TEXTOS DOCENTS

126

DISSENY DIGITAL BÀSIC

Atilà Herms Berenguer
Sebastià A. Bota Ferragut
Josep Bosch Estrada
Josep M. López Villegas

Departament d'Electrònica

UNIVERSITAT DE BARCELONA

TEXTOS DOCENTS

126

DISSENY DIGITAL BÀSIC

Atilà Herms Berenguer
Sebastià A. Bota Ferragut
Josep Bosch Estrada
Josep M. López Villegas

Departament d'Electrònica

Publicacions i Edicions

UNIVERSITAT DE BARCELONA

ÍNDEX

Introducció v

CAPÍTOL I

1. Representació d'informació 1
1.1 Introducció 1
1.2 Representació dels nombres. Sistemes de numeració 1
1.3 Conversió de nombres 3
1.3.1 Conversió de qualsevol base a base 10 3
1.3.2 Conversió de base 10 a qualsevol base 3
1.3.3 Conversió de base 2 a octal i hexadecimal 5
1.3.4 Codis decimals codificats en binari, BCD 5
1.4 Representació de nombres negatius 6
1.4.1 Representació signe-mòdul 6
1.4.2 Representació de nombres negatius amb complements 7
1.5 Representació de caràcters alfanumèrics i de control 9
1.6 Representació de nombres en punt flotant 9
1.6.1 Real curt 11
1.6.2 Real llarg 11
1.6.3 Aritmètica en punt flotant 12

CAPÍTOL II

2. Àlgebra de Boole 13
2.1 Introducció. 13
2.2 Definicions bàsiques 14
2.3 Teoremes de l'àlgebra de Boole 14
2.3.1 Dualitat 14
2.3.2 Teoremes bàsics 15
2.4 Funcions de commutació 15
2.4.1 Àlgebra de les funcions de commutació 16
2.4.2 Altres operadors lògics 16
2.4.3 Conjunt complet d'operadors 17
2.5 Portes lògiques digitals 17
2.6 Forma estàndard de les funcions lògiques 18
2.7 Maxtermes i mintermes 20
2.7.1 Suma de productes 21
2.7.2 Producte de sumes 21

CAPÍTOL III

3. Simplificació de funcions lògiques 23
3.1 Introducció 23
3.2 Representació de les funcions de Boole en mapes de Karnaugh 24
3.2.1 Simplificació de funcions amb mapes de Karnaugh 26
3.2.2 Mapa de Karnaugh en suma de productes 26
3.2.3 Mapa de Karnaugh en producte de sumes 28
3.3 Funcions especificades incompletament. 30
3.4 Mapes de Karnaugh amb variables entrades al mapa 30
3.5 Minimització tabular 34
3.5.1 Mètode de Quine-McCluskey. Determinació dels implicants primers 35

3.5.2 Selecció d'un conjunt òptim d'implicants primers 37
3.5.3 Funcions especificades incompletament 39

CAPÍTOL IV
4. Blocs lògics combinacionals 43
4.1 Blocs lògics combinacionals 43
4.1.1 Senyals de dades i senyals de control 44
4.2 Decodificadors i codificadors 45
4.2.1 Decodificadors 46
4.2.2 Codificadors 47
4.2.3 Convertidors de codis 48
4.3 Multiplexors i desmultiplexors 49
4.3.1 Multiplexors 49
4.3.2 Implementació de funcions lògiques amb multiplexors 50
4.3.3 Teorema d'expansió de Shanon 50
4.3.4 Desmultiplexors 53
4.4 Comparador binari 53

CAPÍTOL V
5. Lògica programable 57
5.1 Introducció 57
5.2 PLA (*Programmable Logic Array*) 57
5.3 Taula de programació d'una PLA 58
5.4 PAL (*Programmable Array of Logic*) 60
5.5 Tipus de PAL 62
5.6 Programació 64

CAPÍTOL VI
6. Circuits aritmètics 67
6.1 Suma binària 67
6.1.1Estructura del sumador d'n bits 69
6.1.2 Sumadors amb càlcul anticipat de l'arrossegament 70
6.2 Resta binària 71
6.3 Sumadors BCD 72
6.4 Unitats Aritmetico-Lògiques 74
6.5 Multiplicació binària 78
6.5.1 Multiplicació per acumulació de productes parcials 79
6.5.2 Agrupació de multiplicadors 80
6.5.3 Multiplicació en taula LUT 81
6.6 Divisió binària 82
6.7 Components estàndard 82
6.7.1 Sumadors 83
6.7.2 Unitats de generació anticipada de *carry* 83
6.7.3 Unitat Aritmetico-Lògiques 83
6.7.4 Multiplicadors 83

CAPÍTOL VII
7. Sistemes seqüencials, circuits biestables 85
7.1 Circuits multivibradors 85
7.1.1 Multivibrador monoestable 85
7.1.2 Multivibrador aestable 86
7.1.3 Multivibrador biestable 86
7.2 Biestables 87

7.2.1Biestable SR NOR ... 87
7.2.2 Biestable SR NAND ... 89
7.2.3 Comportament temporal anormal ... 90
7.3 *Latch* ... 90
7.3.1 *Latch* SR ... 90
7.3.2 Entrades directes o asincròniques ... 91
7.3.3 El *latch* D ... 91
7.4 *Flip-flop* ordenador seguidor ... 92
7.4.1 *Flip-flop* SR ordenador seguidor (*master-slave*) ... 93
7.4.2 *Flip-flop* D ordenador seguidor (*master-slave*) ... 94
7.4.3 *Flip-flop* JK ordenador seguidor (*master-slave*) ... 95
7.5 *Flip-flop* actiu per flanc ... 97
7.5.1 *Flip-flop* D actiu per flanc ... 97
7.5.2 *Flip-flop* JK actiu per flanc de baixada ... 98
7.5.3 *Flip-flop* T ... 99
7.6 Sumari ... 100

CAPÍTOL VIII

8. Registres i comptadors ... 101
8.1 Registres ... 101
8.1.1 Registre d'entrada paral·lel, de sortida paral·lel ... 101
8.1.2 Registre de desplaçament (*Shift register*) ... 102
8.1.3 Registre de desplaçament bidireccional amb càrrega paral·lela ... 103
8.1.4 Transferència de dades entre registres ... 104
8.2 Comptadors ... 104
8.2.1 Comptadors asíncrons ... 105
8.2.2 Comptador asíncron binari ... 105
8.2.3 Comptador asíncron no binari ... 106
8.2.4 Comptadors síncrons ... 107
8.2.5 Comptador síncron binari ... 107
8.2.6 Síntesis de comptadors de mòdul arbitrari ... 108
8.2.7 Comptador síncron reversible ... 109
8.2.8 Bloqueig de comptadors ... 111
8.2.9 Comptadors d'anell ... 111
8.2.10 Comptadors d'anell trenat ... 112
8.3 Generadors de seqüència ... 113
8.4 Components integrats estàndard ... 115

CAPÍTOL IX

9. Sistemes seqüencials síncrons ... 117
9.1 Màquines d'estat finit ... 117
9.2 Mètodes de descripció de màquines d'estat finit ... 119
9.2.1 Circuits de Moore ... 119
9.2.2 Circuits de Mealy ... 120
9.2.3 *Timming* ... 121
9.3 Síntesis de màquines d'estat ... 122
9.3.1 Exemple 1: màquina de Moore ... 123
9.3.2 Exemple 2: màquina de Mealy ... 127
9.4 Simplificació de diagrames d'estat ... 129
9.4.1 Mètode de la taula d'implicants ... 131
9.4.2 Assignació d'estats ... 134

CAPÍTOL X

10. Sistemes seqüencials asincrònics o de realimentació directa 135
10.1 Introducció 135
10.2 Anàlisis de circuits asincrònics de mode fonamental 136
10.3 Derivació de la taula d'excitació 136
10.4 Taula de transició 137
10.5 Carreres 139
10.6 Errors estàtics 139
10.7 Errors dinàmics 140
10.8 Síntesis de sistemes seqüencials asincrònics 141
10.8.1 Derivació de la taula de flux primitiu 142
10.8.2 Derivació de la taula d'estats reduïda, eliminació d'estats redundants 143

BIBLIOGRAFIA 145

INTRODUCCIÓ

L'assignatura de Fonaments de Commutació

Aquest text intenta ser una guia i un punt de referència per als alumnes de l'assignatura **Fonaments de Commutació**, assignatura bàsica del pla d'estudis d'enginyeria electrònica de la Universitat de Barcelona.

Els coneixements que es presenten en l'assignatura de Fonaments de Commutació són emprats directament en altres assignatures d'Enginyeria Electrònica. En concret, aquesta assignatura és bàsica per al desenvolupament de les assignatures:

Disseny Microelectrònic I
Disseny Microelectrònic II
Sistemes Digitals
Electrònica dels Circuits Digitals

i està directament relacionada amb les assignatures:

Dispositius Electrònics
Disseny de Circuits Integrats Específics
Anàlisi de Circuits Integrats

D'altra banda, l'assignatura està concebuda de manera que pràcticament no es necessita cap tipus de coneixement previ. Per aquest motiu i pels seus continguts intrínsecs, és una assignatura que pot ser oferta com a optativa o de lliure elecció als estudiants d'altres ensenyaments.

Objectius de l'assignatura

Des d'un punt de vista general, el principal objectiu d'aquest text és donar a conèixer els fonaments bàsics del que avui dia es coneix amb el nom d'*electrònica digital*. Concretant més, els objectius bàsics que té l'assignatura i que intenta recollir aquest text són:

- Conèixer la importància i els camps d'aplicació dels sistemes digitals.
- Proporcionar els coneixements suficients per entendre el funcionament dels circuits electrònics digitals.
- Introduir el sistema binari de numeració, l'àlgebra de Boole i les funcions de commutació.
- Poder analitzar i sintetitzar circuits lògics dedicats a la implementació de funcions booleanes.
- Introduir l'alumne en la utilització de circuits digitals comercials de baixa/mitjana complexitat.
- Presentar el concepte de sistemes seqüencials, elements de memòria i màquines d'estat.
- Presentar l'anàlisi i la síntesi de sistemes lògics seqüencials.

Desenvolupament temàtic de l'assignatura

El text consta de deu temes, organitzats en tres grans blocs:

A. FONAMENTS DE LA TEORIA DE LA COMMUTACIÓ

Aquesta part serveix d'introducció a l'assignatura i tracta dels fonaments matemàtics dels sistemes digitals: representació binària, àlgebra de commutació, manipulació de funcions de commutació.

Els temes d'aquesta primera part són:

Tema 1: Representació d'informació

Tema 2: Àlgebra de Boole

Tema 3: Simplificació de funcions lògiques

B. CIRCUITS COMBINACIONALS

En la segona part es passa del concepte de *funció de commutació* al de *circuit lògic* (circuit electrònic que implementa de manera física el comportament representat per la funció de commutació). Es defineix el concepte de *circuit lògic combinacional,* juntament amb els seus criteris de disseny, les seves possibilitats ...

Els temes corresponents són:

Tema 4: Circuits lògics combinacionals

Tema 5: Lògica programable

Tema 6: Circuits aritmètics

C. CIRCUITS SEQÜENCIALS

Els circuits seqüencials, aquells en què el seu estat de sortida depèn de la història prèvia del circuit, són tractats en la tercera part de l'assignatura. Seguint una aproximació semblant a la que s'utilitza en la segona part, s'estudiaran els blocs bàsics seqüencials (biestables i *flip-flop*), les principals famílies de circuits integrats seqüencials i, els fonaments per a la síntesi de màquines d'estat finit i finalment els sistemes seqüencials asincrònics.

Els temes corresponents són:

Tema 7: Sistemes seqüencials: circuits biestables

Tema 8: Registres i comptadors

Tema 9: Màquines d'estat finit

Tema 10: Sistemes seqüencials asincrònics

Aspectes pràctics de l'assignatura Fonaments de Commutació

Al llarg del text s'intenta reflectir l'aspecte pràctic de la matèria. Així, es comenten les possibilitats reals d'implementació que ofereix la tecnologia actual. A més, les sessions teòriques són completades amb sessions experimentals en què l'alumne posa en pràctica els coneixements teòrics adquirits; en concret, les pràctiques estan concebudes com un exercici de disseny de diferents sistemes digitals, evolucionant des del disseny de circuits de baixa complexitat fins al disseny de circuits de major complexitat; per tant, posant en pràctica els aspectes teòrics d'aquest text. Aquest fet pretén ressaltar el caire pràctic que s'intenta donar en general a totes les assignatures d'Enginyeria Electrònica.

CONTRAPORTADA

Aquest text ha estat redactat per cobrir les necessitats d'un curs introductori de disseny digital. Sota aquest punt de vista, es desenvolupen els principis fonamentals del disseny digital, des de la representació binària de la informació fins a l'estudi dels sistemes seqüencials, passant pels mètodes de síntesi de funcions combinacionals. S'intenta agafar com a referència una metodologia de disseny actual, lligada a les característiques pròpies de la tecnologia VLSI.

CAPÍTOL I

1. REPRESENTACIÓ D'INFORMACIÓ

1.1. Introducció

Un **sistema digital** és qualsevol sistema de transmissió o processament d'informació en el qual la informació es representa mitjançant quantitats físiques restringides a valors discrets. En contrast amb els sistemes analògics, que tracten amb magnituds que poden variar de forma contínua, un sistema digital que vulgui tractar, mesurar i/o controlar variables contínues ha de representar-les primer en format digital, és a dir, ha de restringir-les a valors discrets.

Com que és fàcil realitzar components físics capaços de diferenciar dos estats, generalment dos nivells de tensió, la base 2 és la més utilitzada per a les variables discretes, encara que existeixen sistemes molt determinats que treballen amb altres bases. Així, definim la **variable binària** com la variable representada en base 2. El sistema matemàtic que utilitza dos dígits és anomenat **sistema binari**. Els fonaments dels sistemes binaris van ser establerts pel matemàtic britànic George Boole (1815-1864) en el seu tractat *An Investigation of the Laws of Thought* (1854). La seva teoria, basada en la lògica aristotèlica, tracta dos conceptes: veritable i fals (que podem representar com a 0 i 1), i va trobar una aplicació pràctica vuitanta-quatre anys més tard, quan Shannon va desenvolupar la seva teoria de la commutació basada en l'àlgebra de Boole.

1.2. Representació dels nombres. Sistemes de numeració

En un sistema de base b, un nombre N es pot representar mitjançant un polinomi de potències de la base:

$$N = \underbrace{a_n b^n + a_{n-1} b^{n-1} + ... + a_0 b^0}_{\text{part entera}} + \underbrace{a_{-1} b^{-1} + ... + a_{-p} b^{-p}}_{\text{part fraccionària}}$$

on $0 \le a_i \le b$

n + 1 és el nombre de dígits enters
p és el nombre de dígits fraccionaris

Exemple:

base 10: $87.54_{10} = 8x10^1 + 7x10^0 + 5x10^{-1} + 4x10^{-2}$
base 2: $1011.11_2 = 1x2^3 + 0x2^2 + 1x2^1 + 1x2^0 + 1x2^{-1} + 1x2^{-2}$

En un sistema binari els termes a_i només poden valer 0 o 1 i reben el nom de **bits** (el bit és l'element mínim d'informació). També es defineix com a **byte** el grup de vuit bits i com a **paraula** el conjunt de dos bytes. També són unitats usuals els **kilobits** (1kb = 2^{10} bits = 1024 bits), el **kilobyte** (1 kB = 1024 bytes),el **megabit** i el **megabyte** (1 Mb = 2^{20} bits = 1024 kb = 1.048.576 bits).

Els sistemes més usats, a part del decimal, són el binari (b = 2), l'octal (b = 8) i l'hexadecimal (b = 16). Els nombres en base hexadecimal es representen amb els dígits següents:

hexadecimal	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
decimal	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Decimal	Binari	Octal	Hexadecimal
0	0	0	0
1	1	1	1
2	10	2	2
3	11	3	3
4	100	4	4
5	101	5	5
6	110	6	6
7	111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F
16	10000	20	10
17	10001	21	11
18	10010	22	12
19	10011	23	13
20	10100	24	14
32	100000	40	20
50	110010	62	32
60	111100	74	3C
64	1000000	100	40
100	1100100	144	64
255	11111111	377	FF
1000	1111101000	1750	3E8

Taula 1.1

A la taula 1.1 es representen uns quants nombres en base decimal, binària, octal i hexadecimal.

1.3. Conversió de nombres

1.3.1. Conversió de qualsevol base a base 10

La conversió d'un nombre de qualsevol base a decimal es realitza representant el nombre mitjançant el seu polinomi equivalent, amb la seva pròpia base i operant-lo en base 10.

Exemple

De base 2 a base 10:

$$1101.11_2 = 1.2^3 + 1.2^2 + 1.2^0 + 1.2^{-1} + 1.2^{-2} = 13.75_{10}$$

De base 16 a base 10:

$$1FA7.32_{16} = 1.16^3 + 15.16^2 + 10.16^1 + 7.16^0 + 3.16^{-1} + 2.16^{-2} = 8103.195313_{10}$$

1.3.2. Conversió de nombres en base 10 a qualsevol base

En aquest cas, la part entera (N_E) i la fraccionària (N_F) es tracten per separat:

Part entera

La part entera es pot representar com a $N_E = N_E' \cdot b + R$. Si escrivim N_E en base b, resulta:

$$N_E = a_{n-1}b^{n-1} + a_{n-2}b^{n-2} + ... + a_0b^0 = (a_{n-1}b^{n-2} + a_{n-2}b^{n-3} + ... + a_1)\cdot b + a_0$$

És a dir, la resta obtinguda en dividir N_E per b és el coeficient a_0. Anomenem

$$N'_E = a_{n-1}b^{n-2} + a_{n-2}b^{n-3} + ... + a_1$$

i tornem a dividir per b:

$$N'_E/b = a_{n-1}b^{n-3} + a_{n-2}b^{n-4} + ... + (\mathbf{a_1/b})$$

És a dir, la resta resultant de dividir per segona vegada correspon al terme a_1. Si continuem el procés aniríem obtenint tots els dígits de N_E en la base b, sent l'últim quocient el més significatiu.

Exemple:

De base 10 a base 2. Anem a convertir el nombre 524_{10}:

	quocient	resta	coeficient
524 : 2	= 262	**0**	= a_0
262 : 2	= 131	**0**	= a_1
131 : 2	= 65	**1**	= a_2
65 : 2	= 32	**1**	= a_3
32 : 2	= 16	**0**	= a_4
16 : 2	= 8	**0**	= a_5
8 : 2	= 4	**0**	= a_6
4 : 2	= 2	**0**	= a_7
2 : 2	= **1** = a_9	**0**	= a_8

$524_{10} = 1000001100_2$

Part fraccionària

En aquest cas tenim que:

$$N_F = a_{-1}b^{-1} + a_{-2}b^{-2} + ... + a_{-p}b^{-p}$$

Si ho multipliquem tot per b:

$$N_Fb = \mathbf{a_{-1}} + a_{-2}b^{-1} + ... + a_{-p}b^{-p+1}$$

La part entera obtinguda de la multiplicació correspon al coeficient a_{-1} (el més significatiu).
Diem:

$$N'_F = a_{-2}b^{-1} + ... + a_{-p}b^{-p+1}$$

Tornem a multiplicar per b:

$$N'_Fb = \mathbf{a_{-2}} + ... + a_{-p}b^{-p+1}$$

La part entera correspon a a_{-2}, i així successivament es van obtenint tots els coeficients fins que es faci 0. Cal tenir en compte que una fracció decimal amb un nombre finit de dígits pot convertir-se, en una altra base, en un nombre infinit de dígits.

Exemple:

Conversió de 0.825_{10} a base 2

	resultat	part entera
0.825 x 2 =	1.65	**1**
0.650 x 2 =	1.30	**1**
0.300 x 2 =	0.60	**0**
0.600 x 2 =	1.20	**1**
0.200 x 2 =	0.40	**0**
0.400 x 2 =	0.80	**0**
0.800 x 2 =	1.60	**1**

$0.825_{10} = 0.1101001..2$

Aleshores: $524.825_{10} = 1000001100.1101001..{}_2$

Exemple:

Conversió de base 10 a base 16**.** $58506.8435_{10} \Rightarrow 16$

Part entera

58506 :	16	= 3656	resta	10 (**A**)
3656 :	16	= 228		**8**
228 :	16	= 14 (**E**)		**4**

Part fraccionària

0.8435 :	16	= 13.496	resta	**D**
0.496 :	16	= 7.936		**7**
0.936 :	16	= 14.976		**E**
0.976 :	16	= 15.616		**F**

$58506.8435_{10} = E48A.D7EF_{16}$

1.3.3. Conversió de base 2 a octal i hexadecimal

L'interès pels sistemes octal (base 8) i hexadecimal (base 16) prové del fet que la conversió de nombres entre base 2, base 8 i base 16 és molt simple, ja que $2^3 = 8$ i $2^4 = 16$. Això permet agrupar els nombres en grups de tres i quatre bits i convertir-los directament. Per convertir un nombre de base 8 a 2 (2 a 8) es converteix cada xifra en el seu equivalent binari (i viceversa):

Exemple

(base 8 a base 2) $3_8 = 011_2$
$2_8 = 010_2$
$5_8 = 101_2$
$6_8 = 110_2$

Aleshores: $325.6_8 = 011010101.110_2$

(base 2 a base 8) 11011001.101100

011	011	001 .	101	100	
3	3	1	5	4	= 331.54

En base 16 la conversió es fa igual però amb grups de quatre bits. Aquesta base s'utilitza molt per designar adreces de memòria, perquè els nombres són més manejables i fàcilment convertibles a binari.

Exemple

(hexadecimal a binari) FE91 = 1111 1110 1001 0001

1.3.4. Codis decimals codificats en binari (BCD)

El sistema binari és molt útil. A vegades, però, és convenient utilitzar altres sistemes, especialment el decimal, sempre conservant els avantatges del binari. D'aquí la utilització del codi BCD (*Binary Coded Decimal*), és a dir, decimal codificat en binari. També se'l coneix com 8421 BCD (pel pes de cada dígit). En aquest codi, cada dígit decimal es representa pel seu equivalent binari (4 bits); per exemple, el nombre 641 es representa per:

$641_{10} \Rightarrow 0110\ 0100\ 0001$ (BCD natural)
6 4 1

Aquest codi és útil per representar nombres en *displays* numèrics LED o LCD, ja que cada grup de 4 bits s'introdueix a les entrades del dígit corresponent. L'inconvenient és que calen 4 bits per dígit; així, amb 12 bits es pot representar fins al 999, mentre que en binari es pot arribar a $2^{12}-1 = 4.095$.

Un altre codi de la família BCD és l'anomenat BCD excés-3, que s'obté sumant 0011 (3) a cada combinació BCD; així, el 5 (0101 en binari) es converteix en 1.000. Aquest codi sol ser utilitzat en detecció d'errors. A la taula 2 es presenten altres codis binaris amb pesos assignats.

Els codis 84(-2)(-1) i el 2421 són codis simètrics: el darrer nombre és el complement a 1 (es veu més endavant) del primer, el penúltim del segon, etc. Existeixen altres codis sense pes assignat a cada posició del bit. Per exemple, els codis de distància unitat, on els nombres successius (adjacents) difereixen només d'un bit. El codi Gray n'és un exemple; quan s'apliquen les regles de l'àlgebra de Boole se'n veu la utilitat.

Dígit decimal	(BCD) 8421	Excés 3	84(-2)(-1)	BCD Aiken 2421	Biquinari 5043210	Codi Gray
0	0000	0011	0000	0000	0100001	0000
1	0001	0100	0111	0001	0100010	0001
2	0010	0101	0110	0010	0100100	0011
3	0011	0110	0101	0011	0101000	0010
4	0100	0111	0100	0100	0110000	0110
5	0101	1000	1011	1011	1000001	0111
6	0110	1001	1010	1100	1000010	0101
7	0111	1010	1001	1101	1000100	0100
8	1000	1011	1000	1110	1001000	1100
9	1001	1100	1111	1111	1010000	1101
10	-	-	-	-	-	1111
11	-	-	-	-	-	1110
12	-	-	-	-	-	1010
13	-	-	-	-	-	1011
14	-	-	-	-	-	1001
15	-	-	-	-	-	1000

Taula 1.2

1.4. Representació de nombres negatius

Evidentment, és necessari desenvolupar un mètode per poder representar el signe d'una manera que resulti manejable per als sistemes digitals. Hi ha tres mètodes:

- Representació signe - mòdul
- Representació en complement a 1
- Representació en complement a 2

1.4.1. Representació signe - mòdul

Suposem una informació tractada per un sistema digital. El nombre de bits que constitueixen la informació que accepta l'ordinador està definit i és fix (longitud de la paraula). Com que els ordinadors només empren 0 i 1 lògics (és a dir, dos nivells de tensió), el concepte de signe s'ha de representar per un d'aquests valors. Una solució immediata és reservar un dels bits d'informació, habitualment el més significatiu (el de més pes), per indicar el signe (0/+; 1/-).

La resta de bits corresponen al mòdul del nombre.

$$N_{10} = b_{n-1} \cdot b_{n-2} \ldots b_0 = (-1)^{b_{n-1}} (b_{n-2} \cdot 2^{n-2} + \ldots + b_0)$$

Cal tenir en compte que el 0 pot ser representat de dues maneres: 0000 i 1000. Amb signe - mòdul es poden representar nombres entre:

$$(1 - 2^{n-1}) \leq N_{10} \leq (2^{n-1} - 1) \text{ ; on } n = \text{núm. de bits}$$

Per exemple, amb 4 bits podem representar des del -7 fins al 7 (1111, 0111).

1.4.2. Representació de nombres negatius amb complement

Una alternativa consisteix a representar l'enter negatiu mitjançant un nombre complementari, de manera que la resta de dos nombres es transforma directament en la suma d'un amb el complement de l'altre. Així se simplifiquen molt els processos aritmètics. Hi ha dos tipus de complements.

1.4.2.1. Representació en complement a 1

Consisteix a representar els nombres enters negatius amb un altre nombre format a partir d'una constant K i el mòdul del nombre. Així, si el nombre que s'ha de representar és -Z, el seu complement es calcula com K-Z (observeu que K-Z, a més, correspon a un cert nombre positiu). El valor de K depèn del nombre de bits de la paraula i del tipus de complement escollit.

Així, en complement a 1 (Ca1):

$$K = 2^{n-1} - 1$$

On n és el nombre de bits que s'utilitzen (incloent-hi el bit de signe). El bit de signe es continua usant, ja que ens indica si un determinat nombre correspon a un positiu o a un negatiu. Aleshores, el Ca1 d'un nombre -Z es fa situant un 1 en el bit de signe i representant en el mòdul el resultat de l'operació:

$$K - Z = (2^{n-1} - 1) - Z$$

És a dir, com que posar un 1 a la posició n-1 equival a sumar 2^{n-1}, tenim que l'expressió analítica del Ca1 d'un nombre és:

$$Ca1(Z) = 2^{n-1} + (2^{n-1} - 1) - Z$$

Per exemple (en representació de sis bits):

+12 = 001100

El primer bit és 0; això indica que el valor és positiu. En canvi, -12 està representat pel Ca1(12). Calculem-ho:

$K-12 = (2^{6-1} - 1) - 12 = 31 - 12 = 19 = 10011_2$
Ca1(12) = 110011

El primer bit, 1**,** indica que és un nombre negatiu, que està complementat i que per saber el nombre que li correspon cal descomplementar-lo. En aquest cas el 0 està representat doblement per 000000 i per 111111.

Una manera senzilla d'obtenir el complement a 1 consisteix a canviar els 0 per 1 i viceversa:

+12 = 001100 -12 = 110011

El marge de valors que es pot representar amb n bits és:

$$-(2^{n-1} - 1) \leq N \leq (2^{n-1} - 1)$$

Vegem com es realitzen les operacions de suma i resta; en concret, l'operació resta equival a sumar el valor complementat del subtrahend.

Suposem que treballem amb paraules de sis bits. Farem l'operació 25-13 en Ca1:

25	=	011001	Minuend
-13	=	110010	Cal del subtrahend
(1)		001011	Suma
		1	Afegim el bit d'arrossegament final
12	=	001100	Resultat

En complement a 1, quan el resultat és positiu es produeix un bit d'arrossegament (*carry*), que cal sumar al resultat parcial per tal d'obtenir el resultat correcte (aquest bit s'anomena *end-around-carry*).

Quan el resultat és negatiu no es produeix aquest desbordament. Per exemple, 14-22:

14	=	001110	
-22	=	101001	Ca1 de 22
-8	=_	110111	Ca1 de 8

Cal tenir present que quan se sumen dos nombres i el resultat excedeix el marge de representació, es produeix un desbordament aritmètic acompanyat d'un canvi anòmal en el signe del resultat que ha de ser detectat per un circuit especial.

1.4.2.2. Representació en complement a 2

La representació és igual que en el cas anterior, però ara la constant K val, per a una representació d'n bits:

$$K = 2^{n-1}$$

Aleshores, el Ca2 d'un nombre -Z es fa situant un 1 en el bit de signe i representant en el mòdul el resultat de:

$$K - Z = 2^{n-1} - Z$$

És a dir, l'expressió analítica és ara:

$$Ca2(Z) = 2^{n-1} + 2^{n-1} - Z$$

Per exemple (amb sis bits): $+12 = 001100$

Per calcular en complement a 2 hem de fer:

$K-12 = (2^{6-1} - 12) = (32 - 12) = 20 = 10100$
$Ca2(12) = 110100$

Com abans, el primer bit indica el signe. En aquest cas, el 0 té una única representació: 000000.
El Ca2 d'un nombre pot obtenir-se sumant 1 al Ca1.

$$Ca1(12) + 1 = 110011 + 1 = 110100 = Ca2(-12)$$

Aquesta és la manera senzilla de calcular el Ca2 d'un nombre binari: donat un nombre en binari i anant de dreta a esquerra (del bit menys significatiu al més significatiu), es deixen tots els 0 iguals fins al primer 1, que tampoc es canvia. A partir d'aquest primer 1 s'intercanvien els 1 per 0 i viceversa.

+12	=	0 0 1 1 0 0
Ca2(12)	=	1 1 0 1 0 0

El marge de valors és una mica superior: $-(2^{n-1}) \leq N \leq (2^{n-1} - 1)$. Com en el Ca1, veiem com són les operacions. Si fem 25-13:

25	=	011001	Minuend
-13	=	110011	Ca2 del subtrahend
12	= (1)	001100	Resultat

En aquest cas també es produeix desbordament quan el resultat és positiu, però no cal sumar-lo, simplement s'ignora. Si, en canvi, el resultat és negatiu, el bit d'arrossegament és 0.

14	=	001110	
-22	=	101010	Ca2 de 22
-8	=	111000	Ca2 de 8

1.5. Representació de caràcters alfanumèrics i de control

Existeixen diversos codis estàndard per a la representació de caràcters alfanumèrics, de puntuació, de control, símbols matemàtics i tota mena de caràcters no numèrics. Els primers codis utilitzaven cinc bits/caràcter. Així només es podien representar trenta-dos caràcters. Actualment els codis estàndard tenen set o vuit bits/caràcter, de manera que poden representar 128 o 256 caràcters, respectivament. Un d'aquests codis és el codi **ASCII** (*American Standard Code for Information Interchange*), que s'utilitza molt per representar caràcters a partir d'un teclat. En aquest codi s'utilitzen set bits per representar tots els caràcters numèrics i les lletres majúscules i minúscules, com també els caràcters usuals de puntuació i control d'impressores, alimentació de línia, tabulador, retorn de carro, etc. En molts ordinadors aquest esquema s'ha estès i s'hi ha afegit un vuitè bit, de manera que s'han obtingut 128 caràcters més, que generalment són símbols gràfics.

A la taula 1.3 es mostra el conjunt de codis ASCII. Els trenta-dos primers caràcters són caràcters de control de format d'impressió, de control de transmissió, separadors d'informació, etc. A la mateixa taula es presenta un altre codi utilitzat per a caràcters alfanumèrics anomenat EBCDIC (*Extended BCD Interchange Code*), que utilitza els vuit bits d'un byte per representar la informació.

Sovint en alguns ordinadors els codis usats per a entrada són diferents dels de sortida. Suposem, per exemple, que el teclat usat per a entrada de dades produeixi un codi EBCDIC i que la pantalla per representar dades treballi en ASCII. Òbviament, l'ordinador ha de fer la conversió entre els dos codis, buscant la correspondència entre els dos codis amb la taula de conversió.

1.6. Representació de nombres en punt flotant

El punt flotant és una notació molt similar a la científica. Consisteix a representar els nombres mitjançant un exponent i una mantissa. L'exponent i la mantissa depenen de la longitud de les paraules utilitzades per l'ordinador o, a vegades, de l'aplicació en concret. L'exponent sol tenir entre cinc i vint bits i la mantissa, de vuit a cent. Habitualment, la mantissa és un nombre fraccionari comprès entre:

$$1/2 \leq |M| < 1$$

Per tant, el primer bit de la mantissa sempre és 1, quan és així es diu que està normalitzada, sino cal normalitzar el nombre, desplaçant a l'esquerra la mantissa fins que el primer 1 estigui al lloc més significatiu (en cas de que $M < 0.5$) i restar a l'exponent un nombre igual als llocs que s'ha desplaçat la mantissa. Per exemple, si volem representar el nombre $N = -0.00100110$ en punt flotant, veiem que N no està normalitzat: hem de desplaçar la mantissa dos llocs a l'esquerra (multiplicar per 4) per tal que sigui ≤ 0.5.

$$N = 2^2 \cdot (-.10011000)$$

Car	ASCII	EBCDIC	Car	ASCII	EBCDIC	Car	ASCII	EBCDIC	Car	ASCII	EBCDIC
NUL	0	0		32	64	@	64			96	
SOH	1	1	!	33	90	A	65	193	a	97	129
STX	2	2	"	34	127	B	66	194	b	98	130
ETX	3	3	#	35	123	C	67	195	c	99	131
EOT	4	5	$	36	91	D	68	196	d	100	132
ENQ	5	45	%	37	108	E	69	197	e	101	133
ACK	6	46	&	38	80	F	70	198	f	102	134
BEL	7	47	'	39	125	G	71	199	g	103	135
BS	8	22	(	40	77	H	72	200	h	104	136
HT	9	05	)	41	93	I	73	201	i	105	137
LF	10	37	*	42	92	J	74	209	j	106	145
VT	11	11	+	43	78	K	75	210	k	107	146
FF	12	12	,	44	107	L	76	211	l	108	147
CR	13	13	-	45	96	M	77	212	m	109	148
SO	14	14	.	46	75	N	78	213	n	110	149
SI	15	15	/	47	97	O	79	214	o	111	150
DLE	16	16	0	48	240	P	80	215	p	112	151
DC1	17	17	1	49	241	Q	81	216	q	113	152
DC2	18	18	2	50	242	R	82	217	r	114	153
DC3	19	19	3	51	243	S	83	226	s	115	162
DC4	20		4	52	244	T	84	227	t	116	163
NAK	21	61	5	53	245	U	85	228	u	117	164
SYN	22	50	6	54	246	V	86	229	v	118	165
ETB	23		7	55	247	W	87	230	w	119	166
CAN	24	24	8	56	248	X	88	231	x	120	167
EM	25	25	9	57	249	Y	89	232	y	121	168
SUB	26	63	:	58		Z	90	233	z	122	169
ESC	27	39	;	59	94	[	91		{	123	
FS	28	28	<	60	76	\	92		|	124	79
GS	29	29	=	61	126	]	93		}	125	
RS	30	30	<	62	110	^	94		~	126	
US	31	31	?	63	111	_	95	109	DEL	127	07

Taula 1.3

on $|M|$ compleix la condició anterior. Tant la mantissa com l'exponent poden ser representats en signe-mòdul o bé en complement a 2. Si reservem vuit bits per a l'exponent i representem la mantissa en complement a 2, tenim que:

$$N_{fp} = 00000010 \quad 10110100$$
$$\text{Exp.} \quad \text{Mant.}$$

A la mantissa s'ha afegit un nové bit, que representa el signe 1 = negatiu). Normalment, però, els nombres en punt flotant es solen representar en signe-mòdul

El valor de l'exponent pot considerar-se com el nombre de desplaçaments que cal fer per normalitzar la mantissa (dins del marge $1/2 \leq |M| < 1$). Si $|M| > 1$ s'ha de desplaçar a la dreta i l'exponent cap a l'esquerra

el mateix nombre de posicions, per exemple: N = 101.101 pot escriure's com N = 2^3 (.101101). En canvi, si $|M| \leq ½$, la mantissa es desplaça cap a l'esquerra, de manera que el bit més significatiu és 1 i ara l'exponent cap a la dreta.

Les normes de la IEEE defineixen tres tipus de representació de nombres reals en punt flotant, segons la precisió: real curt (32 bits), real llarg (64 bits) i real temporal (80 bits).

1.6.1. Real curt

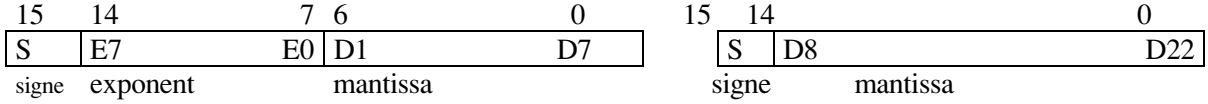
És una representació del tipus signe - mòdul. Els nombres es representen amb 32 bits (4 bytes, és a dir, 2 paraules de 16 bits), dividits en 1 bit per al signe, 8 bits per a l'exponent i 23 per a la mantissa:

Hem dit que el bit D1 de la mantissa ha de ser 1. No obstant això, cal tenir present que en aquest format existeix un bit "fantasma", D0 = 1, que no apareix explícitament en la mantissa; és a dir, en realitat el nombre es representa en la forma 1.xxxxx. Aleshores la conversió és:

$$N = (-1)^S \, 2^{EXP-127} \left(1 + \sum_{1}^{23} \frac{D_i}{2^i} \right)$$

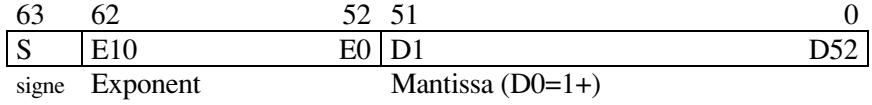
A l'exponent se li resta 127 perquè així es poden representar nombres des de 10^{-38} fins a 10^{+38}.

En alguns sistemes digitals, el real curt s'ha de representar amb dues paraules de 16 bits. En aquest cas es repeteix el bit de signe:



1.6.2. Real llarg

Els nombres es representen amb 64 bits (8 bytes), dividits en 1 bit per al signe, 11 bits per a l'exponent i 52 per a la mantissa:

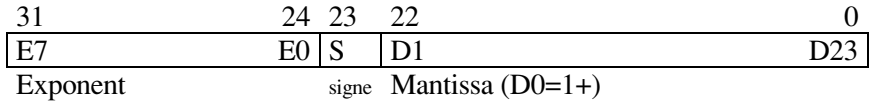


També existeix el bit "fantasma" D0, que val 1. Aleshores la conversió és:

$$N = (-1)^S \, 2^{EXP-1023} \left(1 + \sum_{1}^{52} \frac{D_i}{2^i} \right)$$

A l'exponent se li resta 1023 perquè així es poden representar nombres des de 10^{-308} fins a 10^{+308}.

Moltes empreses (IBM, Borroughs, Cray...) tenen formats propis per a la representació de nombres en punt flotant. Microsoft utilitza un format una mica diferent, que permuta el signe i l'exponent:



D0 hi és implícit. Quan s'utilitza doble precisió, la mantissa s'allarga fins a 52, però l'exponent queda igual. Per això el desbordament (o*verflow*) continua sent 10 ± 38.

1.6.3. Aritmètica amb punt flotant

Les operacions de suma i resta de dos nombres representats amb punt flotant és directa si ambdós tenen el mateix exponent. Si és diferent, primer és necessari un alineament. L'alineament es fa desplaçant cap a la dreta la mantissa del nombre amb menor exponent. El desplaçament ve determinat per la diferència entre els dos exponents. Per exemple, sumarem X + Y, on $X = 2^6$ (.101) i $Y = 2^{11}$ (.101101). Hem de desplaçar la mantissa d'X cinc llocs a la dreta i posar l'exponent a 11. Aleshores:

$X = 2^{11}$ (.00000101)
$Y = 2^{11}$ (.101101)

$Z = 2^{11}$ (.10111001)

En el producte (divisió) es multipliquen (divideixen) les mantisses i se sumen (resten) els exponents. Per exemple, si $X = 2^{Ex} \cdot M_x$; $Y = 2^{Ey}M_y$, tenim que:

$$X \cdot Y = (2^{Ex+Ey})(M_x \cdot M_y) = 2^{Ev}M_v \quad \text{i} \quad X/Y = (2^{Ex-Ey})(M_x/M_y) = 2^{Ew}M_w$$

El resultat ha de normalitzar-se si la mantissa no està dins del rang fixat.