al pinului de intrare la care se aplică, atunci funcţia logică realizată în interiorul conturului simbolului este activată când semnalul este negat. Acest caz trebuie evitat ori de câte ori este posibil, deoarece, pentru a înţelege cum funcţionează circuitul, suntem obligaţi să reţinem mental o negaţie logică.

5.1.5 Organizarea desenului

Schemele logice se desenează, în mod normal, cu intrările porţilor orientate spre stânga şi cu ieşirile spre dreapta. Simbolurile elementelor logice mai complexe se desenează, în orientare normală, tot cu intrările spre stânga şi ieşirile spre dreapta.

O schemă completă ce ocupă o întreagă planşă se desenează cu intrările în sistem în stânga şi cu ieşirile spre dreapta, sensul general de circulaţie al semnalelor fiind de la stânga la dreapta. Dacă în mijlocul planşei apare o intrare sau o ieşire, aceasta trebuie prelungită până în marginea din stânga, respectiv din dreapta, a planşei. În acest fel, persoana care consultă schema poate găsi toate intrările şi ieşirile urmărind doar marginile schemei. Toate conexiunile căilor de semnal de pe planşă trebuie figurate, în măsura în care este posibil; dacă schema este prea aglomerată, se poate întrerupe figurarea unora dintre căile de semnal, însă întreruperea trebuie semnalată la ambele capete.

Uneori, schemele bloc se desenează fără ca liniile să se intersecteze, pentru un aspect mai „aerisit", însă în schemele logice nu se procedează aşa niciodată. Aici este permisă intersectarea liniilor, iar conexiunile sunt marcate clar prin puncte. Totuşi, unele sisteme de proiectare asistată de calculator şi unii proiectanţi desenează nişte puncte de conexiune greu de zărit. Pentru a se putea face deosebirea dintre liniile intersectate şi cele interconectate, la aceste sisteme s-a adoptat convenţia de a fi acceptate numai conexiunile în „T", ca în figura 5-6.

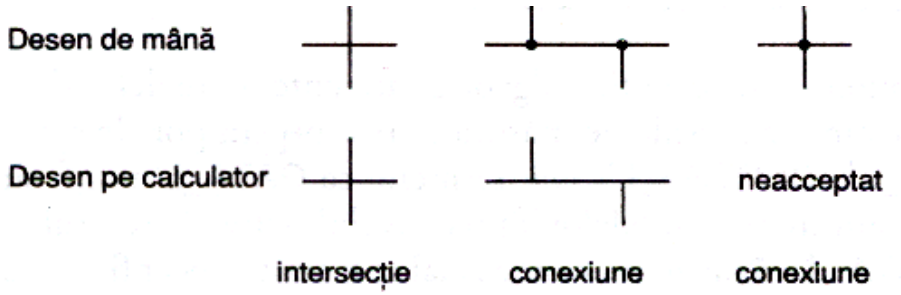


Figura 5-6 Intersecţii de linii şi conexiuni

Cel mai uşor se lucrează cu schemele care încap pe o singură planşă. Schemele care nu încap pe o singură coală trebuie împărţite pe mai multe coli separate astfel încât numărul de conexiuni la trecerea de pe o planşă pe alta să fie minim.

O schemă ce se întinde pe mai multe planşe are, de obicei, o structură „orizontală", aşa cum observaţi în figura 5-7.

Un alt mod de a aşeza planşele este într-o structură ierarhică, care este prezentată în figura 5-8. Într-o astfel de manieră, schema „de la nivelul cel mai de sus" ocupă o singură planşă, putând îndeplini rolul de schemă bloc. De obicei, ea nu conţine porţi sau alte elemente logice; aici apar doar blocurile corespunzătoare principalelor subsisteme, cu interconexiunile respective. Blocurile sau subsistemele sunt definite, la rândul lor, în planşele de la nivelurile inferioare, care pot conţine fie detaliile obişnuite la nivel de porţi, fie alte blocuri, a căror definiţie se află în planşe de nivel ierarhic inferior acestora.

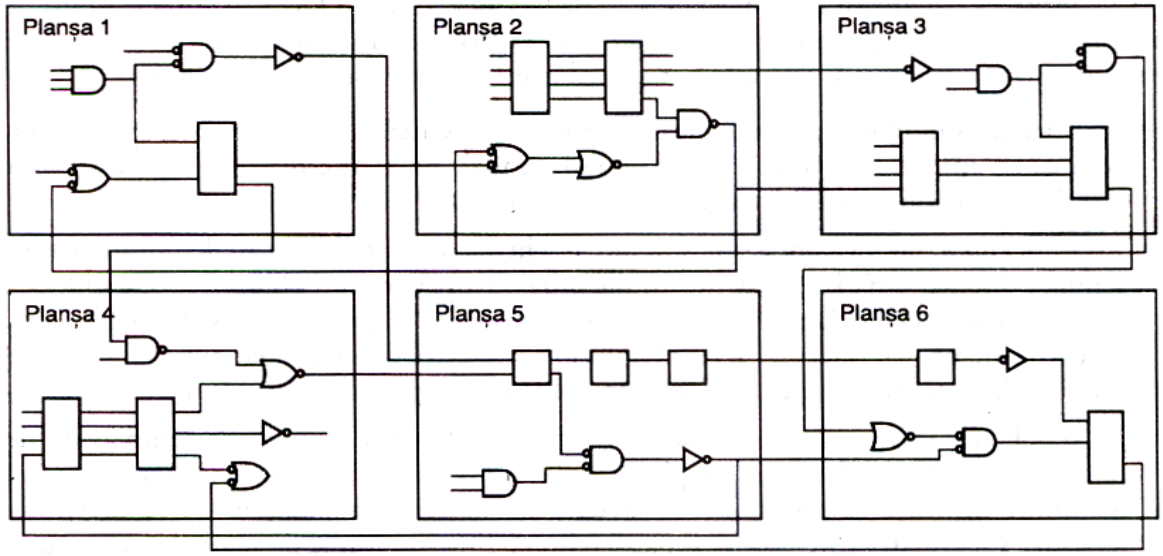


Figura 5-7 Schemă cu structură orizontală

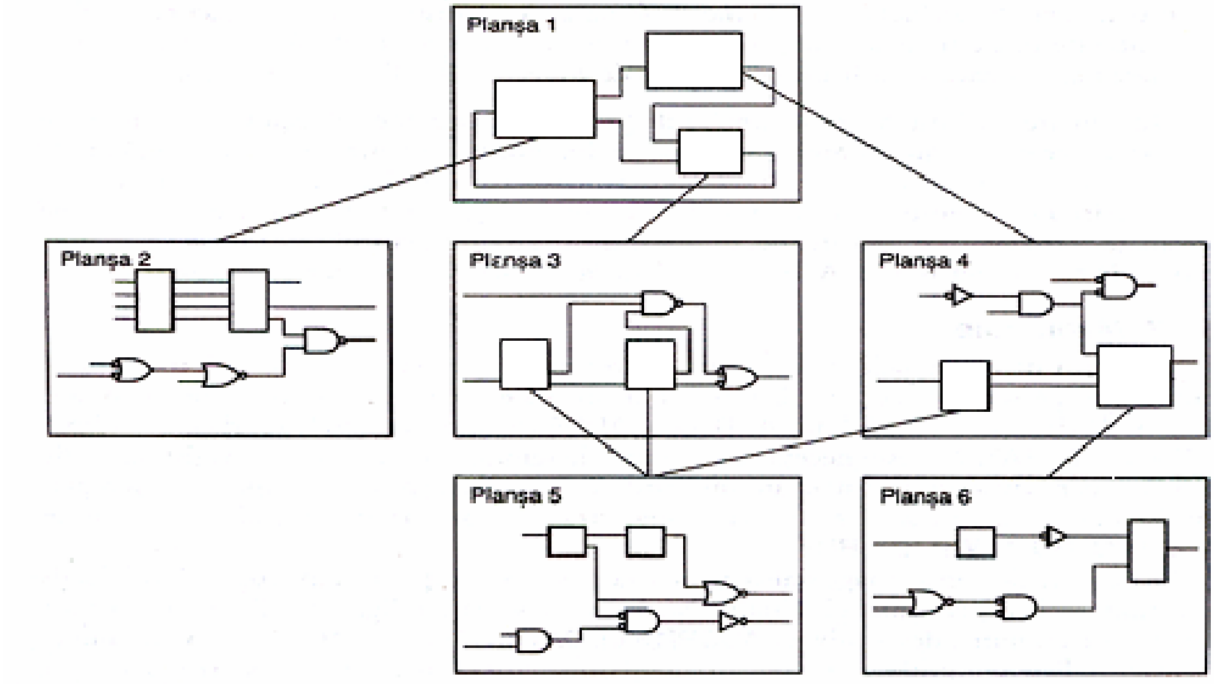


Figura 5-8 Schemă cu structură ierarhică

5.1.6 Magistrale

O magistrală este un ansamblu a două sau mai multe linii de semnal asociate. De exemplu, un sistem cu microprocesor poate avea o magistrală de

adrese cu 16 linii, **ADDRO ... ADDR15**, şi o magistrală de date cu 8 linii, **DATAO ... DATA7**. Nu este însă necesar ca denumirile semnalelor din aceeaşi magistrală să fie asemănătoare sau să se afle într-o anumită ordine. De exemplu, un sistem cu microprocesor poate avea o magistrală de comenzi conţinând cinci semnale: **ALE**, **MIO**, **RD_L**, **WR_L** şi **RDY**.

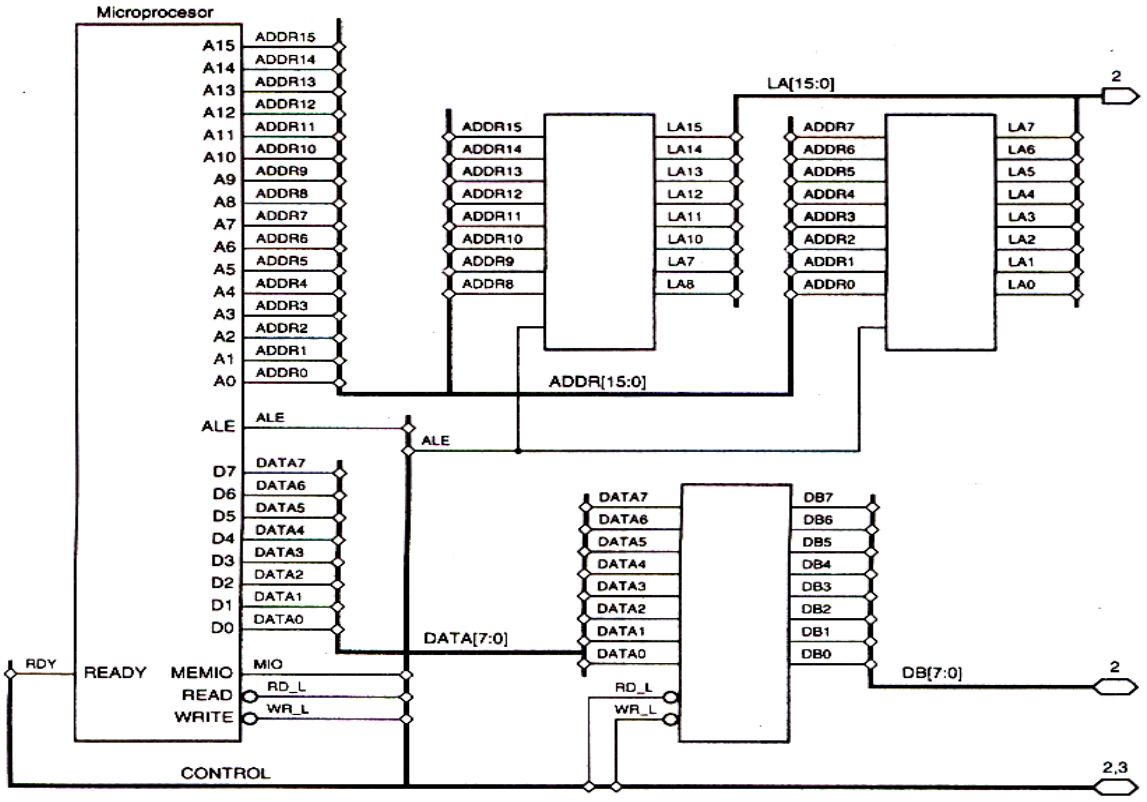


Figura 5-9 Exemple de magistrale

Pe diagramele logice, magistralele beneficiază de notaţii speciale, în scopul simplificării desenului şi al creşterii inteligibilităţii. După cum observaţi în figura 5-9, fiecare magistrală poartă câte un nume descriptiv, ca **ADDR[15:0]**, **DATA[7:0]** sau **CONTROL**. Într-un nume de magistrală se pot utiliza paranteze drepte şi caracterul „**:**” pentru precizarea unui domeniu. Magistralele se reprezintă în desene cu linii mai groase decât liniile de semnal obişnuite. Semnalele separate care se introduc sau se preiau dintr-o magistrală se reprezintă prin conectarea la magistrală a unei linii de semnal obişnuite, în dreptul căreia scriem numele semnalului respectiv. Uneori se marchează punctul de conexiune printr-un simbol de formă specială, ca în exemplu.

Sistemele de proiectare asistată de calculator urmăresc separat fiecare semnal dintr-o magistrală. Când se ajunge în faza de construire a unui circuit corespunzător unei scheme, liniile de semnal cuprinse într-o magistrală sunt considerate ca trasate separat.

Simbolurile din marginea din dreapta a figurii 5-9 sunt trimiteri către alte planşe. Ele arată că **LA** se continuă în planşa 2, **DB** este bidirecţională şi se

continuă în planşa 2, iar **CONTROL** este bidirecţională şi se continuă în planşele 2 şi 3.

5.2 PLD combinaţionale

5.2.1 Matrici logice programabile

Cronologic, primele PLD au fost *matricele logice programabile (PLA - programmable logic array). PLA* sunt dispozitive combinaţionale cu două niveluri **AND-OR**, care pot fi programate pentru a implementa orice expresie logică sumă de produse în limita dimensiunilor dispozitivului. Aceste limite sunt:

- numărul de intrări (n)
- numărul de ieşiri (m)
- numărul de termeni produs (p)

Am putea descrie un asemenea dispozitiv ca „PLA *n* x *m cu p* termeni produs". În general, *p* este mult mai mic decât numărul de mintermeni de *n* variabile (2^n). Deci un PLA nu poate realiza orice funcţii logice cu *n* intrări şi *m* ieşiri; utilitatea lui se limitează la funcţiile ce pot fi exprimate ca sumă de produse cu maximum *p* termeni produs.

Un *PLA n x m* cu p termeni produs conţine *p* porţi **AND** cu câte *2n* intrări şi *m* porţi **OR** cu câte *p* intrări. În figura 5-10 este prezentat un PLA de dimensiuni modeste, cu patru intrări, şase porţi **AND**, trei porţi **OR** şi tot atâtea ieşiri. Fiecare intrare este conectată la câte un circuit tampon care furnizează atât varianta directă, cât şi pe cea complementată a semnalului, spre a fi utilizate în interiorul matricei. Conexiunile posibile din interior sunt marcate cu **X**; dispozitivul se programează prin păstrarea exclusiv a conexiunilor necesare practic. Conexiunile selectate sunt *fuzibile,* constând fie din material fuzibil fizic, fie din celule de memorie nevolatilă, în funcţie de tehnologia aplicată.

Deci la intrările fiecărei porţi **AND** se poate regăsi orice subset alcătuit din semnale primare de intrare şi complementele acestora. În mod asemănător, la intrările fiecărei porţi **OR** se poate regăsi orice subset format din semnale de ieşire ale porţilor **AND**.

Cum se vede în figura 5-11, un PLA poate fi reprezentat şi printr-o schemă mai compactă. Mai mult decât atât, dispunerea elementelor acestei scheme seamănă foarte bine cu dispunerea lor reală pe cipul de PLA.

PLA din figura 5-11 poate realiza oricare trei funcţii logice combinaţionale cu patru intrări ce pot fi scrise ca sumă de produse folosind un total de maximum şase termeni produs diferiţi.

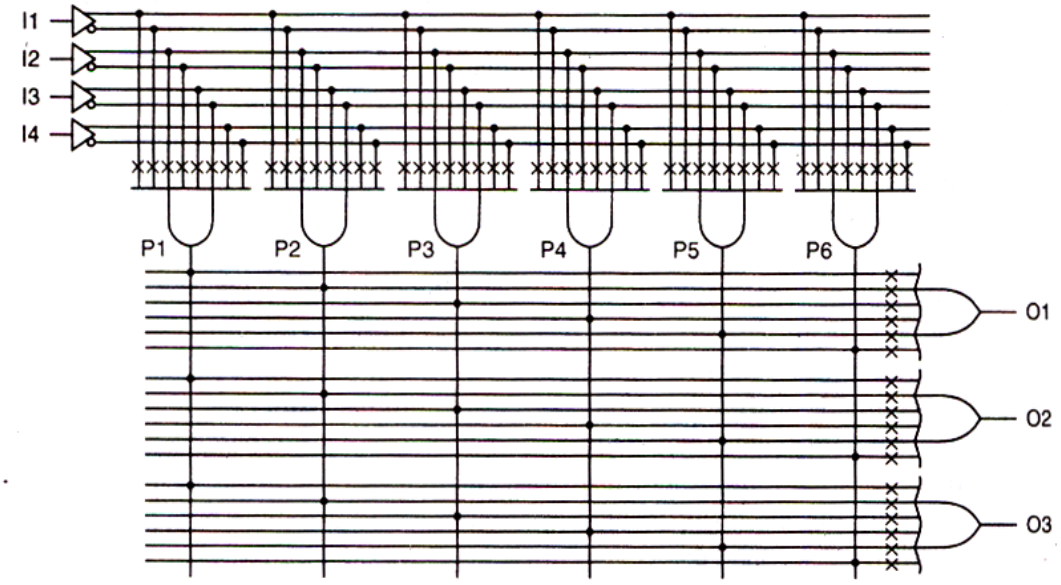


Figura 5-10 PLA 4x3 cu şase termeni produs

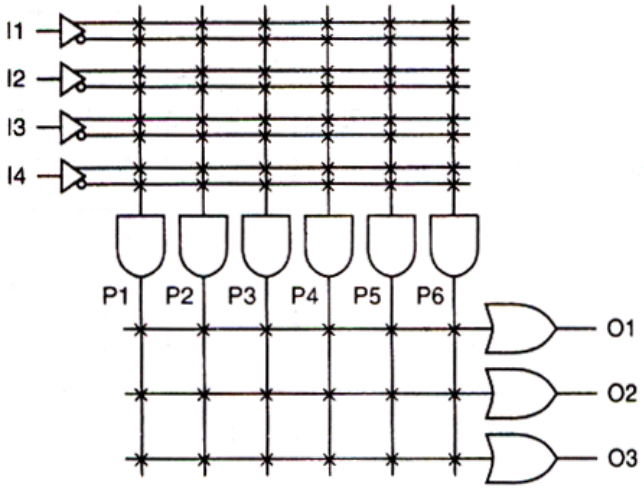


Figura 5-11 Reprezentare compactă a unui PLA 4x3 cu şase termeni produs

O1 = I1·I2 + I1'·I2'·I3'·I4'
O2 = I1·I3' + I1'·I3·I4 + I2
O3 = I1·I2 + I1·I3' + I1'·I2'·I4'

Ecuaţiile de mai sus au în total opt termeni produs, dar primii doi termeni din ecuaţia **03** sunt identici cu primii termeni ai ecuaţiilor **01** şi **02**. Matricea de conexiuni programate, din figura 5-12, trebuie să corespundă acestor ecuaţii logice.

Uneori, ieşirea unui PLA trebuie programată astfel încât să fie constant 1 sau constant 0. Nu este greu, cum se observă şi în figura 5-13. Termenul produs **P1** este întotdeauna 1deoarece linia sa de produse nu este conectată la nici o

intrare şi deci este totdeauna forţat în **HIGH**; acest termen constant 1 comandă ieşirea **O1**.

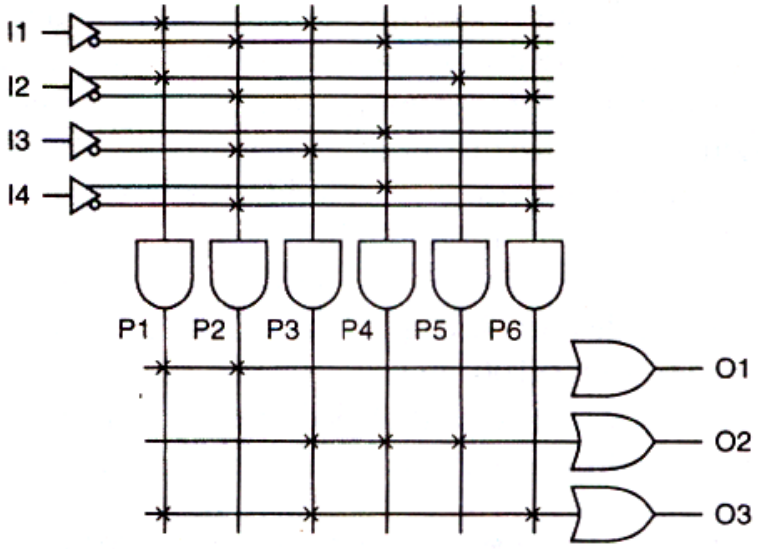


Figura 5-12 PLA 4x3 programat cu un set de trei ecuaţii logice

Ieşirea **O2** nu este comandată de nici un termen produs, fiind deci totdeauna 0. O altă metodă de a obţine o valoare de ieşire 0 constantă este cea aplicată în cazul ieşirii **O3**. Termenul produs **P2** este conectat la fiecare dintre variabilele de intrare, precum şi la complementele acestora; ca urmare, valoarea lui este totdeauna 0 (X • X’ = 0).

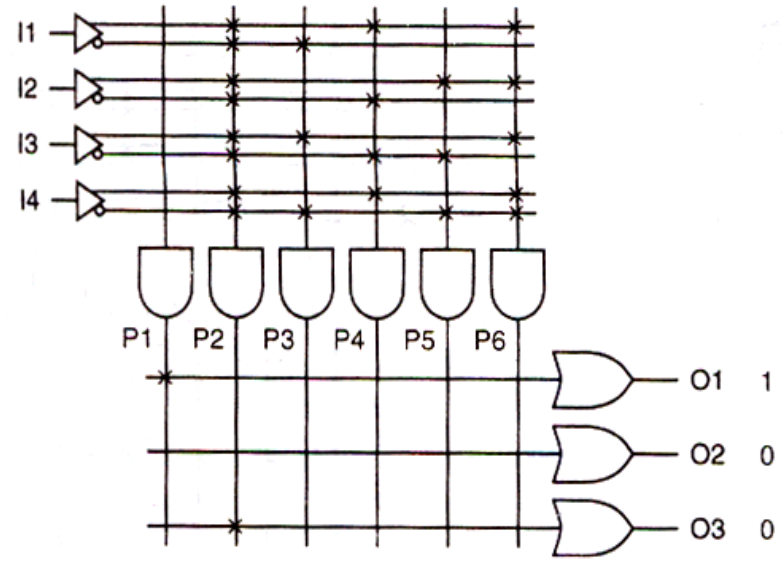


Figura 5-13 PLA 4x3 programat pentru a produce semnale de ieşire constante 0 şi 1

PLA cu care am lucrat în exemple are prea puţine intrări, ieşiri şi porţi **AND** (termeni produs) pentru a fi foarte util. Desigur că un PLA cu *n* intrări ar putea utiliza 2^n termeni produs pentru a implementa toţi mintermenii de *n*

variabile posibili. De fapt, numărul de termeni produs corespunzători unor PLA obişnuite, de uz larg, este mult mai mic, de ordinul a 4 ... 16 pentru fiecare ieşire, indiferent de valoarea *n*.

Un caz particular de PLA şi în prezent cel mai utilizat tip de PLD este *dispozitivul logic cu matrice programabilă* (PAL – *programmable array logic*). Spre deosebire de PLA, la care atât matricea de porţi **AND**, cât şi cea de porţi **OR** sunt programabile, matricea de porţi **OR** a unui dispozitiv PAL este fixă.

5.3 Decodoare

Decodorul este un circuit logic cu mai multe intrări şi mai multe ieşiri, care converteşte semnalele de intrare codate în semnale de ieşire codate, codurile de intrare şi de ieşire fiind diferite. În general, codul de intrare este construit pe mai puţini biţi decât codul de ieşire, iar între cuvintele de cod de intrare şi cuvintele de cod de ieşire există o *corespondenţă biunivocă.*

Structura generală a unui circuit de decodare este cea din figura 5-14. Intrările de activare, dacă există, trebuie să fie confirmate pentru ca decodorul să realizeze corespondenţa intrare-ieşire în mod normal. În caz contrar, decodorul asociază tuturor cuvintelor de intrare un singur cuvânt de cod de ieşire – „disabled” („neactivat”).

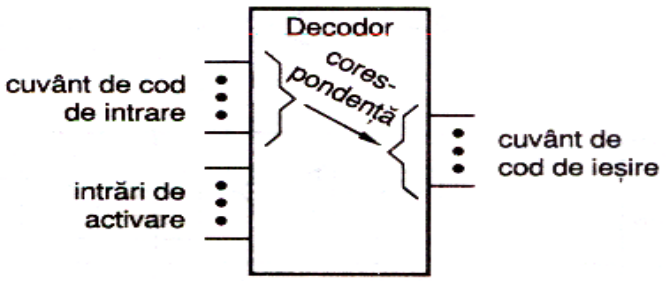


Figura 5-14 Structura circuitului de decodare

Pentru semnalul de intrare, cel mai frecvent se utilizează un cod binar de *n* biţi, în care un cuvânt de *n* biţi reprezintă una dintre cele 2^n valori codate diferite, în mod normal - numerele întregi de la 0 la 2^n-1. Uneori, codurile binare de *n* biţi se trunchiază, reprezentându-se astfel mai puţin de 2^n valori. De exemplu, în codul BCD, combinaţiile de 4 biţi de la 0000 la 1001 reprezintă cifrele zecimale 1...9, iar combinaţiile de la 1010 la 1111 nu sunt utilizate.

Pentru semnalul de ieşire, cel mai frecvent se utilizează un cod *1 din m,* care conţine *m* biţi, în orice moment fiind confirmat unul dintre biţi. Astfel, pentru un cod *1 din 4* cu valorile de ieşire active în **HIGH**, cuvintele de cod sunt *0001, 0010, 0100* şi *1000*. Dacă valorile de ieşire sunt active în **LOW**, cuvintele de cod sunt *1110, 1101, 1011* şi *0111*.

5.3.1 Decodoare binare

Cel mai comun circuit de decodare este decodorul cu *n* intrări şi 2^n ieşiri *sau decodorul binar.* La un asemenea decodor, codul de intrare este un cod binar de *n* biţi, iar cel de ieşire este un cod 1 din 2^n. Decodorul binar se foloseşte când este necesară activarea unei singure ieşiri din 2^n posibile, corespunzător unei valori de intrare exprimate cu *n* biţi.

Pentru exemplificare, în figura 5-15(a) apar intrările şi ieşirile unui decodor cu 2 intrări şi 4 ieşiri, iar tabelul 5-1 este tabelul de adevăr care îi corespunde. Cuvântul de cod de intrare **I1**, **I0** reprezintă un număr întreg cuprins între 0 şi 3. În cuvântul de cod de ieşire **Y3**, **Y2**, **Y1**, **Y0**, $\mathbf{Y}_i$ are valoarea 1 dacă şi numai dacă îi corespunde cuvântul de cod de intrare ce reprezintă numărul *i* în binar, iar *intrarea de activare (enable)* **EN** este 1. Dacă **EN** este 0, toate valorile de ieşire sunt 0. Circuitul la nivel de porţi corespunzător decodorului cu două intrări şi patru ieşiri este cel din figura 5-15(b). Fiecare poartă **AND** realizează *decodarea* câte unei combinaţii cuvânt de cod de intrare **I1**, **I0**.

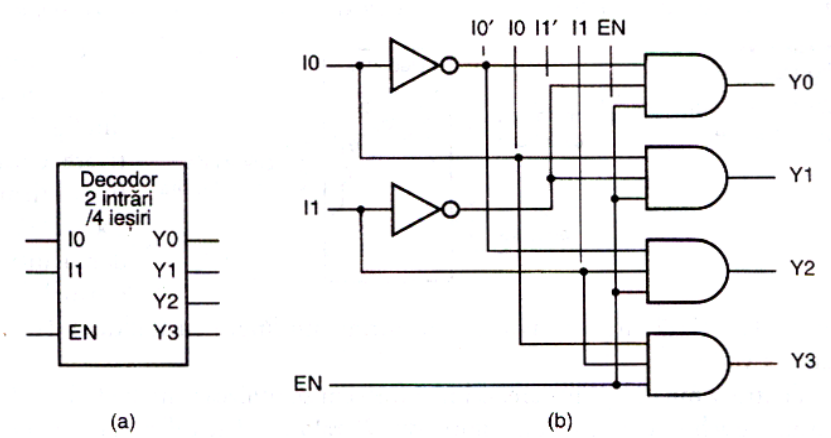


Figura 5-15 Decodor cu 2 intrări şi 4 ieşiri: (a) intrările şi ieşirile; (b) schema logică

Tabel 5-1 Tabelul de adevăr pentru decodorul binar cu 2 intrări şi 4 ieşiri

Intrări			Ieşiri			
EN	I1	I0	Y3	Y2	Y1	Y0
0	x	x	0	0	0	0
1	0	0	0	0	0	1
1	0	1	0	0	1	0
1	1	0	0	1	0	0
1	1	1	1	0	0	0

Codul de intrare al unui decodor binar de *n* biţi nu trebuie să reprezinte neapărat numere întregi între 0 şi 2^n-1. Nu este necesar nici să se folosească toate ieşirile unui decodor, şi nici să se decodeze toate combinaţiile de intrare posibile. De exemplu, un *decodor zecimal* sau *BCD* decodează numai primele zece combinaţii de intrare binare, de la *0000* la *1001*, pentru a genera valorile de ieşire **YO**...**Y9**.

5.3.2 Decodorul dublu cu două intrări şi patru ieşiri 74x139

Componenta MSI *74x139* conţine două decodoare cu două intrări şi patru ieşiri, independente şi identice. Schema la nivel de porţi a acestui CI este cea din figura 5-16(a). Circuitul ‘139 are ieşirile şi intrarea de activare active în **LOW**. Majoritatea decodoarelor MSI au fost concepute iniţial cu ieşirile active în **LOW** deoarece porţile TTL inversoare sunt, în general, mai rapide decât cele neinversoare. Circuitul ‘139 mai are nişte inversoare suplimentare pe intrările de selectare. În absenţa acestora, la fiecare intrare de selectare ar fi conectate trei sarcini de c.a. sau c.c., în loc de una singură, crescând din această cauză solicitarea asupra dispozitivului care comandă intrarea respective, din punctul de vedere al parametrului fanout.

În figura 5-16(b) apare un simbol logic al circuitului 74x139. Remarcaţi că toate denumirile semnalelor din interiorul conturului simbolului sunt active în **HIGH** (nu au sufixul „**L**”) şi că prin cerculeţele inversoare se arată că intrările şi ieşirile sunt active în **LOW**.

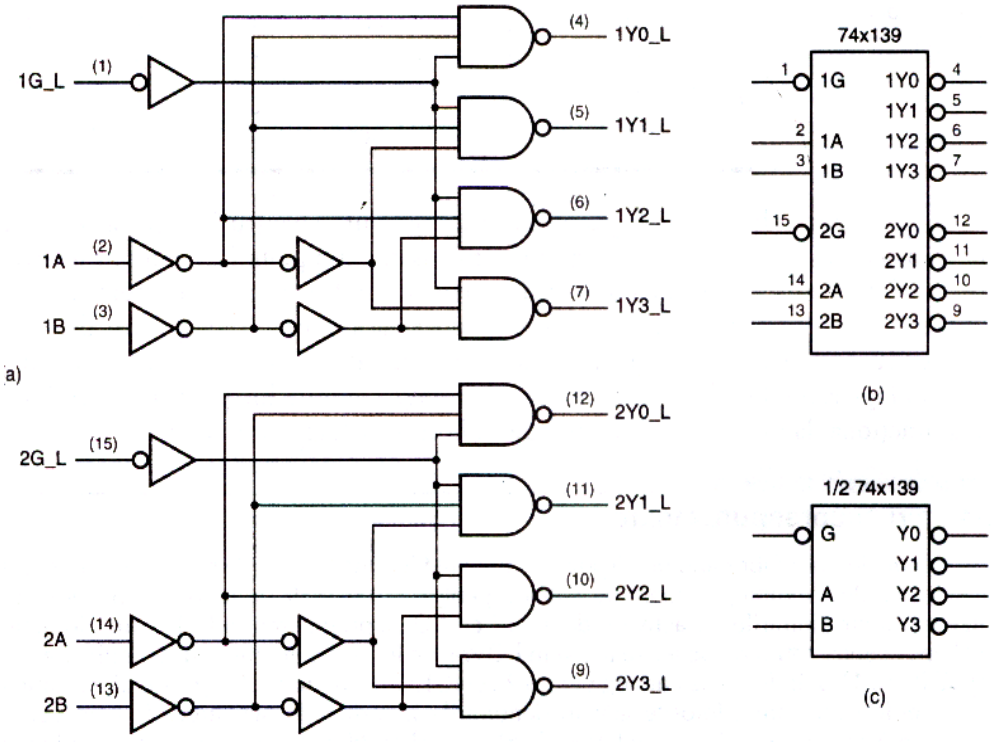


Figura 5-16 Decodorul dublu cu două intrări şi patru ieşiri 74x139: (a) schema logică, inclusiv numerotarea pinilor la o capsulă standard DIP cu 16 pini; (b) simbolul logic tradiţional; (c) simbolul logic aferent unui singur decodor

Tabel 5-2 Tabelul de adevăr pentru jumătate din decodorul dublu cu două intrări şi patru ieşiri 74x139

Intrări			Ieşiri			
G_L	B	A	Y3_L	Y2_L	Y1_L	Y0_L
1	x	x	1	1	1	1
0	0	0	1	1	1	0
0	0	1	1	1	0	1
0	1	0	1	0	1	1
0	1	1	0	1	1	1

5.3.3 Decodorul cu trei intrări şi opt ieşiri 74x138

74x138 este un decodor MSI cu 3 intrări şi 8 ieşiri. În figura 5-17 este prezentată schema circuitului la nivel de porţi logice şi simbolul lui. Ca şi 74x139, 74x138 are ieşirile active în **LOW** şi trei intrări de activare (**G1**, **G2A_L**, **G2B_L**), oricare dintre acestea trebuind să fie confirmată pentru ca ieşirea selectată să fie confirmată. Funcţia logică a dispozitivului este simplă - o ieşire este confirmată dacă şi numai dacă decodorul este activat şi ieşirea respectivă este selectată.
(ex.:**Y5=G1·G2A·G2B · C·B'·A; Y5_L'=G1·G2A_L'·G2B_L' · C·B'·A**).

Tabel 5-3 Tabelul de adevăr pentru decodorul cu 3 intrări şi 8 ieşiri 74x138

Intrări						Ieşiri							
G1	G2A_L	G2B_L	C	B	A	Y7_L	Y6_L	Y5_L	Y4_L	Y3_L	Y2_L	Y1_L	Y0_L
0	x	x	x	x	x	1	1	1	1	1	1	1	1
x	1	x	x	x	x	1	1	1	1	1	1	1	1
x	x	1	x	x	x	1	1	1	1	1	1	1	1
1	0	0	0	0	0	1	1	1	1	1	1	1	0
1	0	0	0	0	1	1	1	1	1	1	1	0	1
1	0	0	0	1	0	1	1	1	1	1	0	1	1
1	0	0	0	1	1	1	1	1	1	0	1	1	1
1	0	0	1	0	0	1	1	1	0	1	1	1	1
1	0	0	1	0	1	1	1	0	1	1	1	1	1
1	0	0	1	1	0	1	0	1	1	1	1	1	1
1	0	0	1	1	1	0	1	1	1	1	1	1	1

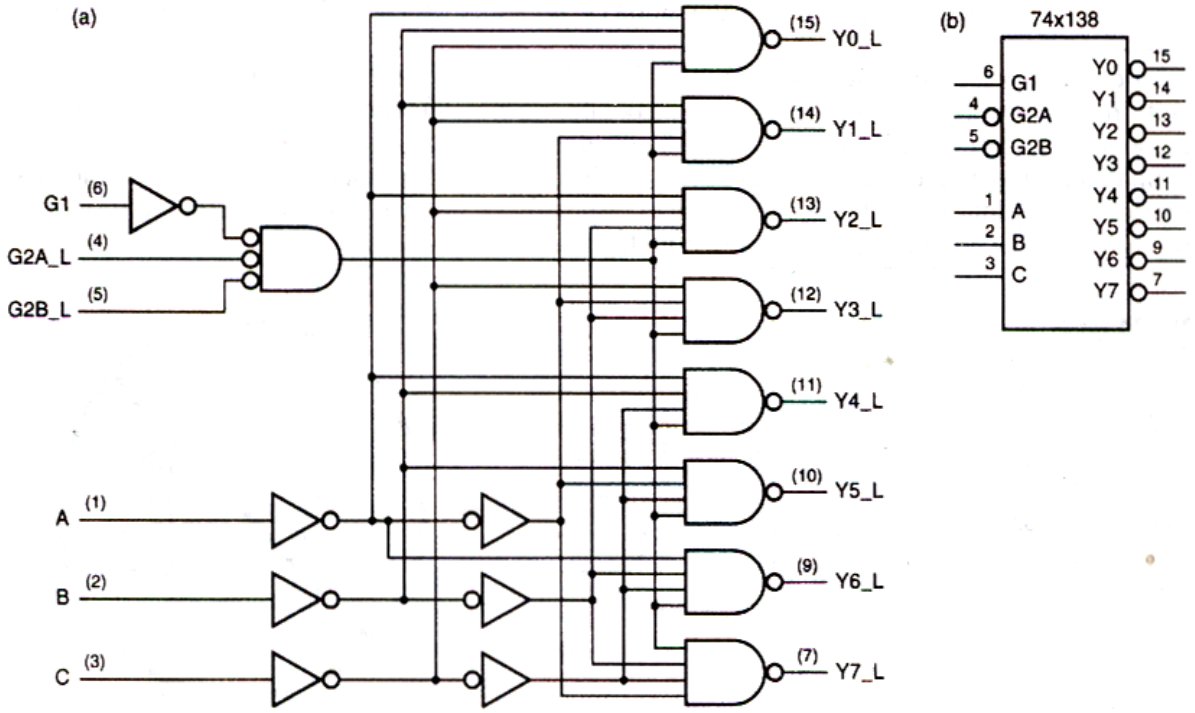


Figura 5-17 Decodorul cu 3 intrări şi 8 ieşiri 74x138: (a) schema logică, inclusiv numerotarea pinilor pentru capsula DIP cu 16 pini; (b) simbolul logic tradiţional

5.3.4 Conectarea în cascadă a decodoarelor binare

Pentru decodarea unor cuvinte de cod mai mari se pot utiliza mai multe decodoare binare conectate în cascadă. În figura 5-18 se arată cum se pot combina două decodoare cu 3 intrări şi 8 ieşiri pentru a obţine un decodor cu 4 intrări şi 16 ieşiri. Întrucât intrările de activare ale dispozitivului 74x138 pot fi active atât în **HIGH**, cât şi în **LOW**, există posibilitatea activării unui decodor sau a celuilalt în funcţie de starea celui mai semnificativ bit de intrare. Decodorul din partea de sus a desenului (U1) este activat când **N3** este 0, iar decodorul de jos (U2) este activat când **N3** este 1. Pentru a prelucra cuvinte de cod chiar mai mari decât acestea, decodoarele binare se pot conecta în cascadă ierarhic.

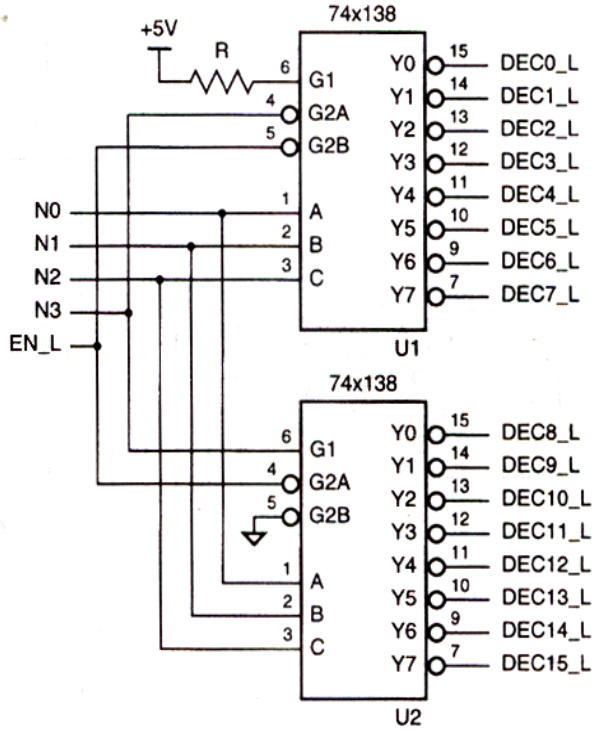


Figura 5-18 Schema unui decoder cu 4 intrări şi 16 ieşiri, realizat cu decodoare 74x138 conectate în cascadă

5.3.5 Decodoare pentru şapte segmente

Priviţi la încheietura mâinii şi veţi vedea, probabil, un *afişor cu şapte segmente*. Acest tip de afişare, la care se utilizează, în mod normal, diode luminescente (LED) sau elemente de afişare cu cristale lichide (LCD), se foloseşte la ceasuri, calculatoare şi instrumente care afişează date numerice. O cifră apare când se luminează o submulţime a celor şapte segmente prezentate în figura 5-19(a).

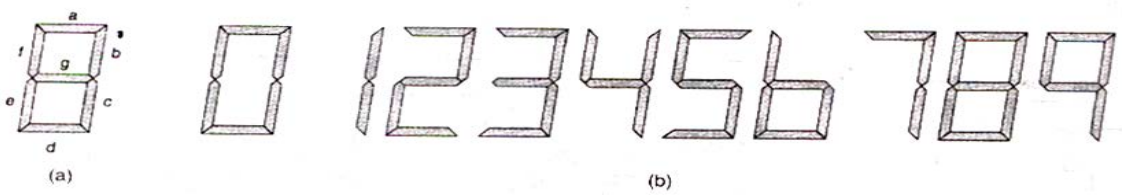


Figura 5-19 Afişare cu şapte segmente: (a) notarea segmentelor; (b) cifrele zecimale

Un *decodor pentru şapte segmente* are drept cod de intrare un cod BCD de 4 biţi şi drept cod de ieşire, codul de şapte segmente din figura 5-19(b). În figura 5-20 şi în tabelul 5-4 apar schema logică şi, respectiv, tabelul de adevăr aferente unui decodor pentru şapte segmente 74x49. Cu excepţia conexiunii corespunzătoare „intrării de ştergere” („blanking”), **BI_L**, fiecare dintre ieşirile unui 74x49 este implementarea unor sume de produse minimale corespunzătoare segmentului respectiv, considerând că pentru combinaţiile de intrare ce nu reprezintă cifre zecimale valorile sunt „indiferente”. Structura **NOT-OR-AND**, utilizată pentru fiecare ieşire, poate părea puţin mai ciudată, dar este echivalentă, cu o poartă **AND-OR-NOT**, care are o structură suficient de rapidă şi de compactă pentru a fi folosită în CMOS sau TTL.

Majoritatea elementelor de afişare moderne cu şapte segmente au încorporate decodoare, astfel ca unui asemenea dispozitiv i se poate aplica direct un cuvânt BCD de 4 biţi. Multe dintre decodoarele discrete cu şapte segmente mai vechi sunt dotate cu ieşiri speciale pentru valori mari de tensiune sau de curent, adecvate comandării elementelor de afişare de dimensiuni şi puteri mari.

Tabel 5-4 Tabelul de adevăr pentru decodorul pentru şapte segmente

Intrări					Ieşiri						
BI_L	D	C	B	A	a	b	c	d	e	f	g
0	x	x	x	x	0	0	0	0	0	0	0
1	0	0	0	0	1	1	1	1	1	1	0
1	0	0	0	1	0	1	1	0	0	0	0
1	0	0	1	0	1	1	0	1	1	0	1
1	0	0	1	1	1	1	1	1	0	0	1
1	0	1	0	0	0	1	1	0	0	1	1
1	0	1	0	1	1	0	1	1	0	1	1
1	0	1	1	0	0	0	1	1	1	1	1
1	0	1	1	1	1	1	1	0	0	0	0

1	1	0	0	0	1	1	1	1	1	1	1
1	1	0	0	1	1	1	1	0	0	1	1
1	1	0	1	0	0	0	0	1	1	0	1
1	1	0	1	1	0	0	1	1	0	0	1
1	1	1	0	0	0	1	0	0	0	1	1
1	1	1	0	1	1	0	0	1	0	1	1
1	1	1	1	0	0	0	0	1	1	1	1
1	1	1	1	1	0	0	0	0	0	0	0

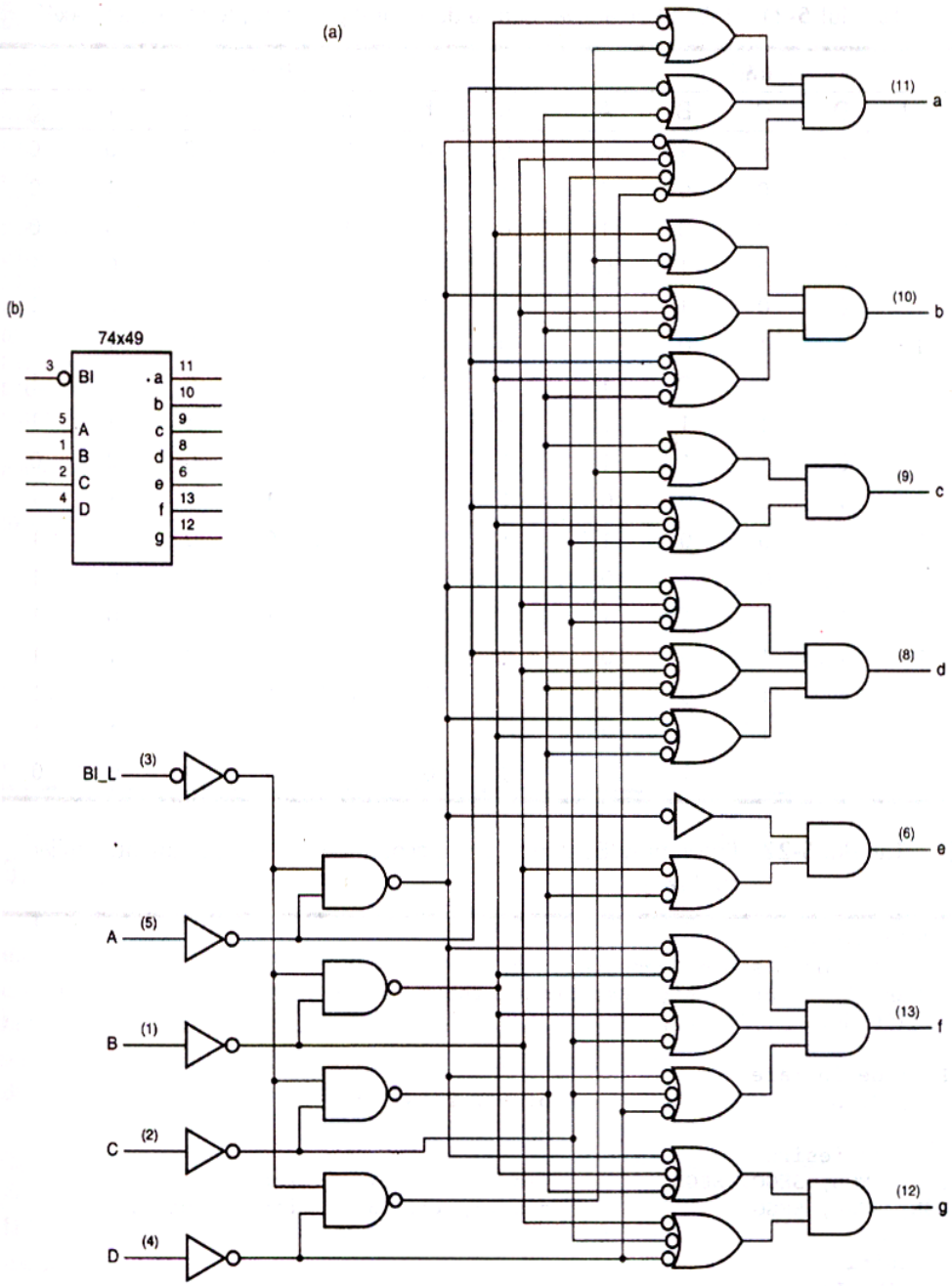


Figura 5-20 Decodorul pentru şapte segmente 74x49: (a) schema logică, inclusiv numerotarea pinilor; (b) simbolul logic tradiţional

5.4 Circuite de codare

Codul de ieşire al unui decodor are, în mod normal, mai mulţi biţi decât codul de intrare. Când codul de ieşire al unui dispozitiv are *mai puţini* biţi decât codul de intrare, dispozitivul este denumit, în general, *circuit de codare.*

Cel mai simplu circuit de codare ce poate fi construit este, probabil, cel cu 2^n intrări şi *n* ieşiri (sau *binar).* Aşa cum arată figura 5-21(a), acesta funcţionează exact invers decât un decodor binar: Codul lui de intrare este 1 din 2^n, iar la ieşire apare un cod binar de *n* biţi. Iată ecuaţiile aferente unui circuit de codare cu 8 intrări şi 3 ieşiri, intrările fiind **I0** ... **I7**, iar ieşirile, **Y0** ... **Y2***:*

Y0=I1+I3+I5+I7
Y1=I2+I3+I6+I7
Y2=I4+I5+I6+I7

Circuitul logic corespunzător este prezentat în figura 5-21(b). În general, un circuit de codare cu 2^n intrări şi *n* ieşiri poate fi construit din n porţi **OR** cu câte 2^{n-1} intrări. Bitul *i* din codul de intrare se conectează la poarta **OR** cu numărul j dacă bitul j din reprezentarea binară a numărului i este 1.

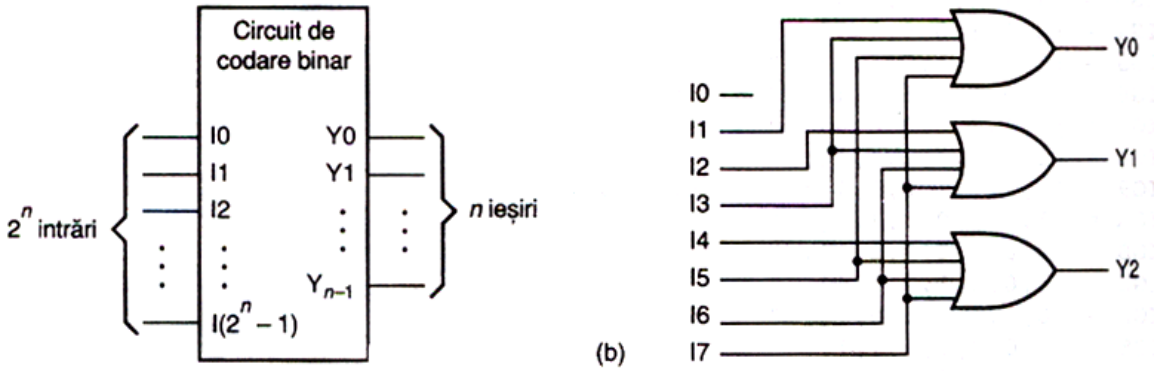


Figura 5-21 Circuit de codare binară: (a) structura generală; (b) circuit de codare cu 8 intrări şi 3 ieşiri

5.4.1 Matrice de priorităţi

Ieşirile codate 1 din 2^n ale unui decodor binar de *n* biţi se utilizează, în general, pentru a comanda un grup de 2^n dispozitive, dintre care cel mult unul trebuie să fie activ în orice moment. Invers, să considerăm un sistem cu 2^n *intrări,* fiecare dintre acestea reprezentând o cerere de servire, ca în figura 5-22. Asemenea structuri se întâlnesc frecvent în subsistemele de intrări/ieşiri din microprocesoare, intrările putând reprezenta cereri de întrerupere.

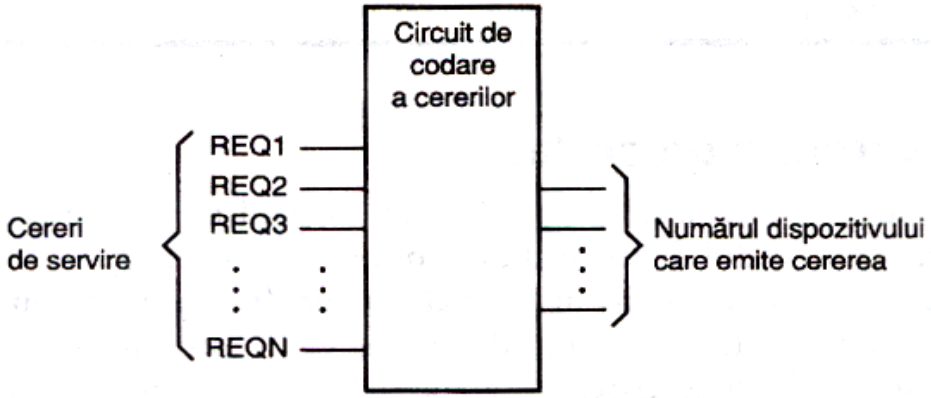


Figura 5-22 Sistem cu 2n dispozitive ce trebuie servite şi un circuit de codare a cererilor care arată în orice moment ce semnal de cerere este confirmat

În astfel de cazuri poate părea firească utilizarea unui circuit de codare ca acela din figura 5-22, pentru a urmări intrările şi a indica în orice moment care dintre ele adresează o cerere de servire. Dar circuitul funcţionează corect numai dacă există certitudinea că la un moment dat poate fi confirmată cel mult o intrare. Dacă există simultan mai multe cereri, la ieşirea circuitului de codare apar semnale inacceptabile.

Soluţia este atribuirea de *priorităţi* liniilor de intrare, astfel încât, atunci când sunt confirmate mai multe cereri, circuitul de codare să genereze numărul dispozitivului care emite cererea cu cel mai înalt grad de prioritate. Dispozitivul ce realizează această codare se numeşte *matrice de priorităţi.*

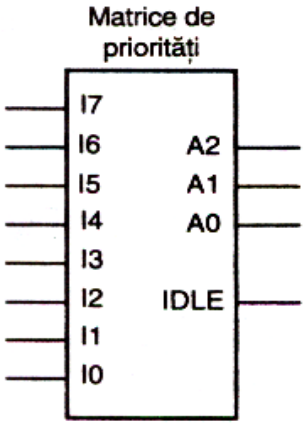


Figura 5-23 Simbolul logic al unei matrice de priorităţi generice cu 8 intrări

Simbolul logic al unei matrice de priorităţi cu 8 intrări este cel din figura 5-23. Intrarea **I7** are cel mai înalt grad de prioritate. La ieşirile **A2...A0** se regăseşte numărul intrării confirmate cu gradul de prioritate cel mai înalt, dacă un asemenea semnal de intrare există. Ieşirea **IDLE** (de aşteptare) este confirmată dacă nu este confirmată nici una dintre intrări.

Pentru a scrie ecuaţiile logice corespunzătoare semnalelor de ieşire ale matricei de priorităţi, trebuie să definim în prealabil opt variabile intermediare,

H0...H7, astfel ca $\mathbf{H_i}$ să fie 1 dacă şi numai dacă $\mathbf{I_i}$ este intrarea cu valoarea 1 şi cu cel mai înalt grad de prioritate:

H7=I7'
H6=I6·I7'
H5=I5·I6'·I7'
.....
H0=I0·I1'·I2'·I3'·I4'·I5'·I6'·I7'

Utilizând aceste semnale, ecuaţiile corespunzătoare semnalelor de ieşire **A2...A0** se aseamănă cu cele aferente unui decodor binar simplu:

A2=H4+H5+H6+H7
A1=H2+H3+H6+H7
A0=H1+H3+H5+H7

Ieşirea IDLE este 1 dacă nici una dintre intrări nu este 1:

IDLE=(I0+I1+I2+I3+I4+I5+I6+I7)'=I0'+I1'+I2'+I3'+I4'+I5'+I6'+I7'

5.4.2 Matricea de priorităţi 74x148

74x148 este o matrice de priorităţi MSI cu 8 intrări. Simbolul său logic apare în figura 5-24, iar schema sa, în figura 5-25. Principala diferenţă dintre acest CI şi matricea de priorităţi „generică" din figura 5-23 constă în faptul că intrările şi ieşirile sunt active în **LOW**. De asemenea, mai există o intrare de activare, **EI**, care trebuie să fie confirmată pentru ca oricare dintre ieşiri să fie confirmată. Tabelul de adevăr apare integral în tabelul 5-5.

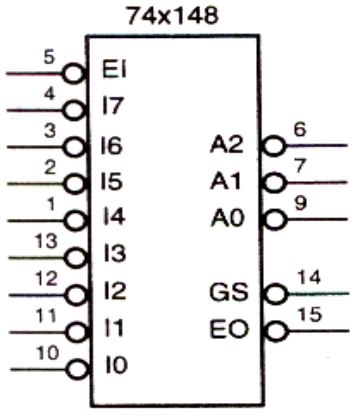


Figura 5-24 Simbolul logic al matricei de priorităţi cu 8 intrări 74x148

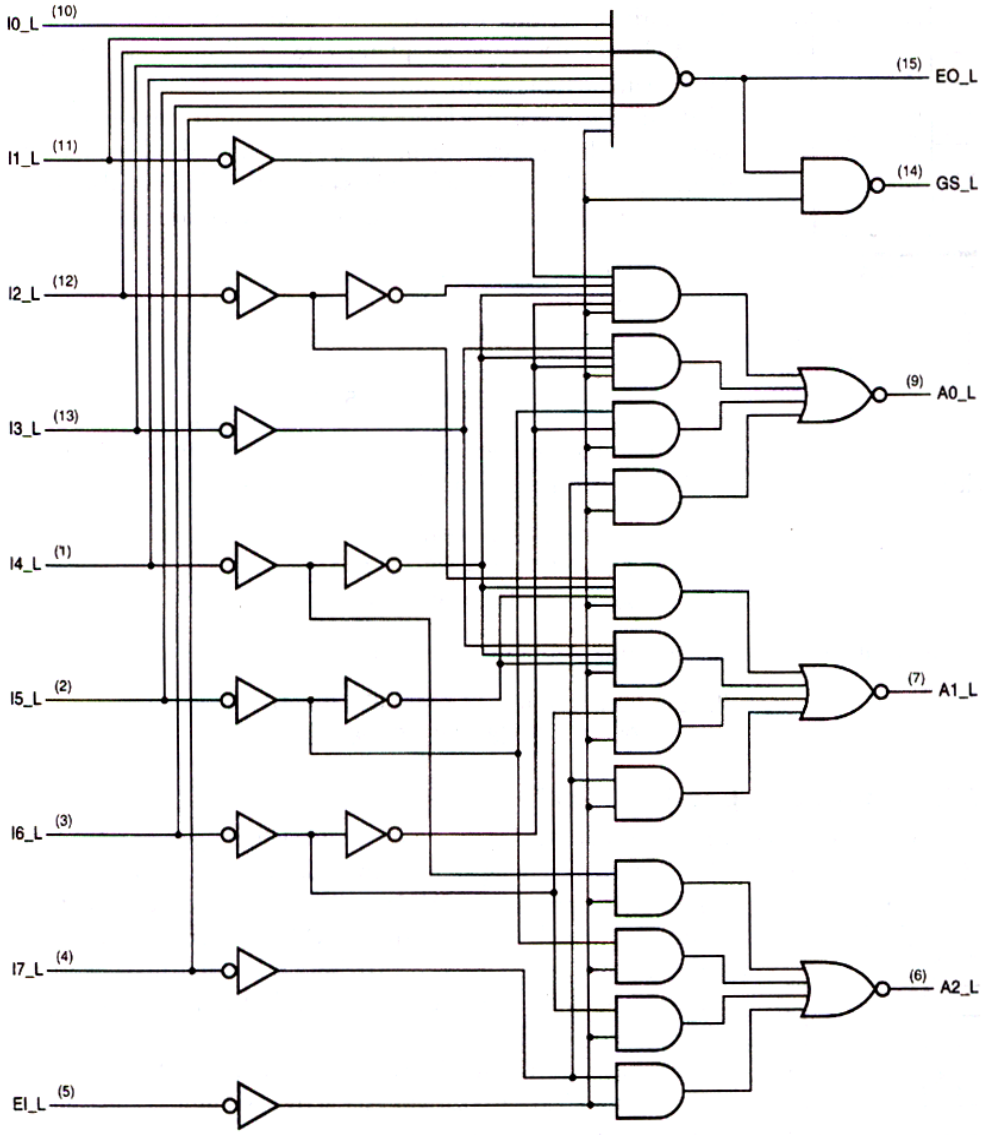


Figura 5-25 Schema logică a matricei de priorităţi cu 8 intrări 74x148, inclusiv numerotarea pinilor pentru capsula DIP cu 16 pini

Tabel 5-5 Tabelul de adevăr pentru o matrice de priorităţi cu 8 intrări 74x148

Intrări									Ieşiri				
EI_L	I0_L	I1_L	I2_L	I3_L	I4_L	I5_L	I6_L	I7_L	A2_L	A1_L	A0_L	GS_L	EO_L
1	x	x	x	x	x	x	x	x	1	1	1	1	1
0	x	x	x	x	x	x	x	0	0	0	0	0	1
0	x	x	x	x	x	x	0	1	0	0	1	0	1
0	x	x	x	x	x	0	1	1	0	1	0	0	1
0	x	x	x	x	0	1	1	1	0	1	1	0	1
0	x	x	x	0	1	1	1	1	1	0	0	0	1
0	x	x	0	1	1	1	1	1	1	0	1	0	1
0	x	0	1	1	1	1	1	1	1	1	0	0	1
0	0	1	1	1	1	1	1	1	1	1	1	0	1
0	1	1	1	1	1	1	1	1	1	1	1	1	0

5.5 Dispozitive cu trei stări

5.5.1 Circuite tampon cu trei stări

Dispozitivul de bază cu trei stări este *circuitul tampon cu trei stări,* numit şi *circuit de comandă cu trei stări.* În figura 5-26 sunt prezentate simbolurile logice a patru circuite tampon cu trei stări. Simbolul de bază este cel de circuit tampon neinversor (5-26(a), 5-26(b)) sau inversor (5-26(c), 5-26(d)). Semnalul suplimentar din partea superioară a simbolului reprezintă *o intrare de activare a celor trei stări,* care poate fi activă în **HIGH** (5-26(a), 5-26(c)) sau în **LOW** (5-26(b), 5-26(d)). Când intrarea de activare este confirmată, dispozitivul se comportă ca un circuit tampon sau ca un inversor obişnuit. Când intrarea de activare este negată, ieşirea dispozitivului este „flotantă", cu alte cuvinte, ieşirea trece în starea de înaltă impedanţă, echivalentă cu deconectarea.

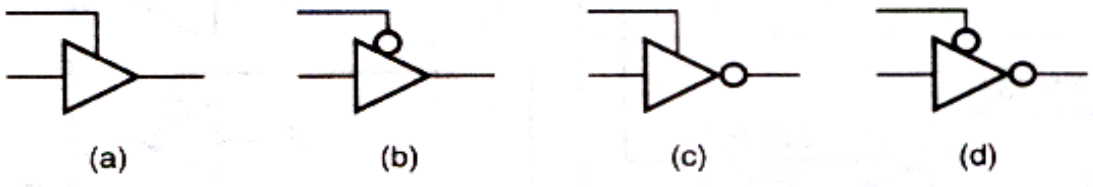


Figura 5-26 Diverse circuite tampon cu trei stări: (a) neinversor, cu intrarea de activare cu nivel activ HIGH; (b) neinversor, cu intrarea de activare cu nivel activ LOW; (c) inversor, cu intrarea de activare cu nivel activ HIGH; (d) inversor, cu intrarea de activare cu nivel activ LOW

Dispozitivele cu trei stări permit utilizarea în comun, de către mai multe surse, a unei singure „linii partajate", atâta timp cât, în orice moment, pe linie „vorbeşte" un singur dispozitiv. Figura 5-27 ilustrează modul în care se poate realiza acest lucru. Trei biţi de intrare, **SSRC2 ... SSRC0**, selectează una dintre cele opt surse de date ce pot comanda o singură linie, **SDATA.**

Un decodor cu 3 intrări şi 8 ieşiri, 74x138, asigură că numai una dintre cele opt linii **SEL** este confirmată în orice moment, permiţând numai unuia dintre circuitele tampon cu trei stări să comande linia **SDATA**. Dar dacă nu sunt confirmate toate liniile **EN**, nici unul dintre circuitele tampon cu trei stări nu este activat. În acest caz, valoarea logică de pe **SDATA** este nedefinită.

Dispozitivele tipice cu trei stări sunt concepute astfel încât să treacă în starea **Hi-Z** mai repede decât ies din ea. Înseamnă că dacă ieşirile a două dispozitive cu trei stări sunt conectate la o linie comună şi dezactivăm unul dintre ele concomitent cu activarea celuilalt, primul dispozitiv se va deconecta de la linie înainte ca al doilea să se conecteze. Acest aspect este important, deoarece, dacă ambele dispozitive ar comanda linia simultan şi fiecare ar încerca să menţină o valoare de ieşire opusă valorii de la celălalt dispozitiv (0 şi 1), ar apărea un supracurent care ar genera zgomot în sistem. Situaţia descrisă este denumită şi *conflict.*

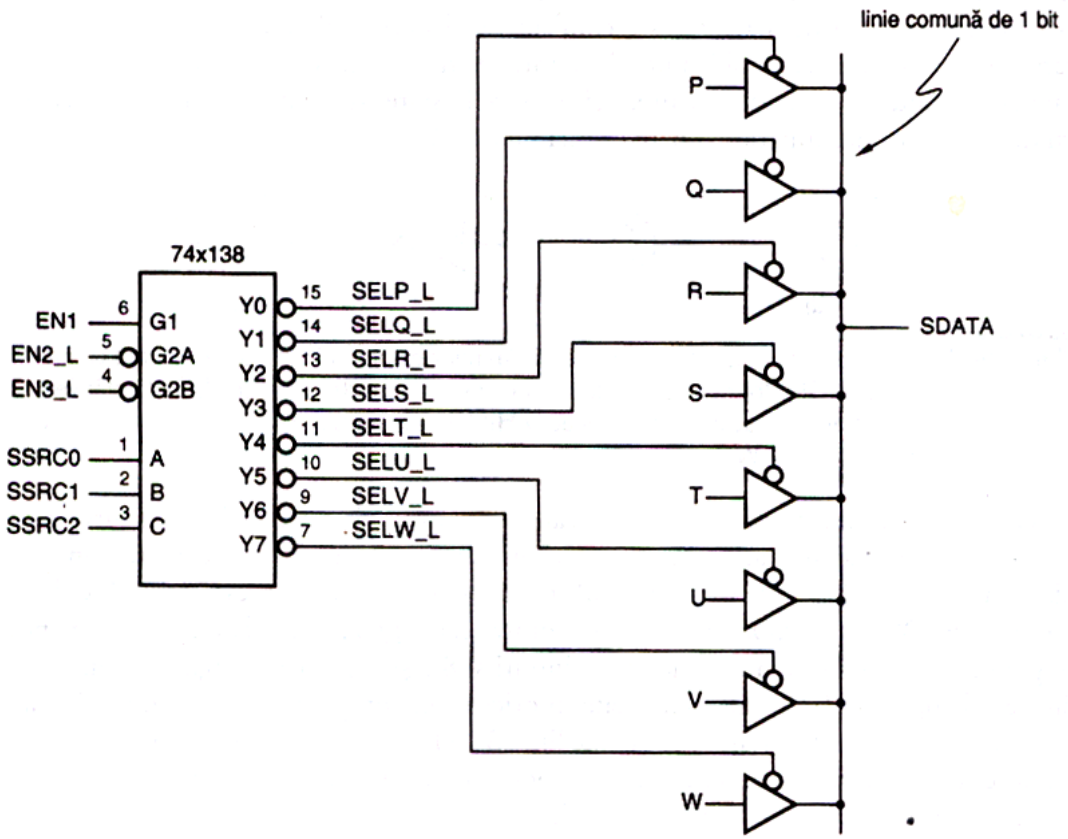


Figura 5-27 Opt surse folosind în comun o linie partajată cu trei stări

Din nefericire, din cauza decalajelor şi asimetriilor temporale din circuitele de comandă, este greu de impus ca intrările de activare ale mai multor dispozitive cu trei stări să se modifice simultan. Chiar şi atunci când există această posibilitate, apar probleme dacă la aceeaşi linie se conectează dispozitive cu trei stări aparţinând unor familii de circuite logice caracterizate de viteze diferite.

Singurul mod cu adevărat sigur în care se pot folosi dispozitivele cu trei stări este conceperea unei scheme logice de comandă care să garanteze apariţia, pe linia comună, a unui *timp mort,* în care nici unul dintre dispozitive să nu comande linia. Timpul mort trebuie să fie suficient de îndelungat pentru a soluţiona şi problemele din cazurile cele mai defavorabile de decalaje între momentele de închidere şi de deschidere ale dispozitivelor, precum şi de asimetrie a semnalelor de comandă cu trei stări.

5.6 Multiplexoare

Multiplexorul este un comutator digital care transmite la ieşire datele provenite de la una dintre cele n surse disponibile. Figura 5-28(a) prezintă intrările şi ieşirile unui multiplexor de *b* biţi, cu *n* intrări. Există *n* surse de date, fiecare de *b* biţi, şi *b* biţi de ieşire. La multiplexoarele comercializate în mod

obişnuit, n =1, 2, 4, 8 sau 16 şi b =1, 2 sau 4. Există şi intrări cu care se pot selecta cele n surse, deci s = [$\log_2 n$]. O intrare de activare, **EN**, permite ca multiplexorul să-şi facă treaba; când **EN**=0, toate ieşirile sunt 0. Adesea, denumirea „multiplexor” este prescurtată în *mux*.

Figura 5-28(b) prezintă un circuit cu comutatoare echivalent, în linii mari, cu un multiplexor. Însă, spre deosebire de circuitul cu comutatoare mecanice, multiplexorul este un dispozitiv unidirectional: informaţiile circulă numai dinspre intrări (în stânga) către ieşiri (în dreapta).

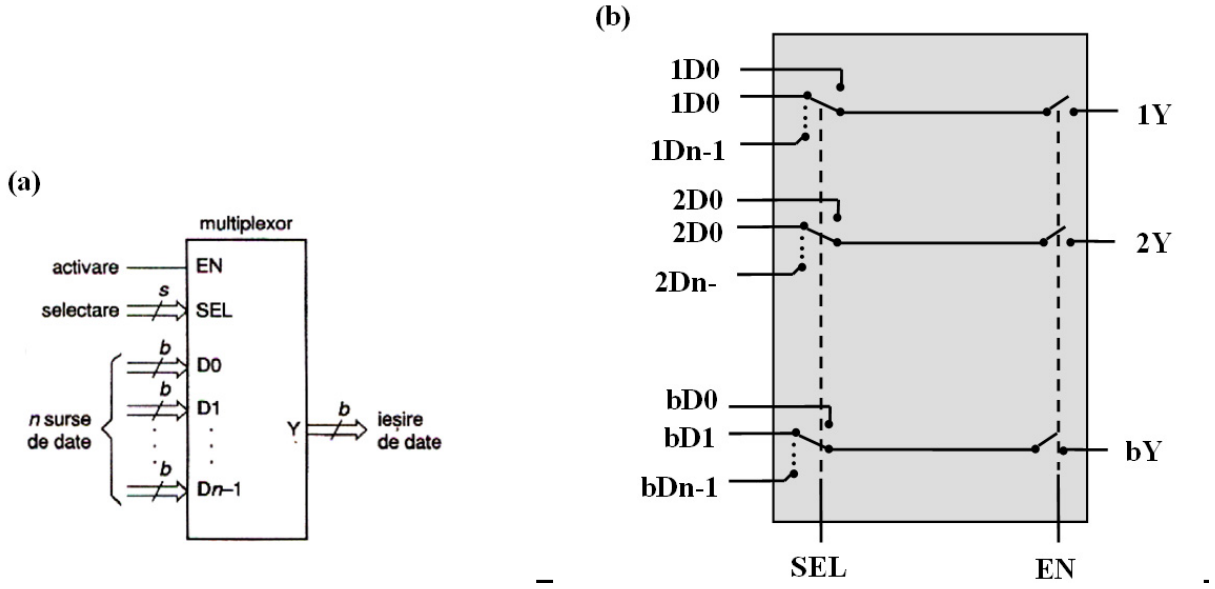


Figura 5-28 Structură de multiplexor: (a) intrările şi ieşirile; (b) circuit echivalent funcţional

Este evident că multiplexoarele sunt dispozitive utile în orice aplicaţie în care datele trebuie transferate din mai multe surse către o singură destinaţie. O aplicaţie banală din domeniul calculatoarelor este multiplexorul dintre registrele procesorului şi unitatea aritmetică logică (ALU) a acestuia. Pentru exemplificare, să considerăm un procesor de 16 biţi în care fiecare instrucţiune ocupă câte un câmp de 3 biţi ce arată care dintre cele opt registre urmează a fi folosit. Câmpul de 3 biţi este conectat la intrările de selectare ale unui multiplexor de 16 biţi, cu 8 intrări. Intrările de date ale multiplexorului sunt conectate la cele opt registre, iar ieşirile sale de date sunt conectate la ALU, pentru ca instrucţiunea să fie executată utilizând registrul selectat.

5.6.1 Multiplexoare MSI standard

Dimensiunile multiplexoarelor MSI disponibile pe piaţă sunt limitate de numărul de pini ce pot fi montaţi pe o capsulă de CI necostisitoare. Multiplexoarele de uz larg se produc în capsule cu 16 pini.

La una dintre extreme se află circuitul 74x151, din figura 5-29, care selectează dintre opt intrări de 1 bit. Intrările de selectare sunt denumite **C**, **B** şi

A, **C** fiind cea mai semnificativă numeric. Intrarea de activare **EN_L** este activă în **LOW,** dar sunt disponibile variante de ieşire active atât în **HIGH**, (**Y**), cât şi în **LOW** (**Y_L**).

Tabel 5-6 Tabelul de adevăr pentru multiplexorul de 1 bit cu 8 intrări 74x151

Intrări				Ieşiri	
EN_L	C	B	A	Y	Y_L
1	x	x	x	0	1
0	0	0	0	D0	D0’
0	0	0	1	D1	D1’
0	0	1	0	D2	D2’
0	0	1	1	D3	D3’
0	1	0	0	D4	D4’
0	1	0	1	D5	D5’
0	1	1	0	D6	D6’
0	1	1	1	D7	D7’

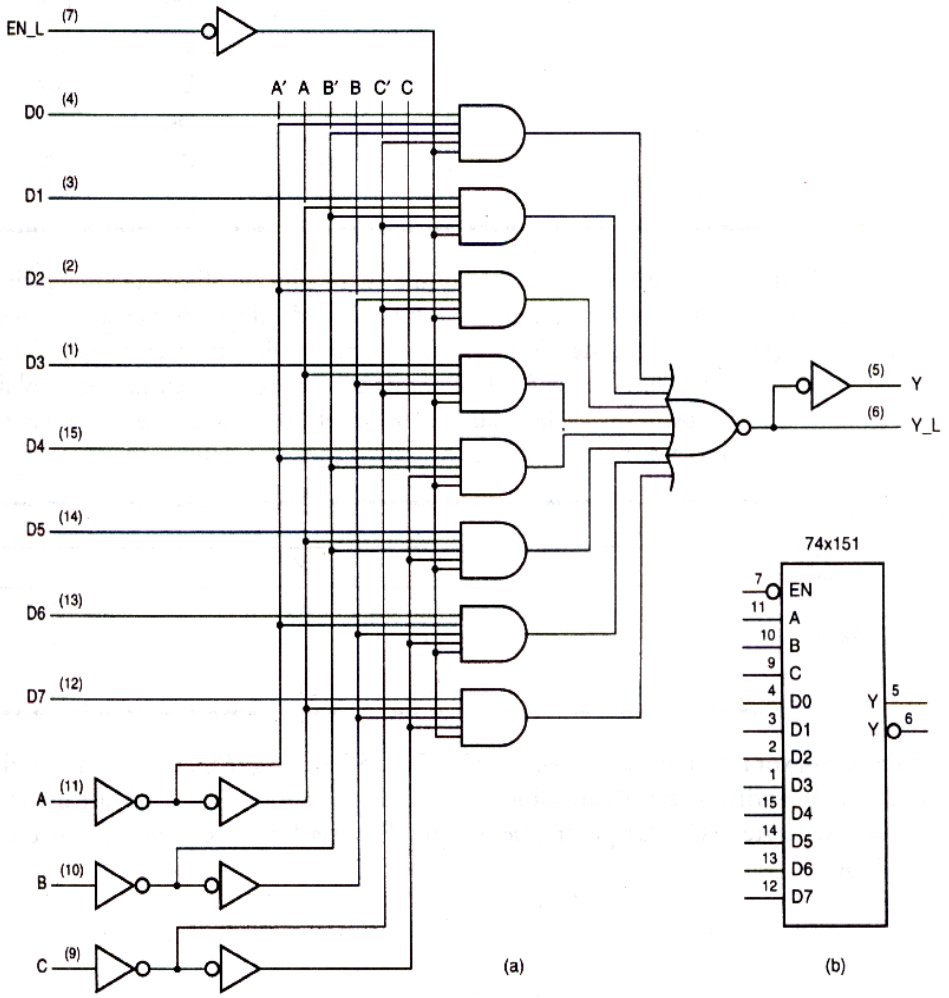


Figura 5-29 Multiplexor de 1 bit cu 8 intrări 74x151: (a) schema logică, inclusiv numerotarea pinilor; (b) simbolul logic tradiţional

La cealaltă extremă, ca multiplexoare cu capsule cu 16 pini, se află 74x157, din figura 5-30, care selectează dintre două intrări de câte 4 biţi.

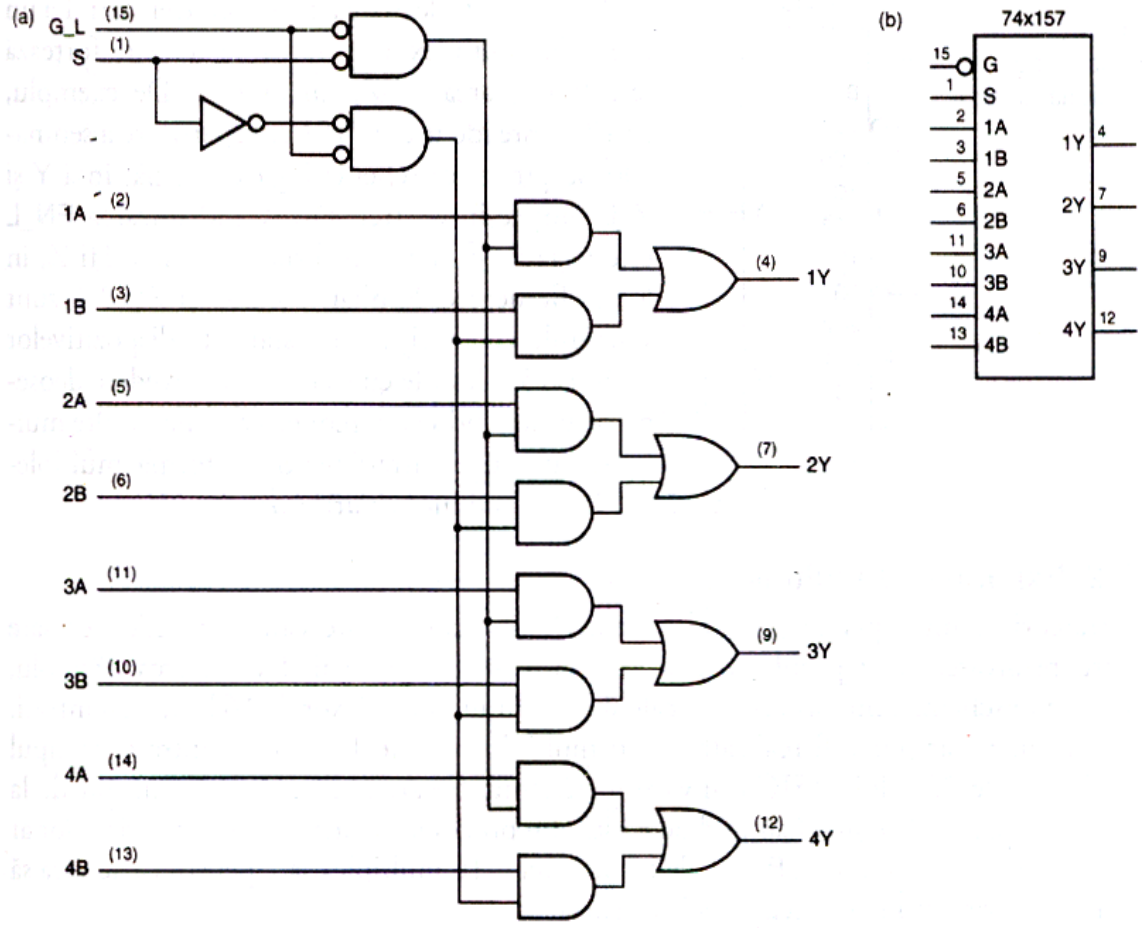


Figura 5-30 Multiplexorul de 4 biţi, cu 2 intrări, 74x157: (a) schema logică, inclusiv numerotarea pinilor pentru capsula standard DIP cu 16 pini; (b) simbolul logic tradiţional

Tabel 5-7 Tabelul de adevăr pentru multiplexorul de 4 biţi, cu 2 intrări, 74x157

Intrări		Ieşiri			
G_L	S	1Y	2Y	3Y	4Y
1	x	0	0	0	0
0	0	1A	2A	3A	4A
0	1	1B	2B	3B	4B

Un multiplexor intermediar, între 74x151 şi 74x157, este 74x153, care are 4 intrări a câte 2 biţi. Acest dispozitiv, al cărui simbol logic este prezentat în figura 5-31, are intrări de activare separate (**1G, 2G**) pentru fiecare bit. Funcţia lui se vede foarte clar din tabelul 5-8.

Tabel 5-8 Tabelul de adevăr pentru multiplexorul de 2 biţi, cu 4 intrări, 74x153

Intrări				Ieşiri	
1G_L	2G_L	B	A	1Y	2Y
0	0	0	0	1C0	2C0
0	0	0	1	1C1	2C1
0	0	1	0	1C2	2C2
0	0	1	1	1C3	2C3
0	1	0	0	1C0	0
0	1	0	1	1C1	0
0	1	1	0	1C2	0
0	1	1	1	1C3	0
1	0	0	0	0	2C0
1	0	0	1	0	2C1
1	0	1	0	0	2C2
1	0	1	1	0	2C3
1	1	x	x	0	0

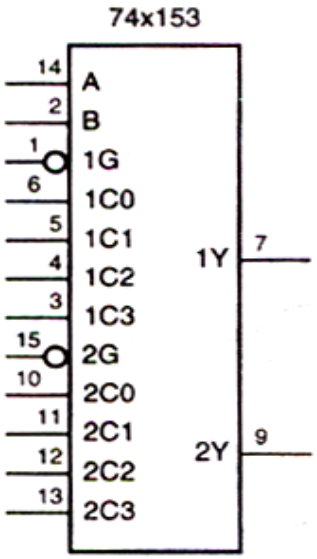


Figura 5-31 Simbolul logic tradiţional pentru multiplexorul 74x153

Unele multiplexoare au ieşiri cu trei stări. La un astfel de multiplexor, intrarea de activare forţează ieşirile în starea **Hi-Z**, nu în zero. Ieşirile cu trei stări se dovedesc deosebit de utile când se interconectează mai multe multiplexoare cu n intrări pentru a forma multiplexoare de capacitate mai mare.

5.6.2 Multiplexoare, demultiplexoare şi magistrale

Un multiplexor poate fi utilizat pentru a selecta una dintre *n* surse ce urmează să transmită date pe o magistrală. La celălalt capăt al magistralei poate fi folosit un *demultiplexor*, pentru a direcţiona datele de pe magistrale către una dintre cele *m* destinaţii posibile. O asemenea aplicaţie, în care se utilizează o magistrală de 1 bit, este descrisă în figura 5-32(a) prin analogia cu o schemă cu comutatoare, pe care am mai folosit-o. De fapt, în schemele bloc de circuite logice, multiplexoarele şi demultiplexoarele sunt adesea reprezentate prin

simbolurile trapezoidale din figura 5-32(b), pentru a sugera vizual că datele dintr-o sursă selectată dintre mai multe surse ajung pe magistrale şi apoi sunt orientate către o destinaţie selectată dintre mai multe destinaţii.

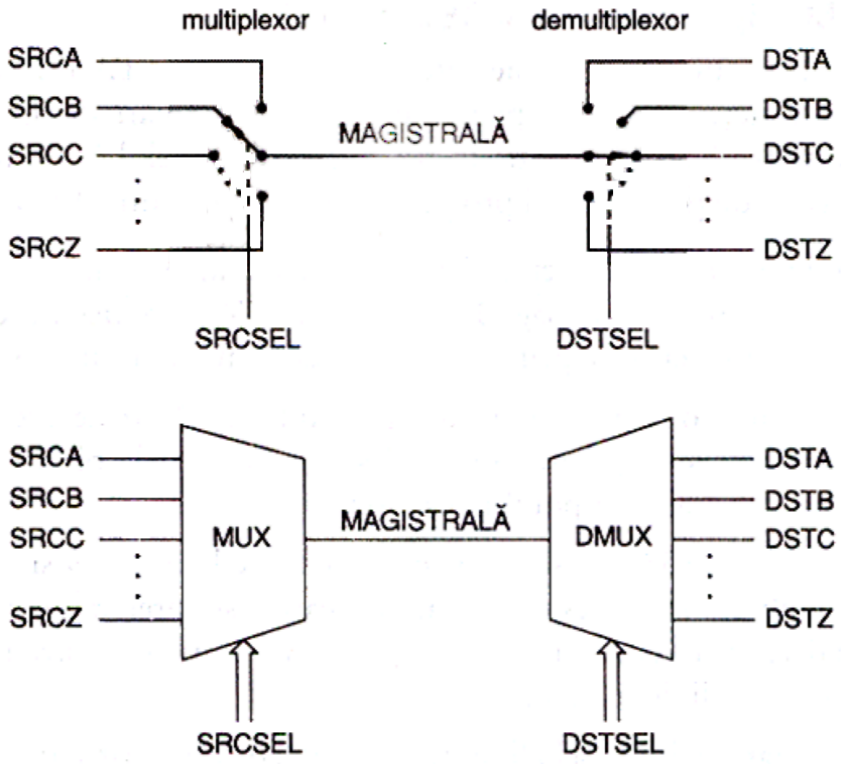


Figura 5-32 Un multiplexor ce comandă o magistrală şi un demultiplexor comandat de aceasta: (a) schema echivalentă cu comutatoare; (b) simbolurile folosite în schemele bloc

Funcţia pe care o realizează un demultiplexor este exact inversul funcţiei realizate de un multiplexor. De exemplu, un demultiplexor de 1 bit cu n ieşiri are o intrare de date şi s intrări de selectare a uneia dintre cele $n=2^s$ ieşiri de date. În funcţionare normală, toate ieşirile, cu excepţia celei selectate, sunt 0; la ieşirea selectată se regăseşte intrarea de date. Definiţia poate fi generalizată pentru un demultiplexor de b biţi cu n ieşiri; un astfel de dispozitiv are b intrări de date, iar cele s intrări de selectare ale sale selectează una dintre cele $n=2^s$ mulţimi de b ieşiri de date. Ca demultiplexor poate fi folosit un decodor binar cu o intrare de activare, ca în figura 5-33. Intrarea de activare a decodorului este conectată la linia de date, iar intrările sale de selectare stabilesc care dintre liniile de la ieşirea lui va fi comandată de bitul de date. Restul de linii de ieşire se neagă. Deci 74x139 poate fi utilizat ca demultiplexor de 2 biţi cu 4 ieşiri cu intrările şi ieşirile de date active în **LOW**, iar 74x138 poate servi ca demultiplexor de 1 bit cu 8 ieşiri.

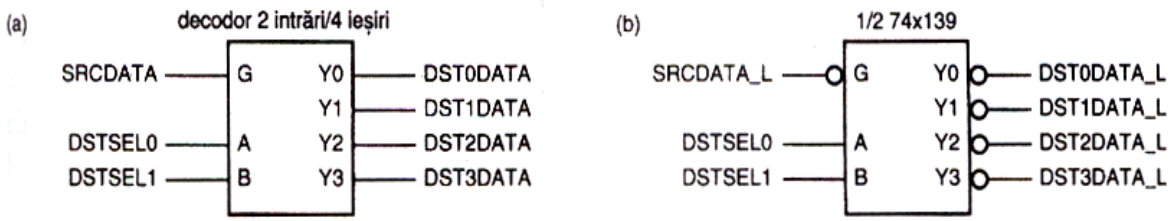


Figura 5-33 Folosirea unui decodor cu 2 intrări şi 4 ieşiri ca demultiplexor de 1 bit cu 4 ieşiri: (a) reprezentarea generică; (b) 74x139

5.7 Porţi OR exclusiv şi circuite de paritate

5.7.1 Porţi OR exclusiv şi NOR exclusiv

O poartă **OR exclusiv *(XOR)*** este o poartă cu două intrări a cărei ieşire este 1 dacă numai una dintre intrările sale este 1. Altfel spus, o poartă **XOR** generează ieşirea 1 dacă intrările sale sunt diferite. O poartă **NOR exclusiv *(XNOR)*** sau poartă de *echivalenţă* este exact opusul porţii **XOR**: ea generează ieşirea 1 dacă intrările sale sunt identice. Tabelul de adevăr al acestor funcţii apare în tabelul 5-9. Operaţia **XOR** este reprezentată uneori prin simbolul „⊕”, adică:

$$X \oplus Y = X' \cdot Y + X \cdot Y'$$

Tabel 5-9 Tabelul de adevăr pentru funcţiile XOR şi XNOR

X	Y	X ⊕Y (XOR)	(X ⊕Y)' (XNOR)
0	0	0	1
0	1	1	0
1	0	1	0
1	2	0	1

Deşi OR **exclusiv** nu este una dintre funcţiile de bază ale algebrei de comutaţie, în practică, porţile **XOR** discrete se utilizează destul de frecvent. În majoritatea tehnologiilor de comutaţie, funcţia **XOR** nu poate fi realizată direct; pentru obţinerea ei se folosesc scheme cu mai multe porţi, ca în figura 5-34.

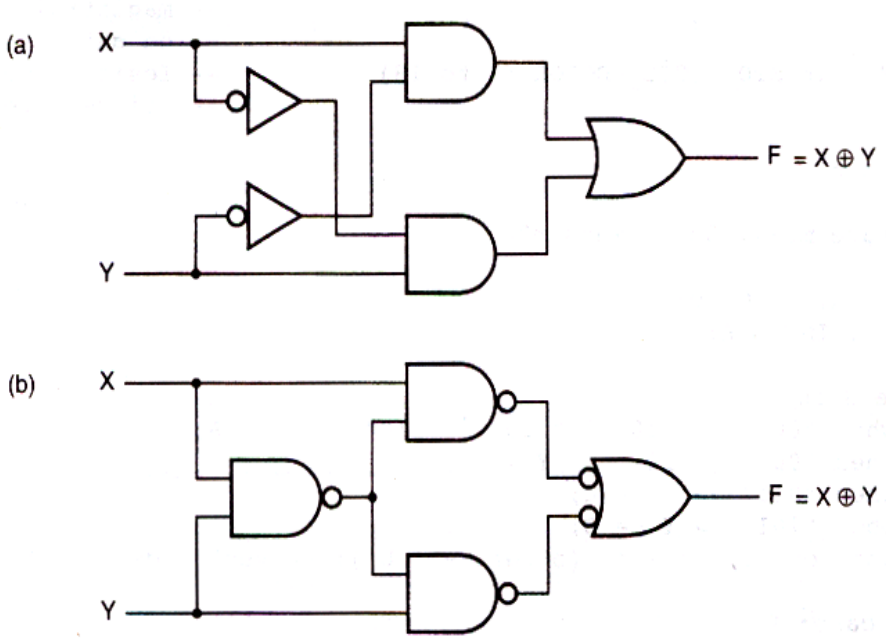


Figura 5-34 Scheme cu mai multe porţi, pentru realizarea funcţiei **XOR** cu două intrări: (a) cu **AND-OR**; (b) cu **NAND**

Simbolurile logice pentru funcţiile **XOR** şi **XNOR** sunt prezentate în figura 5-35. Pentru fiecare dintre aceste funcţii există câte patru simboluri echivalente. Toate variantele prezentate sunt consecinţa unei reguli simple: oricare două semnale (de intrare sau de ieşire) ale unei porţi **XOR** sau **XNOR** pot fi complementate fără ca funcţia logică obişnuită să se modifice.

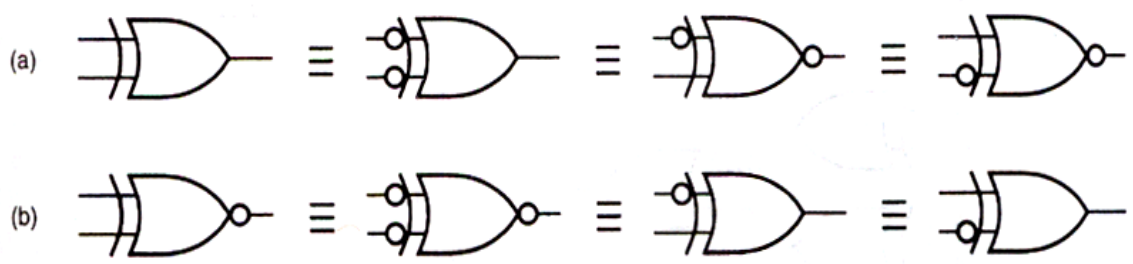


Figura 5-35 Simbolurile echivalente pentru: (a) porţi XOR; (b) pentru porţi XNOR

5.7.2 Circuite de paritate

După cum se vede în figura 5-36(a), *n* porţi **XOR** pot fi conectate în cascadă pentru a forma un circuit cu *n+1* intrări şi o singură ieşire. Un asemenea circuit este numit *circuit de imparitate* deoarece ieşirea lui este 1 dacă numărul de intrări 1 este impar. Circuitul din 5-36(b) este tot un circuit de imparitate, dar este mai rapid, deoarece porţile sale sunt dispuse într-o structură ramificată. Dacă se inversează ieşirea oricăruia dintre aceste circuite, se obţine un *circuit de paritate*, care are ieşirea 1 când numărul de intrări 1 este par.

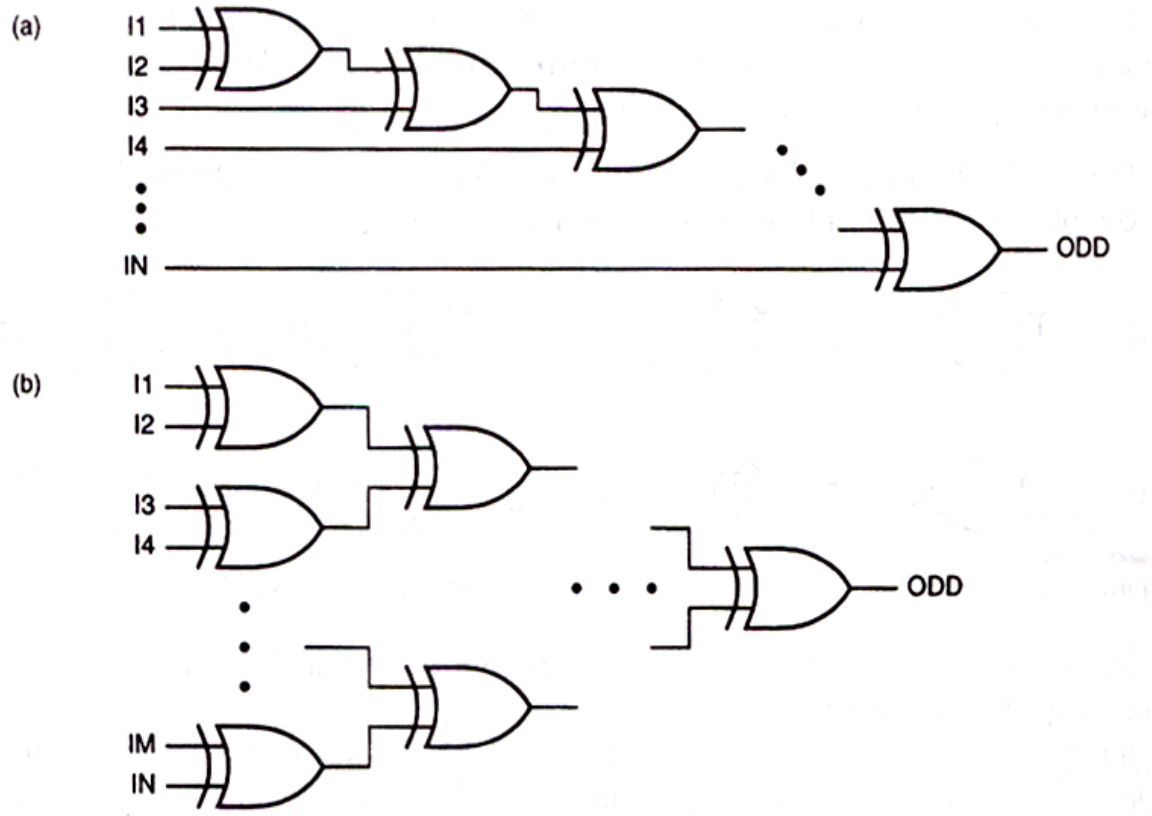


Figura 5-36 Conectarea în cascadă a porţilor XOR: (a) conexiuni înlănţuite; (b) structură ramificată

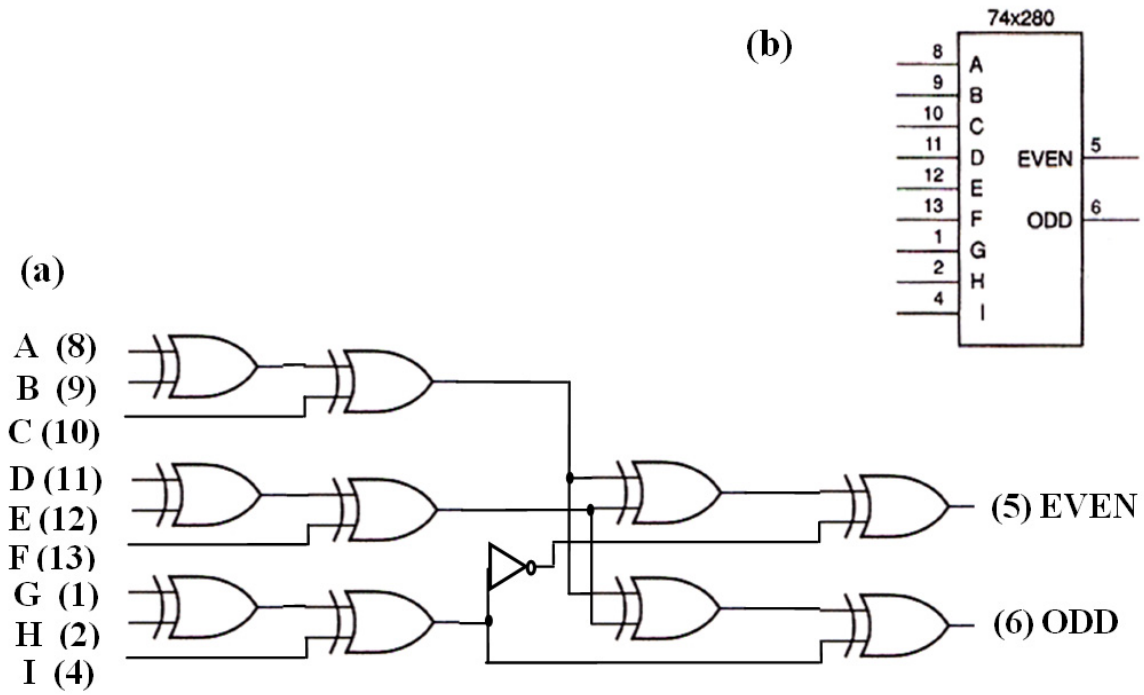


Figura 5-37 Generatorul de paritate pară/impară de 9 biţi 74x280: (a) schema logică, inclusiv numerotarea pinilor pentru capsula DIP cu 16 pini; (b) simbolul logic tradiţional

5.7.3 Generatorul de paritate de 9 biţi 74x280

În loc să construim un circuit de paritate de mai mulţi biţi cu porţi **XOR** discrete, este mai economic să realizăm toate porţile **XOR** în aceeaşi capsule MSI, lăsând accesibile la pinii externi doar intrările şi ieşirile primare. Generatorul de paritate de 9 biţi 74x280, din figura 5-37, este un astfel de dispozitiv. El are nouă intrări şi două ieşiri ce arată dacă numărul de intrări 1 este par sau impar.

5.8 Comparatoare

Compararea a două cuvinte binare pentru a afla dacă sunt egale este o operaţie mult utilizată de sistemele de calcul şi de interfeţele dispozitivelor. Un circuit ce compară două cuvinte binare şi indică egalitatea acestora se numeşte *comparator*. Unele comparatoare interpretează cuvintele de intrare ca numere precedate sau nu de semn şi indică şi relaţia de ordine dintre cuvinte (mai mare sau mai mic). Asemenea dispozitive sunt numite adesea *comparatoare de amplitudine.*

5.8.1 Structura de comparator

Porţile **OR exclusiv** şi **NOR exclusiv** pot fi considerate comparatoare de 1 bit. În figura 5-38(a) este ilustrată interpretarea unei porţi XOR 74x86 drept comparator de 1 bit. Ieşirea activă în **HIGH**, notată **DIFF**, este confirmată când intrările sunt diferite. Ieşirile a patru porţi **XOR** sunt aplicate la intrările unor porţi **OR** pentru a forma comparatorul de 4 biţi din figura 5-38(b). Ieşirea **DIFF** este confirmată dacă oricare dintre perechile de biţi de intrare conţine biţi diferiţi.

Conectând un număr suficient de mare de porţi **XOR** şi porţi **OR** de dimensiuni suficient de mari, se pot construi comparatoare cu orice număr de biţi de intrare.

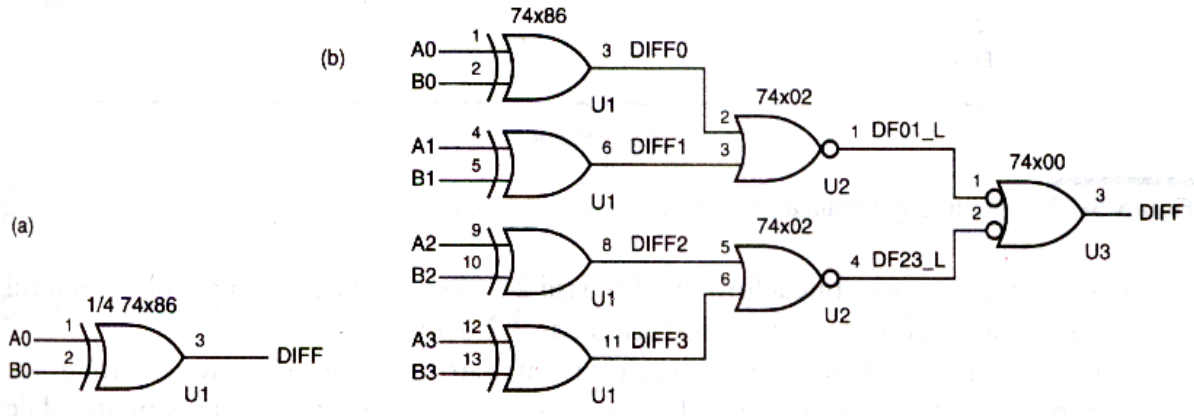


Figura 5-38 Comparatoare cu 74x86: (a) comparator de 1 bit; (b) comparator de 4 biţi

5.9 Sumatoare şi circuite de scădere

Adunarea este operaţia aritmetică efectuată cel mai frecvent în sistemele digitale. *Sumatoarele* combină doi operanzi aritmetici conform regulilor de adunare prezentate în capitolul 2. Sumatoarele pot efectua şi scăderi ca adunări între descăzut şi scăzătorul complementat (negat), dar se pot construi şi *circuite de scădere* care efectuează scăderile direct.

5.9.1 Semisumatoare şi sumatoare complete

Cel mai simplu sumator, denumit *semisumator*, adună doi operanzi de un bit, **X** şi **Y**, rezultatul fiind o sumă de doi biţi. Suma poate lua valori de la 0 la 2, necesitând doi biţi pentru a fi exprimată. Bitul de ordin inferior al sumei poate fi numit **HS** *(half sum – semisumal)* iar bitul de ordin superior poate fi numit **CO** *(carry out - transport către exterior).* Pentru **HS** şi **CO** se pot scrie următoarele formule:

$$\mathbf{HS=X\oplus Y=X'\cdot Y+X\cdot Y'}$$
$$\mathbf{CO=X\cdot Y}$$

Pentru a aduna operanzi de mai mulţi biţi trebuie să asigurăm efectuarea transportului între poziţiile biţilor. Blocul structural ce efectuează această operaţie se numeşte *sumator complet.* Pe lângă intrările **X** şi **Y**, ai căror biţi urmează a fi adunaţi, un sumator complet este prevăzut cu o intrare pentru bitul de transport, **CIN**. Suma celor trei valori de intrare poate fi cuprinsă între 0 şi 3, putând fi exprimată tot prin doi biţi de ieşire, **S** şi **COUT**, ale căror formule sunt:

$$\mathbf{S=X\oplus Y\oplus CIN=X\cdot Y'\cdot CIN'+X'\cdot Y\cdot CIN'+X'\cdot Y'\cdot CIN+X\cdot Y\cdot CIN}$$
$$\mathbf{COUT=X\cdot Y+X\cdot CIN+Y\cdot CIN}$$

Aici, **S** este 1 dacă există un număr impar de intrări 1, iar **COUT** este 1 dacă două sau mai multe intrări sunt 1.

Un circuit care poate realiza funcţiile descrise prin formulele sumatorului complet este cel din figura 5-39(a). Simbolul său logic este prezentat în 5-39(b). Uneori, simbolul folosit este cel din 5-39(c), pentru ca desenele ce conţin sumatoare complete, conectate în cascadă, să poată fi desenate mai „aerisit”.

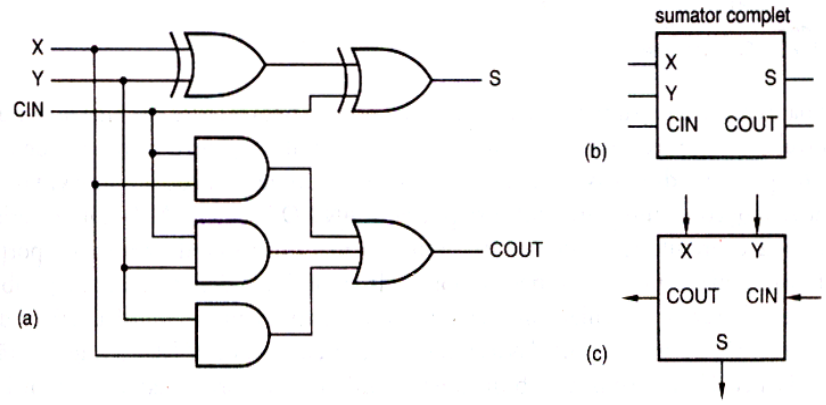


Figura 5-39 Sumator complet: (a) schema circuitului la nivel de porţi; (b) simbolul logic; (c) variantă de simbol logic adecvată conectării în cascadă

5.9.2 Sumatoare pieptene

Două cuvinte binare a câte *n* biţi pot fi adunate utilizând un *sumator pieptene*, adică un circuit format din *n* etaje de sumator complet conectate în cascadă, fiecare dintre ele prelucrând câte un bit. În figura 5-40 apare circuitul unui sumator pieptene de 4 biţi. Intrarea de transport către bitul cel mai puţin semnificativ este, în mod normal, fixată la 0, iar ieşirea de transport a fiecărui sumator complet este conectată la intrarea de transport a următorului sumator complet mai semnificativ.

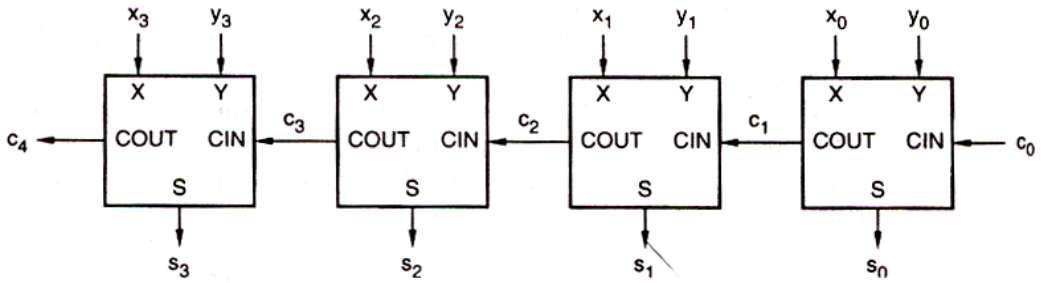


Figura 5-40 Sumator pieptene de 4 biţi

Sumatoarele complete sunt lente din cauză că, în cazul cel mai defavorabil, transportul trebuie să se propage de la sumatorul complet cel mai puţin semnificativ până la cel mai semnificativ. O astfel de situaţie se întâlneşte, de exemplu, în cazul în care unul dintre termeni este 11...11, iar celălalt este 00...01.

5.9.3 Circuite de scădere

Un *circuit de* scădere *complet* prelucrează un bit conform algoritmului de scădere în binar, biţii de intrare fiind **X** (descăzutul), **Y** (scăzătorul) şi **BIN** (împrumutul din exterior), iar biţii de ieşire fiind **D** (diferenţa) şi **BOUT**

(împrumutul către exterior). Putem scrie următoarele ecuaţii logice corespunzătoare tablei scăderii în binar:

D=X⊕Y⊕BIN
BOUT=X'·Y+X'·BIN+Y·BIN

Aceste formule seamănă foarte mult cu cele corespunzătoare sumatorului complet. Putem astfel construi un circuit de scădere complet pe baza unui sumator complet, ca în figura 5-41. Pentru a nu avea probleme, am dat circuitului sumatorului complet din 5-41(a) un nume fictiv, „74x999". Aşa cum se vede în 5-41(c), putem interpreta funcţia aceluiaşi circuit fizic ca scădere completă, atribuindu-i un alt simbol, cu semnalele de împrumut din exterior, către exterior şi scăzător active în **LOW**.

Deci pentru a construi un circuit pieptene de scădere a doi operanzi de n biţi, activi în **HIGH**, putem utiliza n dispozitive „74x999" şi inversoare, ca în figura 5-40(d). Observaţi că, în cazul scăderii, intrarea de împrumut de la nivelul bitului celui mai puţin semnificativ trebuie negată (adică nu există împrumut), ceea ce, pentru o intrare activă în **LOW**, înseamnă că, fizic, pinul trebuie să fie 1 sau **HIGH**. Situaţia este exact opusă faţă de adunare, unde acelaşi pin de intrare corespunde transportului din exterior, activ în **HIGH**, adică 0 sau **LOW**.

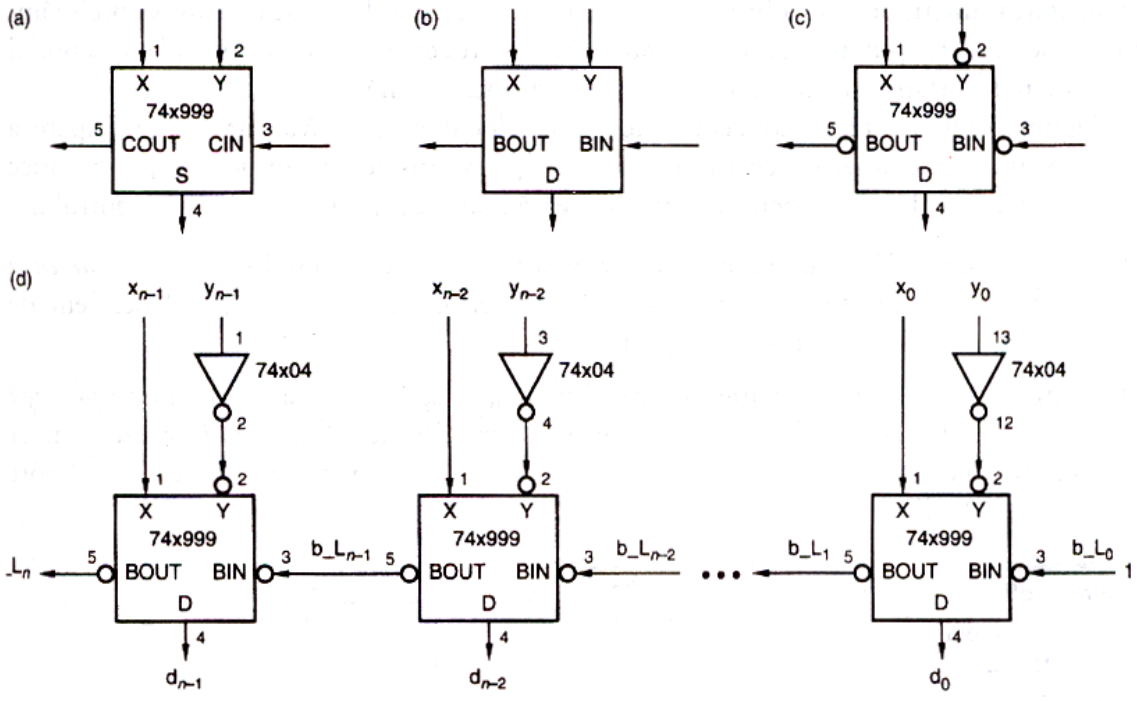


Figura 5-41 Transformarea circuitelor de adunare în circuite de scădere: (a) sumator complet; (b) circuit de scădere complet; (c) interpretarea circuitului din (a) drept circuit de scădere complet; (d) circuit de scădere pieptene

5.9.4 Sumatoare cu anticiparea transportului

Ecuaţia logică aferentă bitului *i* al sumei obţinute cu un sumator binar se poate scrie, de fapt, destul de simplu: $s_i = x_i \oplus y_i \oplus c_i$

Complexitatea creşte când exprimăm c_i în funcţie de $x_0 - x_{i-1}, y_0 - y_{i-1}$, iar după explicitarea funcţiilor **XOR** obţinem o întreagă încurcătură. Dar dacă dorim să preîntâmpinăm acest lucru, putem măcar să simplificăm aspectul expresiei c_i, cu ajutorul conceptului de anticipare a transportului.

Figura 5-42 ilustrează conceptul de bază. Blocul numit „matrice de anticipare a transportului" calculează c_i pentru un număr fix şi restrâns de niveluri logice, pentru orice valoare *i* rezonabilă. Două definiţii caracterizează matricea de anticipare a transportului:

- Pentru o anumită combinaţie de intrări x_i şi y_i se spune că etajul sumator *i generează* transport dacă produce un transport către exterior de 1 (c_{i+1}=1) independent de valoarea intrărilor $x_0 - x_{i-1}, y_0 - y_{i-1}$ şi c_0.
- Pentru o anumită combinaţie de intrări x_i şi y_i se spune că etajul sumator *i propagă* transporturile dacă produce un transport către exterior de 1 (c_{i+1}=1) în prezenţa unei combinaţii de intrare formată din $x_0 - x_{i-1}$, $y_0 - y_{i-1}$ şi c_0, pentru care apare un transport de 1 (c_{i+1}=1) dinspre exterior.

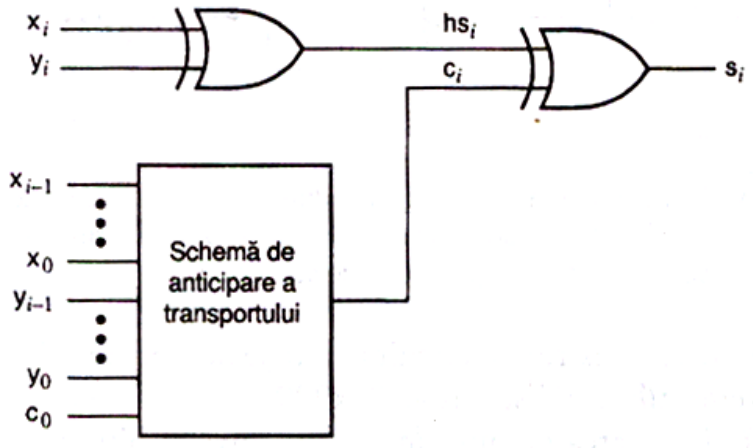


Figura 5-42 Structura unui etaj de sumator cu anticipare a transportului

În conformitate cu definiţiile de mai sus, putem scrie ecuaţiile logice pentru un semnal ce generează transport, g_i, şi pentru un semnal ce propagă transporturile, p_i, corespunzătoare fiecărui etaj al unui sumator cu anticipare a transportului:

$$g_i = x_i \cdot y_i$$
$$p_i = x_i + y_i$$

Înseamnă că un etaj generează necondiţionat un transport dacă ambii biţi ai operanzilor sunt 1 şi că propagă transporturile dacă minimum unul dintre biţii operanzilor este 1. Acum putem exprima semnalul de la ieşirea de transport a unui etaj în funcţie de semnalele de generare şi propagare:

$$c_{i+1}=g_i + p_i \cdot c_i$$

În figura 5-43 este prezentată schema unui circuit sumator pe 4 biţi. Observaţi că al doilea nivel este realizat cu semnale active în **LOW**, deoarece porţile inversoare sunt în general mai rapide decât cele neiversoare.

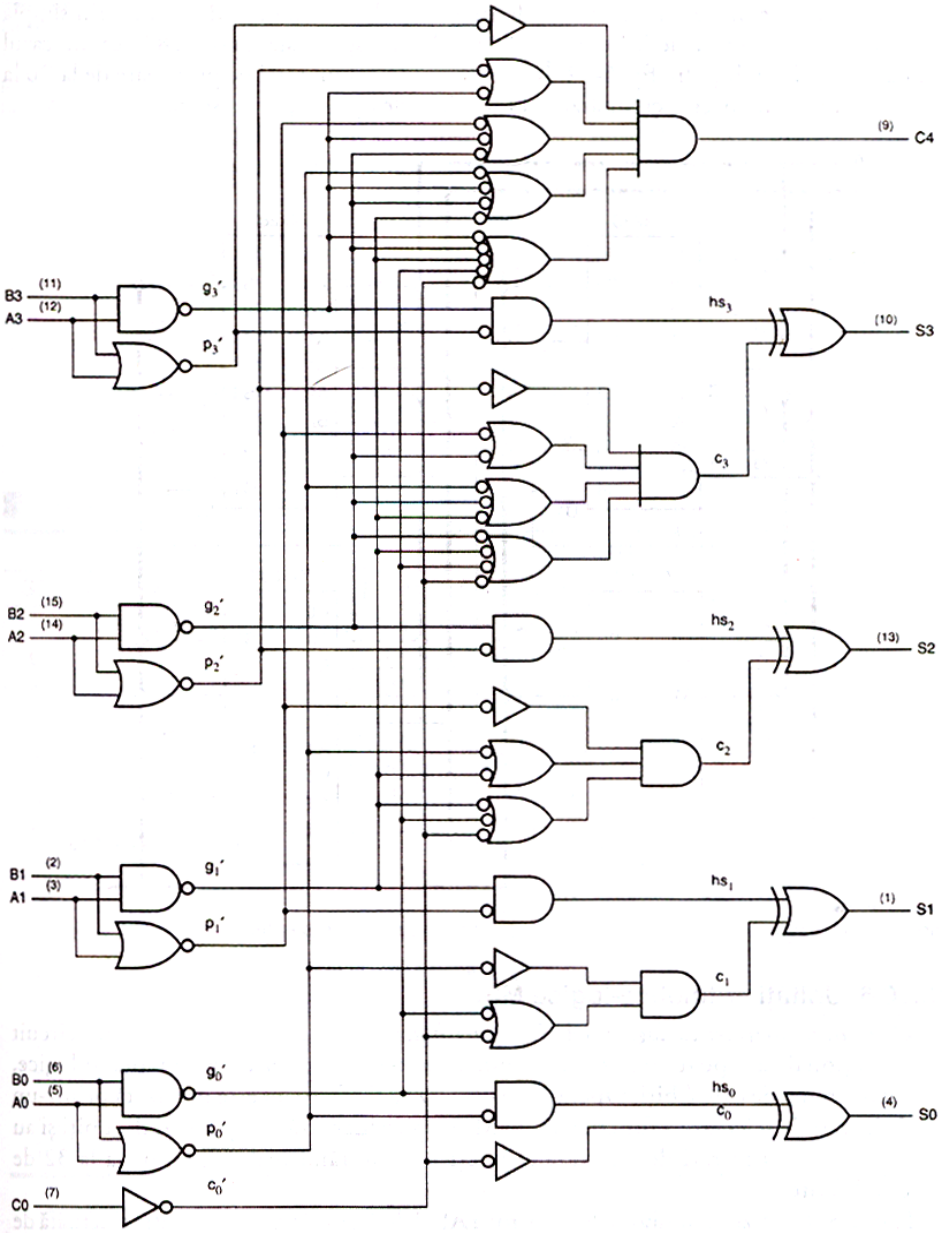


Figura 5-43 Schema logică a unui sumator binar pe 4 biţi

6. Circuite logice secvențiale. Bistabili

Circuitele logice sunt de două feluri: „combinaţionale” şi „secvenţiale”. Un circuit logic *combinaţional* este acela ale cărui ieşiri depind doar de intrările sale curente. Rotatorul pentru selectarea canalelor de la modelele vechi de televizoare seamănă cu circuitele combinaţionale, „ieşirea” lui selectând un canal numai în funcţie de „intrarea” lui curentă, poziţia butonului.

Circuitul logic *secvenţial* este acel circuit ale cărui ieşiri nu depind numai de intrările sale curente, ci şi de succesiunea anterioară a intrărilor, posibil nedeterminată în timp. Circuitul comandat de butoanele cu revenire pentru comutarea pe canalul următor sau pe cel precedent ale unui televizor sau videocasetofon este un circuit secvenţial, întrucât canalul selectat depinde de succesiunea anterioară de apăsări pe butoanele de comutare a canalelor cu câte unul înainte/înapoi, cel puţin din momentul în care v-aţi aşezat în faţa televizorului, acum 10 ore, şi poate chiar de când aţi conectat prima oară aparatul la priza din perete.

Deci este supărător şi adesea imposibil să descrieţi comportarea unui circuit secvenţial prin intermediul unui tabel ce conţine ieşirile în funcţie de o succesiune de intrări care a fost recepţionată înaintea momentului de timp curent. Pentru a şti în ce direcţie mergeţi, trebuie să ştiţi unde vă aflaţi. În cazul selectorului de canale TV, este imposibil de aflat care este canalul curent selectat, privind numai la secvenţa de apăsări anterioare pe butoanele înainte/înapoi, indiferent dacă studiem 10 sau 1000 de apăsări anterioare ale butoanelor. Este necesară încă o informaţie –„starea” curentă a selectorului de canale.

Starea unui circuit secvenţial este un ansamblu de *variabile de stare* ale căror valori conţin, în orice moment, toate informaţiile despre perioada anterioară necesare pentru cunoaşterea comportării viitoare a circuitului.

În exemplul cu selectorul de canale, numărul canalului curent este starea curentă. În interiorul televizorului, acest număr poate fi stocat sub forma a şapte variabile binare de stare, cu care se poate reprezenta un număr zecimal cuprins între 0 şi 127. Cunoscând starea curentă (numărul canalului), putem spune întotdeauna care va fi starea următoare, în funcţie de intrări (apăsări pe butoanele de comutare înainte/înapoi cu câte un canal). În acest exemplu, una dintre ieşirile evidente ale circuitului secvenţial este codarea stării înseşi, adică afişarea numărului canalului. Alte ieşiri, din interiorul televizorului, pot fi funcţii combinaţionale exclusiv de stare (de ex. selectarea acordului pe VHF/UHF/cablu) sau atât de stare, cât şi de intrare (de ex., la unele modele de

TV, stingerea aparatului când starea curentă este 0 şi se apasă butonul "înapoi").

Nu este necesar ca variabilele de stare să aibă o semnificaţie fizică directă şi, în multe cazuri, există mai multe posibilităţi de a alege variabilele pentru a descrie un anumit circuit secvenţial. De exemplu, la selectorul de canale TV, starea poate fi memorată sub forma a trei cifre BCD sau a 12 biţi, multe dintre combinaţiile de biţi (din cele 4096 posibile) nefiind utilizate.

În circuitele logice digitale, variabilele de stare sunt valori binare corespunzătoare anumitor semnale logice din circuit. Un circuit cu *n* variabile de stare binare are 2^n stări posibile. Oricât de mare ar fi numărul 2^n, el este întotdeauna finit, şi niciodată infinit, de aceea circuitele secvenţiale sunt denumite uneori *automate de stări finite.*

La majoritatea circuitelor secvenţiale, schimbările de stare au loc la momente stabilite de un semnal de *tact* asincron. Figura 6-1 prezintă diagramele temporale şi denumirile caracteristice unui semnal de tact tipic. Prin convenţie, un semnal de tact este activ în HIGH dacă modificarea stărilor are loc pe frontul ascendent al semnalului de tact, adică atunci când semnalul de tact trece în starea HIGH, şi este activ în LOW în cazul complementar. *Perioada de tact* este intervalul de timp dintre două tranziţii succesive în acelaşi sens, iar *frecvenţa de tact* este inversul perioadei. Primul front sau impuls dintr-o perioadă de tact sau, uneori, însăşi perioada sunt denumite *impuls de tact. Factorul de comandă* reprezintă procentual intervalul de timp din perioada în care semnalul de tact are nivelul de confirmare. Sistemele digitale tipice, de la ceasurile digitale şi pană la supercalculatoare, folosesc un oscilator cu cristal de cuarţ pentru a genera semnalul de tact asincron. Frecvenţele de tact pot fi cuprinse între 32,768 KHz (pentru ceasuri) şi 500 MHz (pentru microprocesoarele CMOS RISC cu un ciclu de 2 ns); sistemele „tipice" cu componente TTL şi CMOS au frecvenţe de tact din domeniul 5 ... 150 MHz.

În capitolul de faţă vom discuta despre două tipuri de circuite secvenţiale care se regăsesc în majoritatea schemelor practice cu componente discrete. *Circuitul secvenţial cu reacţie* utilizează porţi obişnuite şi bucle de reacţie pentru a conferi unui circuit logic capacitate de memorare, creându-se astfel blocuri structurale de circuite secvenţiale, cum sunt circuitele latch şi CBB, utilizate în scheme de complexitate mai mare. *Automatele de stări sincronizate cu semnal de tact* utilizează aceste blocuri structurale, în special CBB de tip **D** active pe fronturi, pentru a crea circuite ale căror intrări sunt explorate şi ale căror ieşiri comută sincron cu un semnal de tact de comandă. Alte tipuri de circuite secvenţiale, cum sunt cele de mod fundamental general, de mod multi-impuls şi multi-fază, îşi găsesc uneori aplicaţii în sistemele de înaltă performanţă şi VLSI.

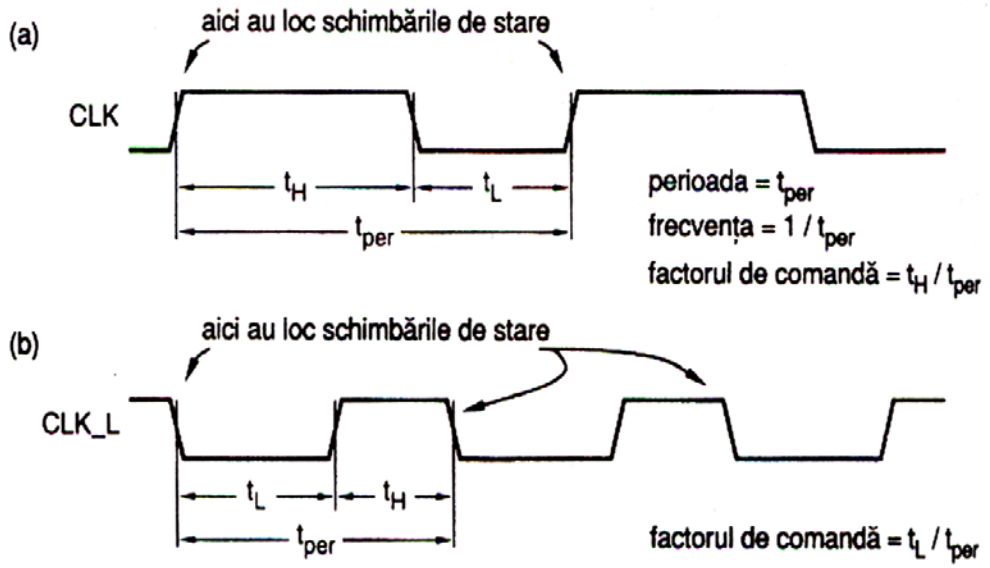


Figura 6-1 Semnale de tact: (a) active în HIGH; (b) active în LOW

6.1 Elemente bistabile

Cel mai simplu circuit secvenţial constă dintr-o pereche de inversoare ce formează o buclă de reacţie, ca în fig. 6-2. *El* are *zero* intrări şi două ieşiri, **Q** şi **Q_L**.

6.1.1 Analiza digitală

Circuitul din fig. *7-2* este denumit frecvent *bistabil,* deoarece o analiză strict digitală arată că el are două stări stabile. Când **Q** este HIGH, inversorul din partea de jos are intrarea HIGH şi ieşirea LOW, forţând în HIGH ieşirea inversorului din partea de sus, aşa cum am şi considerat în ipoteză. Dar când **Q** este LOW, inversorul de jos are intrarea LOW şi ieşirea HIGH, ceea ce forţează **Q** în LOW - o altă stare stabilă. Pentru a descrie starea circuitului putem folosi o singură variabilă de stare - starea semnalului **Q**; sunt posibile două stări, **Q** = 1 şi **Q** = 0.

Elementul bistabil este aşa de simplu că nici nu are intrări, deci nu există nici o cale de a-i comanda sau modifica stările. La alimentarea iniţială a circuitului, acesta trece aleator în una dintre cele două stări şi rămâne aşa la infinit.

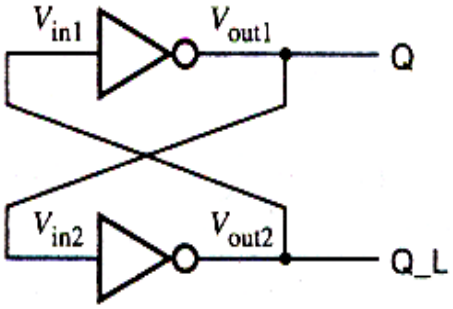


Figura 6-2 Pereche de inversoare ce formează un element bistabil

6.2 Circuite latch şi circuite basculante bistabile

Circuitele latch şi circuitele basculante bistabile (CBB) sunt blocurile structurale de bază ale majorităţii circuitelor secvenţiale. Sistemele digitale tipice folosesc astfel de circuite încorporate în dispozitive clasificate după funcţiile pe care le realizează, preambalate în capsule de circuite integrate standardizate. În mediile de proiectare de ASIC, circuitele latch şi CBB formează celule predefinite tipice, cu parametrii stabiliţi de furnizorul de ASIC. Însă, în orice CI sau ASIC standardizat, fiecare celulă latch sau CBB este proiectată de obicei ca un circuit secvenţial cu reacţie, din porţi logice separate şi bucle de reacţie.

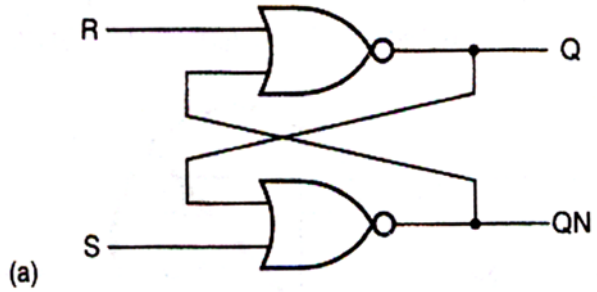
Toţi proiectanţii de circuite digitale denumesc prin cuvintele *circuit basculant bistabil (flip*-flop*),* prescurtat *CBB,* un dispozitiv secvenţial care, în mod normal, preia eşantioane din semnalele de intrare şi îşi modifică semnalele de ieşire numai la momente determinate de un semnal de tact. De asemenea, cei mai mulţi proiectanţi de circuite digitale folosesc denumirea *latch* pentru un dispozitiv secvenţial care îşi supraveghează intrările permanent şi îşi schimbă ieşirile în orice moment, fără a depinde de un semnal de tact.

6.2.1 Circuite latch S-R

În fig. 6-3 este prezentat un circuit *latch* **S-R** *(set-reset)* cu porţi **NOR**. Circuitul are două intrări, **S** şi **R**, şi două ieşiri, notate cu **Q** şi **QN**, **QN** fiind, în mod normal, complementul lui **Q**. Semnalul **QN** mai este uneori notat cu **Q'** sau **Q_L**.

Dacă atât **S**, cât şi **R** sunt 0, circuitul se comportă ca un element bistabil - apare o buclă de reacţie care îl menţine într-una dintre cele două stări logice, **Q** = 0 sau **Q** = 1. Aşa cum observaţi în fig. 6-3(b), este necesar ca fie **S**, fie **R** să fie confirmate pentru ca bucla de reacţie să fie forţată să treacă în starea dorită. **S** *fixează (set)* sau *prefixează (preset)* ieşirea **Q** în 1; **R** *readuce în 0 (reset) sau,şterge (clear)* ieşirea **Q**. După negarea uneia dintre intrările **S** şi **R**, circuitul

latch rămâne în starea în care a fost adus. Figura 6-4(a) prezintă comportarea funcţională a unui circuit latch **S-R** când i se aplică la intrare o secvenţă de impulsuri uzuală. Săgeţile arată cauzalitatea - corespondenţa dintre tranziţiile de la intrare şi cele produse la ieşire.



S	R	Q	QN
0	0	ultimul Q	ultimul QN
0	1	0	1
1	0	1	0
1	1	0	0

(b)

Figura 6-3 Circuit latch S-R: (a) schema cu porţi NOR a circuitului; (b) tabelul funcţiei

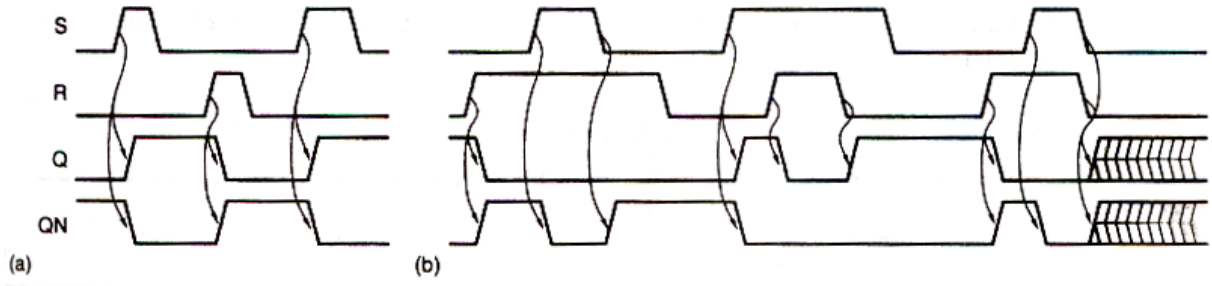


Figura 6-4 Funcţionare tipică a unui circuit latch S-R: (a) intrări "normale"; (b) S şi R confirmate simultan

În majoritatea aplicaţiilor cu circuite latch **S-R**, ieşirea **QN** (sau **Q'**) este întotdeauna complementul ieşirii **Q**. Totuşi, notaţia **Q'** nu este foarte corectă, deoarece există un caz când această ieşire nu este complementul lui **Q**. Dacă atât **S**, cât şi **R** sunt 1, aşa cum se întâmplă în câteva puncte din fig. 6-4(b), atunci ambele ieşiri sunt forţate să treacă în 0. Când negăm oricare intrare, ieşirile revin la funcţionarea complementară, ca de obicei. Dar dacă negăm simultan ambele intrări, starea următoare a circuitului devine imprevizibilă, acesta putând intra, practic, în oscilaţie sau în starea metastabilă. Metastabilitatea se mai poate manifesta daca pe **S** sau **R** se aplică un impuls 1 prea îngust.

În fig. 6-5 apar trei simboluri logice diferite pentru acelaşi circuit latch **S-R**. Simbolurile se deosebesc prin modul în care este tratată ieşirea complementară. Cronologic, simbolul (a) a fost primul utilizat, denumirea semnalului activ în LOW sau complementar fiind înscrisă în interiorul simbolului dreptunghiular al funcţiei. În proiectarea cu cerculeţe se preferă însă cea de-a doua formă, cu un cerculeţ inversor în exteriorul dreptunghiului. Ultimul simbol este, evident, incorect.

În fig. 6-6 sunt definiţi parametrii temporali ai circuitului latch **S-R**. *Timpul de propagare* este timpul necesar ca o tranziţie a unui semnal de intrare

să producă o tranziţie a unui semnal de ieşire. Un anumit circuit latch sau un CBB pot fi caracterizate de valori diferite ale timpului de propagare - câte o valoare pentru fiecare pereche de semnale de intrare şi ieşire. De asemenea, timpul de propagare poate avea valori diferite după cum tranziţia semnalului de ieşire se face de la LOW la HIGH sau de la HIGH la LOW. La un circuit latch **S-R**, o tranziţie LOW-HIGH a intrării **S** poate produce o tranziţie LOW-HIGH a ieşirii **Q**, deci timpul de propagare este $t_{pLH(SQ)}$, corespunzător tranziţiei 1 din figură. În mod asemănător, o tranziţie LOW-HIGH a intrării **R** poate produce o tranziţie HIGH-LOW a ieşirii 0, deci timpul de propagare este $t_{pHL(RQ)}$, corespunzător tranziţiei 2 din figură. În desen nu apar tranziţiile corespunzătoare ale ieşirii **QN**, timpii de propagare respectivi fiind $t_{PHL(SQN)}$ şi $t_{pLH(RQN)}$.

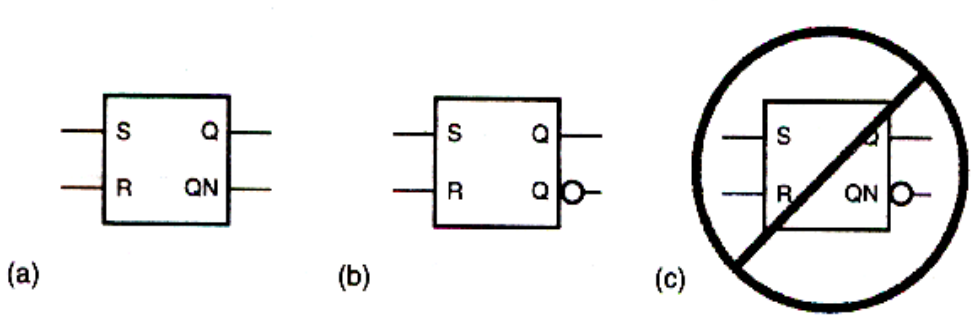


Figura 6-5 Simboluri pentru circuitul latch S-R: (a) fără cerculeţ; (b) preferat în proiectarea cu cerculeţe; (c) incorect din cauza dublei negaţii

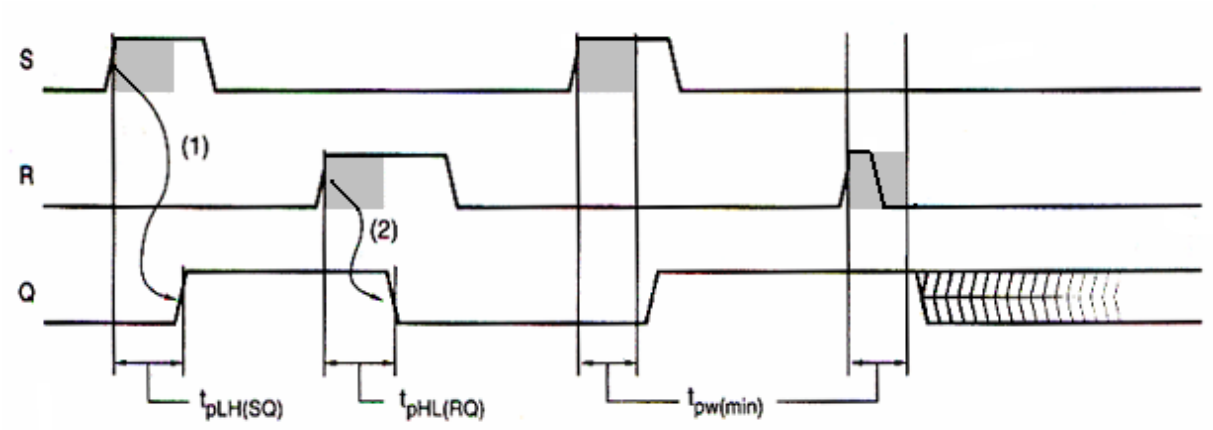


Figura 6-6 Parametri temporali ai unui circuit latch S-R

În cataloage se precizează, de obicei, *lăţimea minimă a impulsului* pentru intrările **S** şi **R**. Aşa cum vedeţi în fig. 6-6, circuitul poate să intre în starea metastabilă şi să se menţină aşa un interval de timp nedeterminat dacă pe **S** sau pe **R** se aplică un impuls de lăţime mai mică decât cea minimă, notată $t_{pw(min)}$. Circuitul poate fi scos categoric din starea metastabilă numai prin aplicarea pe **S** sau pe **R** a unui impuls de o lăţime egală sau mai mare ca lăţimea minimă.

6.2.2 Circuite latch

Un *circuit latch* **S'-R'** (citeşte "non **S** - non **R**"), cu intrările *set* şi *reset* active în LOW, se poate construi cu porţi **NAND**, ca în fig. 6-7(a). În familiile de circuite logice TTL şi CMOS, circuitele latch **S'-R'** sunt utilizate mai frecvent decât cele **S-R** deoarece porţile **NAND** sunt preferate porţilor **NOR**.

Aşa cum arată şi tabelul funcţiei din fig. 6-7(b), funcţionarea unui circuit latch **S'-R'** este asemănătoare cu cea a circuitului **S-R**, dar cu două diferenţe majore. În primul rând, **S'** şi **R'** sunt active în LOW, deci circuitul reţine în memorie starea sa anterioară când **S'** = **R'** = 1; intrările active în LOW sunt indicate clar în simbolul din (c). În al doilea rând, când **S**' şi **R'** se confirmă simultan, ambele ieşiri ale circuitului trec în 1, nu în 0, ca la circuitul **S-R**. Cu excepţia acestor deosebiri, circuitul **S'-R'** funcţionează la fel ca **S'-R'**, inclusiv în ceea ce priveşte problemele de temporizare şi de metastabilitate.

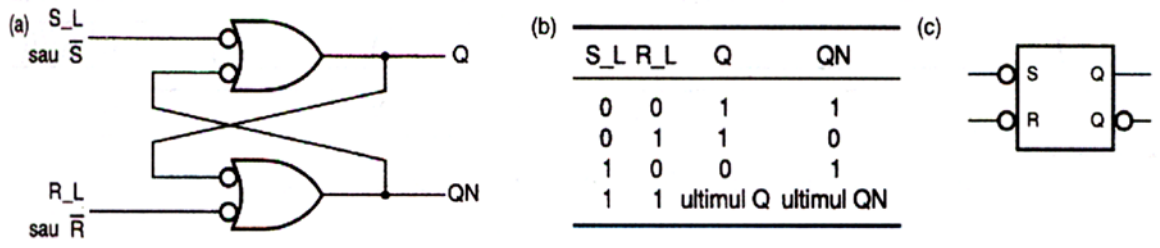


Figura 6-7 Circuit latch S'-R': (a) schema cu porţi NAND a circuitului; (b) tabelul funcţiei; (c) tabelul logic

6.2.3 Circuit latch S-R cu intrare de activare

Circuitele latch **S'-R'** sunt permanent sensibile la variaţiile intrărilor **S** şi **R**. Dar ele se pot modifica uşor astfel încât să prezinte această sensibilitate numai atunci când este confirmată o intrare de activare **C**. În fig. 6-8 observaţi un circuit latch **S-R** cu intrare de activare. Aşa cum arată tabelul funcţiei, circuitul se comportă ca un latch **S-R** când **C** = 1 şi îşi păstrează starea anterioară când **C** = 0. Comportarea acestui circuit la aplicarea unui set tipic de semnale de intrare este ilustrată in fig. 6-9. Dacă atât **S**, cât şi **R** sunt 1 atunci când **C** trece din 1 în 0, circuitul se comportă ca un latch **S-R** la care **S** şi **R** se neagă simultan, adică starea următoare este imprevizibilă şi ieşirea poate deveni metastabilă.

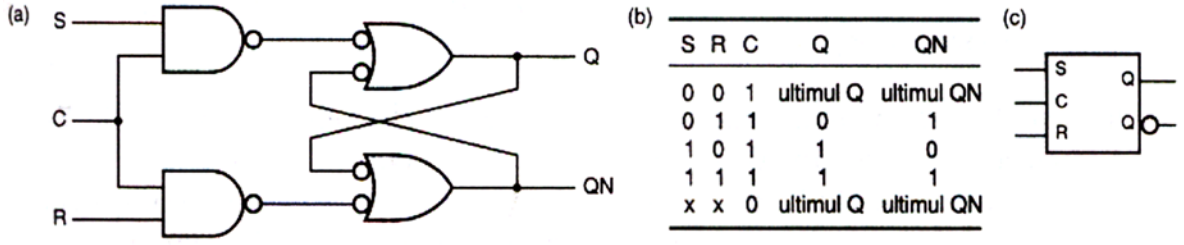


Figura 6-8 Circuit latch S-R cu intrare de activare: (a) schema cu porţi NAND a circuitului; (b) tabelul funcţiei; (c) simbolul logic

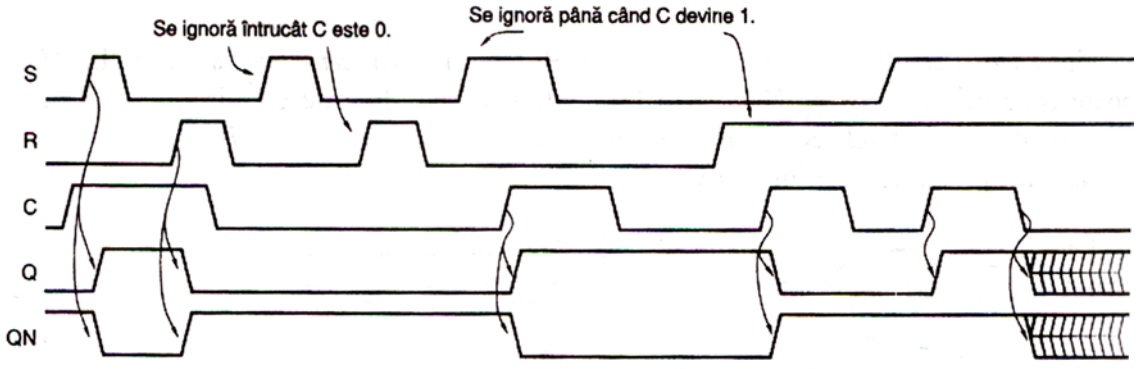


Figura 6-9 Funcţionarea tipică a unui circuit latch S-R cu intrare de activare

6.2.4 Circuit latch D

Circuitele latch **S-R** se întrebuinţează în aplicaţii de comandă, unde se pune frecvent problema activării unui indicator *(flag)* ca răspuns la apariţia anumitor condiţii şi a dezactivării acestuia atunci când condiţiile încetează să existe. Deci comandăm intrările *set* şi *reset* oarecum independent. Dar uneori avem nevoie de circuite latch pentru simpla stocare a unor biţi de informaţie, fiecare bit parvenindu-ne pe câte o linie de semnal şi trebuind să fie stocat undeva. În asemenea aplicaţii se poate folosi un *circuit latch **D***.

Figura 6-10 prezintă un circuit latch **D**. Schema lui logică este uşor de recunoscut deoarece este cea a unui latch **S-R** cu intrare de activare la care se adaugă un inversor, pentru a genera intrările **S** şi **R** dintr-un semnal de intrare unic, **D** (de date). În acest mod se elimină situaţiile supărătoare ce caracterizează circuitele latch **S-R**, când **S** şi **R** pot fi confirmate simultan. Intrarea de comandă a unui latch **D**, notată cu **C** în (c), mai este denumită uneori **ENABLE**, **CLK** sau **G**, iar în unele scheme de latch **D** este activă în LOW.

Un exemplu de comportare funcţională a circuitului latch **D** este prezentat în fig. 6-11. Când intrarea **C** este confirmată, ieşirea **Q** urmăreşte intrarea **D**. În acest caz, se spune despre latch că este "deschis", iar calea dintre intrarea **D** şi ieşirea **Q** este "transparentă", motiv pentru care circuitul este numit deseori *latch transparent.* Când intrarea **C** este negată, circuitul se "închide"; ieşirea **Q**

păstrează ultima sa valoare şi nu mai variază ca răspuns la **D**, atâta timp cât **C** rămâne negată.

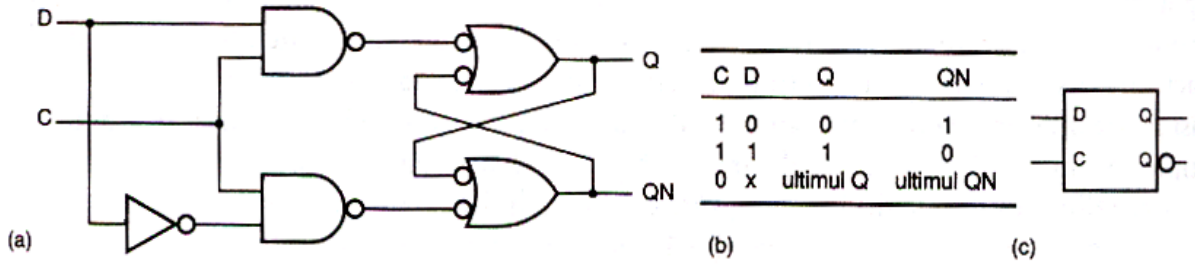


Figura 6-10 Circuit latch D: (a) schema cu porţi NAND a circuitului; (b) tabelul funcţiei; (c) simbolul logic

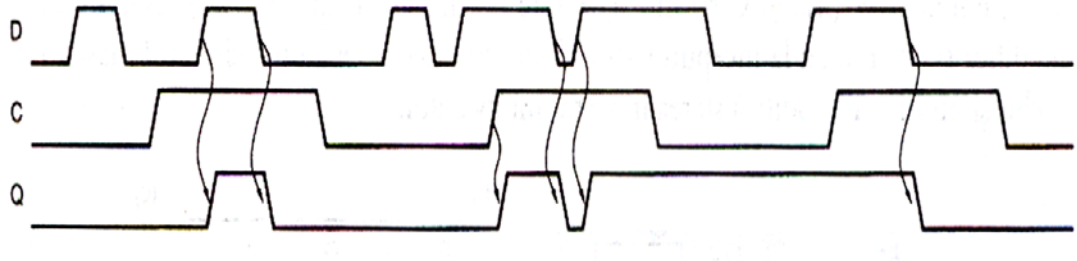


Figura 6-11 Comportarea funcţională a circuitului latch D cu diferite semnale de intrare

În fig. 6-12 este prezentată mai detaliat comportarea în timp a circuitului latch **D**. Apar patru parametri temporali diferiţi, aferenţi semnalelor ce se propagă de la intrările **C** sau **D** la ieşirea **Q**. De exemplu, în cazul tranziţiilor 1 şi 4, circuitul este "închis" iniţial, iar intrarea **D** este opusă faţă de ieşirea **Q**, deci atunci când **C** trece în 1, circuitul se "deschide" şi ieşirea 0 se modifică după un interval de timp $t_{pLHCCQ)}$ sau $t_{pHL(CQ\}}$. În cazul tranziţiilor 2 şi 3,intrarea **C** este deja 1 şi circuitul este deja deschis, deci **Q** urmăreşte în mod transparent variaţiile intrării **D**, cu întârzierile $t_{pH1,(DQ)}$ şi $t_pLH(DQ)$. Alţi patru parametri, care nu apar aici, precizează decalajele temporale aferente ieşirii **QN**.

Deşi circuitele latch **D** elimină problema ce apare la cele de tip **S-R** când **S** = **R** = 1, ele nu elimină şi problema metastabilităţii. Aşa cum se vede în fig. 6-12, există o fereastră de timp (haşurată) în zona frontului descendent al lui **C**, în care intrarea **D** nu trebuie să se modifice. Fereastra începe cu un interval de timp t_{setup} înainte de frontul descendent al lui ***C;*** t_{setup}, se numeşte *timp de pregătire (setup time).* Fereastra se termină după un interval de timp t_{hold} după frontul descendent; t_{hold} se numeşte *timp de menţinere (hold time).* Dacă **D** se modifică în orice moment al ferestrei, în timpul de pregătire şi în cel de menţinere, ieşirea circuitului latch devine imprevizibilă şi poate intra în metastabilitate, ca în cazul ultimului front descendent din desen.

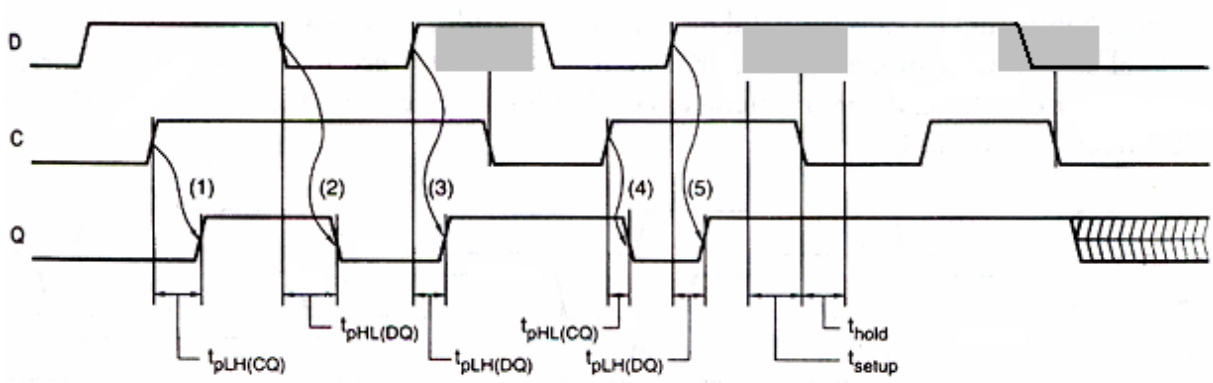


Figura 6-12 Parametri temporali ce caracterizează un circuit latch D

6.2.5 CBB de tip D, activ pe front

Un *CBB de tip **D**, activ pe frontul pozitiv* este format dintr-o pereche de circuite latch **D**, ca în fig. 6-13, iar circuitul astfel obţinut explorează intrarea **D** şi îşi modifică ieşirile **Q** şi **QN** numai pe frontul ascendent al unui semnal de comandă **CLK**. Primul circuit latch este denumit *master (stăpân);* este deschis şi urmăreşte intrarea când **CLK** este 0. Când **CLK** trece în 1, circuitul master se închide şi ieşirea lui este transferată celui de-al doilea circuit latch, numit *slave (sclav).* Circuitul slave este deschis atâta timp cât **CLK** este 1, dar îşi modifică starea numai la începutul acestui interval, deoarece atunci circuitul master este închis şi nu îşi mai modifică starea în restul intervalului.

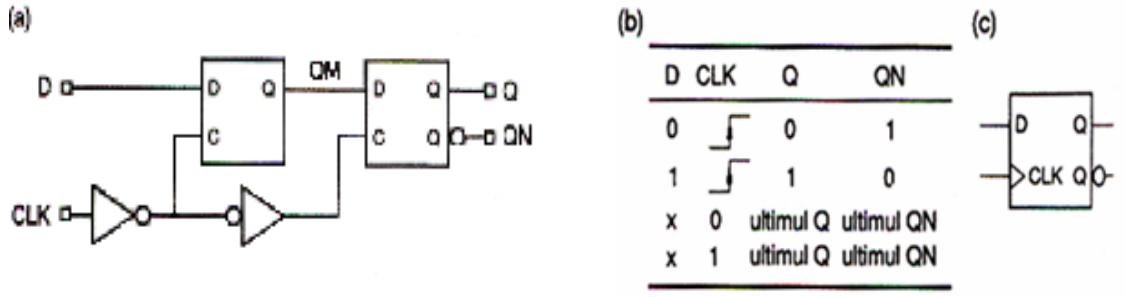


Figura 6-13 CBB de tip D activ pe frontul pozitiv: (a) schema cu circuite latch D; (b) tabelul funcţiei; (c) simbolul logic

Triunghiul de la intrarea **CLK** a CBB de tip **D** indică modul în care se comportă circuitul - activ pe fronturi - şi se numeşte *indicator al intrării dinamice.* În fig. 6-14 sunt prezentate câteva exemple de comportare funcţională a CBB pentru câteva tranziţii de intrare. Semnalul **QM** este semnalul de ieşire al circuitului latch master. Observaţi că acest semnal se modifică numai când **CLK** este 0. Când **CLK** trece în 1, valoarea de curent corespunzătoare lui **QM** este transferată în **Q**, **QM** fiind împiedicat să se modifice pană când **CLK** trece din nou în 0.

Figura 6-15 prezintă mai detaliat comportarea în timp a CBB de tip **D**. Toţi timpii de propagare se măsoară de la frontul ascendent al semnalului **CLK**, deoarece acesta este singurul eveniment care produce modificarea ieşirilor. Valorile timpilor de propagare pot fi diferite pentru tranziţiile de ieşire LOW-HIGH şi HIGH-LOW.

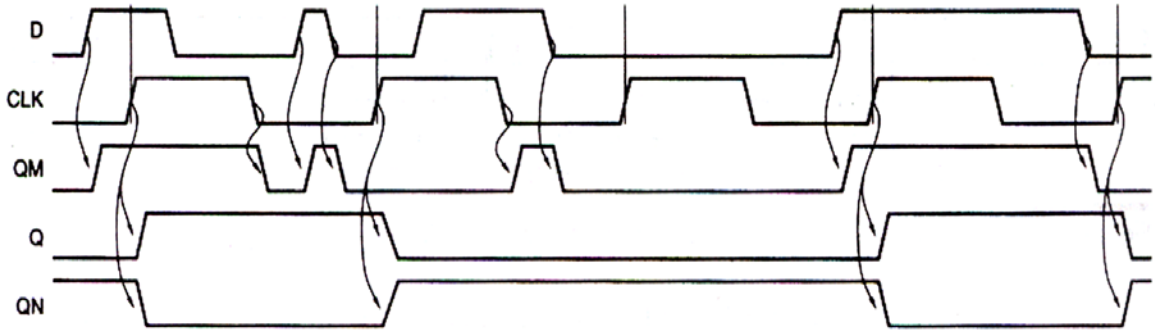


Figura 6-14 Comportarea funcţională a unui CBB de tip D, activ pe frontul pozitiv

Asemenea unui circuit latch **D**, CBB de tip **D**, activ pe fronturi este caracterizat de o fereastră temporală de pregătire şi menţinere, interval în care intrarea **D** nu trebuie să se modifice. Fereastra se află în zona frontului de activare al semnalului **CLK**, iar în fig. 6-15 este evidenţiată prin haşuri. Dacă timpii de pregătire şi de menţinere nu sunt respectaţi, ieşirea CBB va trece într-o stare stabilă, 0 sau 1, dar imprevizibilă. Însă în unele cazuri ieşirea va oscila sau va intra în starea metastabilă de la jumătatea distanţei dintre 0 şi 1, aşa cum este exemplificat pe desen pentru penultimul impuls de tact. Când CBB intră în starea metastabilă, el va reveni într-o stare stabilă numai după un interval de timp determinat probabilistic. El mai poate fi adus într-o stare stabilă prin aplicarea unui alt front de activare al semnalului de tact, pentru care intrarea **D** respectă timpii de pregătire şi de menţinere impuşi, ca în cazul ultimului impuls de tact din figură.

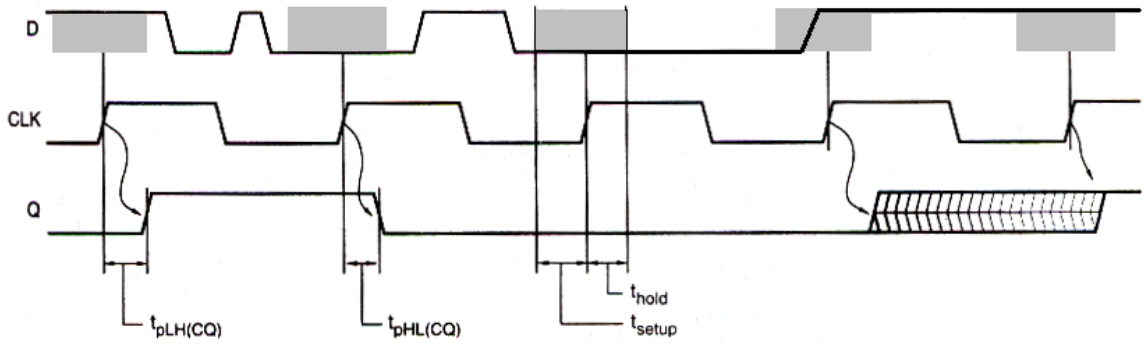


Figura 6-15 Comportarea temporală a unui CBB de tip D, activ pe frontul pozitiv

Un *CBB de tip **D**, activ pe frontul negativ* are doar intrarea de tact inversată, astfel că toate fenomenele descrise mai sus sunt activate de frontul descendent al semnalului **CLK_L**; prin convenţie, activarea pe frontul descendent este considerată de nivel activ LOW. Tabelul funcţiei şi simbolul logic pentru acest tip de CBB sunt cele din fig. 6-16.

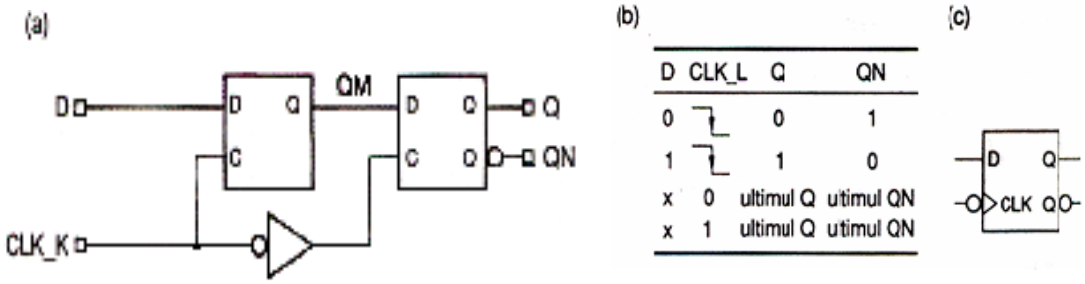


Figura 6-16 CBB de tip D, activ pe frontul negativ: (a) schema cu circuite latch D: (b) tabelul funcţiei: (c) simbolul logic

Unele CBB de tip **D** au intrări asincrone, care pot fi folosite pentru a trece forţat circuitul într-o anumită stare, independent de intrările **CLK** şi **D**. Aceste intrări, denumite uzual **PR** *(preset*- prefixare) şi **CLR** *(clear* - ştergere), se comportă ca intrările *set* şi *reset* ale unui circuit latch **S-R**. Simbolul logic şi circuitul **NAND** cu porţi aferente unui CBB de tip **D**, activ pe fronturi, prevăzut cu aceste intrări sunt prezentate în fig. 6-17. Unii proiectanţi de circuite logice folosesc intrările asincrone pentru realizarea unor funcţii secvenţiale dificile, însă este de preferat ca aceste intrări să fie păstrate pentru iniţializare şi testare, pentru a aduce forţat un circuit secvenţial într-o stare iniţială cunoscută.

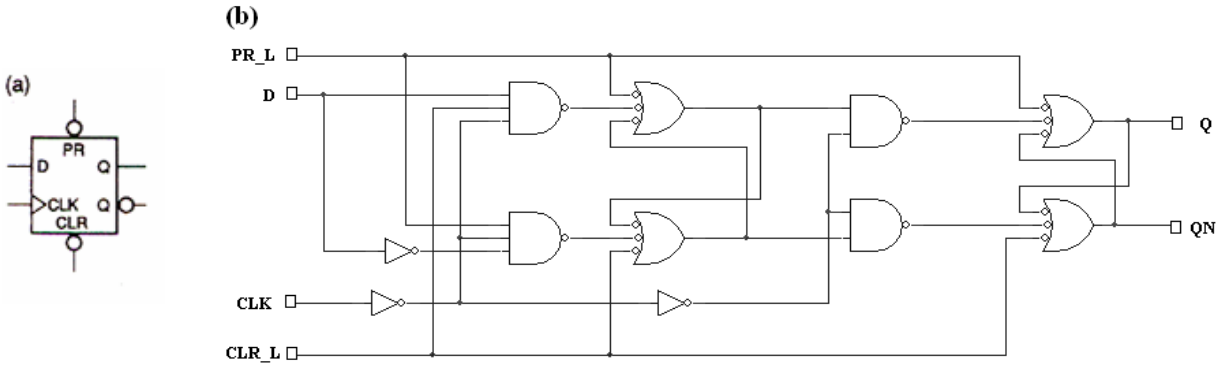
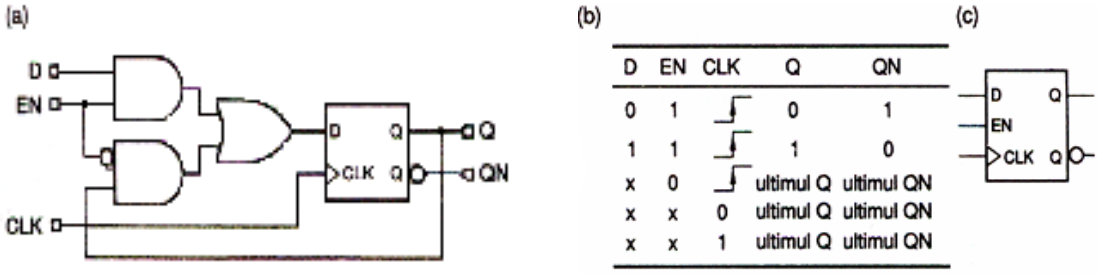


Figura 6-17 CBB de tip D, activ pe frontul pozitiv, cu intrări de prefixare şi ştergere: (a) simbolul logic: (b) schema cu porţi NAND

6.2.6 CBB de tip D, activ pe front şi cu intrare de activare

Una dintre funcţiile cerute, de obicei, la CBB de tip **D** este capacitatea de păstrare a ultimei valori stocate, fără a trece la o nouă valoare pe frontul de tact. Acest lucru se realizează prin adăugarea unei *intrări de activare (enable),* notată

cu **EN** sau **CE** *(clock enable - activarea tactului).* Denumirea "activarea tactului" este sugestivă, însă funcţia intrării suplimentare nu se obţine prin vreo comandă a semnalului de tact. În realitate, aşa cum arată figura 6-18(a), un multiplexor cu două intrări comandă valoarea ce se aplică la intrarea CBB de tip **D** din interior. Dacă **EN** este confirmat, se selectează semnalul extern de intrare **D**; dacă **EN** este negat, se foloseşte ieşirea curentă a CBB. Tabelul funcţiei rezultate este cel din (b). Simbolul CBB se vede în (c); la unele CBB, intrarea de activare este activă în LOW, fiind reprezentată cu un cerculeţ inversor.



D	EN	CLK	Q	QN
0	1	↑	0	1
1	1	↑	1	0
x	0	↑	ultimul Q	ultimul QN
x	x	0	ultimul Q	ultimul QN
x	x	1	ultimul Q	ultimul QN

Figura 6-18 CBB de tip D, activ pe frontul pozitiv, cu intrare de activare: (a) schema circuitului; (b) tabelul funcţiei; (c) simbolul logic

6.2.7 CBB de explorare

O funcţie importantă pentru testarea dispozitivelor ASIC este aşa-numita *capacitate de explorare.* Se urmăreşte posibilitatea de a se comanda intrarea **D** a CBB cu o altă sursă de date, la testarea dispozitivului. Când toate CBB lucrează în modul de testare, o schemă de testare poate fi "preluată prin scanare" de către ASIC prin intermediul intrărilor alternative de date de la CBB. După încărcarea schemei de testare în ASIC, CBB revin în modul de lucru "normal", toate fiind comandate de semnalul de tact, ca de obicei. După unul sau mai multe impulsuri de tact, CBB trec din nou în modul de testare, iar rezultatele testării sunt explorate din nou şi furnizate la exterior.

În fig. 6-19(a) apare schema unui CBB cu capacitate de explorare tipic. Nu este nimic altceva decât un CBB de tip **D** cu un multiplexor cu două intrări conectat la intrarea **D**. Când intrarea **TE** *(test enable* - activarea testării) este negată, circuitul se comportă ca un CBB de tip **D** obişnuit. Când **TE** este confirmată, ea nu mai culege date de la **D**, ci de la **TI** *(test input* - intrarea de testare). Această comportare funcţională este descrisă în (b), iar simbolul dispozitivului este reprodus în (c).

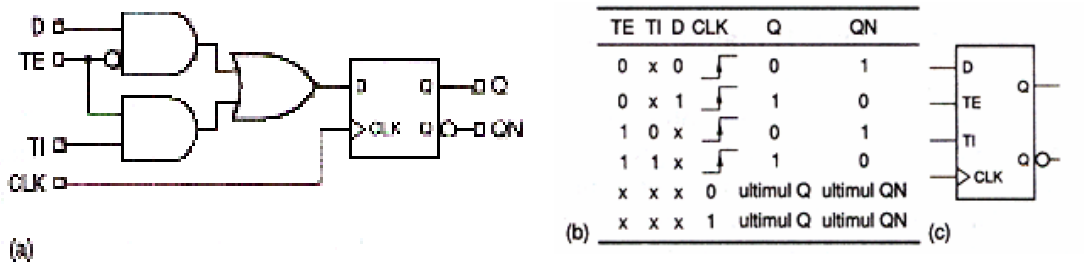


Figura 6-19 CBB de tip D, activ pe frontul pozitiv, cu capacitate de explorare: (a) schema circuitului; (b) tabelul funcţiei; (c) simbolul logic

Intrările suplimentare servesc la conectarea într-un *lanţ de explorare a* tuturor CBB dintr-un ASIC, pentru efectuarea testării. În fig. 6-20 este dat un exemplu simplu de lanţ de explorare cu patru CBB. Intrările **TE** ale tuturor CBB sunt conectate la o intrare **TE** generală, iar ieşirea 0 a fiecărui CBB este conectată în serie la intrarea **TI** a altuia. Conexiunile **TI**, **TE** şi **TO** (test output - ieşire de testare) sunt destinate exclusiv testării; restul schemei logice, conectată la intrările **D** şi ieşirile **Q** şi care realizează funcţia de bază a circuitului, nu apare în desen.

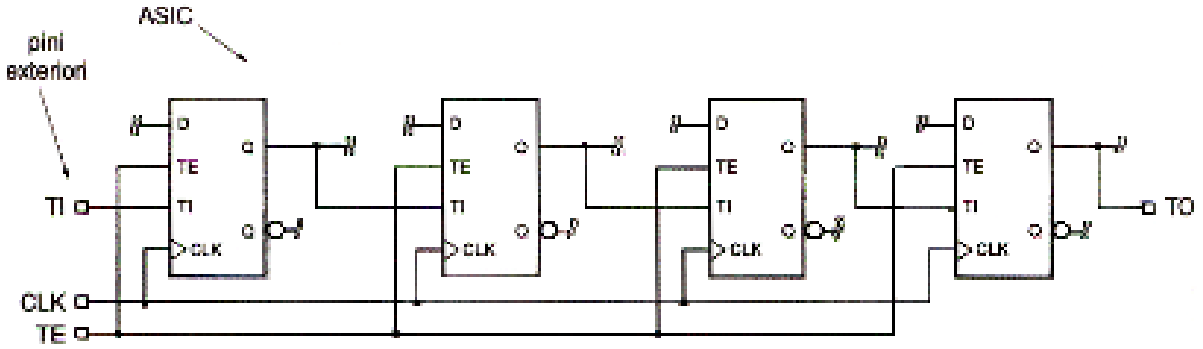


Figura 6-20 Lanţ de explorare cu 4 CBB

Pentru testarea circuitului, inclusiv a schemei logice de bază, intrarea generală **TE** este confirmată pe durata a *n* impulsuri de tact, timp în care la intrarea **TE** generală se aplică *n* biţi din vectorul de testare, care sunt exploraţi (preluaţi) de cele *n* CBB; în fig. 6-20, *n = 4*. Apoi **TE** este negată şi circuitul este lăsat să funcţioneze pe o durată de unul sau mai multe impulsuri de tact. Noua stare a circuitului, reprezentată de noile valori ale celor *n* CBB, poate fi observată (explorată) la **TO** confirmând **TE** pentru un timp de încă *n* impulsuri de tact. Pentru ca procesul de testare să fie mai eficient, la intrare poate fi explorat un alt vector de testare în acelaşi timp în care rezultatul precedent este explorat la ieşire.

Există mai multe tipuri diferite de CBB de explorare, corespunzând diferitelor tipuri de funcţii de bază ale CBB. De exemplu, CBB de tip **D** cu

intrare de activare din fig. 6-18 poate fi prevăzut cu capacitatea de explorare prin înlocuirea multiplexorului intern cu două intrări cu un multiplexor cu 3 intrări. La fiecare impuls de tact, CBB preia unul dintre semnalele **D**, **TI** sau starea sa curentă, în funcţie de valorile intrărilor **EN** şi **TE**. Şi alte tipuri de CBB, ca **J-K** şi **T**, pe care le vom prezenta în continuare, pot fi prevăzute cu capacitate de explorare.

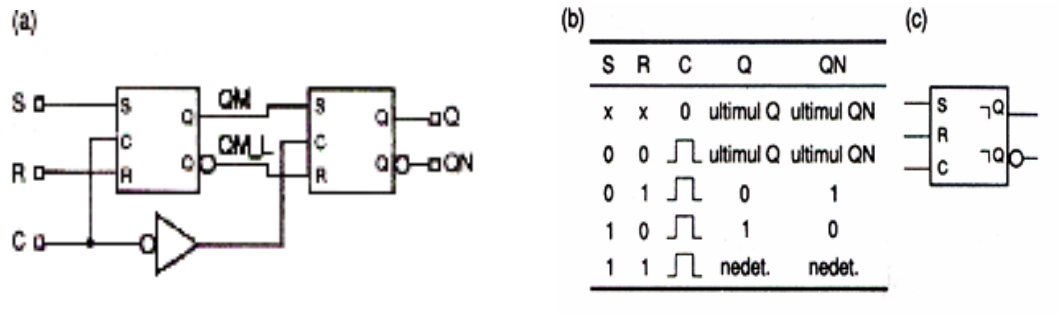
6.2.8 CBB de tip S-R master/slave

Am discutat mai devreme despre utilitatea circuitelor latch **S-R** în aplicaţiile de "comandă", când este posibil să existe condiţii independente pentru fixarea *(set)* şi ştergerea *(reset)* unui bit de comandă. Dacă se presupune că bitul de comandă trebuie să se modifice numai la anumite momente de timp, determinate de un semnal de tact, atunci avem nevoie de un CBB de tip **S-R** care, întocmai ca unul de tip **D**, îşi modifică ieşirile numai pe anumite fronturi ale semnalului de tact.

Dacă, în CBB de tip **D**, activ pe frontul negativ, din fig. 6-16, înlocuim circuitele latch de tip **S-R** cu unele de tip **D**, obţinem *CBB de tip* **S-R** *master/slave* din fig. 6-21. Ca şi un CBB de tip **D**, cel de tipul **S-R** îşi modifică ieşirile numai pe frontul descendent al unui semnal de comandă, **C**. Noua valoare de ieşire depinde însă de valoarea de intrare nu numai în intervalul frontului descendent, ci în întregul interval ce precede frontul descendent şi în care **C** este 1. Aşa cum se vede în fig. 6-22, un impuls îngust, aplicat pe **S** în orice moment din acest interval fixează circuitul latch master la o anumita valoare; similar, un impuls pe **R** îl readuce în starea iniţială. Valoarea transferată la ieşirea CBB pe frontul descendent al semnalului **C** depinde de acţiunea precedentă asupra circuitului latch master - *set* sau *reset* - în intervalul cât **C** a fost 1.

Simbolul logic al CBB de tip **S-R** master/slave, din fig. 6-21, nu este prevăzut cu un indicator al intrării dinamice, deoarece CBB nu este, de fapt, activ pe fronturi. El se aseamănă mai mult cu un circuit latch ce urmăreşte intrarea în întregul interval în care **C** este 1, însă ieşirea lui se modifică astfel încât să reflecte valoarea finală păstrată numai când **C** trece în 0. Pe simbol, un *indicator de întârziere a ieşirii* arată că semnalul de ieşire nu se modifică înainte de a se nega intrarea de activare, **C**. CBB care se comportă în acest mod sunt numite uneori *CBB active la impuls.*

Funcţionarea unui CBB de tip **S-R** master/slave este imprevizibilă dacă atât **S**, cât şi **R** se confirmă pe frontul descendent al lui **C**. În acest caz, ambele ieşiri, **Q** şi **QN**, ale circuitului latch master sunt 1 chiar în faţa frontului descendent. Când **C** trece în 0, ieşirile circuitului latch master se modifică imprevizibil, putând deveni chiar metastabile.



S	R	C	Q	QN
x	x	0	ultimul Q	ultimul QN
0	0	↓	ultimul Q	ultimul QN
0	1	↓	0	1
1	0	↓	1	0
1	1	↓	nedet.	nedet.

Figura 6-21 CBB de tip S-R master/slave: (a) schema cu circuite latch S-R; (b) tabelul funcţiei; (c) simbolul logic

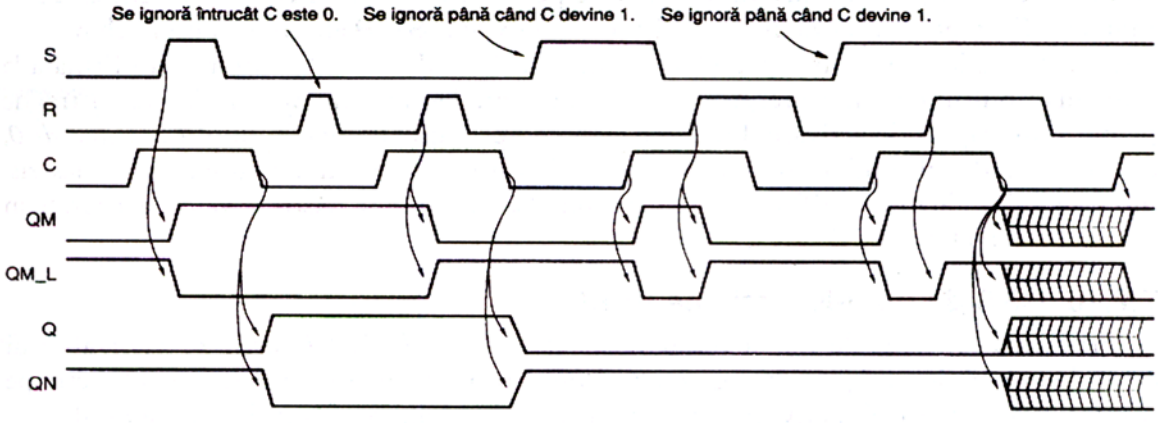
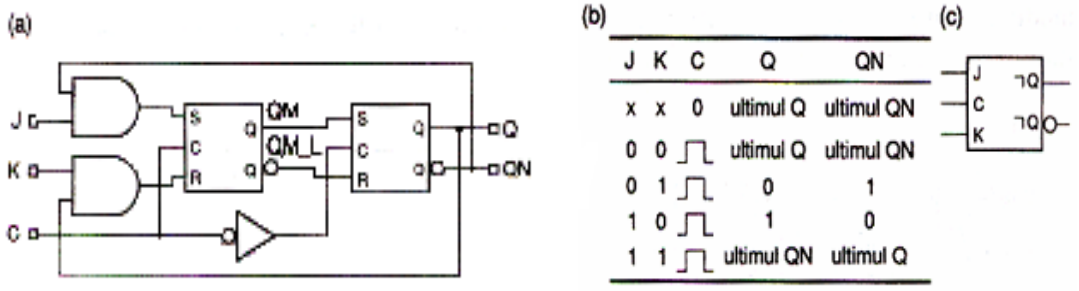


Figura 6-22 Comportarea internă şi funcţională a unui CBB de tip S-R master/slave

6.2.9 CBB de tip J-K master/slave



J	K	C	Q	QN
x	x	0	ultimul Q	ultimul QN
0	0	↓	ultimul Q	ultimul QN
0	1	↓	0	1
1	0	↓	1	0
1	1	↓	ultimul QN	ultimul Q

Figura 6-23 CBB de tip J-K master/slave: (a) schema cu circuite latch S-R; (b) tabelul funcţiei; (c) simbolul logic

Soluţionarea situaţiei când **S** şi **R** sunt confirmate simultan este oferită de *CBB de tip **J-K** master/slave.* Intrările **J** şi **K** sunt analoage cu **S** şi **R**. Deosebirea este că, aşa cum arată fig. 6-23, confirmarea intrării **J** duce la confirmarea intrării **S** a circuitului master numai dacă ieşirea **QN** curentă a CBB este 1 (deci **Q** = 0), iar confirmarea intrării **K** duce la confirmarea intrării **R** a circuitului master numai dacă ieşirea **Q** curentă este 1. Astfel, dacă **J** şi **K** se confirmă simultan, CBB trece în starea opusă stării sale curente.

Figura 6-24 prezintă comportarea funcţională a unui CBB de tip **J-K** master/slave pentru un set de intrări tipic. Remarcaţi că nu este necesar ca intrările **J** şi **K** să fie confirmate la sfârşitul impulsului de activare pentru ca ieşirea CBB să se modifice la acel moment. De fapt, din cauza intercondiţionării intrărilor **S** şi **R** ale circuitului latch master, este posibil ca ieşirea CBB să treacă în 1 chiar dacă nu **J**, ci **K** este intrarea confirmată la sfârşitul impulsului de activare. Acest comportament, denumit *capcană de* 1, este ilustrat de penultimul impuls de activare din desen. Un comportament analog, numit *capcană de 0,* este ilustrat de ultimul impuls de activare. Datorită modului de funcţionare descris, intrările **J** şi **K** ale unui CBB de tip **J-K** master/slave trebuie să păstreze valori acceptate în întregul interval în care **C** este 1.

6.2.10 CBB de tip J-K, activ pe front

Problema capcanei de 1 şi de 0 este rezolvată la *CBB de tip **J-K**, activ pe front*, al cărui echivalent funcţional apare în fig. 6-25. Având în interior un CBB de tip **D**, activ pe front, CBB de tip **J-K**, activ pe front explorează intrările pe frontul ascendent al semnalului de tact şi generează valoarea de ieşire următoare conform "ecuaţiei caracteristice" **Q* = J • Q' + K' • Q**.

În fig. 6-26 este prezentată comportarea funcţională tipică unui CBB de tip **J-K**, activ pe front. Asemenea intrării **D** a unui CBB de tip **D**, activ pe front, în vederea unei funcţionări corecte, intrările **J** şi **K** ale unui CBB de tip **J-K** trebuie să respecte valorile date de producător pentru timpii de pregătire şi de menţinere, având ca referinţă frontul de activare al semnalului de tact.

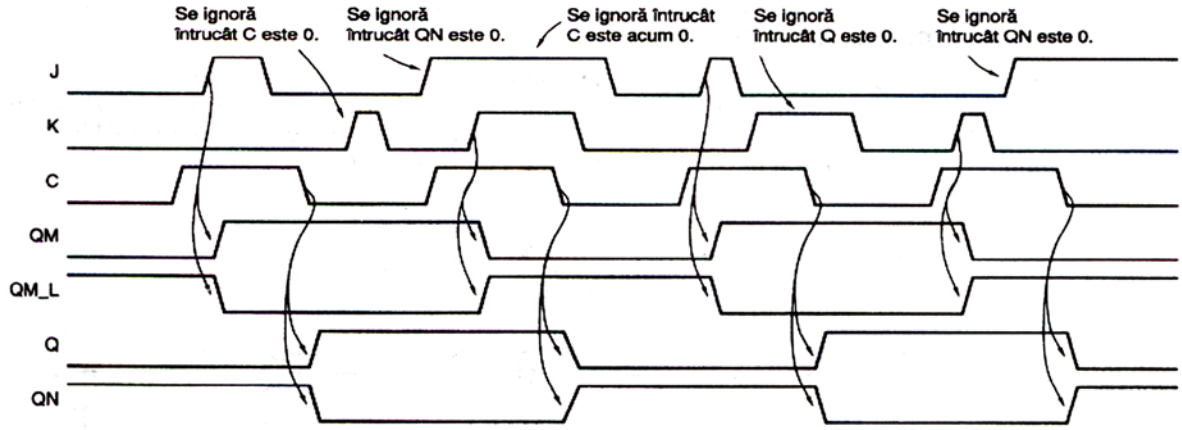


Figura 6-24 Comportarea internă şi funcţională a unui CBB de tip J-K master/slave

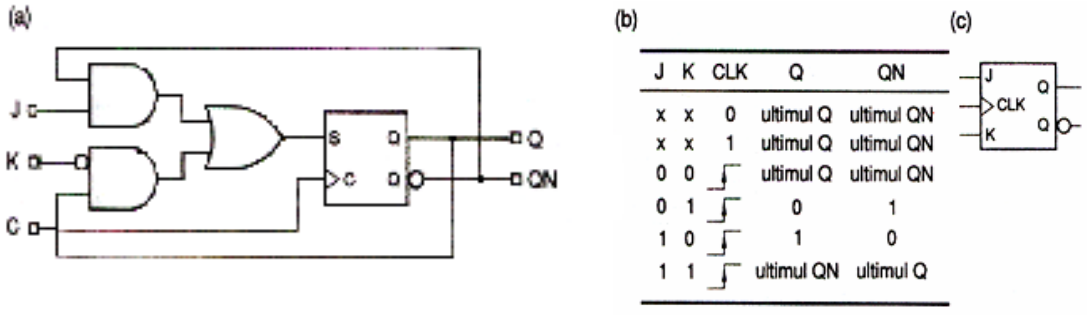


Figura 6-25 CBB de tip J-K, activ pe front: (a) schema echivalentă, cu un CBB de tip D, activ pe front; (b) tabelul funcţiei; (c) simbolul logic

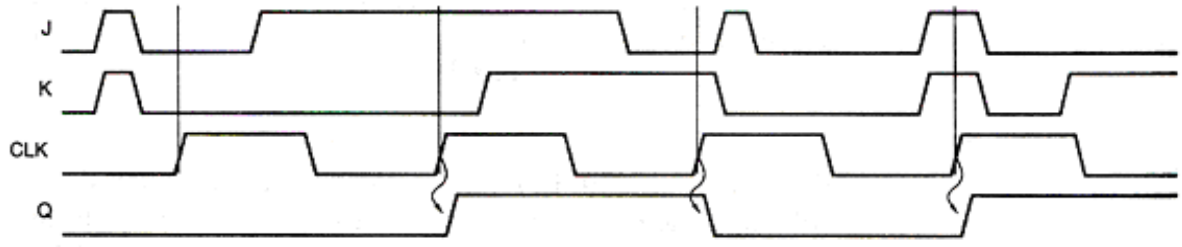


Figura 6-26 Comportarea funcţională a unui CBB de tip J-K, activ pe front

Deoarece CBB de tip **J-K**, active pe front elimină problema capcanelor de 1 şi 0 şi pe cea a confirmării simultane a ambelor intrări de comandă, ele au scos din competiţie, în mare măsură, tipurile mai vechi active pe front. *74x109* este un CBB de tip **J-K'**, activ pe front, din familia TTL, cu intrarea **K** activă în LOW (şi denumită **K'** sau **K_L**).

Cea mai răspândită aplicaţie a CBB de tip **J-K** o constituie automatele de stări sincronizate cu semnal de tact. Funcţia logică prin care se obţine starea următoare a unui CBB de tip **J-K** este uneori mai simplă decât cea pentru CBB de tip **D**. Totuşi, majoritatea automatelor de stări se proiectează tot cu CBB de tip **D**, deoarece metodologia de proiectare este ceva mai simplă şi pentru că majoritatea dispozitivelor cu logică secvenţială programabilă conţin CBB de tip **D**, nu **J-K**. Prin urmare, vom acorda mai multă atenţie circuitelor bistabile de tip **D**.

6.2.11 CBB de tip T

*CBB de tip **T*** (de la *toggle* = circuit bistabil) îşi schimbă starea la fiecare impuls de tact. Figura 6-27 prezintă simbolul şi ilustrează comportarea unui CBB de tip **T**, activ pe frontul pozitiv. Remarcaţi că semnalul de la ieşirea **Q** a CBB are o frecvenţă egală exact cu jumătate din frecvenţa semnalului de intrare, **T**. În fig. 6-28 vedeţi cum se obţine un CBB de tip **T** dintr-unul de tip **D** sau **J-K**. Bistabilele de tip **T** sunt folosite cu precădere la numărătoare şi divizoare de frecvenţă.

În numeroase aplicaţii ale CBB de tip **T**, circuitul nu trebuie să basculeze la fiecare impuls de tact. În asemenea aplicaţii se pot folosi *CBB de tip **T** cu intrare de activare*. Aşa cum se vede în fig. 6-29, CBB îşi schimbă starea pe frontul semnalului de tact numai dacă semnalul de activare **EN** este confirmat. Ca şi intrările **D**, **J** şi **K** de la celelalte CBB active pe front, intrarea **EN** trebuie să respecte valorile timpilor de pregătire şi menţinere, a căror referinţă este frontul de activare al semnalului de tact. În fig. 6-30 apar circuitele din fig. 6-28, uşor modificate, prevăzute cu o intrare **EN**.

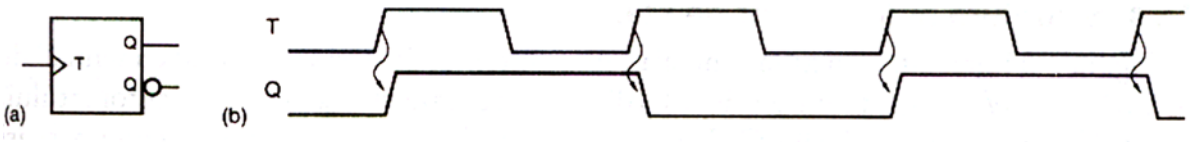


Figura 6-27 CBB de tip T, activ pe frontul pozitiv: (a) simbolul logic; (b) comportarea funcţională

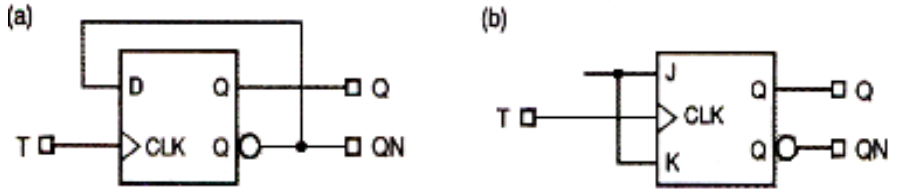


Figura 6-28 Configuraţii de circuit posibile pentru un CBB de tip T: (a) cu CBB de tip D; (b) cu CBB de tip J-K

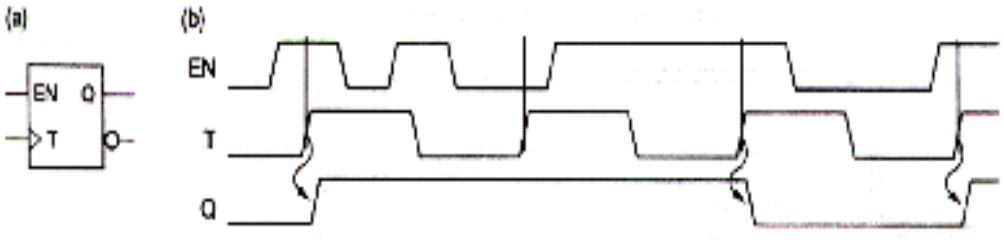


Figura 6-29 CBB de tip T, activ pe frontul pozitiv, cu intrare de activare: (a) simbolul logic; (b) comportarea funcţională

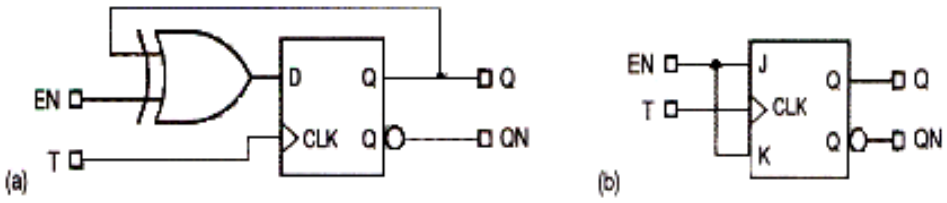


Figura 6-30 Scheme posibile pentru un CBB de tip T cu intrare de activare: (a) cu CBB de tip D; (b) cu CBB de tip J-K

7. Circuite logice secvențiale – numărătoare, registre de deplasare, circuite iterative

7.1 Numărătoare

Denumirea *numărător* este utilizată, în general, pentru orice circuit secvenţial sincron a cărui diagramă de stări conţine un singur ciclu, ca aceea din fig. 7-1. Numărul de stări din ciclul unui numărător se numeşte *modulo.* Un numărător cu *m* stări este numit adesea *numărător modulo m* sau, uneori, *numărător divizor cu m.* Un numărător modulo un număr diferit de puterile lui 2 prezintă stări suplimentare care nu sunt folosite în funcţionare normală.

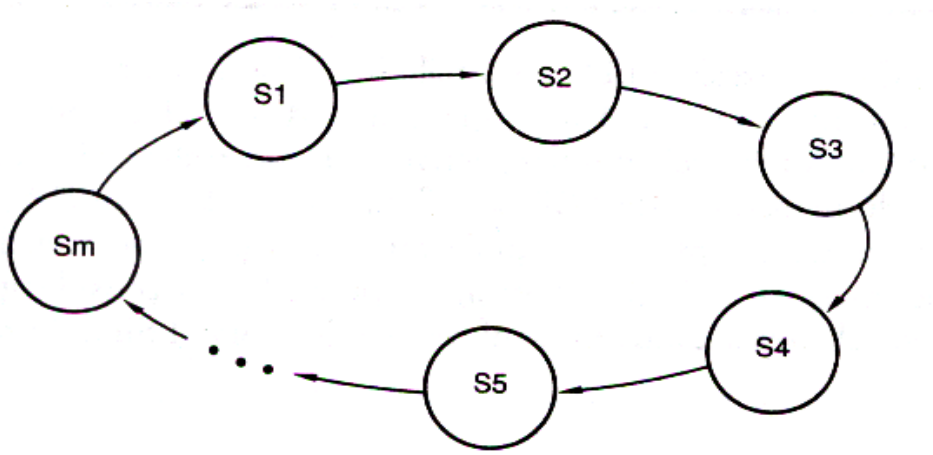


Figura 7-1 Structura generală a diagramei de stări a unui numărător – un singur ciclu

Tipul de numărător cel mai mult utilizat este, probabil, *numărătorul binar de n biţi.* Un astfel de numărător are *n* CBB şi 2^n stări, care sunt parcurse în ordinea *0, 1, 2,...,* 2^n*-l, 0, 1, ...* . Codul fiecărei stări corespunde întregului binar respectiv, de *n* biţi.

7.1.1 Numărătoare pieptene

Un numărător binar de *n* biţi se poate construi cu doar *n* CBB şi nici un fel de alte componente, pentru orice valoare *n.* În fig. 7-2 este prezentat un asemenea numărător cu *n*=4. Un CBB de tip **T** îşi schimbă starea (basculează) pe fiecare front ascendent al semnalului de tact de la intrarea sa. Deci fiecare bit al numărătorului trece în valoarea opusă dacă şi numai dacă bitul imediat precedent trece din 1 în 0. Aceasta corespunde succesiunii normale de numărare

în binar, trecerea unui anumit bit din 1 în 0 generând un transport către următorul bit mai semnificativ. Numărătorul se numeşte *numărător pieptene* deoarece informaţia de transport „unduieşte" de la biţii cei mai puţin semnificativi către biţii mai semnificativi, câte un bit o dată.

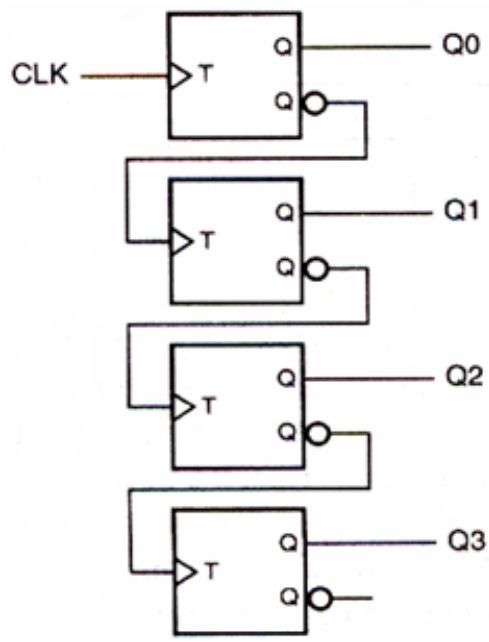


Figura 7-2 Numărător binar pieptene de 4 biţi

7.1.2 Numărătoare sincrone

La un numărător pieptene sunt necesare mai puţine componente decât la orice alt tip de numărător binar, însă preţul plătit pentru aceasta este viteza mai scăzută decât a oricărui alt tip de numărător binar. În cazul cel mai defavorabil, când bitul cel mai semnificativ este cel ce trebuie să se modifice, ieşirea nu devine validă decât după un timp $n \bullet t_{TQ}$ după apariţia frontului ascendent al semnalului **CLK**, unde t_{TQ} este timpul de propagare de la intrarea la ieşirea unui CBB de tip **T**.

La un *numărător sincron,* intrările de tact ale tuturor CBB sunt conectate la acelaşi semnal **CLK**, astfel că ieşirile tuturor CBB se modifică simultan, după un interval de timp de numai t_{TQ} ns. Aşa cum se vede în fig. 7-3, în acest scop este necesară utilizarea unor CBB de tip **T** cu intrări de activare; ieşirea basculează pe frontul ascendent al semnalului **T** dacă şi numai dacă **EN** este confirmat. Schema logică combinaţională de la intrările **EN** impune care dintre CBB basculează pe fiecare front ascendent al lui **T**.

Aşa cum arată fig. 7-3, se poate prevedea şi un semnal de control al activării, **CNTEN**. Fiecare CBB de tip **T** basculează dacă şi numai dacă semnalul **CNTEN** este confirmat şi toţi biţii de ordine inferioare ai numărătorului sunt 1. Ca şi în cazul numărătorului binar pieptene, un numărător binar sincron de *n* biţi se poate construi folosind o anumită schemă logică pentru

fiecare bit, în cazul de faţă - un CBB de tip **T** cu intrare de activare şi o poartă **AND** cu două intrări.

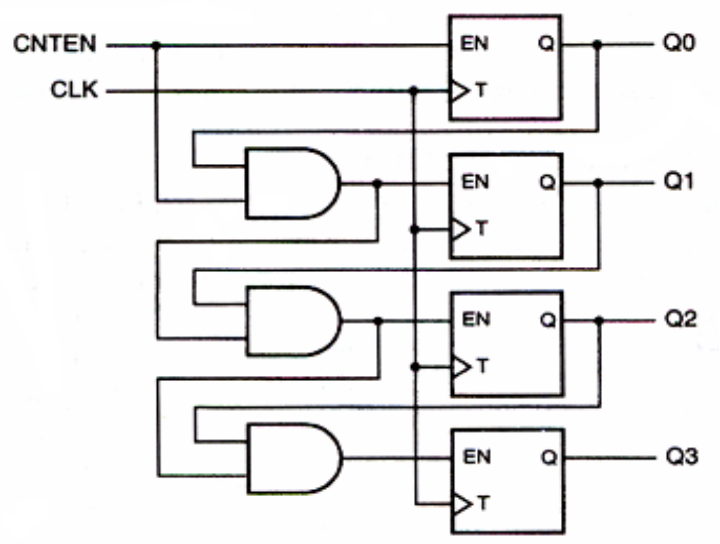


Figura 7-3 Numărător sincron binar de 4 biţi cu circuit logic de activare serie

Structura de numărător din fig. 7-3 este denumită uneori *numărător sincron serie,* deoarece semnalele combinaţionale de activare se propagă serial de la biţii cei mai puţin semnificativi către cei mai semnificativi. Dacă perioada de tact este prea scurtă, este posibil ca timpul de propagare de la LSB la MSB ai numărătorului să nu fie suficient. Problema aceasta este eliminată în fig. 7-4 prin comandarea fiecărei intrări **EN** cu o poartă **AND** folosită numai în acest scop, rezultând o schemă logică cu un singur nivel. Am obţinut astfel un *numărător sincron paralel* - cea mai rapidă structură de numărător binar.

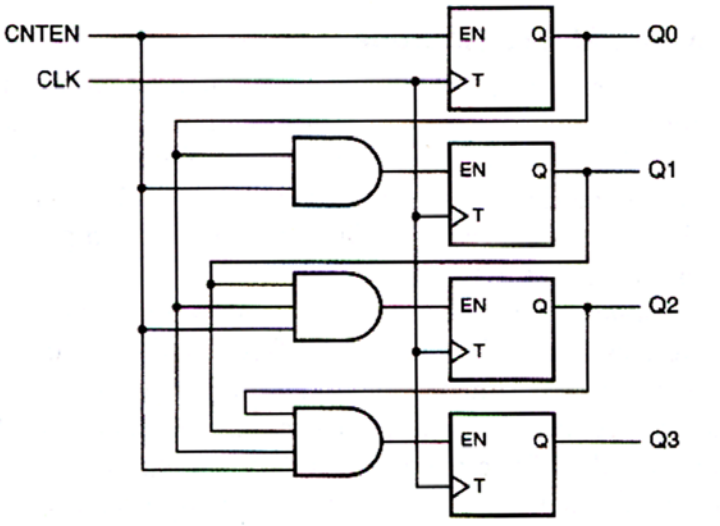


Figura 7-4 Numărător sincron binar de 4 biţi cu circuit logic de activare paralel

7.1.3 Numărătoare MSI şi aplicaţii ale lor

Cel mai utilizat numărător MSI este *74x163,* un numărător binar sincron de 4 biţi, cu intrări de încărcare şi de ştergere active în LOW, având simbolul logic tradiţional din fig. 7-5. Funcţionarea sa este sistematizată în tabelul de stări din tabelul 7-1, iar schema sa logică internă este cea din fig. 7-6.

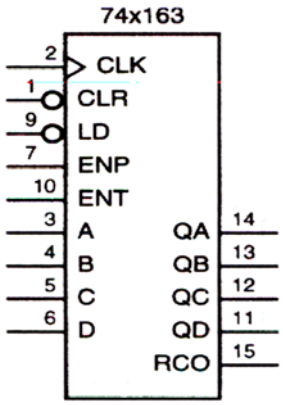


Figura 7-5 Simbolul logic tradiţional al numărătorului 74x163

În interiorul unui ‘163, CBB utilizate sunt de tip **D**, nu **T**, pentru ca funcţiile de încărcare şi de ştergere să se realizeze mai uşor. Fiecare intrare **D** este comandată de un multiplexor cu două intrări, format dintr-o poartă **OR** şi două porţi **AND**. Ieşirea multiplexorului este 0 dacă semnalul de intrare **CLR_L** este confirmat. În caz contrar, poarta **AND** din partea de sus lasă să treacă semnalul de date de intrare **(A, B, C** sau **D**) către ieşire, dacă este confirmat semnalul de intrare **LD_L**. Dacă nici unul dintre semnalele **CLR_L** şi **LD_L** nu este confirmat, poarta **AND** din partea de jos transmite la ieşirea multiplexorului semnalul de la ieşirea unei porţi **XNOR**.

La ‘163, funcţia de numărare este realizată de porţile **XNOR**. Una dintre intrările unei porţi **XNOR** corespunde bitului numărat (**QA**, **QB**, **OC** sau **QD**); cealaltă intrare este 1, complementând bitul numărat dacă şi numai dacă ambele semnale de activare, **ENP** şi **ENT,** sunt confirmate, iar toţi biţii de numărare de ordine inferioare sunt 1. Semnalul **RCO** („ripple carry out” - transport pieptene către exterior) indică un transport din poziţia bitului celui mai semnificativ şi este 1 când toţi biţii de numărare sunt 1, iar **ENT** este confirmat.

Deşi majoritatea numărătoarelor MSI sunt prevăzute cu intrări de activare, ele se utilizează frecvent în mod *asincron*, caz în care sunt activate continuu. În fig. 7-7 se arată conexiunile necesare pentru ca un ‘163 să funcţioneze în acest mod, iar în fig. 7-8 se văd formele de undă obţinute. Observaţi că, începând cu **QA**, frecvenţa fiecărui semnal este egală cu jumătate din frecvenţa semnalului precedent. Astfel, un ‘163 asincron poate fi folosit ca numărător divizor cu 2, cu 4, cu 8 şi cu 16, dacă se ignoră în mod adecvat biţii de ieşire de ordine superioară.

Tabel 7-1 Tabelul de stări pentru un numărător binar de 4 biţi 74x163

Intrări				Starea curentă				Starea următoare			
CLR_L	LD_L	ENT	ENP	QD	QC	QB	QA	QD*	QC*	QB*	QA*
0	x	x	x	x	x	x	x	0	0	0	0
1	0	x	x	x	x	x	x	D	C	B	A
1	1	0	x	x	x	x	x	QD	QC	QB	QA
1	1	x	0	x	x	x	x	QD	QC	QB	QA
1	1	1	1	0	0	0	0	0	0	0	1
1	1	1	1	0	0	0	1	0	0	1	0
1	1	1	1	0	0	1	0	0	0	1	1
1	1	1	1	0	0	1	1	0	1	0	0
1	1	1	1	0	1	0	0	0	1	0	1
1	1	1	1	0	1	0	1	0	1	1	0
1	1	1	1	0	1	1	0	0	1	1	1
1	1	1	1	0	1	1	1	1	0	0	0
1	1	1	1	1	0	0	0	1	0	0	1
1	1	1	1	1	0	0	1	1	0	1	0
1	1	1	1	1	0	1	0	1	0	1	1
1	1	1	1	1	0	1	1	1	1	0	0
1	1	1	1	1	1	0	0	1	1	0	1
1	1	1	1	1	1	0	1	1	1	1	0
1	1	1	1	1	1	1	0	1	1	1	1
1	1	1	1	1	1	1	1	0	0	0	0

Circuitul ‘163 este integral sincron; cu alte cuvinte, ieşirile lui se modifică numai pe frontul ascendent al semnalului **CLK**. În unele aplicaţii este necesară o funcţie de ştergere asincronă, aşa cum este prevăzută la *74x161*. ‘161 are aceeaşi repartizare a semnalelor la pini ca şi ‘163, dar intrarea sa **CLR_L** este conectată la intrările asincrone de ştergere ale CBB din interior.

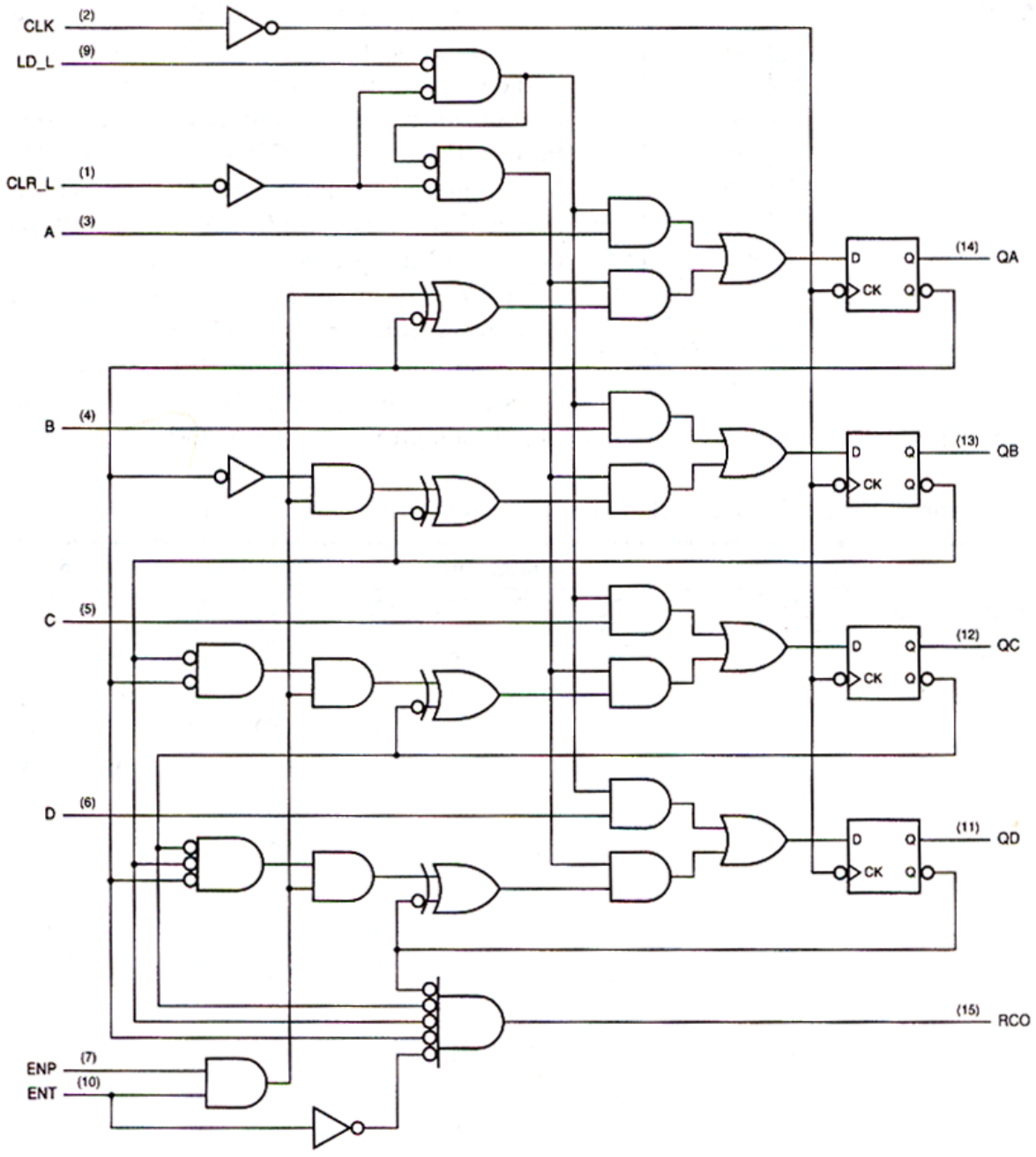


Figura 7-6 Schema logică a numărătorului binar sincron de 4 biţi 74x163, inclusiv numerotarea pinilor la capsula standard „dual in line” cu 16 pini

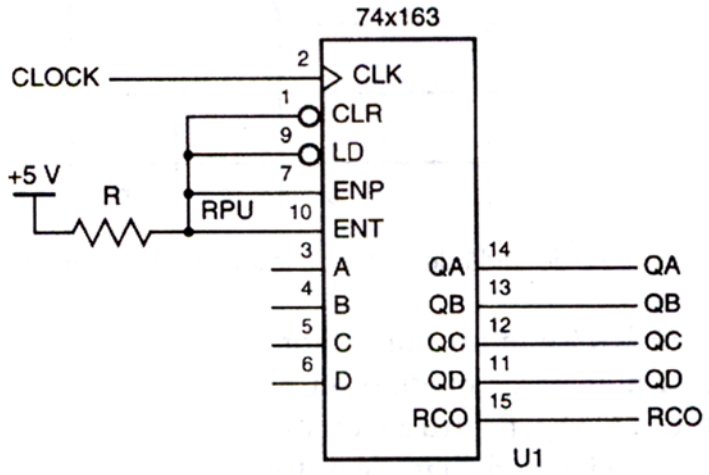


Figura 7-7 Conexiuni la un 74x163 pentru funcţionare în mod asincron

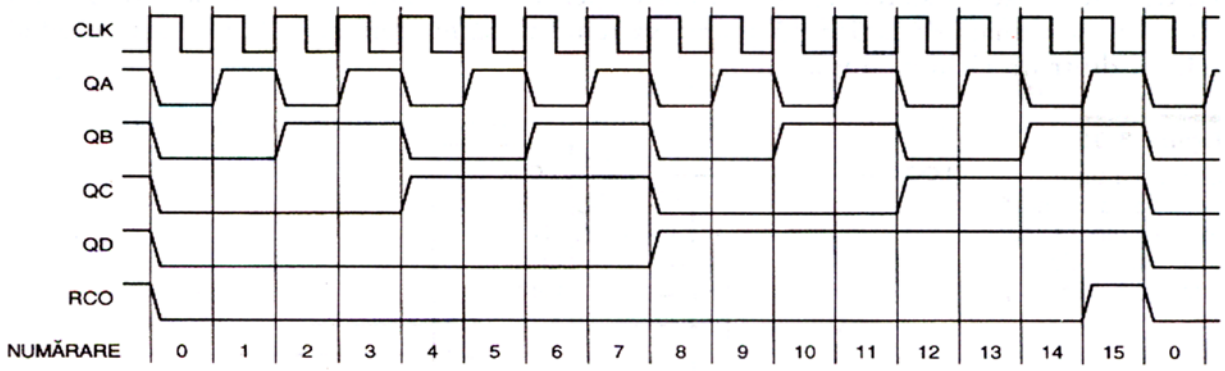


Figura 7-8 Formele de undă de tact şi de ieşire la un numărător asincron divizor cu 16

74x160 şi *74x162* sunt alte variante, cu aceeaşi repartizare a semnalelor la pini şi aceleaşi funcţii generale ca şi ‘161 şi ‘163, însă succesiunea de numărare este modificată astfel încât să se treacă în starea 0 după starea 9. Prin urmare, aceste numărătoare sunt modulo 10, fiind denumite uneori *numărătoare decadice*. În fig. 7-9 sunt prezentate formele de undă de ieşire la un ‘160 sau ‘162 asincron. Deşi semnalele de ieşire **QD** şi **QC** au frecvenţa egală cu o zecime din cea a semnalului **CLK**, factorul lor de comandă nu este de 50%, iar semnalul **QC**, cu frecvenţa egală cu o cincime din cea de intrare, nu are factorul de comandă constant.

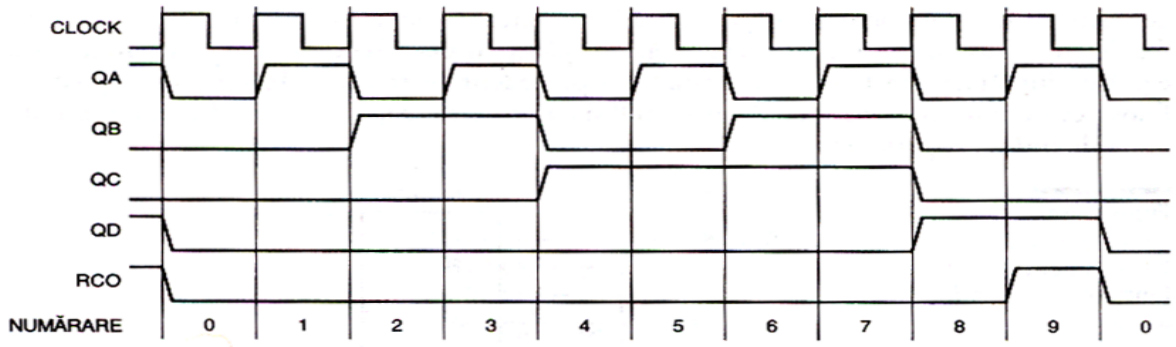


Figura 7-9 Formele de undă de tact şi de ieşire la un numărător asincron divizor cu 10

Cu toate că ‘163 este un numărător modulo 16, el poate fi conectat astfel încât să funcţioneze ca numărător modulo un număr mai mic ca 16 folosindu-se semnalele de intrare **CLR_L** sau **LD_L** pentru a scurta succesiunea de numărare normală. Ca exemplu, în fig. 7-10 se prezintă o modalitate de utilizare a unui ‘163 ca numărător modulo 11. Ieşirea **RCO**, care sesizează starea 15, este folosită pentru a impune starea următoare 5, deci circuitul va număra de la 5 la 15 şi apoi va reîncepe să numere de la 5, numărul total al stărilor dintr-un ciclu de numărare fiind 11.

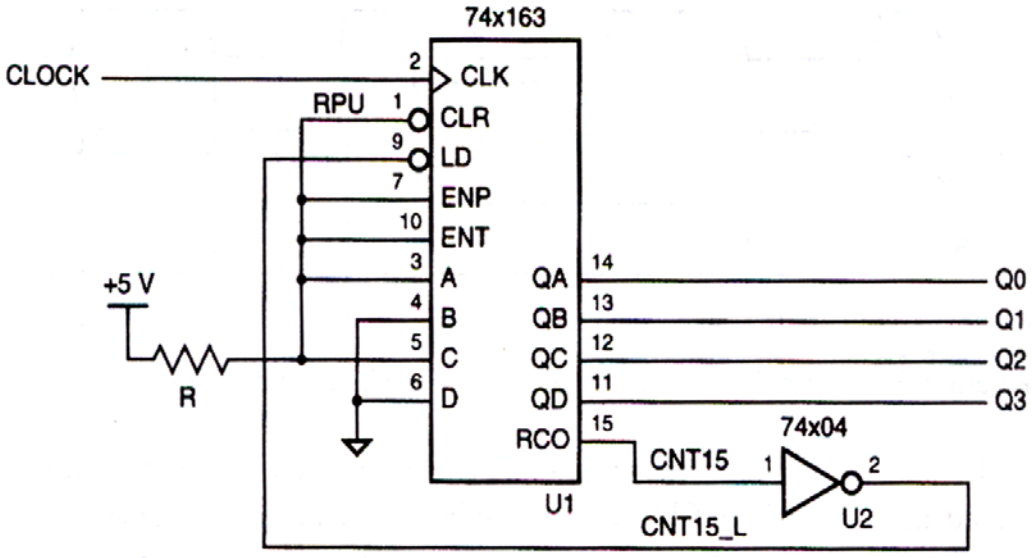


Figura 7-10 Utilizarea unui 74x163 ca numărător modulo 11, cu succesiunea de numărare 5, 6, …, 15, 5, 6, ….

Un alt mod de numărare modulo 11 cu un ‘163 este cel din fig. 7-11. Acest circuit foloseşte o poartă **NAND** pentru a sesiza starea 10, impunând 0 ca stare următoare. Remarcaţi că pentru sesizarea stării 10 (1010 în binar) se foloseşte o singură poartă cu două intrări. Deşi, în mod normal, pentru sesizarea condiţiei **CNT10 = 03 • 02' • 01 • 00'** ar trebui utilizată o poartă cu 4 intrări, cea cu două intrări se foloseşte de faptul că, în succesiunea normală de numărare

de la 0 la 10, nici o altă stare nu are **Q3** = 1 şi **Q1** = 1. În general, pentru a sesiza starea N a unui numărător binar ce numără de la 0 la N, trebuie să combinăm prin **AND** numai biţii de stare care au valoarea 1 în codarea lui N în binar.

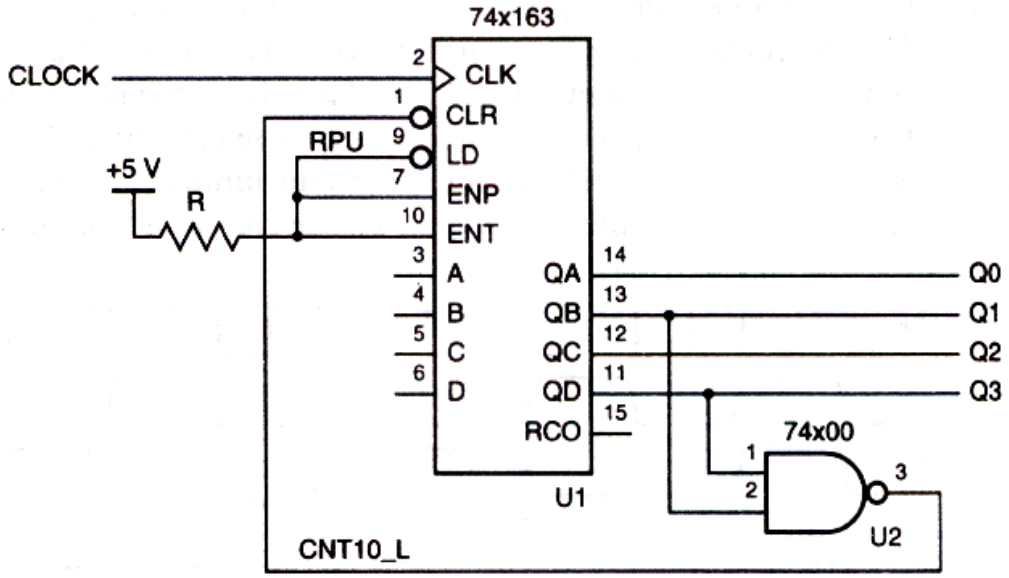


Figura 7-11 Utilizarea unui 74x163 ca numărător modulo 11, cu succesiunea de numărare 0, 1, 2, …, 10, 0, 1, ….

Există şi alte modalităţi de a construi un numărător modulo 11 cu ‘163. Alegerea schemei - una dintre cele prezentate sau o combinaţie a lor - depinde de cerinţele aplicaţiei.

Un numărător binar modulo un număr mai mare ca 16 se construieşte prin conectarea în cascadă a mai multor 74x163. În fig. 7-12 sunt prezentate conexiunile generale ale unui asemenea numărător. Intrările **CLK**, **CLR_L** şi **LD_L** ale tuturor dispozitivelor ‘163 sunt conectate în paralel, astfel că toate numără sau se şterg sau se încarcă simultan. Un semnal de control al activării numărării (**CNTEN**) este conectat la dispozitivul ‘163 de ordin inferior. Ieşirea **RCO4** se confirmă dacă şi numai dacă dispozitivul ‘163 de ordin inferior este în starea 15 şi **CNTEN** este confirmat; **RCO4** este conectată la intrările de activare ale dispozitivului ‘163 de ordin superior. În acest mod, atât informaţiile de transport, cât şi semnalul de control al activării numărării sunt conectate în stil „pieptene” între ieşirea unui etaj de numărare de 4 biţi şi etajul următor. Ca şi la numărătorul serial sincron din fig. 7-3, schema de faţă poate fi extinsă astfel încât să se construiască numărătoare cu orice număr de biţi; viteza maximă de numărare este limitată de timpul de propagare al semnalului de transport pieptene prin toate etajele.

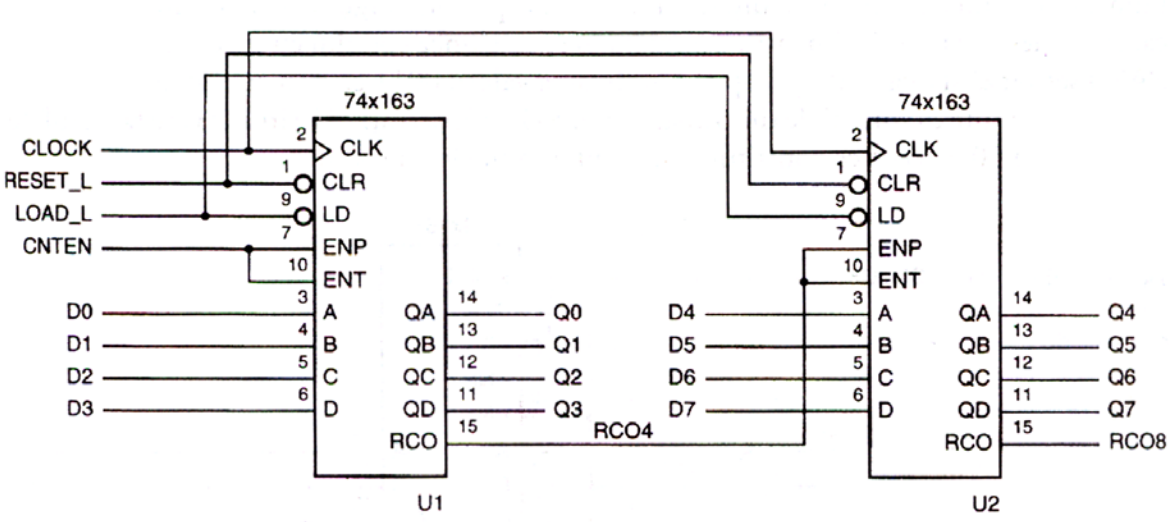


Figura 7-12 Conectarea generală în cascadă a mai multor numărătoare 74x163

7.2 Registre de deplasare

7.2.1 Structura unui registru de deplasare

Un registru de deplasare este un registru de *n* biţi cu posibilitatea de deplasare a datelor stocate în el cu câte o poziţie de un bit la fiecare impuls de tact. În fig. 7-13 este prezentată structura unui registru de deplasare cu intrare şi ieşire seriale. *Intrarea serie*, **SERIN**, indică un nou bit ce urmează a fi deplasat către o extremitate la fiecare impuls de tact. Acest bit apare la *ieşirea serie*, **SEROUT**, după n impulsuri de tact, iar la impulsul de tact *n* + 1 se pierde. Deci un registru de deplasare de *n* biţi, cu intrare şi ieşire seriale poate fi folosit pentru a întârzia un semnal cu n impulsuri de tact.

Un *registru de deplasare serie-paralel*, prezentat în fig. 7-14, are ieşiri pentru toţi biţii stocaţi, care pot fi preluaţi de alte circuite. Un astfel de registru de deplasare poate fi folosit la efectuarea *conversiei serie-paralel.*

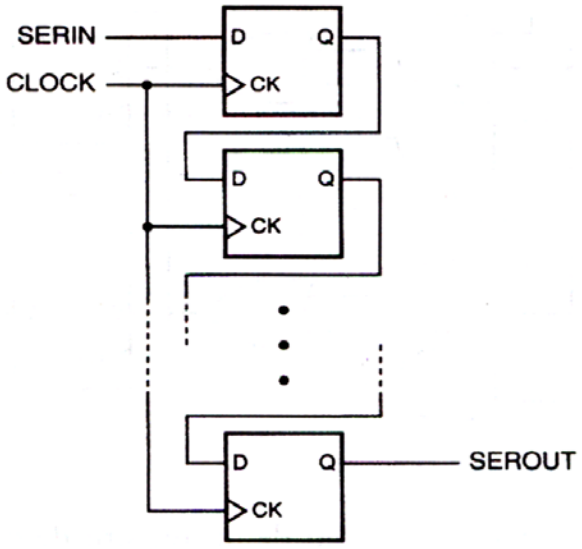


Figura 7-13 Structura unui registru de deplasare serie-serie

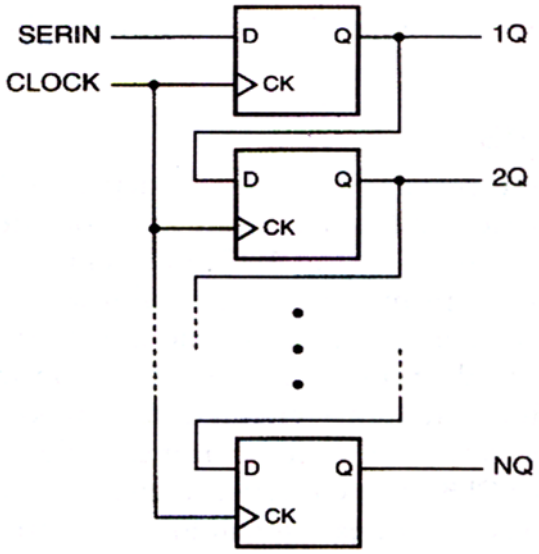


Figura 7-14 Structura unui registru de deplasare serie-paralel

Invers, este posibil să construim un *registru de deplasare paralel-serie*. În fig. 7-15 vedeţi structura generală a unui asemenea dispozitiv. La fiecare impuls de tact, registrul fie încarcă date noi de la intrările **1D** ... **ND**, fie îşi deplasează conţinutul curent, în funcţie de valoarea de la intrarea de comandă **LOAD/SHIFT** (pe care o putem numi **LOAD** sau **SHIFT_L**). În interior, dispozitivul foloseşte un multiplexor cu două intrări, conectat la intrarea D a fiecărui CBB, pentru a selecta unul dintre cele două cazuri. Un registru de deplasare *paralel-serie* poate utilizat la efectuarea *conversiei paralel-serie.*

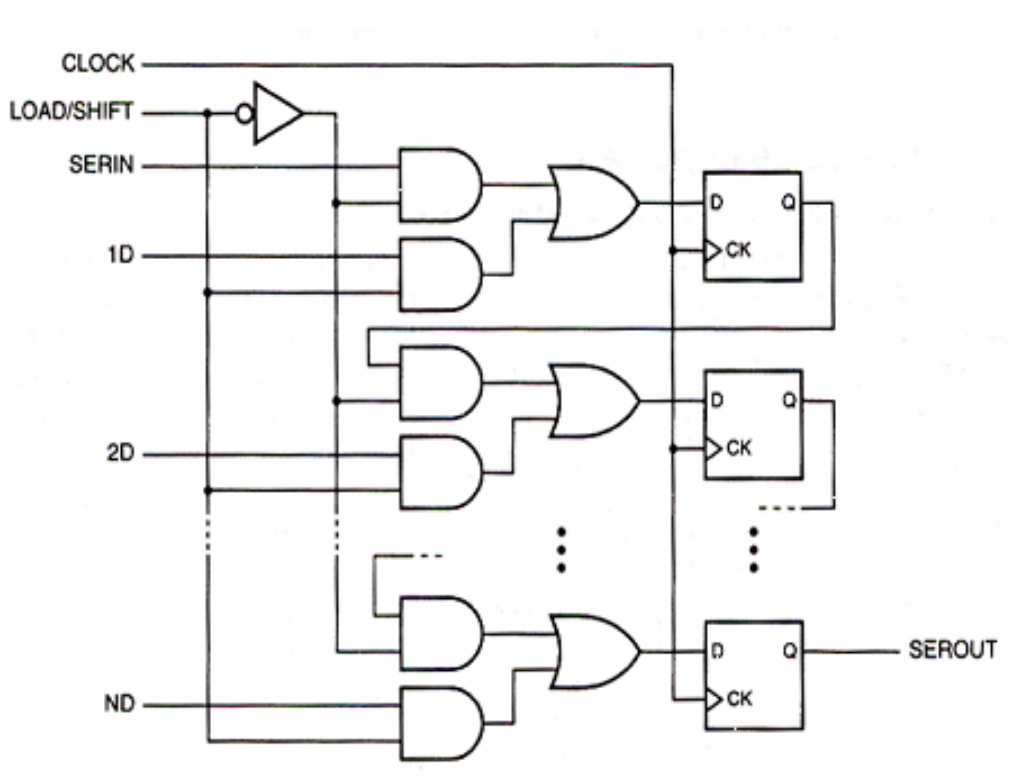


Figura 7-15 Structura unui registru de deplasare paralel-serie

Dacă prevedem ieşiri pentru toţi biţii stocaţi de un registru de deplasare cu intrare paralel, obţinem un *registru de deplasare paralel-paralel*, ca acela din fig. 7-16. Un asemenea dispozitiv este destul de general pentru a fi utilizat în oricare dintre aplicaţiile registrelor de deplasare descrise mai sus.

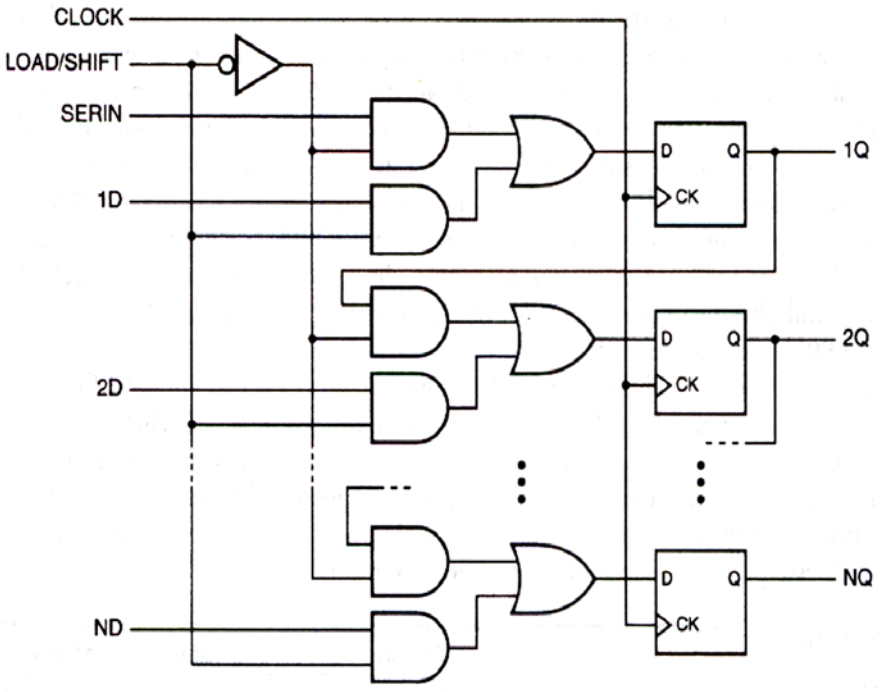


Figura 7-16 Structura unui registru de deplasare paralel-paralel

7.2.2 Registre de deplasare MSI

Figura 7-17 prezintă simbolurile logice a trei registre de deplasare MSI de 8 biţi, larg utilizate. *74x164* este un dispozitiv serie-paralel cu o intrare asincronă de ştergere (**CLR_ L**). El are două intrări seriale, combinate în interior prin **AND.** Aceasta înseamnă că atât **SERA**, cât şi **SERB** trebuie sa fie 1 pentru ca în primul bit al registrului să se introducă un 1.

74x166 este un registru de deplasare paralel-serie ce are, de asemenea, o intrare asincronă de ştergere. El efectuează deplasarea când **SH/LD** este 1, iar în caz contrar încarcă date noi. '166 are un circuit de tact mai puţin întâlnit, numit „circuit de tact cu porţi"; el are două intrări de tact, conectate la CBB din interior ca în fig. 7-17 (c). Proiectanţii dispozitivului '166 au dorit ca intrarea **CLK** să fie conectată la un semnal de tact asincron, iar **CLKINH** să fie confirmat când se doreşte blocarea semnalului **CLK**, astfel ca la impulsul de tact următor să nu aibă loc nici deplasare, nici încărcare, iar conţinutul curent al registrului să se păstreze. Însă, pentru ca dispozitivul să funcţioneze în modul descris, trebuie ca semnalul **CLKINH** să se modifice numai când **CLK** este 1; în caz contrar, la CBB din interior apar fronturi nedorite ale semnalului de tact. O modalitate mai sigură de a obţine o funcţie de „menţinere" se întâlneşte la dispozitivele prezentate în continuare.

74x194 este un registru de deplasare MSI de 4 biţi, bidirecţional, paralel-paralel. Schema sa logică apare în fig. 7-18. Registrele de deplasare studiate anterior sunt numite *registre de deplasare unidirecţionale,* deoarece efectuează deplasarea într-un singur sens. '194 este un *registru de deplasare bidirecţional,* deoarece conţinutul său poate fi deplasat în oricare dintre cele două sensuri, în funcţie de o intrare de comandă. Cele două sensuri sunt denumite „la stânga" şi „la dreapta", dar nu este obligatoriu ca schema logică şi simbolul logic să fie reprezentate orientate în acest mod. La un '194, la *stânga* înseamnă „în sensul dinspre **QD** spre **QA**", iar la *dreapta* înseamnă „în sensul dinspre **QA** spre **QD**". Schema logică şi simbolul dispozitivului '194 prezentate aici se conformează denumirilor dacă le rotiţi cu 90° în sensul acelor de ceasornic.

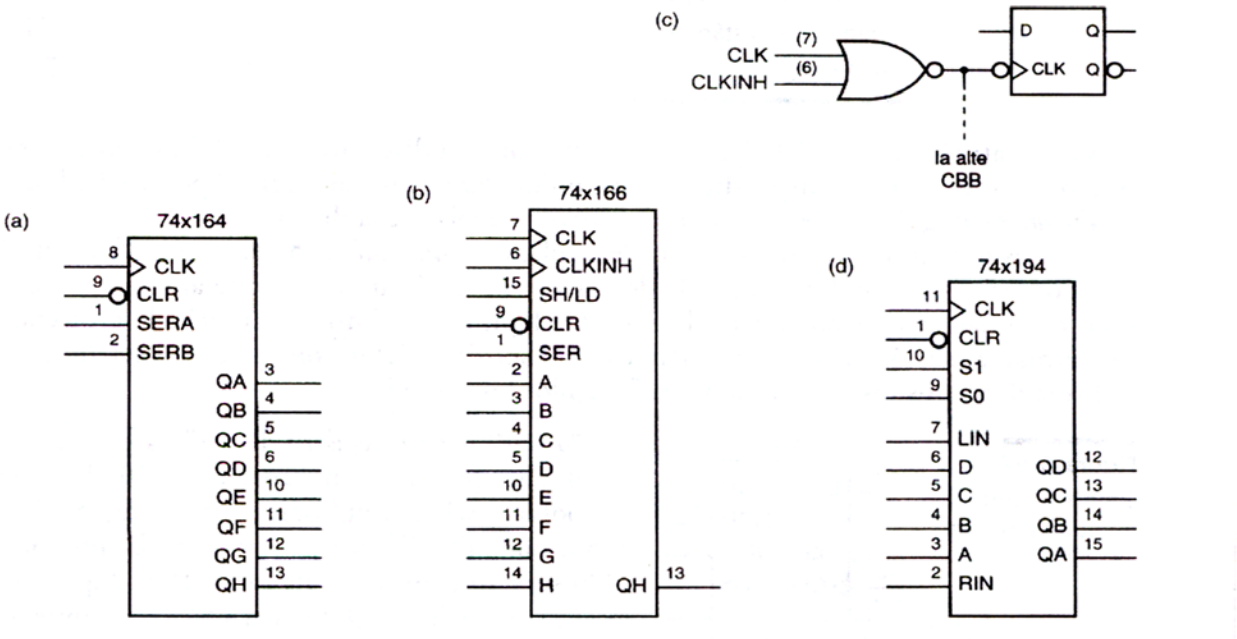


Figura 7-17 Simbolurile logice tradiţionale ale unor registre de deplasare MSI: (a) registrul de deplasare de 8 biţi, serie-paralel, 74x164; (b) registru de deplasare de 8 biţi, paralel-serie, 74x166; (c) circuitul echivalent pentru intrările de tact ale dispozitivului 74x166; (d) registrul de deplasare universal 74x194

Tabelul 7-2 este tabelul funcţiilor dispozitivului 74x194. Este un tabel foarte comprimat, deoarece nu apar coloanele aferente majorităţii intrărilor (**A ... D, RIN, LIN**) sau stărilor curente **QA** ... **QD**. Totuşi, datorită faptului că valoarea din fiecare stare următoare este exprimată ca funcţie de aceste variabile implicite, tabelul defineşte complet funcţionarea dispozitivului '194 pentru toate cele 2^{12} combinaţii posibile stare curentă-intrare şi, cu siguranţă, este preferabil unui tabel cu 4096 de rânduri!

Tabel 7-2 Tabelul funcţiilor pentru registrul de deplasare universal 74x194

	Intrări		Starea următoare			
Funcţia	S1	S0	QA*	QB*	QC*	QD*
Menţinere	0	0	QA	QB	QC	QD
Deplasare la dreapta	0	1	RIN	QA	QB	QC
Deplasare la stânga	1	0	QB	QC	QC	LIN
Încărcare	1	1	A	B	C	D

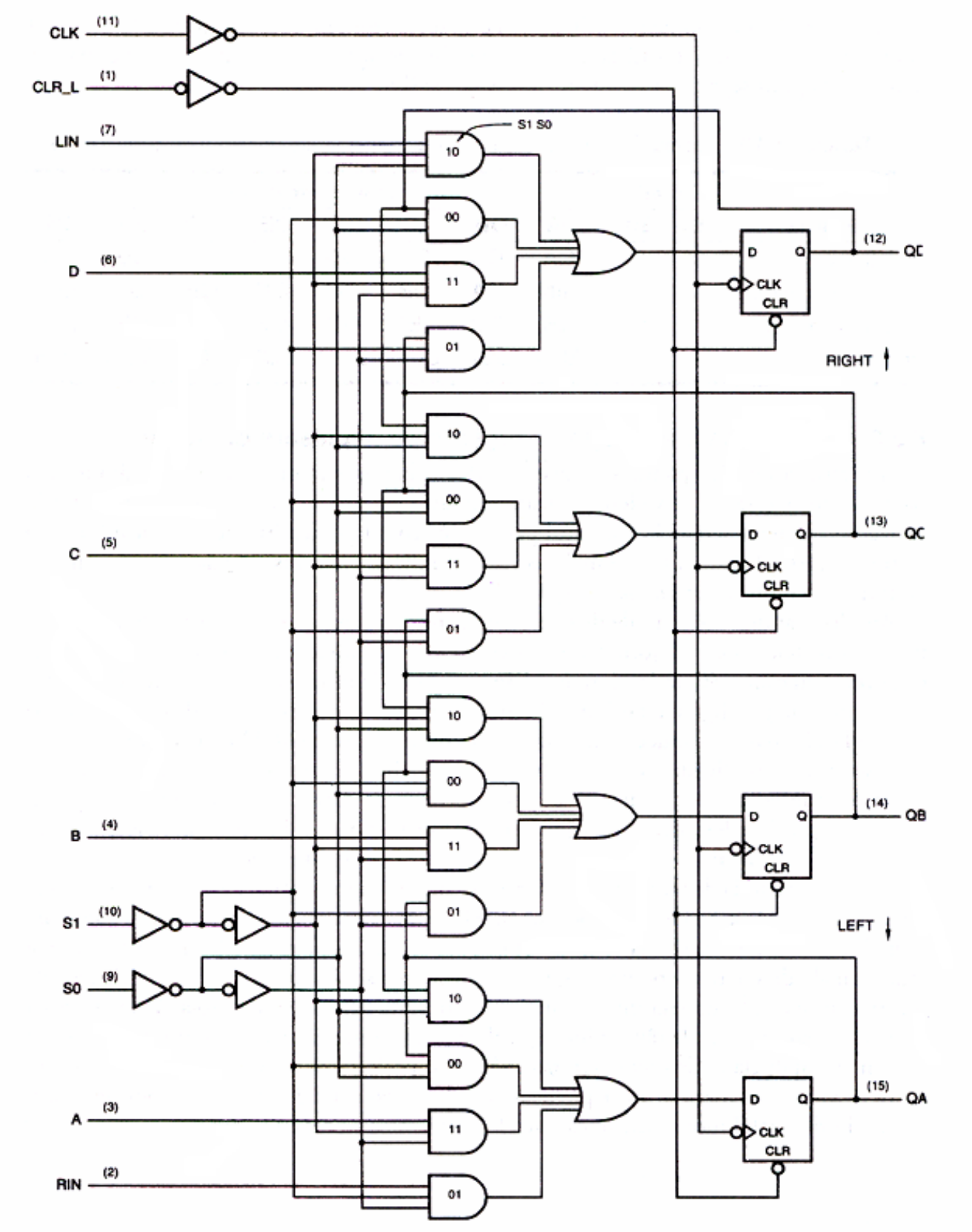


Figura 7-18 Schema logică a registrului de deplasare universal de 4 biţi 74x194, inclusiv numerotarea pinilor la capsula standard „dual in line” cu 16 pini

Observaţi că intrarea **LIN** *(left-in* - intrare-stânga) a dispozitivului '194 este localizată, prin proiect, în partea „dinspre dreapta" a cipului, dar este intrarea serială pentru deplasare la stânga. Analog, **RIN** este intrarea serială pentru deplasare la dreapta.

'194 este denumit uneori registru de deplasare *universal,* deoarece poate fi făcut să funcţioneze ca oricare dintre tipurile de registre de deplasare prezentate anterior (de ex.: unidirecţional, serie-paralel, paralel-serie).

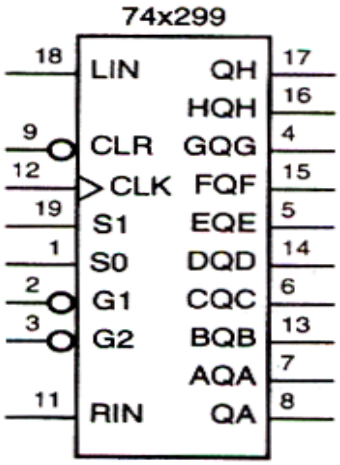


Figura 7-19 Simbolul logic tradiţional pentru 74x299

74x299 este un registru de deplasare universal de 8 biţi, prezentat în capsule cu 20 de pini; simbolul şi schema lui logică apar în fig. 7-19 şi 7-20. Funcţiile şi tabelul de funcţii aferente unui '299 sunt similare celor ale unui '194, după cum se vede în tabelul 7-3. Pentru a face economie de pini, '299 utilizează, pentru intrare şi ieşire, linii bidirecţionale cu trei stări, cum se observă în schema logică. În timpul operaţiilor de încărcare (**S1 SO** = 11), circuitele de comandă ale liniei cu trei stări sunt dezactivate, iar datele se încarcă prin pinii **AQA** ... **HQH**. În alte momente, prin aceiaşi pini se transmit biţii stocaţi, dacă **G1_L** şi **G 2_L** sunt confirmate. Biţii stocaţi la extremităţile din dreapta şi din stânga sunt permanent accesibili la pinii separaţi, exclusiv de ieşire, **QA** şi **QH**.

Tabel 7-3 Tabelul funcţiilor pentru registrul de deplasare universal de 8 biţi 74x299

	Intrări		Starea următoare							
Funcția	S1	S0	QA*	QB*	QC*	QD*	QE*	QF*	QG*	QH*
Menținere	0	0	QA	QB	QC	QD	QE	QF	QG	QH
Deplasare la dreapta	0	1	RIN	QA	QB	QC	QD	QE	QF	QG
Deplasare la stânga	1	0	QB	QC	QED	QD	QF	QG	QH	LIN
Încărcare	1	1	AQA	BQB	CQC	DQD	EQE	FQF	GQG	HQH

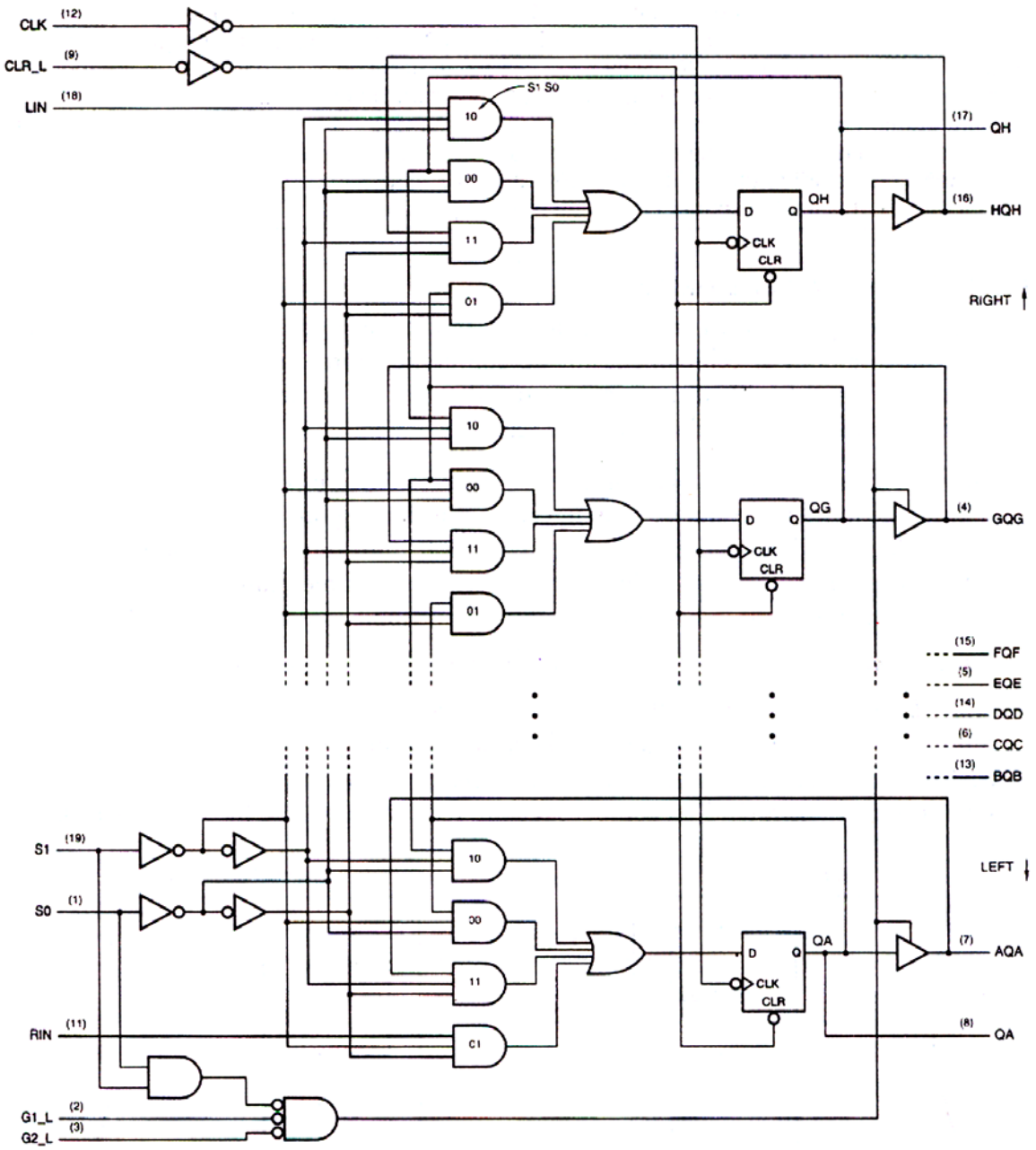


Figura 7-20 Schema logică a registrului de deplasare universal de 8 biţi 74x299, inclusiv numerotarea pinilor la capsula standard „dual in line” cu 20 de pini

7.2.3 Cea mai mare aplicaţie cu registre de deplasare din lume

Aplicaţia cea mai răspândită a registrelor de deplasare este conversia datelor din format paralel în format serial, în vederea transmiterii sau a stocării lor, precum şi readucerea prin conversie a datelor seriale în formatul paralel, în scopul prelucrării sau al afişării lor. Exemplul cel mai frecvent întâlnit de transmisie de date seriale, cu care aveţi de-a face zilnic, este *telefonia digitală.*

Ani de-a rândul, companiile de telefonie au instalat echipamente digitale de comutaţie în centralele telefonice. Majoritatea aparatelor telefonice din locuinţe sunt conectate analogic, printr-un cablu bifilar, la centrala telefonică.

Apoi, la intrarea în centrala telefonică, un convertor analogic-digital eşantionează semnalul digital de voce de 8000 de ori pe secundă (o dată la 125 μs) şi produce o succesiune corespunzătoare de 8000 de octeţi, care reprezintă semnul şi amplitudinea semnalului analogic în fiecare punct explorat. În continuare, vocea dumneavoastră se transmite digital, prin *canale seriale cu* viteza de 64 Kbps, în întreaga reţea de telefonie, pană când este convertit din nou în semnal analogic de un convertor digital-analogic din centrala telefonică de care aparţine postul apelat.

Lăţimea de bandă de 64 Kbps, necesară unui singur semnal digital de voce, este mult *mai mică* decât cea care se poate obţine pe o singură linie de semnal digital sau pe una cu comutare prin CI digitale. De aceea, majoritatea echipamentelor telefonice digitale *multiplexează* mai multe canale de 64 Kbps pe un singur cablu, economisind atât cabluri, cât şi CI digitale pentru comutaţie.

7.2.4 Numărătoare cu registru de deplasare

Obiectul conversiei serie-paralel îl constituie datele, însă registrele de deplasare pot prelucra la fel de bine şi informaţii de altă natură. Unui registru de deplasare i se poate adăuga un circuit logic combinaţional, obţinându-se un automat de stări cu diagrama de stări ciclică. Un asemenea circuit este denumit *numărător cu registru de deplasare.* Spre deosebire de numărătoarele binare, numărătoarele cu registru de deplasare nu numără într-o succesiune binară ascendentă sau descendentă, fiind totuşi utile în numeroase aplicaţii de „comandă”.

7.3 Numărătoare în inel

Cel mai simplu numărător cu registru de deplasare utilizează un registru de deplasare de *n* biţi pentru a obţine un numărător cu *n* stări, numit *numărător în inel.* În fig. 7-21 apare schema logică a unui numărător în inel de 4 biţi. Registrul de deplasare universal 74x194 este cablat astfel încât în mod normal să efectueze deplasarea către stânga. Însă, atunci când semnalul **RESET** este confirmat, se încarcă 0001 (consultaţi tabelul de funcţii al dispozitivului ‘194 - tabelul 7-2). Când semnalul **RESET** este negat, ‘194 efectuează o deplasare spre stânga la fiecare impuls de tact. Intrarea serie **LIN** este conectată la ieşirea „din extremitatea stângă”, astfel că stările următoare sunt *0010, 0100, 1000, 0001, 0010, ...* Deci numărătorul parcurge patru stări unic determinate, înainte de a relua ciclul. Diagrama temporală este cea din fig. 7-22. În general, un numărător în inel de *n* biţi trece, într-un ciclu, prin *n* stări.

La numărătorul în inel din fig. 7-21 apare o problemă importantă: fiabilitatea. Dacă singurul bit de ieşire 1 se pierde din cauza unei disfuncţii temporare de echipament (de ex., din cauza zgomotului), numărătorul trece în

starea 0000 şi rămâne acolo o veşnicie. De asemenea, dacă apare un 1 suplimentar (de ex., dacă se creează starea 0101), numărătorul intră într-un ciclu de stări eronat şi se menţine în el la infinit. Aceste probleme sunt destul de evidente când desenăm diagrama de stări *completă a* circuitului de numărare, care are 16 stări. Aşa cum vedeţi în fig. 7-23, există 12 stări ce nu fac parte din ciclul de numărare normal. Dacă, dintr-o anumită cauză, numărătorul părăseşte ciclul normal de numărare, nu mai revine la acesta.

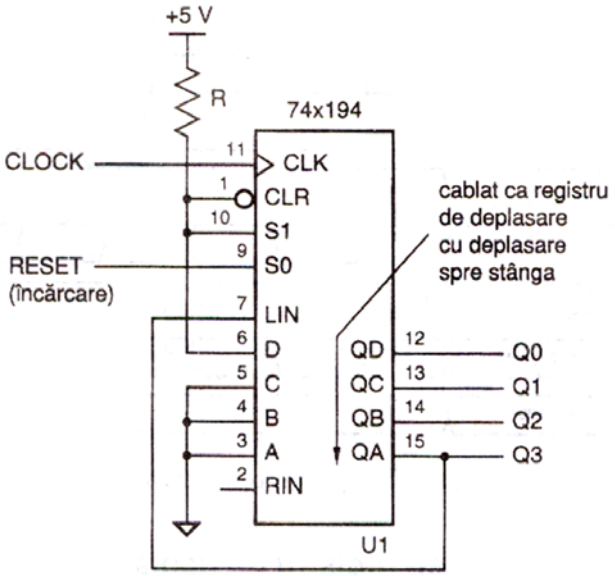


Figura 7-21 Cea mai simplă schemă de numărător în inel de 4 biţi, cu 4 stări şi un singur 1 transmis

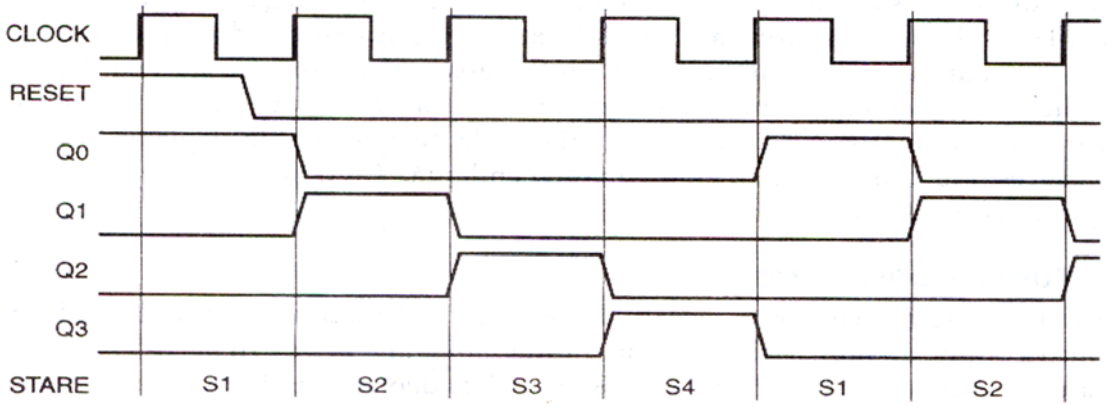


Figura 7-22 Diagrama temporală aferentă unui numărător în inel de 4 biţi

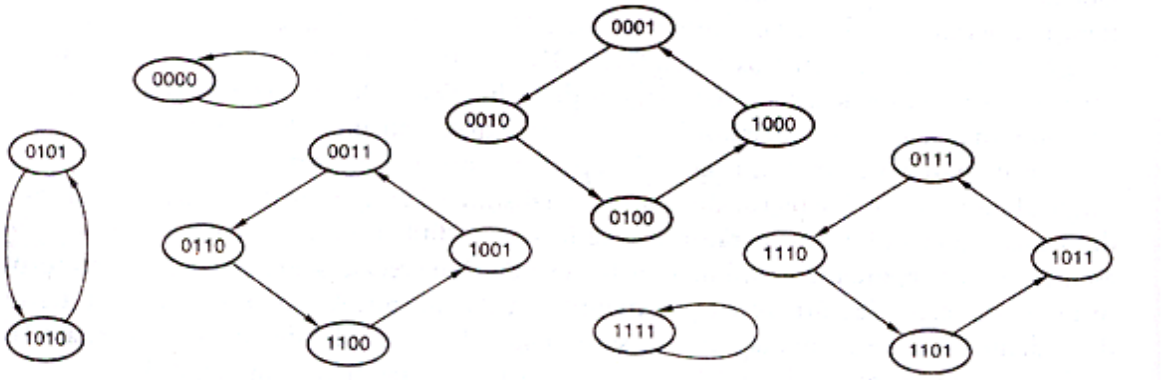


Figura 7-23 Diagramă de stări aferentă unui numărător simplu în inel

Un *numărător cu auto*-corecţie este conceput astfel încât tranziţiile din toate stările anormale să conducă la stări normale.

În fig. 7-24 este prezentat un *numărător în inel cu auto-corecţie.* Circuitul utilizează o poartă **NOR** pentru a introduce un 1 la **LIN** numai când ultimii trei biţi (cei mai puţin semnificativi) sunt 0. Ca urmare, se obţine diagrama de stări din fig. 7-25; din toate stările anormale se revine în ciclul normal. Remarcaţi că la acest circuit nu apare necesitatea unui semnal **RESET** explicit. Indiferent de starea iniţială a registrului de deplasare la conectarea tensiunii de alimentare, după patru impulsuri de tact se ajunge în starea 0001. Deci un semnal explicit de reiniţializare este necesar numai dacă se doreşte ca numărătorul să pornească simultan cu alte dispozitive din acelaşi sistem sau să fie prevăzut cu o stare de pornire cunoscută, în vederea simulării.

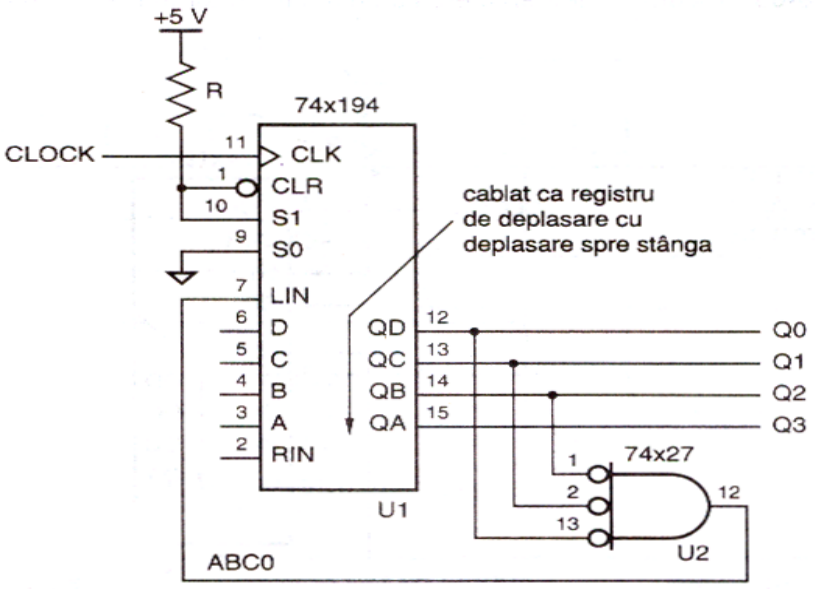


Figura 7-24 Numărător în inel de 4 biţi, cu 4 stări şi un singur 1 transmis, cu auto-corecţie

În general, la un numărător în inel de *n* biţi cu auto-corecţie, se foloseşte o poartă **NOR** cu *n* - 1 intrări, iar dintr-o stare anormală se iese după *n* - 1 impulsuri de tact.

În cazul familiilor de circuite logice **CMOS** şi **TTL**, în general, porţile **NAND** de dimensiuni mari se realizează mai uşor decât porţile **NOR**, deci este mai convenabil să se proiecteze numărătoarele în inel cu auto-corecţie ca în fig. 7-26. Între stările ciclului normal al unui asemenea numărător se transmite un singur 0.

Principalul avantaj al numărătoarelor în inel, în aplicaţii de comandă, este faptul că stările lor apar în forma decodată 1 din *n* direct la ieşirile CBB. Cu alte cuvinte, în fiecare stare este confirmată o singură ieşire de CBB.

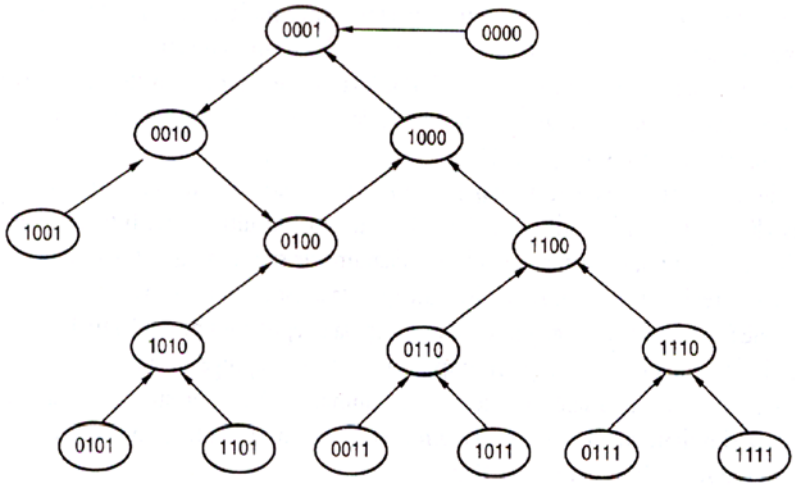


Figura 7-25 Diagramă de stări aferentă unui numărător în inel cu auto-corecţie

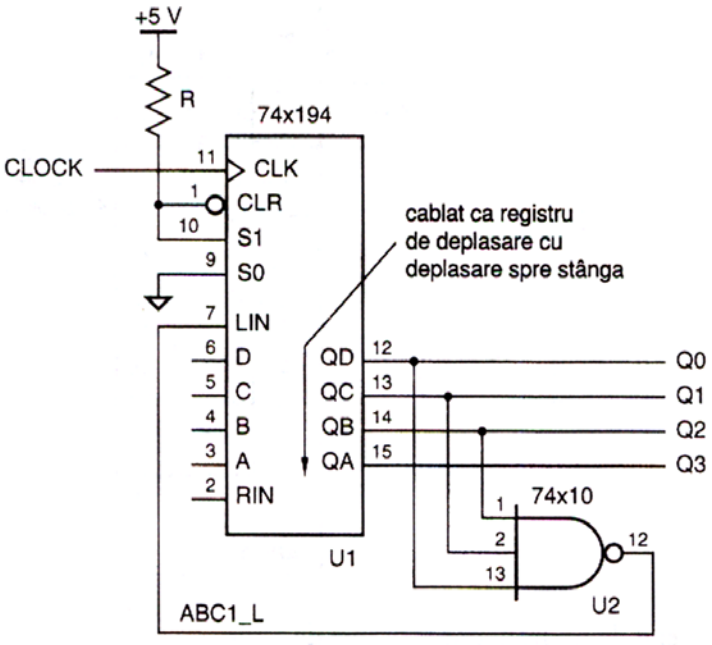


Figura 7-26 Numărător în inel de 4 biţi, cu 4 stări şi un singur 0 transmis, cu auto-corecţie

7.4 Numărătoare Johnson

Un registru de deplasare de *n* biţi, cu complementul ieşirii serie adus înapoi la intrarea serie constitute un numărător cu 2*n* stări cunoscut ca *numărător în inel răsucit, Moebius* sau *Johnson*. În fig. 7-27 se prezintă circuitul de bază al unui numărător Johnson, iar în fig. 7-28, diagrama temporală aferentă lui. Stările normale ale acestui numărător apar în lista din tabelul 7-4. Dacă atât ieşirile directe, cât şi cele complementare ale fiecărui CBB sunt accesibile, fiecare stare normală a numărătorului poate fi decodată cu o poartă **AND** sau **NAND** cu două intrări, cum se arată în tabel.

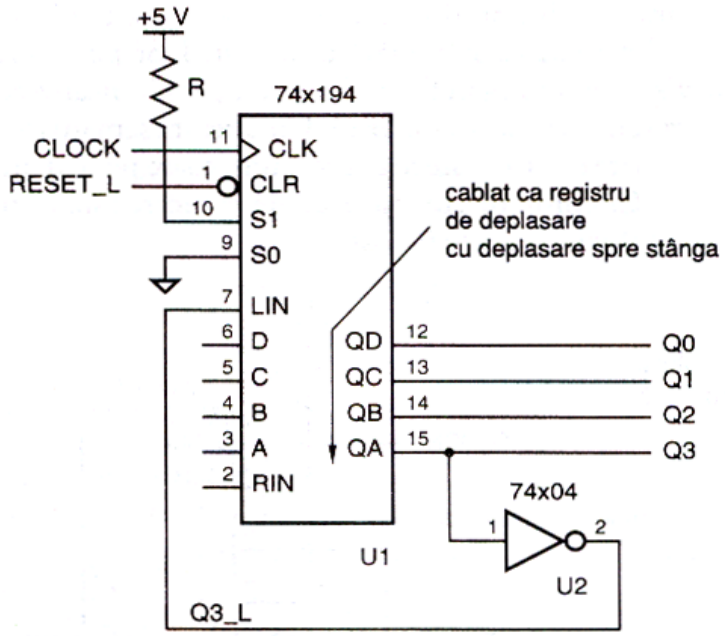


Figura 7-27 Numărător Johnson de bază, de 8 biţi, cu 8 stări

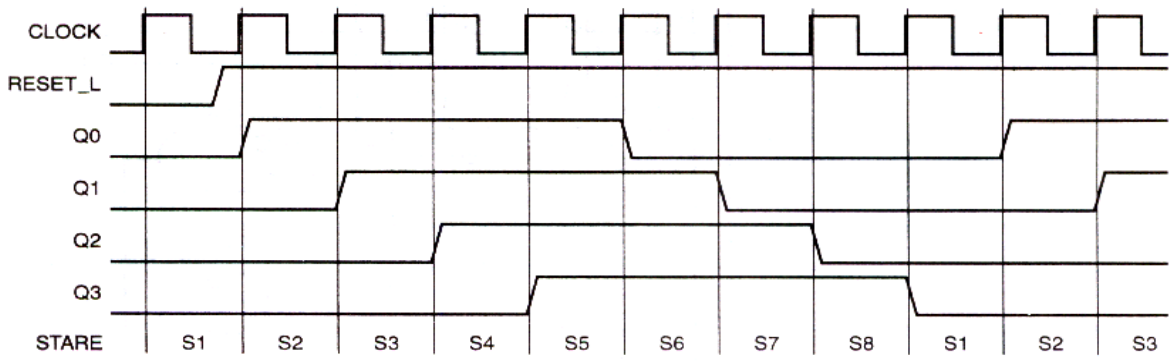


Figura 7-28 Diagramă temporală aferentă unui numărător Johnson de 4 biţi

Tabel 7-4 Stările unui numărător Johnson de 4 biţi

Denumirea stării	Q3	Q2	Q1	Q0	Decodare
S1	0	0	0	0	Q3′ · Q0′
S2	0	0	0	1	Q1′ · Q0
S3	0	0	1	1	Q2′ · Q1
S4	0	1	1	1	Q3′ · Q2
S5	1	1	1	1	Q3 · Q0
S6	1	1	1	0	Q1 · Q0′
S7	1	1	0	0	Q2 · Q1′
S8	1	0	0	0	Q3 · Q2′

Un numărător Johnson de *n* biţi are 2^n - *2n* stări anormale, prezentând deci aceleaşi probleme de fiabilitate ca şi numărătorul în inel. Se poate proiecta un *numărător Johnson cu autocorecţie,* de 4 biţi, ca în fig. 7-29. Acest circuit încarcă starea 0001 ca stare următoare ori de câte ori starea curentă este 0xx0. Un circuit asemănător, cu o singură poartă **NOR** cu două intrări, poate efectua corecţia corespunzătoare pentru un numărător Johnson cu orice număr de biţi. Circuitul de corecţie trebuie să încarce starea 00...01 ca stare următoare ori de câte ori starea curentă este 0x ...x0.

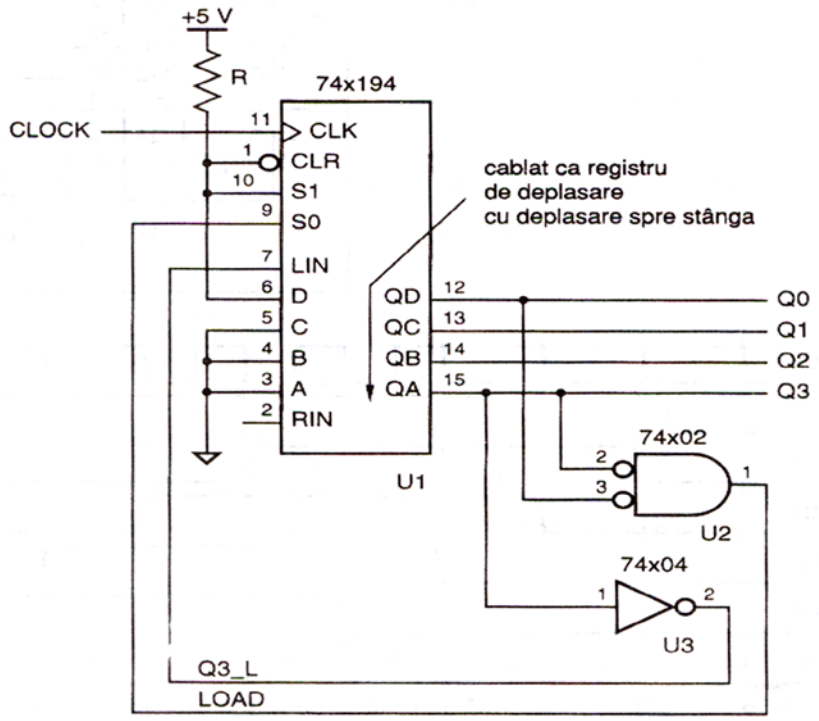


Figura 7-29 Numărător Johnson de 4 biţi, cu 8 stări, cu auto-corecţie

7.5 Numărătoare cu registru de deplasare şi reacţie liniară

Numărătoarele cu registru de deplasare de *n* biţi pe care le-am prezentat până acum au mult mai puţine stări normale decât numărul maxim, de 2^n. Un *numărător cu registru de deplasare şi reacţie liniară (LFSR - Linear Feedback Shift-Register)* de *n* biţi poate avea $2^n - 1$ stări, adică aproape numărul maxim. Un asemenea numărător este numit adesea *generator de secvenţe de lungime maximă.*

Principiul numărătoarelor **LFSR** se bazează pe teoria *câmpurilor finite*, care a fost elaborată de matematicianul francez Evariste Galois (1811-1832) cu puţin înainte de a fi ucis într-un duel cu un adversar politic. Funcţionarea unui numărător **LFSR** corespunde operaţiilor într-un câmp finit cu 2^n elemente.

Figura 7-30 prezintă structura unui numărător **LFSR** de *n* biţi. Intrarea serie a registrului de deplasare este conectată la suma modulo 2 a unor anumiţi biţi de ieşire. Aceste conexiuni de reacţie determină succesiunea stărilor numărătorului. Prin convenţie, ieşirile sunt întotdeauna numerotate şi deplasate în sensul indicat.

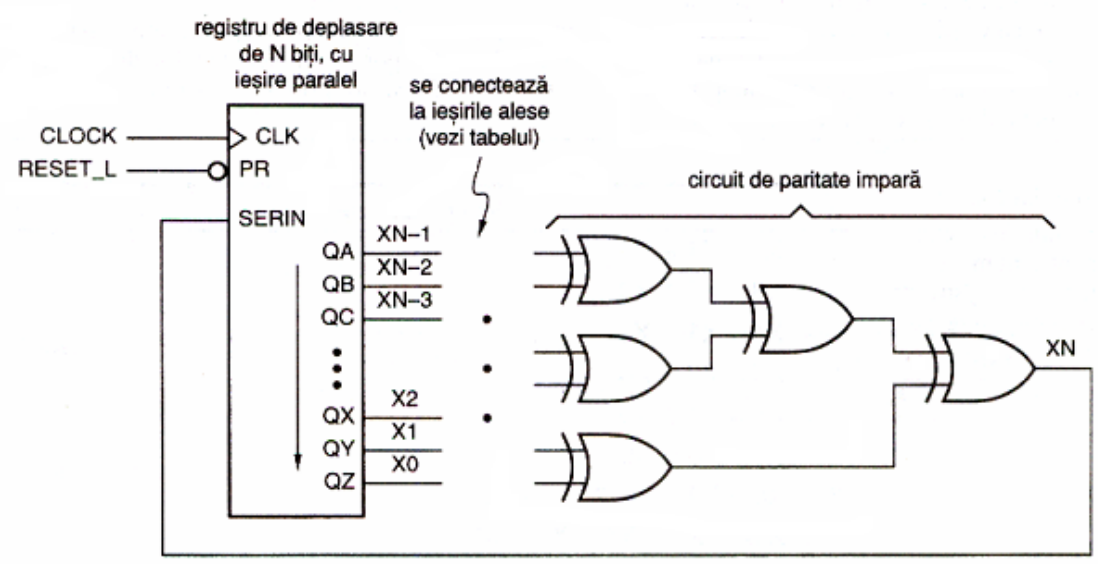


Figura 7-30 Structura generală a unui numărător cu registru de deplasare şi reacţie liniară

Folosind teoria câmpurilor finite, se poate arăta că pentru orice valoare *n* există cel puţin o ecuaţie de reacţie ce determină parcurgerea de către numărător a tuturor celor 2^n - 1 stări diferite de zero, înainte de reluarea ciclului. Aceasta se numeşte *secvenţă de lungime maximă.*

În tabelul 7-5 se găsesc ecuaţiile de reacţie din care rezultă secvenţele de lungime maximă corespunzătoare valorilor *n* alese. Pentru orice valoare *n* mai mare ca *3* există mai multe ecuaţii de reacţie diferite, ce generează secvenţe de lungime maximă.

Un numărător **LFSR** realizat conform fig. 7-30 nu va putea parcurge niciodată toate cele 2^n stări posibile. Indiferent de schema de conectare, starea următoare stării ce conţine numai zerouri este aceeaşi, adică numai zerouri.

Tabel 7-5 Ecuaţii de reacţie pentru numărătoare cu registru de deplasare şi reacţie liniară

n	Ecuația de reacție
2	X2 = X1 ⊕ X0
3	X3 = X1 ⊕ X0
4	X4 = X1 ⊕ X0
5	X5 = X2 ⊕ X0
6	X6 = X1 ⊕ X0
7	X7 = X3 ⊕ X0
8	X8 = X4 ⊕ X3 ⊕ X2 ⊕ X0
12	X12 = X6 ⊕ X4 ⊕ X1 ⊕ X0
16	X16 = X5 ⊕ X4 ⊕ X3 ⊕ X0
20	X20 =X3 ⊕ X0
24	X24 = X7 ⊕ X2 ⊕ X1 ⊕ X0
28	X28 = X3 ⊕ X0
32	X32 = X22 ⊕ X2 ⊕ X1 ⊕ X0

Schema logică a numărătorului **LFSR** de 3 biţi este dată în fig. 7-31. Succesiunea stărilor acestui numărător apare în primele trei coloane ale tabelului 7-6. Pornind din orice stare diferită de zero, în cazul din tabel 100, numărătorul trece prin şapte stări înainte de a ajunge din nou în starea de pornire.

Un numărător **LFSR** se poate modifica astfel încât să aibă 2^n stări prin adăugarea stării ce conţine exclusiv zerouri, aşa cum se pune în evidenţă prin haşurare la numărătorul de 3 biţi din fig. 7-31. Succesiunea de stări obţinută este cea din ultimele trei coloane ale tabelului 7-6. La un numărător **LFSR** de *n* biţi se poate obţine aceeaşi comportare adăugându-se o poartă **XOR** şi una **NOR** cu *n*-1 intrări conectate la toate ieşirile registrului de deplasare, cu excepţia ieşirii **X0**.

Tabel 7-6 Succesiunea stărilor la numărătorul LFSR de 3 biţi din fig. 7-31

Succesiunea inițială			Succesiunea modificată		
X2	X1	X0	X2	X1	X0
1	0	0	1	0	0
0	1	0	0	1	0
1	0	1	1	0	1
1	1	0	1	1	0
1	1	1	1	1	1
0	1	1	0	1	1
0	0	1	0	0	1
1	0	0	0	0	0
.	.	.	1	0	0
.	.	.	.	.	.

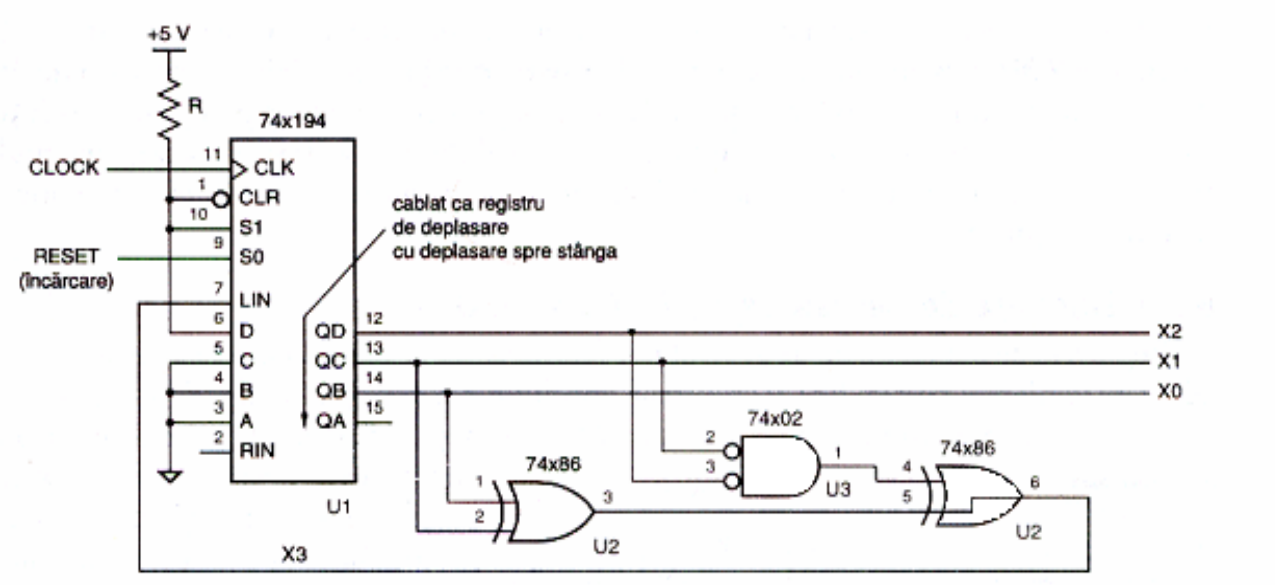


Figura 7-31 Numărător LFSR de 3 biţi; componentele haşurate constituie modificările necesare pentru adăugarea stării ce conţine exclusiv zerouri

La un numărător **LFSR**, stările nu sunt parcurse în ordinea de numărare din binar. Dar aceste numărătoare se utilizează în aplicaţii pentru care o astfel de caracteristică este avantajoasă. Una dintre principalele aplicaţii ale numărătoarelor **LFSR** este generarea semnalelor de intrare pentru testarea circuitelor logice. În majoritatea cazurilor, cu o secvenţă de numărare „pseudo-aleatorie", generată de un numărător **LFSR**, şansele de a depista erorile sunt mai mari decât dacă se foloseşte o secvenţă de numărare în ordinea din binar. Numărătoarele **LFSR** intră şi în circuitele de codare şi decodare aferente unor coduri detectoare şi corectoare de erori, printre care şi codurile **CRC**.

În comunicaţiile de date, numărătoarele **LFSR** se utilizează frecvent pentru a „amesteca" şi a „reconstitui" formatul datelor transmise prin modemuri şi interfeţe de reţea de mare viteză, inclusiv pentru reţeaua Ethernet, la 100 Mbps. Acest lucru se realizează prin combinarea prin **XOR** a ieşirii din **LFSR** cu fluxul de date de utilizator. Chiar dacă fluxul de date de utilizator conţine trenuri lungi de 0 şi 1, prin combinarea cu ieşirea pseudo-aleatorie din **LFSR** se îmbunătăţeşte factorul de umplere al semnalului transmis şi se creează mai multe tranziţii, ceea ce face ca informaţia de sincronizare să poată fi reconstituită la recepţie cu mai multă uşurinţă.

8. Memorii

Orice circuit secvenţial are o memorie de un anumit tip, întrucât stochează câte un bit. Cu toate acestea, folosim cuvântul „memorie” atunci când ne referim la biţi memoraţi într-un mod structurat, de obicei sub formă de tablou bidimensional, în care se accesează simultan biţii de pe un rând.

Capitolul de faţă descrie câteva tipuri diferite de organizare a memoriei. Aceleaşi tipuri de memorii pot fi înglobate în cipuri VLSI de dimensiuni mai mari, unde sunt combinate cu alte circuite, pentru a realiza diverse funcţii utile.

Aplicaţiile circuitelor de memorie sunt numeroase şi diversificate. În unitatea centrală de prelucrare a unui microprocesor *(CPU - Central Processing Unit)* poate fi utilizată o „memorie accesibilă numai pentru citire” *(read-only memory),* în care sunt definiţi primii paşi parcurşi pentru executarea instrucţiunilor din setul de instrucţiuni al CPU. Pe lângă CPU, o „memorie statică” rapidă poate servi ca memorie cache (ascunsă), pentru a păstra instrucţiunile şi datele utilizate recent. Subsistemul principal de memorie al microprocesorului poate conţine sute de milioane de biţi în „memoria dinamică", cea care stochează integral sisteme de operare, programe şi date.

Aplicaţiile memoriilor nu se limitează la microprocesoare, nici măcar la sistemele exclusiv digitale. De exemplu, unele echipamente din sistemul de telefonie publică folosesc memorii accesibile numai pentru citire pentru efectuarea unor diverse transformări ale semnalelor de voce digitizate, iar „memoriile statice” rapide sunt folosite ca „reţea de comutare”, direcţionând vocea digitizată către utilizatori. Multe aparate portabile de ascultat discuri compacte „citesc cu anticipaţie” şi memorează câteva secunde de semnal audio într-o „memorie dinamică”, astfel că aparatul redă sunetul continuu chiar dacă, fizic, funcţionează discontinuu (pentru aceasta este necesară stocarea semnalului audio cu peste 1,4 milioane de biţi pe secundă). Există numeroase exemple de aparatură audio/video modernă, în care memoriile servesc la stocarea temporară a semnalelor digitizate, urmând ca, prin prelucrarea semnalelor digitale, să se obţină performanţe superioare.

8.1 Memoria cu acces numai pentru citire

O *memorie cu acces numai pentru citire (read-only memory – ROM)* este un circuit combinaţional cu *n* intrări şi *b* ieşiri, ca în figura 8-1. Intrările se numesc *intrări de adrese şi* se notează, tradiţional, cu AO, Al, ..., An-1. Ieşirile se numesc *ieşiri de date şi* se notează, de obicei, cu DO, D1, ..., Db-1.

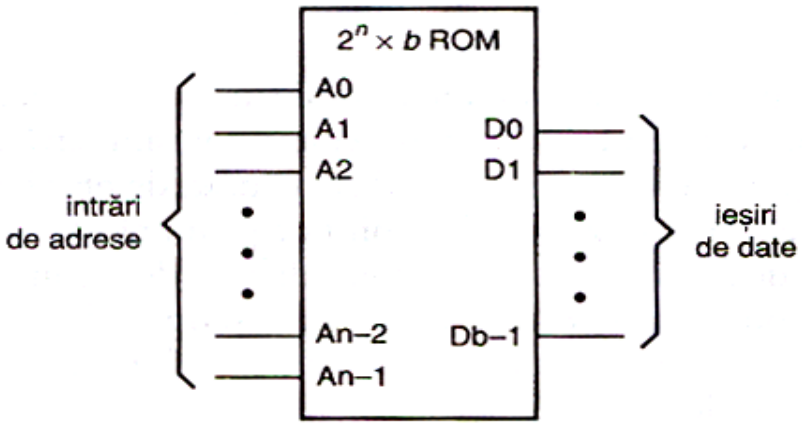


Figura 8-1 Structura de bază a unei memorii ROM 2^nxb

O memorie ROM „stochează" tabelul de adevăr al unei funcţii logice combinaţionale cu *n* intrări şi *b* ieşiri. De exemplu, tabelul 8-1 este tabelul de adevăr al unei funcţii combinaţionale cu 3 intrări şi 4 ieşiri; acesta poate fi stocat într-o ROM 2^3x4 (8x4). Neglijând timpii de propagare, ieşirile de date ale unei ROM sunt, în orice moment, biţii de ieşire din rândul din tabelul de adevăr selectat de intrările de adrese.

Deoarece o ROM este un circuit combinaţional, corect ar fi să spunem că nu este câtuşi de puţin o memorie. Sub aspectul funcţionării circuitelor digitale, o ROM poate fi tratată ca orice alt element logic combinaţional. Cu toate acestea, puteţi considera că informaţiile sunt „stocate" în ROM în procesul de fabricaţie sau de programare.

Deşi considerăm că ROM este un tip de memorie, dintr-un punct de vedere se deosebeşte foarte mult de majoritatea tipurilor de circuite integrate de memorie. ROM este o *memorie nevolatilă,* adică păstrează informaţiile stocate şi în absenţa tensiunii de alimentare.

Tabel 8-1 Tabelul de adevăr al unei funcţii logice combinaţionale cu 3 intrări şi 4 ieşiri

Intrări			Ieşiri			
A2	A1	A0	D3	D2	D1	D0
0	0	0	1	1	1	0
0	0	1	1	1	0	1
0	1	0	1	0	1	1
0	1	1	0	1	1	1
1	0	0	0	0	0	1
1	0	1	0	0	1	0
1	1	0	0	1	0	0
1	1	1	1	0	0	0

8.1.1 Utilizarea ROM pentru funcţii logice combinaţionale „aleatorii"

Tabelul 8-1 este, de fapt, tabelul de adevăr al unui decodor cu 2 intrări şi 4 ieşiri, cu comanda polarităţii de ieşire - funcţie ce poate fi realizată cu porţi

discrete ca în figura 8-2. Prin urmare, există două căi de a construi un decodor: cu porţi discrete sau cu o ROM 8x4 ce conţine tabelul de adevăr, ca în figura 8-3.

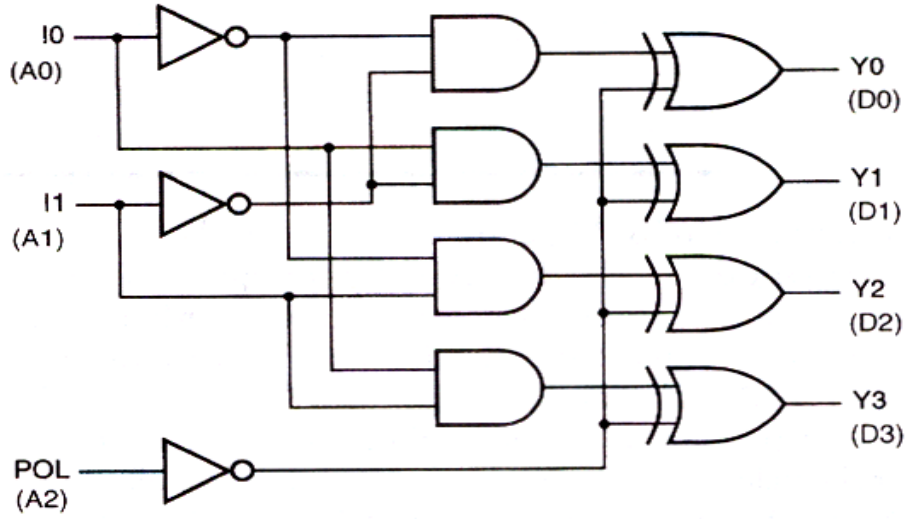


Figura 8-2 Decodor cu 2 intrări şi 4 ieşiri, cu comanda polarităţii de ieşire

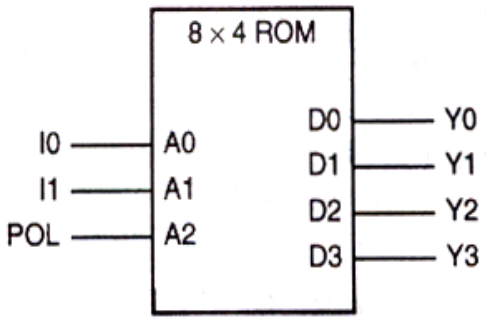


Figura 8-3 Modul în care se conectează o ROM 8x4, ce stochează tabelul 8-1, pentru a se obţine un decodor cu 2 intrări şi 4 ieşiri

Modul de asociere dintre intrările şi ieşirile decodorului şi intrările şi ieşirile ROM, din figura 8-3, rezultă din modul în care a fost întocmit tabelul de adevăr din tabelul 8-1. Prin urmare, implementarea fizică, cu ROM, a decodorului nu este unică. Cu alte cuvinte, putem să scriem rândurile sau coloanele tabelului de adevăr într-o altă ordine şi atunci vom folosi o ROM diferită fizic de cea precedentă, realizând însă aceeaşi funcţie logică, prin simpla asociere a semnalelor decodorului cu alte intrări şi ieşiri ale ROM. Putem la fel de bine să considerăm că schimbăm denumirile intrărilor de adrese şi ale ieşirilor de date ale ROM.

Un alt exemplu simplu de funcţie ce poate fi realizată cu ROM este înmulţirea a două numere binare de 4 biţi, neprecedate de semn.

Conţinutul unei ROM se descrie, în mod normal, printr-un fişier ce conţine câte o valoare pentru fiecare adresă din ROM. Avantajul proiectării cu

ROM este faptul că, de obicei, putem scrie un program simplu, într-un limbaj de nivel înalt, pentru a calcula ce trebuie stocat în ROM.

8.1.2 Structura internă a ROM

Mecanismul prin care ROM „stochează" informaţii variază de la o tehnologie de ROM la alta. În majoritatea cazurilor, prezenţa sau absenţa unei diode sau a unui tranzistor constituie diferenţa dintre un 0 şi un 1.

În figura 8-4 este dată schema unei ROM 8x4 primitive, pe care o puteţi construi singuri cu un decodor MSI şi câteva diode. Intrările de adrese selectează ieşirea decodorului ce urmează a fi confirmată. Fiecare ieşire a decodorului este numită *linie de cuvânt,* deoarece ea selectează un rând sau un cuvânt din tabelul stocat în ROM. În figură este ilustrată situaţia cu **A2...A0 = 101** şi **ROW5_L** confirmat.

Fiecare linie verticală din figura 8-4 se numeşte *linie de bit,* deoarece corespunde unui bit de ieşire din ROM. 0 linie de cuvânt confirmată impune trecerea în **LOW** a unei linii de bit dacă între linia de cuvânt şi cea de bit este conectată o diodă. În rândul 5 există o singură diodă, iar linia de bit corespunzătoare acesteia (**D1_L**) este forţată în **LOW.** Liniile de bit sunt urmate de circuite tampon inversoare, pentru a se obţine ieşirile ROM **D3 ... DO = 0010**, în cazul de faţă.

În circuitul ROM din figura 8-4, fiecare intersecţie dintre o linie de cuvânt şi o linie de bit corespunde unui bit de „memorie". Dacă la intersecţie se află o diodă, se stochează un l, în caz contrar se stochează un 0.

În figura 8-5 este prezentată o variantă de ROM având ca elemente de stocare tranzistoare MOS. Similar se pot realiza şi module ROM cu tranzistoare bipolare. Variantele cu tranzistore asigură un prag al nivelului de zgomot mai bun în comparaţie cu variantele cu diode.

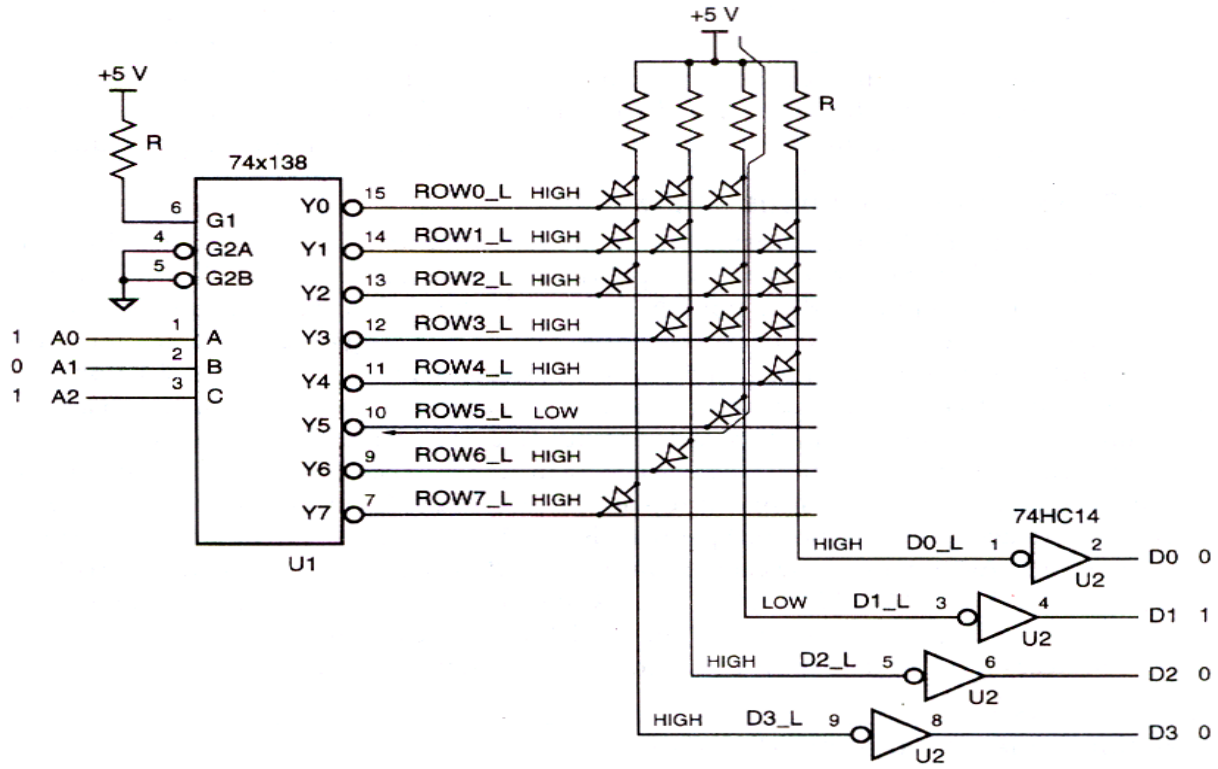


Figura 8-4 Schemă logică de ROM 8x4 simplă, cu diode

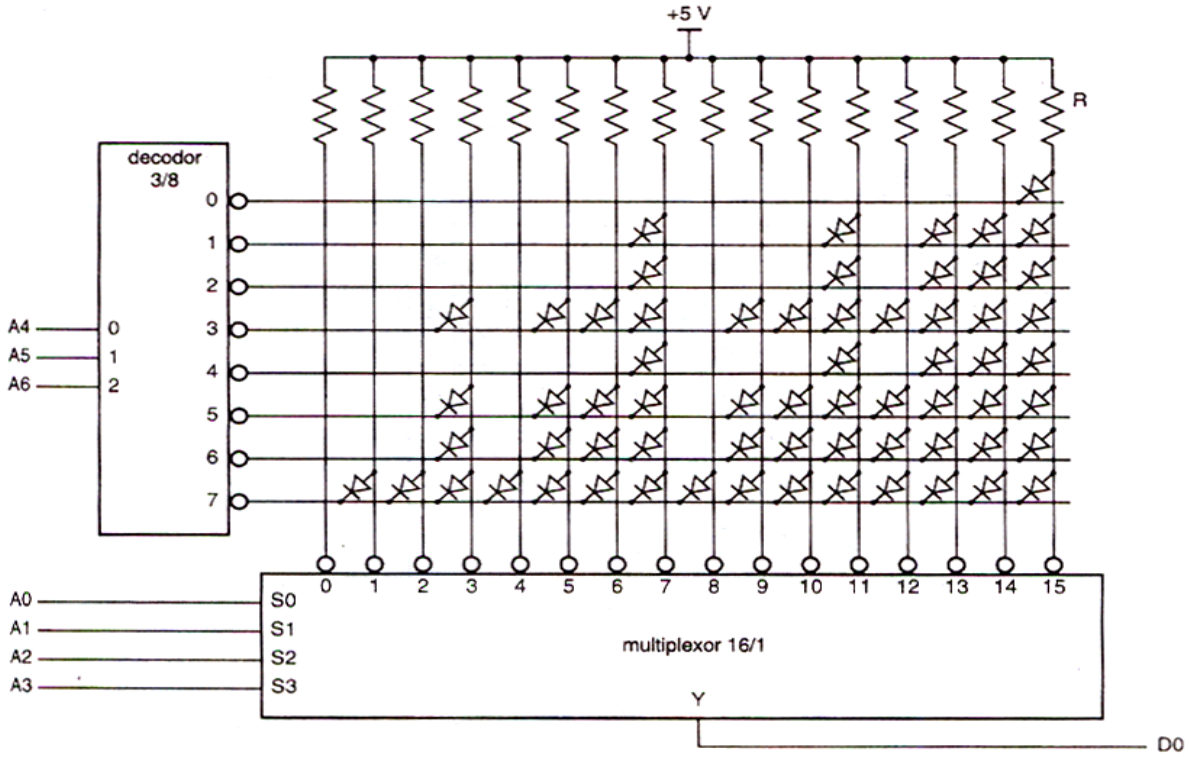


Figura 8-5 Structură internă a ROM 128x1 cu decodare bidimensională

8.1.3 Decodarea bidimensională

Ca principiu, decodarea bidimensională constă în aranjarea celulelor ROM într-o matrice de dimensiuni cât mai egale (cât mai apropiată de un pătrat). Ca exemplu, în figura 8-5 este prezentată o structură internă posibilă pentru o ROM 128x1. Cei trei biţi de adrese de ordine superioare, **A6...A4**, servesc la selectarea unui rând. Pe fiecare rând se stochează 16 biţi, începând de la adresele (**A6**, **A5**, **A4**, **0**, **0**, **0**, **0**). Când la intrările ROM se aplică o adresă, toţi cei 16 biţi din rândul selectat se „citesc" în paralel, pe liniile de bit. Un multiplexor cu 16 intrări selectează bitul de date dorit conform biţilor de adrese de ordine inferioare.

Decodarea bidimensională permite construirea unei ROM 128x1 dintr-un decodor cu 3 intrări şi 8 ieşiri şi un multiplexor cu 16 intrări (cu o complexitate comparabilă cu cea a unui decodor cu 4 intrări şi 16 ieşiri). O ROM 1Mx1 poate fi construită dintr-un decodor cu 10 intrări şi 1024 de ieşiri şi un multiplexor cu 1024 de intrări, ceea ce nu este simplu, dar este mult mai simplu decât varianta unidimensională.

Pe lângă faptul că reduce complexitatea decodării, decodarea bidimensională mai prezintă un avantaj: cipul rezultat are dimensiunile apropiate de cele ale unui pătrat, lucru important pentru fabricare şi încapsulare. Un cip conţinând fizic o matrice 1Mx1 ar fi foarte lung şi subţire, neputând fi construit economic.

La ROM cu mai multe ieşiri de date, matricele de stocare corespunzătoare fiecărei ieşiri de date pot fi îngustate, în scopul realizării unei dispuneri pe cip de formă aproximativ pătrată. De exemplu, figura 8-6 prezintă o posibilă dispunere pentru o ROM 32Kx8.

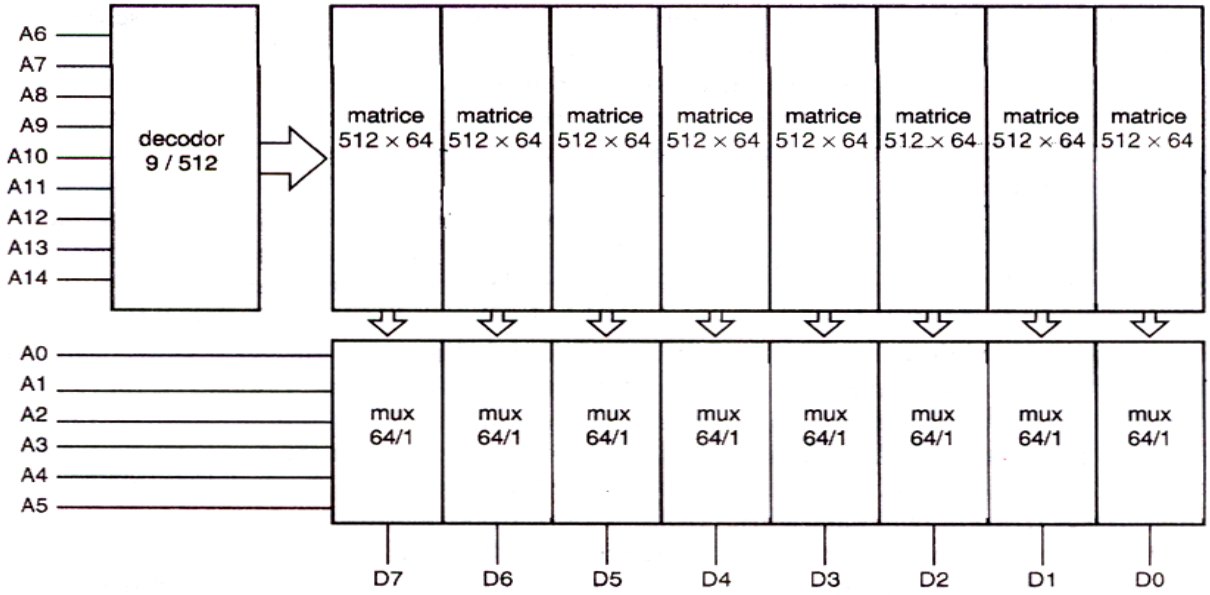


Figura 8-6 Posibilă dispunere la o ROM 32Kx8

8.1.4 Tipuri de ROM comercializate

Primele module ROM erau construite cu diode discrete. Memoriile ROM moderne se fabrică sub formă de CI monocip. Pentru „programarea" informaţiilor stocate în ROM se utilizează diverse metode, sistematizate în tabelul 8-2.

Tabel 8-2 Tipuri de ROM comercializate

Tipul	Tehnologia	Perioada de citire	Perioada de scriere	Observaţii
ROM cu mască	NMOS, CMOS	10 ... 200 ns	4 săptămâni	O singură scriere; putere mică
ROM cu mască	Bipolară	< 100 ns	4 săptămâni	O singură scriere; putere mare; densitate scăzută
PROM	Bipolară	< 100 ns	10 ... 50 μs/byte	O singură scriere; putere mare; fără cost suplimentar de mască
EPROM	NMOS, CMOS	25 ... 200 ns	10 ... 50 μs/byte	Reutilizabile; putere mică; fără cost suplimentar de mască
EEPROM	NMOS	50 ... 200 ns	10 ... 50 μs/byte	Limită de scriere/locaţie: 10.000 ... 100.000

Cele mai multe ROM integrate din primele generaţii erau *ROM programabile cu mască* (sau, pe scurt, *ROM cu mască).* Un asemenea dispozitiv se programează conform unei grile de conectare/neconectare, existentă într-una din *măştile* utilizate în procesul de fabricaţie a CI. Pentru a programa ROM sau a înscrie informaţiile în ea, beneficiarul furnizează fabricantului, pe o dischetă sau pe un alt mediu de transfer, o listă a ceea ce doreşte să fie conţinut în ROM. Fabricantul foloseşte informaţiile pentru a realiza una sau mai multe măşti adaptate cerinţelor, iar ROM fabricate cu aceste măşti respectă grila impusă.

Costul suplimentar al măştilor necesare fabricării unor ROM cu mască se ridică la câteva mii de dolari, pentru „particularizarea" conform cerinţelor. Din cauza costului măştilor şi a intervalului de câteva săptămâni, necesar, de obicei, pentru obţinerea cipurilor programate, ROM cu mască se utilizează astăzi numai în aplicaţiile de volum mare, pentru cele de volum mic existând variante mai economice.

O memorie programabilă cu acces numai pentru citire (programmable read-only memory - PROM) se aseamănă cu o ROM cu mască, însă utilizatorul poate să stocheze valori de date (adică „să programeze o PROM") în doar câteva minute, folosind un *programator de PROM.* Cipurile de PROM au, din fabricaţie, toate diodele sau tranzistoarele „conectate". Aceasta înseamnă că toţi biţii sunt fixaţi la o anumită valoare, de obicei 1. Cu ajutorul programatorului de PROM, biţii doriţi se fixează la valoarea inversă. La PROM bipolare, operaţia se realizează prin vaporizarea micilor *punţi fuzibile* din interior, fiecare punte corespunzând câte unui bit. Vaporizarea unei punţi se efectuează selectând-o prin intermediul liniilor de adrese şi de date ale PROM şi aplicând apoi

dispozitivului un impuls de tensiune mare (10 ... 30 V), pe un pin de intrare special.

Primele PROM bipolare aveau probleme de fiabilitate. Uneori, biţii stocaţi se modificau din cauza vaporizării incomplete a punţilor, care „se formau la loc”; alteori apăreau defecte intermitente din cauza resturilor de punţi ce se deplasau liber în interiorul capsulelor de CI. Aceste probleme au fost însă rezolvate, iar acum tehnologia cu punţi fuzibile este fiabilă şi se foloseşte nu doar la PROM bipolare, ci şi la circuitele PLD bipolare.

O memorie programabilă, cu ştergere şi acces numai pentru citire (erasable programmable read-only memory- EPROM) se programează ca şi PROM, dar poate fi şi „ştearsă”, aducându-se din nou la starea cu toţi biţii 1, prin expunere la radiaţii ultraviolete. La EPROM se foloseşte o altă tehnologie, numită „MOS cu porţi flotante”.

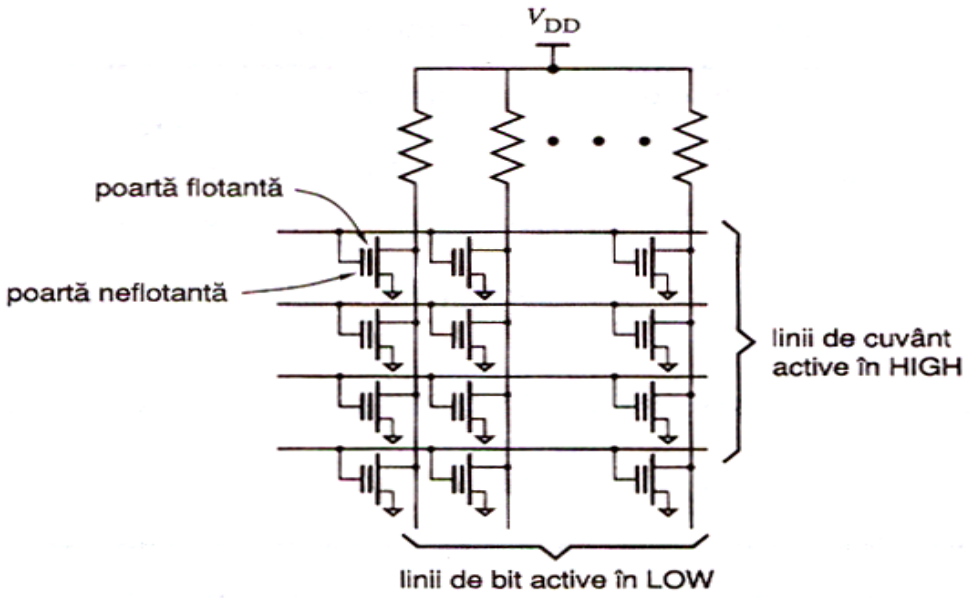


Figura 8-7 Matrice de stocare în EPROM cu tranzistoare MOS cu poartă flotantă

După cum vedeţi în figura 8-7, în fiecare locaţie de bit a unei EPROM se află câte un *tranzistor MOS cu poartă flotantă.* Fiecare tranzistor are două porţi. Poarta „flotantă” nu este conectată, fiind înconjurată de un material izolator cu impedanţă extrem de mare. Pentru a programa o EPROM, programatorul aplică o tensiune mare pe poarta neflotantă corespunzătoare fiecărei locaţii de bit unde trebuie stocat un 0. Astfel se produce o străpungere temporară a materialului izolator, permiţând acumularea unei sarcini negative pe poarta flotantă. Sarcina negativă se păstrează şi după îndepărtarea tensiunii mari. În timpul operaţiilor de citire ulterioare, sarcina negativă împiedică deschiderea tranzistorului MOS atunci când acesta este selectat.

Fabricanţii de EPROM „garantează” că un bit programat corespunzător îşi menţine 70% din sarcină cel puţin 10 ani, chiar dacă dispozitivul este depozitat la 125°C, deci EPROM se încadrează, fără nici un dubiu, în categoria „memorii

nevolatile". Dar EPROM se pot şi şterge. Materialul izolator din jurul porţii flotante devine uşor conductor dacă este expus la radiaţie ultravioletă cu o anumită lungime de undă. Deci EPROM se pot şterge prin expunerea cipurilor la radiaţii ultraviolete, de obicei 5...20 de minute. Capsulele cipurilor de EPROM sunt prevăzute, în mod normal, cu o fereastră de cuarţ prin care cipul poate fi expus la radiaţia de ştergere.

Probabil cea mai uzuală aplicaţie a EPROM este stocarea programelor din sistemele cu microprocesor. EPROM se utilizează de obicei în perioada elaborării programelor, când programele sau alte informaţii din EPROM trebuie modificate de mai multe ori, pentru depanare. Cu toate acestea, ROM şi PROM sunt, în general, mai ieftine decât EPROM de capacităţi similare. Ca urmare, după finalizarea unui program, în producţie se pot folosi ROM sau PROM, pentru reducerea costurilor. Multe PROM din momentul actual sunt, de fapt, EPROM cu capsule ieftine, fără fereastră de cuarţ; uneori, ele sunt numite *ROM uniprogramabile (one-time programmable - OTP).*

O memorie programabilă, cu ştergere electrică şi acces numai pentru citire (electrically erasable programmable read-only memory -EEPROM) se aseamănă cu EPROM, însă fiecare bit stocat poate fi şters separat, prin metode electrice. Porţile flotante din EEPROM sunt înconjurate de un strat izolator mult mai subţire şi pot fi şterse prin aplicarea unei tensiuni de polaritate opusă tensiunii de încărcare, aplicată pe porţile neflotante. La EEPROM de capacitate mare (de 1 Mbit sau mai mare) ştergerea se poate face simultan numai în blocuri de dimensiuni fixe, de obicei de 128...512 Kbit (16...64 Kbyte). Asemenea memorii mai sunt numite EPROM zonale sau memorii zonale (flash), deoarece ştergerea se face „pe zone".

Aşa cum se indică în tabelul 8-2, programarea sau „scrierea" unei locaţii de EEPROM durează mult mai mult decât citirea ei, deci EEPROM nu poate înlocui memoriile cu citire/scriere, pe care le vom prezenta mai târziu, tot în capitolul de faţă. De asemenea, din cauză că stratul izolator este atât de subţire, acesta se poate uza în urma operaţiilor de programare repetate. De aceea, EEPROM pot fi reprogramate doar de un număr limitat de ori, cam de 10.000 de ori pentru o locaţie. Prin urmare, EEPROM se utilizează, de obicei, pentru stocarea datelor ce trebuie menţinute în lipsa alimentării echipamentelor, dar care nu se modifică foarte des, ca de pildă, datele implicite de configurare pentru un calculator.

8.1.5 Intrările de comandă şi temporizarea ROM

Adesea este necesar să se conecteze ieşirile unei ROM la o magistrală cu trei stări, cu diverse dispozitive ce pot comanda magistrala în diferite momente. De aceea, majoritatea cipurilor de ROM de pe piaţă sunt prevăzute cu ieşiri de date cu trei stări şi cu o *intrare de activare a ieşirilor (OE - output enable),* care trebuie confirmată pentru ca ieşirile să fie activate.

Numeroase aplicaţii cu ROM, în special stocarea programelor, sunt realizate cu mai multe ROM conectate la o magistrală, o singură ROM comandând magistrala la un moment dat. Cele mai multe ROM sunt prevăzute cu o *intrare de selectare a cipurilor (CS - chip select),* pentru simplificarea schemelor unor astfel de sisteme. În afară de **OE**, şi intrarea **CS** a unei ROM trebuie confirmată pentru ca ieşirile cu trei stări să fie activate.

Figura 8-8 arată în ce mod se utilizează intrările **CS** şi **OE**, în interiorul unei ROM obişnuite.

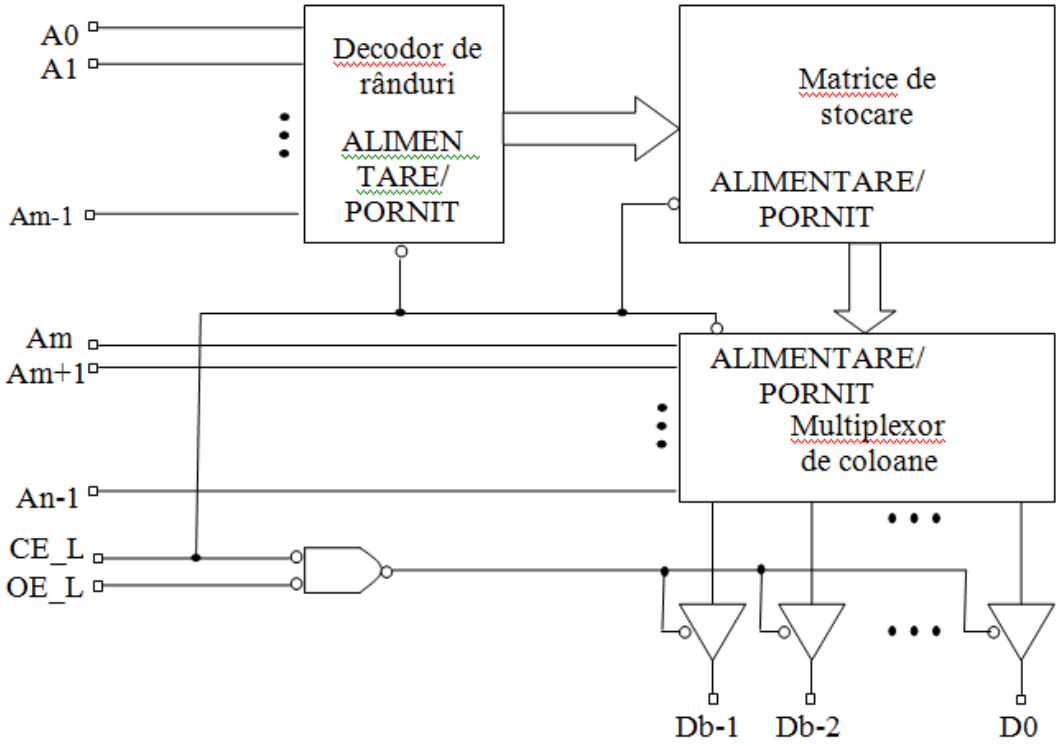


Figura 8-8 Structură internă cu ROM, cu indicarea utilizării intrărilor de comandă

În ceea ce priveşte caracteristicile temporale ale unei ROM obişnuite, trebuie să avem în vedere următorii parametrii:

- *Timpul de acces de la adresare (t_{AA}).* Timpul de acces de la adresare, pentru o ROM, este timpul de propagare de la intrările de adrese stabile la ieşirile de date validate. Când proiectanţii vorbesc despre „un ROM de 100 ns”, se referă, de obicei, la acest parametru.

- *Timpul de acces de la selectarea cipului (t_{ACS}).* Timpul de acces de la selectarea cipului, la o ROM, este timpul scurs din momentul confirmării intrării **CS** până la validarea ieşirilor de date. La unele cipuri, acesta este mai îndelungat decât timpul de acces de la adresare, deoarece cipului îi este necesar un anumit interval de timp pentru conectarea la tensiunea de alimentare. La alte cipuri, acest timp este mai sturt, deoarece **CS** comandă numai activarea ieşirilor.

- *Timpul de activare a ieşirilor (t_{OE}).* Acest parametru are, de obicei, valoarea mult mai mică decât timpul de acces. La ROM, timpul de

activare a ieşirilor este timpul scurs din momentul în care atât **OE**, cât şi **CS** sunt confirmate, până când circuitele de comandă a ieşirilor cu trei stări au ieşit din starea **Hi-Z**. În acel moment, în funcţie de durata de stabilitate a intrărilor de adrese, datele de ieşire pot fi validate sau nu.

- *Timpul de dezactivare a ieşirilor (t_{OZ}).* La ROM, timpul de dezactivare a ieşirilor este timpul scurs din momentul în care atât **OE**, cât şi **CS** sunt negate, până când circuitele de comandă a ieşirilor cu trei stări au trecut în starea **Hi-Z**.

- *Timpul de menţinere a ieşirilor (t_{OH}).* La ROM, timpul de menţinere a ieşirilor este timpul cât ieşirile rămân validate după o modificare a intrărilor de adrese sau după negarea uneia dintre intrările **OE_L** sau **CS_L**.

Ca şi la alte componente, fabricantul precizează valorile maxime şi, uneori, şi pe cele tipice, ale parametrilor temporali. De obicei, pentru t_{OE} şi t_{OH} se dau şi valorile minime. Valoarea t_{OH} minimă este, în majoritatea cazurilor, 0; cu alte cuvinte, timpul minim de propagare prin circuitele logice combinaţionale din ROM este 0.

8.1.6 Aplicaţii ale ROM

Cea mai des întâlnită aplicaţie a ROM este stocarea programelor în sistemele cu microprocesoare. Dar există şi numeroase aplicaţii în care prin ROM se poate implementa, cu costuri scăzute, o funcţie logică combinaţională complexă sau „aleatorie”.

Un alt domeniu în care sunt folosite circuite ROM este reţeaua de telefonie digitală. Când un semnal analogic de voce este preluat într-un sistem obişnuit de telefonie digitală, el este eşantionat de 8000 de ori pe secundă şi transformat într-o succesiune de octeţi ce reprezintă semnalul analogic în fiecare punct de eşantionare. În acest domeniu, o simplă ROM ieftină poate înlocui cu succes circuitul de atenuare digital cu componente discrete, mult mai complex.

O altă aplicaţie a ROM în telefonia digitală este circuitul digital de teleconferinţă. În reţeaua de telefonie analogică, legăturile de teleconferinţă între trei sau mai multe puncte se realizează destul de simplu. Este suficient să conectaţi împreună cablurile de telefonie analogică şi obţineţi o joncţiune de sumare, astfel încât fiecare participant îi poate auzi pe toţi ceilalţi.

Desigur că în reţeaua digitală s-ar produce un haos dacă s-ar conecta împreună semnalele digitale de ieşire. În cazul unui circuit digital de teleconferinţă se foloseşte un sumator digital, care generează eşantioane de ieşire corespondenţă sumei eşantioanelor de intrare. Funcţia acestui circuit destul de complex poate fi realizată cu o singură ROM.

Circuitele realizate cu ROM, pe lângă uşurinţa şi rapiditatea proiectării, prezintă şi alte avantaje importante, fiind, în general mai rapide decât cele mai multe dispozitive MSI/SSI şi PLD şi chiar decât FPGA sau cipurile LSI cu

comandă specială, fabricate cu tehnologii comparabile. Un alt avantaj semnificativ este şi faptul că funcţia unei ROM se modifică simplu, doar prin schimbarea matricei de stocare, fără a necesita modificarea conexiunilor exterioare.

Nu trebuie uitat faptul că preţurile de ROM şi alte dispozitive logice structurate sunt în scădere continuă, făcând ca utilizarea acestora să devină mai economică, iar densităţile lor sunt în creştere, ducând la lărgirea spectrului de probleme ce se pot rezolva cu un singur cip.

Dar circuitele construite cu ROM au şi câteva dezavantaje: pentru funcţii de complexitate redusă sau medie, circuitele cu ROM pot să fie mai costisitoare, să consume mai multă putere sau să fie mai lente decât cele realizate cu câteva dispozitive SSI/MSI şi PLD sau cu FPGA de mici dimensiuni.

8.2 Memoria de citire/scriere

Denumirea de *memorie de citire/scriere (read/write memory- RWM)* este dată matricelor de memorie în care se pot stoca şi din care se pot prelua informaţii în orice moment. Majoritatea memoriilor de citire/scriere utilizate la ora actuală în sistemele digitale sunt *memorii cu acces aleatoriu (random-access memory-RAM),* ceea ce înseamnă că timpul de citire sau de scriere al unui bit din memorie este independent de locaţia din RAM a bitului. Din acest punct de vedere, ROM sunt tot memorii cu acces aleatoriu, însă denumirea „RAM" este folosită, în general, pentru a desemna doar memoriile de citire/scriere cu acces aleatoriu.

Într-o *RAM statică (SRAM),* o dată ce un cuvânt a fost înscris într-o locaţie, el rămâne stocat acolo atâta timp cât cipul este alimentat, cu excepţia cazului în care în acea locaţie se înscrie un alt cuvânt. Într-o *RAM dinamică (DRAM),* datele stocate în fiecare locaţie trebuie reîmprospătate periodic prin citirea lor şi o nouă înscriere în acelaşi loc; dacă nu se procedează aşa, datele dispar. În secţiunea de faţă vom prezenta ambele tipuri de memorie.

Majoritatea RAM „îşi pierd memoria" atunci când li se întrerupe alimentarea; acestea sunt o formă de *memorie volatilă.* Altele „îşi păstrează memoria" şi în absenţa tensiunii de alimentare; acestea sunt numite *memorii nevolatile.* Exemple de RAM nevolatile sunt vechile memorii cu miez magnetic şi memoriile statice CMOS moderne, prezentate în capsule de dimensiuni foarte mari, ce includ şi o baterie cu litiu cu o durată de viaţă de 10 ani. De curând au apărut RAM nevolatile *fero-electrice;* aceste dispozitive combină elemente magnetite şi electronice pe un singur cip de CI, care îşi păstrează starea şi în absenţa tensiunii de alimentare, exact ca vechile memorii cu miez magnetic.

8.3 RAM statice

8.3.1 Intrări şi ieşiri la RAM statice

Ca şi ROM, RAM au intrări de adrese şi de comenzi şi ieşiri de date, însă au şi intrări de date. Intrările şi ieşirile unei RAM statice simple, de 2^nxb biţi, sunt prezentate în figura 8-9. Intrările de comenzi sunt asemănătoare cu cele de la ROM, existând în plus o *intrare de activare a scrierii (WE- write enable).* Când **WE** este confirmată, datele de intrare se înscriu în locaţia de memorie selectată.

Locaţiile de memorie dintr-o RAM statică se comportă mai curând ca nişte circuite latch de tip D, decât ca nişte CBB de tip D, active pe front. Cu alte cuvinte, ori de câte ori intrarea **WE** este confirmată, circuitele latch corespunzătoare locaţiei de memorie selectate sunt „deschise" (sau „transparente"), iar datele de intrare intră în circuitul latch şi îl parcurg. Valoarea stocată practic este cea existentă în momentul în care circuitul latch se închide.

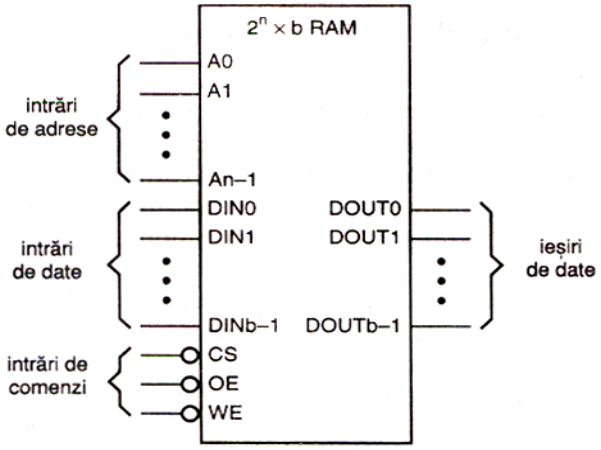


Figura 8-9 Structura de bază a unei RAM 2^nxb

La RAM statice sunt definite, în mod normal, două operaţii de acces:

- *Citire:* la intrările de adrese este furnizată o adresă în timp ce **CS** şi **OE** sunt confirmate. Ieşirile circuitelor latch corespunzătoare locaţiei de memorie selectate se regăsesc la **DOUT.**
- *Scriere:* la intrările de adrese este furnizată o adresă, iar la **DIN** se aplică un cuvânt de date, apoi **CS** şi **WE** se confirmă. Circuitele latch corespunzătoare locaţiei de memorie selectate se deschid, iar cuvântul de la intrare se stochează acolo.

8.3.2 Structura internă a RAM statice

Fiecare bit de memorie (sau *celulă de SRAM)* dintr-o RAM statică are comportarea funcţională a circuitului din figura 8-10. Dispozitivul de stocare din fiecare celulă este un circuit latch de tip D. Când intrarea **SELL** a unei celule este confirmată, data stocată se transmite la ieşirea celulei, care este conectată la

o linie de bit. Când atât **SEL_L**, cât şi **WR_L** sunt confirmate, circuitul latch este deschis, stocând un nou bit de date.

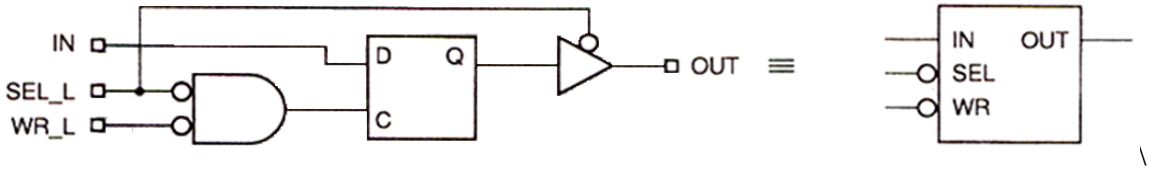


Figura 8-10 Comportarea funcţională a unei RAM statică

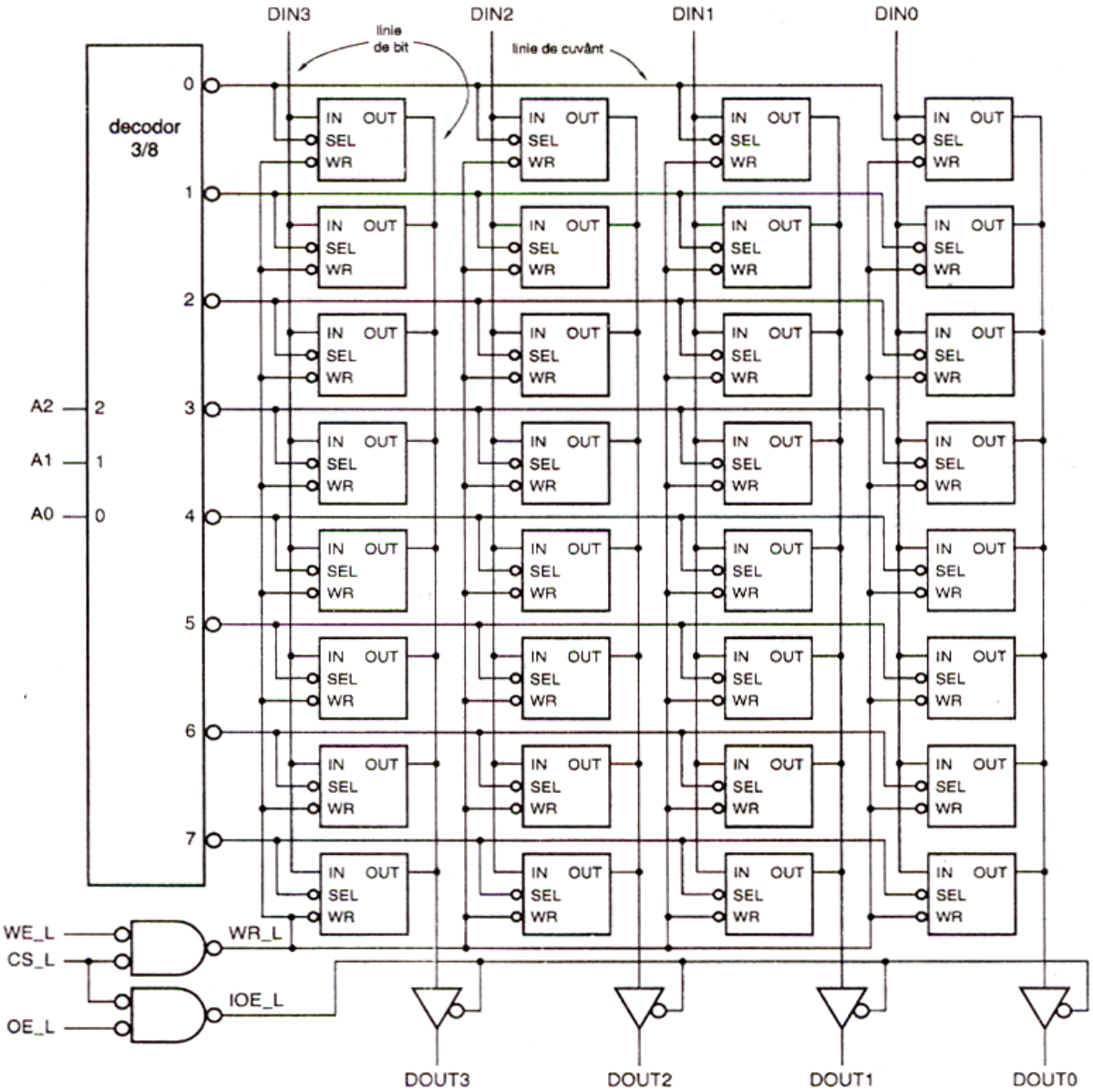


Figura 8-11 Structura internă a unei SRAM 8x4

Celulele de SRAM sunt interconectate, într-o matrice, cu o schemă suplimentară de comenzi, formând o RAM statică integrală, ca aceea din figura

8-11, în cazul unei SRAM 8x4. Ca şi la o ROM simplă, un decodor conectat la liniile de adrese selectează din RAM rândul accesibil la un moment dat.

Deşi figura 8-11 prezintă un model întrucâtva simplificat de structură internă de SRAM, câteva aspecte importante ale comportării SRAM sunt puse în evidenţă cu multă exactitate:

- În timpul operaţiilor de citire, datele de ieşire sunt funcţie combinaţională de intrările de adrese, ca şi la ROM. Schimbarea liniilor de adrese în timp ce magistrala de date de ieşire este activată nu produce disfuncţii. Pentru timpul de acces pentru operaţiile de citire se ia ca referinţă momentul în care ultima intrare de adrese a devenit stabilă.

- În timpul operaţiilor de scriere, datele de intrare sunt stocate în circuitele latch. Aceasta înseamnă că datele trebuie să se conformeze timpilor de pregătire şi menţinere, luându-se ca referinţă frontul următor al semnalului de activare a circuitelor latch. Cu alte cuvinte, nu este necesar ca datele de la intrarea D a unui circuit latch să fie stabile în momentul în care **WR_L** se confirmă intern; ele trebuie să fie stabile numai cu un anumit timp înainte ca **WR_L** să fie negat.

- În timpul operaţiilor de scriere, intrările de adrese trebuie să fie stabile cu un anumit timp de pregătire înainte ca **WR_L** să fie confirmat intern şi încă un anumit timp de menţinere după negarea semnalului **WR_L.**

- **WR_L** se confirmă intern numai când atât **CS_L**, cât şi **WE_L** sunt confirmate. Prin urmare, un ciclu de scriere începe atunci când atât **CS_L**, cât şi **WE_L** sunt confirmate şi se încheie când oricare dintre acestea se neagă. Valorile timpilor de pregătire şi menţinere corespunzători adreselor şi datelor sunt precizate luându-se ca referinţă aceste evenimente.

8.3.3 Parametrii temporali la RAM statice

În cazul memoriilor RAM statice sunt precizate două categorii de parametrii temporali: pentru scriere şi pentru citire.

În continuare sunt prezentaţi parametrii temporali ale căror valori se precizează, în mod obişnuit, pentru operaţiile de citire dintr-o RAM statică:

- *Timpul de acces de la adrese (t_{AA}).* În ipoteza că **OE** şi **CS** sunt deja confirmate sau vor fi confirmate după un timp suficient de scurt pentru a fi neglijabil, acest parametru arată cât timp este necesar pentru ca datele de ieşire să devină stabile după o modificare de adresă. Când proiectanţii vorbesc despre o „SRAM de 70 ns”, de obicei se referă la parametrul în discuţie.

- *Timpul de acces de la selectarea cipului (t_{ACS}).* În ipoteza că adresa şi **OE** sunt deja stabile sau vor fi stabile după un timp suficient de scurt pentru a fi neglijabil, acest parametru arată cât timp este necesar pentru ca datele de ieşire să devină stabile după o confirmare a semnalului **CS.** Adesea, are valoare egală cu t_{AA}, însă uneori este mai mare, la SRAM

prevăzute cu mod de „întrerupere a alimentării”, şi mai mic la SRAM la care nu este prevăzut acest mod de funcţionare.

- *Timpul de activare a ieşirilor (t_{OE}).* Este timpul necesar circuitelor tampon de ieşire cu trei stări pentru a ieşi din starea de impedanţă mare atunci când atât **OE**, cât şi **CS** sunt confirmate. Valoarea parametrului este, în mod normal mai mică decât t_{ACS}, deci RAM poate avea acces intern la date înainte de confirmarea semnalului **OE**; caracteristica descrisă se foloseşte la realizarea unor timpi de acces mici, evitându-se totodată, ”conflictele de magistrală” în multe aplicaţii.

- *Timpul de dezactivare a ieşirilor(t_{OZ}).* Este timpul necesar circuitelor tampon de ieşire cu trei stări pentru a intra în starea de înaltă impedanţă după negarea unuia dintre semnalele **OE_L** şi **CS_L**.

- *Timpul de menţinere a ieşirilor (t_{OH}).* Acest parametru precizează cât timp rămân valide datele de ieşire după o modificare a intrărilor de adrese.

Diagrama temporală şi parametrii temporali caracteristici operaţiilor de citire din SRAM sunt identici cu cei prezentaţi la operaţiile de citire din ROM.

În ceea ce priveşte parametrii temporali aferenţi operaţiilor de scriere aceştia sunt descrişi în continuare:

- *Timpul de pregătire a adreselor înainte de scriere (t_{AS}).* Toate intrările de adrese trebuie să fie stabile cu acest interval de timp înainte ca atât **CS**, cât şi **WE** să fie confirmate. În caz contrar, datele stocate în locaţii imprevizibile pot fi alterate.

- *Timpul de menţinere a adreselor după scriere (t_{AH}).* Analog cu t_{AS}, toate intrările de adrese trebuie menţinute stabile acest interval de timp după negarea unuia dintre semnalele **CS** sau **WE**.

- *Timpul de pregătire pentru selectarea cipului înainte de încheierea operaţiei de scriere (t_{CSW}).* **CS** trebuie confirmat cel puţin cu acest interval de timp înainte de încheierea ciclului de scriere, pentru a se putea selectă o celulă.

- *Lăţimea impulsurilor de scriere(t_{WP}).* WE trebuie confirmat cel puţin acest interval de timp pentru ca memorarea datelor în circuitele latch din celulele selectate să fie fiabilă.

- *Timpul de pregătire a datelor înainte de încheierea operaţiei de scriere (t_{DS}).* Toate intrările de date trebuie să fie stabile cu acest interval de timp înainte de încheierea ciclului de scriere. În caz contrar, este posibil ca datele să nu fie memorate în circuitele latch.

- *Timpul de menţinere a datelor după încheierea operaţiei de scriere (t_{DH}).* Analog cu t_{DS}, toate intrările de adrese trebuie menţinute stabile acest interval de timp după încheierea ciclului de scriere.

Fabricanţii de SRAM precizează că există două tipuri de ciclu de scriere: *scriere comandată prin WE şi scriere comandată prin CS*. Singura diferenţă dintre ele constă în succesiunea semnalelor - care dintre **WE** şi **CS** este ultimul confirmat şi primul negat la activarea operaţiei interne de scriere în SRAM.

8.4 RAM dinamice

Celula de memorie de bază din SRAM este un circuit latch de tip D care cuprinde patru porţi în schema cu componente discrete şi patru până la şase tranzistoare într-un cip LSI de SRAM, proiectat la comandă. Pentru construirea unor RAM cu densitate mai mare (cu mai mulţi biţi pe cip), proiectanţii de cipuri au inventat celule de memorie ce necesită doar câte un tranzistor pentru fiecare bit.

8.4.1 Structura de RAM dinamică

Cu un singur tranzistor nu se poate construi un element bistabil. De aceea, celulele de memorie dintr-o *RAM dinamică (DRAM)* stochează informaţia într-un condensator minuscul, la care se ajunge printr-un tranzistor MOS. Figura 8-12 prezintă celula de stocare a unui bit dintr-o DRAM, la care accesul este dat de conectarea liniei de cuvânt la o tensiune **HIGH**.

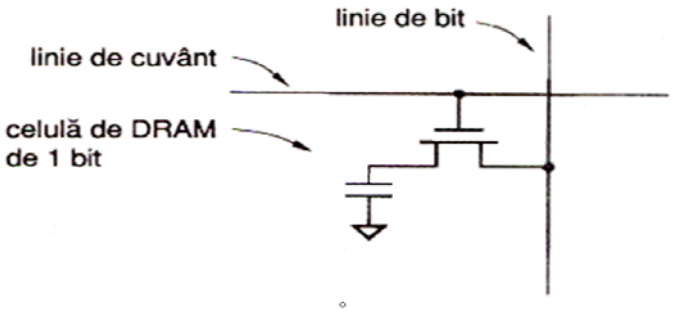


Figura 8-12 Celulă de stocare a unui bit în DRAM

Pentru stocarea unui 1, la linia de bit se conectează o tensiune **HIGH**, de la care condensatorul se încarcă prin tranzistorul „deschis”. Pentru stocarea unui 0, condensatorul se descarcă datorită tensiunii **LOW** de pe linia de bit.

Pentru a citi o celulă din DRAM, linia de bit se *preîncarcă întâi* la o valoare de tensiune situată la jumătatea intervalului dintre **HIGH** şi **LOW**, iar apoi linia de cuvânt trece în **HIGH**. În funcţie de tensiunea de pe condensator - **HIGH** sau **LOW**, se aduce linia de bit reîncărcată la o tensiune puţin mai mare sau puţin mai mică. Un *amplificator de detecţie* sesizează această modificare şi reface un 1 sau un 0, după caz. Reţineţi că prin citirea unei celule se distruge tensiunea iniţială stocată în condensator, deci datele recuperate trebuie rescrise în celulă după citire.

Condensatorul dintr-o celulă de DRAM are o capacitate foarte mică, dar tranzistorul MOS prin care se face accesul are o impedanţă foarte mare. De aceea, timpul necesar pentru descărcarea de la o tensiune **HIGH** la una apropiată de **LOW** este relativ lung (mai multe milisecunde). În tot acest timp, condensatorul stochează un bit de informaţie.

Bineînţeles că nu ar fi prea plăcut să reporneşti un calculator o dată la câteva milisecunde, din cauză că îi dispare conţinutul memoriei. Ca urmare, sistemele de memorii DRAM utilizează *cicluri de împrospătare* pentru actualizarea periodică a tuturor celulelor de memorie. Aceasta presupune citirea secvenţială a conţinutului uşor alterat al fiecărei celule prin preluarea lui într-un circuit latch de tip D, urmată de rescrierea unei valori clar delimitate - **HIGH** sau **LOW**, preluată din acel circuit.

În figura 8-13 observaţi schema bloc a structurii interne a unei DRAM 64Kx1. Matricea logică are 64Kx1 biţi, dar cea fizică este pătrată, conţinând 256x256 de biţi. Deşi memoria cuprinde 64K locaţii, cipul prezintă numai opt *intrări de adrese multiplexate.* O adresă completă, de 16 biţi, se aplică cipului în două trepte, comandate de două semnale: *RAS_L (row address strobe - semnal intermitent de adrese de rând) şi CAS_L (column address strobe - semnal intermitent de adrese de coloană).* Prin multiplexarea semnalelor de intrare pentru adrese se economisesc pini, aspect important în cazul sistemelor de memorii compacte, şi se asigură compatibilitatea oarecum naturală cu metodele de acces la DRAM, ce comportă două etape şi pe care le vom prezenta pe scurt.

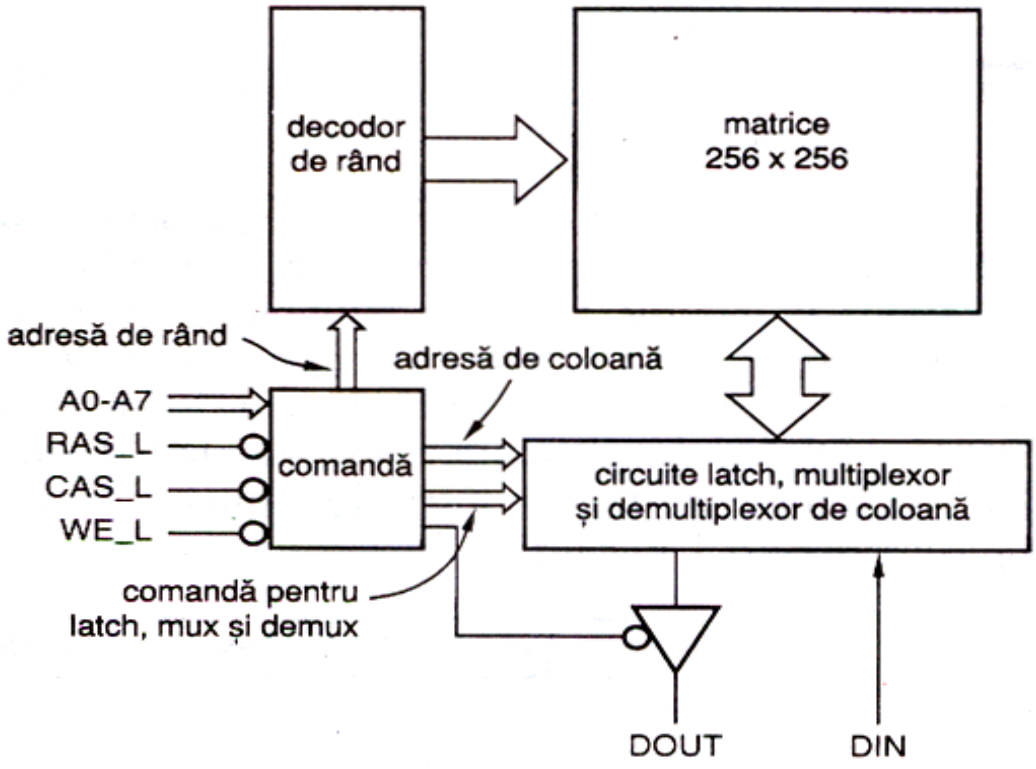


Figura 8-13 Structura internă a unei DRAM 64Kx1

DRAM de capacităţi mai mari conţin matrice de dimensiuni mai mari şi, uneori, mai multe matrice. Unul dintre avantajele folosirii mai multor matrice este simplificarea problemelor de proiectare electrică şi fizică ce pot apărea în cazul unei matrice foarte mari. Dar şi mai important este paralelismul ce poate apărea când se utilizează mai multe matrice. Aşa cum vom vedea în subsecţiunea următoare, funcţionarea unei DRAM este mult mai complicată decât cea a unei SRAM. Valorificând existenţa mai multor matrice în DRAM de mare capacitate şi de mare viteză, un circuit de comandă pentru DRAM moderne poate efectua în paralel câteva operaţii; de pildă, se poate încheia o operaţie de scriere într-una dintre matrice în timp ce se iniţiază o operaţie de citire în altă matrice. În acest mod se măreşte cantitatea de informaţii utile prelucrate de memorie.

8.4.2 Caracteristicile temporale ale RAM dinamice

Pentru diversele tipuri de DRAM şi modurile lor de funcţionare există mai multe variante de caracteristici temporale. În secţiunea de faţă vom descrie ciclurile cele mai utilizate la DRAM convenţionale şi le vom analiza pe baza structurii interne a dispozitivelor. Aspectul cel mai şocant în ceea ce priveşte caracteristicile temporale ale DRAM este faptul că nu se foloseşte un semnal de tact. În lipsa acestuia, operaţiile din DRAM se iniţiază şi se desfăşoară pe ambele fronturi - ascendent şi descendent - ale semnalelor **RAS_L** şi **CAS_L**. Într-adevăr, temporizarea este destul de dificilă, dar acest mod de lucru se menţine de peste 30 de ani!

În figura 8-15 apar caracteristicile temporale pentru un *ciclu de împrospătare comandat numai de semnalul RAS.* Ciclul prezentat este utilizat pentru împrospătarea unui rând de memorie, fără a se citi sau scrie propriu-zis, nici un fel de date la pinii exteriori ai cipului de DRAM. Ciclul începe prin aplicarea unei adrese de rând la intrările de adrese multiplexate (opt biţi, în cazul unei DRAM 64Kx1), iar **RAS_L** este confirmat. DRAM stochează adresa rândului într-un *registru de adrese de rând,* pe frontul descendent al semnalului **RAS_L**, şi citeşte rândul selectat din matricea de memorie, înscriindu-i conţinutul într-un *circuit latch de rând,* situat pe acelaşi cip. Când **RAS_L** este negat, conţinutul rândului se scrie din nou, fiind preluat din circuitul latch.

Pentru a împrospăta o întreagă DRAM 64Kx1, proiectantul sistemului trebuie să verifice dacă 256 de astfel de cicluri, în care se utilizează toate cele 256 de adrese de rând posibile, se pot executa o dată la patru milisecunde. Pentru generarea adreselor de rând se poate folosi un numărător extern de 8 biţi, iar pentru iniţierea unui ciclu de împrospătare o dată la 15,6 μs se foloseşte un circuit de temporizare.

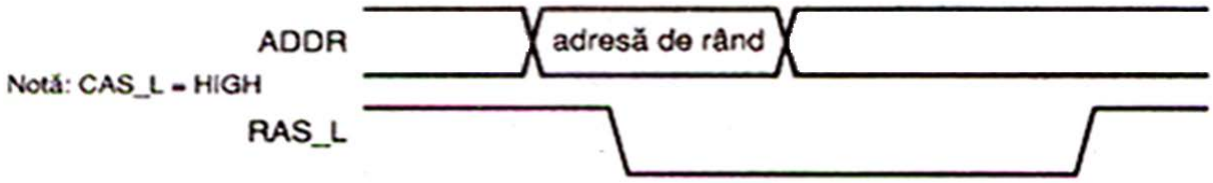


Figura 8-14 Caracteristici temporale pentru ciclul de împrospătare a unei DRAM prevăzute numai cu semnal RAS

Un ciclu de citire, ca acela din figura 8-16, începe asemănător unui ciclu de împrospătare, un rând selectat fiind citit prin preluarea în circuitul latch de rând. Apoi se aplică o adresă de coloană la intrările de adrese multiplexate, aceasta fiind stocată, pe frontul descendent al semnalului **CAS_L**, într-un *registru de adrese de coloană* situat pe acelaşi cip. Adresa de coloană serveşte la selectarea unui bit din rândul citit anterior, bit ce se regăseşte la pinul **DOUT** al DRAM. Fiind un pin cu trei stări, **DOUT** este activat ca ieşire atâta timp cât **CAS_L** este confirmat. În acest interval de timp, întregul rând se scrie din nou în matrice, imediat ce **RAS_L** este negat.

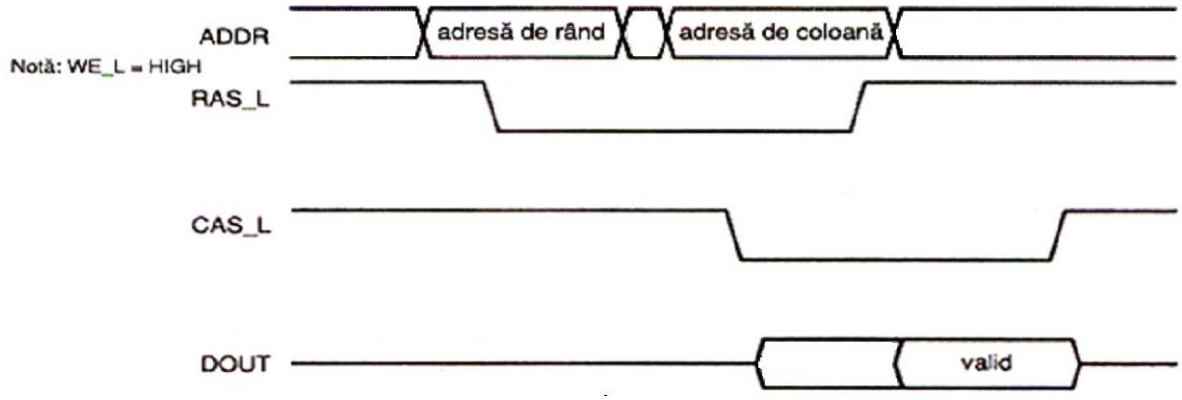


Figura 8-15 Caracteristici temporale pentru un ciclu de citire din DRAM

Un *ciclu de scriere,* ca acela din figura 8-16, începe la fel ca un ciclu de citire sau de împrospătare. Dar, în cazul de faţă, semnalul *WE_L (write enable - activarea scrierii)* trebuie confirmat înainte de confirmarea semnalului **CAS_L**, pentru a se selecta un ciclu de scriere. În acest mod se realizează şi dezactivarea pinului **DOUT** pentru tot restul ciclului, chiar dacă semnalul **CAS_L** va mai fi confirmat ulterior. După ce rândul selectat este citit, prin preluarea conţinutului său în circuitul latch de rând, **WE_L** impune adăugarea la conţinutul circuitului latch de rând, în poziţia bitului selectat prin adresa de coloană, a bitului de intrare de la pinul **DIN**. Când rândul este rescris ulterior în matrice, pe frontul ascendent al semnalului **RAS_L**, el conţine noua valoare în coloana selectată.

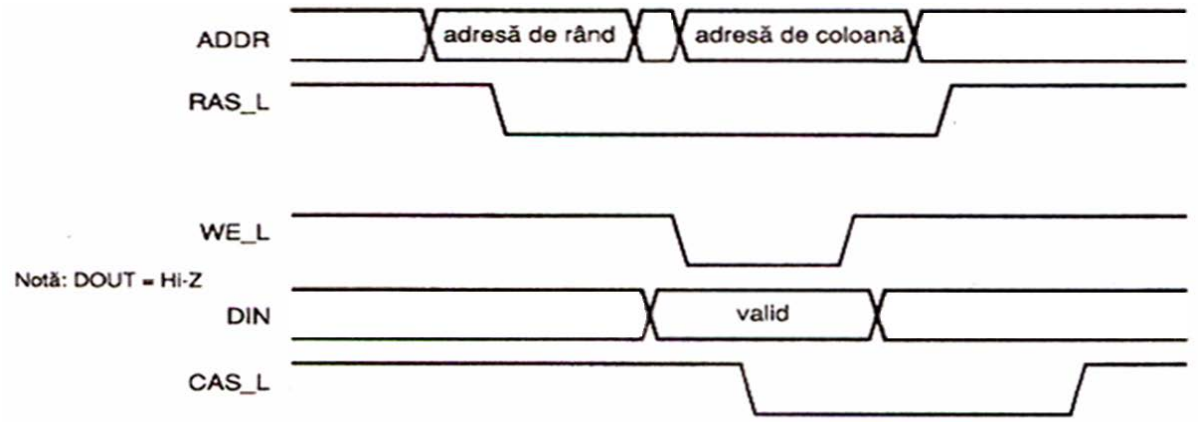


Figura 8-16 Caracteristici temporale pentru un ciclu de citire din DRAM

9. Recomandări pentru realizarea practică a schemelor electronice cu circuite integrate digitale

În vederea obţinerii proiectului final al unui montaj electronic, inginerul electronist trebuie să parcurgă mai multe faze. Se începe prin definirea funcţiilor care trebuie îndeplinite de circuit, stabilirea mărimilor de intrare şi a celor de ieşire şi a condiţiilor de funcţionare (surse disponibile de alimentare, gamă de temperatură, condiţii mecanice – grad de protecţie climatic, solicitări mecanice, şocuri şi vibraţii – şi, nu în ultimul rând, condiţiile de mediu electromagnetic: surse de perturbaţii, permanente şi accidentale, căi de propagare, etc.). În funcţie de cele enumerate anterior, se poate alege (proiecta) configuraţia şi se calculează componentele schemei electronice, eventual în mai multe variante, cu testări intermediare, prin software de analiză, sau prin încercări practice, obţinându-se o variantă finală, care poate fi încercată sub formă de *model experimental*. Se proiectează circuitul imprimat, care ţine cont de condiţiile de mediu şi se poate trece la faza următoare, construirea *prototipului*, care trebuie să răspundă tuturor condiţiilor impuse prin tema de proiect. Dacă şi această fază este trecută cu succes, urmează stabilirea tehnologiei de fabricaţie (care poate diferi de cea folosită la construcţia prototipului) şi realizarea *seriei „0”*, respectiv a unui număr redus de bucăţi care vor fi testate în vederea validării tehnologiei de fabricaţie. Oricare din etapele amintite poate fi repetată, chiar şi de mai multe ori, dacă rezultatele nu îndeplinesc complet cerinţele. După trecerea tuturor probelor impuse seriei „0”, se trece la producţia finală, de *serie mare*.

Schemele electronice digitale, construite pe circuite integrate numerice, pornesc de la analiza condiţiilor logice dintre intrări şi ieşiri, pe baza cărora se sintetizează funcţiile logice îndeplinite de configuraţia care trebuie proiectată. Dimensionarea componentelor este aproape inexistentă, circuitele logice legându-se între ele în mod direct. Lucrările de proiectare, în această fază, sunt în special de calcule binare, urmate de alegerea circuitelor capabile să execute funcţiile proiectate.

În general, prima etapă, sinteza funcţiilor logice, nu ridică probleme notabile. Mai departe, însă, proiectarea cu circuite integrate digitale trebuie făcută ţinând cont de realităţi tehnice care nu pot fi ignorate, cum ar fi: sursele de tensiune şi semnal existente şi variaţiile lor previzibile, timpii de propagare ai semnalelor şi întârzierile posibile ale acestora, legate de componentele parazite ale elementelor de circuit, fan-in-ul şi fan-out-ul circuitelor, capabilitatea

acestora de comandă, variaţia pragurilor de comutare, emisia şi recepţia de perturbaţii electromagnetice prin traseele de cablaj sau alte conductoare.

În cele ce urmează sunt prezentate câteva reguli şi recomandări de bază, a căror nerespectare duce frecvent la defectarea circuitelor digitale, sau la erori în funcţionarea lor. Din păcate, respectarea ad-litteram a acestor enunţuri nu garantează 100% reuşita proiectării, existând şi cazuri în care măsuri suplimentare trebuie aplicate.

Prima problemă, după sinteza funcţiilor logice, este alegerea familiei (familiilor) de circuite integrate digitale prin care se vor implementa funcţiile proiectate. În afara caracteristicilor enunţate în capitolele anterioare, pentru utilizarea practică trebuie avute în vedere recomandările practice ale producătorului.

9.1 Circuite TTL (seriile 74xx, 74Sxx, 74LSxx):

Sunt circuite vechi, nu se mai realizează proiecte noi cu circuite TTL. Recomandările vizează echipamentele vechi, încă folosite, care se defectează.

- Efectul diodelor de limitare asupra tensiunilor negative de intrare: Circuitele TTL sunt prevăzute cu diode de limitare pe intrări, pentru minimizarea efectelor impulsurilor negative. Totuşi, aceste diode nu trebuie folosite pentru limitarea tensiunilor negative de lungă durată, mai ales pentru seriile LS. Menţinerea unei tensiuni sub 0,5V pe intrare mai mult de 0,5μs poate altera nivelul logic SUS, prin activarea unui circuit parazit, ceea ce conduce la apariţia unor erori logice.
- Dispunerea intrărilor nefolosite: Intrările în gol, neconectate în circuit, înrăutăţesc imunitatea la zgomot în curent alternativ şi viteza de comutare a circuitului logic. Pentru optimizarea performanţelor, fiecare intrare trebuie conectată la o sursă de mică impedanţă. Intrările nefolosite ale porţilor SAU, SAU-NU trebuie legate la masă sau la o ieşire aflată în stare JOS. Intrările nefolosite ale porţilor ŞI, ŞI-NU trebuie legate la o tensiune mai mare de 2,7V, dar nu mai mare de 5V, sau la o ieşire aflată în stare SUS.
    - Practic, intrările active JOS ale circuitelor TTL se leagă la masă. Intrările active SUS se leagă la Vcc printr-o rezistenţă de 1 … 10kΩ. În cazul circuitelor TTL-LS, intrările nefolosite active SUS pot fi legate la Vcc direct, dacă traseele sunt scurte şi sursa este bine decuplată.
    - Intrările nefolosite active SUS pot fi legate la ieşirea unei porţi nefolosite forţată cu ieşirea SUS.

- Intrările nefolosite ale porţilor ŞI, ŞI-NU pot fi legate la o intrare folosită a aceleaşi porţi, dacă nu se depăşeşte fan-out-ul porţii anterioare în starea SUS.

- Porţile nefolosite: Se recomandă ca ieşirile porţilor nefolosite să fie forţate în starea SUS prin legarea câte unei intrări a porţilor ŞI, ŞI-NU sau a tuturor intrărilor porţilor SAU, SAU-NU la masă. În acest mod se reduce puterea consumată şi sunt disponibile ieşiri în starea SUS la care se pot lega intrări nefolosite.

- Mărirea fan-out-ului: Pentru a mări fan-out-ul, intrări şi ieşiri ale porţilor din aceeaşi capsulă pot fi conectate în paralel. Nu se recomandă utilizarea porţilor din capsule diferite pentru punerea în paralel, deoarece, datorită diferenţelor mari de timpi de comutare, apar impulsuri mari de curent pe alimentare, ceea ce nu este dăunător pentru circuite, dar poate altera funcţionarea logică dacă porţile furnizează semnale de tact altor circuite.

- Diodele de izolare: Niciodată nu trebuie inversată polaritatea tensiunii de alimentare. Trecerea unui curent de 100mA prin diodele de izolare faţă de substrat duce la defectarea catastrofică a circuitului.

- Cerinţele semnalului de tact: Cele mai multe bistabile TTL sunt circuite master-slave, ceea ce le face sensibile la nivelul impulsului de tact aplicat pe intrare. În particular, funcţionarea circuitelor este dependentă de duratele fronturilor impulsurilor de ceas. Nivelul de curent continuu la care informaţia este transferată din master (secţiunea de intrare) în slave (secţiunea de ieşire) este nivelul normal de prag al dispozitivelor. În general, acest nivel este 1,4V la 25^{o}C şi se schimbă cu -4mV/^{o}C. Când intrarea de tact atinge nivelul de prag, comutarea etajelor de ieşire formează impulsuri de curent în terminalul de masă. Multiple comutări, la care se adaugă zgomotul de masă, pot creşte referinţa de masă cu circa 500mV, modificând astfel nivelurile de tensiune pe intrările de tact. Este astfel posibil ca intrările de tact active pe front de creştere să primească impulsuri multiple de ceas. Din acest motiv, durata frontului de creştere a impulsului de tact aplicat intrărilor active pe front de creştere trebuie să fie inferioară întârzierii propagării între intrarea de tact şi ieşire. Acest deziderat este atins dacă se respectă încărcarea ieşirii porţii care formează tactul şi dacă traseul de cablaj dintre ieşirea porţii care comandă şi intrarea de tact nu depăşeşte 300 … 400mm. Dacă semnalul de tact este transmis prin linii mai lungi de 400mm, toate intrările de tact trebuie grupate la capătul de recepţie al liniei pentru a evita problemele de reflexie la capătul de emisie. Anumite circuite TTL (numărătoare şi registre) sunt proiectate şi realizate cu un tampon pe intrarea de tact, care prezintă şi

un mic histerezis, pentru a minimiza duratele fronturilor semnalului de ceas şi, implicit, efectul zgomotului.

- Ieşiri TTL legate împreună: Dintre toate configuraţiile etajelor de ieşire TTL, doar ieşirile „colector în gol" (open collector) şi „3 stări" (3-state) sunt concepute pentru a fi legate între ele. Ieşirile TTL standard nu trebuie legate laolaltă decât dacă nivelurile lor logice sunt permanent aceleaşi – toate SUS, sau toate JOS. La conectarea împreună a ieşirilor colector în gol sau 3 stări, trebuie respectate câteva reguli generale:
  - Ieşiri colector în gol: Aceste dispozitive sunt utilizate în cazurile în care două sau mai multe ieşiri legate în conexiune SAU cablat pot fi în opoziţie logică la un moment dat. Ieşirile legate împreună trebuie conectate printr-un rezistor (pull-up resistor) la Vcc, prin care se stabileşte nivelul logic SUS. Doar anumite circuite tampon sau inversoare pot fi legate la tensiuni mai mari decât Vcc. Limitele între care poate fi ales rezistorul se determină prin relaţiile următoare:

$$R(min) = \frac{V_{CC}(Max) - V_{OL}}{I_{OL} - N_2(I_{IL})}$$

$$R(max) = \frac{V_{CC}(Min) - V_{OH}}{N_1(I_{OH}) - N_2(I_{IH})}$$

  - Ieşiri 3 stări: Ieşirile 3 stări sunt fabricate pentru a fi legate laolaltă, dar nu sunt destinate să fie active simultan. Pentru minimizarea zgomotului şi pentru a evita defectarea circuitelor prin putere disipată excesivă, cel mult o ieşire 3 stări trebuie să fie activă în orice moment. Acest lucru necesită ca semnalele de comandă ale intrărilor de autorizare (enable) să nu se suprapună. Dacă se folosesc decodoare TTL pentru intrările de autorizare, decodorul trebuie dezactivat pe durata schimbării adresei. Deoarece toate decodoarele TTL sunt susceptibile de a decoda impulsuri scurte, nesuprapunerea semnalelor de autorizare nu poate fi garantată pe durata schimbării adreselor. În general, se preferă circuite de decodare la care tranziţia JOS - SUS să fie mai rapidă decât cea SUS – JOS, ţinând cont că intrările de autorizare sunt active JOS.
- Decuplarea surselor de alimentare: Montarea condensatoarelor de decuplare pe alimentare este obligatorie în cazul circuitelor TTL. În general, se acceptă 10nF pentru fiecare poartă comandată sincron şi cel puţin 100nF pentru 20 porţi care lucrează nesincronizat. Numărătoarele şi registrele de deplasare sunt sensibile în măsură mai mare la zgomote pe trasele de alimentare Vcc şi masă. Regula este de a

decupla fiecare opt bistabile interne printr-un condensator de 100nF, sau un condensator pentru două capsule. Condensatorul trebuie montat cât mai aproape posibil de terminalele de alimentare ale capsulei. Circuitele tampon (buffer) şi circuitele care comandă linii de semnal (line-driver) trebuie decuplate foarte bine, ţinând cont de curenţii tranzitorii de valori ridicate necesari pentru încărcarea şi descărcarea liniilor.

- Stabilizarea pe placă a alimentării: În multe sisteme digitale consumul de curent este ridicat şi, totuşi, este asigurat de o sursă unică. Circuitele TTL generează impulsuri scurte de curent mare pe alimentări datorită suprapunerii conducţiei tranzistoarelor finale, pe durata comutărilor, formându-se astfel zgomot pe alimentarea cu Vcc. Un stabilizator de tensiune chiar pe placa cu circuite TTL va stabiliza tensiunea aproape de circuite şi, în plus, va izola din punct de vedere al zgomotului pe alimentare plăcile între ele. Aplicaţiile care utilizează această tehnică pot lucra cu surse principale nestabilizate.
- Emisia şi recepţia pe linii de transmisie: Conexiunile dintre diverse circuite TTL nu trebuie strânse laolaltă, legate sau grupate împreună. Dimpotrivă, trebuie utilizate legături punct – la – punct, preferabil deasupra unui plan de masă care reduce cuplajul dintre trasee. Liniile monofilare nu trebuie să depăşească lungimea de 600mm; pentru distanţe mai mari, de 400mm, planul de masă este esenţial pentru asigurarea performanţelor sistemului. Pentru distanţe peste 600mm trebuie folosite cablu coaxial sau pereche de conductoare torsadate. Impedanţa caracteristică a traseului imprimat cu plan de masă este de aproximativ 150Ω, iar al perechii torsadate de aproximativ 120Ω (pentru cabluri Φ0,4mm). Pentru protecţie superioară la diafonii se folosesc cablurile coaxiale, dar acestea au impedanţe caracteristice mici, ceea ce le face greu de comandat. Pentru performanţe optime, trebuie alese cabluri coaxiale cu impedanţă de 100Ω. La capătul de recepţie se montează rezistenţe între Vcc şi ieşirea cablului (intrarea porţii), care îmbunătăţesc marginea de zgomot. Dacă efectele reflexiilor sunt inacceptabile, liniile de transmisie trebuie terminate pe impedanţele lor caracteristice. În figura 9-1 se arată terminarea liniei pe o rezistenţă egală cu impedanţa caracteristică legată către Vcc. Acest mod de terminare solicită suplimentar poarta de emisie în starea JOS, când trebuie sa absoarbă si curentul care trece prin rezistenţa terminală. Legarea ieşirii liniei la mediana unui divizor compus din două rezistenţe egale între ele, având valori duble faţă de impedanţa caracteristică, reduce suprasolicitarea în curent a porţii de emisie la 50%. Este recomandabil ca poarta de emisie să fie folosită exclusiv

pentru comanda liniei de transmisie, dacă lungimea liniei depăşeşte 1500mm.

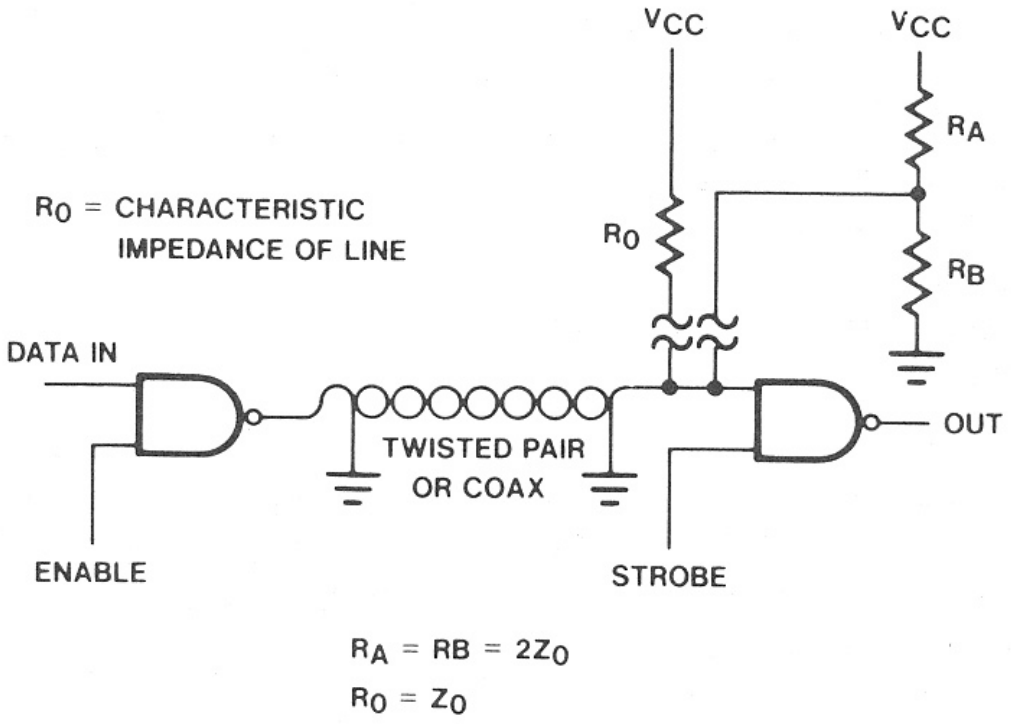


Figura 9-1 Conectarea circuitelor TTL prin cablu coaxial

9.2 Circuite CMOS normale (seria CD4xxx):

Sunt circuite lente, mai ales la tensiuni mici. Se folosesc încă, în aplicaţii la tensiune mai mare de 5V (12V, 15V), unde beneficiază de o margine de zgomot superioară, sau în cazul în care timpul mare de răspuns este o calitate!

- Alimentarea circuitelor integrate CMOS se va face cu respectarea în permanenţă a regulii $V_{SS} < V_{in} < V_{DD}$. Astfel, dacă la intrarea circuitului se folosesc surse de impulsuri de mică impedanţă sau surse de alimentare separate, alimentarea circuitului CMOS trebuie conectată prima. Ordinea se inversează la deconectarea alimentării. Nerespectarea acestei succesiuni poate duce la defectarea diodelor de protecţie pe intrări.
- Înserierea unor impedanţe pe alimentarea circuitelor CMOS duce la întârzierea conectării reale a alimentării, caz în care se poate produce efectul descris anterior.
- Inversarea polarităţii tensiunii de alimentare trebuie evitată prin mijloace de protecţie adecvate, în caz contrar provoacă defectarea circuitelor CMOS.

- Intrările neutilizate ale porţilor logice trebuie conectate la V_{DD} sau V_{SS}, altfel, datorită impedanţelor foarte mari de intrare, ieşirile porţilor pot fi în zona de comutare în care curenţi mari parcurg tranzistoarele de ieşire, ceea ce supraîncălzeşte dispozitivul. Este complet neindicat ca intrările CMOS să fie neconectate – spre deosebire de familia TTL, intrările CMOS nu au o stare implicită în gol.
- Din cauza impedanţelor foarte mari de intrare şi pericolului de străpungere prin sarcini electrostatice, intrările standard CMOS sunt echipate cu reţele de protecţie la descărcări electrostatice, care le protejează în majoritatea cazurilor (figura 9-2). Totuşi, se recomandă luarea unor măsuri suplimentare de precauţie la păstrarea şi manipularea circuitelor:
  - Utilizarea unor suprafeţe de lucru conductive;
  - Legarea la masă a echipamentelor folosite în manipularea (transport, montaj, lipire) circuitelor;
  - Păstrarea circuitelor în ambalaje antistatice, cutii, baghete sau alte incinte conductive.

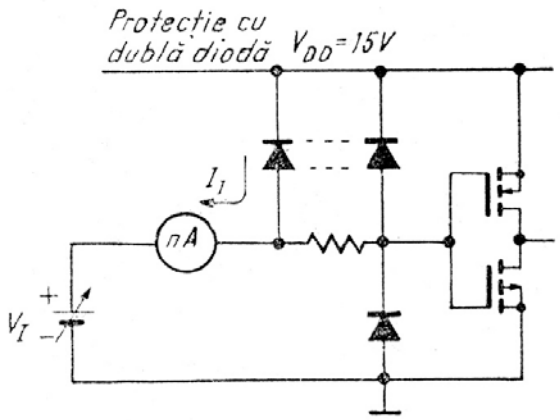


Figura 9-2 Protecția intrărilor CMOS

- Se interzice conectarea ieşirilor standard CMOS în configuraţia SAU-CABLAT deoarece funcţia logică nu este îndeplinită dar etajele de ieşire se pot supraîncărca reciproc în curent. În general, conectarea paralelă a ieşirilor porţilor CMOS este de evitat, excepţie făcând cazul în care porţile aparţin aceleiaşi capsule şi primesc pe intrări acelaşi semnale de comandă.
- Impedanţele de sarcină ale ieşirilor CMOS nu trebuie conectate la tensiuni în afara intervalului delimitat de tensiunile de alimentare, altfel existând pericolul intrării în conducţie a diodelor din ieşiri .

- Tensiunea de alimentare nu trebuie să depăşească valorile limită absolute pentru fiecare familie CMOS.
- Sursa de alimentare trebuie decuplată pe fiecare placă cu circuite CMOS prin condensatoare de decuplare între 10nF … 100nF, fiind recomandată montare de condensatoare de decuplare în apropierea circuitelor care comută la frecvenţe mari.
- Circuitele CMOS nu necesită tensiune stabilizată, dacă se respectă cerinţele de filtraj şi decuplare.
- Se va evita depăşirea puterii maxime care poate fi disipată de capsulă.
- Scurtcircuitarea ieşirilor la tensiunile de alimentare, când tensiunea de alimentare este superioară valorii de 5V este interzisă, existând riscul defectării tranzistoarelor finale din etajul de ieşire, care pot disipa maximum 100mW.
- Conectarea unui număr mare de intrări la o singură ieşire este posibilă pentru frecvenţe mici de comutare (fan-out-ul ieşirii CMOS este mai mare de 50), dar la frecvenţe mari se produce o supraîncărcare capacitivă a etajului de ieşire, care duce la supraîncălzirea acestuia. Suplimentar, suprasarcina capacitivă măreşte timpul de tranziţie între nivelurile logice, ceea ce poate provoca funcţionarea defectuoasă a montajului.
- Semnalele de comandă pentru intrările CMOS nu trebuie să aibă fronturi lungi de tranziţie, ceea ce ar menţine etajul de ieşire în zona liniară un timp nepermis de mare, urmat de disipaţie excesivă de căldură. Pentru semnale lent variabile trebuie folosite circuite cu prag pe intrare, triggere Schmitt (ex. 4093, 40106), sau comparatoare (ex. LM339).
- Intrările CMOS comandate de ieşiri CMOS de pe alte plăci vor fi legate prin rezistenţe de valoare mare către V_{DD} sau V_{SS} pentru a evita rămânerea în gol la conectarea alimentărilor.
- Conectarea circuitelor CMOS care lucrează la tensiuni de alimentare diferită se va face cu respectarea regulii $V_{SS} < V_{in} < V_{DD}$. În acest sens se vor folosi circuite interfaţă: divizoare rezistive, circuite cu tranzistoare şi rezistenţe, comparatoare integrate, porţi cu drena în gol, sau circuitele speciale de interfaţă 4049 şi 4050.
- Transmisia la distanţă a semnalelor între circuite CMOS este limitată de timpii de comutare şi de lungimea şi natura liniilor de transmisie.

Tabel 9-1 Întârzieri ale semnalelor prin diverse tipuri de linii

Tipul liniei	Timp de întârziere [ns]	Lungimea liniei
COAXIAL cu Z_0=50Ω	3,9	1 m
COAXIAL din polietilenă cu Z_0=50Ω	5	1 m
Feeder TV cu Z_0=75Ω	5	1 m
Cablu răsucit (bifilar) Z_0=110Ω	4,8	1 m

9.3 Circuite CMOS de mare viteză – HCMOS (seriile HC/HCT):

Sunt circuite moderne, în utilizare curentă, seria HC fiind compatibilă cu microcontrollere operând în 2V … 6V, iar seria HCT fiind compatibilă TTL, în 5V. Circuitele integrate 74HC şi 74HCT sunt ideale pentru a fi folosite în proiectarea echipamentelor noi sau pentru înlocuirea circuitelor LSTTL în cele vechi, prin aceasta îmbunătăţindu-se şi performanţele electrice ale montajelor în multe privinţe.

Tabelul 9-2 prezintă comparativ performanţele familiilor de circuite integrate.

Seria HCMOS, fiind o subfamilie a circuitelor integrate CMOS se supune tuturor regulilor enunţate în subcapitolul anterior. Anumite particularităţi ale familiei HCMOS, datorate frecvenţelor de lucru net superioare, sunt prezentate suplimentar:

- Semnalele de tact aplicate circuitelor bistabile trebuie să aibă timpi de creştere şi descreştere inferiori valorii de catalog – 500ns / 4,5V alimentare, altfel sunt posibile comutări multiple datorită ridicării potenţialului planului de masă cu până la 0,5V din cauza curenţilor mari prin etajele de ieşire pe durata comutării. Bistabilele J-K (74HC/HCT73, 74, 107, 109 şi 112) sunt prevăzute cu intrări de tact cu trigger Schmitt, care înlătură limitarea anterioară.
- În sistemele comandate sincron de acelaşi semnal de tact, diferenţele în nivelurile de prag ale intrărilor pot da erori logice în cazul fronturilor de creştere / descreştere de durate mari. Pentru evitarea acestui tip de eroare, duratele maxime de creştere / descreştere ale fronturilor trebuie să fie inferioare dublului întârzierii la propagare prin bistabil (figura 9-3).
- Frecvenţa maximă de lucru pentru un bistabil singular este dată de următoarea relaţie: $f_{max} = 1/2\,t_w$. În sistemele comandate sincron de acelaşi semnal de tact, (figura 9-4), pot apărea factori care să micşoreze frecvenţa maximă de lucru, altfel, putând să apară erori logice. În cazul prezentat,

$$f_{max} = \frac{1}{t_{P\,max}(CP\ la\ Q_1) + t_{P\,max}(CL) + t_{SU}(Q_2\ la\ CP)}$$

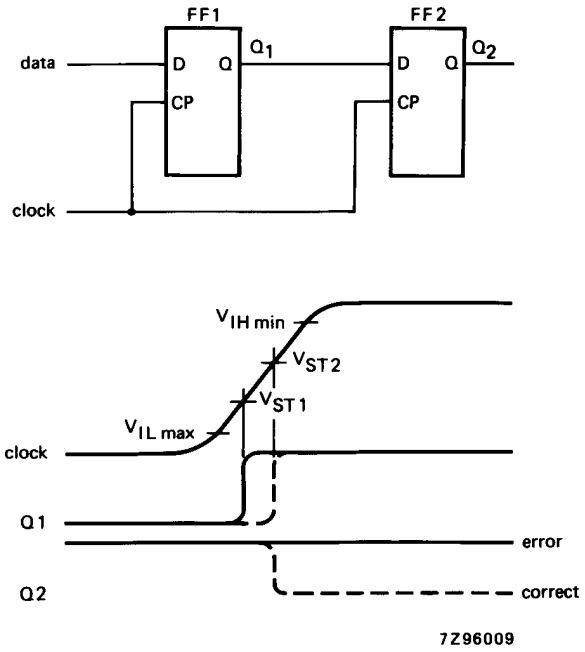


Figura 9-3 Comanda sincronă a bistabililor

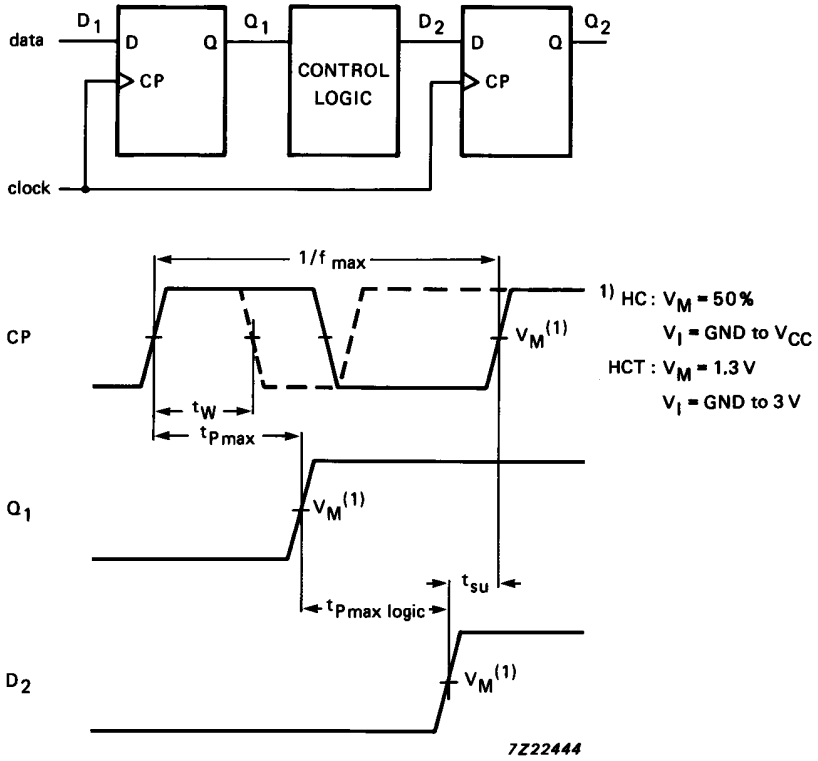


Figura 9-4 Timpii de propagare și reacție la comanda sincronă a bistabililor

Tabel 9-2 Comparație între tehnologiile CMOS și TTL la Vcc=5V, Tamb=25 și CL=15pF

technology	HCMOS	metal gate CMOS		standard TTL	low-power Schottky TTL	Schottky TTL	advanced low-power Schottky TTL	advanced Schottky TTL	Fairchild advanced Schottky TTL
parameters / family	74HC	4000 CD	4000 HE	74	74LS	74S	74ALS	74AS	74F
Power dissipation, typ. (mW)									
Gate static	0.0000025	0.001		10	2	19	1.2	8.5	5.5
Gate dynamic @100 kHz	0.075	0.1		10	2	19	1.2	8.5	5.5
Counter static	0.000005	0.001		300	100	500	60	–	190
Counter dynamic @100 kHz	0.125	0.120		300	100	500	60	–	190
Propagation delay (ns)									
Gate typical	8	94	40	10	9.5	3	4	1.5	3
Gate maximum	14	190	80	20	15	5	7	2.5	4
Delay/power product (pJ)									
Gate at 100 kHz	0.52	9	4	100	19	57	4.8	13	16.5
Maximum clock frequency (MHz)									
typical	55	4	12	25	33	100	60	160	125
D-type flip-flop									
minimum	30	2	6	15	25	75	40	–	100
Counter typical	45	2	6	32	32	70	45	–	125
Counter minimum	25	1	3	25	25	40	–	–	100
Output drive (mA)									
standard outputs	4	0.51	0.8	16	8	20	8	20	20
bus outputs	6		1.6	48	24	64	24	48	64
Fan-out (LS-loads)									
standard outputs	10	1	2	40	20	50	20	50	50
bus outputs	15		4	120	60	160	60	120	160

- Puterea disipată de dispozitivele HCMOS depinde hotărâtor de frecvenţa de comutare, ca de altfel şi la circuitele CMOS clasice, dar, datorită frecvenţelor superioare posibile, trebuie luată în considerare în proiectare:
  - În regim static, datorită conducţiei în contratimp a tranzistoarelor din etajele de ieşire, curentul de alimentare (fără sarcină) este extrem de redus (câţiva nA), iar puterea disipată este foarte mică, practic neglijabilă.
  - În regim dinamic, se disipă putere prin încărcarea şi descărcarea capacităţilor parazite din circuitul integrat şi de sarcină. De asemenea, există o putere considerabilă pe timpul comutării, cauzată de suprapunerea conducţiei tranzistoarelor din etajul final, dar, acest fenomen este răspunzător doar de 10% din puterea generată pe dispozitiv. Puterea totală în regim dinamic se poate calcula prin relaţia: $P_D = C_{PD}V_{CC}^2 f_i + \sum(C_L V_{CC}^2 f_0)$, unde: C_{PD} este capacitatea disipativă de putere (acest parametru este caracterizat în foile de catalog), f_i este frecvenţa de intrare, f_o este frecvenţa de ieşire, C_L este capacitatea sarcinii conectate la ieşire şi V_{CC} este tensiunea de alimentare.
- Deşi tensiunea de alimentare a circuitelor 74HC este limitată inferior la 2V, pentru a asigura compatibilitatea cu procesoarele şi memoriile de joasă tensiune, pentru aplicaţii liniare, cum ar fi oscilatoarele RC, se recomandă alimentarea circuitelor cu minimum 3V pentru a asigura o zonă suficientă de funcţionare în regiunea liniară.
- Tensiunea maximă de funcţionare a circuitelor 74HC/HCT este 7V, peste aceasta circuitele defectându-se.
- Tensiunea mică de lucru şi curentul de alimentare redus permit rezervarea alimentării circuitelor din seria HCMOS cu acumulatoare sau elemente galvanice (2 elemente înseriate alcaline sau unul cu Li, în cazul elementelor galvanice, sau 2 elemente NiMH înseriate sau o celulă Li-Ion, Li-Pol…, pentru acumulatoare).
- Gama larga a tensiunilor de alimentare (2V … 6V) sugerează faptul că seria HCMOS nu necesită stabilizarea tensiunii de alimentare. Totuşi, schimbarea tensiunii de alimentare influenţează frecvenţa maximă de lucru, imunitatea la zgomot şi puterea consumată. Perturbaţiile în impuls afectează funcţionarea circuitelor, motiv pentru care este necesară o bună decuplare a tensiunii de alimentare. Valorile recomandate sunt de 22nF pentru 2, …, 5 capsule şi 1μF tantal pentru fiecare 10 capsule. Circuitele de transmisie pe linie (drivere de linie sau magistrală) trebuie decuplate cu minimum 22nF, cât mai aproape

de capsulă. În cazul utilizării unui stabilizator de tensiune, se recomandă decuplarea ieşirii acestuia cu 10μF, …, 50 μF.

- Intrările circuitelor HCMOS sunt protejate la polarizare în afara domeniului de alimentare prin reţele R-D (figura 9-5). Chiar în aceste condiţii, se recomandă ca tensiunea aplicată pe intrări să nu ia valori în afara intervalului tensiunilor de alimentare, sau dacă acest lucru nu este posibil, trebuie limitat curentul de intrare, conform catalogului (20mA pentru curent pozitiv, 3mA, …, 14mA, pentru curent negativ, depinzând de numărul intrărilor afectate pe capsulă). Convertoarele de nivel logic HIGH către LOW, 74HC4049 şi 74HC7050, sunt protejate pe intrare doar pentru tensiuni negative (figura 9-6).

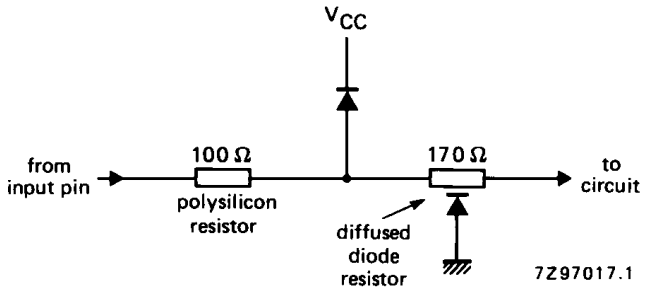


Figura 9-5 Protecția la polarizare în afara domeniului de alimentare a intrărilor circuitelor HCMOS

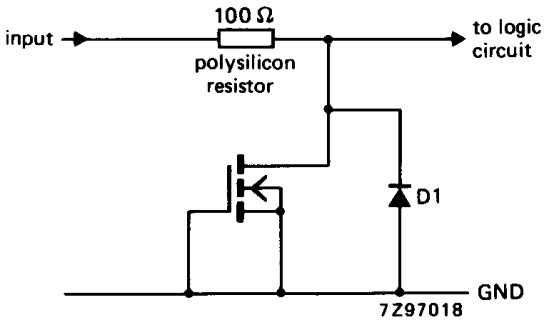


Figura 9-6 Protecția pentru tensiuni negative a intrărilor circuitelor HCMOS

Recomandările prezentate anterior au un caracter general şi neexhaustiv. Proiectantul de circuite logice cu integrate digitale trebuie să utilizeze, în afara recomandărilor generale, informaţiile specifice cuprinse în datele de catalog ale fiecărui circuit integrat în parte.

Lista figurilor

Lista tabelelor

Bibliografie

1. John F. Wakerly – *Digital Design: Priciples and Practices, Third Edition* – Prentice Hall, 2000
2. I. Spânulescu, S.I. Spânulescu – *Circuite integrate digitale şi sisteme cu microprocesoare* – Editura Victor, 1996
3. Paul R. Gray, Robert G. Meyer – *Analysis and Design of Analog Integrated Circuits* – John Wiley & Sons, Inc., 1993
4. N. Drăgulescu – *Agenda radioelectronistului* – ediţia I, II, Editura Tehnică
5. Eleodor Gh. Bistriceanu – *Algebre booleene şi circuite digitale* – Bucureşti, MatrixRom, 1997
6. Sorin Dan Anghel – *Circuite electronice analogice si digitale* – Cluj-Napoca, Universitatea „Babes-Bolyai”, 2002
7. Gheorghe Stefan si Virgil Bistriceanu – *Circuite integrate digitale* - Cluj-Napoca, Editura Albastra, 2000
8. Gheorghe Stefan - *Circuite si sisteme digitale* - Bucuresti : Editura Tehnica, 2000
9. Gheorghe Stefan, Virgil Bistriteanu - *Circuite integrate digitale* - Bucuresti : Editura Didactica si Pedagogica, 1992
10. Gheorghe Stefan - *Circuite integrate digitale* - Bucuresti : Denix, 1993
11. Gheorghe M.Stefan - *Functie si structura în sistemele digitale* - Bucuresti : Editura Academiei Române, 1991
12. Radu Mutihac - *Proiectarea circuitelor VLSI analogice si digitale* - Bucuresti : Editura Universitatii din Bucuresti, 1999
13. www.alldatasheet.com
14. www.intel.com
15. www.amd.com
16. www.ti.com
17. www.philips.com