Microcontrolere: arhitectură şi aplicații

Descrierea CIP a Bibliotecii Naționale a României
ZOICAN, SORIN
Microcontrolere : arhitectură şi aplicații / Sorin Zoican. - Bucureşti : Editura Academiei Oamenilor de Ştiință din România, 2011
Bibliogr.
Index
ISBN 978-606-8371-04-7

004.318

Editura Academiei Oamenilor de Ştiință din România

Adresa: Splaiul Independenței, nr. 54, sectorul 5, cod 050094 Bucureşti, România

Redactor: ing. Mihail CĂRUȚAŞU

Documentarist: ing. Ioan BALINT

Coperta: ing. sist. Adrian Nicolae STAN

Sorin Zoican

Microcontrolere: arhitectură şi aplicații

Editura Academiei Oamenilor de Ştiință din România

Bucureşti

2011

PREFAȚĂ

Lucrarea "Microcontrolere-Arhitectură şi aplicaţii" se adresează celor care doresc să cunoască funcționarea şi utilizarea microcontrolelor şi este utilă tuturor celor care doresc să înceapă studiul acestora.

În această lucrare se prezintă în mod gradat modul în care se realizează un microprocesor, de la circuite logice simple la circuite complexe, programabile şi ilustrează sistemele cu microprocesor, care integrate într-un singur circuit integrat, formează un microcontroler. De asemenea se indică modalitățile practice pentru implementarea unei aplicații în timp real cu microcontroler.

Lucrarea este structurată în 6 capitole.

Capitolul 1 se referă la noțiuni de aritmetică binară şi introduce principalele noțiuni despre circuitele logice.

Capitolul 2 descrie functionarea automatelor secvențiale programabile care vor sta la baza realizării unităților centrale de prelucrare (procesoare), ce constituie nucleul unui microcontroler.

Capitolul 3 se referă la unitatea centrală de prelucrare (UCP) în arhitectura clasică von Neumann. Sunt ilustrate: funcționarea microcalculatorului elementar, metode pentru creşterea performanțelor UCP şi subsisteme ale UCP - unitatea aritmetică, unitatea de comandă, unitatea de adresare. Se exemplifică arhitectura unor procesoare specializate în prelucrarea digitală a semnelelor.

Capitolul 4 descrie modul în care un procesor se poate sincroniza cu dispozitive externe. În acest capitol se prezintă conceptul de întrerupere şi problemele referitoare la sistemul de întreruperi al unui microcontroler. Sunt ilustrate tipuri de transfer de intrare-ieşire (paralel, serial, programat sau prin DMA). Se prezintă modul de proiectare a selecției memoriei şi a porturilor de intrare - ieşire.

Capitolul 5 ilustrează două arhitecturi de microcontrolere: arhitectura AVR (Atmel) – pentru microcontrolere de uz general şi arhitectura ADSP218x (Analog Devices) – pentru microcontrolere pentru prelucrarea

de semnal. Sunt descrise modelele de programare, organizarea datelor, semnalele procesorului, realizarea unui sistem cu microcontroler.

Capitolul 6 prezintă modul in care se poate realiza o aplicație de timp real cu microcontroler şi limbajul de programare pentru microcontrolerul ADSP218x. De asemenea, se indică modalități pentru optimizarea programelor scrise în limbaj C, pentru microcontrolere.

Corecturile şi sugestiile cititorilor vor fi întotdeauna bine primite.

Bucureşti

Prof. dr. ing.Sorin Zoican

CUPRINS

1. ELEMENTE DE ARITMETICĂ BINARĂ ŞI CIRCUITE DIGITALE ELEMENTARE

1.1. Elemente de algebră booleană

Algebra logică sau algebra booleană a fost concepută ca o metodă simbolică de tratare a funcţiilor logice formale, fiind ulterior extinsă şi în alte domenii. Deoarece între logica formală şi circuitele de comutaţie (cu două stări) există o corespondenţă directă (logica formală studiază valoarea de adevăr sau de fals a unor afirmaţii ce pot fi numai adevărate sau false, iar circuitele de comutaţie sunt realizate prin interconectarea unor comutatoare ce au două stări: închis sau deschis), algebra booleană s-a impus ca cea mai importantă modalitate de analiză şi sinteză a circuitelor de comutaţie.

Algebra booleană se defineşte axiomatic astfel: fie (M, •, +) o mulţime cu două operaţii şi M={0, 1} cu următoarele proprietăţi (axiome):

a) pentru orice elemente x1, x2 M există relaţiile (parte stabilă):
x1 • x2$\in$M şi x1 + x2 $\in$M

b) operaţiile • şi + sunt comutative
x1 • x2 = x2 • x1 şi x1 + x2 = x2 + x1

c) operaţiile • şi + sunt asociative
(x1 • x2) • x3 = x1 • (x2 • x3) şi (x1 + x2) + x3 = x1 + (x2 + x3)

d) operaţiile • şi + sunt distributive una faţă de alta:
x1 • (x2 + x3) = x1 • x2 + x1 • x3;
x1 + (x2 • x3) = (x1 + x2) • (x1 + x3);

e) există element neutru pentru fiecare operaţie
x • 1 = x şi x+0 = x;

f) există element invers (complementar) pentru fiecare operaţie
x • x = x şi x+x = 1;

Valorile 0 şi 1 modelează logic stările circuitelor logice (digitale) care au două stări. O funcţie f: M -> M^n, M = {0, 1} se numeşte funcţie booleană. M^n reprezintă produsul cartezian M x M x M x ... x M.

Funcţiile logice sunt reprezentate cu ajutorul simbolurilor din figura 1.1:

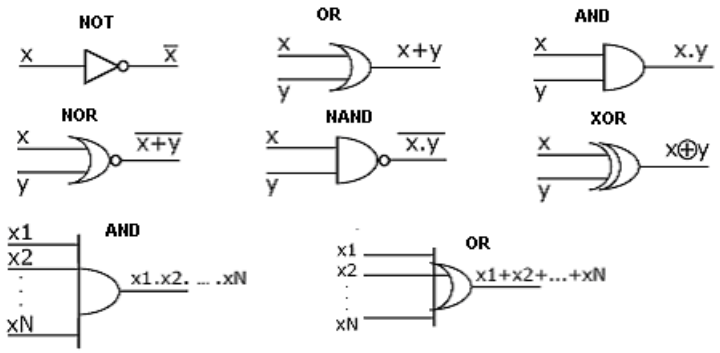


Figura 1.1. Reprezentarea funcţiilor logice

1.2 Coduri numerice

1.2.1 Reprezentări în virgulă fixă

Codurile numerice se împart în coduri binare şi coduri binar-zecimale. Codurile binare sunt reprezentări ale numerelor utilizate în sistemele digitale. Există următoarele tipuri de coduri binare:

1.2.1.1 Reprezentarea în modul şi semn (MS)

Fie $(b_{n-1}\ b_{n-2} \ldots b_0 \bullet b_{-1} \ldots b_{-m}) = (N)_2$

Valoarea numărului în baza 10 este:

$$N = (-1)^{b_{n-1}} \bullet \sum_{i=-m}^{n-2} b_i \bullet 2^i$$

Gama dinamică a numerelor reprezentate în MS este $\pm\ (2^{n-1} - 2^{-m})$ (valoarea maximă, în modul este atinsă pentru bi=1, i = - m, n-2, adică

$$\sum_{i=-m}^{n-2} 2^i = \frac{2^{-m}(2^{n-1+m} - 1)}{2^{-1}} = 2^{n-1} - 2^{-m}$$

Uzual m=0 (numere întregi). În acest caz - numere întregi reprezentate în MS pe n biţi - gama dinamică devine $\pm(2^{n-1} -1)$. În multe cazuri domeniul $\pm(2^{n-1} - 1)$ se scalează cu 2^{n-1}, adică

$N' = N/ 2^{n-1}$, deci virgula va fi plasată imediat după bitul de semn; se reprezintă pe n biţi numere fracţionare cu semn. Gama dinamică pentru reprezentarea MS a numerelor fracţionare este $\pm(1-1/2^{n-1})$.

1.2.1.2 Reprezentarea în complement de 1 (C1)

Dacă N>0 atunci valoarea lui zecimală este (în reprezentarea MS):

$(N)MS = 0 \bullet 2^{n-1} + \sum_{-n}^{n-2} b_i \bullet 2^i$

Numărul negativ (-N) se reprezintă în complement de 1 prin $(-N)_{C1} = 1\ b_{n-2}...b_{-m}$, unde $b_i=1-b_i$ (complement faţă de 1).

Pentru a reprezenta în C1 un număr negativ se consideră valoarea lui absolută şi în reprezentarea acesteia se complementează toţi biţii.

Exemplu: $(9)_{C1} = 00001001 = 9H$

$(-9)_{C1} = 11110110 = F6H = 15x16+6 = 246 = 255-9 = (2^n -1)- 9$

Rezultă o a doua regulă pentru obţinerea reprezentării valorilor negative ale unui număr întreg în C1: scăderea din 2^n-1 a valorii sale absolute.

Gama dinamică pentru reprezentarea C1 este $\pm(2^{n-1} - 2^{-m})$ (la fel ca în cazul reprezentării MS).

Dezavantajul reprezentărilor MS şi C1 este acela că valoarea 0 se exprimă în 2 moduri diferite; de exemplu pentru (MS) valoarea 0 este 000...0 sau 1000...0.

1.2.1.3. Reprezentarea în complement de 2 (C2)

Pentru numere pozitive reprezentarea este identică cu cea din (C1) şi (MS). Dacă numărul este negativ atunci $(N)_{C2} = 2^n - |N|$ = valoarea zecimală a codului numărului negativ N. Regula generală pentru N # 0 este $(-N)_{C2} = 2^n -N$.

Dacă numărul binar are reprezentarea în C2: b_{n-1}, b_{n-2}, ..., $b_0.b_{-1}...b_{-m}$, atunci valoarea sa în zecimal este dată de relaţia:

$$N = (-1)b_{n-1} \bullet 2^{n-1} + \sum_{i=-m}^{n-2} b_i \bullet 2^i$$

Gama dinamică pentru reprezentarea în C2 este: $-2^{n-1} <= N <= 2^{n-1} - 2^{-m}$ deoarece pentru N>=0 $b_{n-1}=0$ şi valoarea maximă se obţine pentru $b_i=1$, i = -m, n-2, deci $N_{max} = \sum_{i=-m}^{n-2} b_i \bullet 2^i = 2^{n-1} - 2^{-m}$, iar pentru N<0 $b_{n-1}=1$ şi valoarea minimă se obţine pentru $b_i=0$, i = -m,n-2, deci $N_{min} = -2^{n-1}$

Prin scalare cu 2^{n-1} se obţine reprezentarea în C2 a numerelor fracţionare. Gama dinamică (pentru m=0) este: $-1 <= N <= 1-\frac{1}{2^{n-1}}$

Există două moduri de obţinere a reprezentării lui (-N) în (C2):

1) Se adaugă 1 la reprezentarea binară în C1 alui (-N)

2) Se scade din 2n valoarea zecimală a lui N şi apoi se reprezintă în binar (ca număr fără semn).

În tabelul 1 sunt prezentate implementările MS, C1 şi C2 pentru numerele întregi pe 8 biţi. Se observă că în reprezentarea C2 numărul 0 se exprimă într-un singur mod spre deosebire de reprezentările MS şi C1.

1.2.2. Reprezentări în virgulă mobilă (flotantă)

Un număr N real se reprezintă în virgulă mobilă prin două numere binare: un număr fracţionar cu semn pe m biţi numit mantisă (M) şi un număr întreg cu semn pe n biţi numit exponent (E), conform relaţiei $N=M\bullet 2^E$. Dacă exponentul este reprezentat în C2 atunci: $-2^{n-1} \le E \le 2^{n-1}-1$, mantisa respectă inegalitatea $0 \le |M| \le 1-\frac{1}{2^{m-1}} < 1$.

Rezultă gama dinamică pentru reprezentarea în virgulă mobilă (numere diferite de 0): $\frac{1}{2^{m-1}} \bullet 2^{-2^{n-1}} \le |N| \le (1-\frac{1}{2^{m-1}}) \bullet 2^{2^{n-1}-1} < 2^{2^{n-1}} - 1$

(deoarece dacă N#0 cea mai mică mantisă este $\frac{1}{2^{m-1}}$, iar cel mai mic exponent este -2^{n-1}). Uzual n=8, m=24 (exponent pe un octet, iar mantisa pe 3 octeţi). Gama dinamică este mult mai mare decât în cazul reprezentărilor în virgulă fixă. Pentru n=8 rezultă o gamă cuprinsă între 10^{-38} şi 10^{38}.

Tabelul 1

Valoarea fără semn a codului de reprezentare a numărului			**Valoarea obţinută prin interpretarea codului binar ca număr cu semn**		
zecimal	**binar**	**hexazecimal**	**(MS)**	**(C1)**	**(C2)**
0	00000000	00	0	0	0
1	00000001	01	1	1	1
2	00000010	02	2	2	2
...	...	...	...	...	...
127	01111111	7F	127	127	127
128	10000000	80	0	- 127	- 128
...	...	...	...	...	...
254	11111110	FE	-126	-1	-2
255	11111111	FF	-127	0	-1

1.3. Elemente de aritmetică binară

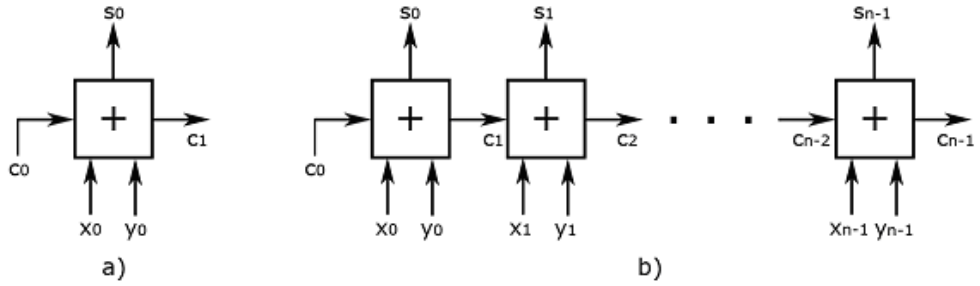
1.3.1. Adunarea şi scăderea numerelor binare

Adunarea şi scăderea numerelor binare de 1bit se realizează conform tabelului 2. in care T - transport în rangul superior şi I - împrumut din rangul superior.

Suma numerelor de 1bit se realizează cu ajutorul sumatorului de 1bit repreyentat în figura 1.2; pentru numere de n biți, suma se efectuează cu un sumator de n biți care reprezintă o extensie a sumatorului de 1bit.

Tabelul 2

x	y	x+y	T	x-y	I
0	0	0	0	0	0
0	1	1	0	1	1
1	0	1	0	1	0
1	1	0	1	0	0



a) Sumator pe un bit
b) Sumator pe 'n' biti
(se aduna x_{n-1} ... x_0 cu y_{n-1} .... respectiv y_0 si
transportul c0, iar rezultatul este S_{n-1} .... S_0 si transportul c_n)

Figura 1.2. Sumatorul binar

Funcționarea sumatorului de n biți, asociată bitului de rang i este ilustrată in tabelul 3.

Tabelul 3

	Intrări			Ieşiri	
	xi	**yi**	**Ci**	**Si**	**Ci+1**
0	0	0	0	0	0
1	0	0	1	1	0
2	0	1	0	1	0
3	0	1	1	0	1
4	1	0	0	1	0
5	1	0	1	0	1
6	1	1	0	0	1
7	1	1	1	1	1

1.3.2. Adunarea şi scăderea în C2

În reprezentarea în complement de 2 adunarea şi scăderea se pot realiza cu aceeaşi unitate aritmetică (sumatorul pe n biți); acest lucru constituie un avantaj important. Scăderea lui Y din X se realizează prin adunarea la X a complementului de 2 al lui Y.

Fie X,Y doi operanzi, R rezultatul operației de adunare (scădere) şi $(X)_{C2}$, $(Y)_{C2}$, $(R)_{C2}$ reprezentarea în complement de 2 a lui X,Y, respectiv R. Există următoarea relație: $(X+Y)_{C2} = (X)_{C2} + (Y)_{C2}$, cu condiția să nu existe depăşiri de gamă. Depăşirea (overflow) reprezintă, în cazul operațiilor aritmetice, modificarea bitului de semn datorită faptului că rezultatul operației este mai mare decât limitele gamei dinamice admise de reprezentarea numerică.

Suma algebrică se poate efectua în (C2) prin adunarea reprezentărilor în (C2) ale operanzilor. Rezultatul, pozitiv sau negativ va fi corect cu condiția să nu existe depăşiri de gamă.

Exemplu: n=8; G = (-128,127)

1. a>0, b>0; a=10, b=4; a+b=14; $(a)_{C2}=10$; $(b)_{C2}=4$; $(a)_{C2}+(b)_{C2}=14$

2. a<0, b<0; a = -1; b = -3; a+b = -4; $(a)_{C2}$ = 256 - 1 = 255 = FFH
$(b)_{C2}$ = 256-3=253=FDH deci $(-1)_{C2}+(-3)_{C2}$ = FF+FD=1|FCH = FCH =
= $(-4)_{C2}$

3. a≥0 b<0

3.1. a = 1; b = -3; a+b = -2; $(a)_{C2}$ = 1; $(b)_{C2}$ = 256 - 3 = FDH

$(1)_{C2} + (-3)_{C2} = 1+FD = FEH = (-2)_{C2}$

3.2. a = 3; b = -1; a+b = 2; $(a)_{C2}$ = 3; (b) = FFH

$(3)_{C2} + (-1)_{C2} = 3+FF = 1|02 = 2$

1.3.3. Depăşirea aritmetică (overflow)

Depăşirea aritmetică poate apare numai dacă operanzii sunt de acelaşi semn şi rezultatul este mai mare decât M sau mai mic decât (-M '), unde gama dinamică G=[-M ', M]. Pentru numere întregi pe n biți reprezentarea în C2 poate fi ilustrată ca în figura 1.3.

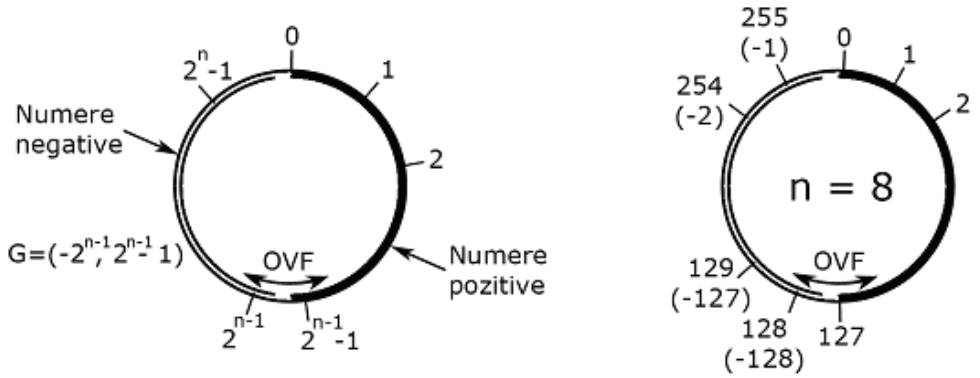


Figura 1.3. Depăşirea aritmetică

De exemplu (pentru n=8): 127+2 = 129>127 deci rezultă depăşire aritmetică. Efectuând adunarea în (C2) se obține: $(127)_{C2}$ + $(2)_{C2}$ = 7FH + 2H = 81H = 10000001 = $(-127)_{C2}$

S-a alterat bitul de semn, deci a apărut depăşire aritmetică.

1.3.4. Detectarea depăşirii aritmetice

Fie a, b $\in$ G.

1. Dacă a şi b sunt de semne contrare, nu apare depăşire aritmetică deoarece |a+b| $\in$ G.

2. Dacă a şi b sunt de acelaşi semn poate apare depăşire aritmetică. În tabelul asociat operației de adunare se observă, pentru cazurile 1 şi 6, dacă considerăm că

X_i şi Y_i reprezintă biţii de semn ai operanzilor, iar Si reprezintă bitul de semn al operaţiei, că:

$X_{n-1} = 0$, $Y_{n-1} = 0$, $C_{n-1} = 1$ --> $S_{n-1} = 1$, $C_n = 0$
$X_{n-1} = 1$, $Y_{n-1} = 1$, $C_{n-1} = 0$ --> $S_{n-1} = 0$, $C_n = 1$

Deoarece semnul rezultatului a fost alterat rezultă că a apărut overflow. Se observă că există relaţia (în ambele cazuri):

$C_{n-1} \oplus C_n = 1$. Notăm cu OVF depăşirea aritmetică, deci:

$OVF = C_{n-1} \oplus C_n$, unde $\oplus$ reprezintă suma modulo 2 (SAU EXCLUSIV).

1.3.5. Înmulţirea numerelor binare

Se va prezenta algoritmul de înmulţire a numerelor întregi cu semn reprezentate în C2 pe n biţi. Fie $(X)_{C2} = (-1)x_{n-1} \bullet 2^{n-1} + \sum_{i=0}^{n-2} x_i \bullet 2^i$ şi $(Y)_{C2} = (-1)y_{n-1} \bullet 2^{n-1} + \sum_{i=0}^{n-2} y_i \bullet 2^i$.

Se notează produsul numerelor X şi cu P, deci P=XY. Dacă X şi Y sunt reprezentaţi pe n biţi, produsul P va avea lungime dublă, 2n biţi.

Produsul P se poate calcula prin mai mulţi algoritmi; viteza de execuţie (de calcul) depinde de tipul algoritmului utilizat.

Există algoritmi de înmulţire cu un pas (înmulţitor cu n+n intrări şi 2n ieşiri - analog cu tabla înmulţirii - la ieşire se obţine imediat produsul) şi algoritmi în n paşi (n este numărul de biţi pentru operanzii ce se inmulţesc) secvenţiali. În acest caz viteza de lucru este mai mică, dar complexitatea circuitelor utilizate este mai redusă decât în cazul algoritmilor cu un pas.

În continuare se va studia un algoritm secvenţial de înmulţire a numerelor binare întregi reprezentate pe n biţi.

Luând în considerare modul de scriere a lui X rezultă:

$$P = XY = \{(-1)\, X_{n-1}2^{n-1} + X_{n-2}\, 2^{n-2} + ... + X_0\, 2^0\}Y =$$
$$= (-1)\, X_{n-1}2^{n-1}Y + X_{n-2}\, 2^{n-2}Y + ... + X_0\, 2^0 Y =$$
$$= \{2^{-1}\,[(-1)X_{n-1}(Y2^n) + 2^{-1}(X_{n-2}Y2^n + ... + 2^{-1}(X_0Y2^n)]\}$$

Înmulţirea cu 2^n este echivalentă cu o deplasare la stânga cu n biţi, iar înmulţirea cu 2^{-n} este echivalentă cu o deplasare la dreapta cu n biţi.

Deînmulţitul Y, înmulţit succesiv cu biţii înmulţitorului X, x_i, se adună la produsul P (care iniţial va lua valoarea 0), dupa care are loc o deplasare spre dreapta a lui P cu 1 bit, ş.a.m.d.

Rezultă algoritmul de înmulțire din figura 1.4.

În implementarea fizică a algoritmului, înmulțirile cu 2^n şi 2^{-1} se realizează prin deplasări la stânga sau la dreapta. Deplasarea la dreapta se execută cu conservarea bitului de semn (deplasare aritmetică). Dacă la adunarea anterioară a rezultat OVF=1 atunci înseamnă că bitul de semn deja s-a alterat, deci trebuie inversat.

Deci deplasarea la dreapta se face după următoarea regulă:
$p_{n-1} \oplus OVF \dashrightarrow p_{n-1}$, $p_{i+1} \dashrightarrow p_i$, i=0,..., n-2, unde p_i sunt biții produsului P.

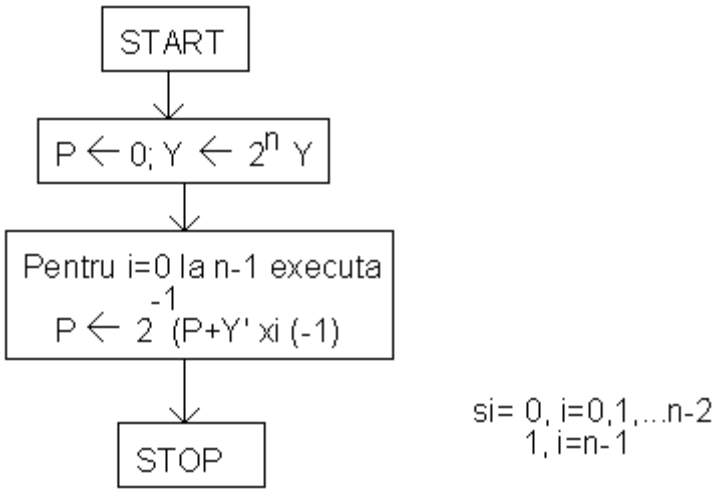


Figura 1.4. Algoritmul de inmultire binară

1.4. Circuite logice

1.4.1. Introducere

Circuitele logice (digitale) se pot clasifica în circuite combinaționale (CLC) şi circuite secvențiale (CLS). Circuitele combinaționale sunt caracterizate de faptul că ieşirea depinde de intrare prin intermediul unor funcții boolene; în cazul circuitelor secvențiale, ieşirea depinde de intrare şi de starea circuitului (care conține istoria intrărilor). Circuitele combinaționale nu au "memorie" spre deosebire de circuitele secvențiale.

Circuitele digitale reprezintă fizic (electric) variabile boolene (în tensiune sau curent). În figura 1.5 sunt prezentate principalele caracteristici ale circuitelor logice. Orice circuit logic este caracterizat de tensiunile de ieşire V_{OH}, V_{OL} şi de tensiunile acceptate la intrare V_{IH}, V_{IL} corespunzătoare nivelelor logice de "1" logic (high) şi de "0" logic (low). Aceste valori sunt alese astfel încât să se asigure interconectarea între circuitele logice (domeniu $V_{OH} \subset$ domeniu V_{IH} şi domeniu $V_{OL} \subset$ domeniu V_{IH}). Există o stare specială de ieşire numită starea de înaltă impedanță (high-Z) care permite conectarea mai multor ieşiri împreună.

1.4.2. Circuite combinaționale (sisteme de ordin 0)

Sistemul de ordin 0 nu are memorie; modificarea intrării produce modificarea ieşirii, impusă de funcția ***f***, aproape simultan (după un timp de propagare mic). Circuitele combinaționale sunt realizate cu porți logice.

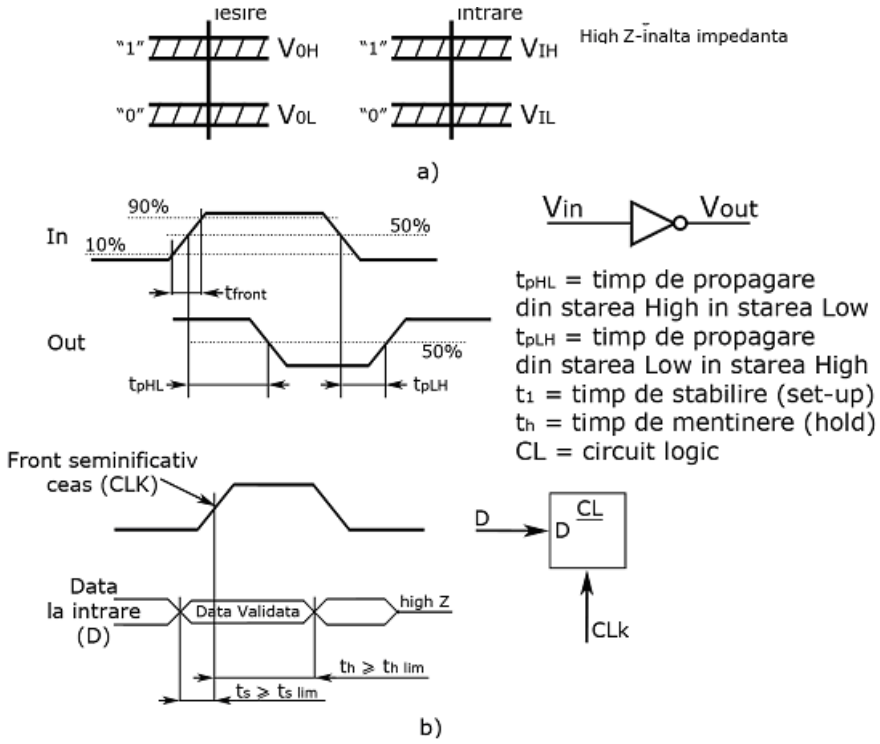


Figura 1.5. Caracteristicile circuitelor logice

Dintre circuitele combinaționale uzuale se remarcă: decodificatorul (demultiplexorul), multiplexorul, sumatorul binar, comparatorul, buffer-ul, codificatorul, memoria ROM şi ariile logice programabile.

1.4.3. Circuite secvențiale (sisteme de ordin 1)

Un sistem de ordin 1 este realizat din sisteme de ordin 0 conectate printr-o buclă de reacție. Sistemul de ordin 1 are memorie. Ca exemple de sisteme de ordin

1 se pot enumera: latch-ul asincron, latch-ul sincron, latch-ul adresabil, memoria RAM şi registrele.

1.4.4. Automate secvențiale (sisteme de ordin 2)

Se notează **X** = mulțimea semnalelor de intrare, **Y** = mulțimea semnalelor de ieşire, **Q** = mulțimea stărilor. Se defineşte funcția **Λ:X x Q --> Q** - funcția de tranziție a stărilor (funcția de tranziție internă) şi **δ:X x Q --> Y** - funcția de ieşire (funcția de tranziție externă). Mulțimea semnalelor de intrare este $\mathbf{X}=\{x_1,x_2,...x_n\}$, mulțimea semnalelor de ieşire este $\mathbf{Y}=\{y_1,y_2,...y_m\}$, iar mulțimea stărilor este $\mathbf{Q}=\{q_1,q_2,...,q_p\}$. În figura 1.6 sunt reprezentate două modele de automate secvențiale:

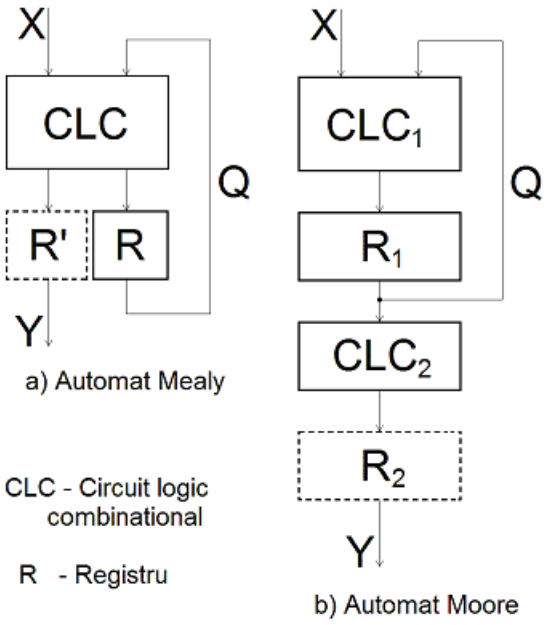


Figura 1.6. Automate secventiale

Vom nota $\mathbf{Q_{k+1}}$, $\mathbf{Q_k}$ vectorii de stare ai automatului la momentele *k+1*, respectiv *k*, $\mathbf{X_k}$ vectorul de intrare la momentul k şi $\mathbf{Y_{k+1}}$ vectorul de ieşire la momentul k+1. Pentru automatul de tip Mealy există relațiile $\mathbf{Q_{k+1}} = \Lambda(\mathbf{X_k}, \mathbf{Q_k})$ şi $\mathbf{Y_{k+1}} = \delta(\mathbf{X_k}, \mathbf{Q_k})$. Pentru un automat Moore $\mathbf{Q_{k+1}} = \Lambda(\mathbf{X_k}, \mathbf{Q_k})$ şi $\mathbf{Y_{k+1}} = \delta(\mathbf{Q_k})$.

Modelele Mealy şi Moore nu reprezintă clase de sisteme diferite, ci modele diferite. Un acelaşi automat poate fi modelat ca automat Moore (pentru număr mare de stări) sau ca automat Mealy (pentru număr mic de stări). Descrierea

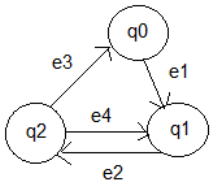
automatelor se poate face prin tablouri (tabele) sau prin grafuri de tranziții ca în figura 1.7.

Q(stare)/ X(intrare)	x_1	...	x_i	...	x_n
q_1					
...					
q_j			q_{ij}		
...					
q_p					

a) tabela starii urmatoare

Q (stare)	Y (ieşire)
q_1	y_1
...	...
q_p	y_p

b) tabela iesirilor



c) descrierea prin graf de tranzitii

q0, q1, q2 - stari

e1, e2, e3, e4 - evenimente de intrare

Figura 1.7. Descrierea automatelor secventiale

Ca exemple de automate secventiale se pot enumera: bistabilul T, bistabilul D, numărătorul.

Automatele secvenţiale vor fi structurate astfel încât să se simplifice circuitele CLC, din structura acestora şi să se poată modifica evoluția automatului prin citirea unor informații din exterior. Se ajunge la un nou concept – automat secvențial programabil – care stă la baza arhitecturii unui procesor sau microcontroler.