# Electrònica digital no programable

Santiago Cerezo Salcedo, Xavier Mesas Laserna, Ángel L. Miguel Rodríguez, Rafael Rodríguez Coronel

**Adaptació de continguts**: Santiago Cerezo Salcedo

**Electrònica**

# Índex

## Introducció

Des de fa unes poques dècades, l'electrònica digital està revolucionant tot el que ens envolta. Ha canviat la manera d'escoltar música (passant dels discos de vinil als CD-ROM i a la música en format MP3), la manera de veure la televisió (passant de la televisió analògica a la TDT), la manera de comunicar-nos (amb la telefonia mòbil, el correu electrònic i els xats), la manera d'informar-nos (amb Internet i els satèl·lits)... Bé podríem omplir unes quantes pàgines enumerant tots els canvis que l'electrònica digital està produint a les nostres vides.

L'àrea dels automatismes també ha estat molt influenciada per l'electrò- nica digital des que, al final de la dècada dels anys seixanta, la indústria va trobar en les noves tecnologies digitals una solució més eficient que els sistemes de control elèctrics basats en relés i interruptors. Així van apa- rèixer els primers controladors lògics programables (PLC), dispositius ba- sats en l'electrònica digital molt utilitzats al món de l'automatització.

Per aquest motiu, en un mòdul introductori com aquest és necessari in- cloure una unitat sobre lògica digital.

En aquesta unitat treballarem amb la lògica digital i els components digitals més bàsics. Entendre com funcionen aquests components és imprescindible per poder treballar després amb controladors lògics programables.

En l'apartat "Principis bàsics de lògica digital" veurem els fonaments matemàtics de la lògica digital. Veurem quin sistema de numeració fan servir les màquines digitals, com realitzen les operacions matemàti- ques i com fan càlculs lògics elementals.

En l'apartat "Circuits combinacionals i seqüencials" farem un repàs dels elements bàsics dels sistemes digitals: comptadors, petites calcu- ladores, circuits comparadors, etc. Tots aquests elements integrats for- men sistemes digitals més complexos com els PLC.

Per treballar els continguts d'aquesta unitat, és convenient que feu totes les activitats i els exercicis d'autoavaluació. Sobretot és molt important que practiqueu amb el sistema de numeració binari, fent conversions a altres siste- mes i realitzant senzills càlculs.

## Resultats d'aprenentatge

En finalitzar aquesta unitat, l'alumne/a:

1. Reconeix circuits lògics combinacionals determinant les seves característiques i aplicacions.
    - Utilitza diferents sistemes de numeració i codis.
    - Descriu les funcions lògiques fonamentals utilitzades en els circuits electrònics digitals.
    - Representa els circuits lògics mitjançant la simbologia adequada.
    - Interpreta les funcions combinacionals bàsiques.
    - Identifica els components i blocs funcionals.
    - Munta o simula circuits.
    - Verifica el funcionament dels circuits
    - Identifica les diferents famílies d'integrats i la seva aplicació.
    - Realitza les tasques que cal fer individualment amb autosuficiència i seguretat.
2. Reconeix circuits lògics seqüencials determinant les seves característiques i aplicacions.
    - Descriu diferències entre circuits combinacionals i seqüencials.
    - Descriu diferències entre sistemes síncrons i asíncrons.
    - Identifica els components i blocs funcionals.
    - Utilitza els instruments lògics de mesura adequats.
    - Munta o simula circuits.
    - Verifica el funcionament de circuits bàsics seqüencials.
    - Descriu aplicacions reals dels circuits amb dispositius lògics seqüencials.
    - Realitza les tasques que cal fer individualment amb autosuficiència i seguretat.

# 1. Principis bàsics de lògica digital

La lògica digital és la "manera de pensar i fer càlculs" que tenen totes les màquines digitals, que poden ser tant una senzilla calculadora com un autòmat programable o un potent ordinador. Per tant, per entendre com funcionen aquestes màquines digitals és necessari saber operar amb lògica digital.

A diferència dels sistemes analògics, els sistemes digitals només poden prendre un conjunt de valors discrets i canvien de valor per salts.

Els equips digitals utilitzen el sistema de numeració binari, el qual opera únicament amb dos dígits, el 0 i l'1. Com que els éssers humans pensem amb els sistema de numeració decimal, s'han desenvolupat tècniques matemàtiques per fer conversions entre sistemes de numeració: de binari a decimal, de decimal a binari, de binari a hexadecimal, d'hexadecimal a binari, de decimal a hexadecimal i d'hexadecimal a decimal.

Els sistemes digitals fan operacions matemàtiques amb dades expressades en binari. Com a instrument matemàtic per treballar amb sistemes digitals utilitzem l'àlgebra de Boole.

## 1.1 Senyals analògics i senyals digitals

Al camp de l'electricitat i l'electrònica, hi ha dues maneres de representar informació, que anomenem **analògica i digital**, i que es distingeixen per la naturalesa dels valors que poden prendre les variables que ens donen una mesura d'aquesta informació.

Un **senyal analògic** és aquell que varia de forma contínua (sense salts) i pot prendre infinits valors al llarg del temps.

Termòmetre analògic de mercuri

**Termòmetre de mercuri**

El clàssic termòmetre de mercuri està format per un tub de vidre totalment tancat dins del qual hi ha un líquid, en la majoria dels casos mercuri o alcohol.

El principi de funcionament que permet a aquest instrument mesurar la temperatura és que el volum del líquid que conté canvia de manera uniforme segons la temperatura, de manera que quan el líquid s'escalfa s'expandeix i quan es refreda es contrau. Aquest canvi de volum és visible a través del vidre i es manifesta fent més llarga o més curta la columna de líquid que hi ha dins del tub.

En aquest cas, la informació és la temperatura i la variable que conté la informació és la columna del líquid. Com ja sabem, aquest instrument no és elèctric ni electrònic, però la naturalesa d'aquest senyal és clarament analògica, ja que la longitud de la columna de

líquid canvia de manera contínua (sense salts) i pot prendre infinits valors (dins dels límits de l'escala del termòmetre).

Un **senyal digital** és aquell que només pot prendre un conjunt de valors discrets i canvia de valor per salts.

Termòmetre digital

**Termòmetre digital**

Els termòmetres més moderns no es basen en el mètode clàssic del termòmetre de mercuri. Aquests instruments disposen d'una petita pantalla en la qual es visualitza un nombre que representa la temperatura mesurada.

En aquest cas, la informació també és la temperatura i la variable que conté la informació és el nombre que es veu a la pantalla.

Aquesta variable és digital, ja que el nombre mostrat a la pantalla no varia d'una manera contínua amb la temperatura, sinó que normalment canvia en salts discontinus o discrets de 0,1 graus.

## 1.2 Sistemes de numeració en els sistemes digitals

A la nostra vida quotidiana, els éssers humans treballem amb deu xifres (0, 1, 2, 3, 4, 5, 6, 7, 8 i 9) que combinem per representar qualsevol número i per fer càlculs: és el que s'anomena **sistema de numeració decimal**.

Els equips digitals (ordinadors, autòmats programables, etc.) treballen amb un mètode per comptar i fer càlculs molt particular: tan sols treballen amb les xifres 0 i 1. És el que s'anomena **sistema de numeració binari**.

Com que els nombres binaris poden ser molt llargs, sovint s'utilitzen altres sistemes de numeració per representar-los: són els sistemes de numeració **hexadecimal** i **BCD**

Per entendre com funcionen els equips digitals és necessari que sapigueu operar amb nombres binaris, hexadecimals i BCD, i que sapigueu convertir un nombre d'un sistema a un altre.

### 1.2.1 Sistema de numeració decimal

**Sabíeu...**

... que al llarg de la història han sorgit molts sistemes de numeració diferents? El nostre sistema decimal actual va ser inventat a l'Índia i transmès a Europa pels àrabs.

**Dígit**

Un dígit és cadascun dels signes o els símbols que s'empren en un determinat sistema de numeració per formar els diferents nombres.

L'ésser humà fa servir la numeració decimal pel costum dels homes primitius de comptar amb els dits de la mà: d'aquesta manera era més fàcil fer la correspondència entre cada objecte i un dit de la mà.

Amb el temps, aquests homes primitius van haver de representar quantitats per escrit i va sorgir la necessitat d'assignar un símbol o dígit a cada nombre, fet que va originar els deu símbols (dígits) que es fan servir en el sistema decimal:

$$0,\ 1,\ 2,\ 3,\ 4,\ 5,\ 6,\ 7,\ 8 \text{ i } 9$$

Però, és clar, normalment volem representar més de deu nombres. Cap problema: sabem que el sistema decimal permet representar qualsevol nombre sencer afegint-hi noves xifres. Així, amb dues xifres decimals, podem representar 100 nombres (del 0 al 99), amb tres xifres podem representar 1.000 nombres (del 0 al 999) i així consecutivament.

**Descomposició d'un nombre decimal**

El nombre 2.364 consta de 4 xifres que representen el nombre d'unitats (4), desenes (6), centenes (3) i milers (2).

Si fem servir una mica de matemàtiques, podem descompondre aquest nombre tal com s'indica:

$$2.364 = 2 \cdot 1.000 + 3 \cdot 100 + 6 \cdot 10 + 4 \cdot 1$$

Una altra manera d'escriure la suma anterior és fent servir la potenciació:

$$2.364 = 2 \cdot 10^3 + 3 \cdot 10^2 + 6 \cdot 10^1 + 4 \cdot 10^0$$

Escrivint el nombre com una suma de potències, queda molt més clar per què el sistema decimal es diu que és un sistema en base deu: cada dígit multiplica la base (el 10) elevada a un exponent.

Recordeu aquesta manera de descompondre un nombre perquè us serà molt útil quan estudieu els sistemes binari i hexadecimal.

### 1.2.2 Sistema de numeració binari

El sistema de numeració binari també és anomenat sistema de numeració en base dos.

El sistema binari fa servir únicament dos dígits, el 0 i l'1, en lloc dels deu dígits que fa servir el sistema decimal.

Així, amb una xifra només podem representar dos nombres: el 0 i l'1. Per representar més de dos nombres hem d'afegir noves xifres a l'esquerra, tal com havíem fet en el cas decimal.

Cada una d'aquestes xifres és el que es coneix com a **bit** i , per tant, podem dir que un nombre binari es compon d'una sèrie de bits.

- Amb dos bits podem representar quatre nombres: 00, 01, 10 i 11.
- Amb tres bits podem representar vuit nombres: 000, 001, 010, 011, 100, 101, 110 i 111.
- Amb quatre bits podem representar setze nombres: 0000, 0001, 0010, 0011, 0100, 0101, 0110, 0111, 1000, 1001, 1010, 1011, 1100, 1101, 1110 i 1111.
- I així consecutivament.

En la taula 1.1 podeu veure l'equivalència entre els primers nombres dels sistemes decimal i binari.

**Taula 1.1.** Equivalència entre els setze primers nombres decimals i binaris

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Decimal** | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| **Binari** | 0 | 1 | 10 | 11 | 100 | 101 | 110 | 111 |
| **Decimal** | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| **Binari** | 1000 | 1001 | 1010 | 1011 | 1100 | 1101 | 1110 | 1111 |

Hi ha una relació entre el nombre de bits que té un codi binari i la quantitat de nombres que es poden representar. Aquesta important relació és la següent:

$$N = 2^n$$

On $N$ és la quantitat de nombres que es poden representar amb ***n*** bits.

L'agrupació de vuit bits és tan utilitzada en electrònica i, sobretot, en informàtica, que rep un nom especial: *byte*.

**Quants nombres podem representar amb vuit bits?**

La resposta és $2^8$ o, el que és el mateix, 256.

Com que el primer nombre és el 0 (00000000), l'últim que es podrà representar és el 255 (11111111).

Treballar amb nombres binaris de molts dígits no és gaire còmode i és fàcil que ens equivoquem, per exemple, quan hem de copiar un nombre d'un lloc a un altre en un exercici.

**De quatre bits en quatre bits**

Per què s'acostumen a agrupar els nombres grans en quatre bits? Com veureu més endavant, agrupar d'aquesta manera un nombre binari ens facilitarà la tasca de convertir-lo a hexadecimal en el cas que sigui necessari.

Una manera d'intentar fer més llegible els nombres binaris consisteix a separar els nombres grans en grups de quatre bits començant per la dreta.

Per exemple, un nombre tan poc llegible com

100101010101110101

es podria escriure de la manera següent:

10 0101 0101 0111 0101

En qualsevol cas, recordeu que el truc de separar en grups de quatre no és obligatori. Tan sols és una recomanació per facilitar-vos la tasca de treballar amb nombres binaris.

Fixeu-vos que, en aquest cas, el grup situat més a l'esquerra no té quatre bits sinó dos. Si voleu, podeu representar aquest grup amb quatre bits afegint a l'esquerra els 0 que calguin:

0010 0101 0101 0111 0101

**Indicació de nombre binari**

De vegades, s'afegeix el subíndex 2 per indicar que un nombre està expressat en binari. Per exemple, el nombre binari 101 s'escriu $101_2$. També es pot indicar que un nombre és binari amb el sufix *b*. Per exemple, el nombre binari 101 s'escriu $101_\text{b}$.

### 1.2.3 Sistema de numeració hexadecimal

El sistema de numeració hexadecimal utilitza setze dígits:

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

El mecanisme que es fa servir per construir nombres és similar al que hem vist en

el sistema decimal i binari:

El sistema de numeració hexadecimal també és anomenat sistema de numeració en base setze.

- Amb una xifra podem representar 16 nombres: del 0 al F. Per representar més de 16 nombres, l'únic que hem de fer és afegir noves xifres hexadecimals a l'esquerra.
- Amb dues xifres podem representar 256 nombres: 00, 01, 02, ..., 0F, 10, 11, 12, ..., 1F, 20, 21, ... 2F, ... ..., F0, F1, F2, ... FF
- Amb tres xifres podem representar 4.096 nombres: 000, 001, ..., 00F, 010, ..., 0F0, 0F1, 0FF, ..., FFA, FFB, ..., FFF
- I així consecutivament.

A la taula 1.2 es representen els primers nombres decimals i la seva correspondència en hexadecimal.

**Indicació de nombre hexadecimal**

De vegades, s'afegeix el prefix *0x* per indicar que un nombre està expressat en hexadecimal. Per exemple, el nombre hexadecimal 13 s'escriu 0x13. També es pot indicar que un nombre és hexadecimal amb el sufix *H*. Per exemple, el nombre hexadecimal 13 s'escriu 13H.

**Taula 1.2.** Equivalència entre els vint primers nombres decimals i hexadecimals

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| **Repr. decimal** | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| **Repr. hexadecimal** | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| **Repr. decimal** | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 |
| **Repr. hexadecimal** | A | B | C | D | E | F | 10 | 11 | 12 | 13 |

## 1.3 Conversió entre sistemes de numeració

Els sistemes digitals treballen amb el sistema de numeració binari, però els éssers humans pensem amb el sistema de numeració decimal. Per tant, es fa necessari saber convertir nombres binaris en decimals i a l'inrevés.

El sistema de numeració hexadecimal s'utilitza per representar nombres binaris d'una manera més compacta. També es necessari saber convertir nombres hexadecimals al sistema binari i decimal i a l'inrevés.

Per fer totes aquestes conversions s'utilitzen unes tècniques matemàtiques molt senzilles.

### 1.3.1 Conversió de binari a decimal

Per convertir un nombre binari a decimal s'ha de fer una suma de termes en els quals cada dígit binari multiplica el nombre 2 elevat a un exponent.

Per exemple, l'equivalent decimal del nombre binari $110101_2$ es pot calcular de la manera següent:

$$1 \cdot 2^5 + 1 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0$$

Podem eliminar els termes que multipliquen per zero:

$$1 \cdot 2^5 + 1 \cdot 2^4 + 1 \cdot 2^2 + 1 \cdot 2^0$$

Ara només cal substituir cada potència pel seu valor decimal ($2^5$ = 32, $2^4$ = 16, $2^2$ = 4 i $2^0$ = 1) i fer la suma de tots els termes:

$$32 + 16 + 4 + 1 = 53$$

Ja hem acabat! El nombre binari $\mathbf{110101_2}$ és el nombre decimal **53**.

Aquest problema es pot resoldre gràficament, mitjançant una taula com la mostrada a la taula 1.3. A la part superior de la taula apareix l'equivalent decimal de les diferents posicions dels dígits binaris (d'esquerra a dreta, des de $2^7$ fins a $2^0$).

**Taula 1.3.** Plantilla per convertir de binari a decimal.

| 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 |
|---|---|---|---|---|---|---|---|
| | | | | | | | |

A la taula, hi escriurem **de dreta a esquerra** el nombre binari i després sumar els valors decimals de les caselles que estiguin a 1.

Per exemple, el número binari 10001100 es faria com es mostra a la taula 1.4.

**Taula 1.4.** Conversió del 10001100 binari a decimal.

| 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |

Sumem els termes a 1, obtenint:

$$128 + 8 + 4 = 140$$

### 1.3.2 Conversió de decimal a binari

Un mètode gràfic per convertir un nombre binari a decimal utilitza una taula com la mostrada a la taula 1.5. A la part superior de la taula apareix l'equivalent decimal

de les diferents posicions dels dígits binaris (d'esquerra a dreta, des de $2^7$ fins a $2^0$).

**Taula 1.5.** Plantilla per convertir de decimal a binari.

| 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 |
|---|---|---|---|---|---|---|---|
| | | | | | | | |

Per convertir a decimal, únicament calia posar el nombre binari sobre la plantilla, escriure'l de dreta a esquerra i sumar els valors que siguin a 1.

La qüestió que ens ocupa ara és la contrària: a partir del nombre decimal volem obtenir l'equivalent binari. Per fer això, anirem omplint la fila inferior de la plantilla **d'esquerra a dreta** amb uns o zeros, comptant fins a arribar al nombre decimal desitjat.

Per exemple, volem trobar l'equivalent binari del nombre decimal 89. Utilitzant la plantilla obtindrem el mostrat a la taula 1.6.

**Taula 1.6.** Plantilla per convertir de decimal a binari.

| 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 |

El raonament que hem seguit per omplir la plantilla és el següent:

1. Comencem per la cel·la de més a l'esquerra (la de pes 128). Posem la cel·la a 0 perquè si la poséssim a 1, el nombre decimal obtingut seria, com a mínim, el 128 i ens passaríem de 89.
2. Continuem amb la següent cel·la a la dreta (la de pes 64). Aquesta la posem a 1 perquè amb 64 encara no arribem a 89.
3. Continuem amb la següent cel·la a la dreta (la de 32). Aquesta la posem a 0 perquè si la poséssim a 1, tindríem 64 + 32 = 96 i ens passaríem de 89.
4. Continuem amb la següent cel·la a la dreta (la de pes 16). Aquesta la posem a 1 perquè així tenim 64 + 16 = 80 i encara no arribem a 89.
5. Continuem amb la següent cel·la a la dreta (la de pes 8). Aquesta la posem a 1 perquè així tenim 64 + 16 + 8 = 88 i encara no arribem a 89.
6. Continuem amb la següent cel·la a la dreta (la de pes 4). Aquesta la posem a 0 perquè si la poséssim a 1, tindríem 64 + 16 + 8 + 4 = 92 i ens passaríem de 89.
7. Continuem amb la següent cel·la a la dreta (la de pes 2). Aquesta la posem a 0 perquè si la poséssim a 1, tindríem 64 + 16 + 8 + 2 = 90 i ens passaríem de 89.
8. Continuem amb la següent cel·la a la dreta (la de pes 1). Aquesta la posem a 1 perquè així tenim 64 + 16 + 8 + 1 = 89 i obtenim el valor decimal desitjat.

Per tant, el nombre binari equivalent al decimal **89** és el $\mathbf{0101\ 1001_2}$.

### 1.3.3 Conversió de binari a hexadecimal

Passar un nombre binari al seu equivalent hexadecimal és molt fàcil, perquè cada dígit hexadecimal es codifica directament amb quatre dígits binaris.

> Amb un nombre binari qualsevol, només cal fer grups de quatre bits començant per la dreta del nombre binari.
> Cadascun d'aquests grups es correspondrà amb un símbol del sistema hexadecimal, de manera que la seqüència ordenada d'aquests símbols és l'equivalent hexadecimal del nombre binari.

Fixeu-vos que aquest mètode és aplicable a qualsevol nombre binari, independentment del nombre de bits que tingui.

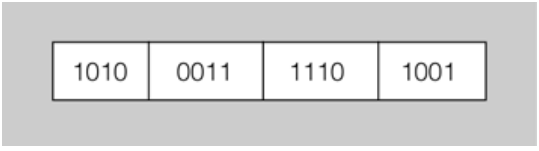
Per a la vostra referència, en la taula 1.7 teniu les equivalències entre els 16 primers nombres dels sistemes de numeració decimal, binari i hexadecimal.

**Taula 1.7.** Equivalència entre els nombres decimal, binari i hexadecimal

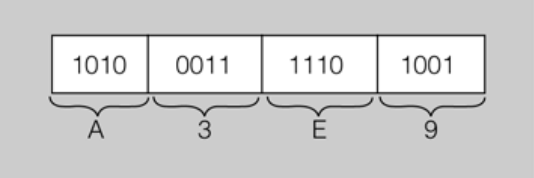
| Decimal | Binari | Hexadecimal |
|---|---|---|
| 0 | 0000 | 0 |
| 1 | 0001 | 1 |
| 2 | 0010 | 2 |
| 3 | 0011 | 3 |
| 4 | 0100 | 4 |
| 5 | 0101 | 5 |
| 6 | 0110 | 6 |
| 7 | 0111 | 7 |
| 8 | 1000 | 8 |
| 9 | 1001 | 9 |
| 10 | 1010 | A |
| 11 | 1011 | B |
| 12 | 1100 | C |
| 13 | 1101 | D |
| 14 | 1110 | E |
| 15 | 1111 | F |

**Quin nombre hexadecimal és el nombre binari 1010 001 1110 1001?**

En primer lloc hem de fer grups de quatre bits començant per la dreta, tal com es mostra en la figura següent:




Seguidament, "traduïm" cada grup de quatre bits al seu equivalent hexadecimal (figura següent), segons la correspondència que es mostra a la taula 1.7.



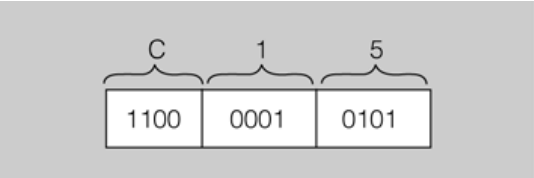

Per tant, l'equivalent hexadecimal del nombre binari 1010 0011 1110 10012 és el 0xA3E9

### 1.3.4 Conversió d'hexadecimal a binari

En un nombre hexadecimal qualsevol, cada dígit es correspondrà amb un grup de quatre bits, de manera que la seqüència ordenada d'aquests bits és l'equivalent binari del nombre hexadecimal

**Quin nombre binari és el nombre hexadecimal 0xC15?**

Únicament cal identificar el grup de quatre bits associat a cada dígit hexadecimal, com es mostra a la figura següent. Per tant, el nombre hexadecimal **0xC15** és el nombre binari **1100 0001 0101**.



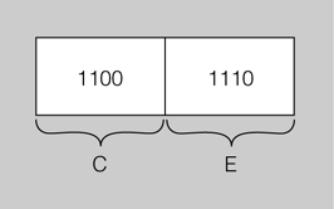

### 1.3.5 Conversió de decimal a hexadecimal

El mètode per convertir un nombre decimal a hexadecimal consisteix a fer dues senzilles accions: en primer lloc es converteix el nombre decimal a binari i, seguidament, es converteix el nombre binari resultant a hexadecimal.

**Quin nombre hexadecimal és el 206 decimal?**

Primer convertirem el nombre decimal 206 a nombre binari, mitjançant el mètode gràfic, obtenint el número binari **11001110**.

A continuació convertim aquest nombre binari a hexadecimal, fent grups de quatre bits i convertint-los al símbol hexadecimal corresponent (figura següent):




Per tant, l'equivalent hexadecimal del nombre decimal 206 és el 0xCE.

### 1.3.6 Conversió d'hexadecimal a decimal

Recordeu que el sistema hexadecimal també s'anomena *sistema de base setze* i, per tant la posició de cada dígit anirà referida a una potència de 16.

Per convertir un nombre hexadecimal a decimal s'ha de fer una suma de termes en els quals cada dígit hexadecimal multiplica el nombre 16 elevat a un exponent.

Recordeu que el prefix 0x indica que el nombre està expressat en hexadecimal.

Per exemple, l'equivalent decimal del nombre 0x1B2 es calcula així:

$$1 \cdot 16^2 + \mathrm{B} \cdot 16^1 + 2 \cdot 16^0$$

Fet això, només ens cal un pas més per passar a decimal: substituir els símbols A, B, C, D, E i F pel seu equivalent decimal (en la taula 1.7 podeu consultar aquestes equivalències) i operar fins a obtenir un nombre decimal:

$$1 \cdot 16^2 + \mathrm{B} \cdot 16^1 + 2 \cdot 16^0 = 1 \cdot 16^2 + 11 \cdot 16^1 + 2 \cdot 16^0$$

El dígit **B** correspon al nombre decimal **11**. Ara només cal substituir cada potència pel seu valor decimal ($16^2$ = 256, $16^1$ = 16 i $16^0$ = 1) i fer la suma de tots els termes:

$$1 \cdot 256 + 11 \cdot 16 + 2 \cdot 1 = 256 + 176 + 2 = 434$$

Ja hem acabat! El nombre hexadecimal **0x1B2** és el nombre **434** en decimal

## 1.4 Altres formes de codificar nombres sencers

Hi ha diferents sistemes per codificar els valors numèrics. Els més utilitzats són els sistemes de numeració decimal, binari i hexadecimal.

Ara bé, els sistemes digitals fan servir altres codificacions, per exemple, per assignar un codi que representi una lletra de l'abecedari o bé que indiqui la posició de l'eix d'un motor.

En casos com aquests es fan servir uns codis dissenyats especialment per satisfer aquestes i altres necessitats. Un d'aquests codis és el BCD, el qual s'utilitza molt en el món dels autòmats programables.

### 1.4.1 Codificació BCD

La sigla BCD prové de l'anglès: ***B**inary **C**oded **D**ecimal*, és a dir, decimal codificat en binari.

Per passar un nombre decimal al seu equivalent BCD només cal passar cada xifra del nombre decimal al seu equivalent en binari en quatre bits, independentment.

Cada xifra del número decimal generarà quatre dígits binaris.

Per exemple, el nombre decimal **9285** seria el nombre **1001 0010 1000 0101** en codi BCD, ja que:

- El 9 és el nombre 1001 en binari.
- El 2 és el nombre 0010 en binari.
- El 8 és el nombre 1000 en binari.
- El 5 és el nombre 0101 en binari.

Fixeu-vos que per representar cada símbol d'un nombre decimal **sempre** s'utilitzen quatre bits en codi BCD.

## 1.5 Aritmètica binària

Les operacions es realitzen de la mateixa forma que en sistema decimal, però, a causa de la senzillesa del sistema binari, es poden fer algunes simplificacions.

Els sistemes digitals (ordinadors, calculadores, autòmats programables...) són capaços de fer operacions aritmètiques amb dades expressades en sistema binari.

### 1.5.1 Suma en sistema binari

Les possibles combinacions de la suma binària són:

- 0 + 0 = 0
- 0 + 1 = 1
- 1 + 0 = 1
- 1 + 1 = 0, i en porto una

Les tres primeres combinacions són evidents. Però la suma d'**1+1**, que en decimal sabem que és 2, s'ha d'escriure en binari amb dos dígits (10). Per tant, la suma d'1+1 és 0 i s'arrossega una unitat que se suma a la següent posició a l'esquerra.

**Quin és el resultat de sumar els nombres binaris 1011 i 0110?**

Fent la suma segons les regles de la suma binària obtenim:

$$\begin{array}{cccccc} & & 1 & 1 & & \\ & & 1 & 0 & 1 & 1 \\ + & & 0 & 1 & 1 & 0 \\ \hline & 1 & 0 & 0 & 0 & 1 \end{array}$$

La mateixa operació però en sistema decimal hauria estat: 11 + 6 = 17.

Podeu observar que el resultat té cinc dígits perquè arrosseguem un 1.

### 1.5.2 Complement a un i complement a dos d'un nombre binari

El **complement a un** i el **complement a dos** són dues eines matemàtiques que faciliten molt les tasques aritmètiques en el sistema binari, sobretot la realització de restes i el treball amb nombres negatius.

#### Complement a un d'un nombre binari

El complement a un (C1) d'un nombre binari és el nombre resultant d'invertir els uns i els zeros d'aquest nombre.

Per exemple, el complement a un del nombre 1101 és el nombre 0010.

#### Complement a dos d'un nombre binari

El complement a dos (C2) d'un nombre binari és el nombre resultant de sumar 1 al seu complement a un. És a dir, C2 = C1 + 1.

Per exemple, per calcular el complement a dos del nombre binari 1001 primer calculem el seu complement a un invertint uns i zeros. Ens dóna el nombre 0110. A aquest complement a un li sumem una unitat i ja tenim el complement a dos: 0110 + 1 = 0111.

Generalment s'assumeix que el C2 és la manera de representar el negatiu d'un número binari.

### 1.5.3 Resta en sistema binari

**Recordeu...**

... que en una resta de dos nombres *A* - *B*, el nombre *A* s'anomena *minuend* i el nombre *B* s'anomena *subtrahend*.

La resta de dos nombres binaris pot obtenir-se sumant al minuend el complement a dos del subtrahend.

Per exemple, fem la resta 1011011 - 0101110:

- Primer calculem el complement a un del subtrahend:**1010001**.
- A continuació calculem el complement a dos d'aquest: **1010010**.
- Finalment sumem el minuend amb el complement a dos del subtrahend.

$$\begin{array}{cccccccc} & & 1 & & & 1 & & \\ & 1 & 0 & 1 & 1 & 0 & 1 & 1 \\ + & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ \hline \underline{1} & 0 & 1 & 0 & 1 & 1 & 0 & 1 \end{array}$$

Podeu comprovar que si "traduïm" la resta binària 1011011 - 0101110 = 0101101 a decimal ens dóna: 91 46 = 45.

En el resultat de la suma ens sobra un bit, ja que arrosseguem un 1 per l'esquerra. Però com que el nombre resultant de la resta no pot ser més gran que el minuend, el bit sobrant s'ignora. Per tant: **1011011 - 0101110 = 0101101**.

### 1.5.4 Representació de nombres negatius en sistema binari

Amb el sistema decimal, els nombres negatius es representen senzillament precedint-los amb el signe **-**. En canvi, en el sistema binari no s'utilitza el signe sinó que es fan servir altres mètodes. El mètode més utilitzat és el del complement a 2 (C2).

**Altres representacions de nombres negatius binaris**

Hi ha altres mètodes per representar nombres negatius en sistema binari (el mètode de signe i magnitud, el mètode de complement a un, el mètode d'excés N...), però no són tan utilitzats com el mètode de complement a dos.

> Els nombres positius es representen amb el sistema de numeració binari normal, i els nombres negatius es representen amb el sistema de numeració binari en **complement a dos**.

En la taula 1.8 es mostra la representació binària des del -128 fins al +127. Si us hi fixeu, el bit de més pes sempre és zero per als nombres positius i sempre és u per als nombres negatius. Aquest bit de més pes s'anomena **bit de signe**, perquè ens indica si el nombre és positiu (bit de signe a zero) o negatiu (bit de signe a un). La resta de bits s'anomenen **magnitud**.

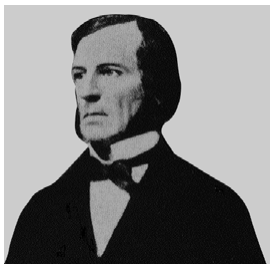
**Taula 1.8.** Representació binària de nombres positius i negatius amb el mètode de complement a dos

| Decimal | Signe | Magnitud | Codi |
|---|---|---|---|
| +127 | 0 | 111 1111 | Natural |
| +126 | 0 | 111 1110 | Natural |
| +125 | 0 | 111 1101 | Natural |
| +124 | 0 | 111 1100 | Natural |
| ... | 0 | ... | Natural |
| +5 | 0 | 000 0101 | Natural |

**Taula 1.8** (continuació)

| Decimal | Signe | Magnitud | Codi |
|---|---|---|---|
| +4 | 0 | 000 0100 | Natural |
| +3 | 0 | 000 0011 | Natural |
| +2 | 0 | 000 0010 | Natural |
| +1 | 0 | 000 0001 | Natural |
| 0 | 0 | 000 0000 | Natural |
| -1 | 1 | 111 1111 | C2 |
| -2 | 1 | 111 1110 | C2 |
| -3 | 1 | 111 1101 | C2 |
| -4 | 1 | 111 1100 | C2 |
| -5 | 1 | 111 1011 | C2 |
| ... | 1 | ... | C2 |
| -125 | 1 | 000 0011 | C2 |
| -126 | 1 | 000 0010 | C2 |
| -127 | 1 | 000 0001 | C2 |
| -128 | 1 | 000 0000 | C2 |

## 1.6 Àlgebra de Boole



**George Boole (1815-1864)**

Matemàtic i filòsof britànic. Va desenvolupar l'anomenada *àlgebra de Boole* com una eina per analitzar la ment humana. L'aplicació als sistemes digitals va ser posterior.

L'àlgebra de Boole es basa en el següent:

- Un conjunt d'elements que únicament poden tenir dos valors possibles i que anomenarem 0 i 1.
- Tres operacions que anomenarem **suma lògica**, **producte lògic** i **negació**.

Un element qualsevol de l'àlgebra de Boole es representa amb una lletra que s'anomena **variable booleana** o, alternativament, **variable binària**. Així, doncs, cada variable binària té associat un valor 1 o un valor 0.

L'àlgebra de Boole és una eina matemàtica que ens permet treballar amb sistemes que només fan servir dos estats clarament diferenciats, com és el cas dels sistemes digitals.

**Què significa 1 i què significa 0?**

Això depèn de l'aplicació i l'únic important és que representa dos estats clarament diferenciats.

Vegem-ne un exemple senzill: molts cotxes incorporen un sistema que informa de l'estat de les portes, de manera que encén un llum indicador si detecta que hi ha una porta oberta.

Com que cadascuna de les portes del vehicle únicament poden estar en dos estats diferents (porta oberta o porta tancada), podem "traduir" aquesta situació a l'àlgebra de Boole d'una manera molt clara.

Per exemple, podríem associar una variable binària a cada porta que anomenarem de la següent manera:

- $A$: variable que representa l'estat de la porta del conductor
- $B$: variable que representa l'estat de la porta del copilot
- $C$: variable que representa l'estat de la porta del darrere esquerra
- $D$: variable que representa l'estat de la porta del darrere dreta

I finalment, associaríem a l'estat "porta oberta" el valor 1 i a l'estat "porta tancada", el valor 0. Òbviament, aquesta elecció ha estat arbitrària i podríem haver fet l'assignació contrària.

Fixeu-vos que amb aquests simples passos hem pogut *traduir* una informació del món real al món matemàtic de l'àlgebra de Boole, de tal manera que si ara ens diguessin:

- $A = 1$
- $B = 1$
- $C = 0$
- $D = 0$

sabríem que les dues portes davanteres del nostre vehicle estan obertes i les dues del darrere, tancades.

**Variables binàries al món real**

Hi ha molts casos del món real que es poden "traduir" al món binari d'uns i zeros. Per exemple, l'estat de les portes d'un vehicle: cada porta del vehicle equival a una variable binària, de manera que si la porta està oberta la variable pren el valor 1 i si la porta està tancada la variable pren el valor 0.

Les tres operacions bàsiques de l'àlgebra de Boole són les següents:

- **Suma lògica**. És representada pel **signe +** i requereix dos operands. Aquesta operació queda definida en la taula 1.9.

**Taula 1.9.** Suma lògica

| a | b | a + b |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

El resultat de la suma lògica és 1 si qualsevol dels operands és 1.

- **Producte lògic**. És representat pel **signe** $\cdot$ i requereix dos operands. Aquesta operació queda definida en la taula 1.10.

**Taula 1.10.** Producte lògic

| a | b | $a \cdot b$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

El resultat del producte lògic és 1 només si tots els operands són 1.

- **Negació**. És una operació que només requereix un operand. Si l'operand és la variable $a$, una línia horitzontal sobre aquesta variable indica que està negada o complementada ($\overline{a}$). Aquesta operació queda definida en la taula 1.11.

**Taula 1.11.** Negació lògica

| a | $\overline{a}$ |
|---|---|
| 0 | 1 |
| 1 | 0 |

L'àlgebra de Boole és el suport matemàtic per al disseny i l'anàlisi dels sistemes digitals i, com tota àlgebra, disposa d'una sèrie de postulats i teoremes. Segurament les més importants són el **teorema d'absorció**, molt útil en simplificacions, i el **teorema de DeMorgan**, el qual ens indica com es converteix un producte lògic en una suma lògica i a l'inrevés.

El **teorema d'absorció** es pot expressar amb aquestes dues expressions algebraiques:

$$a + a \cdot b = a$$
$$a \cdot (a + b) = a$$

El **teorema de DeMorgan** es pot expressar amb aquestes dues expressions algebraiques:

$$\overline{a \cdot b} = \overline{a} + \overline{b}$$
$$\overline{a + b} = \overline{a} \cdot \overline{b}$$

### 1.6.1 Funcions lògiques

**Diferents maneres d'escriure una funció lògica**

En alguns llibres veureu que, al costat del nom de la funció, s'acostuma a escriure entre parèntesis les variables emprades per la funció. Per exemple:

$$f(a, b, c, d) = a \cdot b + c \cdot d$$

Tanmateix, moltes vegades veureu que el signe $\cdot$ del producte lògic no s'escriu, ja que es dóna per suposada la seva existència. Per exemple, la funció anterior es pot escriure així:

$$f(a, b, c, d) = ab + cd$$

Anomenem **funció lògica** una variable binària que pren el valor d'una expressió algebraica formada per altres variables binàries relacionades de la manera que sigui mitjançant les operacions suma lògica, producte lògic i negació.

Per exemple, les següents expressions són funcions lògiques definides per una expressió algebraica:

$$f_1 = a \cdot b + c \cdot d$$

$$f_2 = a \cdot \overline{c} + b$$

Diem que una funció està expressada com una **suma de productes** si la seva expressió algebraica està formada per una suma de termes i cada terme està format per unes variables que es multipliquen entre elles.

Per exemple, la funció següent és una suma de productes:

$$f_1 = a \cdot b \cdot c + c \cdot \overline{d}$$

Diem que una funció està expressada com un **producte de sumes** si la seva expressió algebraica està formada per un producte de termes i cada terme està format per unes variables que se sumen entre elles.

Per exemple, la funció següent és un producte de sumes:

$$f_2 = (a + c) \cdot (b + \overline{d})$$

**Compte amb els parèntesis!**

Pareu atenció que, en el cas dels productes de sumes, és obligatori l'ús de parèntesis en els termes, ja que primer s'han d'avaluar les sumes i després el producte. Si, per error, escrivíssim la funció de l'exemple així:

$$f_2 = a + c \cdot b + \overline{d}$$

entendríem que primer s'hauria de fer el producte $c \cdot b$ i després fer la suma de tots els termes (estaríem al davant d'una funció en forma de suma de productes)

## Com s'avaluen les funcions lògiques?

Donada una expressió algebraica booleana formada per molts termes, hauríem de trobar el valor resultant de la funció seguint aquest ordre:

1. Parèntesis
2. Operació de negació
3. Operació producte lògic
4. Operació suma lògica

**Exemple d'avaluació de funció lògica**

Volem saber quin valor té la funció $f_1 = a \cdot b + c \cdot d$ quan *a* = 0, *b* = 1, *c* = 1 i *d* = 1.

*Solució*

S'hauria d'avaluar seguint els següents passos:

Com que no hi ha parèntesis ni negacions, en primer lloc es farien les operacions de producte lògic:

$$a \cdot b = 0 \cdot 1 = 0 \, c \cdot d = 1 \cdot 1 = 1$$

Per acabar, es faria la suma lògica dels resultats anteriors. En aquest cas, substituïm $a \cdot b$ pel seu valor (0) i $c \cdot d$ pel seu valor (1), i obtindrem:

$$f_1 = 0 + 1 = 1$$

**Exemple d'avaluació de funció lògica**

Volem saber quin valor té la funció $f_2 = \overline{a} \cdot (b + c)$ quan *a* = 1, *b* = 1 i *c* = 1.

En aquest cas, en primer lloc s'ha de fer l'operació que hi ha dins del parèntesi:

$$b + c = 1 + 1 = 1$$

Després s'ha de fer l'operació de negació que té la variable *a*:

$$\overline{a} = \overline{1} = 0$$

I finalment s'ha de fer el producte lògic d'aquests dos resultats:

$$f_2 = 0 \cdot 1 = 0$$

### 1.6.2 Taula de veritat d'una funció lògica

Sense saber-ho, hem vist algunes taules de veritat quan hem estudiat les operacions booleanes bàsiques (taula 1.9, taula 1.10 i taula 1.11).

Una taula de veritat és una manera de representar una funció lògica, en la qual es mostra el valor que pren la funció per a cadascuna de les combinacions de les variables d'entrada de la funció.

Per construir la taula de veritat d'una funció lògica s'han de seguir els següents passos:

1. Dibuixar una taula amb:
   - tantes columnes com variables tingui la funció, **més una columna addicional**
   - tantes files com el nombre de possibles combinacions de les variables d'entrada, **més una fila addicional**
2. En la fila superior es posen els noms de les variables (en les columnes de l'esquerra) i de la funció (en la columna de la dreta).
3. S'omple la resta de files amb les diferents combinacions de valors de les variables d'entrada, en ordre creixent, com si comptéssim en binari natural.
4. S'omple la columna de la dreta amb el valor que pren la funció per a cada combinació de valors d'entrada.

Recordeu el que heu après al subapartat "Sistema de numeració binari": si tenim $n$ variables, podem fer $2^n$ combinacions diferents.

**Exemple: taula de veritat de la funció $f_1=a\cdot \overline{c}+b$**

Per construir la taula de veritat de la funció $f_1 = a \cdot \overline{c} + b$ faríem el següent:

1. Construïm la taula:
   - Amb 3 + 1 (és a dir, 4) columnes, ja que la funció és de tres variables (*a*, *b* i *c*).
   - Amb $2^3$ + 1 (és a dir, 9) files, ja que amb 3 variables es poden fer $2^3$ = 8 combinacions.
2. Posem el nom de les variables (*a*, *b*, *c*) i de la funció ($f_1$) a la fila superior.
3. Omplim a sota totes les combinacions de valors que podem tenir amb tres variables. La primera combinació és 000, la següent 001, la següent 010... fins a arribar a l'última combinació 111 (com si comptéssim en binari).
4. Avaluem la funció per a cadascuna d'aquestes combinacions i posem el resultat a la columna de la dreta.

El resultat obtingut es mostra a la taula 1.12.

### 1.6.3 Funcions equivalents

Una funció booleana es pot definir indistintament amb una expressió algebraica o amb una taula de veritat. Coneixem el mètode per trobar la taula de veritat d'una funció expressada algebraicament.

**Taula 1.12.** Taula de veritat de la funció $f_1 = a \cdot \overline{c} + b$

| *a* | *b* | *c* | $f_1$ |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

El que encara no hem vist és com es fa el pas invers: partir de la taula de veritat d'una funció i trobar una expressió algebraica que descrigui la mateixa funció. Això és el que estudiarem en l'apartat "Formes canòniques d'una funció".

Abans, però, ens podríem plantejar si hi ha més d'una expressió algebraica que verifiqui una taula de veritat donada. I, efectivament, és això.

Expressions algebraiques aparentment diferents poden descriure la mateixa funció lògica. Quan passa això diem que **les funcions són equivalents.**

La manera més fàcil de descobrir si dues funcions són equivalents és trobar la taula de veritat a partir de cadascuna de les expressions algebraiques i comprovar si donen el mateix resultat.

Com a exemple, veurem si són equivalents les funcions següents:

$$f_1 = a + a \cdot b \; f_2 = a \cdot (a + b)$$

Primer omplim la taula de veritat de $f_1$: el resultat el podeu veure a la taula 1.13.

**Taula 1.13.** Taula de veritat de $f_1$

| a | b | $f_1$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

A continuació omplim la taula de veritat de $f_2$: el resultat el podeu veure a la taula 1.14.

**Taula 1.14.** Taula de veritat de $f_2$

| a | b | $f_2$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

Observant les dues taules, veiem que els resultats de les dues funcions són idèntics. Per tant, $f_1$ i $f_2$ **són equivalents**.

### 1.6.4 Formes canòniques d'una funció

A continuació veurem com podem trobar una expressió algebraica d'una funció a partir de la seva taula de veritat. Hi ha dues maneres de fer-ho i es coneixen com a **formes canòniques**.

#### Primera forma canònica

La primera forma canònica es basa a expressar la funció com una **suma de productes** que deduirem de la taula de veritat mitjançant el procediment que utilitzem en l'exemple següent:

1. A la taula de veritat, ens fixarem en les combinacions de valors de les variables d'entrada que fan que la funció valgui 1.
2. Per cadascuna d'aquestes combinacions hem de trobar un terme producte, anomenat **minterm**, amb les següents característiques:
   - Està format per totes les variables de la funció, negades o no.
   - Aquestes variables estan multiplicades.
3. Per saber si la variable ha d'estar negada o no a cada minterm, us heu de fixar en la combinació de valors:
   - Si el valor d'aquesta variable és 0, aquesta variable haurà d'anar negada en el terme producte.
   - Si el valor d'aquesta variable és 1, aquesta variable haurà d'anar directa (no negada) en el terme producte.
4. Finalment, la primera forma canònica s'obté sumant tots els *minterms*.

**Taula 1.15.** Funció per a la qual en volem obtenir la primera forma canònica

| *a* | *b* | *c* | *f* |
|---|---|---|---|
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 0 |

**Exemple: obtenció de la primera forma canònica d'una funció**

A partir de la taula de veritat mostrada en la taula 1.15, trobarem la primera forma canònica de la funció.

Seguint el procediment descrit anteriorment:

1. Primer ens fixem en les combinacions de valors que fan que la funció valgui 1:

- $a = 0$, $b = 0$ i $c = 0$
- $a = 0$, $b = 1$ i $c = 1$
- $a = 1$, $b = 0$ i $c = 1$

2. Per a cadascuna d'aquestes tres combinacions trobarem un terme producte, anomenat **minterm**. Total, 3 termes.

3. Cadascuna de les tres combinacions genera el *minterm* associat:

   - En la primera combinació de valors, totes les variables estan a zero ($a = 0$, $b = 0$ i $c = 0$) i, per tant, aquestes variables estaran invertides en el minterm associat: $\overline{a} \cdot \overline{b} \cdot \overline{c}$.
   - En la segona combinació de valors, només la variable *a* està a zero ($a = 0$, $b = 1$ i $c = 1$). Per tant, el minterm associat serà: $\overline{a} \cdot b \cdot c$.
   - En la tercera combinació de valors, només la variable *b* està a zero ($a = 1$, $b = 0$ i $c = 1$). Per tant, el minterm associat serà: $a \cdot \overline{b} \cdot c$.

4. Finalment, només cal fer la suma lògica dels tres minterms per obtenir la primera forma canònica de la funció:

$$f = \overline{a} \cdot \overline{b} \cdot \overline{c} + \overline{a} \cdot b \cdot c + a \cdot \overline{b} \cdot c$$

### Segona forma canònica

La segona forma canònica es basa a expressar la funció com un **producte de sumes** que deduirem de la taula de veritat mitjançant el següent procediment:

1. A la taula de veritat, ens fixarem en les combinacions de valors de les variables d'entrada que fan que la funció valgui 0.

2. Per cadascuna d'aquestes combinacions hem de trobar un terme producte, anomenat **maxterm**, amb les següents característiques:

   - Està format per totes les variables de la funció, negades o no.
   - Aquestes variables estan sumades.

3. Per saber si la variable ha d'estar negada o no a cada maxterm, us heu de fixar en la combinació de valors:

   - Si el valor d'aquesta variable és 0, aquesta variable haurà d'anar directa (no negada) en el terme suma.
   - Si el valor d'aquesta variable és 1, aquesta variable haurà d'anar negada en el terme suma.

4. Finalment, la segona forma canònica s'obté multiplicant tots els maxterms.

Recordeu que cal usar els parèntesis si es vol expressar correctament la funció com un producte de sumes.

**Exemple: obtenció de la segona forma canònica d'una funció**

A partir de la taula de veritat mostrada en la taula 1.15, trobarem la segona forma canònica de la funció.

Seguint el procediment descrit anteriorment:

1. Primer ens fixem en les combinacions de valors que fan que la funció valgui 0:

   - $a = 0$, $b = 0$ i $c = 1$

- $a = 0$, $b = 1$ i $c = 0$
- $a = 1$, $b = 0$ i $c = 0$
- $a = 1$, $b = 1$ i $c = 0$
- $a = 1$, $b = 1$ i $c = 1$

2. Per a cadascuna d'aquestes cinc combinacions trobarem un **maxterm** o terme producte.
3. Els *maxterms* que corresponen a cada combinació són els següents:
   - A la combinació $a = 0$, $b = 0$ i $c = 1$ li correspon $a + b + \overline{c}$
   - A la combinació $a = 0$, $b = 1$ i $c = 0$ li correspon $a + \overline{b} + c$
   - A la combinació $a = 1$, $b = 0$ i $c = 0$ li correspon $\overline{a} + b + c$
   - A la combinació $a = 1$, $b = 1$ i $c = 0$ li correspon $\overline{a} + \overline{b} + c$
   - A la combinació $a = 1$, $b = 1$ i $c = 1$ li correspon $\overline{a} + \overline{b} + \overline{c}$
4. Finalment, només cal fer el producte lògic dels cinc maxterms per obtenir la segona forma canònica de la funció:

$$f = (a + b + \overline{c}) \cdot (a + \overline{b} + c) \cdot (\overline{a} + b + c) \cdot (\overline{a} + \overline{b} + c) \cdot (\overline{a} + \overline{b} + \overline{c})$$

**Les portes lògiques...**

... són per als sistemes digitals com els maons per a l'arquitectura: malgrat la seva senzillesa, ens permeten construir estructures molt complexes.

### 1.6.5 Portes lògiques bàsiques

Fins i tot els sistemes digitals més complexos estan formats per blocs més petits encarregats de fer operacions senzilles amb senyals binaris. Aquests blocs fonamentals són el que coneixem com a **portes lògiques.**

**Símbols de les portes**

Les portes lògiques més importants disposen de símbols gràfics especials que les representen. Hi ha dues maneres de representar cada porta. Les normes que regulen aquestes dues simbologies són les següents:

- La norma IEEE 91-1973, que va ser la primera i més popular.
- La norma IEEE 91-1984, que és més recent i proporciona informació més precisa, encara que no s'utilitza tan sovint.

Cada porta lògica fa una funció determinada; de totes les possibles funcions existents, les **portes lògiques bàsiques** són aquelles que es constitueixen com a bons elements de disseny de propòsit general. Les portes lògiques bàsiques són la porta AND, la porta OR , la porta NOT, la porta NAND i la porta NOR.

#### Porta AND

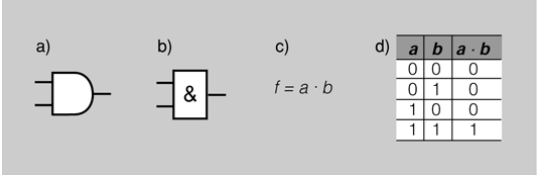
La porta AND fa la funció de producte lògic i rep el nom de la paraula anglesa **and**, que significa *i* en català.

> La sortida d'una porta AND és 1 només si totes les variables d'entrada són 1.

La figura 1.1 mostra el símbols d'una porta AND de dues entrades, però els mateixos símbols serveixen per a un nombre d'entrades més gran, només cal afegir les connexions d'entrada necessàries al símbol.

IEEE és la sigla en anglès de l'Institut d'Enginyers en Electricitat i Electrònica. En català s'acostuma a pronunciar com al E cub.

**Figura 1.1.** Porta AND de dues entrades/a) símbol IEEE 91-1973; b) símbol IEEE 91-1984; c) expressió algebraica; d) taula de veritat

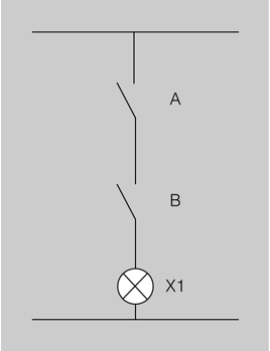



| a | b | a · b |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

La funció que realitza la porta AND és anàloga a la funció que realitzen dos interruptors en sèrie.

Com podeu veure en la figura 1.2, la làmpada X1 tan sols s'encén (és a dir, la sortida es posa a estat 1 lògic) quan els interruptors A i B estan tancats (és a dir, quan les dues entrades estan a 1 lògic).

**Figura 1.2.** Porta AND a partir de dos interruptors en sèrie



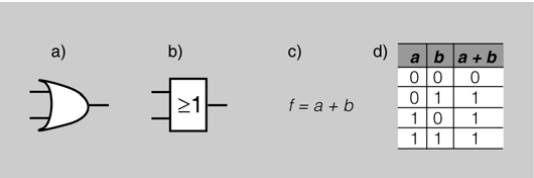

### Porta OR

La porta OR realitza la funció suma lògica i rep el nom de la paraula anglesa **or**, que significa *o* en català.

La sortida d'una porta OR és 1 si, com a mínim, una variable d'entrada és 1.

La figura 1.3 mostra el símbols d'una porta OR de dues entrades, la seva expressió algebraica i la seva taula de veritat.

**Figura 1.3.** Porta OR de dues entrades/a) símbol IEEE 91-1973; b) símbol IEEE 91-1984; c) expressió algebraica; d) taula de veritat

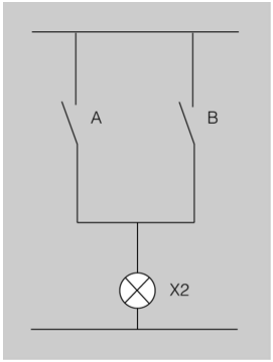



| *a* | *b* | *a + b* |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

La funció que realitza la porta OR és anàloga a la funció que realitzen dos interruptors en paral·lel.

Com podeu veure en la figura 1.4, la làmpada X2 s'encén (és a dir, la sortida es posa a estat 1 lògic) quan, com a mínim, un dels dos interruptors A i B està tancat.

**Figura 1.4.** Porta OR a partir de dos interruptors en paral·lel

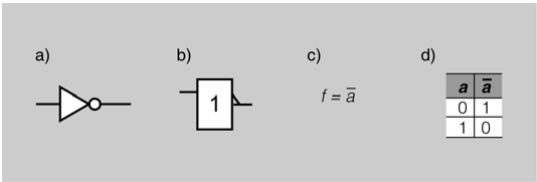



### Porta NOT

Aquesta porta, també coneguda com a **inversor**, només té una entrada i realitza la funció de negació, de manera que la sortida és 1 si l'entrada és 0 i la sortida és 0 si l'entrada és 1.

La figura 1.5 mostra els símbols d'una porta NOT, la seva expressió algebraica i la seva taula de veritat.

**Figura 1.5.** Porta NOT o inversor/a) símbol IEEE 91-1973; b) símbol IEEE 91-1984; c) expressió algebraica; d) taula de veritat




| $a$ | $\overline{a}$ |
|---|---|
| 0 | 1 |
| 1 | 0 |

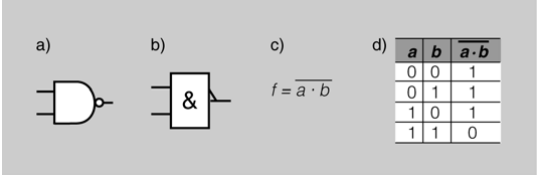
### Porta NAND

El nom d'aquesta porta prové de la unió de les paraules angleses notiand i, com el seu nom indica, és una porta AND amb la sortida invertida.

> La sortida d'una porta NAND és 0 només si totes les variables d'entrada són 1.

La figura 1.6 mostra els símbols d'una porta NAND, la seva expressió algebraica i la seva taula de veritat.

**Figura 1.6.** Porta NAND de dues entrades/a) símbol IEEE 91-1973; b) símbol IEEE 91-1984; c) expressió algebraica; d) taula de veritat




| $a$ | $b$ | $\overline{a \cdot b}$ |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

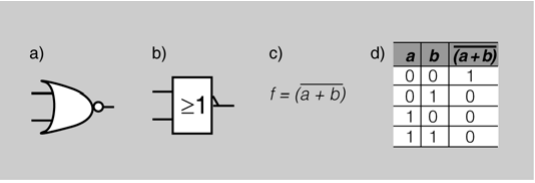
### Porta NOR

La sortida d'una porta NOR és la invertida d'una porta OR. El seu nom prové de la unió de les paraules angleses noti or.

> La sortida d'una porta NOR és 1 només si totes les variables d'entrada són 0.

La figura 1.7 mostra els símbols d'una porta NOR, la seva expressió algebraica i la seva taula de veritat.

**Figura 1.7.** Porta NOR de dues entrades/a) símbol IEEE 91-1973; b) símbol IEEE 91-1984; c) expressió algebraica; d) taula de veritat




| a | b | (a+b) |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 0 |

### Porta XOR

També anomenada **porta OR-exclusiva**. La seva sortida és 1 només quan les dues variables d'entrada tenen valors diferents.

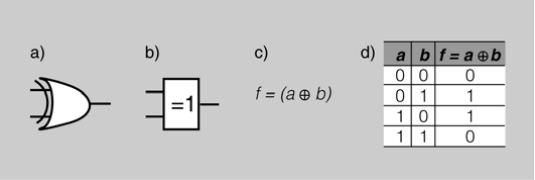
L'operador XOR en les expressions algebraiques és el símbol $\oplus$, un signe de suma encerclat.

En el cas de tenir més de dues entrades, podeu recordar com funciona una porta XOR si recordeu el següent:

> La sortida d'una porta XOR és 1 només si hi ha un nombre senar d'entrades a 1.

La figura 14 mostra els símbols d'una porta XOR, la seva expressió algebraica i la seva taula de veritat.

**Figura 1.8.** Porta XOR de dues entrades/a) símbol IEEE 91-1973; b) símbol IEEE 91-1984; c) expressió algebraica; d) taula de veritat




| a | b | f = a ⊕ b |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

## 2. Circuits combinacionals i seqüencials

Les portes lògiques donen solució a problemes que es plantegen en la vida quotidiana. Darrere de fets tan comuns com la utilització d'una calculadora, el control dels semàfors dels carrers, o fins i tot l'ús dels ordinadors trobem l'electrònica digital i les portes lògiques.

A mesura que els problemes són més complexos, també ho són els circuits per resoldre'ls. D'altra banda hi ha determinats processos que es donen de manera molt freqüent i que, per tant, es poden estandarditzar i tractar com un sol bloc.

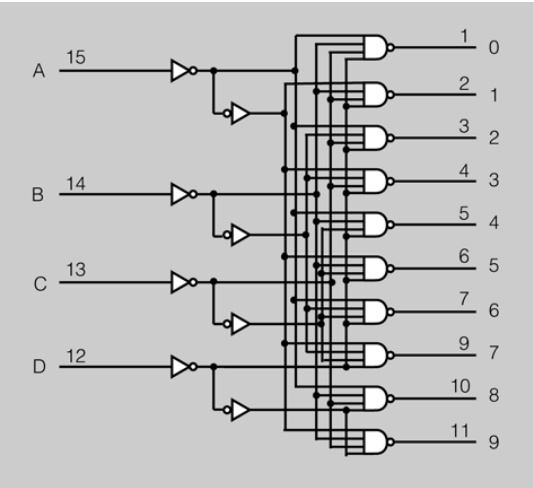
Posem un exemple. Si una fàbrica ha de fer caragols d'unes determinades característiques, amb tota seguretat en comptes de fer-los un per un, de manera artesana, farà un motlle perquè surtin tots iguals.

Bé, doncs, en l'electrònica digital combinacional, i també en la seqüencial, el que es fa és desenvolupar blocs integrats que donen solucions a problemes concrets, sense haver de preocupar-nos de les portes que hi ha a dins i de com es troben interconnectades entre elles.

L'avenç progressiu de les tècniques d'integració ha permès la realització, en un circuit integrat, de sistemes combinacionals i seqüencials complexos formats per un gran nombre de portes lògiques. Aquesta integració redueix el nombre d'elements necessaris, ja que permet la combinació de diferents tipus de portes en un mateix bloc integrat, disminueix el temps de disseny i el nombre de connexions.

En la figura 2.1 es mostra un exemple de circuit combinacional, amb quatre entrades i deu sortides.

**Figura 2.1.** Exemple de circuit combinacional



Els **circuits combinacionals** són blocs formats per portes lògiques bàsiques (NOT, AND, OR, NAND, NOR, OR-exclusiva i NOR-exclusiva) que tenen diferents entrades i sortides, i en els quals els valors de la sortida o de les sortides dependran exclusivament del valor de les entrades en aquell instant.

Els blocs integrats combinacionals donen resposta a aplicacions de caire general, com ara codificadors, descodificadors, multiplexors, desmultiplexors, comparadors, etc.

Els **circuits seqüencials** són blocs formats per portes lògiques bàsiques que tenen diferents entrades i sortides, i en els quals els valors dels senyals de sortida no depenen solament dels valors dels senyals d'entrada, sinó també dels valors que les mateixes sortides tenien abans.

Els principals circuits seqüencials són els registres, els comptadors i els registres de desplaçament.

## 2.1 Multiplexors i desmultiplexors

L'enviament d'informació mitjançant senyals elèctrics d'un lloc a un altre es pot fer utilitzant una o diverses línies. Quan s'envia amb una sola línia estem parlant d'una transmissió en sèrie i quan es fa utilitzant diverses línies parlem d'una transmissió en paral·lel.

Si heu d'enviar una informació d'un byte i disposeu de vuit línies (cables de connexió) per a cada unitat de temps, podreu enviar un byte sencer. Si aquest mateix byte s'envia per una sola línia necessitareu 8 unitats de temps, ja que en aquest cas heu d'enviar un bit darrere l'altre, però necessitareu només un cable per connectar els dos llocs.

Per tant, el sistema que caldrà utilitzar dependrà d'aspectes econòmics, de la distància entre els llocs que volem comunicar i de la velocitat de transmissió de les dades. Aquesta necessitat de passar de diverses línies a una de sola (de paral·lel a sèrie) i a l'inrevés (de sèrie a paral·lel) porta a la multiplexació i a la desmultiplexació, respectivament.

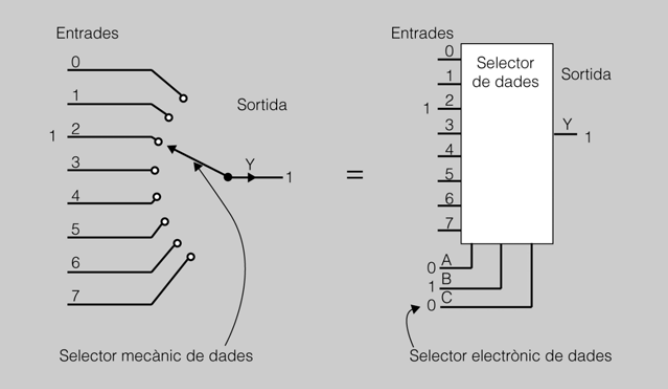
### 2.1.1 Multiplexors

El **multiplexor** és un sistema digital que disposa de *N*entrades d'informació i una única sortida. Mitjançant unes entrades de control, selecciona internament una de les entrades d'informació i la connecta a l'única sortida.

En la figura 2.2 es pot veure com l'entrada de selecció és l'encarregada de connectar, en aquest cas, l'entrada corresponent al 2 amb la sortida Y. Gràficament es pot veure com l'entrada d'informació és en paral·lel i la sortida és en sèrie, només d'una línia.

Fixeu-vos com amb les entrades de control ABC (010 = 2) seleccionem l'entrada de dades 2, i com internament el que es fa és un pont entre aquesta entrada i la sortida (Y). Com que el nivell a l'entrada 2 és 1 (nivell alt), a la sortida trobem aquest nivell 1.

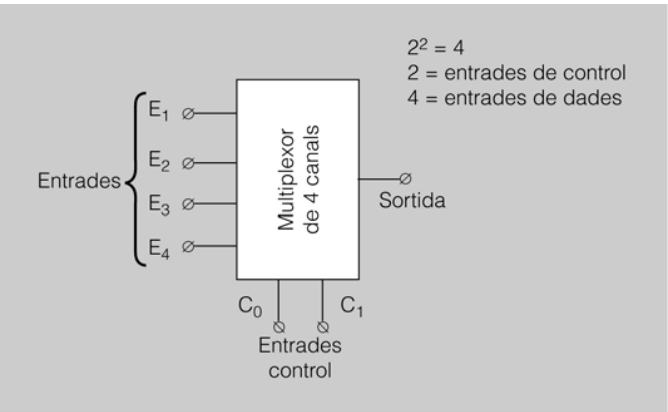
**Figura 2.2.** Simplificació del funcionament d'un multiplexor




Als multiplexors s'ha de complir que les entrades d'informació $N$ siguin igual a $2^M$, essent $M$ el nombre d'entrades de control.

En la figura 2.3, podeu observar com les entrades per multiplexar poden ser com a màxim 4 si el nombre de línies de control és 2.

**Figura 2.3.** Relació entre entrades d'informació i de control a un MUX de quatre línies




**Circuit integrat 74151**

El circuit integrat comercial 74151 és un petit xip de setze pius que implementa la funció de multiplexor de vuit línies.

En la taula 2.1 es mostra com funciona el multiplexor de quatre línies: si les entrades de selecció prenen el valor 00, a la sortida apareix el senyal de l'entrada $E_1$. Si les entrades de selecció prenen el valor 01, a la sortida apareix el senyal de l'entrada $E_2$. Si les entrades de selecció prenen el valor 10, a la sortida apareix el senyal de l'entrada $E_3$. I, finalment, si les entrades de selecció prenen el valor 11, a la sortida apareix el senyal de l'entrada $E_4$.

**Taula 2.1.** Selecció de la sortida a un multiplexor de quatre línies

| $C_1$ | $C_0$ | S |
|---|---|---|
| 0 | 0 | $E_1$ |
| 0 | 1 | $E_2$ |
| 1 | 0 | $E_3$ |
| 1 | 1 | $E_4$ |

### 2.1.2 Desmultiplexors

El desmultiplexor és un sistema digital que disposa d'una sola entrada d'informació i *N*sortides. Mitjançant unes entrades de control, selecciona internament una de les sortides d'informació i la connecta a l'única entrada.

Com podeu comprovar, el seu funcionament és a l'inrevés que el multiplexor. Gràficament ho podeu acabar de veure a la figura 2.4 i figura 2.5.

**Figura 2.4.** Desmultiplexor mecànic de quatre línies

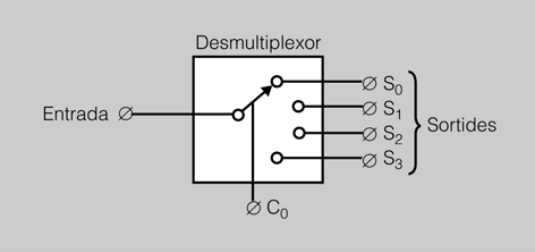
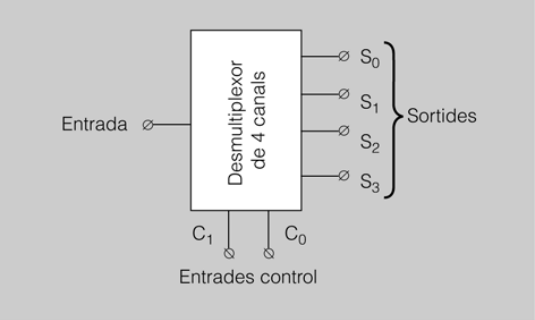



**Figura 2.5.** Desmultiplexor digital de quatre línies




**Circuit integrat 74155**

El circuit integrat comercial 74155 és un petit xip de setze pius que implementa la funció de desmultiplexor d'una a vuit línies.

Als desmultiplexors s'ha de complir que les sortides d'informació $N$ siguin igual a $2^M$, essent $M$ el nombre d'entrades de control.

La taula 2.2 és la taula de funcionament corresponent al desmultiplexor de quatre línies de la figura 2.5. Com podeu observar en la taula, la dada de l'entrada (1 o 0) apareixerà a la sortida seleccionada amb les línies de control $C_1$ i $C_0$.

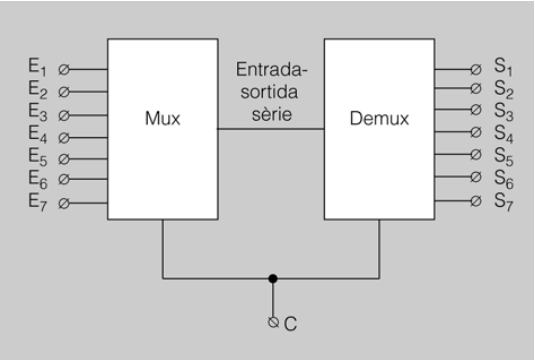
**Taula 2.2.** Taula de veritat d'un desmultiplexor digital de quatre línies

| $C_1$ | $C_0$ | Entrada | $S_3$ | $S_2$ | $S_1$ | $S_0$ |
|---|---|---|---|---|---|---|
| 0 | 0 | E | 0 | 0 | 0 | E |
| 0 | 1 | E | 0 | 0 | E | 0 |
| 1 | 0 | E | 0 | E | 0 | 0 |
| 1 | 1 | E | E | 0 | 0 | 0 |

### 2.1.3 Connexió multiplexor-desmultiplexor

En la figura 2.6 podeu veure l'aplicació del multiplexor (*Mux*) i desmultiplexor (*Demux*) per fer una transmissió de dades. El primer converteix la informació de paral·lel a sèrie; d'aquesta manera, la informació és enviada només amb una línia. La informació, quan arriba al desmultiplexor, es converteix de sèrie a paral·lel per tornar a tenir la informació original.

**Figura 2.6.** Connexió multiplexor-desmultiplexor



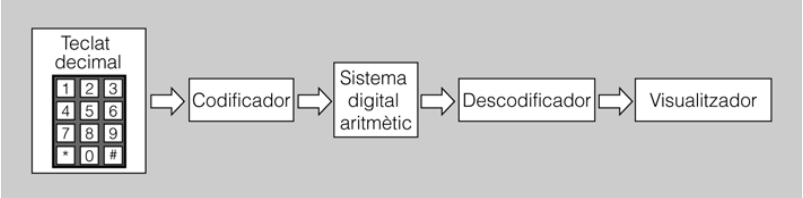

Fixeu-vos com tots dos elements estan connectats per un senyal de control. Aquest té la missió de sincronitzar els dos dispositius. Posarem un exemple. Quan el multiplexor està seleccionant l'entrada $E_2$, el desmultiplexor ha d'estar seleccionant la sortida $S_2$; quan el multiplexor selecciona l'entrada $E_3$, el desmultiplexor ha de seleccionar la sortida $S_3$, etc. És la manera perquè la informació a la sortida sigui la mateixa que a l'entrada.

## 2.2 Descodificadors i codificadors

Per una banda, els sistemes digitals només treballen amb els bits 1 i 0 i, per l'altra, a les persones ens és difícil interpretar grans combinacions d'uns i zeros. Per donar resposta a aquesta situació, s'utilitzen els convertidors de codi, encarregats de traduir el nostre codi al llenguatge que entén el sistema digital. Si agafem com a exemple una calculadora, podem distingir tres grans blocs (figura 2.7):

1. Un bloc codificador, on es produeix una codificació o conversió del codi decimal (teclat) a un codi binari (1 i 0).
2. El sistema encarregat de fer les operacions que només interpreta bits 1 i 0.
3. Un bloc descodificador que tradueix el resultat de l'operació, en binari, a un llenguatge que podem interpretar, en decimal.

**Figura 2.7.** Diagrama de blocs del funcionament intern d'una calculadora




**Traducció**

Els codificadors tradueixen el llenguatge que nosaltres utilitzem al codi que entén el sistema digital. Els descodificadors tradueixen el llenguatge que utilitza el sistema digital a un llenguatge que nosaltres entenem.

Els codificadors són sistemes combinacionals que s'encarreguen de transformar una sèrie de senyals sense codificar en un conjunt de senyals codificats.

Els descodificadors són circuits integrats digitals que fan la conversió d'un codi binari, normalment el BCD, en una forma sense codificar.

### 2.2.1 Descodificadors

Els descodificadors tradueixen el llenguatge que utilitza el sistema digital a un llenguatge que nosaltres puguem entendre.

Els **descodificadors** són circuits combinacionals que si tenen $n$ nombre d'entrades poden presentar fins a $2^n$ nombre de sortides.

**Taula 2.3.** Taula de veritat d'un descodificador de dues entrades

| **Entrades** | | **Sortides** | | | |
|---|---|---|---|---|---|
| **A** | **B** | **$Q_3$** | **$Q_2$** | **$Q_1$** | **$Q_0$** |
| 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 0 | 1 | 0 | 0 |
| 1 | 1 | 1 | 0 | 0 | 0 |

**Descodificador de dues entrades**

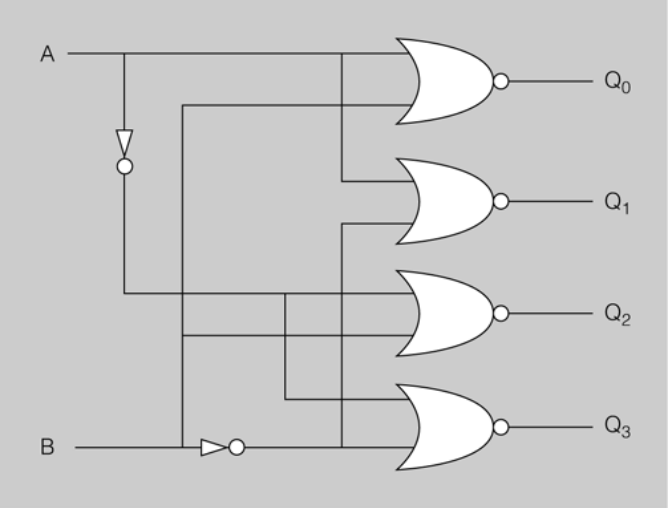
Un descodificador de dues entrades té les següents característiques:

- Nombre d'entrades: $n = 2$
- Nombre de sortides: $N = 2^n = 4$

En la taula 2.3 es mostra la taula de veritat d'aquest circuit. El seu funcionament és molt senzill: per a cada combinació binària de les entrades, es posa a 1 la sortida corresponent.

En la figura 2.8 es mostra el circuit amb portes lògiques que implementa la funció de descodificador de dues entrades.

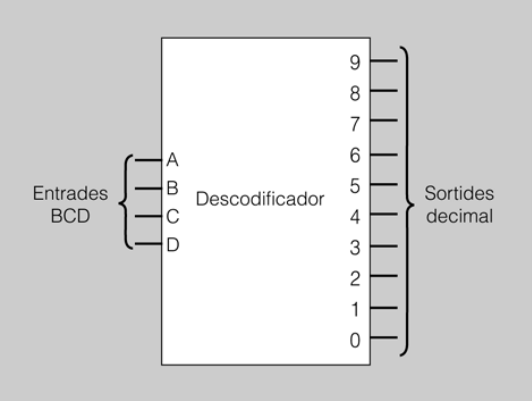
**Figura 2.8.** Circuit amb portes lògiques que fa la funció de descodificador de dues entrades




## Descodificadors BCD/decimal

Els descodificadors que més s'utilitzen són els **descodificadors BCD/decimal**. En la figura 2.9 es mostra un diagrama d'aquests circuits.

**Figura 2.9.** Descodificador BCD-decimal




**Circuit integrat 7442**

El circuit integrat comercial 7442 és un petit xip de catorze pius que implementa la funció de descodificador BCD/decimal.

Els descodificadors tenen quatre línies d'entrada que formen un codi BCD. A la sortida tenen deu línies, de manera que es posa a 1 la sortida que correspon al nombre decimal representat a l'entrada, mentre que les altres sortides es mantenen a 0.

En la taula 2.4 es mostra la taula de veritat del descodificador BCD/decimal.

**TAULA 2.4.** Taula de veritat d'un descodificador BCD/decimal

| Entrades | | | | Sortides | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **A** | **B** | **C** | **D** | $Q_9$ | $Q_8$ | $Q_7$ | $Q_6$ | $Q_5$ | $Q_4$ | $Q_3$ | $Q_2$ | $Q_1$ | $Q_0$ |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Descodificadors BCD/set segments

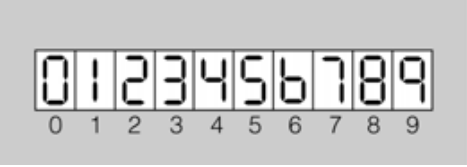
Per veure el funcionament d'aquest tipus de circuit, abans cal fer esment dels dispositius visualitzadors de set segments.

Els visualitzadors de set segments, que podeu veure en la figura **??**, s'utilitzen per visualitzar nombres decimals. Els set segments s'identifiquen amb les lletres *a, b, c, d, e, f* i*g*.

**File ieam9uf1_im28.png does not exist.**

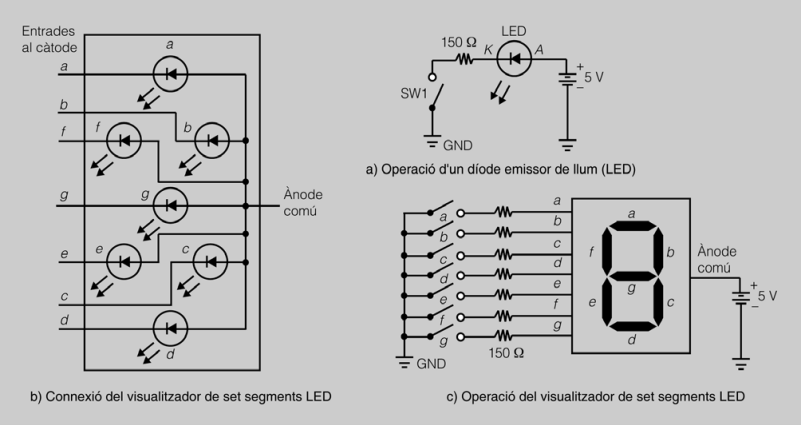
En la figura 2.10 podeu observar com s'aconsegueix la visualització dels deu dígits decimals (del 0 al 9). Per exemple, si s'activen tots menys el segment *g* es visualitza el zero. En canvi, si s'activen el segment *a, b* i*c,* es visualitza el 7, i si s'activen tots els segments, el 8.

**FIGURA 2.10.** Visualitzador de set segments




En la figura 2.11 podeu veure la composició interna d'un visualitzador de set segments, el seu funcionament elèctric i la connexió.

**FIGURA 2.11.** Connexió, composició interna i funcionament d'un visualitzador de set segments




Cada segment està fet a partir d'un díode LED. Quan aquest díode LED és travessat pel corrent elèctric, emet radiacions lluminoses.

Els elements del visualitzador sempre han de portar una resistència limitadora. El valor d'aquesta resistència es calcula mitjançant la següent equació:

$$R = \frac{V_{CC} - V_{LED}}{I_{LED}}$$

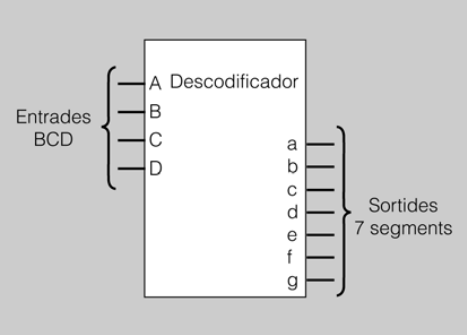
Pel que fa al càlcul, es considera el voltatge LED aproximadament de 2 V i el corrent LED necessari al voltant dels 20 mA, per a LEDs estàndard.

Tal com s'aprecia en la figura 2.11, per a una alimentació $V_{CC}$ de 5 V la resistència limitadora hauria de ser d'uns 150 Ω. Col·locant una resistència limitadora de 500 Ω, podríeu connectar el mateix *visualitzador* a un voltatge d'alimentació de 12 V. Podeu comprobar que en la connexió del visualitzador de set segments, cada segment conté un LED. Tots els ànodes estan connectats entre si (connexió ànode comú) i a positiu. Per tant, quan en el càtode d'un determinat díode apareix un nivell baix, o 0 lògic, el díode emetrà llum.

També podeu trobar visualitzadors amb la connexió de càtode comú. En aquest cas, el segment s'encendrà quan hi apliqueu un nivell alt (1 lògic).

En la figura 2.12 es representa el diagrama en bloc d'un descodificador BCD/ set segments. Com podeu apreciar, hi ha les quatre entrades corresponents al codi BCD i set sortides, una per a cada segment del visualitzador.

**FIGURA 2.12.** Descodificador BCD-set segments




**Circuit integrat 7447**

El circuit integrat comercial 7447 és un petit xip de setze pius que implementa la funció de descodificador BCD/set segments.

En la taula 2.5 es mostra el funcionament del descodificador BCD/set segments.

**TAULA 2.5.** Taula de veritat d'un descodificador BCD/set segments

| **Entrades** | | | | **Sortides** | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| **A** | **B** | **C** | **D** | **a** | **b** | **c** | **d** | **e** | **f** | **g** |
| 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 |
| 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 |

Les combinacions d'entrades entre 1010 (10) i 1111 (15) no pertanyen al codi BCD

## 2.2.2 Implementació de funcions lògiques amb descodificadors

Una de les aplicacions dels descodificadors és la d'implementar funcions lògiques.

**TAULA 2.6.** Taula de veritat de la funció $f = (a \cdot \overline{b} \cdot \overline{c}) + (a \cdot b \cdot \overline{c}) + (\overline{a} \cdot \overline{b} \cdot c) + (a \cdot b \cdot c)$

| **c** | **b** | **a** | **f** |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 |

Per exemple, si volem implementar amb un descodificador la funció lògica $f = (a \cdot \overline{b} \cdot \overline{c}) + (a \cdot b \cdot \overline{c}) + (\overline{a} \cdot \overline{b} \cdot c) + (a \cdot b \cdot c)$, els passos que haurem de seguir seran els següents:

1. Escrivim la taula de veritat de la funció, com es mostra en la taula 2.6.

2. Utilitzem un descodificador amb un nombre d'entrades igual o més gran que el nombre de variables de la funció. Com que, en aquest cas, tenim tres variables ($a$, $b$, $c$) utilitzarem un descodificador BCD/decimal.

3. Com es mostra en la figura 2.13, connectem les variables de la funció a les entrades del descodificador BCD/decimal. L'entrada de més pes del descodificador la connectem a massa, ja que no la utilitzem.

4. Mirem les combinacions de les variables que donen 1:

   - Q1 = 001
   - Q3 = 011
   - Q4 = 100
   - Q7 = 111

5. Finalment, connectem a una porta OR les sortides del descodificador que es posen a 1. En la figura 2.14 es mostra el circuit final.

**Figura 2.13.** Connexió de les variables a, b i c a les entrades del descodificador BCD-decimal

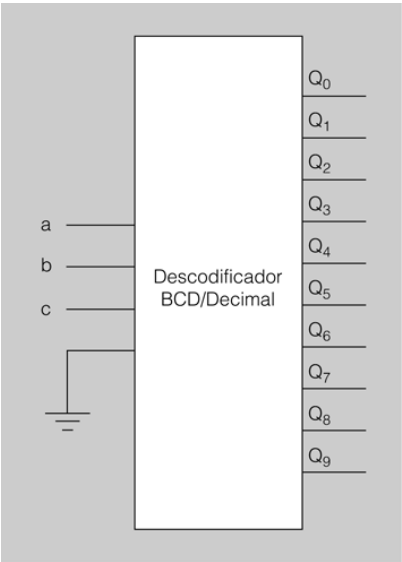
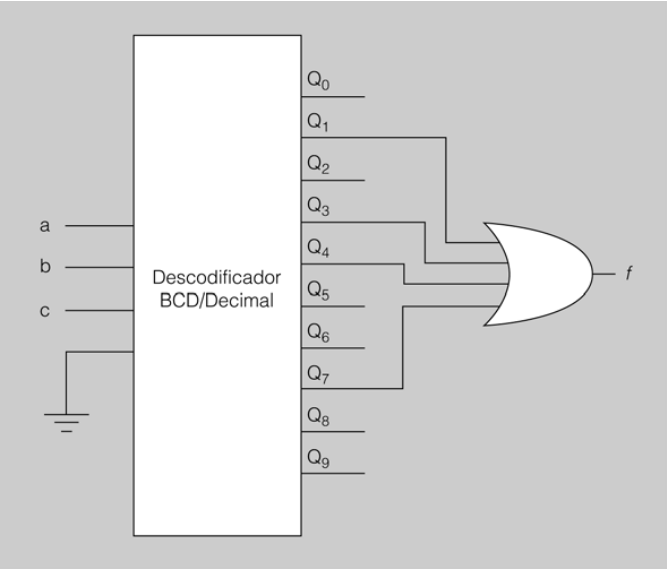


**Figura 2.14.** Circuit amb descodificador que implementa la funció $f=(a\cdot \overline{b}\cdot \overline{c})+(a\cdot b\cdot\overline{c})+(\overline{a}\cdot \overline{b}\cdot c)+(a\cdot b\cdot c)$




D'aquesta manera, a la sortida de la porta OR obtindrem la funció *f*.

### 2.2.3 Codificadors

Els codificadors tradueixen el llenguatge que utilitzem (normalment, codi decimal) al codi que entén el sistema digital (normalment, codi binari).

> Els codificadors són circuits combinacionals que si tenen $2^n$ nombre d'entrades presenten *n* nombre de sortides.

#### Codificador decimal/BCD

Un dels tipus de codificador més utilitzat és el codificador decimal/BCD. En aquest circuit, quan s'activa una de les entrades decimals, les sortides agafen l'estat corresponent al seu codi BCD. En la taula 2.7 es mostra la seva taula de veritat.

Els codificadors decimal/BCD són **prioritaris**, això vol dir que si hi ha més d'una entrada activada, la sortida que s'activarà correspondrà a l'entrada de major pes o de menor pes, segons la seva estructura interna.

Aquests codificadors tenen la sortida $Y_0$ que s'activa en el cas que totes les entrades estiguin a zero. D'aquesta manera es distingeix entre dos casos:

- Quan s'activa l'entrada $E_0$, les sortides $Q_3$, $Q_2$, $Q_1$ i $Q_0$ estan a zero i la sortida $Y_0$, també.
- Quan no hi ha cap entrada activada, les sortides $Q_3$, $Q_2$, $Q_1$ i $Q_0$ estan a zero

i la sortida $Y_0$ es posa a 1.

**Circuit integrat 74147**

El circuit integrat comercial 74147 és un petit xip de setze pius que implementa la funció de codificador decimal/BCD.

**Taula 2.7.** Taula de veritat d'un codificador decimal/BCD amb prioritat al major pes

| Entrades | | | | | | | | | | Sortides | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $E_9$ | $E_8$ | $E_7$ | $E_6$ | $E_5$ | $E_4$ | $E_3$ | $E_2$ | $E_1$ | $E_0$ | $Y_0$ | $Q_3$ | $Q_2$ | $Q_1$ | $Q_0$ |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | X | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | X | X | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | X | X | X | 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 0 | 0 | 0 | 1 | X | X | X | X | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 1 | X | X | X | X | X | 0 | 0 | 1 | 0 | 1 |
| 0 | 0 | 0 | 1 | X | X | X | X | X | X | 0 | 0 | 1 | 1 | 0 |
| 0 | 0 | 1 | X | X | X | X | X | X | X | 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | X | X | X | X | X | X | X | X | 0 | 1 | 0 | 0 | 0 |
| 1 | X | X | X | X | X | X | X | X | X | 0 | 1 | 0 | 0 | 1 |

El valor X indica que és indiferent el valor que s'hi posi, tant 0 com 1

## 2.3 Circuits comparadors

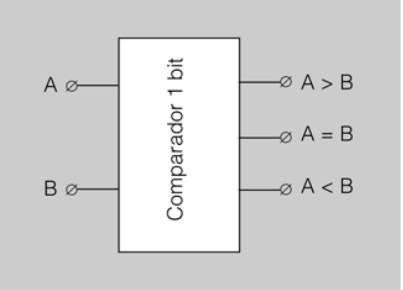
Les aplicacions de lògica digital combinacional requereixen moltes vegades realitzar la comparació entre dades binàries procedents de diferents fonts.

Un circuit comparador és aquell que pot comparar dos nombres binaris ($A$ i $B$) de $n$ bits cadascun i determinar si són iguals ($A = B$), o si no ho són, quin és més gran ($A > B$) o més petit que l'altre ($A < B$).

> Els comparadors són circuits combinacionals que indiquen, a la seva sortida, la relació d'igualtat o desigualtat entre dos nombres ($A$ i $B$) de $n$ bits cadascun.

El circuit comparador més senzill és aquell que permet fer la comparació entre dos nombres d'un bit. En la figura 2.15 podeu veure la seva representació.

**Figura 2.15.** Circuit comparador d'un bit



Com podem observar, hi ha, com a entrades, els dos nombres per comparar $A$ i $B$ i les tres sortides possibles $A > B$, $A = B$ i $A < B$.

**Circuit integrat 7485**

El circuit integrat comercial 7485 és un petit xip de setze pius que permet comparar dos nombres binaris de 4 bits.

La taula de veritat d'aquest circuit comparador d'un bit es mostra a la taula 2.8.

**TAULA 2.8.** Taula de veritat d'un comparador d'un bit

| Entrades | | Sortides | | |
|---|---|---|---|---|
| **A** | **B** | **A > B** | **A = B** | **A < B** |
| 0 | 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | 1 | 0 |

## 2.4 Circuits seqüencials

Un semàfor és un exemple típic de sistema seqüencial.

Fixem-nos en el funcionament d'un semàfor qualsevol del nostre municipi. Un semàfor és un sistema digital típic i les seves sortides són les següents:

- llum vermell per a vehicles
- llum taronja per a vehicles
- llum verd per a vehicles
- llum vermell per a vianants
- llum verd per a vianants

Aquestes cinc sortides només poden prendre dos valors diferents: llum encès o llum apagat. Com a única entrada possible, un semàfor pot presentar el polsador perquè els vianants puguin dir al semàfor que volen creuar el carrer.

Imaginem-nos que arribem a un semàfor passejant. Volem creuar el carrer però el llum dels vianants està en vermell i els vehicles circulen perquè tenen el llum verd encès. Com que tenim pressa, premem el polsador del semàfor i, després de pocs segons, el semàfor dels vehicles es posa en vermell (passant primer pel taronja) i el semàfor dels vianants es posa en verd. Fins aquí tot molt bé: quin semàfor més amable! Ens ha obeït de seguida.

Però què passarà si ara, amb el llum verd per als vianants encès, tornem a prémer el polsador? Doncs que el semàfor no ens farà ni cas i, transcorreguts uns quants segons, continuarà amb la seva seqüència de funcionament normal, deixant passar els vehicles i encenent el llum vermell per als vianants. Si pogués parlar, el semàfor ens diria alguna cosa semblant a: "No cal que tornis a prémer el polsador que ja t'he entès a la primera! Això sí: espavila't perquè en pocs segons jo continuaré fent la meva feina i no et deixaré passar!".

**Cronòmetre**

Penseu en un cronòmetre digital amb un polsador d'inici/aturada i un polsador de reinicialització o *RESET* per tornar a zero. És un sistema digital seqüencial? Per què?

Això significa que el semàfor no sempre dóna les mateixes sortides per a una mateixa combinació de les entrades. Aquest semàfor és un exemple típic de **sistema digital seqüencial**.

En un sistema digital seqüencial, els valors dels senyals de sortida no depenen solament dels valors dels senyals d'entrada, sinó també dels valors que les mateixes sortides tenien abans.

Els circuits seqüencials elementals són els **biestables**. A partir de biestables es poden construir circuits seqüencials més complexos com, per exemple, els **comptadors**i els **registres de desplaçament**.

Us podeu trobar amb llibres tècnics que anomenen al nivell alt i al nivell baix estat de **SET** i estat de **RESET**, respectivament.

### 2.4.1 Biestables: latches i flip-flops

Un biestable és un circuit digital seqüencial que pot tenir dos estats estables a la sortida denominats **nivell alt**i **nivell baix**, en els quals es pot mantenir indefinidament.

Un biestable canvia l'estat de la seva sortida segons les seves entrades i l'estat previ de la sortida.

**Un flanc...**

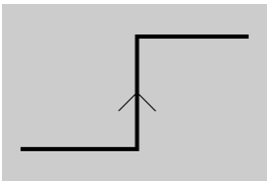
... és una transició d'un senyal digital de 0 a 1 o d'1 a 0. Si la transició és de 0 a 1 s'anomena *flanc positiu* o *flanc de pujada*. Si la transició és d'1 a 0 s'anomena *flanc negatiu* o *flanc de baixada*.

Els dispositius biestables es divideixen en dues categories:

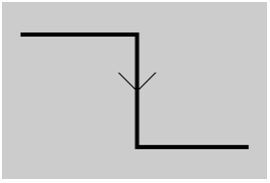
- *Latches*: són biestables asíncrons. Això vol dir que no utilitzen cap senyal de sincronització o rellotge.
- *Flip-flops*: són biestables síncrons. Això vol dir que la seva sortida canvia d'estat únicament en un instant específic d'una entrada de sincronització denominada **rellotge** (habitualment, en un instant concret anomenat **flanc**).

Els *flip-flops* es poden classificar segon el tipus de senyal de rellotge que utilitzen:

- *Flip-flop* disparat per flanc positiu: la sortida del *flip-flop* canvia quan el senyal de rellotge fa una transició de 0 a 1.
- *Flip-flop* disparat per flanc negatiu: la sortida del *flip-flop* canvia quan el senyal de rellotge fa una transició d'1 a 0.



Flanc positiu



Flanc negatiu

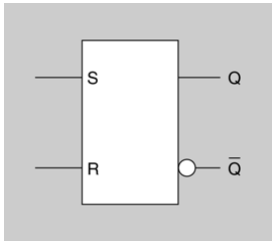
#### Latch S-R (SET-RESET)

El funcionament del *latch* S-R és el següent:

- Si posem l'entrada S (*SET*) a 1 i deixem l'entrada R (*RESET*) a 0, la sortida *Q* del *latch* S-R es posa a 1: és el que s'anomena **estat de SET**.
- Si posem l'entrada R (*RESET*) a 1 i deixem l'entrada S (*SET*) a 0, la sortida *Q* del *latch* S-R es posa a 0: és el que s'anomena **estat de RESET**.

- Si posem les dues entrades S i R a 0, les sortides no canvien: conserven l'estat. Això vol dir posar el mateix valor que tenien abans de canviar l'estat de les entrades. Aquesta característica és molt important, perquè el *latch* "memoritza" el seu estat anterior.
- La sortida $\overline{Q}$ sempre té el valor contrari a la sortida $Q$ (d'això se'n diu que les dues sortides són **complementàries**).

La taula de veritat del *latch* S-R es mostra en la taula 2.9.



Símbol lògic del latch S-R

**Taula 2.9.** Taula de veritat d'un latch S-R

| **Entrades** | | **Sortides** | | |
|---|---|---|---|---|
| **S** | **R** | $Q+$ | $\overline{Q+}$ | **Comentaris** |
| 0 | 0 | $Q$ | $\overline{Q}$ | Les sortides no canvien. Es queden en l'estat anterior |
| 1 | 0 | 1 | 0 | Estat de SET |
| 0 | 1 | 0 | 1 | Estat de RESET |
| 1 | 1 | ? | ? | **Condició no vàlida** |

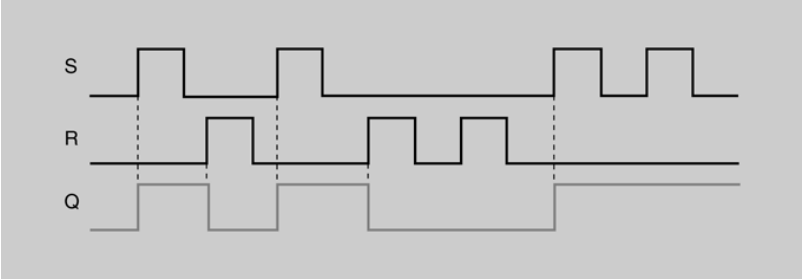
Amb Q+ es designa el valor posterior a la realització de la operació corresponent

I què passa quan posem les dues entrades del *latch* S-R a 1? Doncs que, a causa del funcionament intern del *latch* S-R, no podem predir quines sortides obtindrem.

No hem de posar mai les dues entrades del *latch* S-R a 1 simultàniament perquè no podem predir quin valor tindran les seves sortides. Aquesta imprecisió del valor de les sortides podria provocar problemes greus al circuit on estigués connectat el *latch* S-R.

En la figura 2.16 podem veure l'evolució amb el temps de les les formes d'ona de les entrades i la sortida d'un*latch* S-R. Se suposa que $Q$ es troba inicialment a nivell baix. Observeu que en cap moment les dues entrades es troben a nivell alt simultàniament.

**Figura 2.16.** Formes d'ona de les entrades i la sortida d'un latch S-R



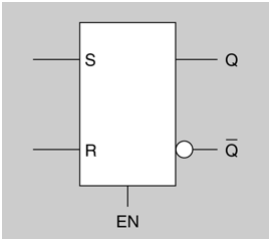

### Latch S-R amb entrada d'habilitació

El nom EN d'una entrada d'habilitació ve de la paraula anglesa *enable*, que vol dir habilitació.

A un *latch* S-R se li pot afegir una nova entrada: **l'entrada d'habilitació EN**.

La funció d'aquesta entrada **EN** és molt senzilla:

- Quan l'entrada EN està a nivell alt, el *latch* està habilitat i funciona com un *latch* S-R normal.
- Quan l'entrada EN està a nivell baix, simplement el *latch* S-R no fa res: per molt que canviem el valor de les entrades S i R, les sortides conserven l'estat anterior.



Símbol lògic del latch S-R amb entrada d'habilitació

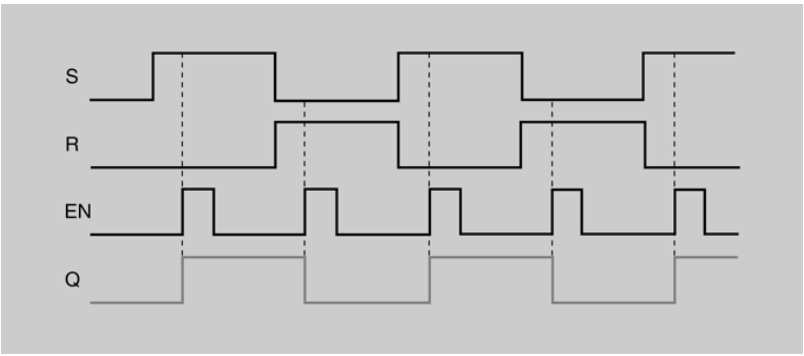
La taula de veritat d'aquest *latch*S-R amb entrada d'habilitació es mostra en la taula 2.10.

**Taula 2.10.** Taula de veritat d'un //latch// S-R amb entrada d'habilitació

| **Entrades** | | | **Sortides** | | |
|---|---|---|---|---|---|
| **EN** | **S** | **R** | $Q+$ | $\overline{Q+}$ | **Comentaris** |
| 1 | 0 | 0 | $Q$ | $\overline{Q}$ | Les sortides no canvien. Es queden en l'estat anterior |
| 1 | 1 | 0 | 1 | 0 | Estat de SET |
| 1 | 0 | 1 | 0 | 1 | Estat de RESET |
| 1 | 1 | 1 | ? | ? | **Condició no vàlida** |
| 0 | X | X | $Q$ | $\overline{Q}$ | Latch no habilitat. Les sortides no canvien |

En la figura 2.17 podem veure les formes d'ona de les entrades i la sortida d'un *latch* S-R amb habilitació. Se suposa que $Q$ es troba inicialment a nivell baix.

**Figura 2.17.** Formes d'ona de les entrades i la sortida d'un latch S-R amb habilitació



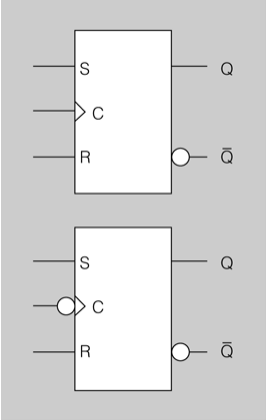

### Flip-flop S-R

En la taula 2.11 es mostra la taula de veritat d'un *flip-flop* S-R disparat per flanc positiu. Com es pot veure, el seu funcionament és similar al del *latch* S-R. L'única diferència és que les sortides només canvien en l'instant en què el senyal **CLK** passa de nivell baix a nivell alt.

**TAULA 2.11.** Taula de veritat d'un flip-flop S-R disparat per flanc positiu

| Entrades | | | Sortides | | |
|---|---|---|---|---|---|
| **CLK** | **S** | **R** | $Q+$ | $\overline{Q+}$ | **Comentaris** |
| X | X | X | $Q$ | $\overline{Q}$ | No hi ha flanc de CLK: les sortides no canvien |
| ↑ | 0 | 0 | $Q$ | $\overline{Q}$ | Les sortides no canvien |
| ↑ | 1 | 0 | 1 | 0 | Estat de SET |
| ↑ | 0 | 1 | 0 | 1 | Estat de RESET |
| ↑ | 1 | 1 | ? | ? | Condició no vàlida |

La fletxa cap amunt significa flanc de pujada

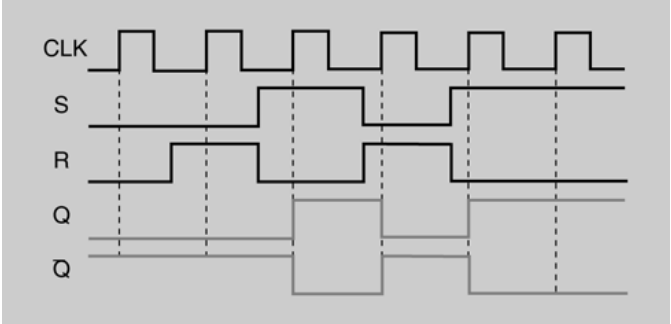



Flip-flops S-R disparats per flanc positiu (superior) i per flanc negatiu (inferior)

Com en el cas del *latch* S-R, mai no hem de posar les entrades S i R del *flip-flop* S-R simultàniament a nivell alt, perquè no podem predir quin valor tindran les seves sortides. Aquesta imprecisió del valor de les sortides podria provocar problemes greus al circuit on estigués connectat el *flip-flop* S-R.

En la figura 2.18 podem veure les formes d'ona de les entrades i la sortida d'un *flip-flop* S-R disparat per flanc positiu. Se suposa que $Q$ es troba inicialment a nivell baix.

**FIGURA 2.18.** Formes d'ona de les entrades i la sortida d'un flip-flop S-R disparat per flanc positiu




## Flip-flop J-K

El *flip-flop* J-K és un dels tipus de *flip-flop* més utilitzats.

**Les denominacions J i K...**

... per a les entrades del *flip-flop* J-K no tenen cap significat conegut, excepte el fet que són dues lletres consecutives de l'alfabet.

El funcionament del *flip-flop* J-K és idèntic al del *flip-flop* S-R en els estats d'operació *SET*, *RESET* i no-canvi. La diferència resideix en el fet que el *flip-flop* J-K **no té condicions no vàlides** com passa amb el *flip-flop* S-R (ara sí hi pot haver dos uns a les entrades).

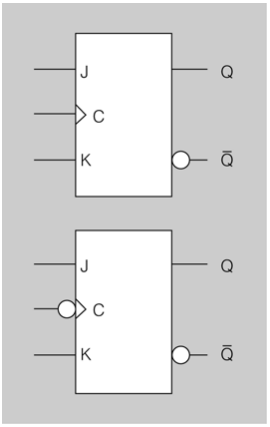
**Taula 2.12.** Taula de veritat d'un flip-flop J-K disparat per flanc positiu

| Entrades | | | Sortides | | |
|---|---|---|---|---|---|
| **CLK** | **J** | **K** | $Q+$ | $\overline{Q+}$ | **Comentaris** |
| X | X | X | $Q$ | $\overline{Q}$ | No hi ha flanc de CLK: les sortides no canvien |
| ↑ | 0 | 0 | $Q$ | $\overline{Q}$ | Les sortides no canvien. |
| ↑ | 1 | 0 | 1 | 0 | Estat de SET |
| ↑ | 0 | 1 | 0 | 1 | Estat de RESET |
| ↑ | 1 | 1 | $\overline{Q}$ | $Q$ | **Basculació**: la sortida canvia al seu estat oposat |

Quan la sortida bascula, vol dir que si hi havia un 1 passa a haver-hi un 0, i a l'inrevés

Com podem observar en la taula de veritat del *flip-flop* J-K (taula 2.12), ara ja no hi ha combinacions no vàlides de les entrades, com passava al *flip-flop* S-R. Quan les dues entrades J i K es posen a nivell alt, la sortida canvia al seu estat oposat. És a dir, ens podem trobar amb dues d'aquestes situacions:

- Si la sortida estava a nivell alt abans del flanc de rellotge, passarà a nivell baix.
- Si la sortida estava a nivell baix abans del flanc de rellotge, passarà a nivell alt.

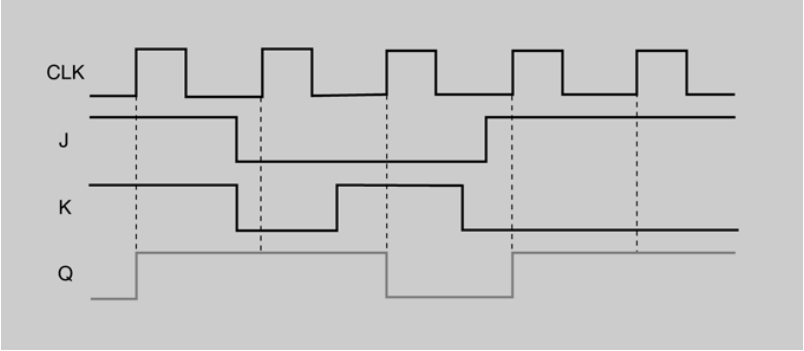



**Circuit integrat 74HC112**

El circuit integrat comercial 74HC112 és un petit xip de setze pius que conté dos *flip-flops* J-K disparats per flanc descendent.

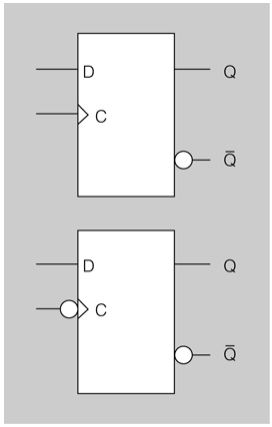
En la figura 2.19 podem veure les formes d'ona de les entrades i la sortida d'un *flip-flop* J-K disparat per flanc positiu. Se suposa que Q es troba inicialment a nivell baix.

**Figura 2.19.** Formes d'ona de les entrades i la sortida d'un flip-flop J-K disparat per flanc positiu



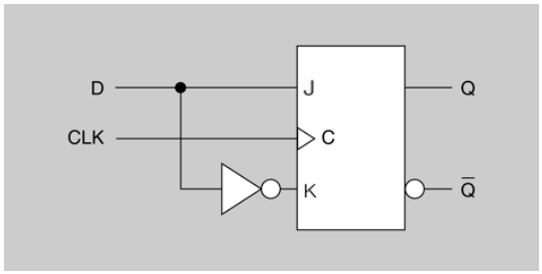

## Flip-flops D i T

El *flip-flop* D està construït a partir d'un *flip-flop* J-K al qual s'han unit les seves entrades J i K mitjançant un inversor, tal com es mostra en la figura 2.20. La taula de veritat d'aquest *flip-flop* D és molt senzilla (taula 2.13).




Flip-flop D disparat per flanc positiu (superior) i per flanc negatiu (inferior)

**FIGURA 2.20.** Flip-flop D construït a partir d'un flip-flop S-R




**TAULA 2.13.** Taula de veritat d'un flip-flop D activat per flanc positiu

| Entrades | | Sortides | | |
|---|---|---|---|---|
| **D** | **CLK** | $Q+$ | $\overline{Q+}$ | **Comentaris** |
| 0 | ↑ | 0 | 1 | Estat de RESET |
| 1 | ↑ | 1 | 0 | Estat de SET |

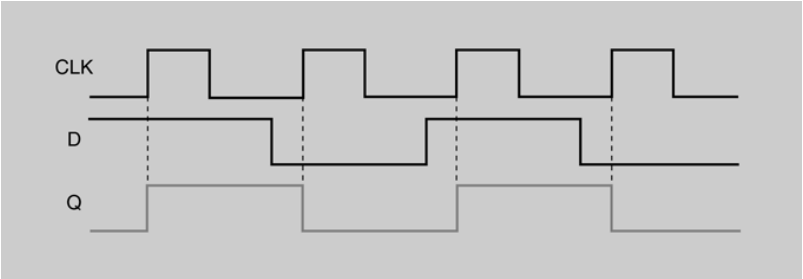
Com veiem en la taula 2.13, la sortida $Q$ conserva el valor que té l'entrada D just en l'instant en el qual es produeix la transició positiva del rellotge. Aquesta propietat fa que el *flip-flop* D s'utilitzi molt com a element de memòria.

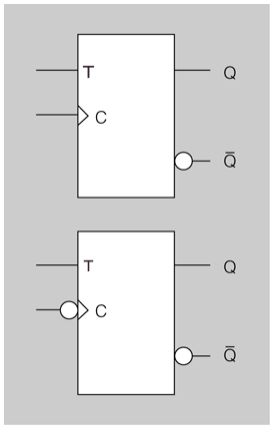
**Flanc positiu o flanc negatiu?**

Quan un *flip-flop* es dispara per flanc negatiu se simbolitza amb un petit cercle a l'entrada del rellotge. Si el *flip-flop* es dispara per flanc positiu el petit cercle no apareix en el seu símbol.

En la figura 2.21 podem veure les formes d'ona de les entrades i la sortida d'un *flip-flop* D disparat per flanc positiu. Se suposa que $Q$ es troba inicialment a nivell baix.

**FIGURA 2.21.** Formes d'ona de les entrades i la sortida d'un flip-flop D disparat per flanc positiu.

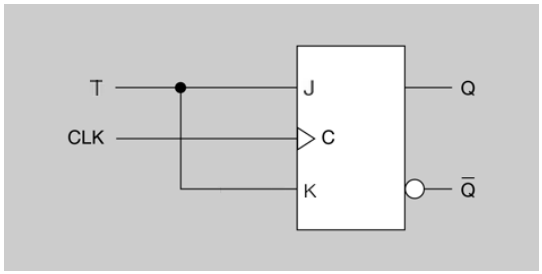



Flip-flop T disparat per flanc positiu (superior) i per flanc negatiu (inferior)

El *flip-flop* T està construït a partir d'un *flip-flop* J-K al qual s'han unit les seves entrades J i K directament, tal com es mostra en la figura 2.22. La taula de veritat d'aquest *flip-flop* T és molt senzilla (taula 2.14).

**FIGURA 2.22.** Flip-flop T construït a partir d'un flip-flop S-R



**Taula 2.14.** Taula de veritat d'un flip-flop T activat per flanc positiu

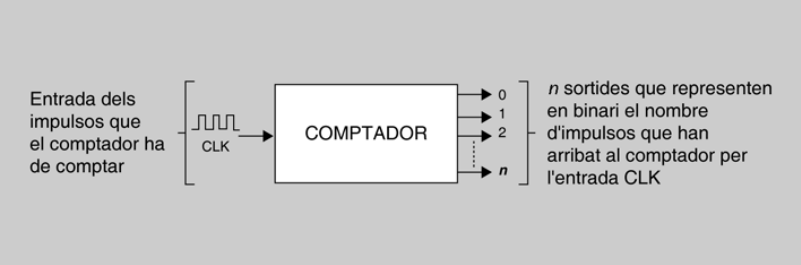
| Entrades | | Sortides | | |
|---|---|---|---|---|
| **T** | **CLK** | $Q+$ | $\overline{Q+}$ | **Comentaris** |
| 0 | ↑ | $Q$ | $\overline{Q}$ | No hi ha canvi d'estat |
| 1 | ↑ | $\overline{Q}$ | $Q$ | Canvi d'estat forçat |

Com veiem en la taula 2.14, la sortida $Q$ canvia el valor que tenia **si hi ha un 1 a l'entrada T** just en l'instant en el qual es produeix la transició positiva del rellotge. Aquesta propietat fa que el *flip-flop* T s'utilitzi molt com a element comptador o com a complement d el rellotge.

### 2.4.2 Comptadors

Un comptador (figura 2.23) és un circuit format per *flip-flops* T, que té una entrada d'impulsos (anomenada **entrada de rellotge** o **CLK**) i un nombre de sortides $n$ que representen **en codi binari**, en cada moment, el nombre d'impulsos que li arriben a l'entrada de rellotge.

**Figura 2.23.** Diagrama de blocs d'un comptador digital




La importància dels comptadors dins del món dels sistemes digitals és molt gran. Només cal pensar en el gran nombre d'aplicacions que poden tenir: detecció del nombre de vehicles que entren en un pàrking, recompte de les peces elaborades en una fàbrica, realització de funcions de cronometratge, etc.


Un cronòmetre és un exemple molt clar de comptador digital

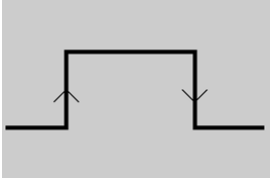
Els comptadors digitals tenen les característiques següents:

- **Mòdul**: és el nombre de combinacions diferents que té un comptador a la seva sortida. Per exemple, si un comptador té 4 bits de sortida i compta des de 0000 fins a 1111, es diu que és de mòdul 16 o, de forma abreujada, és diu que és *MOD16*.
- **Compte ascendent o descendent**: gairebé tots els comptadors digitals tenen un bit d'entrada per seleccionar si el compte es fa de manera ascendent o descendent.
- **Operació asíncrona o síncrona**: als comptadors asíncrons, els *flip-flops* no comparteixen el mateix senyal de rellotge. Als comptadors síncrons, tots els

seus *flip-flops* comparteixen el mateix senyal de rellotge. Els més utilitzats són els comptadors síncrons.

- **Entrada d'esborrat**: molts comptadors disposen d'un bit d'entrada que, quan es posa a zero, fa que totes les sortides es posin a nivell baix (*RESET*).
- **Entrada amb flanc ascendent o descendent**: els comptadors amb flanc ascendent canvien la seva sortida quan en el senyal de rellotge hi ha una transició de baix a alt. Els comptadors amb flanc descendent canvien la seva sortida quan en el senyal de rellotge hi ha una transició d'alt a baix. En la figura 2.24 es mostra un pols de rellotge amb els seus flancs ascendent i descendent.

**Figura 2.24.** Pols de rellotge



Els comptadors més utilitzats són els de quatre bits de sortida

Els comptadors de quatre bits de sortida poden ser:

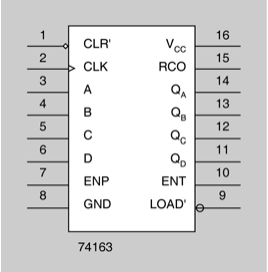
- De mòdul 16 (MOD16): compten des de 0000b fins a 1111b (és a dir, de 0 a 15 en decimal, o de 0 a F en hexadecimal).
- De mòdul 10 (MOD10): compten des de 0000b fins a 1001b (és a dir, de 0 a 9 en decimal). Quan les sortides d'aquests comptadors estan a 1001b i hi ha un nou pols de rellotge a l'entrada, les sortides tornen a 0000b.

**Els comptadors dels PLC**

La majoria de comptadors que porten incorporats els autòmats programables compten fins a valors superiors a 30.000 impulsos.

Per comptar nombres més grans de 16 s'utilitzen diversos comptadors de quatre bits connectats entre ells.

### Comptador binari síncron de quatre bits 74163




74163: comptador binari síncron de 4 bits

El xip 74163 és un exemple de circuit integrat comptador binari de quatre bits.

La utilitat de cada piu d'aquest circuit comptador es descriu a continuació:

- **$Q_A$**, **$Q_B$**, **$Q_C$** i **$Q_D$** (pius núms. 14, 13, 12 i 11, respectivament): **sortides** del comptador. $Q_A$ correspon al bit de menys pes.
- **CLK** (piu 2): **entrada** de rellotge actiu per flanc ascendent.
- **VCC** (piu 16): alimentació del circuit integrat (+5 V).
- **GND** (piu 8): connexió a massa del circuit integrat.

- **A**, **B**, **C** i **D** (pius núm. 3, 4, 5 i 6, respectivament): a aquest comptador li podem indicar per quin nombre ha de començar a comptar. Normalment serà el 0000, però amb les **entrades** A, B, C i D li podem indicar un altra combinació d'inici (A correspon al bit de menys pes)

- $\overline{\text{LOAD}}$ (piu núm. 9): quan s'aplica un nivell baix a l'**entrada** $\overline{\text{LOAD}}$,el comptador assumeix l'estat de les entrades A, B, C i D en el següent flanc ascendent de rellotge.

- $\overline{\text{CLR}}$ (piu núm. 1): **entrada** d'esborrat activa a nivell baix que posa a zero de manera síncrona (és a dir, en el següent flanc ascendent de rellotge) les quatre sortides del comptador.

- **ENP** i **ENT** (pius núm. 7 i 10, respectivament): **entrades** d'habilitació. Aquestes entrades han d'estar a nivell alt perquè el comptador pugui comptar. Quan almenys una de les dues entrades és a nivell baix, el comptador es desactiva.

- **RCO** (piu núm. 15): aquesta **sortida** es posa a nivell alt quan el comptador assoleix el valor 1111 a la seva sortida.

### 2.4.3 Registres de desplaçament

**Aplicacions dels registres de desplaçament**

Les principals aplicacions dels registres de desplaçament són:

- emmagatzematge de dades,
- conversió de dades paral·lel/ sèrie,
- conversió de dades sèrie/ paral·lel,
- introducció de retards en els sistemes de comunicació.

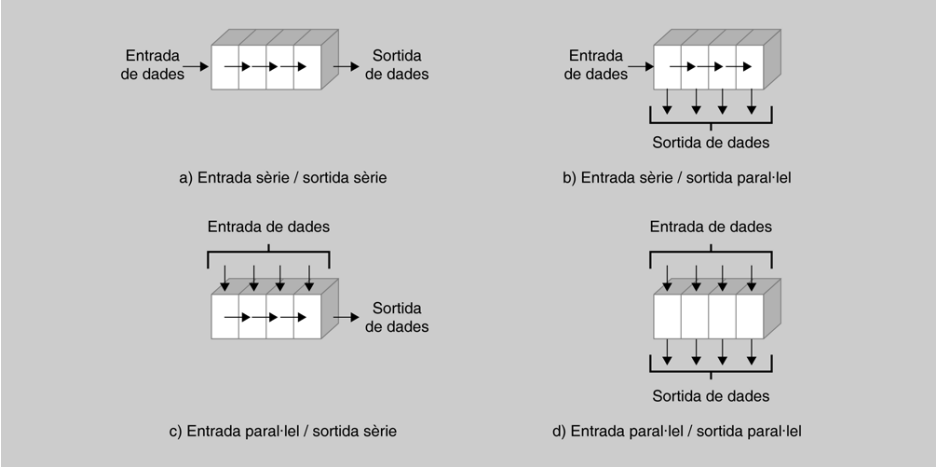
Els registres de desplaçament són una altra aplicació digital seqüencial que, com els comptadors, es poden obtenir mitjançant la connexió de diversos biestables o utilitzant circuits integrats específics.

Els registres de desplaçament són sistemes digitals seqüencials formats per biestables, que s'utilitzen per emmagatzemar i transferir dades digitals.

Els registres de desplaçament es poden classificar segons com s'introdueix la informació i de com s'obté aquesta informació. D'acord amb aquest criteri, podem trobar (figura 2.25):

- registres de desplaçament entrada sèrie/sortida sèrie
- registres de desplaçament entrada sèrie/sortida paral·lel
- registres de desplaçament entrada paral·lel/sortida sèrie
- registres de desplaçament entrada paral·lel/sortida paral·lel.

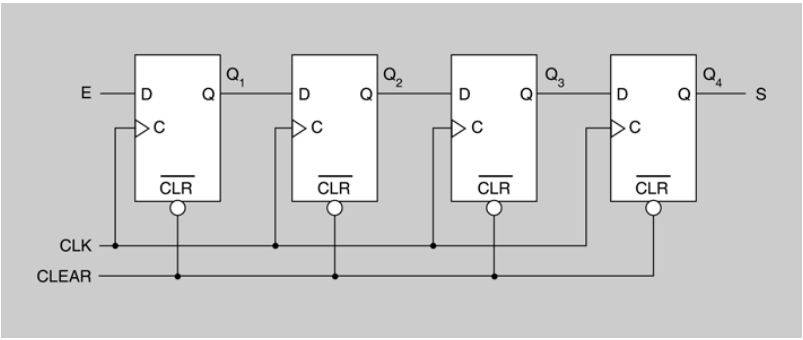
**Figura 2.25.** Registres de desplaçament de quatre bits




### Registre de desplaçament amb entrada sèrie/sortida sèrie

En la figura 2.26 es mostra l'estructura d'un registre de desplaçament entrada sèrie/sortida sèrie de quatre bits, format per quatre *flip-flops* D amb senyal d'esborrat (CLR).

**Figura 2.26.** Registre de desplaçament entrada sèrie-sortida sèrie de quatre bits

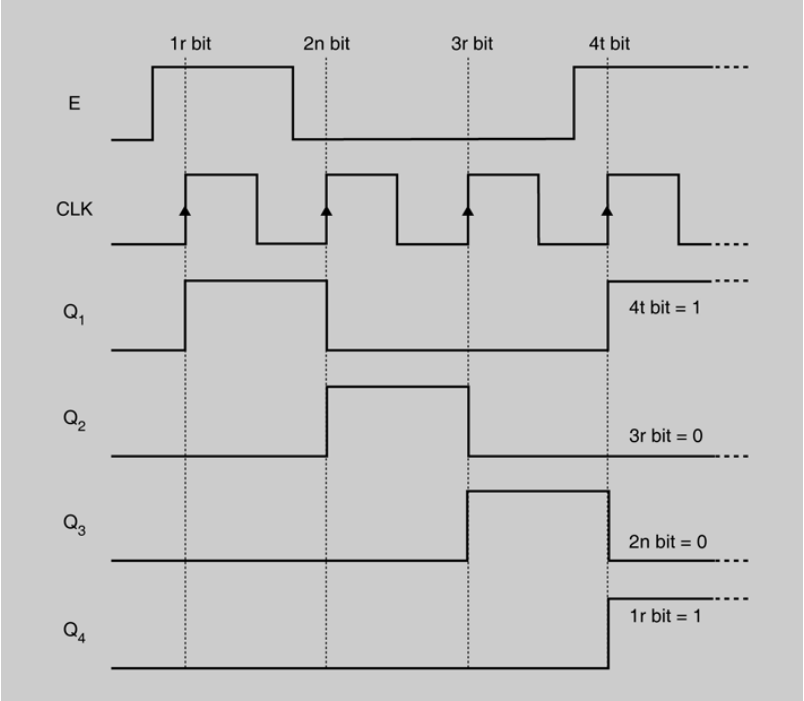



**Estructura interna dels registres de desplaçament**

La majoria dels registres de desplaçament es realitzen a partir de *flip-flops* tipus D.

El funcionament d'aquest registre de desplaçament és ben senzill. Per l'entrada **E** es van introduint les dades. Cada bit dels introduïts serà desplaçat al següent *flip-flop* a cada flanc ascendent del senyal de rellotge, de manera que a partir del quart flanc ascendent la informació introduïda s'obtindrà a la sortida de manera successiva. En la figura 2.27 es mostren els senyals E, CLK, Q1, Q2, Q3 i Q4 quan s'emmagatzema el codi 1001 en un registre de desplaçament sèrie/sèrie.

**Figura 2.27.** Formes d'ona del registre de desplaçament sèrie-sèrie de quatre bits

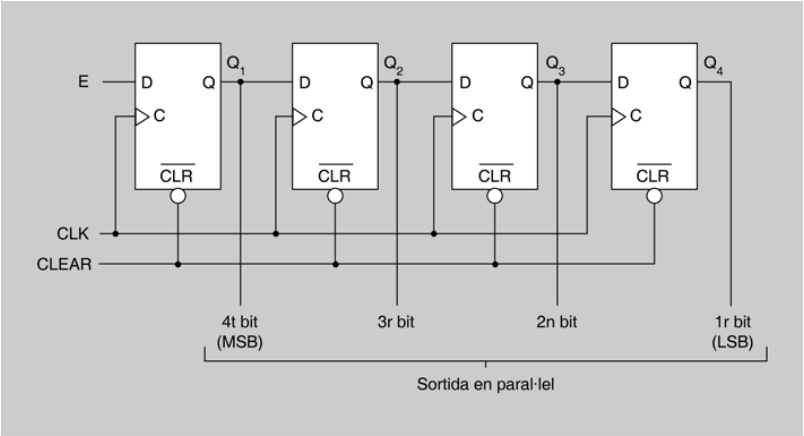



### Registre de desplaçament amb entrada sèrie/sortida paral·lel

Es pot obtenir un registre de desplaçament d'entrada sèrie/sortida paral·lel partint de l'esquema del registre sèrie/sèrie. Només caldrà connectar les sortides paral·lel als punts Q1, Q2, Q3 i Q4 com es mostra en la figura 2.28. El funcionament d'aquest registre és el següent:

- Quan es produeix un flanc ascendent de CLK entra un bit per la línia E i es queda a Q1. En el mateix instant, el bit que era al punt Q1 passa a Q2, el que era a Q2 passa a Q3 i el que era a Q3 passa a Q4.
- La informació introduïda per l'entrada sèrie estarà disponible a les sortides Q1, Q2, Q3 i Q4 al flanc ascendent del quart pols de rellotge.

**Figura 2.28.** Registre de desplaçament entrada sèrie-sortida paral·lel de quatre bits

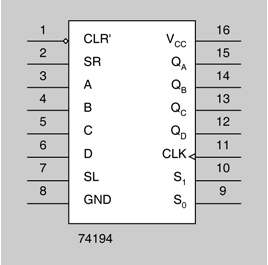



**MSB i LSB**

MSB vol dir ***M**ost **S**ignificant **B**it*, és a dir, el bit més significant, el de més pes.

LSB vol dir ***L**east **S**ignificant **B**it*, és a dir, el bit menys significant, el de menys pes.

### Registre de desplaçament universal




74194: registre de desplaçament universal

Un registre de desplaçament universal és un circuit integrat que incorpora els diferents modes de funcionament de registre, amb entrades i sortides sèrie i paral·lel.

Un exemple clar de registre de desplaçament universal és el circuit integrat 74194.

La utilitat de cada piu d'aquest registre de desplaçament universal es descriu a continuació:

- $\overline{\text{CLR}}$ (piu núm. 1): **entrada** asíncrona d'esborrat, activa a nivell baix.
- $\mathbf{S_0}$ i $\mathbf{S_1}$ (pius núm. 9 i 10, respectivament): **entrades** de selecció del mode d'operació. Aquestes entrades actuen segons la taula 2.15.
- **CLK** (piu núm. 11): **entrada** de rellotge actiu per flanc ascendent.
- **SR** (piu núm. 2): **entrada** sèrie per fer desplaçaments a la dreta.
- **SL** (piu núm. 7): **entrada** sèrie per fer desplaçaments a l'esquerra.
- **A**, **B**, **C** i **D** (pius núm. 3, 4, 5 i 6, respectivament): **entrades** per fer càrregues en paral·lel.
- $\mathbf{Q_A}$, $\mathbf{Q_B}$, $\mathbf{Q_C}$ i $\mathbf{Q_D}$ (pius núm. 15, 14, 13 i 12, respectivament): **sortides** en paral·lel.
- **VCC** (piu 16): alimentació del circuit integrat (+5 V).
- **GND** (piu 8): connexió a massa del circuit integrat.

**Taula 2.15.** Modes d'operació del registre universal 74HC194

| $S_0$ | $S_1$ | Mode d'operació |
|---|---|---|
| 0 | 0 | Rellotge inhibit: no hi ha desplaçament |
| 0 | 1 | Desplaçament a l'esquerra |
| 1 | 0 | Desplaçament a la dreta |
| 1 | 1 | Càrrega en paral·lel de les dades a les sortides |

La càrrega en paral·lel es fa de manera síncrona, aplicant les dades a les entrades en paral·lel i posant les entrades de control de mode $S_0$ i $S_1$ a nivell alt. A partir d'aquest moment, les dades seran transferides a la sortida en el flanc ascendent del pols de rellotge; durant la càrrega en paral·lel, el flux de dades per l'entrada sèrie queda inhibit.