El maquinari: arquitectura i components

José López Vicario
Antoni Morell Pérez

PID_00177251

Universitat Oberta de Catalunya
www.uoc.edu

Índex

Introducció

En aquest mòdul estudiarem els diferents elements de maquinari que conformen els sistemes encastats, partint de la hipòtesi que els estudiants teniu coneixements bàsics tant dels circuits lògics combinacionals com dels seqüencials.

Si classifiquem els diferents components des d'un punt de vista tecnològic, veurem que actualment hi ha al mercat tota una sèrie de dispositius que podem ordenar segons el grau de complexitat. Al capdamunt de la llista, hi trobaríem els circuits integrats a molt gran escala (VLSI, *very large-scale integrated*), els quals poden arribar a integrar milions de transistors en un sol xip. Els circuits VLSI aporten un grau significatiu de funcionalitat i és on trobem els microprocessadors, microcontroladors, matrius de portes programables *in situ* (FPGA, *field programmable gate arrays*), dispositius lògics programables (PLD, *programmable logic devices*), circuits integrats d'aplicació específica (ASIC, *application specific integrated circuits*) i, fins i tot, les memòries. En un nivell per sota d'aquests trobem els circuits integrats a escala mitjana (MSI, *medium-scale integrated*), els quals contenen prop d'un miler de transistors i executen tasques més senzilles que els LSI, per bé que la seva funcionalitat continua sent completa. Finalment, trobem els circuits integrats a petita escala (SSI, *small-scale integrated*) i els component simples. Entre tots ells, s'estableix comunicació mitjançant els senyals elèctrics, els quals poden estar corromputs per soroll o bé per senyals no volguts. A continuació, es desenvolupa una visió del maquinari present en els sistemes encastats seguint aquesta classificació.

En primer lloc, es descriu l'arquitectura bàsica de qualsevol sistema encastat amb els seus blocs principals, és a dir, processador (CPU, unitat central de processament), memòria, entrades i sortides. Inicialment, se centra l'atenció en el nucli del sistema i se'n fa una descripció d'alt nivell. Es continua amb l'estudi dels diferents tipus de memòria existents i quin n'és l'ús. En aquest punt, el discurs té en compte alguns aspectes de nivell més baix. Finalment, parlarem d'altres dispositius menys complexos però de gran transcendència en els sistemes encastats com són els registres, els rellotges, els comptadors i els divisors. En aquest punt, tindrem en compte els diferents efectes que pateixen els senyals elèctrics per tal de tenir-los en compte durant la fase de disseny.

Aquest mòdul finalitza amb l'estudi d'altres parts del maquinari, algunes de les quals són sempre presents i fins i tot poden arribar a ser crítiques, com el subsistema d'alimentació o bé la detecció d'errors en el maquinari. En canvi, altres parts, com ara la comunicació amb l'exterior per diferents ports o bé per comunicacions de ràdio, tindran més o menys importància en funció de l'aplicació específica. Per exemple, en una màquina expenedora només s'utilitzaran per a tasques de manteniment (per exemple, actualitzar preus) i,

en canvi, en una xarxa de sensors sense fil, les comunicacions de ràdio són part indispensable del sistema. En aquest últim cas veiem la importància del subsistema d'alimentació, ja que voldrem sensors amb gran autonomia. En alguns casos, fins i tot, el sensor s'abandona un cop exhaurida la bateria.

Objectius

L'estudi d'aquests mòdul didàctic us ajudarà a assolir els objectius següents:

1. Conèixer els components de maquinari principals que formen un sistema encastat.
2. Conèixer els diferents subsistemes de l'arquitectura d'un sistema encastat.
3. Conèixer els diferents mecanismes de comunicació en un sistema encastat.

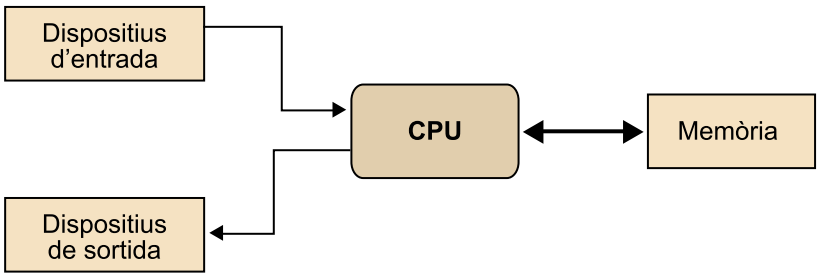
1. Arquitectura del sistema

L'arquitectura bàsica d'un sistema encastat no difereix massa de l'arquitectura de qualsevol computador, en què figuren quatre elements bàsics:

- el nucli o processador (CPU),
- la memòria principal,
- l'entrada del sistema i
- la sortida del sistema,

tal com s'aprecia en la figura següent:

Arquitectura bàsica d'un sistema encastat



En el moment d'execució, la CPU recull de la memòria les instruccions que ha de dur a terme (el programa) i, juntament amb altres dades, que poden ser obtingudes tant de la mateixa memòria del sistema com de les seves entrades, executa les tasques indicades i en retorna el resultat a la memòria o bé a la interfície de sortida.

La CPU és, doncs, la part principal del sistema i tot el disseny i funcionament del sistema encastat gira entorn d'aquesta part.

2. Components dels sistemes encastats

En aquest apartat veurem els components principals que integren un sistema encastat.

2.1. Nucli del sistema

Hi ha quatre opcions principals a l'hora d'escollir la CPU, cadascuna de les quals imposa petites variacions en l'arquitectura del sistema, i que són:

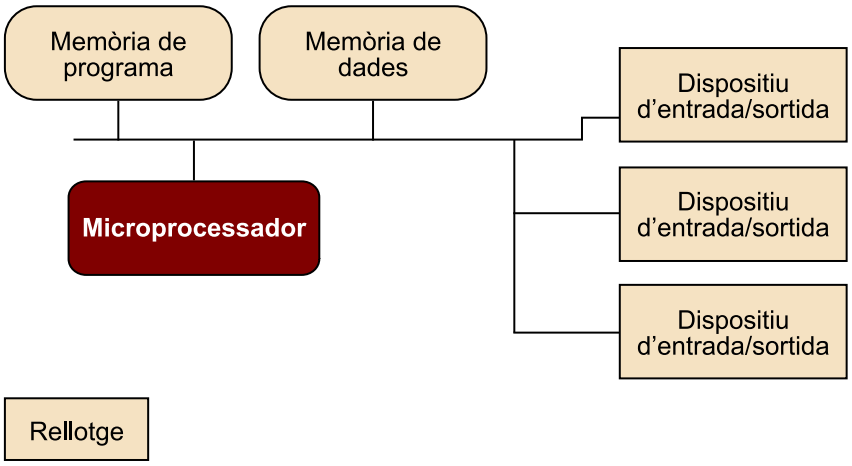
- microprocessador,
- microcomputador,
- microcontrolador i
- processador de senyals digitals (DSP, *digital signal processor*).

2.1.1. El microprocessador

El microprocessador és un dispositiu que integra en un sol xip les parts indispensables de la CPU, és a dir, els registres interns, la unitat aritmeticològica (ALU, *arithmetical and logical unit*) i la unitat de control. Aquesta darrera s'encarrega de gestionar l'execució del programa, és a dir, agafar les instruccions de la memòria, interpretar-les, recollir les dades necessàries per a dur-les a terme, executar-les, lliurar-ne els resultats i passar a la instrucció següent. L'ALU, tal com el seu nom indica, s'ocupa exclusivament de fer els càlculs aritmètics i resoldre les relacions lògiques. Finalment, els registres interns són memòries d'alta velocitat i mida reduïda en què es guarden, d'una manera temporal, les dades que s'utilitzen habitualment durant l'execució, ja sigui perquè és necessari o bé amb l'objectiu d'afavorir la velocitat del sistema.

Quan un sistema encastat està basat en microprocessador, la seva arquitectura típica és la de la figura següent, en què s'aprecien diversos elements externs: la **memòria**, que habitualment es divideix en memòria de dades i de programa (microprogramari, *firmware*), els **dispositius d'entrada/sortida** i el **rellotge** del sistema. La memòria de dades acostuma a ser memòria RAM, mentre que el microprogramari sol ser constituït per memòria ROM i és on es guarda el conjunt d'instruccions que ha d'executar el sistema. En general, l'arquitectura del sistema serà de tipus Von Neumann o Harvard en funció de si es permet l'accés simultani a les dues memòria o no. Per tant, en una arquitectura Harvard seran necessaris dos busos d'informació separats.

Arquitectura d'un sistema encastat basat en un microprocessador



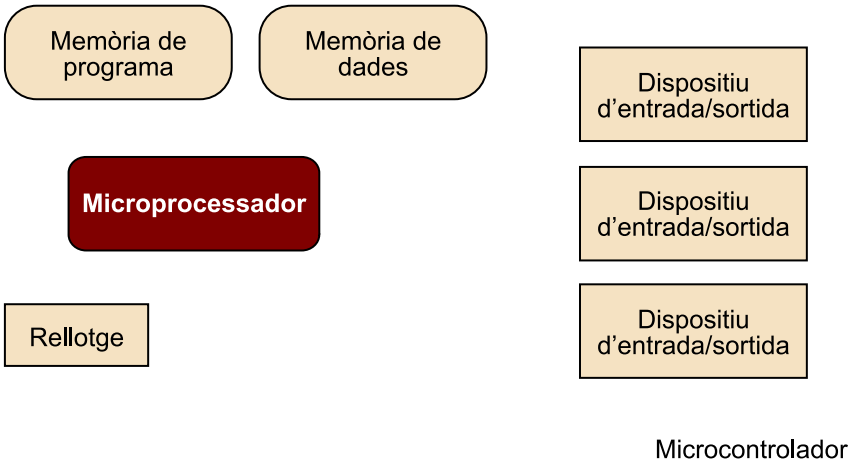
2.1.2. El microcomputador

El microcomputador és una versió més *completa* de microprocessador en el sentit que integra elements indispensables en qualsevol sistema encastat i que en el cas del microprocessador són externs a aquest. Així, doncs, tenim en un sol xip el microprocessador, una certa quantitat de memòria i alguns circuits d'entrada-sortida en els quals connectar-hi els perifèrics.

2.1.3. El microcontrolador

Finalment, l'últim nivell d'integració és representat pels microcontroladors, dispositius que poden arribar a incloure en un únic xip totes les funcionalitats necessàries. Entre d'altres, poden incloure rellotge propi, dispositius d'entrada, convertidors d'analògic a digital i viceversa, ports de comunicació, etc. En l'àmbit dels sistemes encastats, els microcontroladors són especialment útils quan volem obtenir sistemes de baix cost, especialment si s'han de produir a gran escala. En aquest cas, l'arquitectura del sistema és la que apareix en la figura següent:

Arquitectura d'un sistema encastat basat en un microprocessador i amb microcontrolador



2.1.4. El processador de senyals digitals (DSP)

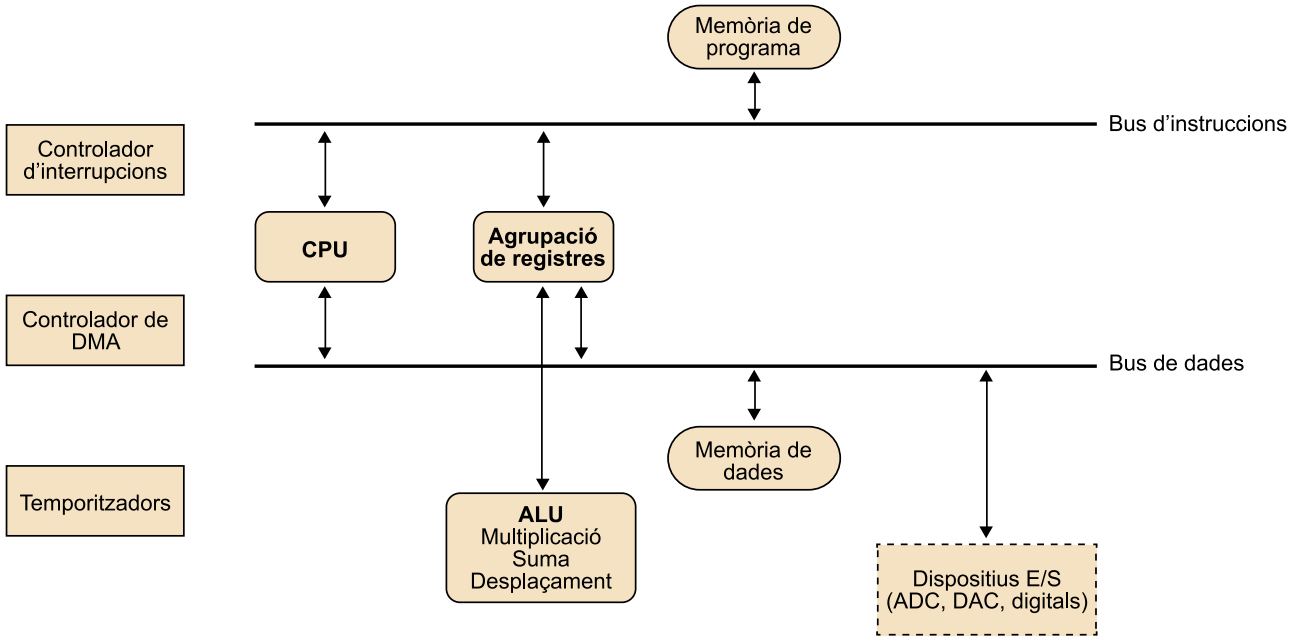
A més dels dispositius de propòsit general que hem comentat fins al moment, cal destacar el DSP com un microprocessador de caràcter específic dissenyat per a oferir un alt rendiment en les aplicacions que utilitzen processament del senyal, com poden ser el tractament de veu, música, vídeo, imatges, etc.

La diferència bàsica entre un DSP i la resta de processadors és la capacitat per a executar operacions aritmètiques com la suma, la multiplicació, el desplaçament de bits o bé la multiplicació-acumulació en un sol salt de rellotge. Aquest fet permet, doncs, fer càlculs matricials o convolucions amb pocs salts. A més a més, acostumen a utilitzar aritmètica de saturació, és a dir, es limiten els resultats de les operacions entre un valor màxim i mínim en comptes de permetre el pas del valor màxim al mínim com succeeix en els comptaquilòmetres analògics dels automòbils.

A causa de la seva especialització, poden ser usats en solitari o també juntament amb un processador de caràcter general que encarrega al DSP les tasques específiques de processament del senyal. Atès que els DSP treballen moltes vegades amb senyals del món real, i per tant analògics, és comú equipar-los amb convertidors analògic-digitals (ADC) i digital-analògics (DAC), i també amb dispositius digitals d'entrada/sortida d'alta velocitat.

En la figura següent, es pot apreciar el diagrama de blocs d'un DSP, on s'aprecia una arquitectura de tipus Harvard amb accés independent a dades i a instruccions, la qual s'utilitza aplicacions d'alta velocitat juntament amb busos de dades d'amplada de banda gran (poden transportar una gran quantitat de bits per segon).

Arquitectura d'un sistema encastat basat en DSP



2.2. Busos de comunicacions (adreces, dades i control)

Tal com s'ha pogut entreveure en els diagrames anteriors, qualsevol sistema encastat, sigui quina sigui la seva arquitectura, necessita establir comunicacions entre les diferents parts que el formen.

Anomenen *busos* aquests canals de comunicació i habitualment es transporta aquesta informació en forma de senyals elèctrics. En la seva forma més simple, podem entendre el bus com un conjunt de "fils" per on viatja la informació i normalment un sol bus interconnecta múltiples dispositius.

Exemple

Un valor de +5 V acostuma a representar un "1" lògic, mentre que un valor de 0 V representa el "0" lògic.

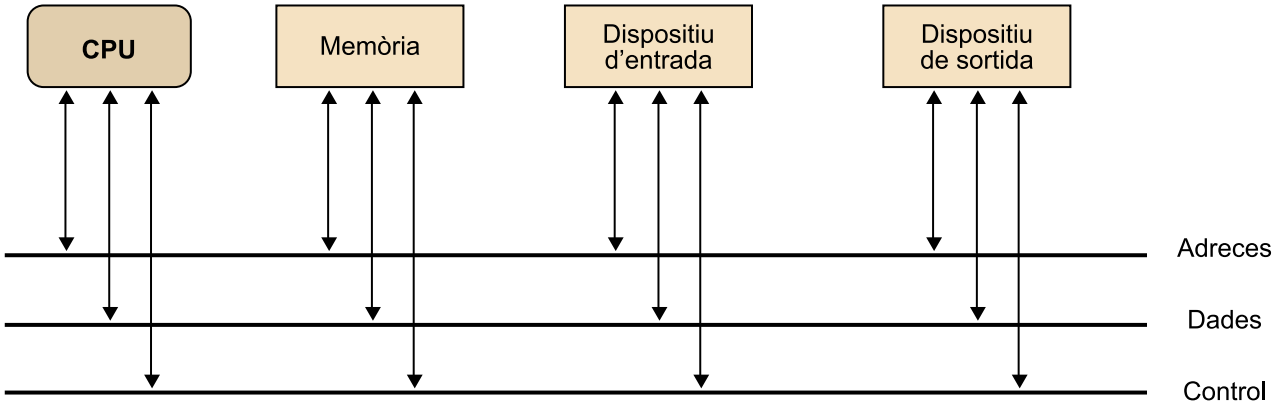
La informació que hi viatja pot ser de tres tipus: **adreces**, **dades** i **control**. En primer lloc, l'adreça ens indica on cal guardar la informació o d'on s'ha d'extreure, ja sigui de la memòria del sistema o dels dispositius entrada-sortida. En segon lloc, les dades contenen la informació que estem processant. Finalment, en el bus de control trobem senyals que són necessaris per al funcionament correcte dels dispositius que formen el sistema. Per exemple, quan accedim a la memòria del sistema per a recollir unes dades determinades, a part de saber-ne l'adreça, cal saber en quin moment estan preparades per a ser llegides. En el món digital, és molt important remarcar que tot allò que no té adreça no existeix, ja que no hi ha manera d'accedir-hi. Pensem en quan esborrem un fitxer del disc dur del nostre PC. La informació (dades) no s'esborra físicament, però sí que deixem de recordar-ne la ubicació (adreça) i, per tant, no podem veure la informació encara que hi romangui.

Exemple

En una màquina expenedora de begudes ens pot interessar saber el nombre d'unitats que resten d'un article determinat, el qual hem d'actualitzar en el moment de fer la reposició i anar decreixent cada cop que una beguda és dispensada. Aquesta dada, que estarà ubicada en una adreça determinada (posició de memòria), és la que viatjarà pel bus de dades.

La figura següent mostra els diferents elements d'un sistema encastat interconnectats mitjançant els busos d'adreces, dades i control. En aquest cas, el bus d'adreces transporta 12 senyals, el bus de dades 16 i el bus de control 4:

Els busos d'adreces, dades i control



Definim l'amplària del bus com el nombre de bits que pot transportar simultàniament i ens dóna una idea aproximada de la velocitat del bus. És a dir, si hem de transferir 64 bits sobre un bus d'amplària de 16 bits, ens caldran quatre instants temporals per a completar la transmissió, mentre que si el bus té una amplària de 64 bits, amb un sol instant en tindrem prou. Així, doncs, queda

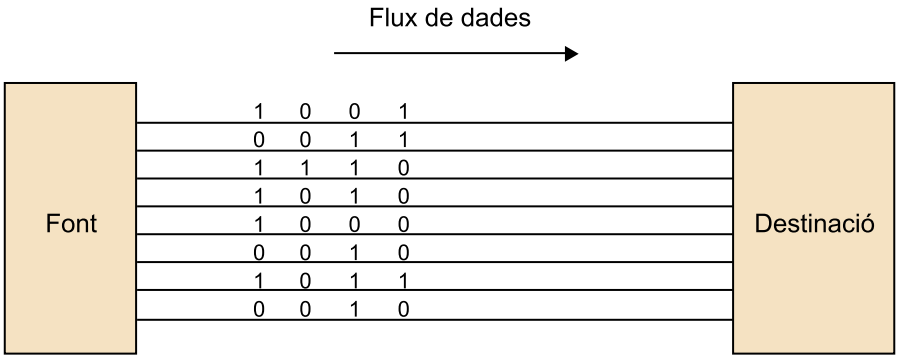
Exemple

Un bus que opera a 800 MHz i té una amplària de 32 bits és capaç de transportar dades a una velocitat de 25,6 GBps.

clar que la velocitat del bus en bits per segon serà determinada pel producte de la seva amplària i la freqüència màxima de funcionament, mesurada en cicles per segon.

En la figura següent, podem veure el moviment de dades entre font i destinació. En aquest cas, calen vuit instants temporals per a transmetre 32 bits. En el cas particular que l'amplària del bus sigui 1, direm que es tracta d'un **bus sèrie**, mentre que en la resta de casos ens hi referirem com a **bus paral·lel**.

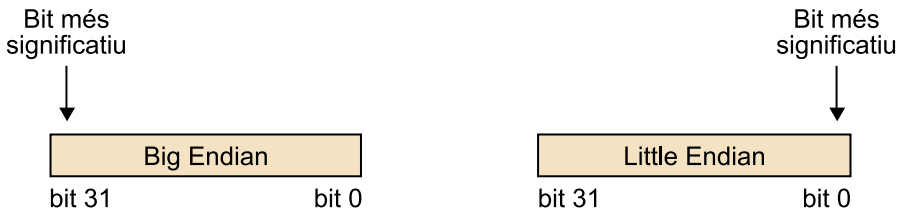
Moviment d'una paraula de 32 bits en un bus de 8 bits



Atès que el sistema treballarà amb paraules de *N* bits per tal de descriure tant les dades com les adreces, cal establir un ordre en els bits que rebem o enviem per mitjà dels busos. En cas que la paraula ocupi completament el bus, cal establir un ordre espacial entre les línies que formen el bus. Si al contrari, una paraula requereix diverses transmissions per a ser enviada (és segur el cas dels busos sèrie), ens cal establir un ordre espaciotemporal. Qualsevol ordenació que es dugui a terme és una simple qüestió de conveni i, en general, s'usa la notació Big Endian, en què el bit 0 (de més a la dreta) correspon al de menys pes (LSB) i el bit amb un índex més gran correspon al de més pes (MSB), o bé la notació Little Endian, que funciona just a l'inrevés.

La figura següent ens mostra ambdues notacions en una paraula de 32 bits:

Notacions Big Endian i Little Endian



2.3. Adreces

Anteriorment hem vist que les dades guardades en memòria necessiten una adreça associada per a poder ser identificades i accedir-hi. Així mateix, també necessitarem identificar els dispositius d'entrada/sortida per a poder interactuar-hi. Si pensem en la memòria del sistema, la podem veure com una agrupació de dades en què cada posició té un índex diferent (adreça) i una capacitat determinada (mida màxima en nombre de bits). En els sistemes encastats, com en qualsevol ordinador, les adreces es codifiquen en binari començant pel 0 i fins al nombre més gran que es pot representar. Per tant, podem dir que no hi ha adreces negatives. Això limita la memòria màxima que el sistema és capaç de suportar, ja que 32 bits proporcionen 4.294.967.296 identificadors diferents i per tant no serem capaços d'encaminar totes les posicions de memòria si aquesta té una mida més gran. És a dir, si les dades tenen una amplària de 8 bits, no podrem tenir memòries més grans de 4 GB amb 32 bits d'adreça. Finalment, cal remarcar que l'adreçament de memòria és una qüestió que moltes vegades no és transparent al programador del sistema, ja que si s'usen llenguatges de programació tipus C++, és habitual l'ús de l'operador & per a obtenir la posició de memòria on s'ubica una variable. D'altra banda, aquesta flexibilitat requereix actuar amb precaució, ja que l'accés descontrolat sobre la memòria pot comportar errors greus en el funcionament del sistema.

Exemple

Si la mida de l'adreça és de 32 bits, totes seran compreses entre 00000000 i FFFFFFFF en hexadecimal.

2.4. Instruccions

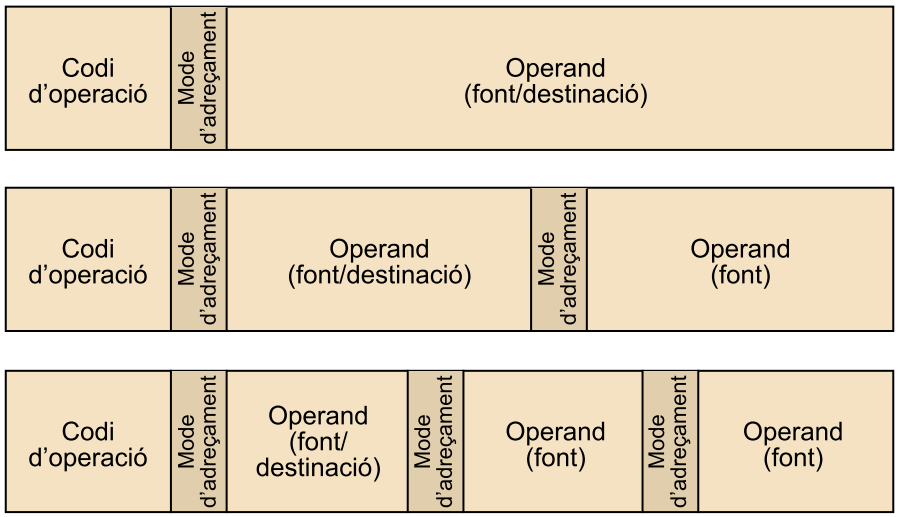
El conjunt d'instruccions d'un sistema encastat conforma el nexe d'unió entre el seu maquinari i el programari. Així, doncs, cada instrucció representa una acció que el maquinari del nostre sistema és capaç de dur a terme. Ens referirem a aquestes accions com a *operacions*, les quals actuen sempre sobre el operands. El nombre d'operands que una operació conté en denota el seu grau, habitualment comprès entre 1 i 3.

A l'hora de representar les instruccions en la memòria de programa i de manera que el processador sigui capaç d'interpretar-les, s'agrupen els bits de dades en camps. Sempre hi ha d'haver un camp destinat a descriure quina és l'operació que s'ha de dur a terme i la resta de camps (habitualment entre 1 i 3) permet localitzar els operands. En la figura següent, podem veure alguns exemples de format d'instruccions en una màquina que utilitza paraules de 32 bits i en notació Big Endian (en Little Endian els camps anirien a l'inrevés de tal manera que el codi d'operació s'ubica sempre en els bits de més pes).

Exemple

Com a exemple, pensem en la instrucció $z = x + y$, escrita en C/C++. Veiem que conté tres operands: els operands font x i y i l'operand destinació z. Si mirem la instrucció en detall, podem observar que en realitat caldrà dur a terme dues operacions bàsiques, que són la de suma i la d'assignació. Per tant, el processador farà primer la suma de x i y, la guardarà temporalment, i n'escriurà el resultat en z. En contraposició, la instrucció $x = x + y$ té només dos operands, però a l'hora d'executar-la caldrà fer dues operacions bàsiques de la mateixa manera que en el cas anterior.

Diferents formats d'instrucció



2.4.1. Tipus d'instrucció

De cara al processador, cada instrucció queda unívocament identificada per una combinació determinada de 0 i 1 i, per tant, el nombre de bits dedicat a aquesta part serà determinat pel total d'instruccions que el micro sigui capaç d'executar. En la fase de programació, no obstant això, no treballem directament amb la informació binària, sinó que utilitzem llenguatges de diferents nivells d'abstracció. El més bàsic (de nivell més baix) és el llenguatge d'assemblador, però hi ha una gran quantitat de micros que es programen amb llenguatges d'alt nivell tipus C/C++. En qualsevol dels casos, necessitarem un programari específic que transformi el nostre codi a 0 i 1, és a dir, a llenguatge de màquina. D'altra banda, el conjunt d'instruccions de què disposa un processador pren una gran importància, ja que defineix la interfície entre el programador i la màquina.

Com que el nostre objectiu en aquests moments és poder classificar el conjunt d'instruccions d'un processador, utilitzarem el llenguatge d'assemblador per tal de mantenir-nos tan a prop com sigui possible del llenguatge de màquina. De fet, la traducció d'assemblador a codi màquina és directa, ja que simplement substituïm els codis d'operació binaris per mnemònics, els quals faciliten que l'ésser humà els interpreti. A grans trets, classifiquem el conjunt d'instruccions en tres gran blocs:

- Instruccions de transferència de dades, que ens permeten moure les dades d'un lloc a l'altre.

- Instruccions de control de flux, que serviran per a controlar l'ordre d'execució de les instruccions que formen un programa.

- Instruccions relacionades amb l'aritmètica i la lògica, que ens permetran fer les operacions lògiques i aritmètiques que siguin necessàries.

Instruccions de transferència de dades

Les instruccions de transferència de dades contenen tres parts clarament diferenciades: la **informació que cal moure**, la seva **localització** o el seu **origen** i el **lloc on es vol guardar** o **destinació**. L'origen i la destinació poden ser:

- La memòria del sistema.
- Un registre.
- Un port d'entrada o sortida del sistema.

Més endavant parlarem dels registres, però de moment direm que són petites memòries volàtils en les quals s'emmagatzema tan sols una paraula i serveixen per a guardar informació de caràcter temporal, com, per exemple, els operands d'una instrucció. Una de les seves característiques més destacades és que són d'accés ràpid tant per a la lectura com per a l'escriptura.

En la taula següent, podem trobar algunes de les instruccions de transferència de dades habituals, per bé que no totes han d'aparèixer en un mateix conjunt d'instruccions. Per exemple, les instruccions LD i ST no les trobarem mai juntament amb la instrucció MOVE.

Exemples d'instruccions de transferència de dades

Instrucció	Significat
LD destinació, origen	*Load*. L'operand origen és transferit al de destinació, que pot ser un registre o la memòria principal.
ST origen, destinació	*Store*. L'operand origen és transferit al de destinació. En aquest cas, l'origen és sempre un registre i la destinació, la memòria del sistema.
MOV destinació, origen	*Move*. Copia l'operand origen a la destinació. Ambdós poden ser memòria o registres.
XCH destinació, origen	*Exchange.* Intercanvia els operands origen i destinació.
PUSH/POP operand	Apilar l'operand a la pila o bé extreure una dada de la pila i transferir-la a l'operand.
IN/OUT destinació, origen	Transferir dades a un port d'entrada o sortida.

Tal com hem comentat anteriorment, just després del codi d'operació apareixen els camps corresponents als operands. No obstant això, hi ha diferents maneres d'obtenir aquests operands i, en conseqüència, cal indicar-ho a l'inici de cada camp, tal com es mostra en la primera figura de l'apartat 2.4. Els modes d'adreçament més comuns són:

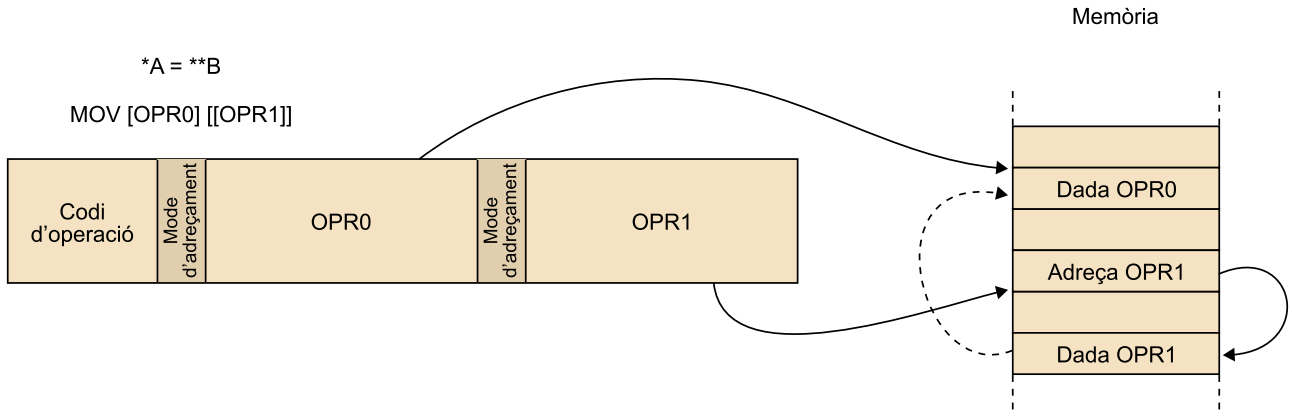
- Immediat.
- Directe i indirecte.
- Directe de registre i indirecte de registre.
- Indexat.

- Relatiu al comptador de programa.

En el mode d'**adreçament immediat**, és la mateixa instrucció la que conté el valor de l'operand i, per tant, és l'opció que requereix un nombre inferior d'accessos a memòria. Aquest mode d'adreçament pot aparèixer en instruccions d'un operand o bé de dos i funciona bé quan el valor que cal transferir és petit (en cas contrari, pot no cabre en la instrucció), com passa en la inicialització de variables, o bé en els índexs del llaços de codi. En el cas d'un sol operand, la destinació ha de ser implícita, com, per exemple, l'acumulador de l'ALU.

En els modes d'**adreçament directe i indirecte**, els operands contenen l'adreça en memòria de les dades que volem transferir. La diferència entre el mode directe i indirecte rau en el nombre de nivells d'indirecció. En l'adreçament directe hi ha un sol nivell d'indirecció i l'adreça que conté l'operand és on es troba la dada d'interès. En canvi, en el mode indirecte, l'operand apunta a una posició de memòria que conté l'adreça on hi ha la dada. En la figura següent, podem veure un exemple de transferència de dades usant adreçament directe i indirecte, en el qual s'utilitza l'operador de referència * de C++, el qual podem interpretar com "el contingut de", és a dir, *A indica el contingut de la posició de memòria que es troba en A. En assemblador, expressarem [*A*] d'una manera equivalent.

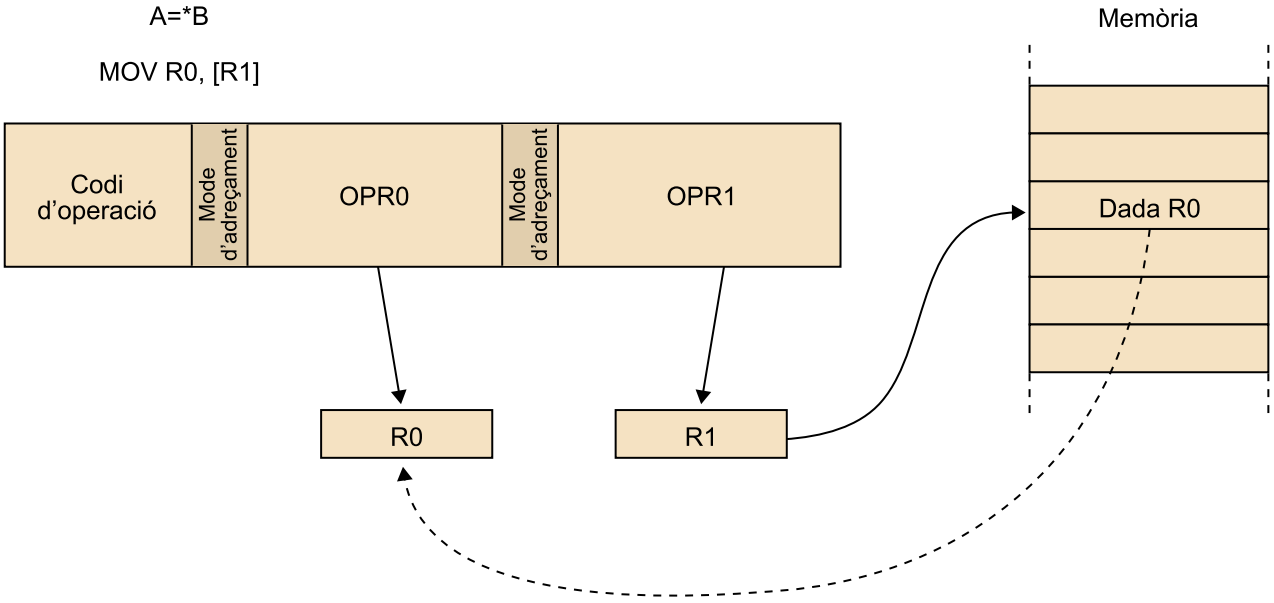
Transferència de dades usant adreçament directe i indirecte



En el mode d'**adreçament directe de registre i indirecte de registre**, l'operand especifica un dels registres del sistema. En el cas d'adreçament directe de registre, el registre seleccionat directament conté la dada d'interès, mentre que en l'adreçament indirecte de registre, aquest conté una adreça de memòria on caldrà accedir per a recuperar o escriure dades. Evidentment, els modes d'adreçament directe i indirecte (usant registres o no) requereixen més accessos a memòria que el mode immediat. L'avantatge d'usar registres és que tenim un ventall més gran de direccions de memòria a on podem apuntar. Fixem-nos que el registre sempre disposarà de la mida de paraula completa del sistema, mentre que un operand en una instrucció ha de compartir aquesta mateixa

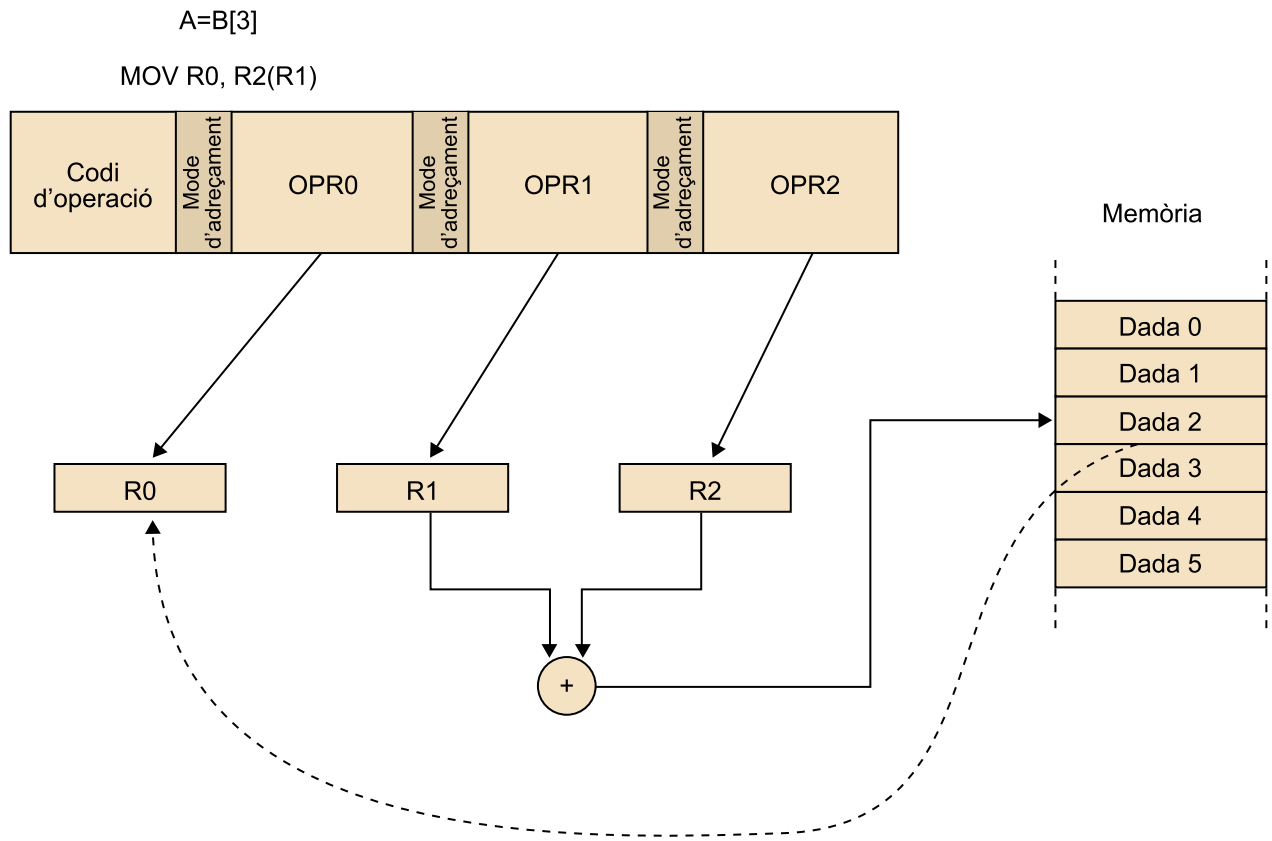
mida amb el codi d'instrucció i la resta d'operands. La figura que incloem a continuació mostra un exemple de transferència de dades usant els modes directe i indirecte per mitjà de registre.

Transferència de dades usant els modes directe i indirecte de registre

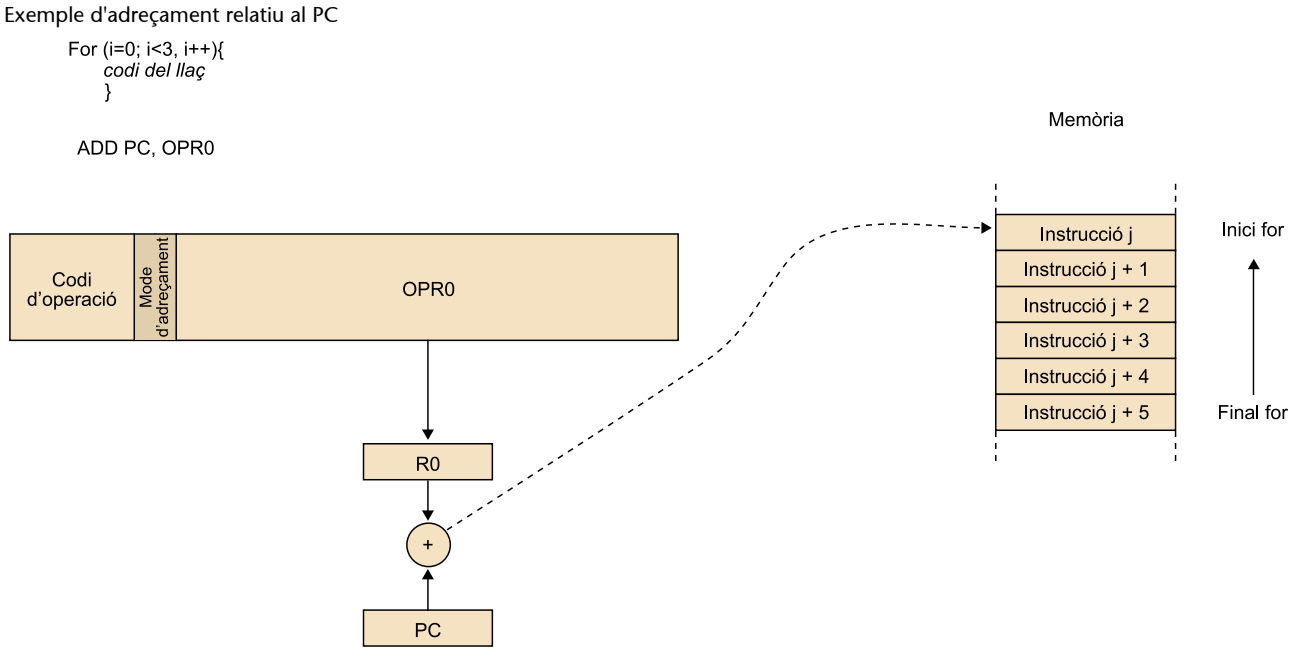


En el mode indexat o de desplaçament, treballem amb un conjunt d'adreces base més òfset, és a dir, l'adreça d'interès s'obté sumant un cert desplaçament a l'adreça base. Es tracta, per tant, d'un mètode adequat per a accedir a estructures de dades de tipus agrupació i el desavantatge més gran que presenta és el temps necessari per a calcular l'adreça volguda i fer l'accés a memòria. En general, l'accés indexat és més costós en aquest sentit que la resta de modes presentats fins al moment. És important remarcar que, després de l'execució de la instrucció, ni l'adreça base ni l'òfset canvien. Podem veure un exemple d'accés indexat en la figura següent, en què l'operand 2 apunta al registre que conté l'òfset, l'operand 1 al registre que conté l'adreça base i finalment l'operand 0 apunta al registre on hi ha l'adreça de destinació. En la figura següent podem veure tant la instrucció escrita en assemblador com l'equivalent en C/C++.

Exemple d'accés indexat



Finalment, el mode d'adreçament relatiu al comptador de programa presenta una mecànica de funcionament semblant al mètode indexat, per bé que hi ha algunes diferències significatives entre tots dos. En primer lloc, l'adreça base és implícita i correspon sempre al valor actual del registre PC (comptador de programa). El que sí que s'indica és l'òfset que cal afegir, que en aquest cas pot ser tant positiu com negatiu. Finalment, un cop calculada la nova adreça, aquesta sobreescriu l'anterior (ubicada al PC). En la figura següent, veiem un exemple en el qual s'utilitza aquest tipus d'adreçament per tal de reiniciar un llaç tipus *for* (cal comprovar que la condició de llaç es compleixi abans de fer el salt). De la mateixa manera que en el cas anterior, el mètode requereix més temps per a completar la instrucció que els tres primers mètodes presentats.



Instruccions de control de flux

Les instruccions de control de flux són les que ens permeten establir l'ordre d'execució d'un programa determinat. En aquest sentit, podem identificar fins a quatre maneres d'operar diferents, que són:

- Execució seqüencial.
- Execució ramificada.
- Execució en llaç.
- Crida a una funció.

La major part de codi de qualsevol programa s'executa en mode seqüencial, és a dir, un cop s'ha completat una instrucció es passa a executar la instrucció que es troba just darrere de la primera en memòria. Un exemple d'execució seqüencial el podem veure en el petit fragment de codi de la figura següent, escrit tant en C/C++ com en assemblador.

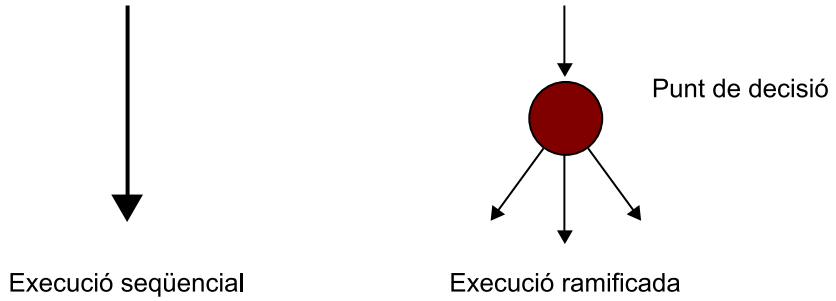
Fragment de codi d'execució seqüencial

```
x = 5;       MOV R1, #5H   // posa 5 en hexadecimal a R1
y = 10;      MOV R2, #AH    // posa 10 en hexadecimal a R2
z = x + y;   ADD R3, R1, R2   // calcula R1 + R2 i ho guarda a R3
```

A diferència d'aquesta última, l'execució ramificada és la que s'inicia en un punt determinat del codi que anomenem **punt de decisió**. A partir d'aquest, es preveuen diverses possibilitats per tal de seguir l'execució. Aquesta és la manera d'operar que tenen instruccions d'alt nivell del tipus *if / else, switch* o bé *case*. Gràficament, podem veure en la figura següent la representació tant de l'execució seqüencial com ramificada. Habitualment, hi ha algun tipus de condició associada a la instrucció i s'utilitza el *flag register* (agrupa diversos

bits, en aquest cas *flags*) per a emmagatzemar temporalment el resultat de la condició. Les condicions més habituals són de tipus lògic entre variables (si són iguals, una més gran que l'altra...) o bé de tipus aritmètic (si el resultat d'una operació és zero, o bé negatiu, o bé provoca un desbordament [*overflow*]).

Execucions seqüencial i ramificada



En funció del resultat obtingut, caldrà executar una part del codi o una altra. Algunes de les instruccions encarregades de fer aquests salts dins del programa es poden veure en la taula següent. Fixeu-vos, però, que també és possible fer un salt a un punt de codi de manera incondicional. En llenguatge d'assemblador, aquests punts de codi es marquen d'una manera senzilla utilitzant etiquetes.

Exemples d'instruccions de salt

Codi d'operació	Descripció
BR etiqueta	Salt incondicional al punt marcat per etiqueta.
BE/BNE etiqueta	Salt al punt marcat per etiqueta quan l'indicador (*flag*) d'igualtat està activat/desactivat.
BZ/BNZ etiqueta	Salt al punt marcat per etiqueta quan l'indicador de zero està activat/desactivat.
BGT etiqueta	Salt al punt marcat per etiqueta quan l'indicador més gran està activat/desactivat.
BV etiqueta	Salt al punt marcat per etiqueta quan l'indicador de desbordament està activat/desactivat.
BC/BNC etiqueta	Salt al punt marcat per etiqueta quan l'indicador *carry* està activat/desactivat.
BN etiqueta	Salt al punt marcat per etiqueta quan l'indicador negatiu està activat/desactivat.

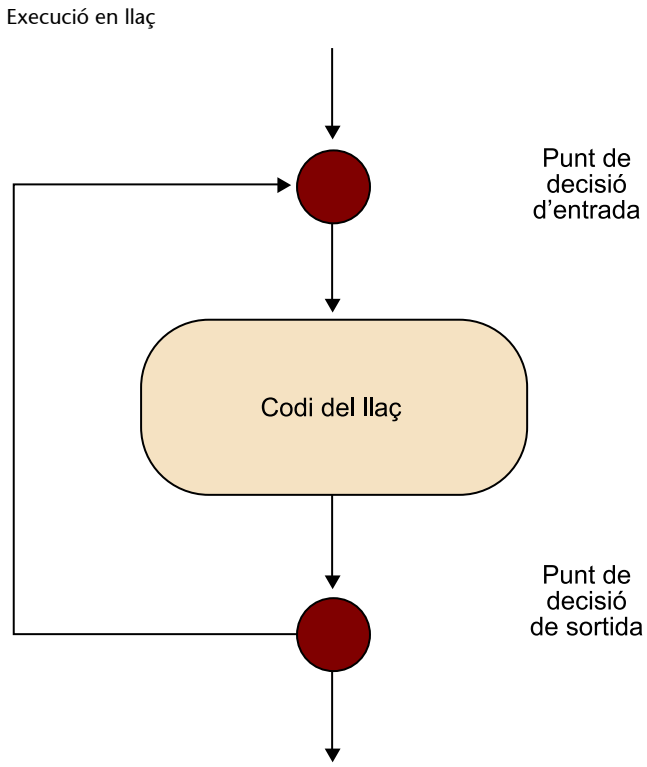
En el quadre següent, veiem un exemple d'un fragment de codi en C/C++ i la seva versió corresponent en assemblador en què s'utilitzen les instruccions de control de flux. Aquí assumim que les variables *x*, *y*, *z*, *a* i *b* es troben ja en els registres R1-R5, respectivament.

Fragment de codi d'execució ramificada

```
If (x==y)  CMP R2, R1          // compara x i y (actualitza el flag d'igualtat)
```

```
  z=a+b;   BE $1              // si són iguals (flag=1) salta a $1
else       SUB R3, R4, R5     // resta a i b posant el resultat a z
  z=a-b;   BR $2              // salt incondicional a l'etiqueta $2
           $1: ADD R3, R1, R2 // calcula R1+R2 i ho guarda a R3
           $2: ...            // continua l'execució del codi
```

En l'*execució en llaç*, de la mateixa manera que en el cas anterior, podem identificar un punt de decisió que ens indicarà si cal continuar executant un tros de codi determinat (el mateix codi del llaç) o no. La decisió es pot prendre abans de començar aquesta part del codi o bé després. En el primer cas direm que es tracta d'un llaç amb condició d'entrada, mentre que el segon serà un llaç amb condició de sortida, tal com veiem en la figura següent. L'execució en llaç és la que utilitzen instruccions d'alt nivell com ara *do/while*, *while* o bé *for*.



Finalment, veiem en el requadre següent un fragment de codi escrit tant en C/C++ com en assemblador en què s'implementa una execució en llaç. Suposem que inicialment el registre R1 ha estat carregat amb varCond i el registre R2 amb index.

Fragment de codi d'execució en llaç

```
                        $1: CMP R1, #AH   // comparem varCond amb 10
while (varCond <10){    BGE $2            // si R1 és més gran o igual que 10, saltem a $2
     index = index+2;   ADD R2, #2H       // afegim 2 a index
     varCond++;         ADD R1, #1H       // afegim 1 a varCond
}                       BR $1             // tornem a l'inici del llaç
```

```
$2: ...          // continua l'execució del codi
```

Finalment, la **crida a una funció** és potser la instrucció de flux més complexa que trobem, en part perquè involucra la pila, una estructura de dades especial. Aquesta estructura ocupa una regió de memòria determinada, exclusivament dedicada a ella, en què les dades entren i surten seguint una política LIFO (darrer d'entrar, primer de sortir), a diferència de l'accés a memòria, on és aleatori. A més a més, sempre tenim un punter associat a la pila o *stack pointer* (SP) que indica la posició de l'última dada emmagatzemada. Les instruccions per a apilar i desapilar són *PUSH origen* i *POP destinació*, respectivament, i cap d'elles no admet el mode d'adreçament immediat. La pila té diverses aplicacions, la majoria d'elles relacionades amb l'execució de subrutines, que són:

- Salvar/restaurar registres.
- Crides successives a subrutines.
- Interfície per a pas de paràmetres a subrutines.

En la primera aplicació, utilitzem la pila per a guardar la informació existent en els registres del processador. Això es fa sobretot quan entrem en una subrutina per tal que no tingui cap efecte col·lateral un cop tornem a la funció que ha efectuat la crida. Abans d'executar la funció apilarem els registres que vulguem guardar i en sortir els desapilarem, però sempre en ordre invers per tal de retornar la informació al registre que toca.

Quan utilitzem la pila per a fer la crida a funcions, mitjançant les instruccions CALL i RET. En el moment que cridem la funció, executem `CALL nomFuncio`, la qual cosa provoca el salt al punt del codi etiquetat com a `nomFuncio` i, a més a més, apila l'adreça de retorn, és a dir, l'adreça en què es troba la instrucció immediatament posterior al `CALL`. Així, en acabar la funció, executarem `RET` per recuperar (desapilar) l'adreça de retorn i continuar amb l'execució de la funció principal.

Finalment, la pila també ens serveix per a passar paràmetres a una funció, per exemple, si no tenim prou registres disponibles. Considerem, per exemple, la funció C que té com a capçalera `void acumula (int *a, int b)`. Si mantenim el conveni *C* de pas de paràmetres, els apilarem de dreta a esquerra, és a dir, primer posarem *b* i després l'adreça de *a*. A part, a dins del cos de la funció, ens pot interessar moure'ns per la pila per poder accedir a aquestes variables. De vegades, com, per exemple, en els sistemes basats en x86, ens trobarem que no podem usar el registre SP indirectament. Llavors, s'utilitza un registre alternatiu: el *base pointer* (BP) en el cas de x86, en què copiem el valor de l'SP. El retorn de paràmetres de la funció, en canvi, es fa habitualment per mitjà de registres.

En la figura següent, podem veure el codi de la subrutina `void acumula (int *a, int b)`, per a un sistema basat en i8086, en què els registres AX i BX s'utilitzen per a operar amb les dades (AX seria el registre de retorn en cas que la funció no fos de tipus void). També veiem l'aspecte de la pila en el moment d'entrar al cos de la funció. Els passos que seguim són:

- Enllaç dinàmic i variables locals: guardem el valor del BP anterior per restaurar-lo abans de sortir. Podria ser que fos necessari per a la funció que ha cridat *acumula*. Si volguéssim posar variables locals a la pila, podríem deixar l'espai necessari a continuació.

- Guardar registres: salvem el valor dels registres AX i BX per restaurar-los al final de la funció, ja que podrien ser necessaris per a la funció principal.

- Cos de la funció: accedim als paràmetres de la funció a partir de BP per posar-los en registres i en fem l'acumulació. Fixem-nos que el resultat ja queda en memòria.

- Restauració de registres i eliminació de variables locals (si n'hi ha): és important seguir l'ordre invers a causa del mecanisme LIFO de la pila.

- Restauració de l'enllaç dinàmic: recuperem el valor anterior de BP.

Si ens fixem en l'estat de la pila, veurem que, arribats a aquest punt, la dada que queda a dalt de tot és l'adreça de retorn i, per tant, en executar RET continuarem amb l'execució de la funció principal sense causar-hi cap mena d'interferència.

Exemple de subrutina en un sistema x86

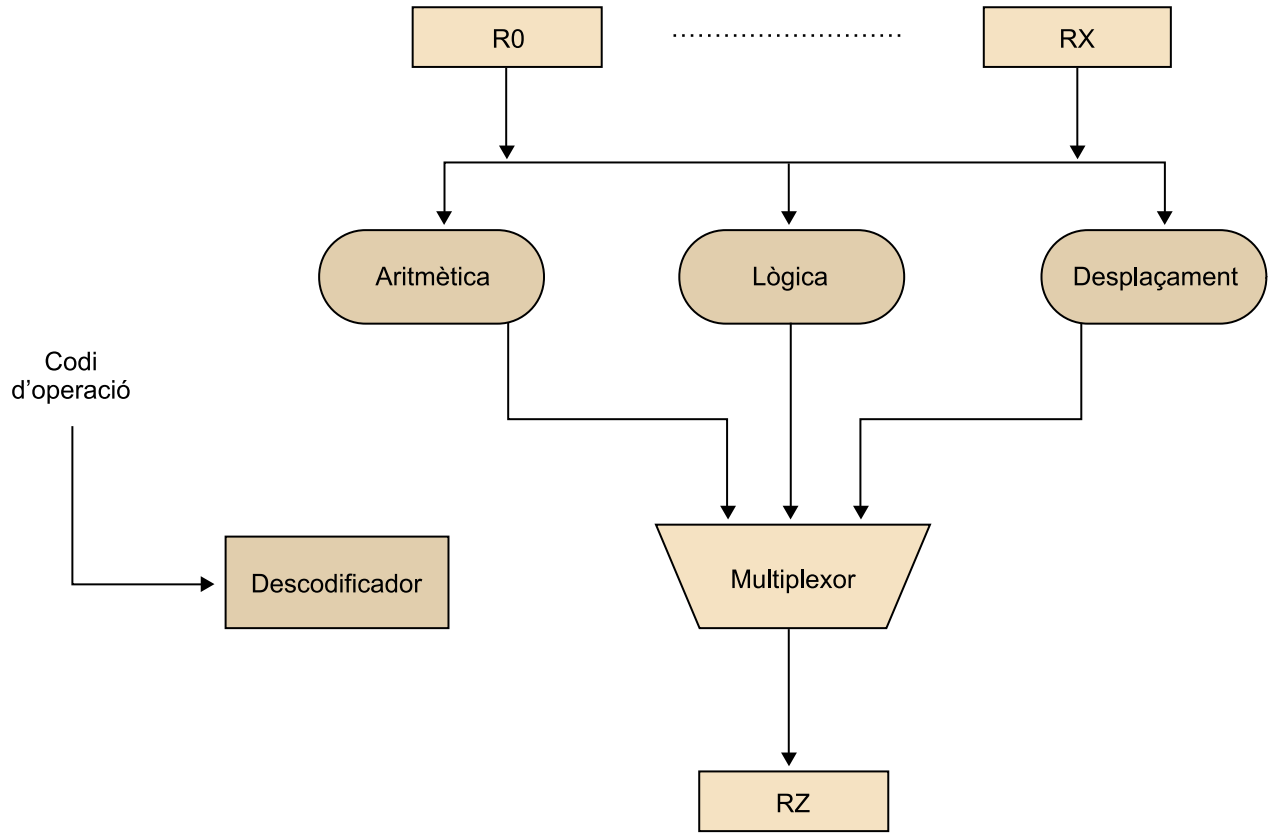
```
acumula:    PUSH BP
            MOV BP, SP
            PUSH BX
            PUSH AX

            MOV BX, [BP+4]
            MOV AX, [BP+6]
            ADD [BX], AX

            POP AX
            POP BX
            POP BP
            RET
```

Instruccions de lògica i aritmètica

En qualsevol sistema encastat, les instruccions de lògica i aritmètica són sempre executades en la unitat aritmeticològica (ALU) del processador, la qual aglutina diversos elements de maquinari per a poder dur a terme les diferents tasques que se li encomanen.

Un possible diagrama funcional d'ALU es pot trobar en la figura següent. Hi apreciem diversos registres que emmagatzemen temporalment els operands, un bloc dedicat a les operacions aritmètiques, un altre de dedicat a les operacions de lògica, un bloc específic per al desplaçament de bits, un multiplexor i un registre de sortida. En funció del codi d'operació que conté la instrucció, el controlador de l'ALU activa un dels blocs anteriors per tal de computar el que se li demana. Finalment, el resultat s'escriu en un registre de sortida per tal de posar-lo a disposició del sistema.

Diagrama de blocs d'una ALU



Típicament el processador del sistema és capaç de dur a terme quatre funcions aritmètiques: suma, resta, divisió i multiplicació, per bé que alguns processadors implementen només suma i resta, i deixen les altres dues operacions per a ser tractades a nivell de programari. Així mateix, trobarem sistemes capaços de treballar amb coma flotant i d'altres que només seran capaços de fer operacions amb enters.

Entre les instruccions lògiques, trobarem típicament les funcions AND, OR i XOR bit a bit. Finalment, les instruccions de desplaçament ens permetran desplaçar bits o paraules a dreta i esquerra. Podem trobar diferents versions de desplaçament en funció de com s'ompli el nou bit generat a conseqüència del desplaçament i com s'utilitzi el bit que se n'extreu.

2.5. Memòria

El subsistema de memòria és una part essencial de qualsevol sistema encastat, ja que és on guardem tant les dades com les instruccions que formen part de les nostres aplicacions. El disseny correcte d'aquest subsistema quant a dimensionament, gestió dels propis recursos i adequació dels temps d'execució afecta directament el rendiment global del sistema. És més, assignacions de memòria inadequades poden fins i tot deixar la nostra aplicació en un punt mort, "penjada". En general, ens podem trobar en dues situacions clarament diferenciades: si la nostra aplicació és petita, la memòria interna del processador pot resultar suficient i, per tant, no tindrem problemes de compatibilitat. En canvi, en aplicacions més grans, haurem de proveir el sistema d'un mòdul extern de memòria. En aquest cas, la compatibilitat, sobretot quant a temps d'execució, és fonamental, tal com veurem més endavant.

Quant al tipus de memòria, d'entrada la podem dividir en dos grans grups: memòria d'accés aleatori o RAM i memòria només de lectura o ROM. A continuació, es presenta una classificació més detallada dels diferents tipus de memòria, per bé que no és exhaustiva. Tenim:

- **RAM**. El seu nom indica que podem accedir a qualsevol posició de memòria en qualsevol moment, a diferència del que succeeix en altres tipus de memòria (normalment antics) com poden ser les cintes magnètiques, on l'accés a una posició determinada passa forçosament per l'accés a posicions anteriors. A més a més, els temps de lectura i escriptura són del mateix ordre de magnitud. D'entre els diferents tipus de memòria RAM podem destacar:
  - **Memòria RAM dinàmica (DRAM)**. Es tracta d'un disseny senzill de la cel·la de memòria elemental (1 bit), basat en l'emmagatzematge de càrrega sobre un condensador. Com que la càrrega emmagatzemada és sempre volàtil, les memòries de tipus DRAM requereixen l'operació de refrescament, és a dir, cada cert temps cal fer una lectura de la memòria i tornar-hi a guardar el valor resultant. Finalment, cal remarcar que la lectura/escriptura es fa d'una manera asíncrona al rellotge del sistema.

  - **Memòria RAM estàtica (SRAM)**. Es tracta d'un disseny més complex que l'anterior i utilitza la filosofia dels biestables (vegeu la figura 24 com a exemple), amb la qual cosa s'evita la necessitat del refrescament. Continuen sent memòries asíncrones com les anteriors però són més ràpides.

  - **Memòria RAM pseudoestàtica (PSRAM)**. Les memòries PSRAM són formades per un nucli de memòria DRAM i una circuiteria perifèrica que s'encarrega tant del refrescament com del control d'adreces. Tècnicament és una memòria DRAM però es comporta com si fos SRAM.

  - **DRAM sincrònica (SDRAM)**. A diferència de la memòria DRAM, la SDRAM sincronitza les adreces, les dades i els senyals de control amb el rellotge del sistema, la qual cosa permet incrementar les taxes de transmissió respecte de la versió asíncrona.

- **ROM**. La memòria ROM és una memòria d'accés aleatori com la RAM però està pensada, com el seu nom indica, només per a lectura. S'hi pot escriure, òbviament, però el procés d'escriptura és sempre molt més lent que el de lectura i normalment ens hi referim com a programació de la ROM. Els tipus més destacats de memòria ROM són:
  - **Memòria ROM programable (PROM)**. Es tracta d'una memòria que es programa un sol cop utilitzant un aparell específic.

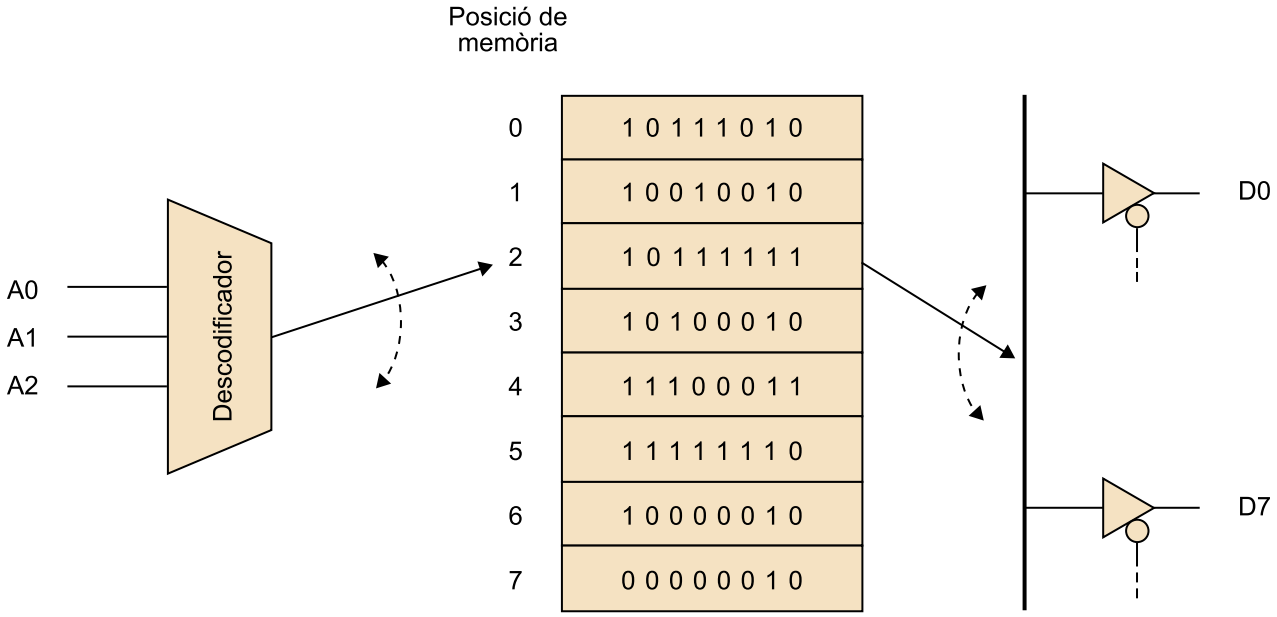
  - **Memòria EPROM (*erasable programmable read only memory*)**. És una memòria de tipus PROM, però amb la particularitat que pot ser esborrada. Per a fer-ho, cal situar la finestra del dispositiu sota llum ultraviolada durant un interval de temps determinat.

  - **Memòria EEPROM (*electrically erasable programmable read only memory*)**. En aquest cas, es pot esborrar elèctricament amb el mateix aparell utilitzat per a la programació.

  - **Memòria flaix**. Es tracta d'un tipus de memòria EEPROM que ha adquirit gran importància en els últims anys (memòries USB, targetes SD, discos durs d'estat sòlid...). En aquest cas, no és necessari treure la memòria del circuit on es troba habitualment per tal de reprogramar-la.

Quant a la interacció entre memòria i sistema, cal dir que la interfície de qualsevol memòria és formada per tres tipus de senyals que, coincidint amb els tipus de busos, són: adreces, dades i senyals de control. Independentment del tipus de memòria i del mecanisme d'emmagatzematge que empri, la interfície més bàsica amb la memòria la podem veure en la primera figura de l'apartat 3.5.2., en què de moment s'han obviat els senyals de control. Fixem-nos que un conjunt de bits del bus d'adreça identifiquen una paraula de la memòria, que es llegeix o s'escriu mitjançant els *buffers* (seguidors de tensió) d'entrada/sortida. No obstant això, i per qüestions de fabricació, és possible que les memòries no s'adeqüin sempre a les necessitats del sistema en termes de mida, és a dir, que no hi hagi prou posicions o adreces o bé que la longitud de la paraula en memòria no sigui suficient. Més endavant veurem com es poden solucionar totes dues situacions, entre d'altres.

Model i interfície bàsics d'una memòria



Els senyals que contenen les adreces són sempre entrades del subsistema de memòria, i també ho són normalment els senyals de control. En canvi, les dades poden ser d'entrada o sortida segons escrivim o llegim en/de la memòria. La funció dels senyals de control és, en general, validar les operacions que s'han de dur a terme. Atès que els senyals digitals experimenten un transitori a l'hora de canviar de nivell, és de vital importància saber en quin moment els valors són vàlids.

Exemple

En el moment en què volem llegir el contingut d'una adreça de memòria determinada, no és suficient indicar-ne l'adreça, sinó que també hem d'indicar en quin moment els senyals que representen l'adreça en qüestió són vàlids per tal d'evitar interpretacions errònies.

A la pràctica, això es tradueix a esperar el temps mínim necessari entre la càrrega de l'adreça i l'activació del senyal de lectura. A continuació, esmenten alguns dels senyals de control més habituals:

- **Chip select (CS)**. Habilita el dispositiu de memòria. Generalment és vàlid amb valor baix i pot servir per a evitar el consum d'energia quan el xip de memòria no s'utilitza.

- **Output enable (OE)**. Control del *buffer* de sortida del dispositiu. Acostuma a ser vàlid amb valor baix i és llavors quan permet la lectura de les dades. En cas contrari, el *buffer* manté un estat d'alta impedància.

- **Read (R)**. Indica que es vol fer una operació de lectura.

- **Write (W)**. Indica que es vol fer una operació d'escriptura.

- **Column address strobe (CAS)**. Indica que els senyals presents al bus de dades corresponen a l'adreça de la columna per al dispositiu. Tal com veurem més endavant, les memòries s'organitzen internament de manera matricial i, per tant, una posició queda completament determinada si n'indiquem la fila i la columna on es troba.

- **Row address strobe (RAS)**. Indica que els senyals presents al bus de dades corresponen a l'adreça de la fila per al dispositiu.

Finalment, la taula següent mostra la presència d'aquests senyals més habituals segons el tipus de memòria amb què treballem.

Senyals de control habituals

Tipus de memòria	CS	OE	R	W	CAS	RAS
ROM	x	x				
SRAM	x	x	x	x		
DRAM	x	x	x	x	x	x

En qüestió de temporització, tot i que es detallarà més endavant tenint en compte el tipus de memòria amb què treballem, sí que podem dir que, a grans trets, se segueix sempre la seqüència següent. En primer lloc, el bus d'adreces i el de dades (en cas d'escriptura) han d'estar preparats o carregats, és a dir, amb la tensió estabilitzada.

Nota

Recordem que els canvis de tensió en els busos i circuits lògics no són immediats, sinó que experimenten un transitori.

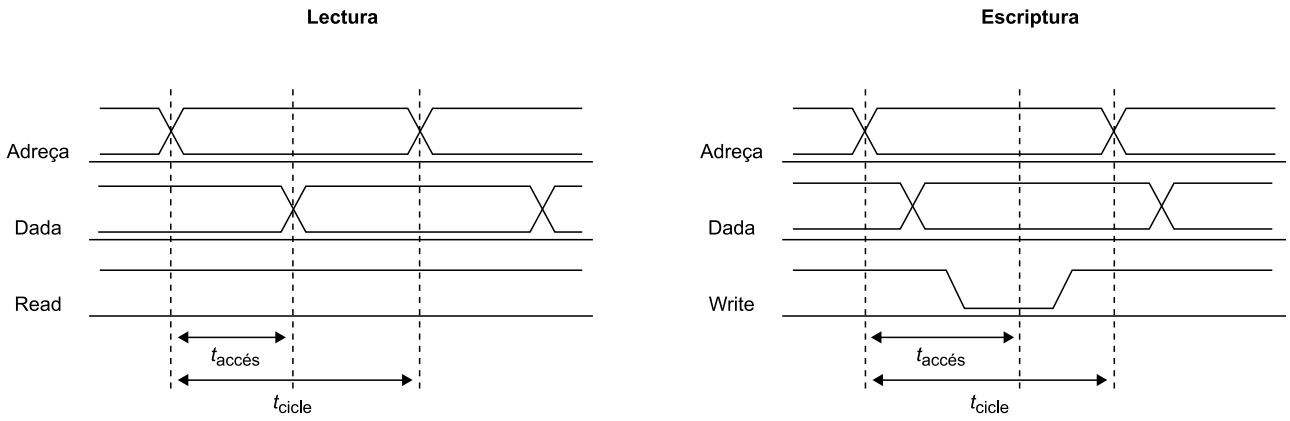
Després d'aquest primer pas, ja és possible donar l'ordre de lectura o escriptura, i assegurar-nos així de no tenir comportaments inesperats. Associada a aquesta temporització, trobem una sèrie de paràmetres que caracteritzen les memòries. Alguns dels més destacats són:

- **Temps d'accés**: és el temps necessari per a obtenir una paraula de la memòria o bé per a escriure-la. Es mesura des del moment en què apliquem l'adreça fins que l'operació finalitza, és a dir, fins que les dades són en el *buffer* de lectura en el cas de lectura o bé fins que les dades s'han emmagatzemat en memòria en el cas d'escriptura (vegeu-ho en la figura següent). Assumim que l'escriptura s'ha completat un temps determinat després que s'iniciï la transició alt-baix en el senyal "Write" per tal d'activar l'escriptura.

- **Temps de cicle (lectura o escriptura)**: és el temps mínim que ha de passar entre dos inicis de lectura o d'escriptura consecutius i, per tant, ens diu quina és la velocitat a la qual podem accedir a la memòria.

- **Amplària de banda**: s'utilitza per a mesurar la capacitat d'una memòria determinada per a transmetre o rebre un flux de dades i es mesura en paraules enviades/rebudes per unitat de temps. La unitat emprada habitualment és el MBps.

Temps d'accés i de cicle en la lectura o escriptura de/en memòria



Finalment, hi ha altres paràmetres temporals associats a certes magnituds de memòria que van més enllà de la paraula. Aquestes altres magnituds són el bloc i la pàgina. Un bloc és simplement una agrupació lògica d'un conjunt de paraules i la seva utilitat es fa evident en el moment que un sistema determinat vol transferir certes quantitats de dades, ja que s'acostuma a fer a nivell de bloc i no pas de paraula. Així mateix, també es defineix la pàgina com una agrupació de blocs. A partir d'aquestes definicions podem introduir:

- **Latència**: és el temps necessari per a obtenir l'adreça del primer element d'una seqüència determinada i accedir-hi.

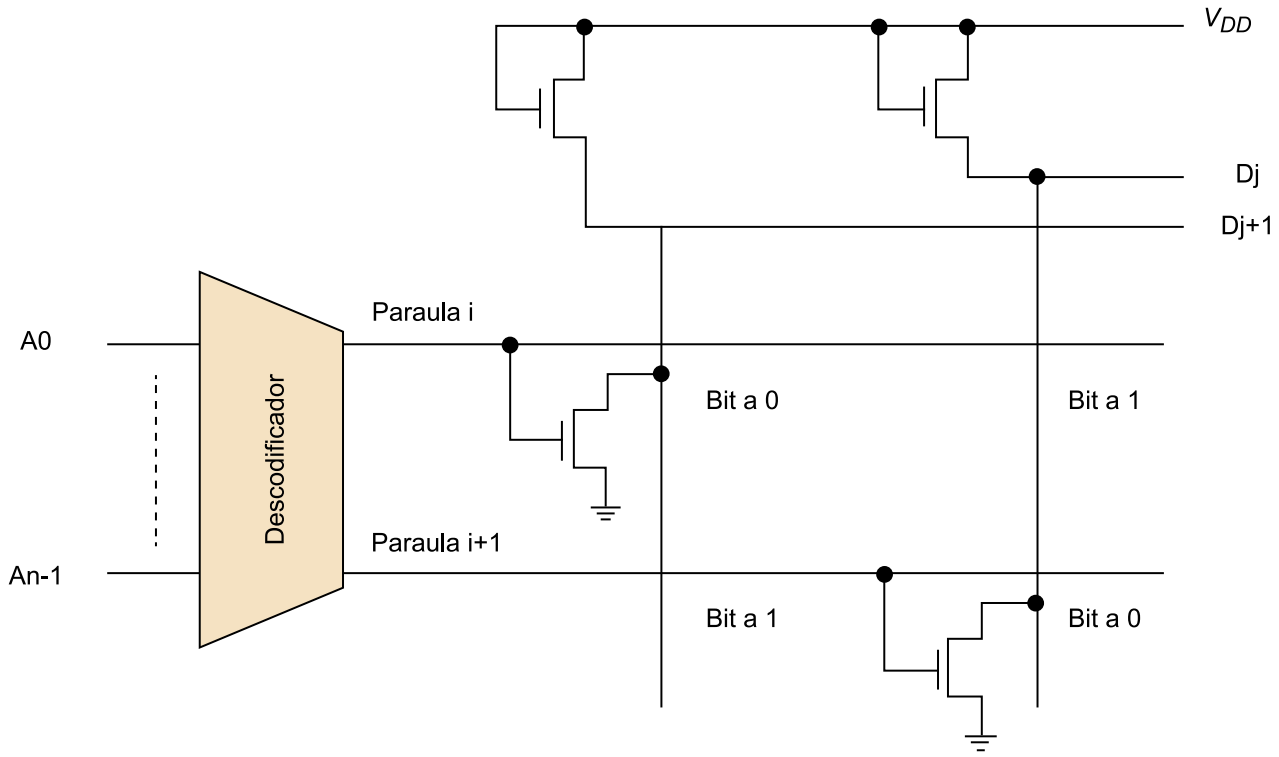
- **Temps d'accés de bloc**: és el temps necessari per a accedir a un bloc complet. Per tant, inclou el temps d'accés al primer element (latència) i la transferència de la resta d'elements.

A continuació, s'aprofundeix en els dos tipus bàsics de memòria esmentats anteriorment: ROM i RAM.

2.5.1. Memòria ROM

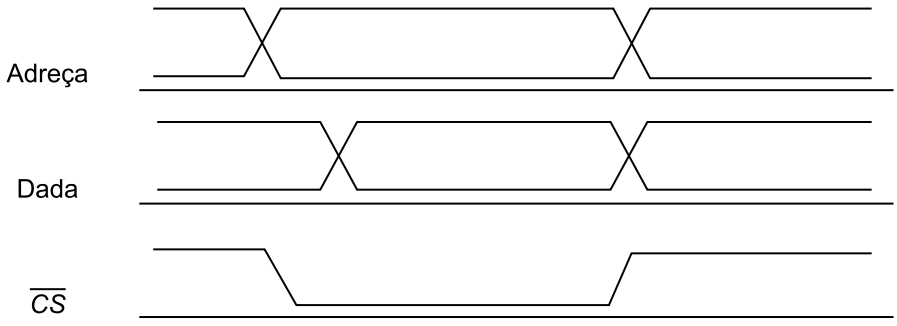
En aquest apartat, considerarem la memòria ROM un dispositiu de només lectura, és a dir, n'obviarem la part de programació o escriptura. Pel que fa a la implementació, es pot aconseguir emmagatzemar un bit amb la presència o no d'un transistor. En la figura següent, veiem un possible esquema de memòria ROM utilitzant únicament transistors NMOS. Fixem-nos que cada fila interna de la memòria és activada o seleccionada per una única combinació dels bits d'entrada, que formen l'adreça. *A priori*, totes les columnes estan carregades a un valor lògic 1 (transistors NMOS amb la porta connectada a la tensió de referència) i, per tant, si no hi ha cap altre element, tots els bits de la paraula seleccionada prendrien un valor lògic 1. No obstant això, la col·locació d'un transistor NMOS connectat a terra fa que la tensió final de la cel·la o bit en qüestió se situï en un valor lògic 0, ja que es drena el corrent a massa.

Memòria ROM implementada amb transistors NMOS



Quant a la interacció amb el dispositiu, haurem d'indicar en primer lloc l'adreça de la qual es vol fer la lectura i també de la qual volem fer l'operació. En l'exemple de la figura següent, fixem-nos que s'habilita el senyal $\overline{CS}$ (actiu amb valor baix). A partir d'aquest moment, cal esperar el temps d'accés determinat pel fabricant per a tenir les dades disponibles al *buffer* de sortida. Cal destacar que en aquest cas no s'ha considerat el senyal de control OE per bé que la seva funcionalitat és necessària, ja que per a evitar conflictes en el bus dades, convé que la sortida de la memòria es mantingui en estat d'alta impedància quan no s'utilitza.

Temporització dels senyals en una memòria ROM



2.5.2. Memòria RAM

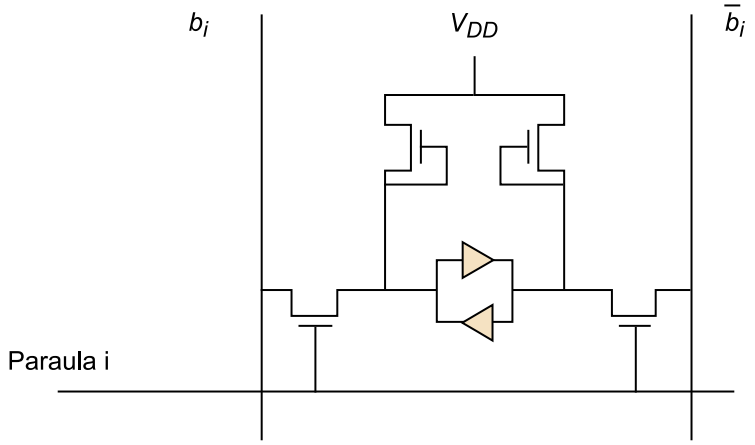
En aquest punt ens centrarem en els dos tipus bàsics de memòria RAM, és a dir, memòria RAM estàtica i memòria RAM dinàmica. En primer lloc, descriurem la cel·la bàsica de memòria en cadascun dels casos i, acte seguit, tractarem de diferents aspectes relacionats amb la integració de la memòria en un sistema encastat.

Memòria RAM estàtica (SRAM)

La cel·la bàsica de memòria SRAM es pot veure en la figura següent i és formada per sis transistors. Recordem que cada inversor és format per dos transistors i, a més, fan falta els dos transistors de càrrega connectats a la tensió de referència. A més a més, fan falta dos transistors més d'accés per tal de poder executar les operacions de lectura i escriptura. La part essencial de la memòria i on s'emmagatzema el bit d'informació és el *latch*[1] que formen els dos inversors.

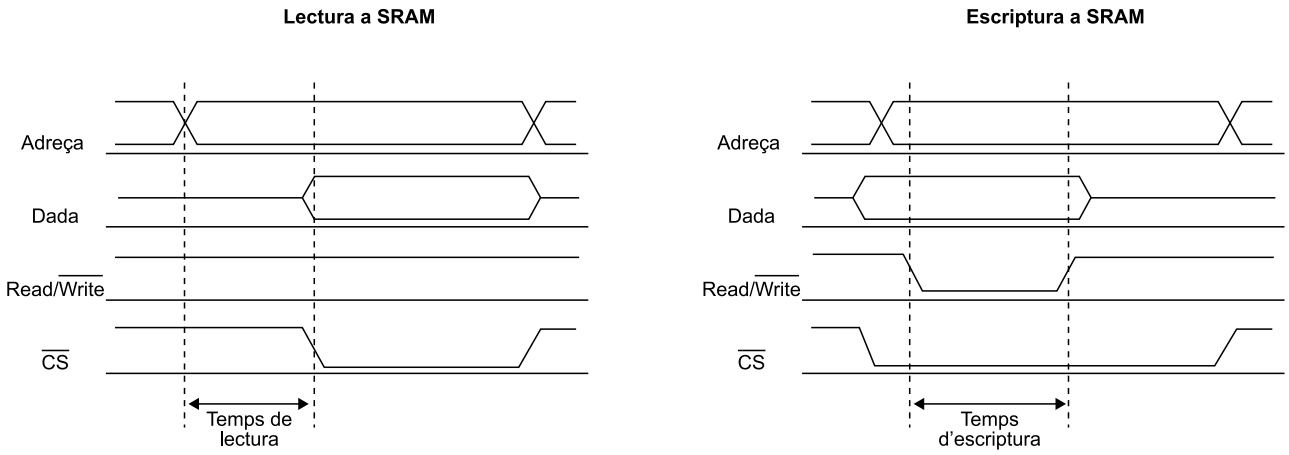
[(1)] *Latch* és sinònim de *biestable* o *flip flop*.

Cel·la bàsica d'una memòria SRAM



Per tal de llegir de la memòria, cal tenir precarregades les línes b_i i $\bar{b}_i$ a una tensió intermèdia entre massa i la de referència. Llavors, quan s'activen els transistors d'accés seleccionant l'adreça de memòria corresponent, la línia b_i i la $\bar{b}_i$ prenen els valors emmagatzemats. Finalment, el valor del bit ha de ser sensat, amplificat i transmès a l'exterior. A l'hora d'escriure, simplement carreguem les línies b_i i $\bar{b}_i$ mitjançant els amplificadors d'escriptura. Un cop s'activin els transistors d'accés, el bit d'informació quedarà emmagatzemat en el *latch* de la figura. En la figura següent veiem la temporització típica en els processos de lectura i escriptura a una memòria SRAM. Fixem-nos que es distingeixen els temps d'accés a memòria per a cadascuna de les operacions, ja que no han de ser iguals.

Temporització dels senyals en una memòria SRAM



En el moment d'integrar una memòria determinada en un sistema encastat, és possible que el processador tingui una mida del bus de dades o d'adreces que coincideixi amb el nombre de línies disponibles a la memòria. En aquest cas, la integració és immediata. Però això no és sempre així. Ens podem trobar que aquesta condició no es compleixi per alguna de les dues bandes, és a dir, pot ser que el nombre de línies del processador no sigui suficient o bé que la memòria quedi curta tant pel que fa a la mida de paraula com del nombre d'adreces possibles. A continuació, es planteja un exemple de disseny el més general possible i se'n proposa una possible solució d'integració.

Suposem que les especificacions del sistema en qüestió requereixen una memòria SRAM capaç d'emmagatzemar 4 K paraules de 16 bits. No obstant això, els mòduls que tenim disponibles són d'1 K paraules i només 8 bits. Així, doncs, seran necessaris 8 mòduls per tal de poder complir les especificacions. A més a més, el processador té un bus de dades de 8 bits d'amplària i un bus d'adreces també de 8 bits. Sabent que, per a poder distingir 4 K adreces diferents ens calen 12 bits, seran necessàries dues transferències tant per a les adreces com per a les dades.

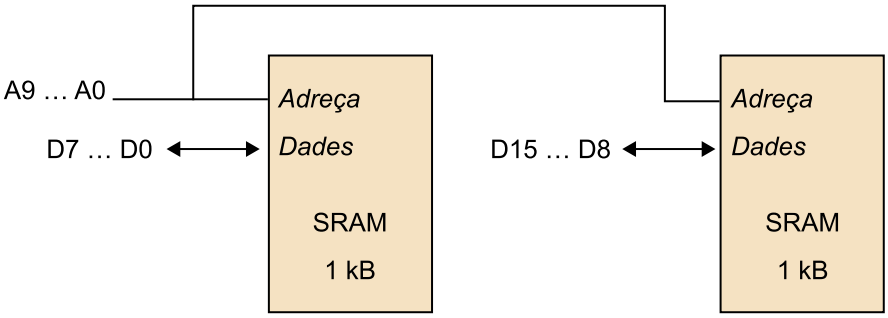
En primer lloc, utilitzarem blocs de dos mòduls de memòria per tal d'obtenir 1 K paraules de 16 bits d'amplària tal com es mostra en la figura següent. Com s'hi pot apreciar, les línies d'adreça són comunes als dos mòduls, mentre que els bits de dades es divideixen entre ells, és a dir, un mòdul s'encarrega dels bits D7...D0 i l'altre dels bits D15...D8. Un cop obtinguts els blocs d'1 K × 16 bits, resulta més o menys senzill ampliar el disseny, en aquest cas quadruplicar, a un sistema SRAM de 4 K × 16 bits. Si disposem de 4 blocs de memòria d'1 K × 16 bits, només cal seleccionar en quin dels 4 blocs hem d'escriure o llegir les dades a cada accés. Fixem-nos que 4 K paraules es representen amb 12 bits. D'aquests, 10 bits (A9...A0) seran necessaris per tal d'especificar la paraula desitjada dins del bloc de memòria. Els altres dos bits (A11, A10) serviran per a determinar de manera unívoca quin bloc de memòria està actiu en cada moment. Fixem-nos que els bits d'adreça A9...A0 han de ser comuns als quatre blocs de memòria i,

Exemple

L'adreça 001011001111 correspon al primer bloc de memòria (00) i dins d'aquest, hem de mirar l'adreça 1011001111.

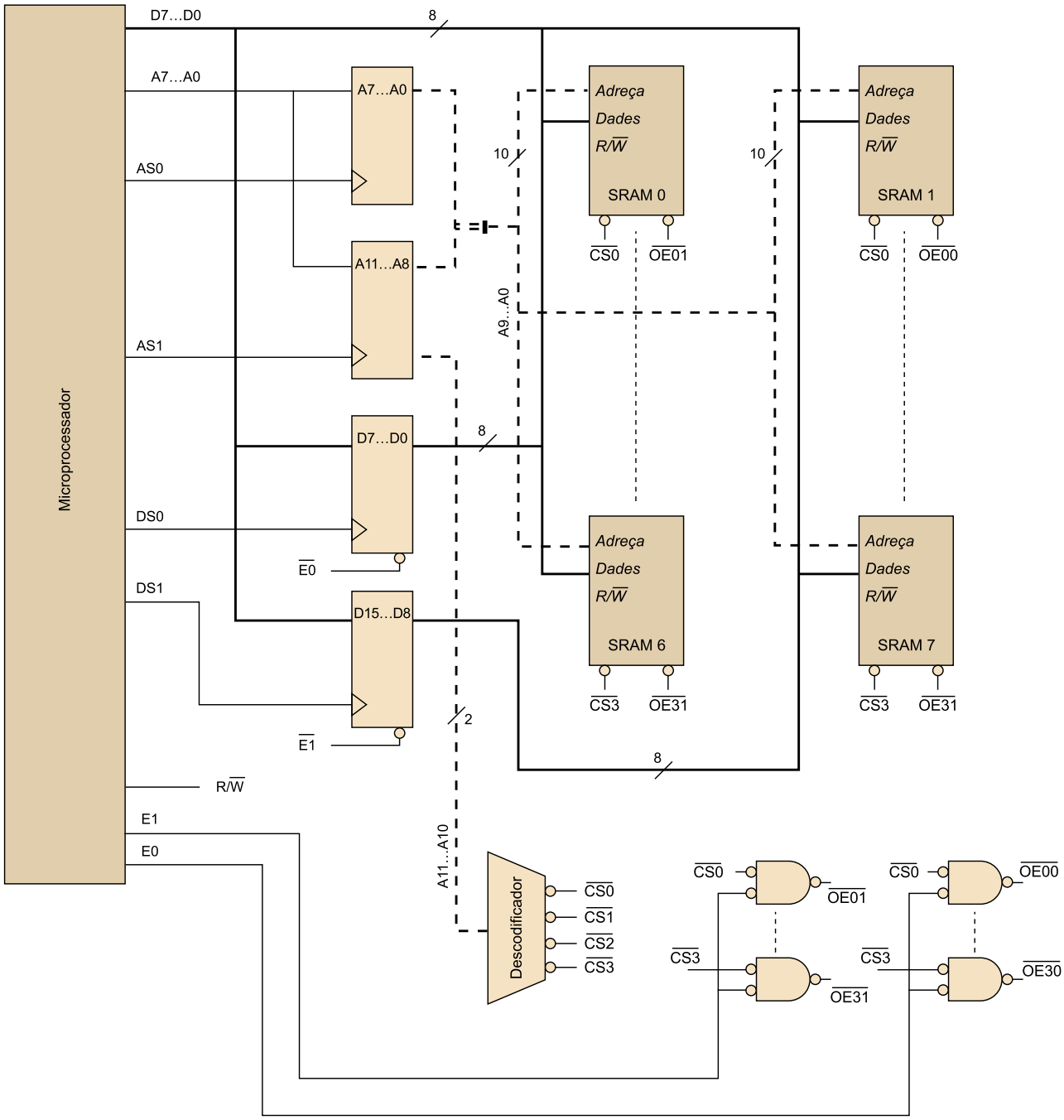
per tant, és imprescindible desactivar els tres últims blocs per tal de no accedir a una posició indesitjada. Un senzill bloc de lògica combinacional ens servirà per a dur a terme aquesta missió.

Memòria SRAM de 16 bits a partir de dos blocs de 8 bits



Finalment, ens cal solucionar el fet que el processador disposi de només vuit línies tant per a les dades com per a les adreces. Recordem que tant les paraules (16 bits) com les adreces (12 bits) del sistema en qüestió requereixen dues transferències. No obstant això, necessitem tots els 16 bits de dades i tots els 12 bits d'adreça disponibles a l'hora d'accedir a la memòria, almenys per a escriure. Amb aquesta finalitat utilitzarem 4 registres (2 per a les dades i 2 per a les adreces). Tot i que parlarem dels registres més endavant, de moment quedem-nos amb la idea que són elements de memòria de mida petita (uns quants bits). Quan hi ha un senyal d'activació, emmagatzemen els bits que es troben a la seva entrada en aquell moment i, acte seguit, es troben en disposició de ser llegits a la sortida fins que hi hagi el senyal següent, moment en què els bits guardats seran reemplaçats pels nous bits d'entrada. El disseny complet del sistema de memòria i la seva interacció amb el microprocessador es pot veure en la figura següent. A continuació detallem el funcionament de les operacions d'escriptura i lectura.

Sistema de memòria SRAM 4 K × 16 bits

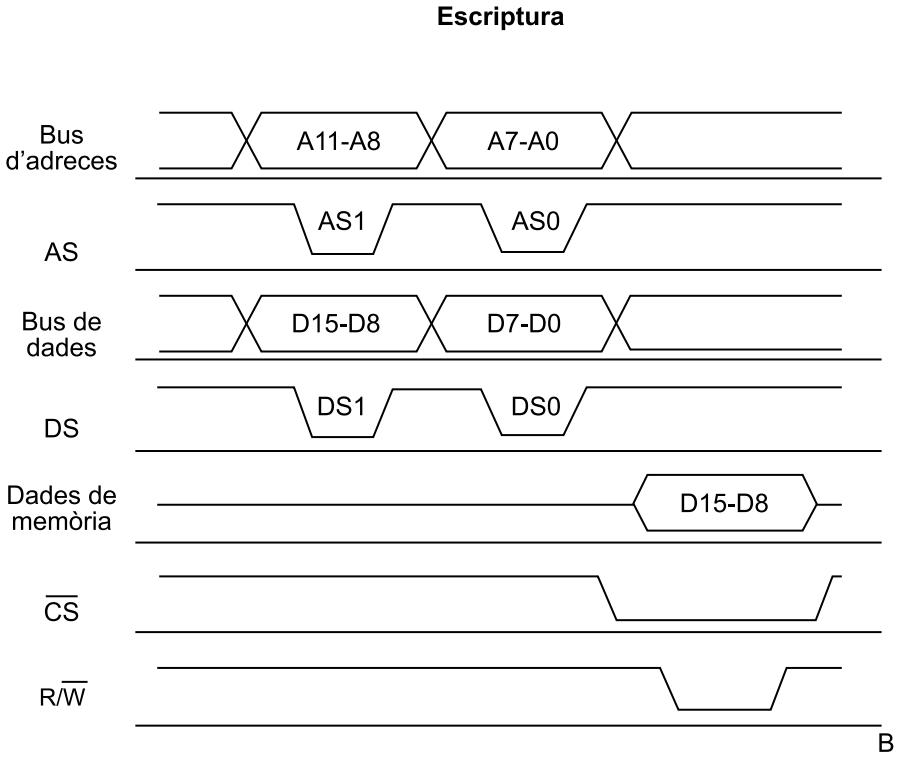


Escriptura a memòria

El procés d'escriptura a memòria es pot resumir en tres passos que segueixen el diagrama temporal de la figura següent. En primer lloc, es carreguen els registres tant d'adreces com de dades. Per a fer-ho, utilitzem els senyals *address strobe* (AS) i *data strobe* (DS) per a indicar quan els senyals estan disponibles en les línies de sortida del processador. Gràcies als bits A11 i A10, llegits en la primera transferència, es calcula el bloc de memòria que cal activar per mitjà del corresponent senyal de *chip select* (CS). Finalment, només cal donar l'ordre d'escriptura per tal d'executar definitivament l'operació. A part, cal mantenir el *buffer* de sortida de dades de les memòries en alta impedància durant tot el procés d'escriptura per tal d'evitar un conflicte en el bus de dades. El lector

pot comprovar en el diagrama de blocs de la figura següent com s'ha tingut en compte aquesta qüestió mitjançant els senyals E1 i E0 del microprocessador, que, tal com veurem a continuació, s'utilitzen per a la lectura de dades en memòria.

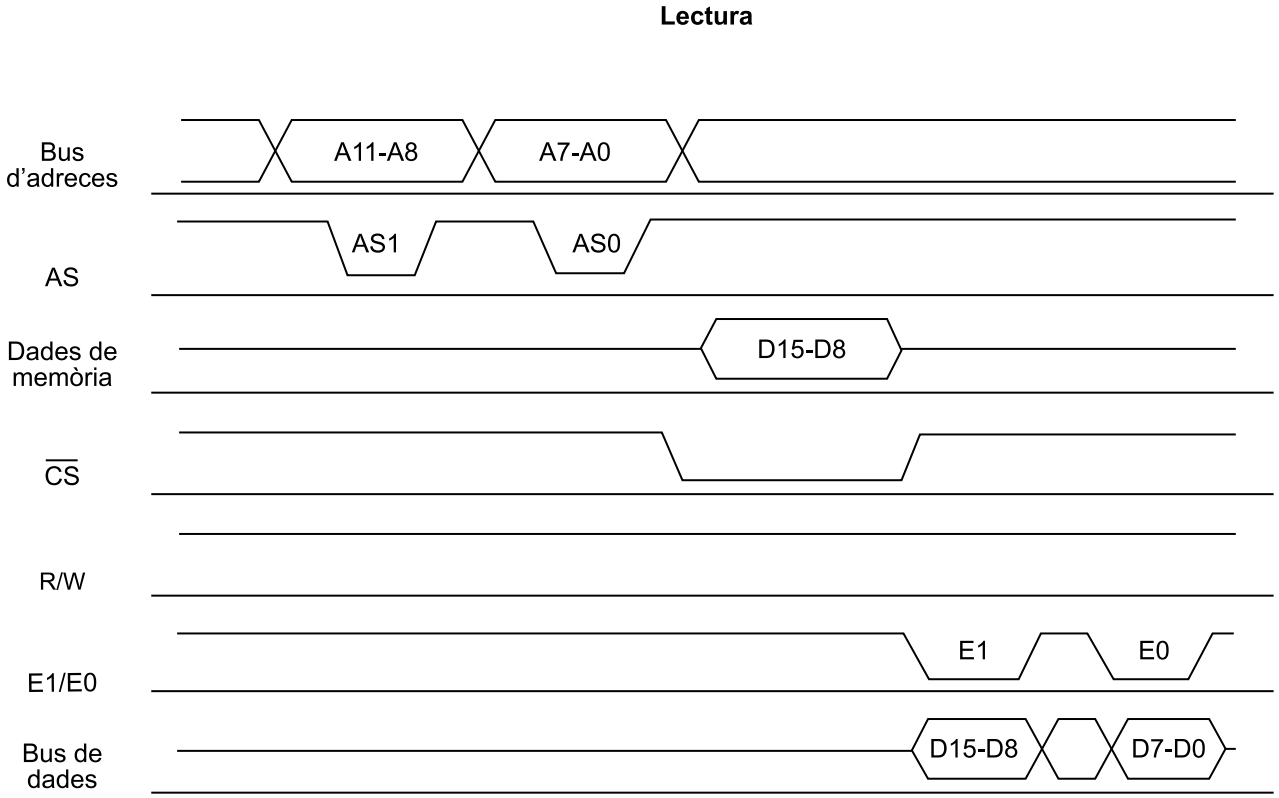
Procés d'escriptura del sistema basat en SRAM



Lectura de memòria

El procés de lectura es troba descrit en el diagrama de la figura següent. En aquest cas, només utilitzem els registres d'adreces com a memòria temporal, ja que els bits de dades viatjaran directament de la memòria al processador en dues transferències de 8 bits cadascuna. Igual que en el cas d'escriptura, cal evitar un conflicte en el bus de dades i, per això, en aquesta ocasió ens hem de preocupar de deixar les sortides dels registres de dades en alta impedància en el moment de la lectura dels bits. Podem dividir el procés de lectura en tres etapes. En primer lloc, es carrega l'adreça volguda als registres corresponents de la mateixa manera que en el cas anterior. Això provoca, en segon lloc, l'habilitació del bloc de memòria on s'ubica la dada, per bé que no està encara disponible ja que els *buffers* de sortida romanen desactivats. Finalment, el senyals d'habilitació que emet el processador (E1 i E0) activen successivament els *buffers* de sortida dels dos mòduls de memòria on es guarda la dada per a acabar-la transferint al microprocessador.

Procés d'escriptura del sistema basat en SRAM



Ja per acabar, cal remarcar que aquest és només un exemple de disseny d'un sistema de memòria SRAM. S'ha procurat fer-lo tan complet com sigui possible, però, òbviament, caldrà estudiar cada cas per separat i tenir en compte les característiques tant del processador com de la memòria amb què treballem.

Exemple

Ens podem trobar que el processador tingui prou línies tant per a adreces com per a dades, per bé que aquesta situació no és l'habitual en sistemes encastats. En l'altre extrem, també ens podem trobar amb un bus per a dades i adreces compartit, que requerirà un disseny semblant a l'exposat, amb un registre per a adreces i un altre per a dades.

En resum, doncs, cal examinar cada cas amb deteniment i adaptar-se a les característiques del maquinari.

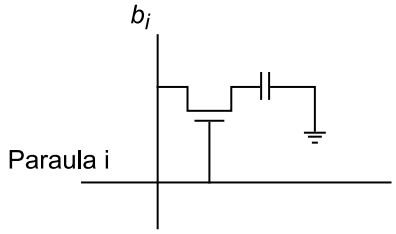
Memòria RAM dinàmica (DRAM)

La cel·la bàsica de memòria DRAM és molt més simple que la de SRAM i es pot veure en la figura següent. Com podem veure, es tracta d'un sol transistor MOS amb els *buffers* d'entrada/sortida corresponents i el mecanisme d'emmagatzematge d'informació és per mitjà de la capacitat paràsita del mateix transistor. D'aquest fet es poden extreure dues característiques importants de les memòries DRAM:

- Permeten integrar quantitats de memòria més grans que les SRAM perquè és menys complexa.

- Com que el mecanisme d'emmagatzematge és volàtil, ja que la càrrega acumulada drena per mitjà de la resistència del transistor, cal renovar contínuament la informació guardada. Per fer-ho, es fa una lectura dels bits per a reescriure'ls immediatament després. Aquesta renovació de la informació es coneix amb el nom de *refrescament* o *cicle de refrescament* i és el fabricant de la memòria qui ha de determinar el temps màxim permès entre refrescaments o intervals de refrescament.

Cel·la bàsica d'una memòria DRAM

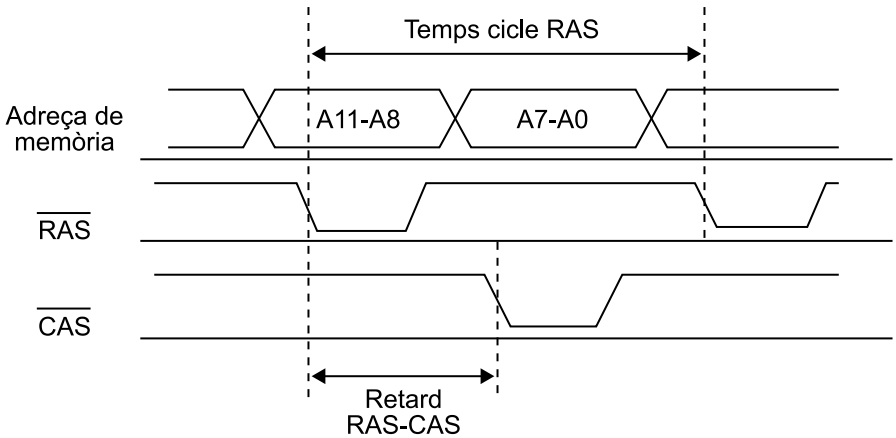


La lectura del bit guardat es fa precarregant b_i a una tensió intermèdia entre massa i la tensió de referència. Després s'activa la porta del transistor i es comparteix així la càrrega emmagatzemada. Per tant, si el valor guardat era un 0, la tensió b_i baixa i si era un 1 la tensió puja. Finalment, cal detectar i amplificar aquesta variació per a portar-la a l'exterior de la memòria. No obstant això, és obvi que el procés de lectura malmet la informació emmagatzemada i és la circuiteria de la mateixa memòria qui s'encarrega de restaurar-la o reescriure-la. L'operació d'escriptura és encara més simple. Només es carrega la línia b_i al valor volgut i s'activa la porta del transistor, amb la qual cosa es carrega o descarrega la capacitat paràsita per a guardar el valor lògic que volem.

Quant a disseny del sistema de memòria, tot el que s'ha vist per a la memòria SRAM es pot traslladar al cas que ens ocupa, per bé que ara hem de tenir en compte algunes consideracions addicionals. La primera és conseqüència de l'empaquetat del xip, ja que moltes vegades no és possible incloure un piu per cada línia o bit d'adreça. La solució en aquests casos és multiplexar els bits d'adreça aprofitant l'estructura matricial de la memòria, és a dir, primer se subministren els bits d'adreça que identifiquen la fila on es troba la nostra paraula i després se subministren els bits que n'identifiquen la columna. Per a gestionar-ho, utilitzem els senyals RAS i CAS que indiquen, respectivament, quan els bits de la fila o de la columna estan disponibles. Associats a aquests senyals, definim els paràmetres temporals següents:

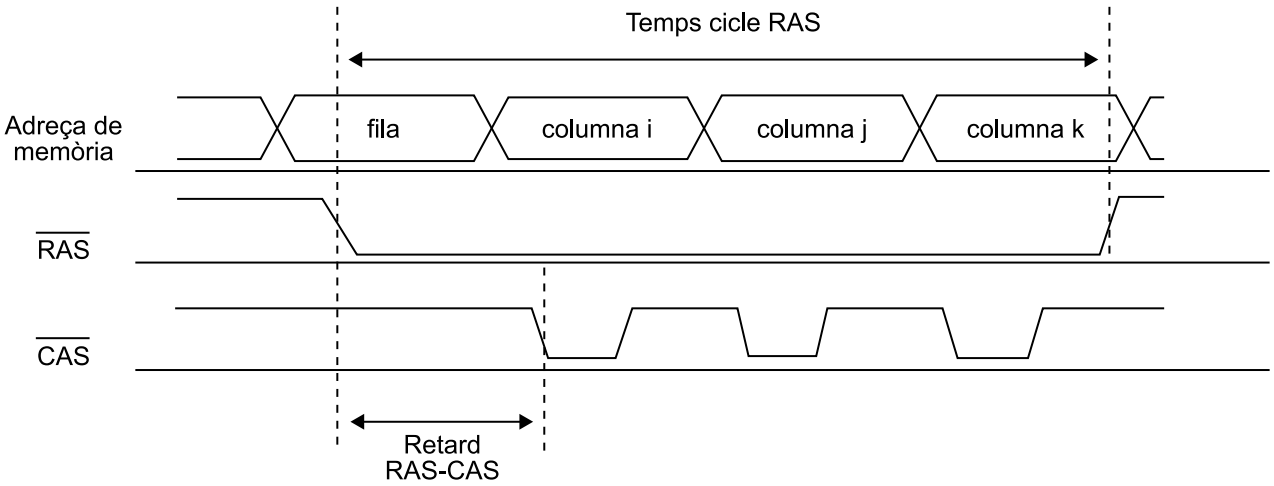
- Temps de cicle del RAS: indica el període d'aquest senyal.
- Retard RAS-CAS: indica la separació que hi ha entre els senyals RAS i CAS.

Temporització dels senyals d'adreça en una memòria DRAM



La figura mostra la senyalització multiplexada d'adreces juntament amb els paràmetres que acabem de definir. Finalment, cal comentar que els diferents tipus de memòria DRAM es distingeixen bàsicament per com s'organitzen internament i per la manera com s'accedeix a les dades. Per posar-ne un exemple, esmentem les memòries DRAM que suporten EDO (*extended data output*) i que permeten accedir a diverses paraules a la vegada. En particular, han de ser paraules que tinguin la part de l'adreça corresponent a la fila en comú i la senyalització emprada és la de la figura següent.

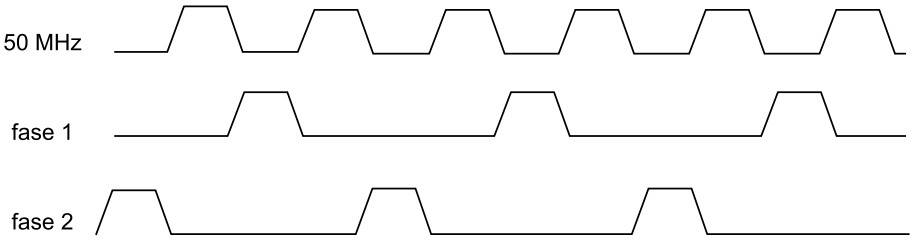
Temporització dels senyals en una memòria DRAM amb EDO



La segona consideració és conseqüència de la necessitat de refrescament de les memòries. Definirem el *període de refrescament* com el període temporal màxim amb què les cel·les de memòria han de ser refrescades per tal d'evitar pèrdues d'informació. Hi ha diverses maneres de refrescar la memòria i cal fer-ho de la manera més eficient possible, ja que el refrescament en si és un treball extra que no ens aporta cap benefici des del punt de vista funcional. Això implica tota una circuiteria addicional que s'encarregui de recórrer les posicions de memòria amb els temps adequats i d'evitar conflictes amb el funcionament normal de la memòria. A continuació, presentem un possible disseny per a una memòria de 4 M paraules de 16 bits cadascuna.

La memòria està organitzada en 4 K files per 1 K columna i és possible refrescar al mateix temps totes les paraules que constitueixen una única fila, amb la qual cosa aconseguim que el refrescament sigui més eficient. En el cas que ens ocupa, assumim que el període de refrescament és de 64 ms i que disposem d'un rellotge de 50 MHz amb dues fases diferents tal com mostra la figura següent.

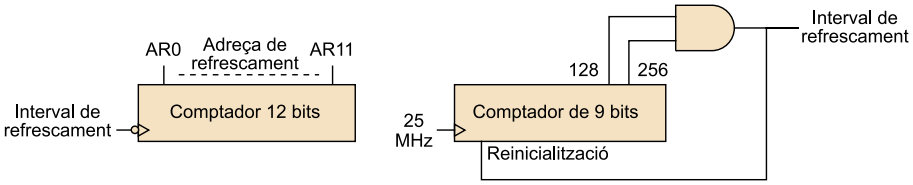
Rellotge de dues fases emprat



En primer lloc, ens ocupem del temps de refrescament. Si tenim en compte les 4 K files de la memòria, veiem que cal refrescar una fila cada 16 µs. Això es pot aconseguir amb un comptador de 9 bits alimentat per una de les fases del rellotge (25 MHz). Fixem-nos que 400 cops de rellotge fan els 16 µs que necessitem. Si volem incloure un cert marge de seguretat i alhora facilitar la implementació, podem considerar 384 cops de rellotge, equivalents a 15,36 µs. Posant una porta AND entre els dos bits de més pes del comptador aconseguim un senyal que es manté a 0 fins que no han transcorregut els 384 períodes de rellotge volguts. Llavors pren el valor 1 i s'hi manté fins que el comptador no dóna la volta, o sigui, quan passa de 511 a 0. Fixem-nos que, si no féssim res més, tindríem un senyal que s'activaria cada 512 cops de rellotge, la qual cosa no ens interessa. Per tant, una solució serà optar per un comptador amb entrada d'inicialització síncrona, que serà alimentada pel senyal de sortida de la porta AND, el mateix que marcarà l'interval de refrescament volgut.

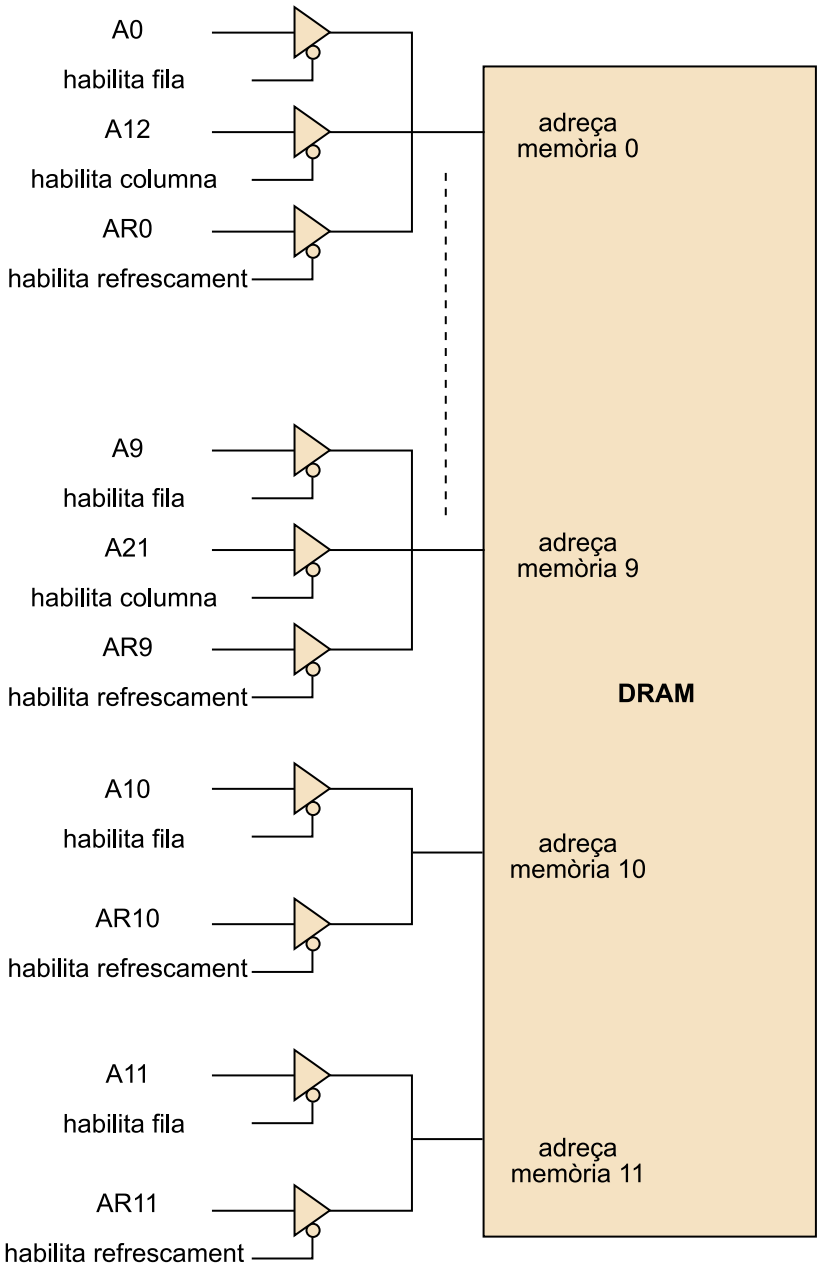
Un cop establert el temps de refrescament, ens cal calcular la posició o posicions de memòria que cal refrescar. En el cas que ens ocupa, on refresquem una fila de memòria cada vegada, un comptador de 12 bits serà suficient per a recórrer totes les posicions, és a dir, les 4 K files. A més a més, activarem el comptador a cada flanc de baixada del senyal interval de refrescament, ja que d'aquesta manera l'adreça que cal refrescar estarà disponible amb tota certesa a l'hora d'executar l'operació. En la figura 34 podem veure tant el generador d'adreces de refrescament com el generador de l'interval de refrescament.

Generador d'adreces de refrescament (esquerra) i de l'interval de refrescament (dreta)



Com que tenim 12 bits per a indicar l'adreça a la memòria DRAM i aquests bits poden tenir fins a tres funcions diferents (fila de memòria per a lectura/escriptura, columna de memòria per a lectura/escriptura o bé fila per a refrescament), utilitzarem *buffers* de sortida per a diferenciar cadascuna d'aquestes funcions i evitar conflictes, tal com es mostra en la figura següent. D'aquesta manera, podrem deixar els *buffers* en estat d'alta impedància quan no s'utilitzin. Això implica el disseny de circuiteria addicional per a generar els senyals de control dels *buffers*.

Control de les adreces (normals i refrescament)



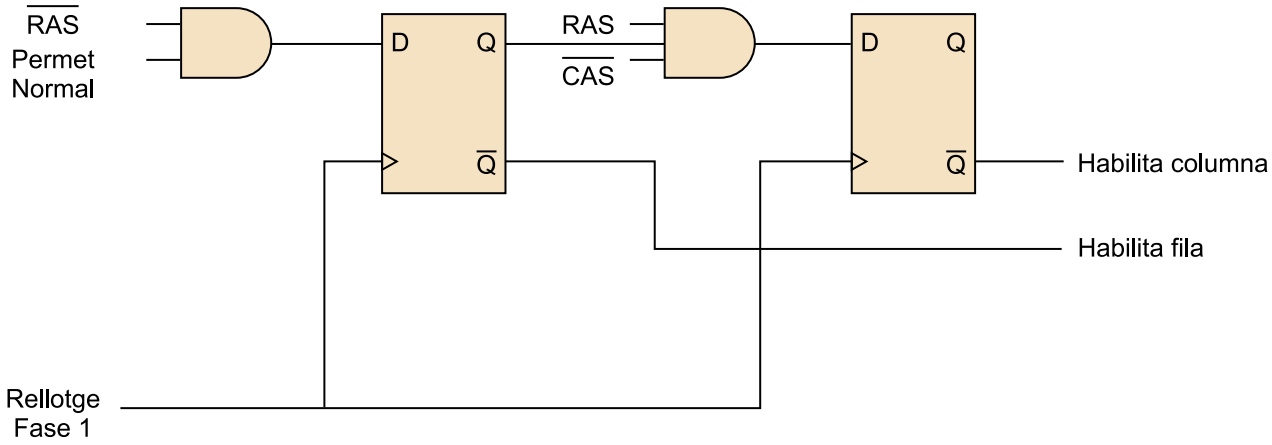
Exemple

En la figura següent, veiem una manera de generar els senyals d'habilitació de fila i columna quan el senyal Permet Normal (que permet les operacions normals de lectura o escriptura) és vàlid. En la mateixa figura, podem veure un diagrama temporal dels senyals que hi intervenen. En aquest exemple, els senyals RAS i CAS provinents del processador estan sincronitzats amb la fase 2 del rellotge, mentre que els senyals d'habilitació que generem són governats per la fase 1. Aquest mecanisme evita interpretar erròniament els senyals prenent algun valor del transitori.

Arribats a aquest punt, hem estat capaços d'adequar la memòria DRAM perquè pugui suportar processos de lectura/escriptura i també de refrescament. No obstant això, encara manca el més important: la gestió temporal d'aquests processos. Fixem-nos que el refrescament de la memòria és una acció més controlada que les operacions de lectura/escriptura habituals, ja que podem predir quines posicions de memòria seran afectades en cada moment. En canvi, l'accés normal pot ser més arbitrari, i deixar oberta la possibilitat de voler accedir a memòria i refrescar al mateix temps. Per tal d'evitar aquest tipus de conflictes, haurem d'incloure un mòdul d'arbitratge. Una possible definició funcional d'aquest és la següent:

- Quan comença una operació de lectura/escriptura, es deixa acabar.
- Si una operació de refrescament ha començat, es deixa acabar tot recordant l'operació normal.
- Si les dues operacions coincideixen, es dóna prioritat a l'operació normal.

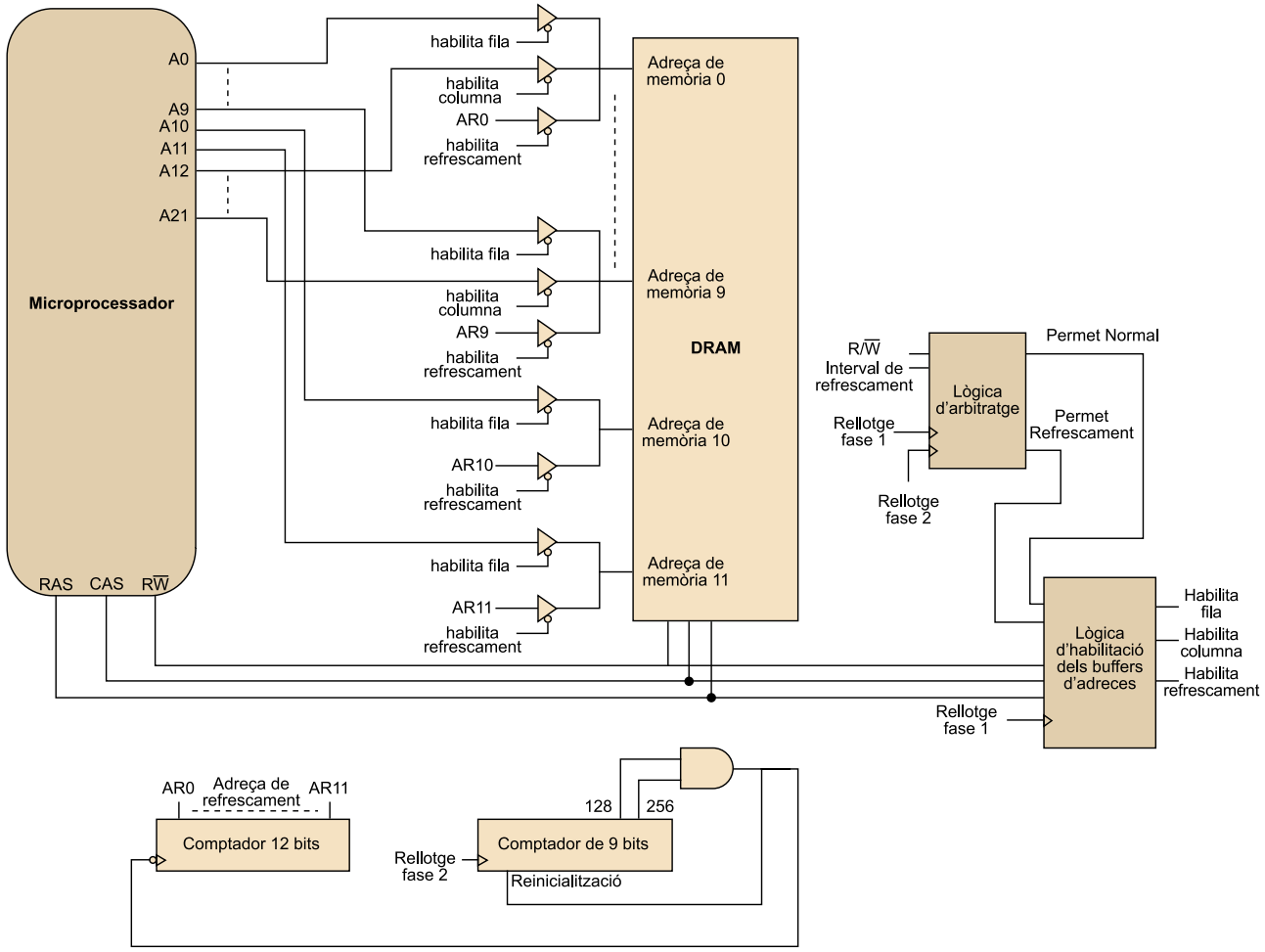
Exemple de generació dels senyals d'habilitació dels *buffers* d'adreça



Finalment, la figura següent mostra el disseny complet del sistema de memòria basat en DRAM, en què usem els senyals següents:

- **Lectura-escriptura**: indica que s'ha iniciat un procés normal de lectura/escriptura.
- **Permet Normal**: indica que el mòdul d'arbitratge permet les operacions de normals de lectura/escriptura.
- **Permet Refrescament**: indica que el mòdul d'arbitratge permet les operacions de refrescament.

Sistema de memòria DRAM complet



2.5.3. Sistema de memòria

En un sistema encastat s'acostumen a combinar memòries de diferents tipus que finalment donen lloc al seu sistema de memòria. Hi haurà una part de memòria ROM en què guardarem les aplicacions i una part de memòria RAM que utilitzarem a l'hora d'executar-les. Així mateix, dividirem la part que correspon a memòria volàtil en tres tipus diferents que alhora són formats per memòries de característiques diferents. A continuació, les presentem de mida més gran a més petita i hi trobem:

- **Memòria secundària**: és opcional en sistemes encastats i és formada per grans memòries amb velocitats d'accés moderades. El cost també és l'inferior de les tres.

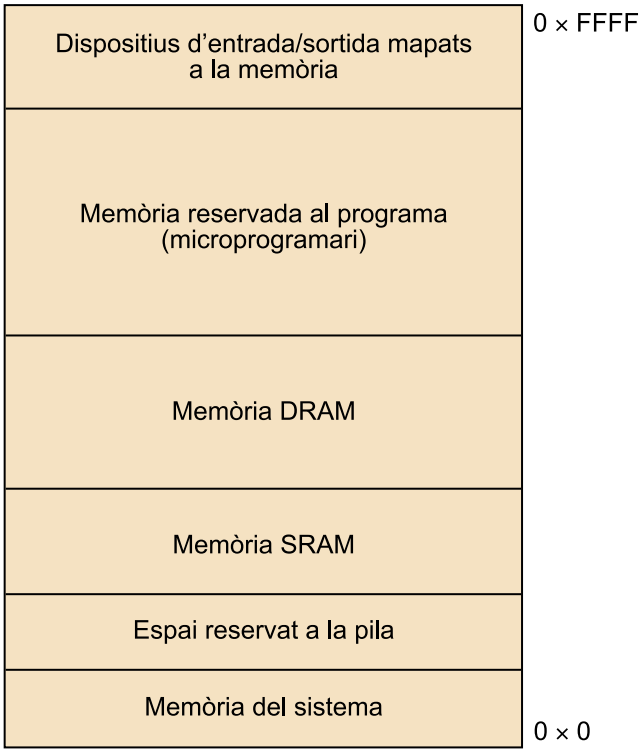
- **Memòria principal o primària**: de mida més petita, és constituïda per memòries més ràpides i també més cares. Típicament són memòries DRAM o bé SRAM de baixa velocitat.

- **Memòria cau**: és la part més petita i ràpida del sistema de memòria. També n'és la més costosa. És formada per memòries SRAM d'alta velocitat.

La lògica que hi ha darrere d'aquesta configuració del sistema de memòria respon al comportament de la majoria d'aplicacions. Així, doncs, hi ha una petita quantitat de dades a les quals s'accedeix d'una manera molt més continuada que a la resta en un moment determinat. Per tant, aquest disseny s'acomoda a aquesta situació i afavoreix la velocitat d'execució del programari si s'ubiquen les dades d'ús més freqüent en la memòria cau i la resta en la memòria principal. També es tracta d'una solució efectiva en termes de cost, ja que es manté la capacitat del sistema de memòria i es penalitza poc en velocitat d'execució respecte a un sistema equivalent que integri solament memòries d'alta velocitat. A vegades, també es consideren els mateixos registres del microprocessador en una categoria addicional que podríem anomenar **memòria cau interna**.

Associat al sistema de memòria, que acabem de comentar i més a nivell lògic, hem de considerar el **mapa de memòria** de qualsevol sistema encastat. S'hi inclouen totes les adreces físiques existents en el sistema i, a part de la zona dedicada a memòria volàtil, comentada anteriorment i que conté tant dades de l'usuari com del sistema, també considerem la memòria de tipus ROM en què es guarden les instruccions de programa o els dispositius d'entrada/sortida mapats a memòria, que van des d'una pantalla o un visor a un disc dur amb accés directe a memòria (DMA). En la figura següent podem veure un exemple de mapa de memòria per a un sistema de 16 bits:

Mapa de memòria en un sistema encastat



2.5.4. *Caching*

Anteriorment, hem vist els diferents tipus de memòria que integren un sistema encastat i també com l'ús de diferents tipus de memòria RAM respon al patró d'execució típic d'una aplicació. Des de ja fa temps, se sap que la majoria d'aplicacions s'executen o bé de manera seqüencial o bé en petits llaços de codi (zona de referència seqüencial), cosa que podem interpretar com una petita finestra d'activitat que es va desplaçant al llarg del codi del nostre programa. A més a més, la velocitat de desplaçament d'aquesta finestra és molt més lenta que la velocitat d'accés de les memòries més ràpides.

Feta aquesta observació, la idea d'agafar blocs de dades de la memòria principal i traslladar-los a una memòria petita i ràpida (la cau) té molt sentit, ja que durant una estona el processador podrà interactuar amb la memòria cau, molt més ràpida, i s'estalviarà l'accés a la memòria principal, més lenta. Aquesta és la idea del *caching*.

El primer que ens cal per a poder fer *caching* és establir una zona de referència on se centri l'activitat del programa durant un interval de temps determinat. Si no és així, el mecanisme resultant no serà vàlid perquè no podrem evitar l'accés continuat a la memòria principal, i s'alentirà així l'execució del programa. Trobem dues maneres principals de definir la zona de referència:

- **Zona de referència espacial**: suposem que en un futur immediat s'accedirà a dades que es troben físicament properes a les utilitzades en el temps present.

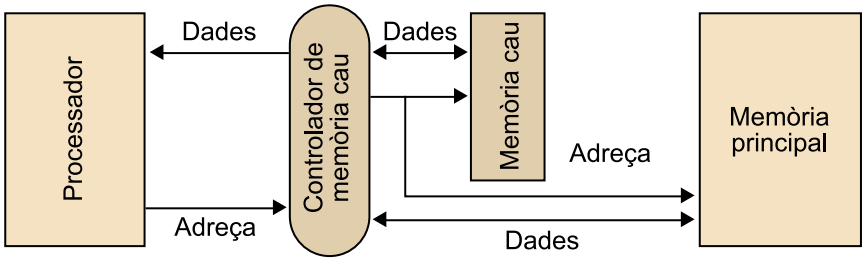
- **Zona de referència temporal**: suposem que les dades utilitzades en el temps actual poden ser reutilitzades en instants futurs propers.

Vegeu també

Un cas particular i conegut de zona de referència espacial és la que hem comentat al principi d'aquest mateix subapartat, altrament coneguda com a **zona de referència seqüencial**, en què l'execució s'esdevé de manera seqüencial i en petits llaços de codi.

Un cop sabem que es donen les condicions per tal que el *caching* sigui efectiu, haurem d'implementar el sistema de *caching* seguint l'arquitectura que es mostra en la figura següent:

Arquitectura del subsistema de *caching*



Com veiem, ens caldrà un controlador encarregat d'executar l'algorisme de *caching* i que farà d'intermediari entre el processador i el sistema de memòria. El controlador serà l'encarregat de decidir quan i com es fan els traspassos de dades entre la memòria cau i la principal. Es tracta, per tant, d'un element crucial de cara al rendiment del sistema.

En temps d'execució, quan el processador necessita alguna dada o instrucció, mira primer la memòria cau. Si la informació es troba allà, diem que hi ha hagut un *cache hit*, s'agafa la informació i es continua l'execució. En cas contrari, diem que hi ha hagut un *cache miss* i hem d'anar a la memòria principal a buscar la informació. Llavors, s'agafa el bloc de dades on hi ha la informació i es trasllada a la cau. Si hi ha lloc, es guarda directament i, si no, s'ha de crear esborrant un bloc existent. Abans de fer-ho, però, cal comprovar si hi ha hagut modificacions en les dades d'aquell bloc i guardar-les en la memòria principal en cas afirmatiu.

A partir d'aquí, cal decidir com es porten a terme totes aquestes operacions, és a dir, com sabem si un bloc està en cau o no, com seleccionem el bloc que s'ha d'esborrar o com sabem si la informació ha estat modificada. A continuació, veurem tres algorismes de *caching* diferents que donen resposta a aquestes preguntes. En tots tres casos, tindrem en compte les especificacions següents del sistema de memòria:

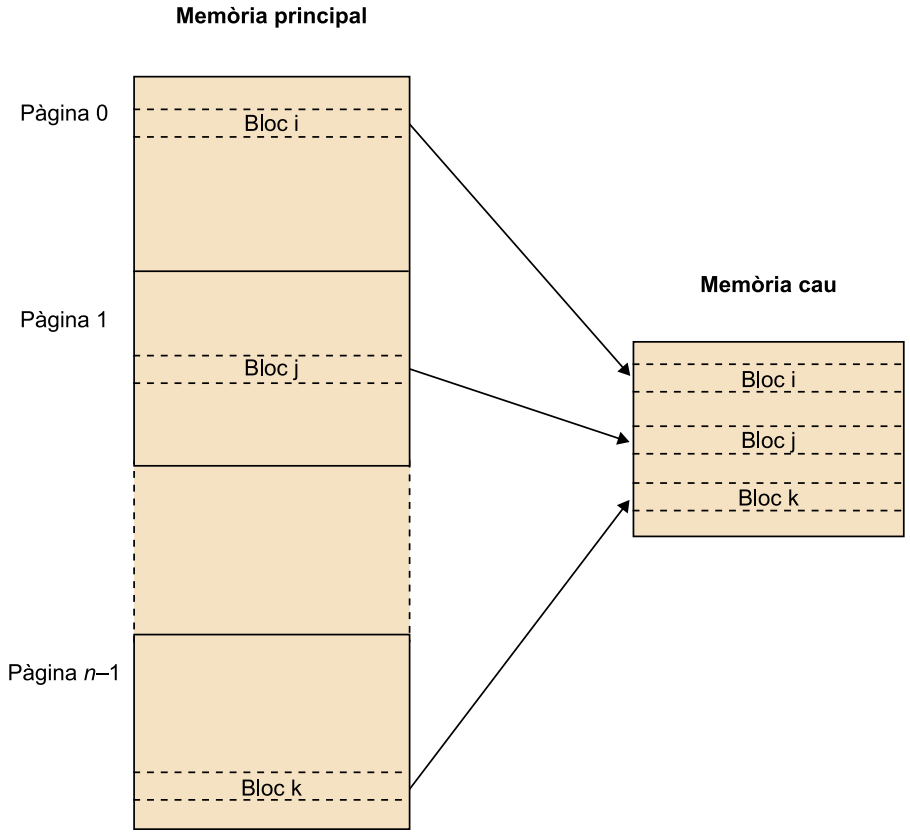
- Es treballa amb paraules de 32 bits d'amplària.
- La cau permet emmagatzemar 64 K de paraules.
- Cada bloc de dades conté 512 paraules i, per tant, la cau permet emmagatzemar 128 blocs.
- El bus d'adreces té una amplària de 32 bits, suficient per a direccionar les 128 M de paraules que conté la memòria principal.
- La memòria principal està dividida en 2 K de pàgines i tenint en compte que la mida del bloc està fixada en 512 paraules, cada pàgina conté 128 blocs. Per tant, una pàgina conté 64 K de paraules.

Implementació amb mapatge directe

Aquesta implementació fixa la mida de la memòria cau igual a la de la pàgina en la memòria principal per tal de poder ubicar directament un bloc en cau dins la pàgina en memòria corresponent. És a dir, independentment de la pàgina de memòria d'on provingui, el *k*-èsim bloc de la *i*-èsima pàgina en memòria principal es trobarà sempre en el lloc corresponent al *k*-èsim bloc en la memòria cau (vegeu-los en la figura següent). Aquest fet facilita molt el

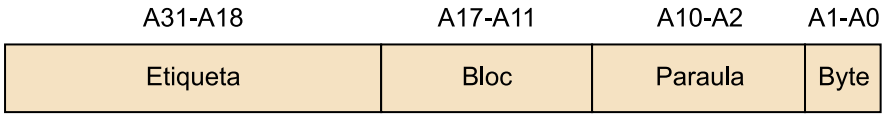
moviment de dades, ja que només ens hem de preocupar per saber a quina pàgina de memòria correspon cada bloc de dades que hi ha dins la cau. La resta és implícit gràcies al mapatge directe.

Sistema de *caching* amb mapatge directe



Si tenim en compte que la memòria principal conté 128 M paraules, 27 bits són suficients per a poder distingir-les. Si, a més a més, volguéssim distingir entre els 4 bytes que formen una paraula, ens farien falta 2 bits addicionals. Suposem aquest darrer cas (29 bits en total). D'aquests, els 2 últims identifiquen el byte dins la paraula, els 9 següents la paraula dins del bloc de dades (512 paraules per bloc) i els 7 següents identifiquen el bloc dins la pàgina (128 blocs per pàgina) tal com es pot veure en la figura següent:

Interpretació de l'adreça des del punt de vista del sistema de *caching*



Fixem-nos que quan el processador vol accedir a una dada determinada, el valor que prenen aquests bits és el mateix tant si anem a buscar la dada directament a memòria com a la cau. En cas d'anar-la a buscar a la cau, però, haurem de comprovar que estem accedint a la rèplica de la pàgina desitjada, és a dir, haurem de veure si els 11 bits següents coincideixen o no.

Una observació

Fixeu-vos que els 3 bits de més pes no s'utilitzen i, per tant, resten a 0.

Aquesta informació addicional es guarda en una estructura especial per a aquest propòsit: la **taula d'etiquetes**. Conté una entrada per a cadascun dels blocs que integren la cau (128 en el nostre cas) i s'hi guarda la mateixa etiqueta, que identifica la pàgina d'on prové el bloc, el **bit de validesa**, que indica si les dades del bloc són vàlides, i el ***dirty bit***, que indica si s'han modificat les dades del bloc. Tot i que no s'utilitza en aquest algorisme, també es pot guardar en la taula un camp temporal, que indica quan s'ha portat el bloc a la cau o quan s'ha usat per últim cop.

Amb tots aquests ingredients, el mecanisme de *caching* és el següent. Inicialment, tenim tots els bits de validesa i els *dirty bit* a FALSE. A partir d'aquí, quan el processador necessita dades es mira si ja es troben en cau i es comproven l'etiqueta i el bit de validesa corresponents. En cas afirmatiu, s'empren les dades. En cas negatiu, es van a buscar a la memòria principal, sempre copiant tot el bloc de dades on es troba la paraula d'interès. Prèviament a la còpia, però, cal verificar si la posició en cau és vàlida (bit de validesa). Si no ho és, es continua amb l'operació, però si és vàlida cal comprovar que no hi hagi hagut modificacions en les dades de la cau (vegeu el *dirty bit*). En cas afirmatiu, copiem prèviament el bloc de cau a la memòria principal per tal de mantenir la informació actualitzada. Aquest mètode d'actualització a nivell de bloc s'anomena *escriptura retardada* i es fonamenta en el fet que si una dada s'ha escrit una vegada, pot tornar a ser escrita al cap de poc temps i, per tant, múltiples escriptures a memòria alentirien el sistema. No obstant això, es pot optar també per l'**escriptura directa**, és a dir, actualitzar la memòria principal cada cop que hi ha un canvi a la cau per a mantenir-les *coherents* en tot moment.

Implementació amb mapatge associatiu

Si optem per un algorisme de *caching* amb mapatge directe facilitem la implementació del sistema a causa de la seva simplicitat. No obstant això, aquesta tècnica pot resultar ineficient en termes de velocitat d'execució.

Exemple

Imaginem un llaç de codi que utilitzi dues dades que es troben al bloc 4 de memòria però en pàgines diferents. És evident que durant l'execució del programa haurem d'anar modificant contínuament el bloc 4 de la cau quan seria més eficient tenir els dos blocs sempre en cau. El mapatge associatiu soluciona aquest problema i permet col·locar un nou bloc de dades en qualsevol posició de la memòria cau.

El funcionament és semblant a l'anterior, per bé que ara cal decidir quin és el bloc de la cau que cal substituir quan es va a buscar una nova dada a la memòria principal. A més a més, el sistema de cerca i identificació es fa més complex, ja que treballarem amb etiquetes més grans que inclouran tant els bits que identifiquen la pàgina com el bloc (18 bits en el nostre cas). Habitualment, es fa una cerca en paral·lel per a accelerar el procés, però això afegeix complexitat i cost al sistema. Fixem-nos també que el mapatge entre l'adreça que proporciona el processador i l'adreça en cau no han de coincidir com abans i, per tant, n'haurem de fer la conversió.

Finalment, a l'hora de decidir quin és el bloc que s'ha de substituir, podem optar per diverses solucions, que poden fer ús de les marques temporals existents en la taula d'etiquetes. Dos dels algorismes més usats són:

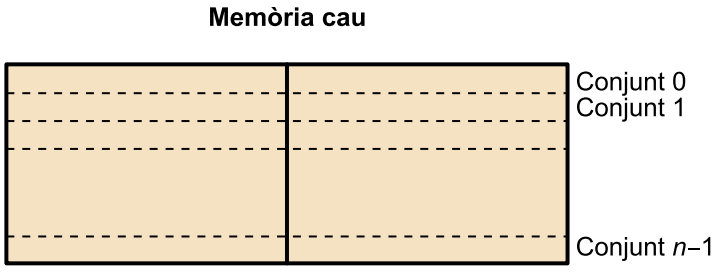
- ***Least recently used*** **(LRU)**: si pensem que el bloc més antic és el que té menys probabilitats de ser tornat a usar, reemplaçarem el bloc més antic en cau.

- ***Most recently used*** **(MRU)**: si, pel contrari, creiem que l'últim bloc usat és el que té menys probabilitats de tornar a ser necessari, optarem per aquesta darrera solució.

Implementació amb mapatge associatiu dins d'un conjunt de blocs

La darrera solució que veiem és a mig camí entre el mapatge directe i l'associatiu i pretén aprofitar-se dels avantatges de cadascuna de les tècniques, és a dir, de la flexibilitat del mapatge associatiu i de la simplicitat del mapatge directe. Per aconseguir-ho, introduïm una nova organització de la memòria cau. Si abans cada entrada de la taula d'etiquetes identificava un bloc de la memòria principal, ara cada fila de la taula serveix per a assenyalar un conjunt de blocs. En el cas que ens ocupa, i optant per conjunts de dos blocs, tindrem una cau de 64 conjunts de 2 blocs tal com mostra la figura següent.

Organització de la cau usant mapatge associatiu dins un conjunt de blocs

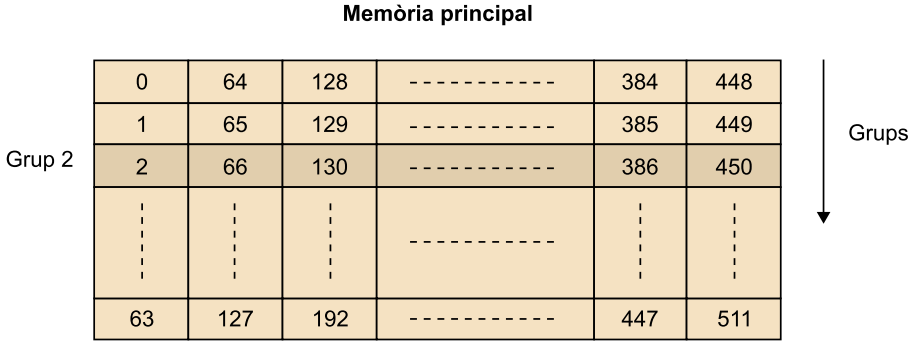


El pas següent és ordenar la memòria principal d'acord amb aquesta organització de la cau. El que fem és dividir la memòria en tants grups com conjunts hi ha en la cau (64 en el nostre cas). Per determinar quins blocs de dades pertanyen a cada grup, calculem el mòdul de l'índex del bloc sobre el nombre de conjunts.

Exemple

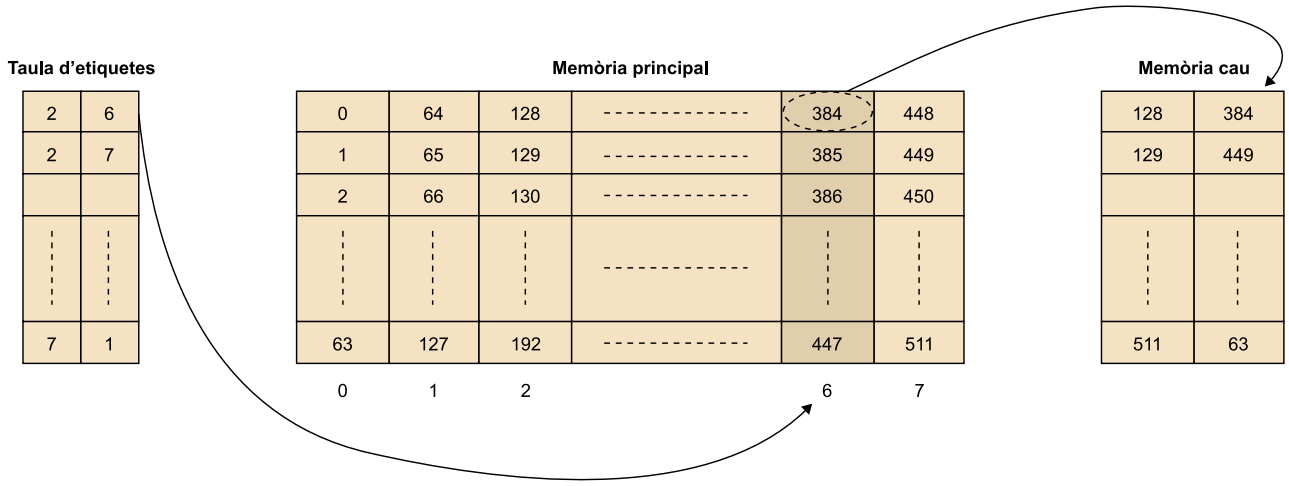
Per exemple, el bloc 453 pertany al grup 5, ja que 453 mod 64 = 5. I dins de cada grup, els blocs que hi pertanyen s'ordenen de més petit a més gran. Aquesta organització de la memòria principal en el nostre cas, on tenim fins a 256 K blocs, es pot veure en la figura 43.

Organització de la memòria principal usant mapatge associatiu dins un conjunt de blocs



Per tant, podrem guardar a la memòria cau fins a 2 blocs de cada grup (d'un total de 4 K). A l'hora de buscar un bloc en cau, sabem que hi ha un mapatge directe entre grup de memòria i grup de cau, mentre que la posició dins del grup queda reflectida en la taula d'etiquetes i s'ha de cercar de manera associativa, tal com mostra la figura 44. L'avantatge és que ara l'espai de cerca és més petit (12 bits en l'exemple) i per tant menys complex.

Funcionament del mapatge associatiu amb conjunts de blocs



2.6. Registres

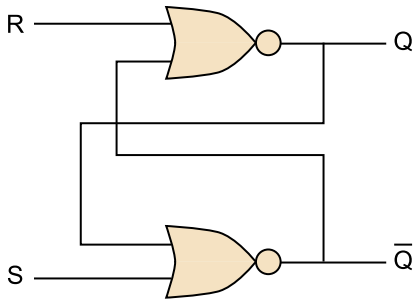
Anteriorment hem parlat i utilitzat els registres. Hem vist que poden ser interns al processador, on tenen un paper important per tal que aquest pugui dur a terme les seves tasques, i també externs. Nosaltres els hem usat per a adaptar el bus d'adreces del processador a un determinat bloc de memòria DRAM multiplexant cada adreça del bus en dues parts: fila i columna. De moment, sabem que són memòries petites (d'uns quants bits d'amplària) i ràpides. Aquests registres tenen la funció d'emmagatzematge de dades, però hi ha també els registres de desplaçament i els *linear feedback shift register* (LFSR).

2.6.1. Registres d'emmagatzematge

Si aprofundim una mica més en aquesta part del maquinari, veurem que la peça bàsica de qualsevol registre és un biestable (capaç de mantenir un estat de dos possibles), ja sigui de tipus *latch* o bé de tipus *flip-flop,* i que ens permet guardar un sol bit d'informació. L'agrupació de més d'un biestable és el que forma els registres i habitualment tenen una mida que és potència de 2, és a dir, acostumem a trobar registres de 4, 8, 16, 32 o 64 bits d'amplària. Hem vist en les SRAM un *latch* molt senzill format per dos inversors. El problema amb aquesta configuració és que la lectura i escriptura del bit no és fàcil: cal incloure-hi els transistors de pas i treballar amb les tensions adequades. El *latch* més simple que ens permet interactuar-hi d'una manera més fàcil és el *latch* Reset Set (RS), que correspon al circuit lògic de la figura següent. El funcionament és el que aquí indiquem: si les entrades R i S són 0, la sortida Q no canvia i per tant serveix per a llegir el bit que emmagatzema el *latch*. En el moment en què una

de les entrades pren valor lògic 1, la sortida és 0 si s'ha activat l'entrada R i 1 si s'ha activat l'entrada S. Aquesta és la manera d'escriure un bit al *latch*. En cas que les dues entrades siguin 1, el valor de sortida és indeterminat.

Latch RS



Per tant, un *latch* és en principi un dispositiu asíncron, és a dir, els canvis a la sortida es produeixen com a conseqüència dels canvis a l'entrada.

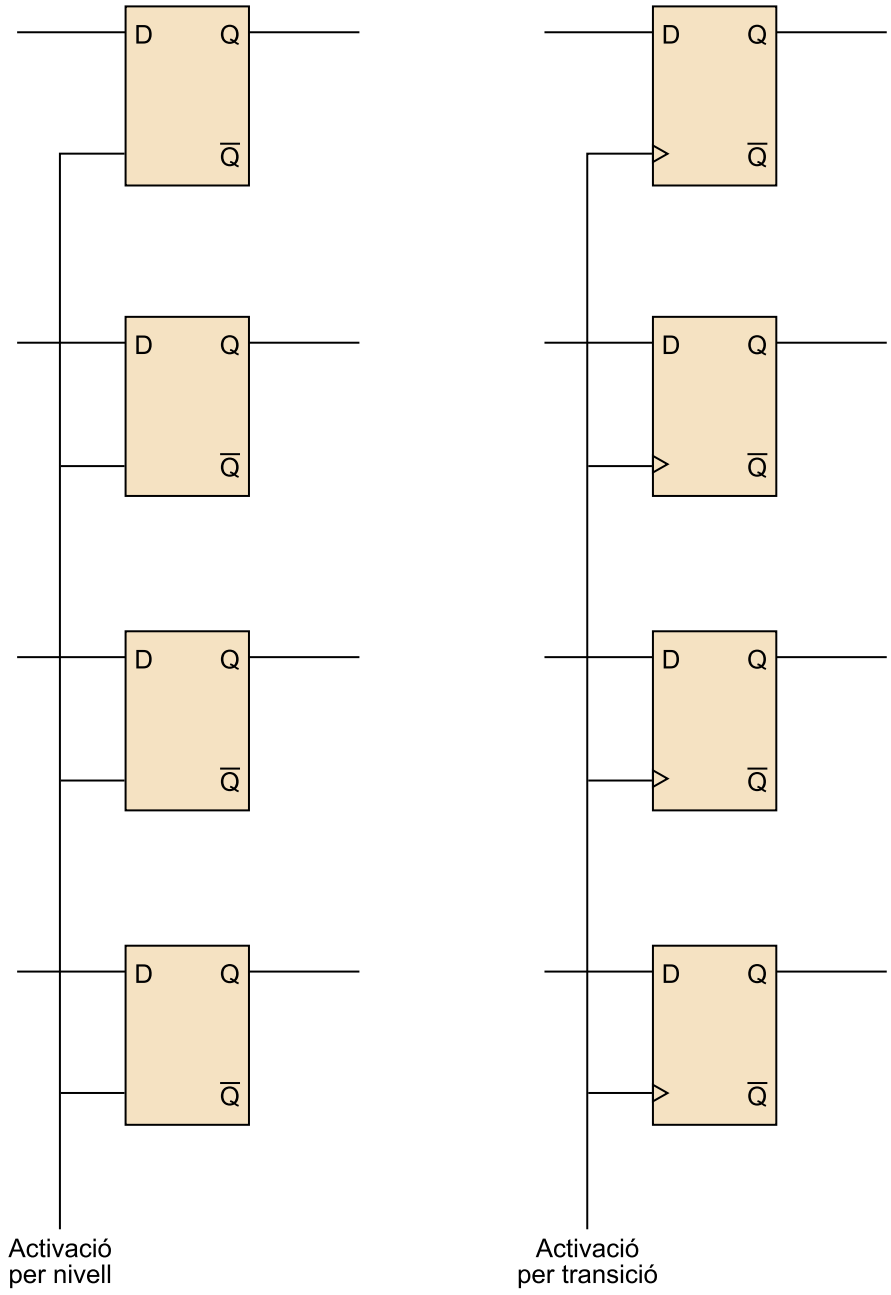
No obstant això, habitualment requerirem cert tipus de sincronisme, encara que no sigui amb el rellotge del sistema. Per exemple, en el disseny anterior de sistema de memòria basat en DRAM hem utilitzat els senyals AS i DS per a indicar quan cal capturar en els corresponents registres d'adreça i de dades els bits a l'entrada. Modificant el *latch* bàsic de la figura precedent, afegint algunes portes lògiques podem aconseguir el comportament volgut, és a dir, un dispositiu que emmagatzema el bit a l'entrada segons un senyal d'habilitació.

En aquest cas, diem que la sincronització és per nivell perquè l'operació d'escriptura es fa quan el senyal d'habilitació pren un valor determinat (alt o baix depenent de la implementació). Aquest *latch* és conegut com a *D latch*, en què la *D* prové de *delay*, ja que es tracta d'un dispositiu que retarda el senyal d'entrada.

La segona opció és utilitzar *flip-flops*, que són els elements bàsics dels sistemes digitals seqüencials, més complexos que els anteriors *latches*. De fet, un *flip-flop* de tipus D és format essencialment per dos *latches* de tipus RS connectats en cascada, encara que la seva funcionalitat sigui gairebé idèntica al D *latch*. La diferència és subtil però extremament important: mentre que el *latch* s'activa per nivell, el *flip-flop* s'activa per flanc (de pujada o baixada segons la implementació). A més a més, és un dispositiu molt més robust, ja que quan el senyal d'habilitació (pot ser el rellotge) es troba a nivell alt, es guarda el bit d'entrada en el primer *latch* però **no es transmet** al segon (sortida del *flip-flop*), cosa que no passa fins que el senyal d'activació o rellotge passa d'1 a 0 (si se sincronitza per flanc de baixada). Per tant, l'entrada i la sortida es troben aïllades, o sigui que les variacions en el senyal d'entrada no passen a la sortida. En la figura següent podem veure un registre de 4 bits implementat amb *latches* (esquerra) o bé amb *flip-flops* (dreta). Fixem-nos que són pràcticament iguals i que treballen amb el mateix senyal d'activació o rellotge. L'única diferència entre els

dos rau en el triangle que hi ha en els *flip-flop*, que precisament serveix per a indicar la sincronització per flanc. Opcionalment, hi pot haver altres senyals comuns com el senyal de Reset si ens interessa poder inicialitzar el registre.

Registre de 4 bits usant *latches* (esquerra) o bé *flip-flops* (dreta)

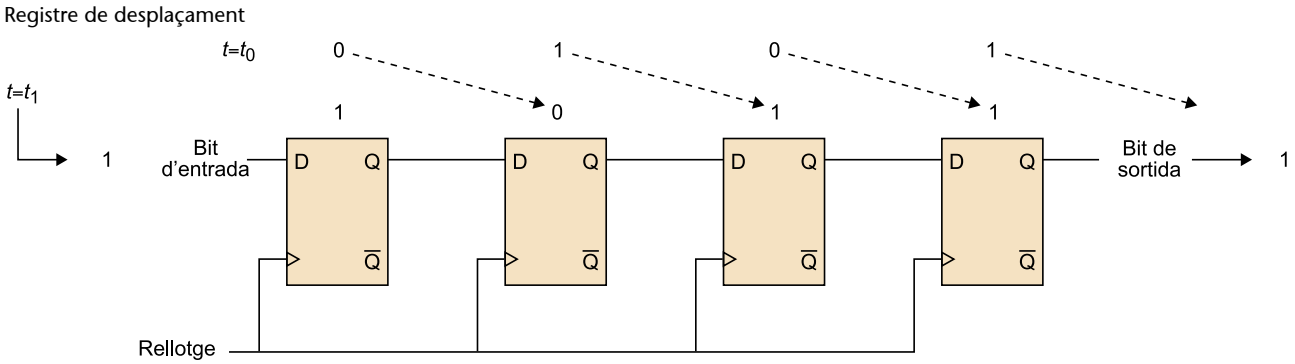


2.6.2. Registres de desplaçament

En determinades ocasions cal transformar dades que tenim en paral·lel a sèrie o bé a l'inrevés per a adaptar-les, per exemple, a un port o bus determinats. En aquest cas, haurem d'utilitzar els registres de desplaçament, que tal com el seu nom indica, permeten desplaçar els bits emmagatzemats. Un desplaçament de

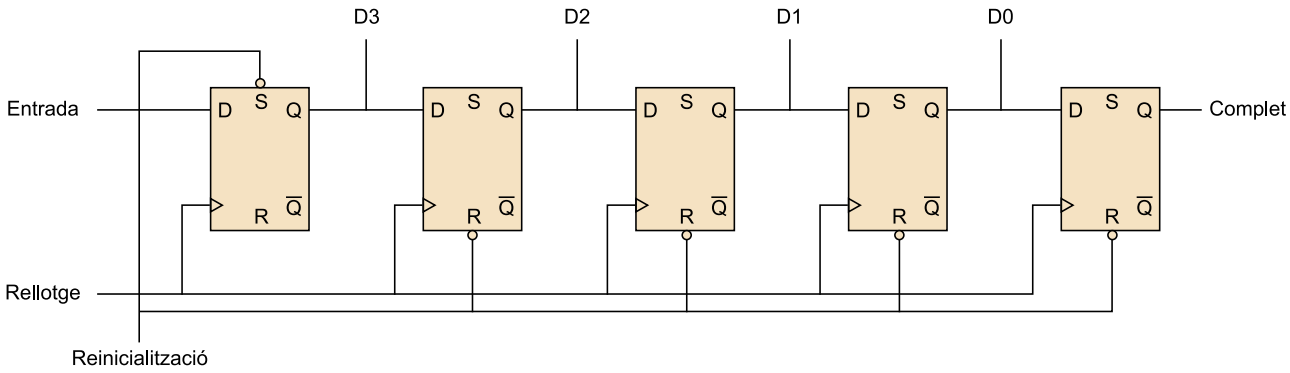
bits pot servir, també, per a fer multiplicacions o divisions per potències de 2, o bé per a generar retards molt ben controlats que poden ser útils en certes aplicacions.

Considerem primer el registre de desplaçament amb entrada sèrie i sortida sèrie. Aquests tipus de registres es construeixen fàcilment connectant *N flip-flops* en cascada amb un rellotge comú. Per exemple, la figura següent mostra un registre de desplaçament de 4 bits. El funcionament és senzill: suposem que en l'instant $t = t_0$ els *flip-flops* emmagatzemen els bits 1010 i que el bit d'entrada és 1. Per tant, la sortida a t_0 és 0 i les entrades dels biestables són 1101, respectivament. Passat un flanc de rellotge, en l'instant t_1, aquests seran els valors emmagatzemats pels *flip-flops* i, per tant, el que veurem a la sortida serà 1. És a dir, el bit 1 ha entrat al registre desplaçant la cadena 101 a la dreta, i ha tret el bit 1 i eliminat el bit 0 anterior, tal com mostra la figura següent. Aquest comportament es repetirà a cada flanc de rellotge la qual cosa provocarà el desplaçament d'un bit cada vegada. A part, hem de tenir en compte que si volem inicialitzar el registre a tot 0, haurem de col·locar un senyal de Reset comú a tots els biestables. És important remarcar que aquest tipus de registre no funcionaria amb *latches* a causa de la transparència que hi ha entre entrada i sortida i, per tant, en un mateix període, el bit d'entrada es podria propagar a més d'un biestable.



Aquest mateix registre de desplaçament ens serveix per a implementar un registre d'entrada en sèrie i sortida en paral·lel. Simplement, hem de llegir les sortides dels biestables al mateix temps. Opcionalment, si cal multiplexar aquests bits en un bus comú, situarem *buffers* de sortida de tres estats a cadascuna de les sortides dels *flip-flops*. D'aquesta manera, podrem posar el registre de desplaçament en estat d'alta impedància quan no es llegeixin els bits guardats. No obstant això, ens caldrà comptar els cops de rellotge transcorreguts per tal de saber quan queda ple el registre. Per evitar-ho, podem modificar aquest disseny afegint-hi un biestable més, el qual serveix per a marcar si el registre és ple. Aquesta variant es pot veure en la figura següent:

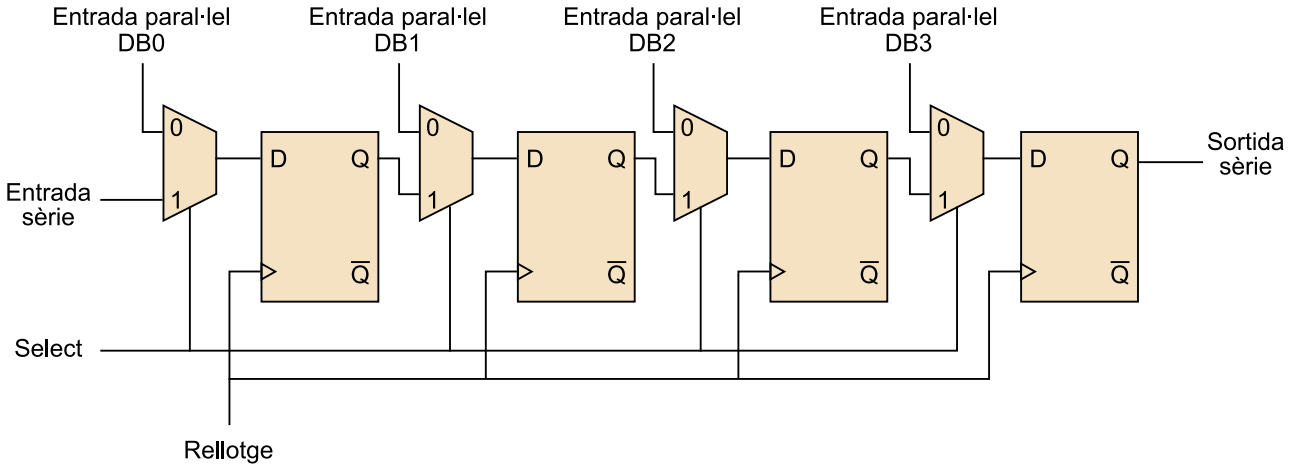
Registre de desplaçament com a convertidor sèrie a paral·lel amb bit de marca



Fixem-nos que abans d'entrar la paraula al registre, cal activar la línia Reset. Això fixarà a 1 la sortida del primer biestable i a 0 la resta. A partir d'aquí, es comença a introduir la paraula. El senyal de complet es mantindrà a 0 fins que el registre estigui ple, moment en el qual podem fer la lectura.

Finalment, veiem el registre de desplaçament d'entrada en paral·lel i sortida en sèrie, que podem copsar en la figura següent. En aquest cas, cal situar un multiplexor de dues entrades i una sortida a cada entrada de biestable, tots controlats per un senyal de selecció. En cas que el senyal prengui valor lògic 1, el registre funciona de manera habitual, desplaçant un bit a cada cop de rellotge. En canvi, quan el senyal pren valor 0, carreguem els bits d'entrada en paral·lel als biestables corresponents, operació que es fa efectiva en el flanc de rellotge següent.

Registre de desplaçament com a convertidor paral·lel a sèrie



2.6.3. Registres LFSR

Els registres *linear feedback shift register* (LFSR) són registres de desplaçament on l'entrada al registre és una combinació lineal dels bits emmagatzemats a cadascun dels biestables. Per tant, no hi ha una entrada externa al registre de desplaçament. A més a més, la combinació lineal no és qualsevol, ja que els valors que multipliquen cadascuna de les variables d'estat són o bé 0 o bé 1, és a dir, l'entrada és una suma *or exclusiva* d'alguns dels bits que guarden els

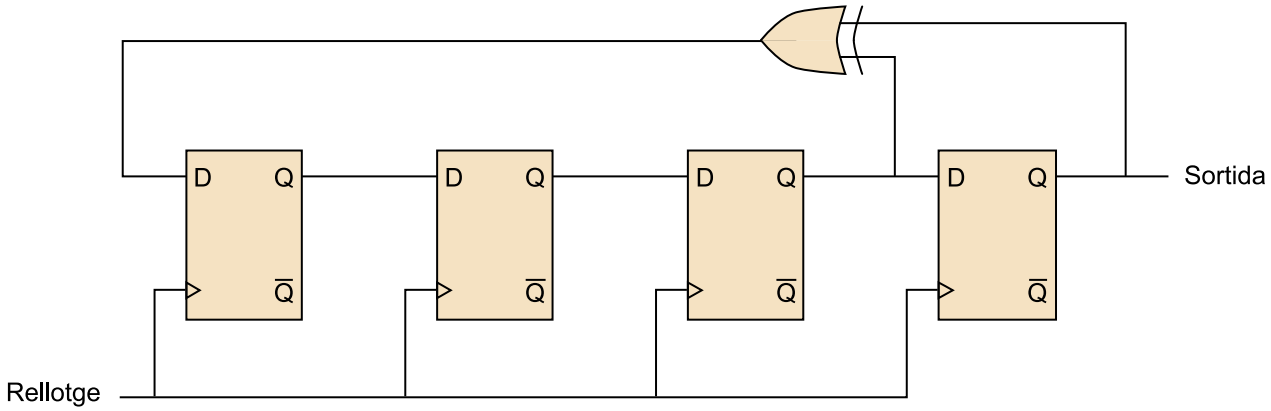
biestables. Per tant, si el registre de desplaçament s'inicialitza a l'estat de tot 0, la realimentació serà sempre 0 i l'estat es mantindrà de manera indefinida. Òbviament, aquesta no és la missió del registre.

A la pràctica, aquests dispositius s'utilitzen per a obtenir seqüències pseudoaleatòries. Per a descriure quines variables intervenen en la realimentació s'utilitza un polinomi C(x) que marca les connexions. Aquest polinomi té grau N, és a dir, el nombre de biestables, en què el terme de grau més alt hi és sempre present, ja que marca l'entrada al registre (la realimentació). A part, almenys un dels altres coeficients també ha de ser 1 per tal d'agafar almenys un dels bits del registre.

Exemple

Per exemple, la figura següent mostra un registre de quatre elements amb el polinomi de connexions $C(x) = 1 + x + x^4$. Evidentment, la seqüència resultant no és totalment aleatòria ja que arribarà un moment en el qual repetirem l'estat del registre (el nombre d'estats és finit) i a partir d'allí la seqüència es repetirà. Sabem que la seqüència de longitud màxima que podem obtenir abans que es comenci a repetir és de $2^N - 1$. Aquestes seqüències s'obtenen sempre amb polinomis de connexions primitius, és a dir, que no es poden factoritzar. Finalment, cal comentar que les seqüències resultants dels LFSR s'acostumen a anomenar seqüències *pseudo noise* (PN), ja que tenen la forma de soroll aleatori. Entre les seves aplicacions, a part de la que acabem d'esmentar, hi ha la de generar vectors aleatoris per a comprovar el funcionament correcte dels sistemes, encriptació, codificació (per exemple, en sistemes d'espectre eixamplat), o sincronització.

Registre LFSR



2.7. Comptadors i divisors

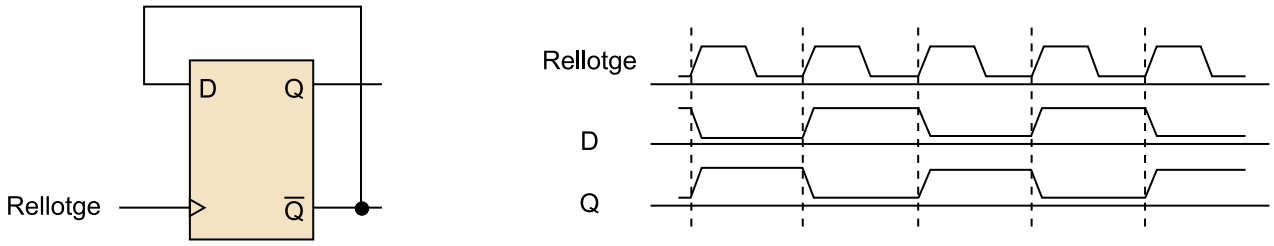
Les tasques de comptar i dividir formen part de molts sistemes encastats i han estat essencials en moltes de les seves aplicacions. Ens interessa comptar per a acumular esdeveniments, comptar bits o determinar si quelcom ha succeït un nombre de cops determinat. També podem modificar un comptador de manera fàcil per tal que compti d'acord amb un esdeveniment que es repeteix periòdicament cada cert temps (un rellotge), de manera que s'obté un temporitzador. Els temporitzadors ens serviran per a mesurar el temps entre dos esdeveniments determinats, crear un compte enrere, etc. Finalment, emprarem els divisors per a generar, a partir d'un senyal de referència, un altre de freqüència

inferior. De vegades, serà el mateix processador qui podrà dur a terme aquestes tasques i d'altres haurem d'integrar aquestes funcionalitats dins el nostre sistema per mitjà de circuits integrats especialitzats.

2.7.1. Divisors i comptadors asíncrons

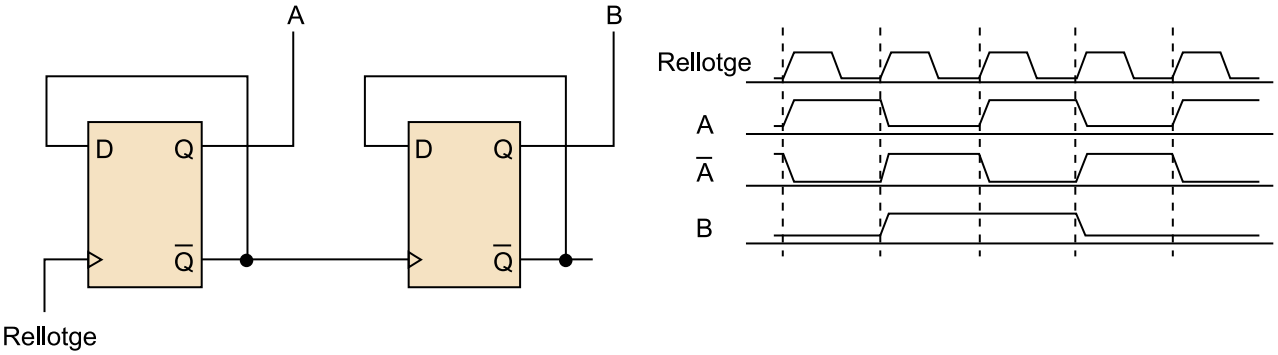
El divisor més senzill que podem construir necessita només un *flip-flop* de tipus D i s'obté connectant la sortida negada a l'entrada tal com es pot veure en la figura següent (esquerra). D'aquesta manera, a cada cop de rellotge actualitzem el valor emmagatzemat pel seu negat i, com podem veure en la figura següent (dreta), el senyal resultant a la sortida té la meitat de freqüència que el rellotge de l'entrada.

Divisor simple per 2



Aquest disseny es pot ampliar connectant més biestables en cascada tal com es mostra en la figura següent (esquerra) pel cas de dos *flip-flops*. Fixem-nos que la realimentació d'un biestable s'usa com a senyal de rellotge per al següent. Així, el funcionament del segon biestable és exactament igual al del primer, però els canvis es produeixen a una freqüència que és la meitat que la del rellotge del primer. Per tant, el senyal de sortida del segon *flip-flop* haurà reduït la freqüència original quatre vegades tal com es pot veure en la figura que es mostra a continuació (dreta):

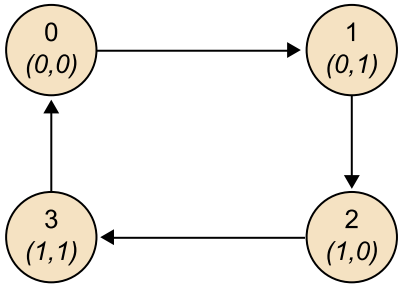
Divisor asíncron per 4



Fixem-nos ara en la seqüència de valors {(B,A)} que es va obtenint de manera periòdica, que és {(B,A)} = {(0,0), (0,1), (1,0), (1,1)}. Veiem que es tracta clarament d'un comptador de 2 bits. Si l'analitzem com una màquina d'estats on l'estat és (B,A), obtenim el diagrama d'estats de la figura següent. Diem que es tracta d'un comptador asíncron, ja que els biestables no comparteixen un mateix senyal de rellotge, sinó que la sortida d'un és rellotge de l'altre. Això fa

que hi hagi un retard de propagació que s'incrementa biestable per biestable i que pot ser significatiu depenent del nombre de biestables i del retard entre l'entrada de rellotge i la sortida en cadascun d'ells. Es tracta, per tant, d'un efecte tipus dòmino que s'ha de tenir en compte si volem descodificar els bits del comptador, ja que patirem el risc de variacions a la sortida a causa de la falta de sincronització entre les sortides dels biestables. En general, és millor evitar aquesta pràctica.

Diagrama d'estats del comptador asíncron de 2 bits



2.7.2. Divisors i comptadors síncrons

Si volem evitar problemes de transitoris no desitjats, no ens queda cap altre remei que usar comptadors síncrons, on tots el biestables treballen amb un senyal de rellotge comú. Els dissenyarem de la mateixa manera que qualsevol circuit de lògica seqüencial.

Exemple

Vegem tot seguit el comptador de 2 bits. En primer lloc, és clar que necessitarem dos *flip-flops* per a poder emmagatzemar els dos bits, que anomenarem A i B. Fixem-nos que el diagrama d'estats que volem generar és el de la figura anterior i, per tant, podem construir la taula de veritat de les entrades als biestables, és a dir, D_A i D_B en funció dels valors de les variables d'estat A i B. Representem aquesta informació en forma de mapa de Karnaugh en la figura següent i usant el mètode del mateix nom obtenim les funcions lògiques que indiquem a continuació:

$$D_A = \overline{A}$$
$$D_B = \overline{A}B + A\overline{B} = A \oplus B$$

Mapes de Karnaugh de les entrades als biestables D_A i D_B

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

D_A

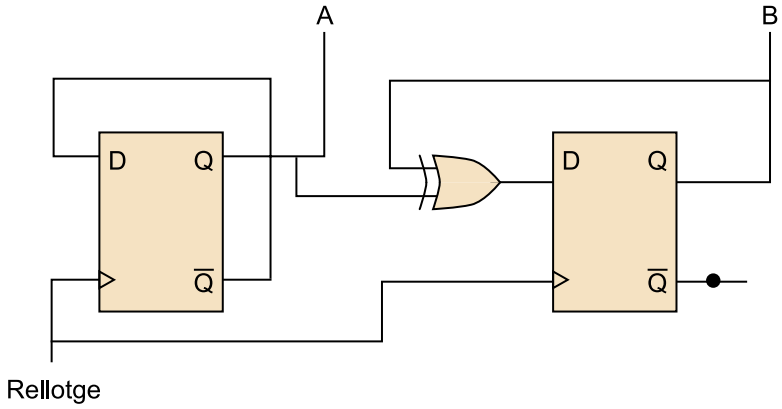
A \ B	0	1
0	1	1
1	0	0

D_B

A \ B	0	1
0	0	1
1	1	0

Això ens dóna finalment el comptador síncron de 2 bits de la figura 55 i, seguint el mateix procediment, podrem obtenir qualsevol comptador que necessitem.

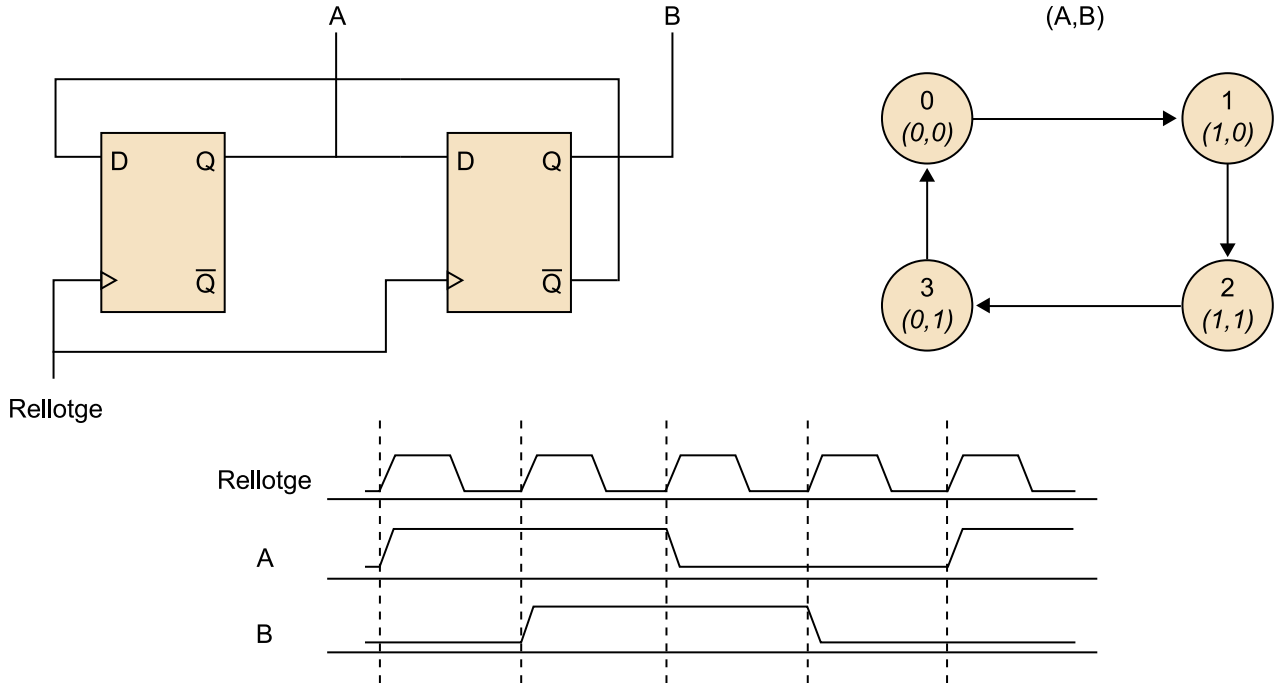
Comptador síncron de 2 bits



2.7.3. El comptador de Johnson

Els comptador de Johnson és un cas particular de comptador que és especialment útil a l'hora de dissenyar bases temporals per als sistemes encastats. El disseny és molt simple: es tracta només d'un registre de desplaçament on es realimenta l'entrada amb la sortida, tal com podem veure en la figura següent per al cas de 2 bits. En la mateixa figura, trobem tant el seu diagrama d'estats com l'evolució temporal de les variables d'estat A i B. Fixem-nos que el comptador de Johnson canvia una de les sortides a cada cop de rellotge i, per tant, per entendre'l pròpiament com a comptador, hem d'interpretar els estats segons la codificació de Gray. Fixem-nos també que la sortides A i B es troben totes dues a la meitat de freqüència que el rellotge però amb fases diferents.

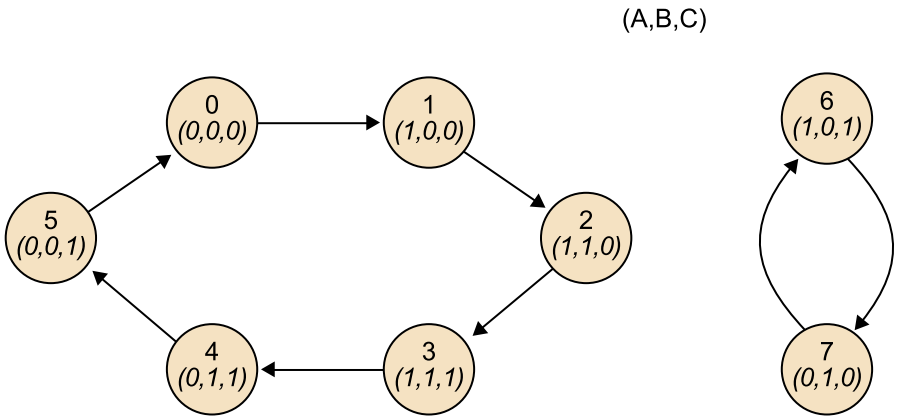
Comptador de Johnson de 2 bits (circuit, diagrama d'estats i resposta temporal)



En els comptadors de més de 2 bits s'ha de tenir en compte una particularitat, per la qual cosa ens fixem ara en el comptador de 3 bits. En cas que el comptador s'inicialitzi a l'estat (A,B,C) = (0,0,0) el comportament és semblant

a l'anterior però recorrent sis estats diferents en lloc de vuit. Els dos estats que queden es van alternant indefinidament, de manera que si el comptador entra en un d'aquests dos estats per accident (soroll o altres causes), el comptador deixa de funcionar com nosaltres volem. Per això és important inicialitzar-lo bé quan el sistema arrenca i fins i tot evitar que pugui caure en un estat no volgut controlant les entrades als biestables. En cas que el funcionament sigui correcte, però, podem veure en el diagrama d'estats de la figura següent que cadascuna de les variables proporciona un senyal de freqüència igual a un sisè de la del rellotge i amb tres fases diferents.

Diagrama d'estats del comptador de Johnson de 3 bits



En general, un comptador de Johnson de N bits té un període de $2N$ en el funcionament correcte, mentre que la resta d'estats $2^N - 2N$ formen un subgraf d'estats il·legals que cal evitar. La seqüència resultant és sempre de tipus Gray i, per tant, ens ajudarà si volem descodificar l'estat, ja que no només es tracta d'un comptador síncron sinó que, a més, només una de les entrades canvia a cada cop de rellotge.

2.8. Rellotge del sistema

El rellotge és un element bàsic de qualsevol sistema encastat i de qualsevol circuit digital, ja que és qui en marca el ritme de funcionament. Per exemple, entre dos cops de rellotge sempre hi ha d'haver el temps suficient per a dur a terme les operacions establertes. Els paràmetres que hem de tenir en compte a l'hora d'escollir el rellotge són, bàsicament:

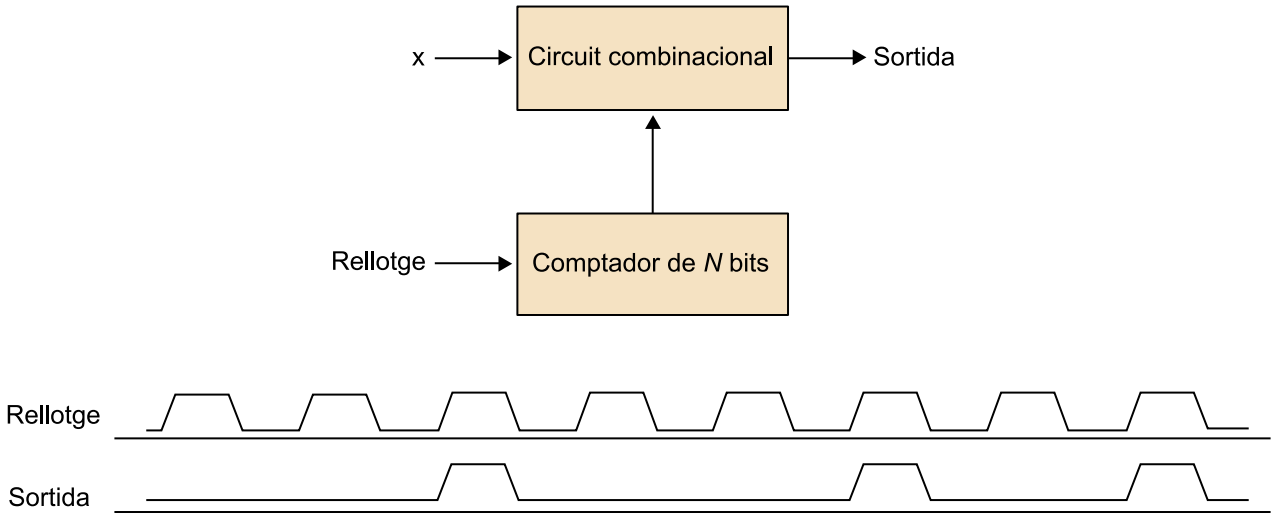
- Freqüència,
- estabilitat i precisió.

A l'hora d'escollir la freqüència del rellotge, el més habitual és que sigui igual o bé superior a la necessària. En el segon cas, podrem utilitzar circuits més o menys senzills que en permetran reduir la freqüència fins a la volguda. Hem vist anteriorment que els comptadors es poden utilitzar com a divisors de freqüència. Hem comentat que, en el cas d'utilitzar comptadors asíncrons, no convé descodificar les sortides del comptador perquè el retard entre els dife-

rents biestables pot causar transitoris no volguts (*gltiches*) que, en darrer terme, poden provocar un funcionament inadequat del sistema. A la pràctica, s'utilitzen divisors asíncrons de fins a 12-14 bits, la qual cosa permet reduccions en freqüència de fins a 16 K. No obstant això, hem de tenir en compte el retard que el senyal de baixa freqüència experimenta respecte del rellotge principal.

En definitiva, sabem com dividir la freqüència per un nombre que sigui potència de 2 o bé múltiple de 2 utilitzant els comptadors (síncrons o asíncrons) o bé els comptadors de Johnson, respectivament. Ara bé, també és possible dividir la freqüència per un nombre fraccionari, la qual cosa s'aconsegueix amb un comptador de N bits i un circuit combinacional en què l'entrada també és un nombre de N bits i la sortida és el senyal de freqüència reduïda, tal com mostrem en la figura següent, suposant $N = 3$ i un factor de reducció de 3/8.

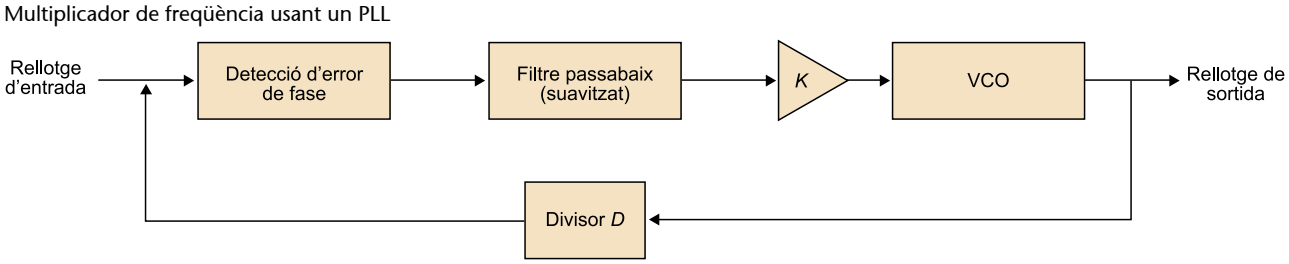
Divisió de la freqüència de rellotge per un nombre fraccionari



En general, si utilitzem un comptador d'N bits, podrem aplicar factors de reducció del tipus $x/2^N$, en què x és el nombre, expressat en binari, que situarem a l'entrada del circuit. Per acabar, caldrà dissenyar aquest circuit combinacional per tal que la seva resposta sigui 1 només en x valors que prengui el comptador, els quals s'han d'escollir uniformement distribuïts entre 0 i 2^N - 1, tal com veiem en l'exemple de la figura.

Finalment, si el que ens cal és augmentar la freqüència de l'oscil·lador, haurem d'implementar un circuit tipus llaç de seguiment de fase (PLL, *phased locked loop*), l'esquema del qual es pot veure en la figura següent, en la qual la freqüència de sortida es multiplica pel valor del divisor de freqüència situat en el llaç de retroalimentació. Per a entendre el funcionament del PLL, considerem primer que la retroalimentació és directa, és a dir, dividim per la unitat. En aquestes circumstàncies, el senyal de sortida es compara amb el d'entrada, és a dir, el rellotge del sistema. La diferència de fase resultant és filtrada passabaix, és a dir, se suavitza aquest senyal d'error, per a atacar (aplicant-hi el guany adequat K) un VCO, és a dir, un oscil·lador controlat per tensió. El circuit,

doncs, segueix el senyal d'entrada i s'hi manté sincronitzat en fase. Si en el llaç de retorn afegim un divisor de freqüència de valor *D*, forcem que el senyal a la sortida del VCO tingui una freqüència *D* vegades superior a la del rellotge, ja que és l'única manera perquè el pugui copiar un cop travessi el divisor.



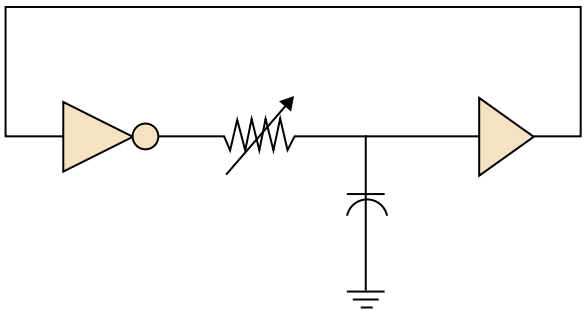
En termes d'estabilitat i precisió, l'elecció del rellotge del sistema dependrà dels requisits de cada aplicació. L'estabilitat determina com canvia la freqüència del rellotge en el temps, mentre que la precisió ens indica com s'ajusta a la seva freqüència nominal. En aplicacions senzilles pot ser suficient un rellotge basat en ressonadors RC (resistència-condensador). En canvi, en aplicacions més exigents, com pot ser el sistema de comunicacions d'un sensor sense fil, necessitarem una base temporal precisa i que no fluctuï en el temps. La majoria de sistemes utilitza cristalls com a ressonadors, com és el cas del popular rellotge de quars. En el cas de necessitar més precisió i estabilitat s'acostuma a estabilitzar els cristalls en temperatura amb petits escalfadors. Hi ha altres tipus de rellotges, com són els basats en microsistemes (MEMS, *microelectromechanical system*) o bé els coneguts rellotges atòmics. El principal avantatge d'utilitzar un ressonador basat en MEMS és que podem integrar el rellotge en la pastilla de silici del nostre integrat i, de cara al futur, es presenta com un competidor seriós als cristalls de quars. Els MEMS, però, tindrien característiques semblants a les de cristalls. En canvi, si volem un rellotge més precís i estable, l'alternativa són els rellotges atòmics. Són els que s'usen, per exemple, en els satèl·lits GPS. La seva aplicació en sistemes encastats encara haurà d'esperar a causa del seu cost, mida i consum, però ja hi ha hagut intents de sintetitzar rellotges atòmics en un xip.

2.8.1. Algunes consideracions pràctiques

En la majoria de casos, optarem per ressonadors basats en cristalls, ja que d'entrada obtindrem una base temporal més estable i precisa. A més a més, ens assegurarem que en utilitzar aquesta base temporal en una còpia del nostre sistema, el funcionament continuï sent el volgut (repetibilitat del disseny). En cas d'usar rellotges basats en ressonadors RC, haurem d'utilitzar dissenys contrastats i comprovar que compleixin els requisits del nostre sistema. Evitarem, per exemple, un disseny com el de la figura 60, que pot arribar a tenir un comportament metastable (diferents punts de ressonància). Aquest com-

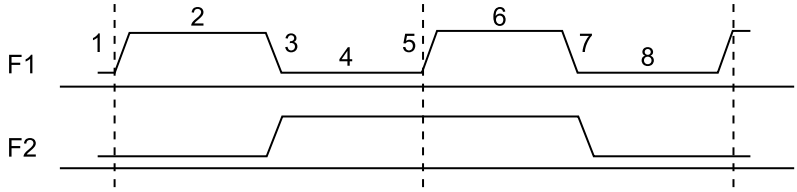
portament es produeix perquè el bloc RC afecta significativament els temps de pujada i baixada del *buffer*, els quals no han estat pensats per a treballar en aquestes condicions.

Disseny RC perillós



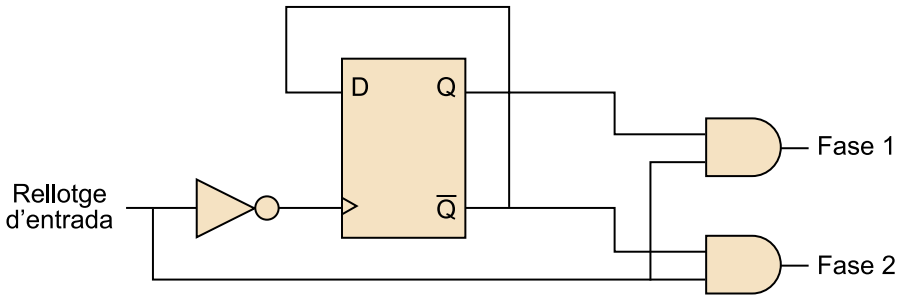
En alguns casos ens interessa tenir una precisió temporal millor que la que ens proporciona el rellotge del sistema. Fixem-nos que en un rellotge normal tenim quatre punts de decisió com a molt, que són: flanc de pujada, flanc de baixada, nivell alt i nivell baix. Si utilitzem un rellotge de dues fases, els punts de decisió s'amplien a vuit, ja que la segona fase del rellotge ens permet distingir els quatre punts de decisió anteriors en dos cicles consecutius de la primera fase del rellotge, tal com veiem en la figura següent. Fixem-nos que els punts 2 i 6 no es poden distingir amb una sola fase (F1), però sí que és possible si tenim en compte la segona fase (F2). En el cas d'utilitzar rellotges de múltiples fases, procurarem sempre derivar les distintes fases a partir d'una mateixa base, ja que d'aquesta manera es mantindran sempre síncrones.

Punts de decisió en un rellotge de dues fases



Hem vist un exemple d'ús del rellotge de dues fases en l'exemple anterior de disseny DRAM. En general, podem usar una fase per a marcar els moments de canvi en el sistema i l'altra per a guardar-ne els resultats. Això ens permet donar prou temps als diferents circuits lògics per a estabilitzar els seus valors, afegint robustesa al sistema. Per exemple, aquesta és una bona opció si treballem amb registres de tipus *latch*, mentre que no és necessari emprant registres basats en *flip-flops*. Una manera senzilla d'obtenir un rellotge de dues fases la podem veure en la figura. Fixem-nos que, potencialment, hi pot haver problemes de transitoris si els senyals a l'entrada de les portes AND arribessin en temps semblants. No obstant això, el retard d'un dels senyals sempre és més gran que l'altre, ja que ha de creuar el biestable, amb la qual cosa no tindrem mai pics de senyal a les sortides.

Obtenció d'un rellotge de dues fases

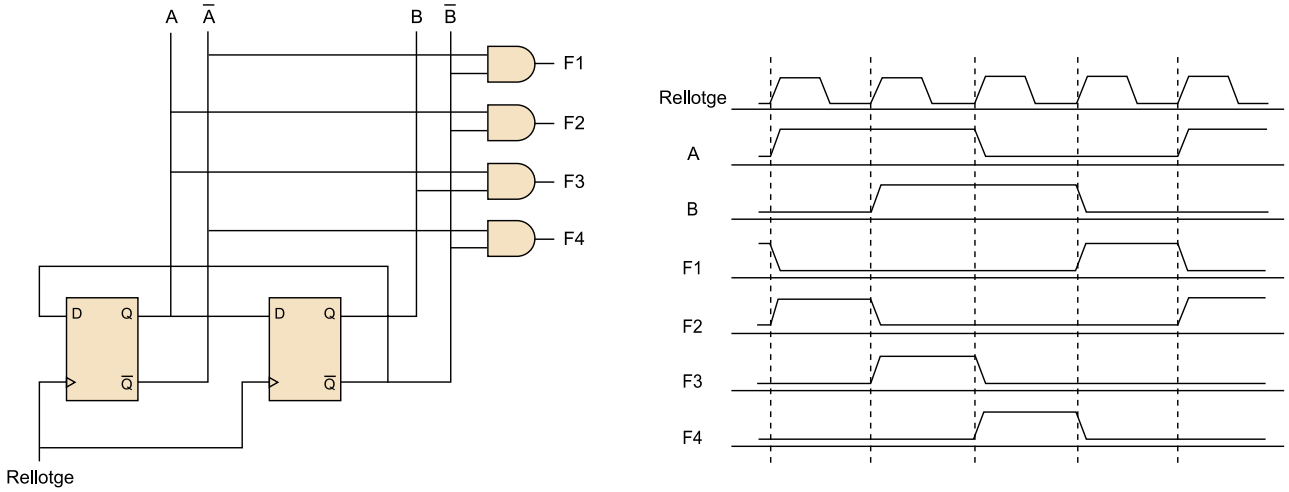


Si volem incrementar el nombre de punts de decisió en un mateix cicle de rellotge, una bona opció és optar pel comptador de Johnson i descodificar-ne les sortides per a obtenir cadascuna de les fases. Recordem que gràcies a la codificació de Gray només canvia una variable d'estat en cada cicle del rellotge base i, per tant, no tindrem mai transitoris en les sortides.

Exemple

En l'exemple de la figura següent podem veure el comptador de Johnson de 2 bits per a obtenir un rellotge de quatre fases, i també el diagrama temporal corresponent.

Obtenció d'un rellotge de dues fases mitjançant un comptador de Johnson de 2 bits



Finalment, i com hem comentat, en el cas d'usar el comptador de Johnson per a obtenir un rellotge, ens hem d'assegurar que aquest no entri en algun dels estats il·legals, la qual cosa seria fatal per al sistema.

3. Comunicació amb l'exterior

Depenent del tipus d'aplicació, el sistema encastat portarà a terme les seves tasques d'una manera aïllada o, d'altra banda, necessitarà comunicar-se amb l'exterior ja sigui per a intercanviar dades amb altres sistemes o per a rebre o donar ordres. En aquest darrer cas, cal proveir el sistema d'un mecanisme de comunicació adequat segons les restriccions de l'equipament i les necessitats de l'aplicació. En aquesta secció, es presenten diferents opcions disponibles per a portar a terme això i es discuteix el grau d'aplicabilitat de cadascuna d'elles. Concretament, la secció es troba dividida en dues parts segons el tipus d'estratègia utilitzada per a portar a terme la comunicació. En la primera part, es presenten els mecanismes basats en l'ús de cablejat, on la comunicació amb l'exterior es fa connectant el sistema encastat amb altres dispositius exteriors amb un cable. Aquesta connexió, concretament, es fa mitjançant un punt d'accés que té el mateix sistema encastat, el qual és conegut amb el nom de *port* i varia depenent de la tecnologia utilitzada. En la segona part, d'altra banda, es discuteixen estratègies que eviten l'ús de cablejat per tal de facilitar les tasques de comunicacions que, en aquest cas, són conegudes com a *estratègies de comunicació sense fil*.

3.1. Ports

En informàtica, els **ports** d'un ordinador serveixen per a crear una interfície de comunicació entre el mateix sistema i altres ordinadors, o bé entre el sistema i altres dispositius que actuïn com a perifèrics. La comunicació en si es fa mitjançant l'ús de cable (de diferent natura segons el tipus de port) i el port, concretament, és el punt d'accés de l'ordinador on es connecta aquest cable. En el cas dels sistemes encastats, els ports també s'utilitzen per a oferir una interfície de comunicació del sistema amb l'exterior, ja sigui amb altres sistemes encastats, amb perifèrics o amb altres tipus de maquinari.

A continuació, es descriuen els tipus de ports més utilitzats en els sistemes encastats, se'n destaquen les característiques més importants i es comenten els aspectes relacionats amb la seva aplicació a aquest tipus de sistemes.

3.1.1. Port sèrie

Juntament amb el port paral·lel, el port sèrie era un dels ports per excel·lència als primers anys que van aparèixer els PC. Concretament, aquest tipus de port s'utilitza per a establir una comunicació punt per punt entre dos dispositius. Quant al nom de *port sèrie*, és determinat pel fet que només es pot enviar un bit de dades a cada instant de temps. Malgrat que hi ha un quants estàndards per

a portar a terme comunicacions mitjançant el port sèrie, normalment, quan es parla d'aquest tipus de port (també conegut com a *port COM*), es considera l'estàndard RS-232.

En la figura següent, es presenta, juntament amb el cablejat relacionat, l'aspecte del port sèrie basat en RS-232 amb el tipus de connector més utilitzat, que és de 9 pius i es coneix amb el nom de *DB-9* (existeix una versió de 25 pius però és menys extensa i més cara). Aquest estàndard concret permet comunicacions a 20 kBps fins a una distància de 25 metres (de cable). No obstant això, cal comentar que la velocitat que es pot establir depèn de la longitud del cable i, en realitat, es poden aconseguir velocitats més altes (entorn dels 115200 Bps).

Port sèrie basat en RS-232 amb connector de 9 pius DB-9

En el cas dels sistemes encastats, aquest port s'utilitza normalment per a portar a terme tasques de programació o depuració del sistema encastat. És una opció atractiva quant a cost i simplicitat i, per aquest motiu, es troba molt estès a una àmplia gamma de dispositius. No obstant això, té com a limitació la baixa velocitat i la mida del port en si; per aquest motiu, es fan servir actualment també altres alternatives per a fer aquestes tasques com és el cas del port USB (com es presenta més endavant).

3.1.2. Port paral·lel

Com s'ha comentat en el cas anterior, el port paral·lel és un altre dels ports clàssics en la informàtica convencional. Normalment, es feia servir per a connectar el PC amb impressores. Tanmateix, aquest ús s'està deixant actualment al cas del port USB.

Aquest tipus de port també estableix una connexió punt per punt però, a diferència del port sèrie, en aquest cas es permet la transmissió de fins a 8 bits en paral·lel (per això es denomina així el port). També cal comentar que segueix l'estàndard de comunicació IEEE 1284, el qual permet velocitats de comunicació que oscil·len entre els 4 MBps i els 16 MBps a unes distàncies entre els 6 i 10 metres (segons la qualitat del cable). En la figura 65 es presenta el port paral·lel.

Port paral·lel

Quant als sistemes encastats, aquest tipus de port no se sol utilitzar gaire. És a dir, rarament aquest tipus de port forma part del maquinari d'un sistema encastat. De vegades, s'inclou perquè el sistema s'ha de connectar amb una impressora o un altre tipus de perifèric, però cada cop és més estrany trobar aquest tipus d'aplicació amb la gran implantació del port USB.

No obstant això, cal comentar que una aplicació interessant del port paral·lel a un sistema encastat és la utilització del mateix per la fase de desenvolupament, concretament per a portar a terme tasques de depuració del funcionament del sistema. Per fer això, s'han d'introduir ordres de crida al port paral·lel a diferents parts del codi i, com que el port paral·lel permet que l'últim valor del *pin* es mantingui fins que aquest no canviï, un pot observar el desenvolupament

Exemple

És molt fàcil inserir LED als pius del port paral·lel i fer-los servir per a indicar a quina línia del codi és.

del codi al sistema còmodament. Aquesta tasca és facilitada pel fet que molts sistemes operatius (com el cas de Linux) ofereix controladors (*drivers*) per a poder portar a terme aquestes tasques sense dificultat.

3.1.3. USB (*universal serial bus*)

Aquest port, desenvolupat pel consorci de companyies USB Implementers Forum (USB-IF), va néixer amb l'objectiu de substituir els ports paral·lel i sèrie. El tipus de port utilitzat és d'una mida més reduïda (tal com es pot observar en la figura següent) i les seves principals característiques són el baix cost de la solució, les velocitats de dades més altes que es poden aconseguir i la facilitat d'interconnexió entre dispositius, la qual es fa seguint la filosofia de la integració automàtica, amb una configuració simplificada i sense la necessitat de reiniciar el sistema (concepte conegut com a ***plug-and-play***).

Port USB

El tipus de connexió d'USB es basa en l'ús d'una topologia en estrella on un dispositiu fa d'ordinador central (*host*) i la resta s'hi connecten. A més, s'ofereix la possibilitat que un ordinador central es connecti a un altre ordinador central, la qual cosa dóna com a resultat la creació d'una topologia en arbre amb diferent branques amb un màxim de cinc nivells i de 127 dispositius connectats a l'ordinador central principal (o *root host*). Tot i que aquest nombre màxim de dispositius pot semblar limitant en xarxes de gran escala, això no sol representar un problema en les típiques configuracions trobades en els sistemes encastats.

Quant a la velocitat, hi ha diferents límits, que es poden aconseguir amb longituds de cables no superiors a 5 metres, i aquests depenen de la versió del port:

- USB 1.0: 1,5 MBps.
- USB 1.1: 12 MBps.
- USB 2.0: 480 MBps, però normalment es treballa a 125 MBps (aquesta és la versió d'USB més estesa actualment).
- USB 3.0: 4,8 GBps (amb una longitud de cable recomanable de 3 metres en aquest cas). Cal esmentar que aquesta tecnologia no s'ha acabat d'instaurar comercialment a causa, en part, de l'increment de cost que comporta.

Un altre dels avantatges d'aquest tipus de port és que els dispositius connectats poden ser alimentats pel mateix port, sempre que el nivell d'alimentació requerida pel dispositiu no sigui massa elevada.

Fruit dels avantatges comentats, el port USB és avui dia una de les solucions més comunes per a poder oferir interconnexió de nombrosos dispositius, i s'ha convertit, en molts casos, en port estàndard com passa en el cas de les impressores i càmeres digitals.

Exemple

Entre els dispositius que es poden connectar mitjançant l'USB, podem esmentar impressores, càmeres digitals, telèfons mòbils, càmeres de vídeo, dispositius d'emmagatzematge, gravadores, teclats, ratolins, etc.

Tot i que aquest tipus de port està principalment orientat a la informàtica de consum, la seva utilització en sistemes encastats ha proliferat en els darrers anys. Tal com s'ha comentat anteriorment, el port USB s'està utilitzant com a substitutiu del port sèrie per a portar a terme tasques de programació i depuració ja sigui perquè el connector és més simple, per la possibilitat de detecció i configuració automàtica, pel baix cost, per la possibilitat de fer servir el mateix port per a alimentar el sistema i perquè té més velocitat.

Una altra utilitat del port USB, atesa la velocitat més alta que pot oferir el sistema, és la connexió del sistema encastat amb un altre dispositiu per tal de transferir dades.

Exemple

Un exemple podria ser el cas que el sistema encastat estigués dedicat al monitoratge d'un esdeveniment físic amb una complexitat de maquinari limitada, per tal d'oferir una solució de baix cost i amb baix consum energètic. Aquest exemple es troba comunament a xarxes de sensors, les quals són xarxes designades pel control i monitoratge d'esdeveniments físics i són formades per un alt nombre de nodes de baixa complexitat amb la finalitat específica de prendre mesures. En aquest cas, el sistema encastat (node de la xarxa) es pot connectar mitjançant USB amb un altre dispositiu més complex encarregat de recol·lectar les dades mesurades per a analitzar-les i processar-les.

D'altra banda, ateses les possibilitats d'interconnexió d'aquest sistema, es pot establir una xarxa de recol·lecció de dades dels diferents dispositius encastats mitjançant l'ús d'USB. Seguint amb l'exemple anterior, es podria formar una xarxa tipus estrella en què els diferents sensors es connectarien amb el dispositiu recol·lector de dades.

Finalment, atesa la gran popularitat del port USB als dispositius de consum, aquest port es pot fer servir per a connectar dispositius addicionals al sistema encastat com poden ser càmeres digitals, de vídeo o altres tipus de perifèrics segons l'aplicació del dispositiu en si.

Exemple

Com a mostra d'això, podríem assenyalar les tasques de monitoratge o vigilància.

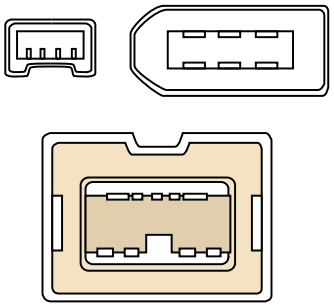
3.1.4. Firewire

La tecnologia d'aquest tipus de port va ser originalment creada per la companyia Apple en la dècada de 1980 (concretament, va ser iniciada el 1986) per a ser posteriorment traslladada a l'IEEE, cosa que va donar com a resultat l'estàndard IEEE1394 (finalitzat l'any 1995). Bàsicament, l'objectiu d'aquesta tecnologia era semblant a la d'USB: oferir altes velocitats a baix cost. Per les seves prestacions, Firewire és normalment utilitzat en dispositius relacionats amb aplicacions d'àudio i vídeo. Hi ha diverses versions de l'estàndard, i ca-

dascuna d'elles ofereix diferents velocitats límit com el Firewire 400 (versió original amb 400 MBps), el Firewire 800 (versió amb 800 MBps) i les versions Firewire s1600 i s3200 (amb 1,6 GBps i 3,2 GBps, respectivament).

Les connexions entre dispositius es fa seguint una topologia en arbre però, en aquest cas, no és necessari que un dels dispositius actuï com a màster de la xarxa. És a dir, es poden fer transmissions directes entre dispositius sense la necessitat que cap d'ells o un altre de la xarxa no actuï com a màster i controli el protocol de comunicacions. Concretament, en aquest cas, s'utilitza un sistema de comunicació purament d'igual a igual (*peer-to-peer*). Cal esmentar que el màxim nombre de dispositius que es poden connectar és de 63 amb unes longituds de cable per connexió d'uns 4,5 metres.

Actualment, les versions més utilitzades són Firewire 400 i 800. En la figura següent, es mostren els connectors utilitzats en aquestes configuracions. En el cas del Firewire 400, es poden utilitzar connectors amb 6 pius o amb 4 pius. En el cas de 6 pius, 4 pius són utilitzats per a enviar dades i els 2 restants per a proveir d'alimentació el dispositiu connectat (fins a 45 W), de manera que s'evita la seva alimentació externa en els casos en què no es necessitin nivells d'alimentació més alts. En el cas dels 4 pius, els pius d'alimentació són eliminats. En el cas de Firewire 800, s'utilitzen connectors de 9 pius (el qual s'adopta també en versions superiors de l'estàndard), en què 6 pius són per a dades i 3 per a alimentació. Malgrat que les diferents versions de Firewire són compatibles entre elles, per a poder connectar dispositius Firewire 400 amb dispositius d'altres versions cal fer servir adaptadors, els quals es poden trobar fàcilment i són proveïts per diverses companyies. A diferència d'USB, la velocitat d'una xarxa Firewire no disminueix quan un dispositiu de velocitat inferior és introduït en la xarxa. En tot cas, disminueixen només les velocitats de transferència cap al dispositiu, o des d'aquest dispositiu, de velocitat inferior.



Connectors utilitzats a Firewire

Esquerra: connector de 4 pius per a Firewire 400. Centre: connector de 6 pius per a Firewire 400. Dreta: connector de 9 pius per a Firewire 800.

Tot i que les darreres versions d'USB ofereixen velocitats nominals semblants a Firewire (per exemple, 480 MBps de l'USB 2.0 enfront de 400 MBps del Firewire 400), el darrer sistema és, a la pràctica, més eficient quant a reducció del temps de transferència per a enviar un mateix volum de dades. Això es deu al fet que la connexió de Firewire és tipus d'igual a igual i, per tant, redueix la complexitat del sistema. El sistema USB, en basar-se en una configuració màster-esclau, pateix uns retards extra relacionats amb tasques portades a terme pel protocol de comunicació. A més de requerir un consum més gran de CPU. D'altra banda, el fet que una xarxa Firewire no requereixi un node que actuï com a màster i que es puguin fer connexions directes, entre dos o diversos equips, ofereix un potencial de gran valor per tal d'establir xarxes d'alta velocitat i de baix cost.

En el cas dels sistemes encastats, no obstant això, l'ús d'aquest tipus de port no és gaire freqüent. Quan és utilitzat, se sol fer per a portar a terme tasques de depuració del sistema aprofitant l'alt volum de dades que es poden transferir. Tanmateix, cal dir que la seva utilització podrà ser més estesa en el futur si augmenta la demanda d'aplicacions que requereixen altes velocitats.

3.1.5. Ethernet

La creació d'Ethernet data de la dècada de 1970 i la seva estandardització és determinada per la norma IEEE 802.3. Ethernet ofereix diferent variants que treballen a diferents velocitats; les versions més populars són les de 10 MBps, 100 MBps i 1 GBps. Aquestes variants utilitzen també diferents versions de cablejat físic, i ofereixen per tant diferents comportaments que es veuen reflectits a la longitud màxima que poden tenir els cables.

Exemple

En el cas de 10 MBps, algunes variants són les referides com a 10Base2 (cable coaxial amb longitud màxima de 185 m) i 10BaseT (parell trenat amb 100 m); aquesta última és la variant més utilitzada.

L'accés al medi és compartit usant-se el protocol d'accés múltiple amb escolta de portadora i detecció de col·lisió (CSMA-CD, *carrier sense multiple access with collision detection*). Els dispositius que utilitzen aquest protocol escolten el medi abans de transmetre els seus paquets per a veure si hi ha alguna altra transmissió activa i, per tant, per tal d'evitar col·lisions. A la vegada que va transmetent les dades, el dispositiu escolta també el medi per tal de veure si hi ha col·lisió amb un altre dispositiu, és a dir, veure si un altre dispositiu també ha començat a transmetre. Si es detecta una col·lisió, s'atura la transmissió i no es torna a provar a transmetre fins que passa un temps aleatori després.

Inicialment, Ethernet es basava a *penjar* tots els dispositius del mateix cable (cable coaxial) i, per aquest motiu, es feia servir aquest tipus de protocol d'accés compartit. Aquest sistema oferia molt bons resultats però era bastant ineficient quan s'aplicava en xarxes relativament grans. Per tal de reduir el nombre de col·lisions i fer la xarxa més robusta, es va canviar a una topologia en estrella en què tots els dispositius es connectaven a un node central. Quant a aquest node central, hi ha dispositius de diferent natura per tal de portar a terme aquesta tasca i ofereixen la possibilitat de dividir la xarxa en diferents segments (o subxarxes), i per tant es divideixen el tràfic global i faciliten l'escalabilitat. Segons les funcionalitats que incorporen, els dispositius més comuns són:

- **Concentrador (*hub*)**: actua com a unió física dels diferents dispositius. És a dir, és un simple repetidor, si rep un paquet d'un dispositiu el retransmet a tots els dispositius que té connectats.

- **Commutador (*switch*)**: és un dispositiu més sofisticat que el concentrador, ja que analitza el paquet rebut per a veure l'adreça de destinació. D'aquesta manera, reenvia només el paquet al dispositiu de destinació o al segment on es troba el dispositiu de destinació o el segment on es troba.

- **Encaminador (*router*)**: és un tipus de dispositiu que se sol utilitzar en xarxes grans on diferents subxarxes (o segments) estan interconnectades.

Bàsicament, té la mateixa funcionalitat que el commutador però també té l'habilitat d'encaminar el paquet per la millor sortida en termes de camí més curt i menys congestionat.

Ethernet és una de les opcions més senzilles i amb un cost més baix de les existents per a muntar una xarxa de dispositius encastats amb una velocitat relativament alta. Sens dubte, el port Ethernet és el més utilitzat per a connectar dispositius en xarxes de computadors i per aquesta raó aquest tipus de port també es troba implementat en moltes solucions encastades. Bàsicament, ofereix la possibilitat de connectar el dispositiu de manera senzilla amb altres computadors, impressores, bases de dades i a Internet.

Exemple

Una opció interessant és fer servir Ethernet per a connectar el sistema encastat a Internet per tal d'oferir una solució remota de monitoratge i control.

Cal comentar que, fa temps, incloure Ethernet a un sistema encastat era força complicat perquè un dispositiu basat en aquest estàndard tenia una alta complexitat, tant quant al circuit com als components requerits. No obstant això, aquesta tasca s'ha facilitat bastant, atès l'alt nivell d'integració assolit darrerament. Concretament, hi ha implementacions de xips de controladors Ethernet específicament dissenyades per a la seva implementació en sistemes encastats, les quals tenen com a característica principal oferir interfícies Ethernet de baix cost i simples. Les dues més populars són les donades pels xips Realtek 8019 i el Cirrus CS8900A. Cal dir, tanmateix, que la majoria d'aquestes implementacions treballen normalment a 10 MBps.

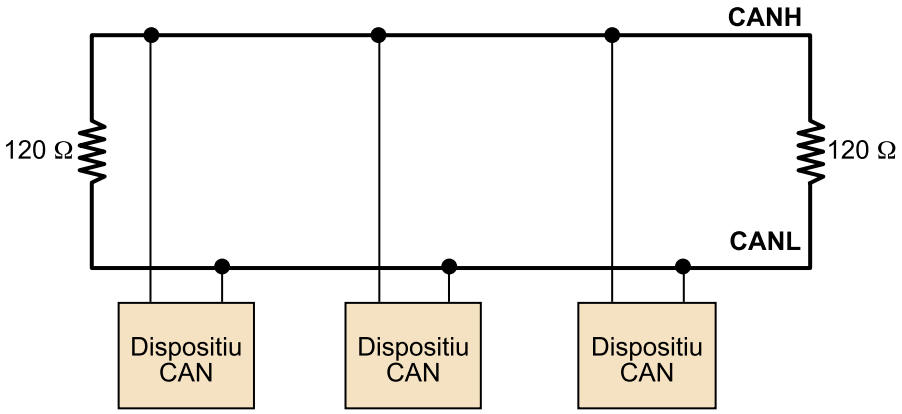
3.1.6. CAN

Aquesta secció la concloem presentant una tecnologia utilitzada en entorns industrials. Aquests tipus d'entorns estan caracteritzats per la seva hostilitat en termes de soroll electromagnètic sever. Per això, són dissenyats tenint en compte altres criteris. En aquesta secció concreta, ens centrem en la tecnologia *controller area network* (CAN), la qual se sol utilitzar en entorns severament afectats per interferència electromagnètica.

En les dècades de 1970 i 1980, l'automoció va sofrir un canvi important en el sentit que els automòbils disposaven cada cop de més dispositius electrònics com són el cas de l'ABS, la transmissió electrònica, el sistema de climatització, el tancament centralitzat, etc. Naturalment, totes aquestes parts dels vehicles s'havien d'interconnectar per tal de poder portar a terme una interacció entre ells ja sigui per a dur a termes ordres de control o per a intercanviar dades. Originalment, tots aquests dispositius es connectaven fent servir cablejat punt per punt, però això limitava tant les prestacions del sistema com la seva escalabilitat i, a més, augmentaven el cost del vehicle i el seu pes. Per a solucionar això, l'empresa Bosch va desenvolupar CAN al finals de la dècada de 1980, el qual estava orientat principalment per a la indústria de l'automoció, però va ser adoptat posteriorment per altres aplicacions industrials. En particular, CAN està confirmada per la International Standards Organization (ISO) en forma de l'estàndard ISO 11898 i ISO 11518-2.

La solució CAN es basa a reemplaçar la complexitat del múltiple cablejat per l'ús d'un únic bus en què tots els dispositius s'hi connecten i en què es poden aconseguir velocitats de fins a 1 MBps amb cables de llargària inferiors a 40 m (el mínim de velocitat és de 40 kBps el qual es pot assolir amb cables d'1 km). Concretament, aquest bus consisteix simplement en una línia balancejada formada per un parell cables (vegeu la figura següent). En la figura, es pot observar també que les línies estan terminades amb resistències de 120 Ω i el motiu d'això és evitar reflexions de les transmissions efectuades sobre la línia.

Xarxa CAN



En una xarxa CAN, els dispositius es poden afegir o desconnectar en qualsevol moment i es permet que diferents dispositius actuïn com a màster, on cadascun d'aquests màsters es pot encarregar del control local de la xarxa. No obstant això, el tipus d'accés al bus es duu a terme fent servir un mecanisme de contenció basat en CSMA/CD + AMP (*carrier sense multiple access with collision detection and arbitration message priority*). Bàsicament, aquest esquema és similar al cas de CSMA/CD utilitzat a Ethernet en el sentit que un node mira la línia abans de transmetre per a veure si està lliure. Si és el cas, va transmetent els diferents bits d'informació i, alhora, va comparant si els bits existents a la línia són els mateixos que els que va transmetent. En cas que no sigui així és perquè hi ha col·lisió. Llavors, es para la transmissió i s'inicia un període d'arbitratge, el qual no es dóna al protocol CSMA/CD, i per això es diu CSMA/CD+AMP. Aquest arbitratge es basa a donar pas al dispositiu amb prioritat més alta, en què la prioritat dels diferents dispositius ha estat assignada durant el disseny del sistema. Quant al nombre de dispositius que es poden connectar a una línia, el màxim és de 30.

Tenint en compte que en un automòbil es troba, des del punt de vista electromagnètic, en un entorn molt sorollós atesa la presència de motors elèctrics, sistemes d'ignició, emissions de ràdio, etc., la comunicació establerta ha de ser altament robusta. Per a poder oferir això, es fa servir un sistema de comunicació balancejada. Quan es fa servir aquest tipus de comunicació la informació s'envia variant els nivells de voltatge de dues línies.

En aquest cas, les línies utilitzades són les línies CANH i CANL (vegeu-les en la figura anterior). Al receptor, es mesura la tensió diferencial d'aquestes dues línies, és a dir, es mesura la tensió CANH-CANL per tal de determinar el bit enviat. Això difereix d'esquemes convencionals en què el receptor extreu la informació mirant el voltatge d'una sola línia. En el cas de CAN concret, un bit 0 s'envia induint les tensions 3,5 V, a la línia CANH, i 1,5 V, a la línia CANL. Per tant, al receptor es tindrà una tensió diferencial CANH-CANL igual a 2 V. En el cas de voler enviar-se un bit 1, les dues línies es posen a 2,5 V tenint com a resultat una tensió diferencial de 0 V. L'objectiu per a utilitzar aquest mecanisme diferencial és reduir el soroll induït per altres fonts electromagnètiques. És a dir, si hi ha un soroll d'aquest tipus, aquest serà induït de la mateixa manera a les dues línies CANH i CANL. Per tant, en obtenir la tensió diferencial CANH-CANL, aquest soroll serà compensat.

Exemple

Si es vol enviar un 0 lògic, CANH serà igual a 3,5 V i CANL a 1,5 V. Si tenim un soroll induït a un instant de temps igual a +1 V, el que es rep de les línies CANH i CANL serà 4,5 V i 2,5 V, respectivament. En obtenir la tensió diferencial, no obstant això, s'obtindran els 2 V nominals corresponents a un bit 0.

El tipus de connector utilitzat a CAN és el mateix que l'utilitzat al port sèrie, és a dir, s'utilitza el connector DB9. No obstant això, és important remarcar que no hi ha compatibilitat entre aquests dos sistemes. En la taula següent, s'indica el senyal que porta cada piu en el cas de CAN.

Número de piu	Utilització
1	No utilitzat
2	CANL
3	GND (massa)
4	No utilitzat
5	No utilitzat
6	GND (massa)
7	CANH
8	No utilitzat
9	Alimentació opcional

Cal comentar que aquest tipus de port es troba molt sovint en dispositius utilitzats en entorns industrials. Per tant, és una opció molt adient si l'aplicació del sistema encastat és d'aquest tipus. Sobretot en cas que treballi en un entorn molt hostil en termes de soroll i interferència, i la limitació de velocitat no sigui un impediment per a l'aplicació concreta.

3.2. Comunicació sense fil

Les comunicacions sense fil, tal com bé indica el seu nom, són aquells tipus de comunicacions que possibiliten la interconnexió de dos o més equips sense la necessitat de fer servir cablejat. La seva utilització és cada cop més comuna en tots els tipus d'aparells, a causa de la comoditat que ofereixen. En el cas d'un sistema encastat, aquest tipus de comunicacions en simplifica enorme-

ment la interoperabilitat i les tasques de comunicacions. Cal dir també que els sistemes encastats poden treballar en escenaris amb molt poca accessibilitat o en entorns en què l'ús de cablejat representi un gran impediment (per exemple, en aplicacions de monitoratge del medi ambient o de la fauna). Per tant, les comunicacions sense fil ofereixen la possibilitat que el sistema encastat es pugui comunicar amb l'exterior d'una manera eficient i, fins i tot, poden ser l'única solució viable en algunes circumstàncies.

3.2.1. IRDA

Aquest tipus de tecnologia està basada en l'ús de comunicació via infrarojos. Tal com es veurà a continuació, les seves característiques difereixen bastant de la resta de tecnologies presentades en aquesta secció, les quals treballen amb comunicacions via ràdio.

L'estandardització Infrared data association (IRDA) va lligada a un consorci d'empreses que es va establir el 1993 per tal de promoure una tecnologia de comunicació de baix cost i basada en l'ús d'infrarojos. L'origen d'aquesta tecnologia prové dels enllaços de comunicacions per infraroig que Hewlett-Packard va incorporar a les seves calculadores.

Per a la transmissió de les dades, hi ha tres categories:

- Serial infrared (SIR) amb velocitats fins a 115,2 kBps.
- Medium speed infrared (MIR) amb velocitats màximes de 1.152.
- MBps i fast infrared (FIR) que pot arribar als 4,0 MBps.

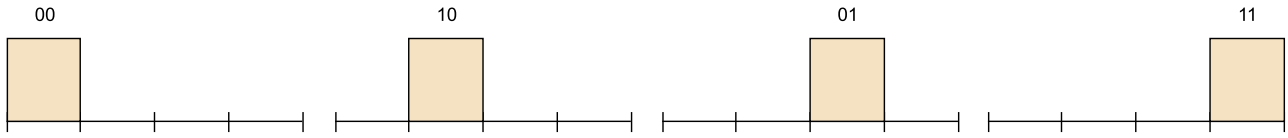
Per a dur a terme la comunicació de dades en els casos SIR i MIR, aquesta es fa en sèrie i mitjançant la modulació dels bits amb polsos de llum tal com es descriu a continuació:

- Si es vol enviar un 0, el temps de transmissió es divideix en dues parts: una inicial amb un temps igual a 3/16 el temps de bit total i una segona part amb la resta del temps de bit. Durant la primera part s'emet un pols de llum d'intensitat contínua, mentre que a la segona no s'emet res.

- Si es vol enviar un 1, no s'emet llum durant el temps de transmissió.

No obstant això, en cas que es transmeti a 4 MBps, la modulació té una petita variació. En aquest cas, el temps de transmissió es divideix en quatre parts i s'utilitza per a enviar dos bits simultàniament. Segons els bits que es vulguin transmetre, només una de les quatre fraccions s'utilitza per a emetre llum tal com es descriu en la figura següent:

Polsos utilitzats en IRDA treballant a 4 MBps

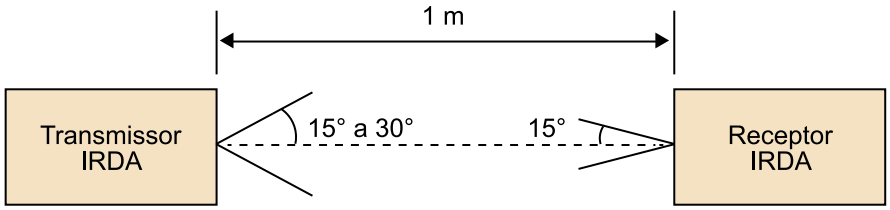


Cada temps de transmissió s'utilitza per a enviar 2 bits i hi ha quatre combinacions diferents de polsos.

Quant a la cobertura, es troba força limitada (tal com es presenta en la figura següent).

Per tant, la utilitat d'aquest sistema està principalment orientada a evitar l'ús d'un cable per a connectar dos dispositius propers. Bàsicament, la seva motivació és evitar els típics problemes trobats quan es vol fer una connexió en un moment determinat i no es pot fer perquè o bé no es disposa d'un cable o bé un dels equips no està ben configurat. En algunes situacions, aquesta cobertura limitada, quant a angle i necessitat de visió directa, es pot veure com una garantia de seguretat a l'enllaç de la comunicació, ja que es dificulta que un intrús intercepti dades.

Esquema de transmissió d'IRDA i angles de visió



Aquest tipus de tecnologia es troba present principalment a ordinadors portàtils, telèfons mòbils, calculadores i PDA, i va gaudir de gran popularitat en la dècada de 1990 i començament de la dècada del 2000. Posteriorment, la seva utilització va disminuir envers l'ús d'altres tecnologies ràdio com Bluetooth i Wireless que gaudien de més cobertura i evitaven la necessitat de visió directa. No obstant això, segons l'aplicació, IRDA pot ser un millor candidat a causa del seu nivell de seguretat i en entorns en què les interferències representin un problema. A més, és una tecnologia que va ser dissenyada per tal de poder ser implementada a un cost molt baix i això és, en part, gràcies al fet que el transmissor i receptor es basen en l'ús d'un LED i un detector fotodíode, respectivament.

3.2.2. Bluetooth

Aquest sistema de comunicacions ràdio va ser ideat per la companyia Ericsson l'any 1995 amb el principal objectiu de proveir un estàndard ràdio capaç de connectar multitud de dispositius de manera senzilla i evitar tasques de configuració. És a dir, la idea bàsicament és connectar dos dispositius equipats amb Bluetooth, que es detectin automàticament i que es pugui establir connexió entre ells d'una manera immediata. L'objectiu, per tant, és similar al del cas d'IRDA, però en aquest cas s'eviten els problemes observats quant a necessitat de visió directa i cobertura escassa en treballar amb un sistema de comunicacions ràdio.

D'altra banda, també va ser dissenyat amb l'objectiu de poder crear petites xarxes (conegudes com a **picoxarxes**) entre els diferents dispositius i facilitar, per tant, l'intercanvi de dades. Actualment, aquest estàndard rep el suport del Bluetooth Special Interest Group, consorci format per més de 1.900 empreses. En els darrers anys, el sistema Bluetooth ha estat principalment destinat als perifèrics, PDA i telefonia mòbil, ja que es veu com un sistema alternatiu a l'ús del cable en entorns relativament propers i d'àrea personal.

D'on ve el nom *Bluetooth*?

L'origen del nom *Bluetooth* prové del rei escandinau Harold Blåtand (segle X), qui es va fer famós per unificar multitud de tribus, bastant dissonants entre elles, de les actuals Noruega i Dinamarca. La traducció de Blåtand a l'anglès és Bluetooth. Ericsson va agafar aquest nom per reflectir el caràcter unificador de Bluetooth en el sentit d'usar-se per a connectar multitud de dispositius diferents via ràdio.

Bluetooth és un estàndard de comunicacions ràdio que permet la transmissió de veu i dades que opera a la banda de freqüències lliure de 2,4 GHz (una de les bandes Industrial, Scientific and Medical o ISM). Aquest estàndard entraria a la categoria de xarxes sense fil de caràcter personal (o *wireless personal area networks*, WPAN), ja que està dissenyat per a proveir de dispositius de baixa potència, mida i cost. Concretament, els dispositius Bluetooth es classifiquen en diferents classes segons la limitació en potència:

- Classe 1: potència màxima de 100 mW i cobertura d'uns 100 m.
- Classe 2: potència màxima de 2,5 mW i cobertura d'uns 10 m.
- Classe 3: potència màxima d'1 mW i cobertura prop d'1 m.

Cal esmentar que aquestes cobertures estan referides a entorns interiors, en què solen operar els dispositius Bluetooth, ja que estan enfocats a xarxes tipus WPAN. A part dels diferents nivells de potència segons la classe del dispositiu, l'estàndard també defineix un sistema de control de potència, en què el transmissor adapta la potència per tal de transmetre només amb la potència necessària que asseguri que el senyal arriba al receptor amb un nivell de potència acceptable. Això es fa determinant una sèrie de llindars de potència al receptor i el resultat obtingut és que s'optimitza la duració de la bateria.

Bandes ISM

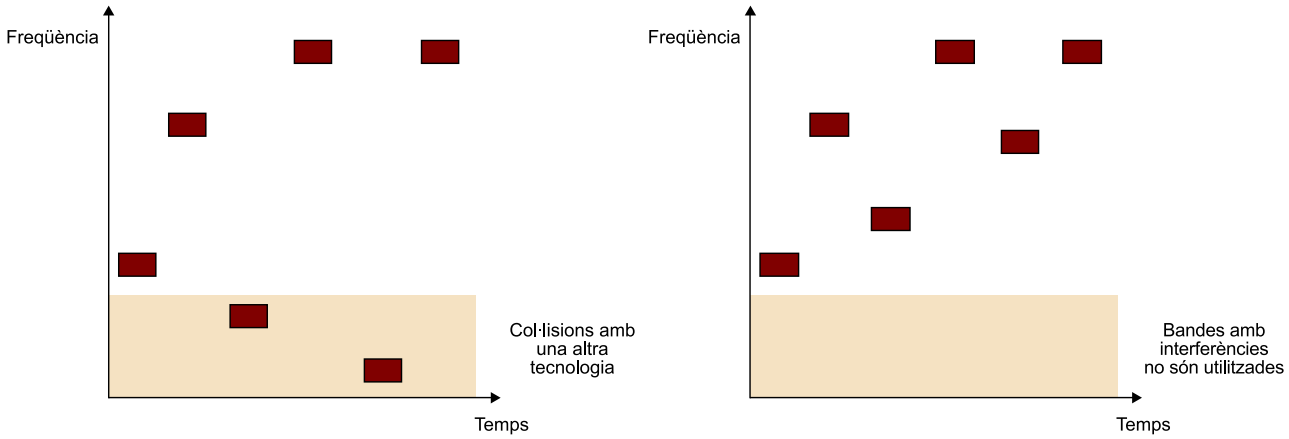
Les bandes ISM són bandes reservades internacionalment per a l'ús de sistemes industrials, científics i mèdics. La més popular és la banda lliure a 2,4 GHz (compresa entre els 2.400 i 2.500 MHz), ja que s'utilitza per a multitud de sistemes ràdio de caràcter personal (WPAN, *wireless personal area networks*) o local (WLAN, *wireless local area networks*) com alguns dels comentats en aquesta secció. Això és propiciat pel fet que, a diferència d'altres bandes, aquesta és una banda lliure en el sentit que no es necessita cap llicència per a

poder transmetre. Quan són necessàries, aquestes llicències són concedides per l'entitat reguladora del país on s'està treballant.

La velocitat de transmissió depèn de la versió de l'estàndard, que cobreix des d'1 MBps ofert per a la versió 1.2 de l'estàndard fins als 3 MBps de la versió 2.0 + Enhanced Data Rate (EDR). El signe + d'aquest darrera versió serveix per a reflectir que EDR és un mode opcional de l'estàndard 2.0 que s'activa per a arribar als 3 MBps. En cas que no s'activi, el dispositiu continua sent compatible amb la versió anterior 1.2. Cal esmentar que hi ha una versió 3.0 + High Speed (HS) que pot arribar fins als 24 MBps. No obstant això, aquesta versió fa servir tecnologia Wi-Fi per a enviar les dades a aquesta velocitat, mentre que Bluetooth es reserva per a establir la connexió.

Tal com s'ha esmentat, Blueetoth treballa a una banda lliure on multitud de dispositius operen. Per tant, hi ha un alt nivell d'interferències procedents d'uns altres equips. Per tal de proveir d'un sistema robust aquestes interferències, Bluetooth fa servir un mecanisme de transmissió anomenat *frequency-hopping spread spectrum*. La idea d'aquest mecanisme és simple: es divideix la banda utilitzada per Bluetooth (entre 2.402 i 2.480 MHz) en 79 canals d'1 MHz i es va canviant de canal al llarg del temps. Concretament, es canvia de canal 1.600 cops per segon i això redueix l'impacte d'interferències d'altres sistemes, els quals rarament ocuparan la banda ISM sencera. Per tal de portar a terme aquesta tasca amb més èxit, es va incloure en la versió 1.2 de l'estàndard una variant més sofisticada d'aquesta tècnica anomenada AFS (*adaptive frequency-hopping spread spectrum*). Aquesta variant detecta quins són els canals utilitzats per altres sistemes amb la finalitat d'evitar-los a la seqüència de salts de canals. Aquest concepte es pot observar en la figura següent:

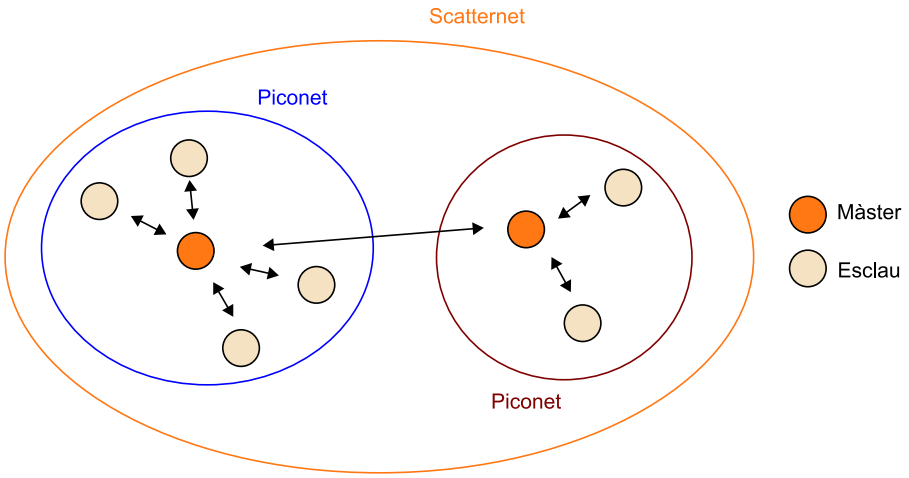
Esquema de funcionament de *frequency-hopping spread spectrum* i la seva variant adaptativa AFS



El tipus de xarxa utilitzat en Bluetooth, referida com a picoxarxa, es basa en el concepte *màster-esclau*, en què un node actua com a màster, de manera que s'encarrega de la sincronització de la xarxa, i els altres prenen el rol d'esclaus. A una picoxarxa, hi pot haver fins a 7 esclaus i diverses picoxarxes es poden enllaçar entre elles i formar el que es coneix com a *xarxa dispersa* (*scatternet*) (vegeu la figura 72). Dintre d'una picoxarxa, la transmissió es divideix en *slots*

de 625 µs, en què l'inici de cada *slot* és marcat pel rellotge intern del màster. El màster utilitza els *slots* parells per a transmetre paquets i els imparells per a rebre'ls, mentre que els esclaus transmeten i reben els *slots* imparells i parells, respectivament. També es permet la transmissió de paquets de 3 i 5 *slots* de duració. En aquest cas, per a respectar la regla comentada en el cas de paquets d'un *slot* de duració, el màster ha d'iniciar les transmissions als *slots* parells, malgrat la transmissió duri més d'un *slot,* i els esclaus als imparells.

Exemple de *scatternet* Bluetooth formada per dues picoxarxes.



En el cas del sistemes encastats, Bluetooth pot ser una bona opció per a portar a terme comunicacions sense fil, a causa del baix cost, mida i baix consum dels xips. No obstant això, depenent del tipus d'aplicació, l'elecció d'aquest estàndard podrà finalment ser canviada pels sistemes ràdio que es presentaran a continuació: Wi-Fi i 802.15.4/Zigbee. D'una banda, veurem que Wi-Fi ofereix més cobertures i velocitats de transmissió. A més, el nombre d'estacions base, per a fer de passarel·la (o *gateway*) a Internet per exemple, està més estès en aquesta tecnologia que en el cas de Bluetooth. Quant al cost d'integració i consum de bateria, la tecnologia 802.15.4/Zigbee ofereix una alternativa més atractiva, ja que va ser dissenyada amb l'objectiu principal d'oferir dispositius ràdio amb un cost, mida i consum extremament baixos.

3.2.3. Wi-Fi

Aquest sistema de comunicacions ràdio està basat en la família d'estàndards IEEE 802.11 i es troba sota la tutela de la WIFI Alliance, la qual és un consorci d'empreses encarregades de la seva certificació per tal d'assegurar la interoperabilitat dels diferents dispositius Wi-Fi. Els principals avantatges de Wi-Fi són:

1) la maduresa de la tecnologia en el sentit que assegura la interoperabilitat entre els diferents fabricants i un bon comportament en dispositius petits;

2) la instal·lació i el manteniment senzills dels equips;

3) el fet que operi a les bandes lliures de 2,4 GHz (com altres sistemes ràdio de característiques similars) i 5 GHz (on no hi ha un solapament tan alt de sistemes i les interferències són més baixes), i

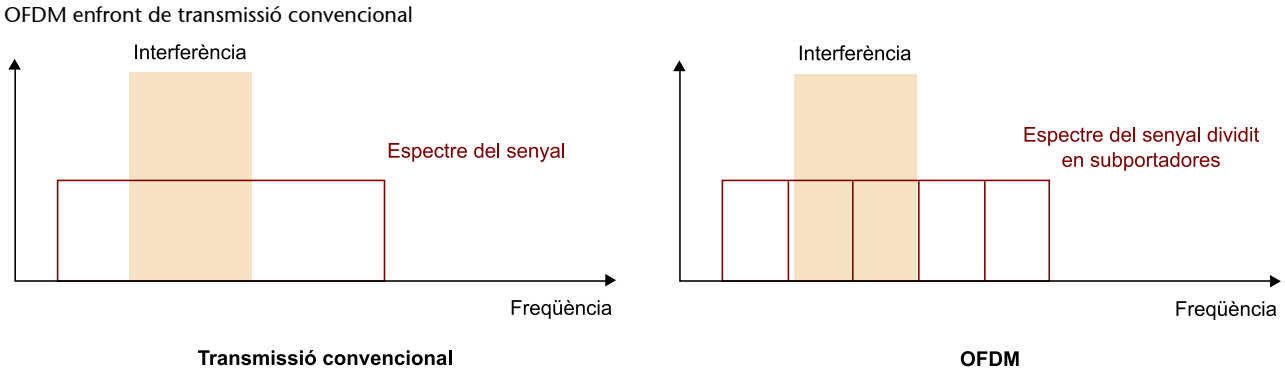
4) la seva gran incursió al mercat. Concretament, Wi-Fi es pot veure com un cas similar a l'observat amb Ethernet en el cas de xarxes cablejades.

A Wi-Fi també hi ha diferents versions amb diferents velocitats de transmissió. No obstant això, aquestes versions a més tenen grans diferències en termes d'implementació de la capa física. Les més destacables són les basades en els estàndards següents:

- **IEEE 802.11b**: opera a la banda de 2,4 GHz, la qual es divideix en 14 canals de 22 MHz compresos entre els 2.401 i 2.495 MHz (alguns canals s'encavalquen). Es poden aconseguir velocitats de transmissió a l'aire d'11 MBps (per a distàncies de 30 m en interiors) i 1 MBps (si la distància creix a 90 m). Aquesta variació de velocitat és causada pel fet que el sistema disposa d'un sistema d'adaptació que consisteix a ajustar la velocitat segons el nivell de senyal rebut al receptor. És a dir, s'augmenta o es redueix la velocitat si la qualitat del senyal és bona o dolenta. D'altra banda, tal com succeeix a Bluetooth, el sistema inclou una tècnica de transmissió orientada a reduir l'impacte de les interferències existents a la banda 2,4 GHZ, la qual es troba saturada de tecnologies diferents. En aquest cas, s'utilitza el que es coneix com a *direct sequence spread spectrum* (DSSS). Aquesta tècnica consisteix a distribuir l'energia del senyal en un ample de banda més gran que el necessari. Per tant, si es rep una interferència, l'impacte de la mateixa serà inferior, ja que no afecta el senyal total, sinó únicament una porció. Finalment, cal dir que aquesta modalitat de Wi-Fi és la més utilitzada i es troba molt present a les llars, on l'estació base Wi-Fi se sol emprar també com a porta d'enllaç a Internet.

- **IEEE 802.11g**: aquest sistema també opera a la banda de 2,4 GHz, la qual es troba dividida en aquest cas en tretze canals de 22 MHz. Concretament fa servir els tretze primers canals del 802.11b (entre 2.401 MHz i 2.483 MHz). En aquest cas, però, les velocitats de transmissió poden augmentar fins a 54 MBps, atès que es fa servir una tècnica de transmissió més sofisticada coneguda amb el nom de multiplexatge per divisió ortogonal de freqüència (OFDM, *orthogonal frequency division multiplexing*). Aquesta tècnica consisteix a dividir la transmissió en diverses transmissions paral·leles, on cadascuna d'elles utilitza un ample de banda més reduït. El concepte es presenta en la figura següent. Tal com es pot veure, tant el sistema de transmissió convencional com l'OFDM utilitzen el mateix ample de banda total; per tant, transmeten amb la mateixa velocitat. Tanmateix, si es té una atenuació gran o una interferència en alguna banda freqüencial (com es mostra en la figura següent), el senyal convencional es troba totalment distorsionat. En el cas d'OFDM, només unes poques subbandes són distorsionades, de manera que se'n redueix l'impacte ja que les dades envia-

des a la resta de subbandes no es perden. Per aquesta raó, la cobertura del 802.11g és més gran que en el cas de 802.11b i es poden arribar als 200 m. En aquest cas concret, l'OFDM també s'utilitza com alternativa a DSSS per a dur a terme la tasca de reduir l'impacte ocasionat per la presència d'altres equips a la banda lliure (només als modes de transmissions més lents, d'1 i 2 MBps, es manté el DSSS per a transmetre el senyal). Aquesta variant de l'estàndard va néixer per a cobrir l'alta demanda existent dels usuaris quant a necessitat de sistemes sense fil i oferir velocitats altes. Per tant, és un sistema que es troba força implantat, com el cas del 802.11b. Concretament, hi ha multitud de dispositius equipats amb ambdues variants (dispositius 802.11b/g).



- **IEEE 802.11a**: aquesta versió de l'estàndard opera a la banda dels 5 GHz; per tant, un dels primers avantatges que presenta és que es troba a una banda que no està tan saturada d'interferències com la banda de 2,4 GHz. Les bandes de treball concretes depenen de la regulació de cada país. En el cas d'Europa, s'utilitzen divuit canals de 20 MHz distribuïts al rang de freqüències 5.180 MHz–5.680 MHz. També utilitza l'OFDM, a causa de les millors prestacions en termes de velocitat ofert amb una velocitat màxima de 54 MBps. Tot i que la banda de 5 GHz ofereix un escenari menys saturat d'interferències, la seva utilització presenta el problema que l'atenuació en funció de la distància és més gran. Això es pot veure repassant la fórmula de Friis, la qual ens dóna el nivell de potència rebuda, P_r (expressada en watts), en un sistema de comunicació sense fil, on se suposa que la transmissió és a l'espai lliure (és a dir, no hi ha obstacles entre el transmissor i el receptor i es consideren negligibles els possibles rebots del senyal transmès):

$$P_r = P_t\left(\frac{c}{4\pi f d}\right)^2$$

en què P_t és la potència transmesa (expressada en watts), c és la velocitat de la llum ($3 \cdot 10^8$ m/s), f és la freqüència portadora (expressada en Hz) utilitzada per a transmetre el senyal i d és la distància en metres entre el transmissor i el receptor. Com es pot veure, la potència rebuda disminueix de manera quadràtica en funció de la distància. Per tant, la cobertura es troba força més limitada. Si a més es tinguessin en compte els possibles obstacles entre el transmissor i receptor, la qualitat del senyal es veuria

força més degradada. Una manera senzilla utilitzada per a modelar les pèrdues per propagació en entorns diferents a l'espai lliure és adaptar la fórmula de Friis de la manera següent:

$$P_r = P_t\left(\frac{c}{4\pi f}\right)^2 \frac{1}{d^\gamma}$$

en què γ és el paràmetre que modela les pèrdues de propagació de l'entorn, que varia normalment entre 2 (espai lliure) i 4-5 (en entorns interiors amb molts obstacles). Certament, modelar la potència rebuda en un entorn real comporta l'ús de models més complexos on s'han de tenir també en compte els paràmetres i no-idealitats del maquinari relacionat amb el dispositiu ràdio (eficiència de les antenes, diagrama de radiació, factor de soroll dels amplificadors, etc.), però aquesta aproximació pot servir per a fer un primer apropament al problema. A més, cal dir que la capacitat de penetrar obstacles també disminueix a mesura que augmenta la freqüència. Per tant, aquesta variant de l'estàndard se sol utilitzar per a desplegar radioenllaços sempre que es pugui assegurar la visió directa entre ells. Bàsicament, l'objectiu és unir xarxes Wi-Fi remotes mitjançant un radioenllaç. El fet que el radioenllaç treballi a la banda de 5 GHz evita que el sistema indueixi o rebi interferències a altres o d'altres sistemes. Segons l'entorn i la qualitat de l'enllaç, les cobertures en exteriors poden arribar fins als 500-1.000 m. Alguns experiments revelen cobertures més grans, però els transmissors i receptors s'emplaçaven a llocs elevats per tal d'assegurar un bon enllaç. En el cas d'entorns interiors, el senyal es degrada considerablement, la qual cosa fa que el sistema no sigui una opció viable per a aquest tipus d'escenaris.

El tipus de xarxa utilitzat a les variants comentades és de topologia en estrella, on un dispositiu actua d'estació base i la resta de dispositius s'hi connecten per tal de portar a terme les comunicacions. A més, és l'estació base la que defineix quin serà el canal (dels disponibles) que es farà servir. Com que es tracta d'un medi sense fil, l'accés al medi dels diferents dispositius és compartit i es porta a terme mitjançant el protocol d'accés múltiple amb escolta de portadora i evitament de col·lisió (CSMA/CA, *carrier sense multiple access with collision avoidance*). Aquest protocol és semblant al CSMA/CD comentat en el cas d'Ethernet, però hi ha algunes diferències, atès que no es treballa en un entorn cablejat. Bàsicament, es porten a terme les mateixes tasques: els dispositius que utilitzen aquest protocol escolten el medi abans de transmetre els seus paquets per tal d'evitar col·lisions. No obstant això, en el moment en què un dispositiu està transmetent la informació, aquest no pot portar a terme la tasca de *collision detection*. És a dir, no pot escoltar el medi a la vegada que està transmetent. Per tant, el mecanisme d'accés varia una mica en portar a terme un procés de *collision avoidance*. Bàsicament, si el dispositiu que vol transmetre detecta que el medi està sent utilitzat, s'espera un temps aleatori fins que no torna a provar a enviar. Aquest temps aleatori permet que si dos dispositius

intenten accedir al medi al mateix moment i s'han d'esperar, s'esperin temps diferents i, d'aquesta manera, es redueix la probabilitat de solapament de les properes transmissions.

Malgrat que la maduresa de la tecnologia ha arribat a un punt en què es disposa de maquinari molt optimitzat, el cost d'aquests dispositius és més gran que el dels dispositius Bluetooth o Zigbee. Per tant, la seva utilització en sistemes encastats serà determinada pel grau de compromís entre complexitat/cost enfront de velocitat de dades. És a dir, aquest sistema serà una opció viable si es volen aconseguir velocitats de transmissió altes. D'altra banda, també dependrà de la cobertura requerida; el Wi-Fi té un avantatge enfront dels seus competidors en aquest sentit. Cal dir que en els darrers anys ha proliferat la implementació de mòduls Wi-Fi, que faciliten la integració d'aquest estàndard a qualsevol sistema, la qual cosa simplifica la seva integració amb sistemes encastats.

3.2.4. IEEE 802.15.4 / ZIGBEE

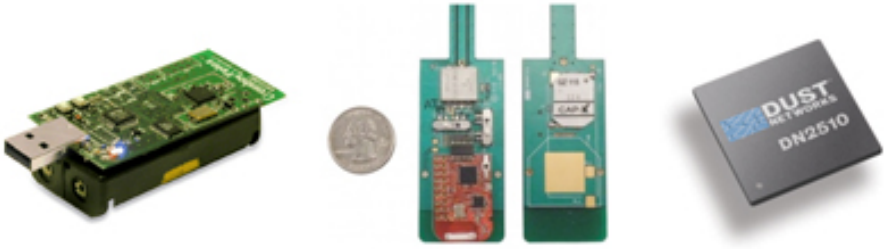
L'estàndard IEEE 802.15.4 va sorgir amb l'objectiu clar d'oferir un sistema de comunicacions ràdio de molt baix cost, d'un consum extremament baix i que permetés l'ús de dispositius de mida molt reduïda. Aquesta motivació era, en part, determinada per la proliferació de xarxes basades en l'ús de dispositius molt senzills que transmeten informació amb velocitats de transmissions molt baixes, les quals són conegudes com a *low-rate wireless personal area networks* (LR-WPAN).

Alguns exemples d'aquests tipus de xarxes són les utilitzades per a portar a terme tasques de monitoratge com poden ser el monitoratge de temperatura en edificis, de condicions climàtiques en boscos i muntanyes, control de la fauna, etc. En aquests tipus d'aplicacions es requereixen multituds de dispositius encastats de molt baixa complexitat i baix cost que portin a terme tasques molt senzilles (per exemple, mesurar la temperatura i enviar-la). També és important que el grau de manteniment sigui el mínim possible i la duració de la bateria sigui bastant alta. Per tant, cal tenir un sistema de comunicació que no encareixi la solució, no augmenti en gran mesura el consum de bateria i no condicioni la mida del dispositiu. Cal esmentar que aquests tipus de xarxes sense fil orientades a tasques de monitoratge, on cada node de la xarxa actua com a sensor d'algun esdeveniment físic, estan proliferant en els darrers anys gràcies als interessos suscitats a la indústria, atesa la reducció dels costos dels dispositius, i són conegudes com a **xarxes de sensors sense fil**[2].

[(2)]Comunament referides amb el terme anglès: *wireless sensor networks* (WSN).

En la figura següent, es presenten alguns exemples de nodes, o sensors, d'aquests tipus de xarxes. Aquests nodes són sistemes encastats en què s'integra tot el maquinari necessari per al seu funcionament autònom, com el sensor en si, el microcontrolador, memòries, font d'alimentació, equipament ràdio, etc. Com s'hi pot observar, el grau d'integració és bastant elevat.

Exemples de sensors de WSN comercials



Tornant a l'estàndard IEEE 802.15.4, les principals característiques d'aquest estàndard són que treballen a velocitats de transmissió reduïdes (fins a 250 kBps) i que s'utilitzen diferents bandes de treball, les quals poden variar segons el país d'aplicació. Les disponibles a Europa són:

- **868-868.6 MHz**: dividida en tres canals amb velocitats de 20 kBps, 100 kBps o 250 kBps.

- **2.400-2.483.5 MHz**: dividida en setze canals amb velocitats de 250 kBps.

Cal dir que també hi ha versions de l'estàndard amb velocitats superiors (de l'ordre de 27 MBps), però aquestes versions es fonamenten en l'ús de la tecnologia *ultra wide band* (UWB). Aquesta tecnologia es basa en la transmissió de polsos extremament estrets, per tant cal fer servir canals amb amples de banda molt grans (superiors a 500 MHz i per això es diuen així). No obstant això, al mercat no hi ha encara solucions competitives que treballin amb aquesta tecnologia. Al seu torn, UWB treballa amb amples de banda molt grans però a costa de reduir la potència de transmissió de manera dràstica i aquests sistemes estan principalment destinats per a treballar a distàncies molt petites.

En les versions de l'estàndard operatives a Europa (i en la majoria de versions disponibles), s'utilitza DSSS com a mitjà per a combatre les interferències d'altres tecnologies. Quant a la cobertura d'aquesta tecnologia, depèn de l'entorn, que varia des dels 10 metres en entorns interiors fins al 100 metres en entorns exteriors amb bones condicions.

Les topologies de xarxes permeses són les de tipus estrella i les d'igual a igual, en què tots els nodes es poden comunicar amb la resta (vegeu-ho en la figura següent). En tot cas, la xarxa necessita que un node actuï com a coordinador. Cal dir que, ateses les restriccions en termes de complexitat dels escenaris on IEEE 802.15.4 treballa, es defineixen dos tipus de nodes:

- **Nodes amb funcionalitat completa**[3]: aquests nodes poden portar a terme operacions amb certa càrrega computacional i poden retransmetre paquets d'altres nodes.

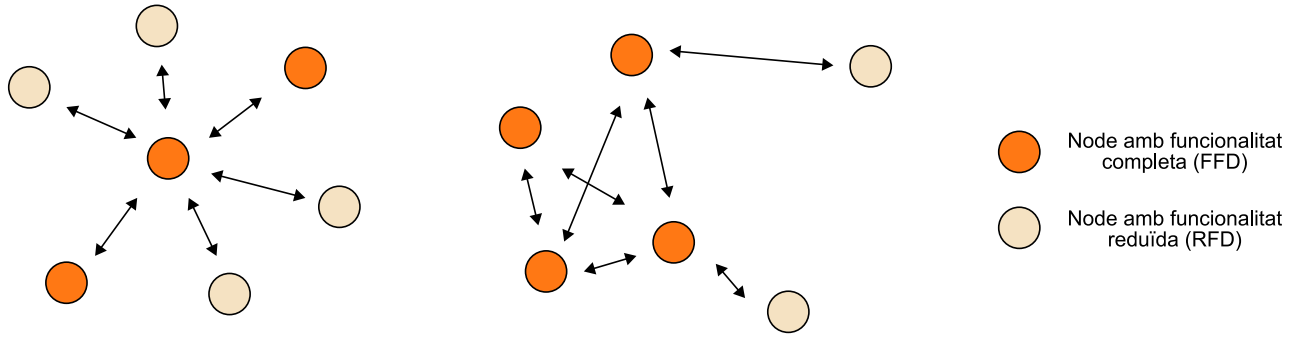
(3)Normalment referits com a *full functional devices* (FFD).

(4)Normalment referits com a *reduced functional devices* (RFD).

- **Nodes amb funcionalitat reduïda**[4]: aquests nodes tenen funcions limitades i únicament són capaços de comunicar-se amb nodes FFD.

Per tant, els coordinadors de la xarxa han de ser FFD forçosament. És a dir, és necessari que hi hagi un node d'aquest tipus per a establir una xarxa. Igualment, per tal de poder establir xarxes d'igual a igual, cal que hi hagi nodes d'aquesta natura a la xarxa per a poder crear les diferents connexions tal com es reflecteix en la figura.

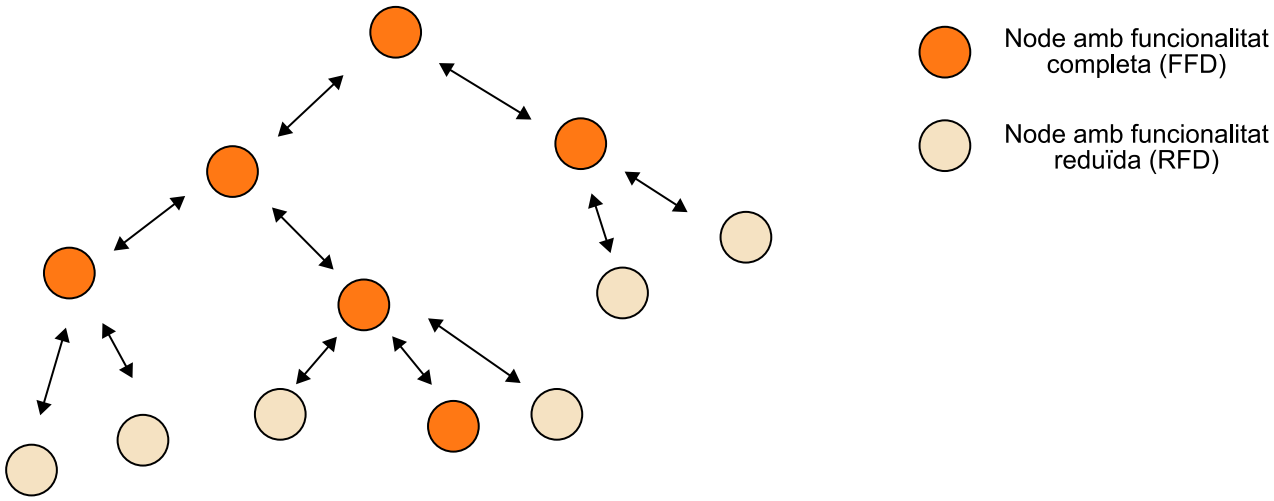
Topologies utilitzades a IEEE 802.15



Esquerra: topologia en estrella. Dreta: topologia d'igual a igual

Cal dir que una variant de topologia d'igual a igual molt utilitzada en aplicacions basades en IEEE 802.15.4 és la coneguda com a *topologia en arbre*, la qual es mostra en la figura següent. Aquesta topologia s'utilitza molt en entorns de monitoratge on hi ha un node que actua com a recol·lector de dades i la resta de nodes de la xarxa li van enviant les dades mesurades. Per tal d'estalviar energia, es transmet a baixa potència per a comunicar-se localment fent servir els nodes més propers i arribar al node recol·lector mitjançant un sistema de comunicació multisalt.

Topologia en arbre



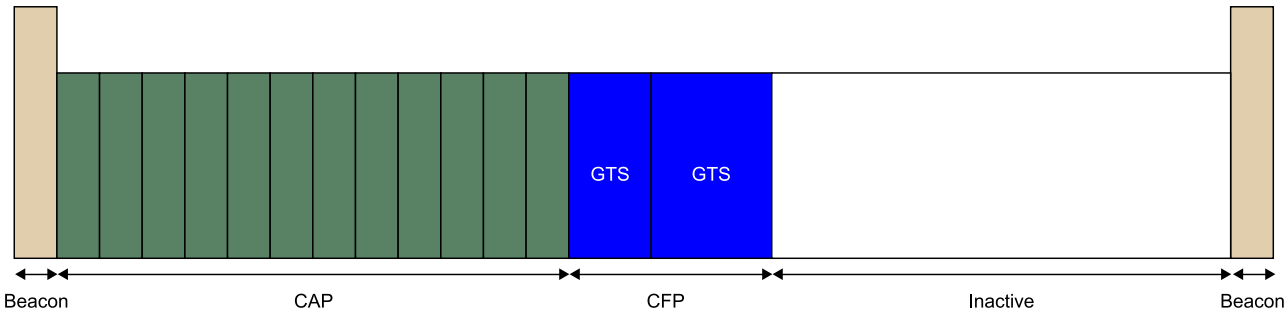
L'accés al medi es pot portar a terme fent servir dos modes d'operació diferents: mode *non-beacon* i mode *beacon*. Les característiques de cadascú d'aquests modes són les següents:

- **Mode *non-beacon***: en aquest mode els nodes de la xarxa no es troben sincronitzats i accedeixen al medi fent servir CSMA-CA. L'avantatge d'aquest mode és que la complexitat de la xarxa és bastant reduïda.

- **Mode *beacon***: en aquest mode el coordinador de la xarxa va enviant una sèrie de paquets coneguts com a *beacons*. Aquests *beacons* serveixen per a sincronitzar la resta dels nodes, de manera que se'ls indica quin és l'inici d'una supertrama, que és la unitat de transmissió utilitzada en aquest estàndard (vegeu-ho en la figura següent). Aquesta supertrama, per la seva banda, es divideix en tres períodes: el període CAP (*contention acces period*), el període CFP (*contention free period*) i un període inactiu (*inactive period*). En el període CAP, els nodes fan servir una versió modificada de CSMA-CA, coneguda com a CSMA-CA amb ranures (o *slotted* CSMA-CA), on es continua fent servir la mateixa estratègia d'accés compartit però les transmissions dels nodes s'han d'ajustar a una sèrie de *slots* temporals definida en la supertrama. En el període CFP, per la seva banda, el coordinador de la xarxa assigna uns *slots* determinats[5] als nodes que han demanat accés per tal d'assegurar que podran enviar dades. Això se sol utilitzar quan s'ha d'assegurar un ample de banda determinat a una sèrie de nodes de la xarxa. Finalment, el període inactiu s'introdueix per a reduir l'activitat dels nodes i, per tant, allargar la durada de les bateries.

[5] En anglès, *guaranteed time slots* (GTS).

Supertrama IEEE 802.15.4 en mode *beacon*



Normalment, a l'estàndard IEEE 802.15.4 s'associa el terme *Zigbee*. Zigbee no és altra cosa que una aliança d'empreses que s'encarrega d'assegurar la interoperabilitat dels seus dispositius basats en l'ús de l'IEEE 802.15.4. Atès que IEEE 802.15.4 defineix només les capes física i d'enllaç, Zigbee s'encarrega de definir les capes superiors per tal d'oferir aquesta interoperabilitat. Això no vol dir que tots els sistemes que treballin amb IEEE 802.15.4 utilitzin Zigbee, ja que hi ha altres opcions i alguns fabricants fan servir les seves pròpies. No obstant això, aquesta és una de les opcions més famoses.

Finalment cal esmentar que, clarament, aquest tipus de comunicació és dels més adequats per a ser implementats en sistemes encastats amb restriccions quant a mida, consum i cost. En el cas de WSN, és la solució preferentment

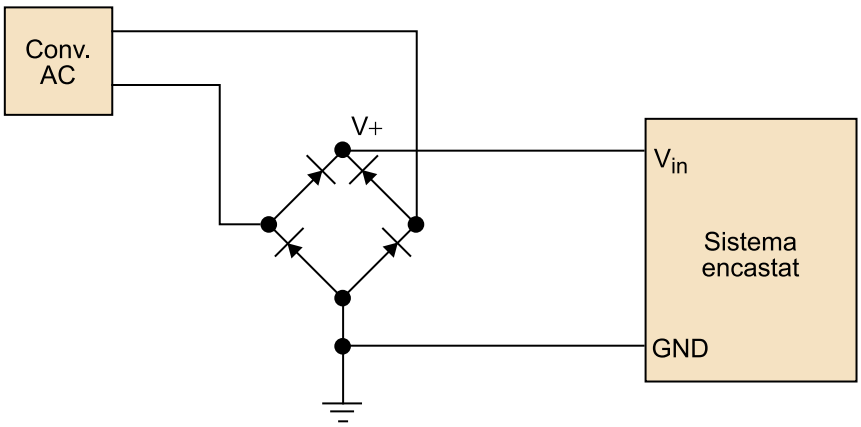
utilitzada. No obstant això, pot ser que l'aplicació concreta requereixi velocitats més altes o més cobertura. En aquest cas, les altres opcions ràdio esmentades en aquesta secció poden presentar una bona alternativa.

4. Subsistema d'alimentació

Lògicament, qualsevol dispositiu electrònic necessita una font d'alimentació per a poder funcionar. Depenent del tipus de dispositiu, la selecció del tipus de font i el disseny del subsistema d'alimentació varia.

En cas que el dispositiu no necessiti ser portable i disposi d'una presa de corrent propera, la millor opció es connectar-lo directament a la xarxa elèctrica. En treballar la xarxa elèctrica en corrent altern (ca) i amb alts nivells de voltatge (230 V en el cas d'Europa), s'ha de transformar per tal que pugui ser útil per al sistema encastat, el qual requereix corrent continu (cc) i un voltatge bastant inferior. Una de les opcions més viables per a fer això, en termes de simplicitat i baix cost, és l'ús d'un adaptador ca, el qual és fàcilment adquirible per qualsevol distribuïdor. Concretament, és com el que s'utilitza en el cas dels telèfons mòbils i altres dispositius d'informàtica de consum. Aquests dispositius transformen el corrent altern d'entrada en corrent continu amb un voltatge que sol ser entre els 5 i 12 V. Una cosa que s'ha de tenir en compte amb l'ús d'aquest tipus d'adaptadors és que el connector que tenen de sortida (que es connecta al sistema encastat) pot tenir diferent polaritats segons el fabricant. Per tant, abans de connectar-lo al sistema encastat s'ha d'assegurar que la polaritat sigui la correcta per a no provocar desperfectes al sistema encastat. Això es pot verificar mitjançant l'ús d'un **multímetre**. Una altra opció és proveir el sistema encastat d'un **pont de díodes**. Un pont de díodes, o pont rectificador, és un circuit que dóna sempre la mateixa polaritat a la sortida independentment de la polaritat d'entrada (vegeu la figura següent, en què les entrades V_{in} i GND es refereixen a les entrades d'alimentació i massa, respectivament). Això es deu a la característica pròpia dels díodes que només deixen passar corrent elèctric a una única adreça.

Esquema del subsistema d'alimentació dissenyat



En la resta de casos, l'opció més comuna és l'ús de bateries. En primer lloc, s'ha de fer una bona selecció de la bateria per tal d'assegurar que ofereix el voltatge correcte i un nivell de corrent suficient. En cas contrari, el sistema pot funcionar erròniament o directament no funcionar. D'altra banda, s'ha de considerar, a part del nivell de corrent mitjà que ofereix la bateria, quin és el corrent de pic que pot oferir.

Exemple

Alguns sistemes encastats poden necessitar només un corrent mitjà de 20 mA, però requerir en alguns moments corrents de pic de 100 mA. Això sol succeir en sistemes amb memòries flaix, les quals requereixen alts corrents quan porten a terme operacions d'escriptura.

En segon lloc, s'ha de tenir un sistema ben dissenyat ja que això pot condicionar bastant la durada de la bateria que pot anar des de qüestions de minuts (si està dissenyat força malament) fins a alguns anys, segons el tipus de maquinari del sistema encastat i l'aplicació. Per a portar a terme aquest disseny el primer pas és tenir un sistema encastat que faci servir dispositius de baix consum. El consum de potència entre xips diferents pot variar força i, molt sovint, hi ha versions de baix consum del dispositiu que es vol integrar. D'altra banda, molts perifèrics i xips de memòria entren de manera automàtica en mode de baix consum quan no estan en ús. Uns altres poden ser posats en aquest mode en accionar una entrada digital o mitjançant una ordre de programari. Depenent de l'aplicació, una opció també pot ser implementar una solució en què se separin els circuits d'alimentació d'alguns dispositius del sistema per a poder-los desconnectar, via control de programari, quan aquests no siguin utilitzats. Finalment, alguns dispositius de molt baix consum, com els sensors, necessiten molt poc nivell de corrent per a funcionar. Per tant, una opció és alimentar-los directament amb les línies d'entrada i sortida pròpies del microcontrolador o microprocessador del sistema. És a dir, es fa servir una línia d'entrada i sortida per a alimentar el dispositiu, ja que aquestes línies poden donar nivells de corrents suficients (20 mA en alguns casos) i, simultàniament, s'aprofita aquesta línia per a activar o desactivar el dispositiu quan sigui necessari.

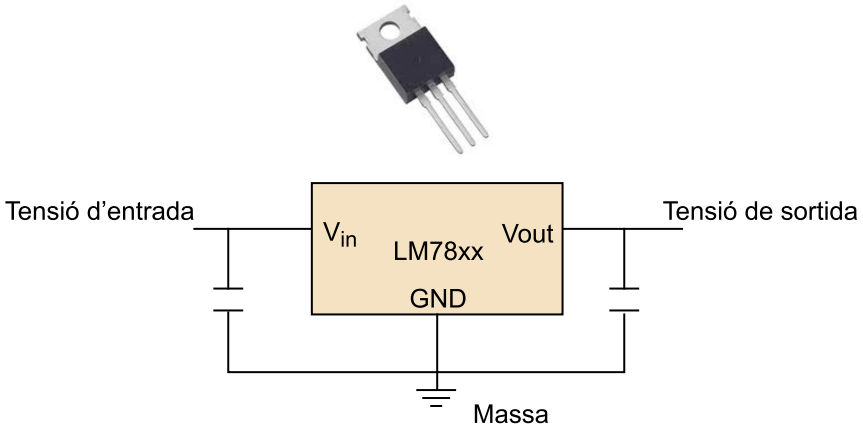
Deixant ara de banda el tipus de font d'alimentació que utilitzi el sistema, és recomanable que el subsistema d'alimentació del sistema encastat faci servir reguladors de voltatge. Un regulador de voltatge és un dispositiu que assegura un nivell fix de tensió contínua a la seva sortida malgrat que la tensió a l'entrada adopti diferents valors o pateixi variacions o soroll. Aquesta opció és recomanable per a assegurar que el nivell de tensió que s'aplica als components del sistema encastat és la desitjada i es manté constant. Tot i que molts dels components són robustos a variacions del nivell de tensió, d'altres són molt sensibles, com, per exemple, els convertidors analògic-digitals que fan la conversió de l'entrada analògica comparant el nivell de tensió d'entrada amb el nivell de tensió d'alimentació del mateix convertidor. És a dir, la quantificació del senyal d'entrada es fa prenent la tensió d'alimentació com a referència. D'altra banda, si el sistema encastat treballa amb bateries, el regulador pot ser un mecanisme eficaç per a combatre les variacions de tensions ocasionades pel mateix consum de la bateria. En la figura següent, es mostra un exemple de re-

gulador comercial, en què les tres potes es corresponen amb la tensió d'entrada (referida normalment com a V_{in}), la tensió de sortida (V_{out}) i la connexió de massa (GND). Aquest regulador concret és referit amb el nom de LM78xx, el qual és fabricat per diverses companyies (com Fairchild i ST Microelectronics) i en què xx es refereix al voltatge de sortida. A la part dreta de la mateixa figura, es mostra el circuit que se sol utilitzar per a la seva integració, el qual es coneix com a *regulador lineal*. Tal com es pot veure, es fa servir un condensador a l'entrada i un altre a la sortida. En aquesta configuració, els condensadors es denominen *condensadors desacobladors* i el seu objectiu és filtrar el soroll electromagnètic present en la línia d'alimentació. Cal esmentar també que alguns reguladors tenen entrades addicionals que actuen com a interruptors per a l'activació o desactivació del circuit. Això pot simplificar el procés d'optimització de consum comentat en el paràgraf anterior quant a desactivar dispositius que no estiguin sent utilitzats.

Exemple

En el cas de LM7805, la sortida seria igual a 5 V.

Regulador comercial (esquerra) i circuit d'un regulador lineal (dreta)



Un altre tema que cal tenir en compte a l'hora de dissenyar el subsistema d'alimentació és que les mateixes línies del subsistema poden causar interferències als senyals que fan servir els altres components del sistema encastat, la qual cosa provoca l'aparició de dades corruptes o, fins i tot, el mal funcionament del sistema. Aquest efecte és bàsicament el soroll electromagnètic comentat en seccions anteriors i algunes possibles pautes que cal seguir per a reduir-lo són les següents:

- **Reduir l'àrea del bucle de corrent**: el bucle de corrent és el camí descrit pel corrent que va alimentant cada dispositiu, que surt de la font per la línia d'alimentació i retorna per la línia de massa. En aquest bucle es generen camps magnètics, els quals són induïts pels corrents que circulen en sentits oposats per ambdues línies. Aquests camps magnètics són causants de soroll electromagnètic i una manera de reduir-lo és col·locant les línies d'alimentació i de massa properes ja que, d'aquesta manera, els camps magnètics generats per cada línia es cancel·len entre ells. Bàsicament, l'objectiu

de tenir un bon disseny és reduir l'àrea del bucle de corrent mitjançant la minimització de la longitud de les línies i apropant aquestes línies el màxim possible. Una opció per a fer això és utilitzant un pla de massa. És a dir, fer servir una superfície conductora gran per a enllaçar-hi totes les connexions de massa i minimitzar, d'aquesta manera, el camí de retorn.

- **Utilització de condensadors desacobladors**: la solució proposada en el punt anterior pot ser difícilment implementable en alguns sistemes perquè la tasca de distribuir l'alimentació, a la vegada que s'assegura un bucle de corrent amb àrea reduïda, pot ser complicada. Una solució al problema és posar a l'entrada d'alimentació de cada component del sistema un mecanisme per tal de conduir el soroll present a la línia cap a massa. Això es pot fer posant condensadors desacobladors a les entrades de cada circuit. Bàsicament, una de les potes del condensador es connecta amb la línia d'alimentació i l'altra a massa. Tenint en compte el principi de funcionament d'un condensador, si la tensió d'entrada és contínua el condensador actua com a circuit obert (és com si no hi hagués res connectat a la línia d'alimentació). Si hi ha alguna fluctuació de corrent, el condensador es va carregant i descarregant, de manera que actua com si conduís el soroll a terra, ja que es manté el nivell de voltatge constant. D'aquesta manera, el soroll pot ser eliminat del subsistema d'alimentació. Tot i que aquesta solució és bastant antiga, es continua fent servir, ja que es considera eficaç.

5. Detecció d'errors de maquinari

Assegurar la fiabilitat dels sistemes encastats és de gran importància, atesos els tipus d'escenaris en què treballen aquests dispositius. En comparació dels sistemes informàtics convencionals, on normalment treballen a entorns adequats, els sistemes encastats troben usualment entorns d'operació amb altes temperatures, humitat, brutícia i soroll electromagnètic, a part del fet que poden estar alimentats amb un sistema de distribució pobre. Per tant, la fiabilitat del maquinari ha de ser realment alta per a poder operar en aquestes condicions. A part, molts dels dispositius poden ser emprats per usuaris amb molt poc coneixement quant a una manipulació adequada del dispositiu. Els dissenys hauran de tenir en compte possibles mals usos del sistema i accions inesperades ja siguin intencionades o no.

Tot i que els sistemes actuals gaudeixen d'uns processos de disseny i verificació acurats i portats a terme per tal de reduir els errors de funcionament dràsticament, aquests errors es poden produir sempre. A més, molts dels sistemes encastats es fan servir a localitzacions remotes, de difícil accés i/o es munten amb la idea de gaudir d'un manteniment limitat o pràcticament nul. Per tant, és de vital importància que el sistema de disseny tingui en compte accions per a detectar possibles errors o mals funcionaments del sistema i s'incloguin mecanismes de recuperació autònoms.

Una tècnica que s'utilitza molt sovint per a detectar errors del sistema i per a portar a terme la seva recuperació és l'ús de sistemes de monitoratge maquinari i programari coneguts com a *temporitzadors de vigilància* (***watchdogs***).

Temporitzadors de vigilància

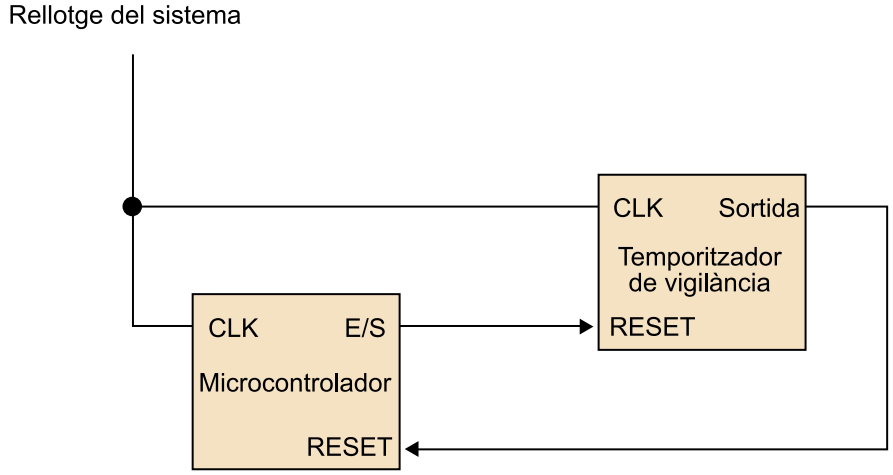
Els temporitzadors de vigilància (*watchdog*, en anglès; 'gos guardià', en català) fan referència a un sistema maquinari o programari encarregat de vigilar si el sistema funciona correctament. Normalment, es basa en l'ús de comptadors temporals i en el control que esdeveniments esperats vagin apareixent amb normalitat. Si succeeix una anomalia, el temporitzador de vigilància s'encarrega de reinicialitzar el sistema.

Cal esmentar que temporitzadors de vigilància per a programari són inclosos en diversos sistemes operatius com ho són els casos de Linux i TinyOs, els quals són sistemes operatius força utilitzats en sistemes encastats i en les xarxes WSN comentades anteriorment.

Quant a les opcions de maquinari, aquestes consisteixen en la inclusió d'un comptador digital i d'un circuit de control addicional al sistema. En la figura següent, es mostra un exemple de configuració. Aquí, el temporitzador de vigilància disposa d'un comptador intern que quan arriba a zero activa la seva sortida. Tal com es pot veure, aquesta sortida està connectada amb el RESET del microcontrolador del sistema encastat; per tant, la seva funció es reinicialitzar-lo. Al mateix temps, el microcontrolador té una de les seves línies d'entrada/sortida (I/O) connectades al RESET del temporitzador de vigilància. Bàsicament, la idea és que el microcontrolador vagi reinicialitzant periòdicament el comptador del temporitzador de vigilància per a evitar que arribi a

zero. Si el sistema sofreix algun mal funcionament, el reinici del comptador no es durà a terme i, per tant, el temporitzador de vigilància acabarà reinicialitzant el sistema encastat.

Esquema de temporitzador de vigilància de maquinari



L'ús d'una opció de programari presenta, per tant, una opció menys complexa d'implementar i amb un cost més reduït. El problema, però, és que no tenen una fiabilitat tan alta com les de maquinari, i es pot donar el cas que no reinicialitzin el sistema quan calgui.

A banda dels temporitzadors de vigilància, també hi ha sistemes de monitoratge de l'estat del maquinari basats, per exemple, en el mesurament de la temperatura d'algunes parts del sistema o del voltatge en algun punt. En el moment en què es detecta algun sobreescalfament o sobrevoltatge, el sistema porta a terme les accions més adients. Aquests sistemes proporcionen una solució interessant per a poder actuar d'una manera eficaç i, fins i tot, preveure mals funcionaments del sistema però requereixen sempre maquinari addicional per a poder implementar el sistema de mesura i monitoratge dels esdeveniments físics.

Resum

En aquest mòdul, s'han estudiat els diferents elements que formen els sistemes encastats, tant des del punt de vista del mateix maquinari com de la interfície entre aquest i el maquinari. Així, s'introdueix, en primer lloc, l'arquitectura bàsica de qualsevol sistema encastat amb els seus elements essencials, que són: processador, memòria, dispositius d'entrada i dispositius de sortida. En segon lloc, es detallen els diferents tipus de processador que hi ha (microprocessador, microcomputador, microcontrolador o bé DSP) i s'analitza com interactuen amb la resta dels elements del sistema per mitjà dels busos de comunicacions (adreces, dades i control). També ha estat objectiu d'estudi la interacció entre màquina i home a nivell de programació, és a dir, s'ha vist quines són les operacions que un processador pot dur a terme (conjunt d'instruccions), com s'indiquen (codi màquina) i quina forma tenen dins el sistema (instruccions binàries). A més a més, s'han estudiat les diferents formes d'execució d'aquestes instruccions (entre d'altres, la seqüencial, en llaç o la crida a subrutines). A partir d'aquí, s'ha posat espacial èmfasi en el disseny del subsistema de memòria, ja que és una part essencial de qualsevol sistema encastat.

A continuació, s'ha dividit la memòria en els tipus RAM o d'accés aleatori i ROM o de només lectura. Posteriorment, s'ha ampliat la classificació per a intentar donar cabuda als principals tipus de memòria que podem trobar al mercat. Després d'això, s'ha treballat la interacció entre memòria i processador, i s'han detallat els senyals que hi intervenen i la temporització que segueixen. Després, s'han estudiat dos casos pràctics de disseny, els quals pretenen recollir la major part de problemes amb què ens podem trobar en una situació real. En particular, s'ha implementat un sistema de memòria basat en SRAM i un altre basat en DRAM. La darrera part dedicada a la memòria ha incidit en alguns dels mecanisme de *caching* més destacats, els quals permeten traspassar petites quantitats de dades de la memòria principal a una memòria més petita i molt més ràpida per tal d'operar-hi i agilitar, així, les operacions del processador.

Finalment, la part de components de maquinari conclou amb l'estudi de peces més petites però no per això menys importants per als sistemes encastats, que són els diferents tipus de registres, els comptadors i divisors i el rellotge del sistema. Els registres poden servir, entre d'altres, com a petites memòries temporals (per a guardar una sola paraula), per a fer conversions sèrie a paral·lel i viceversa i també com a memòries d'entrada sèrie i sortida sèrie. A part, tenen gran ús en els generadors de nombres pseudoaleatoris. Els comptadors i divisors, que s'estudien junts, ja que utilitzen una arquitectura comuna amb finalitats diferents, ens serveixen per a acumular esdeveniments, mesurar retards temporals, establir comptes enrere o bé obtenir rellotges amb diferents bases temporals a partir d'un únic rellotge de sistema, entre d'altres. Per acabar, es repassen els diferents tipus de rellotges que hi ha, ja que és molt important

escollir-ne bé les característiques (estabilitat i precisió) per tal d'assegurar el funcionament correcte del sistema. En el fons, és el rellotge qui marca el ritme d'execució de les diferents operacions en el sistema i s'ha d'assegurar que entre dos cops de rellotge hi hagi sempre prou temps per a fer les tasques encomanades.

Seguidament, s'han presentat els mecanismes de comunicació que poden ser utilitzats per a comunicar els sistemes encastats amb l'exterior. Concretament, s'han enumerat les opcions més comunes i, per cada cas, s'han discutit els avantatges i desavantatges que es troben en aplicar-se en sistemes encastats. Per a fer això, s'han dividit els mecanismes en dos grups d'acord amb l'estratègia utilitzada per a dur a terme la comunicació: mitjançant ports de comunicació (amb cablejat) o fent servir estàndards de comunicacions sense fil. En el primer cas, s'ha fet un repàs de les estratègies més clàssiques (com ho són el port sèrie, paral·lel i Ethernet) fins a les que es troben de manera més comuna en els darrers anys (com USB i Firewire). També s'ha tingut en compte el port CAN, que és un dels ports més utilitzats en entorns industrials. En el cas de les comunicacions sense fil, s'ha començat la discussió presentant la tecnologia IRDA, la qual es basa en infrarojos i va ser molt utilitzada al final de la dècada de 1990 i a l'inici de la de 2000. A continuació, s'han esmentat sistemes de comunicacions sense fil basats en estàndards ràdio, concretament els casos més adients per als tipus d'escenaris trobats en els sistemes encastats. Aquests estàndards són el Bluetooth (molt utilitzat a xarxes tipus WPAN), el Wi-Fi (opció predominant a les WLAN) i l'IEEE 802.15.4/Zigbee (solució cada cop més present, atesa en part a la proliferació de les WSN).

Després d'això, s'ha presentat el subsistema d'alimentació, que és el subsistema encarregat d'alimentar els diferents components d'un sistema encastat. S'han diferenciat aquells casos en què es fa servir la xarxa elèctrica d'aquells que opten per l'ús de bateries i s'han enumerat algunes estratègies enfocades a optimitzar la distribució d'alimentació del sistema. A més, s'ha fet referència al soroll electromagnètic provocat pel mateix subsistema d'alimentació, el qual pot interferir en els diferents components del subsistema. En aquesta adreça, s'han descrit tècniques per tal de reduir aquest soroll basades en la reducció del bucle de corrent del subsistema (per a reduir directament el nivell de soroll generat) o en la utilització de condensadors desacobladors a les entrades dels diferents components (per a filtrar el soroll present).

Finalment, s'ha adreçat el problema de detecció d'errors de maquinari durant el funcionament del sistema. Com que molts dels sistemes encastats es fan servir a localitzacions remotes o de difícil accés, és necessari que disposin de mecanismes de recuperació autònoms. En aquest mòdul, s'ha fet referència a una de les opcions més utilitzades en sistemes encastats: els temporitzadors de vigilància. Tal com s'ha comentat, un temporitzador de vigilància no és altra

cosa que un sistema que s'encarrega de vigilar si el sistema funciona correctament i pot ser de tipus programari o maquinari; el segon cas és una solució més complexa però més robusta.

Activitats

En aquest apartat es proposen alguns exercicis que il·lustren les explicacions que s'han donat en aquest mòdul.

1. Es vol programar la funció void funcio1(int *a, int *b, int c) en codi màquina. Considereu el fragment de codi i l'estat de la pila en el moment de la crida de la funció que apareixen a continuació i identifiqueu-hi els errors existents.

```
funcio1:      MOV BP, SP
              PUSH CX
              PUSH BX
              PUSH AX

              ... (Cos de la funció) ...

              POP BX
              POP AX
              RET
```

2. Disposem de blocs de memòria de 2 MB SRAM per a construir la memòria d'un sistema encastat que treballa amb paraules de 32 bits i necessita poder adreçar 8 milions de paraules en total. Amb aquesta informació, dissenyeu el subsistema de memòria.

3. Es pretén dissenyar el temporitzador de vigilància d'un sistema encastat que treballa amb un rellotge d'1 MHz. El temporitzador de vigilància s'ha d'activar aproximadament cada 500 ms si abans no s'ha activat l'entrada de *reset* corresponent. Per a construir-lo, es disposa d'un comptador síncron de 9 bits amb entrada de *reset* asíncrona i dels biestables i portes lògiques necessàries. Dissenyeu el temporitzador de vigilància en qüestió.

4. Un sistema encastat s'encarrega d'emmagatzemar i transmetre les dades d'una estació meteorològica situada a alta muntanya. Durant el funcionament normal, es transmeten les dades pel Wi-Fi. No obstant això, es dota el sistema d'una targeta SD de 32 GB de capacitat per a emmagatzemar-hi les dades en el cas de no poder establir la comunicació via ràdio. El sistema incorpora, a més, un port USB per a poder bolcar les dades a un disc dur extern si fa falta. Tenint en compte que el sistema és capaç de llegir la SD a una velocitat de 125 K paraules per segon i que la longitud de paraula és de 8 bits, dissenyeu el mòdul de conversió paral·lel a sèrie necessari per a extreure les dades si el rellotge del sistema és d'1 MHz.

5. En una aplicació de videovigilància es vol connectar una videocàmera digital a un sistema encastat encarregat d'emmagatzemar els vídeos enregistrats. La videocàmera digital treballa a 30 FPS (*frames per second* o imatges per segon) amb una resolució de 640 × 480 píxels, en què cada píxel és codificat amb 24 bits. En aquest escenari us demanem seleccionar un mecanisme de comunicació per a connectar el sistema encastat amb la videocàmera. S'ha de tenir en compte que la videocàmera es troba emplaçada a 4 metres del sistema encastat i que es vol assegurar que aquest rebi els vídeos de la videocàmera amb un flux constant de transmissió, el qual sigui prou alt per a operar amb el mínim retard. A més, això es vol fer amb el mínim cost possible i garantint que la transmissió de vídeo no representi un treball extra per al microprocessador del sistema.

6. Considereu una aplicació en què es volen connectar dos sistemes encastats que estan separats 60 m. Per a portar a terme la comunicació, s'opta per l'ús de tecnologia ràdio basada en l'estàndard IEEE 802.15.4 que opera a la banda de 868 MHz. Concretament, es fan servir dispositius amb una potència de transmissió igual a 0 dBm i una sensitivitat al receptor de –82 dBm, en què la sensibilitat es defineix com la potència mínima necessària per a assegurar que el senyal es pot rebre correctament. Suposant que l'escenari es pugui modelar amb la fórmula de Friis fent servir un valor de γ igual a 3, justifiqueu si l'ús d'aquesta configuració ràdio és adient.

7. Continuant amb l'escenari del problema anterior, proposeu una alternativa de tecnologia ràdio que compleixi els requisits presentats. És a dir, se suposa un escenari en què els únics dispositius IEEE 802.15.4 disponibles són els esmentats en el problema anterior però, no obstant això, es pot fer servir algun dispositiu d'un altre tipus de tecnologia ràdio. El fet que s'hagi de canviar de banda freqüencial no comporta cap problema i es continua suposant que, independentment de la tecnologia seleccionada, la sensibilitat del receptor continua sent igual a –82 dBm. Tingueu en compte que la consideració inicial d'utilitzar IEEE 802.15.4 era determinada pel fet que es vol una solució de baix cost i de baix consum.

8. Es vol dissenyar el subsistema d'alimentació d'un sistema encastat. Aquest sistema fa servir la xarxa elèctrica com a font d'alimentació i és format per un microcontrolador, una memòria ROM, una memòria DRAM, una memòria SRAM i un convertidor analògic-digital. Dibuixeu

l'esquema del subsistema sabent que el convertidor necessita una entrada estable a 5 V i tenint en compte que es vol aconseguir un sistema robust a soroll electromagnètic i altres possibles anomalies.

Exercicis d'autoavaluació

1. El format d'una instrucció en codi màquina ha de contenir...

a) el codi d'operació i l'adreça de memòria on s'indica la ubicació dels operands.
b) només el codi d'operació.
c) les adreces dels operands i l'adreça on es troba la instrucció.
d) el codi d'operació i les adreces dels operands, habitualment entre 1 i 3.

2. En un processador, l'ALU...

a) s'encarrega de fer els càlculs numèrics i avaluar les operacions lògiques.
b) s'encarrega de la interacció entre el processador i la memòria.
c) s'encarrega d'agafar les instruccions i descodificar-les.
d) no forma part dels processadors en general, només dels DSP.

3. Les memòries flaix són...

a) memòries de tipus EEPROM. Per tant, la lectura és més ràpida que l'escriptura.
b) memòries de tipus SRAM, ja que no necessiten refrescament per a mantenir la informació que emmagatzemen.
c) memòries DRAM que incorporen un subsistema d'alimentació per a conservar la informació.
d) memòries RAM capaces de mantenir la informació durant unes hores en cas que falli el subsistema d'alimentació.

4. Indiqueu quina de les afirmacions següents és incorrecta referent a la memòria cau.

a) Només té sentit utilitzar-la quan sabem que l'activitat del programa se centra en una petita zona de la memòria durant un període de temps.
b) Treballa sempre amb blocs de la memòria principal.
c) Sempre s'ha d'estar al cas dels canvis que es produeixen en cau per a després reflectir-los a la memòria principal.
d) És la que s'utilitza en els registres interns del processador per a les operacions bàsiques.

5. Els senyals CAS i RAS...

a) són típics de les memòries SRAM.
b) s'utilitzen en les memòries DRAM, típicament més grans, per a indicar l'adreça de la dada en dos cops.
c) només s'utilitzen en algunes memòries ROM antigues.
d) avui en dia estan en desús.

6. Respecte a un comptador de Johnson amb N registres, podem afirmar que...

a) el període és de $2N$ i de vegades apareixen estats prohibits.
b) el període és de $2N - 2N$ i no sempre apareixen estats prohibits.
c) el període és de $2N - 2N$ i sempre hi ha $2N$ estats prohibits.
d) el període és de $2N$ amb $2N - 2N$ estats prohibits.

7. Un dels principals avantatges del comptador de Johnson respecte del comptador síncron amb el mateix nombre de variables d'estat és que...

a) no té cap avantatge, ja que el període és inferior i a més cal controlar els estats prohibits.
b) és més senzill d'implementar.
c) permet descodificar les sortides assegurant-nos que no tindrem transitoris indesitjats.
d) es pot utilitzar com a divisor de freqüència.

8. Els registres LFSR són...

a) registres de desplaçament que s'utilitzen per a retardar un senyal.
b) registres de desplaçament realimentats segons un polinomi de connexions que s'utilitzen per a generar seqüències pseudoaleatòries.

c) un tipus de convertidor sèrie a paral·lel.
d) un tipus de convertidor paral·lel a sèrie.

9. Respecte a la pila del programa, no és cert que...

a) sigui una part de memòria a la qual s'accedeix exclusivament seguint una política LIFO.
b) s'utilitzi per a la crida de subrutines.
c) tingui un punter de pila exclusivament dedicat a apuntar l'última posició ocupada.
d) dins d'una subrutina, la puguem usar per a guardar els registres del processador i evitar així efectes col·laterals.

10. Respecte a la representació binària d'una paraula...

a) la representació és única; sempre el bit de més a la dreta és el de més pes.
b) la representació és única però el bit de més a la dreta és el de menys pes.
c) hi ha dues representacions possibles, que són Big Endian, en què el bit de més a la dreta és el de menys pes, i Little Endian, en què el bit de més a la dreta és el de més pes.
d) hi ha dues representacions possibles, que són Big Endian, en què el bit de més a la dreta és el de més pes, i Little Endian, en què el bit de més a la dreta és el de menys pes.

11. Indiqueu quina de les afirmacions següents és certa respecte de modes d'adreçament.

a) El mode d'adreçament indirecte és més ràpid que el directe.
b) En primera opció, utilitzarem sempre el mode d'adreçament immediat.
c) El mode indexat és adequat per a accedir a estructures de dades.
d) En els sistemes basats en i8086, podem utilitzar el punter de pila per a accedir a les dades d'aquesta usant el mode d'adreçament indirecte de registre.

12. La memòria SDRAM és...

a) un tipus de memòria SRAM d'altes prestacions.
b) una memòria DRAM que es converteix en estàtica gràcies al fet que implementa el refrescament de manera interna.
c) una memòria RAM síncrona i per tant més ràpida que la DRAM convencional.
d) la memòria que hi ha en les targetes de memòria SD.

13. Indiqueu quina afirmació és correcta.

a) En l'estàndard Firewire 800, s'utilitzen normalment connectors de 4 pius, en què els 4 pius són utilitzats per a enviar dades.
b) En l'estàndard Firewire 800, s'utilitzen normalment connectors de 6 pius, en què 5 pius són utilitzats per a enviar dades i el pin restant per a proveir alimentació.
c) En l'estàndard Firewire 800, s'utilitzen normalment connectors de 6 pius, en què 4 pius són utilitzats per a enviar dades i els 2 restants per a proveir alimentació.
d) En l'estàndard Firewire 800, s'utilitzen normalment connectors de 9 pius, en què 6 pius són utilitzats per a enviar dades i els 3 restants per a proveir alimentació.

14. Indiqueu quina afirmació és falsa.

a) En una xarxa de dispositius USB formada per tres dispositius, en què dos treballen amb la versió USB 2.0. i un amb la versió 1.1, la velocitat màxima que podran utilitzar els dispositius USB 2.0 serà de 12 MBps.
b) En una xarxa de dispositius Firewire formada per tres dispositius, en què dos treballen amb la versió Firewire 800 i un amb la versió 400, la velocitat màxima que podran utilitzar els dispositius Firewire 800 serà aproximadament de 800 MBps.
c) En una xarxa de dispositius Firewire formada per tres dispositius, en què dos treballen amb la versió Firewire 400 i un amb la versió 800, la velocitat màxima que podran utilitzar el dispositiu Firewire 800 serà aproximadament de 800 MBps.
d) En una xarxa de dispositius USB formada per tres dispositius, en què dos treballen amb la versió USB 1.1. i un amb la versió 2.0, la velocitat màxima que podrà utilitzar el dispositiu USB 2.0 serà de 12 MBps.

15. La variant d'Ethernet a 10 MBps més utilitzada és...

a) 10Base2 amb parell trenat i 185 m de longitud màxima de cable.
b) 10Base2 amb cable coaxial i 100 m de longitud màxima de cable.
c) 10BaseT amb parell trenat i 100 m de longitud màxima de cable.
d) 10BaseT amb cable coaxial i 185 m de longitud màxima de cable.

16. Indiqueu quina de les afirmacions següents relacionades amb Ethernet és correcta.

a) Un concentrador retransmet el paquet rebut a tots els dispositius que té connectats i un commutador reenvia només el paquet rebut al dispositiu de destinació.
b) Un commutador reenvia només el paquet rebut al dispositiu de destinació i un encaminador retransmet el paquet rebut a tots els dispositius que té connectats.
c) Un commutador retransmet el paquet rebut a tots els dispositius que té connectats i un encaminador reenvia només el paquet rebut al dispositiu de destinació.
d) Un concentrador reenvia només el paquet rebut al dispositiu de destinació i un commutador té l'habilitat d'encaminar el paquet per la millor sortida en termes de la sortida que ofereix el camí més curt i menys congestionat.

17. Els dispositius que treballen amb CAN utilitzen com a mètode d'accés al medi...

a) CSMA/CD.
b) CSMA/CD+AMP.
c) CSMA/CA.
d) CSMA/CA+AMP.

18. El sistema IRDA és adequat quan...

a) es volen connectar dos sistemes encastats que es troben a 3 m.
b) es volen connectar dos sistemes encastats separats per una paret i que es troben a una distància de 50 cm l'un de l'altre.
c) es vol transmetre informació de manera segura.
d) es vol transmetre informació a alta velocitat.

19. Indiqueu quina és l'afirmació més encertada.

a) Bluetooth és una estratègia útil per a portar a terme comunicacions entre dos sistemes encastats amb altes limitacions de bateria, els quals estan separats 200 m i equipats amb dispositius de classe 1.
b) Bluetooth és una estratègia útil per a portar a terme comunicacions entre dos sistemes encastats amb altes limitacions de bateria, els quals estan separats 10 m i equipats amb dispositius de classe 1.
c) Bluetooth és una estratègia útil per a portar a terme comunicacions entre dos sistemes encastats amb altes limitacions de bateria, els quals estan separats 10 m i equipats amb dispositius de classe 2.
d) Bluetooth és una estratègia útil per a portar a terme comunicacions entre dos sistemes encastats amb altes limitacions de bateria, els quals estan separats 10 m i equipats amb dispositius de classe 3.

20. Indiqueu quina de les afirmacions següents és falsa.

a) Wi-Fi basat en IEEE 802.11a treballa amb OFDM i és útil per a unir xarxes remotes, sempre que hi hagi un enllaç de visió directa entre elles.
b) Wi-Fi basat en IEEE 802.11g pot utilitzar DSSS.
c) Wi-Fi basat en IEEE 802.11g és una opció aconsellable per a sistemes encastats situats dintre d'una nau industrial amb una aplicació que requereix connexió amb Internet d'alta velocitat.
d) Wi-Fi basat en IEEE 802.11b opera amb OFDM i pot oferir connexions a 30 m de distància.

21. Un conjunt de sistemes encastats que treballen amb IEEE 802.15.4 i formats per nodes tipus RFD poden...

a) formar una xarxa d'igual a igual.
b) formar una xarxa en estrella.
c) fer servir DSSS com a tècnica de transmissió.
d) formar una xarxa en arbre.

22. Indiqueu quina de les afirmacions següents és falsa.

a) Una xarxa de sistemes encastats que utilitzen IEEE 802.15.4 en mode *beacon* no fa servir CSMA-CA.
b) Una xarxa de sistemes encastats que utilitzen IEEE 802.15.4 en mode *non-beacon* fa servir CSMA-CA.
c) Una xarxa de sistemes encastats que utilitzen IEEE 802.15.4 en mode *beacon* pot fer servir tant CSMA-CA com pot tenir una sèrie de *slots* assignats pel coordinador.
d) Una xarxa de sistemes encastats que utilitzen IEEE 802.15.4 en mode *beacon* poden tenir una sèrie de *slots* assignats pel coordinador.

23. Indiqueu quina estratègia no és adient per a portar a terme el disseny del subsistema d'alimentació d'un sistema encastat.

a) Incloure un pont de díodes a l'entrada d'alimentació del sistema encastat quan aquest s'alimenta mitjançant un adaptador ca.
b) Alimentar tots els dispositius amb la font d'alimentació, inclosos els dispositius de baix consum com els sensors i els LED.
c) Utilitzar un pla de massa.
d) Incloure condensadors desacobladors a les entrades d'alimentació dels components del sistema.

24. En un sistema encastat, un temporitzador de vigilància...

a) és un sistema programari que s'encarrega de vigilar si les dades enviades a un bus de dades contenen errors.
b) és un sistema maquinari encarregat d'assegurar que tots els circuits tenen un nivell d'energia adequat.
c) presenta una solució menys complexa quan s'implementa a nivell programari, però presenta una resposta menys robusta que la seva versió maquinari.
d) presenta una solució més complexa quan s'implementa a nivell programari, però presenta una resposta més robusta que la seva versió maquinari.

Solucionari

Activitats

1. En aquest cas, ens trobem amb una funció que no retorna res i que té tres paràmetres d'entrada, dos dels quals es passen per referència i l'altre directament. Observant la pila, veiem que tot és correcte, és a dir, en el moment de la crida tenim les adreces dels dos paràmetres passats per referència i el valor del paràmetre c. A més, l'adreça de retorn a la funció que ha cridat funcio1 ja s'ha apilat.

Com que haurem de poder accedir als paràmetres que ens han passat mitjançant la pila i l'SP no ens serveix (no el podem usar per a adreçaments indirectes), copiem el contingut de SP al registre BP. Això és correcte. No obstant això, podria ser que la funció que ens ha cridat necessiti el contingut d'aquest registre per a continuar la seva execució i, per tant, convé apilar-lo abans de fer-ne la còpia.

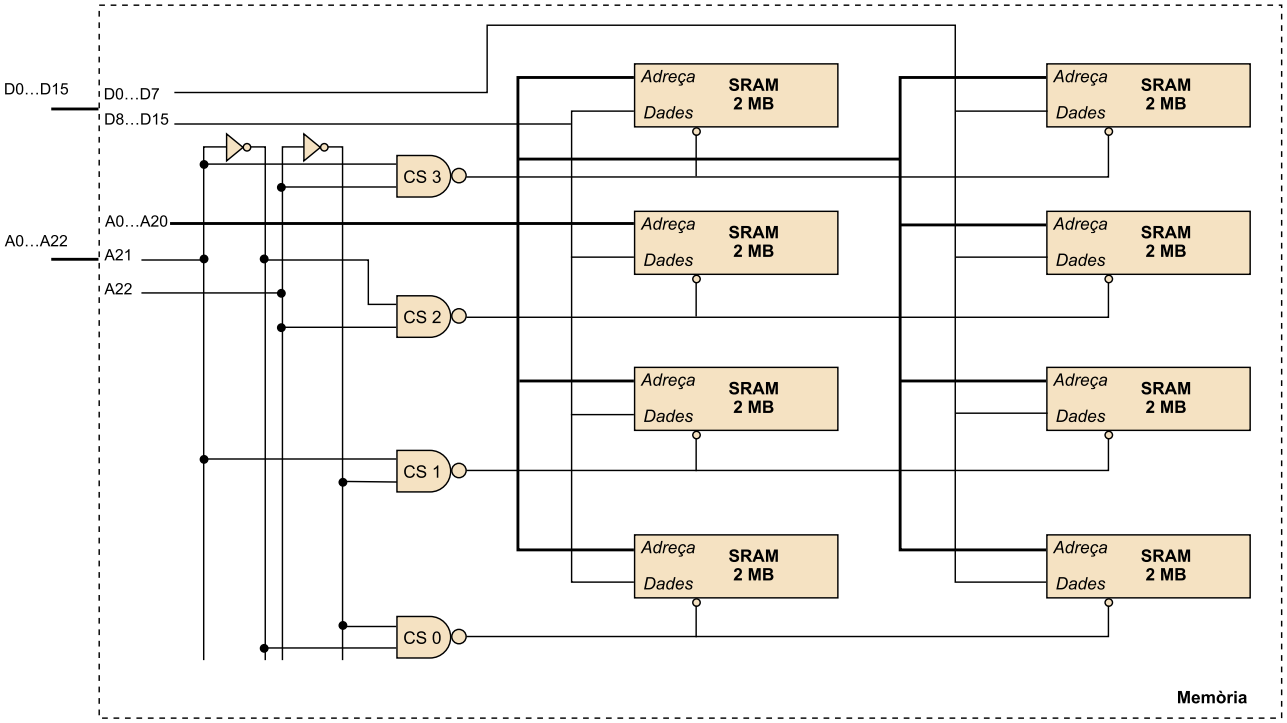
Després, apilem els registres AX, BX i CX. Això vol dir que els usarem durant el cos de la funció. Abans d'acabar la funció, per tant, s'han de restaurar. Aquí hi ha un primer error, ja que els hem de restaurar amb l'ordre invers que els hem apilat. Tal com està, posaríem el valor anterior de AX a BX i el valor anterior de BX a AX. A més a més, en executar RET retornaríem a la posició que indica CX i no a la que toca. Per tant, falta restaurar CX. Finalment, tenint en compte que hem apilat BP, també el restaurarem. El codi correcte és:

```
funcio1:    POP BP
            MOV BP, SP
            PUSH CX
            PUSH BX
            PUSH AX

            ... (Cos de la funció) ...

            POP AX
            POP BX
            POP CX
            POP BP
            RET
```

Fixem-nos, a més, que dins el cos de la funció accedirem a l'adreça de a fent [BP + 4], a l'adreça de *b* fent [BP + 6] i al paràmetre c fent [BP + 8].

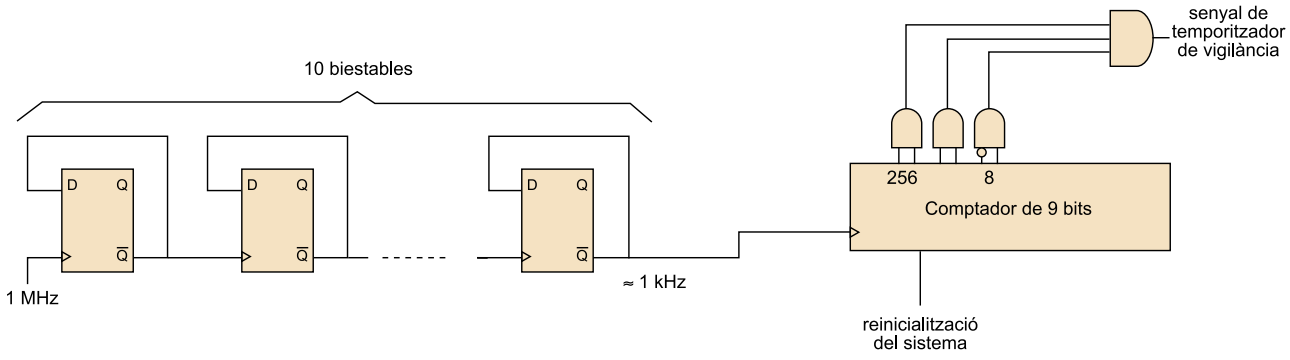
2. En primer lloc, hem de saber com està organitzada la memòria que tenim disponible. Si no ens diuen res més, suposarem que són memòries de 2 milions de paraules de 8 bits cadascuna. Per tant, agrupant quatre blocs de memòria tindrem 2 milions paraules de 32 bits. Finalment, si volem aconseguir els 8 milions de paraules de memòria, haurem de repetir aquesta estructura quatre cops. Finalment, en funció de l'adreça, haurem de distingir a quina d'aquestes estructures guardem la paraula en qüestió. Per a fer-ho, descodificarem els dos bits de més pes de l'adreça, la qual cosa ens permetrà activar només una de les estructures en cada cicle de memòria. El disseny resultant es pot veure en la figura següent:

Disseny resultant de l'activitat 2



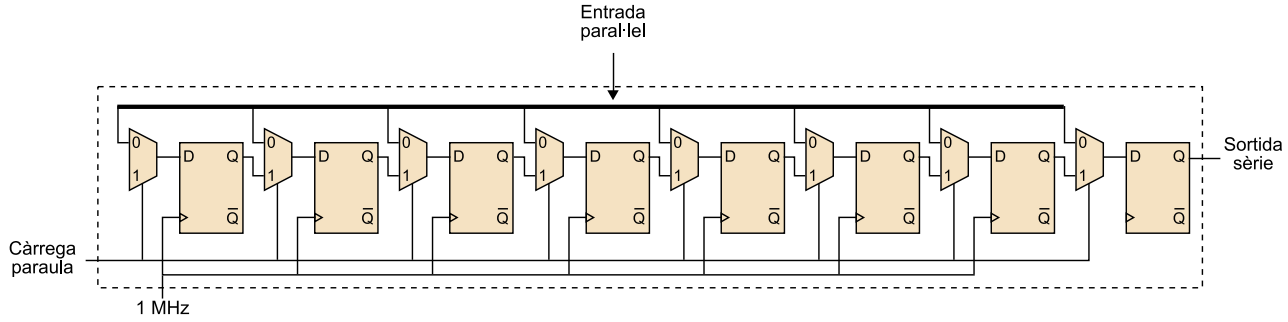
3. El primer que hem de fer és veure quina és la resolució temporal del rellotge del sistema. En aquest cas, 1 MHz correspon a un milió de cicles de rellotge per segon. Per tant, si comptem mig milió de cicles de rellotge aconseguim el nostre objectiu. Per a fer-ho, necessitaríem un comptador de 19 bits que no tenim. La solució és implementar un divisor de freqüència utilitzant biestables. En aquest cas, podem construir un comptador asíncron, ja que l'usarem exclusivament com a rellotge. Així, doncs, si utilitzem 10 biestables, dividirem la freqüència per 210 = 1024, amb la qual cosa obtindrem un rellotge de 976.5625 kHz, en què un període són 1,024 ms. Per a comptar 500 ms amb aquesta nova base temporal, necessitem aproximadament $500\,\text{ms} \,/\, 1{,}024\,\text{ms} \approx 488$ cops de rellotge. Per acabar, descodifiquem el nombre 488 (111101000 en binari) a la sortida del comptador, que serà la sortida del temporitzador de vigilància. De fet, podem observar que en aquest cas, descodificant els sis bits de més pes n'hi haurà prou per a complir el nostre objectiu. El disseny resultant es pot veure en la figura següent:

Disseny resultant de l'activitat 3



4. En aquest cas, hem de veure que el flux de dades en paral·lel de 125 K paraules per segon correspon a 1 MBps en una connexió sèrie. Ara ens falta convertir les dades de paral·lel a sèrie, per la qual cosa utilitzem un registre com el de la figura següent. Cal notar que, en aquest cas, el rellotge del sistema d'1 MHz ja és el que necessitem per a alimentar els biestables.

Disseny resultant de l'activitat 4



5. El primer pas per a resoldre el problema és calcular quina es la velocitat requerida per a treballar amb el senyal de vídeo generat per la videocàmera. La videocàmera opera amb imatges de 640 × 490 píxels de 24 bits. Per tant, cada fotograma té un nombre de bits igual a:

$$640\ x\ 490\ x\ 24\ =\ 7\,372\,800 \text{ bits}$$

Per a obtenir la velocitat requerida, s'ha de tenir en compte que la videocàmera treballa a 30 FPS:

$$v_{req} = 7\,372\,800 \text{ bits/fotogrames } x \text{ 30 fotogrames/s} = 221.18 \text{ Mbits/s}$$

És a dir, s'ha de proveir el sistema d'un mecanisme de comunicació capaç d'assegurar velocitats de 221,18 MBps. Aquest sistema, a la vegada, ha de ser de baix cost. Repassant els mecanismes de comunicació discutits en el mòdul, aquestes velocitats es poden assegurar només amb mecanismes cablejats. D'entre els disponibles, els que asseguren una velocitat d'aquesta magnitud amb un cost reduït, són:

- USB 2.0: 480 MBps amb 5 m de longitud de cable.
- Firewire 400: 400 MBps amb 4,5 m de longitud de cable.

Al mateix temps s'observa que ambdues tecnologies compleixen el requisit de poder establir una connexió a 4 m de distància.

Finalment, per acabar de decidir quina solució fer servir, s'ha de tenir en compte el requisit del sistema quant a assegurar un flux constant i un treball mínim per part del microprocessador del sistema encastat. En aquest sentit, Firewire 400 és la millor opció, ja que és una tecnologia purament d'igual a igual, capaç d'assegurar velocitats sostingudes més grans que en el cas d'USB, i on no es genera càrrega de treball addicional al microprocessador.

Tot i que USB és una solució més extensa comercialment, és en situacions particulars com aquesta (en la qual es demana el disseny d'una solució encastada amb unes condicions molt particulars), on l'ús de Firewire resulta l'opció més adient.

6. En aquest problema, es demana comprovar si, amb la configuració proposada, el receptor pot rebre potències superiors a –82 dBm. Com que es comenta que l'escenari es pot modelar amb la fórmula de Friis fent servir un valor de γ igual a 3, s'haurà de calcular la potència rebuda amb la fórmula següent:

$$P_r = P_t\left(\frac{c}{4\pi f}\right)^2 \frac{1}{d^3}$$

i, un cop fet això, comprovar si $P_r > P_s$, en què P_s és la sensibilitat mencionada anteriorment (expressada en W, en aquest cas). Com que aquesta fórmula treballa en lineal, és a dir, les potències s'expressen en W, i el valor de potència transmesa que ofereix l'enunciat del problema està expressat en dBm (tal com succeeix a la pràctica), el primer que s'ha de fer és passar el valor de potència en dBm al seu equivalent lineal en W. El terme dBm, per la seva banda, es refereix al nivell de potència que es té en escala logarítmica quan aquesta potència es referencia a 1 mW. Per tant, abans de passar-ho a lineal s'haurà de fer la transformació de dBm a dBW, ja que aquest darrer terme és l'equivalent referenciat a 1 W i el seu ús simplifica el procés de transformació a lineal. Per a fer això, s'ha de tenir en compte que per a expressar una potència, P, en dBm i en dBW, les fórmules matemàtiques utilitzades són:

$$P(\text{dBm}) = 10log_{10}(P(\text{W})/1\text{mW}) = 10log_{10}(P(\text{W})/1{\cdot}10^{-3}\text{W})$$
$$P(\text{dBW}) = 10log_{10}(P(\text{W})/1\text{W})$$

de les quals es pot veure fàcilment que:

$$P(\text{dBm}) = 10log_{10}(P(\text{W})/1{\cdot}10^{-3}\text{W}) = 10log_{10}(P(\text{W})/1W) - 10log_{10}(1/10^{-3})$$
$$= P(\text{dBW}) + 30\text{dB}$$

Aplicant aquestes igualtats, quedarà la transformació següent:

$$P_t(\text{dBm}) = 0\ \text{dBm} \Rightarrow P_t(\text{dBW}) = 0\ \text{dBm}\ -\ 30\ \text{dB}\ =\ -30\ \text{dBW}$$

i amb aquest valor, es pot fer ja el pas de dBW a W de la manera següent:

$$P_t(\text{W}) = 10^{P_t(\text{dBW})/10} = 10^{-30/10}\ =\ 1{\cdot}10^{-3}\ W$$

Quant a la resta de paràmetres de la fórmula, c es refereix a la velocitat de la llum, $c = 3 \cdot 10^8$ m/s; f és la freqüència de treball de la tecnologia ràdio, $f = 868 \cdot 10^6$ Hz, i $d = 60$ m tal com menciona l'enunciat del problema. Utilitzant aquests valors tenim que:

$$P_r = P_t\left(\frac{c}{4\pi f}\right)^2\frac{1}{d^3} = 1{\cdot}10^{-3}\left(\frac{3{\cdot}10^8}{4\pi 868{\cdot}10^6}\right)^2\frac{1}{60^3} = 3.5{\cdot}10^{-12}\text{W}$$

Per comparar aquesta potència rebuda amb la sensibilitat, es converteix el resultat anterior a dBm, ja que l'ús de l'escala logarítmica ofereix una manera més còmoda de treballar quan s'opera amb aquestes magnituds de potència. Fent la transformació s'obté que:

$$P_r(\text{dBm}) = 10log_{10}\left(3.5{\cdot}10^{-12}\text{W}\right) + 30\text{dB} = -84.55\text{dBm}$$

Tal com es pot veure, la potència rebuda, P_r(dBm) = –84.55 dBm, és inferior a la sensibilitat del receptor, igual a P_s(dBm) = –82 dBm. Per tant, l'ús d'aquesta configuració no és adient per a l'escenari del problema.

7. Tal com s'ha vist en la solució del problema anterior, la configuració proposada basada en IEEE 802.15.4 no és adient, ja que la potència rebuda a 60 m no és més gran que la sensibilitat del receptor. Com que l'enunciat diu que l'única configuració disponible per a IEEE 802.15.4 és aquesta, cal buscar-ne una alternativa tecnològica. Tenint en compte que es busca una solució de baix cost i consum, el candidat següent és Bluetooth. Tal com s'ha vist en teoria, hi pot haver dispositius Bluetooth de tres classes segons la potència de transmissió:

- Classe 1: 100 mW de potència.
- Classe 2: 2,5 mW de potència.
- Classe 3: 1 mW de potència.

Atès el fet que més potència implica més consum, interessa escollir un dispositiu amb el mínim nivell de potència possible. Entre les tres classes possibles, el dispositiu de classe 3 queda directament descartat, ja que la configuració base, basada en IEEE 802.15.4, també operava a 1 mW. A més, en el cas de Bluetooth, la potència rebuda encara serà més baixa perquè s'opera a la banda de 2.400 MHz. Per tant, l'opció següent que cal verificar és un dispositiu de la classe 2. Com en el problema anterior, es fa servir la fórmula de Friis, però en aquest cas s'ha de tenir en compte que $P_t = 2{,}5 \cdot 10^{-3}$ i $f = 2.400 \cdot 10^6$ Hz:

$$P_r = P_t\left(\frac{c}{4\pi f}\right)^2\frac{1}{d^3} = 2.5{\cdot}10^{-3}\left(\frac{3{\cdot}10^8}{4\pi 2400{\cdot}10^6}\right)^2\frac{1}{60^3} = 1.14{\cdot}10^{-12}\text{W}$$

si es passa a dBm:

$$P_r(\text{dBm}) = 10log_{10}\left(1.14{\cdot}10^{-12}\text{W}\right) + 30\text{dB} = -89.43\ \text{dBm}$$

Tal com es pot observar, el valor de potència no arriba a superar la sensibilitat. De fet, la potència rebuda en aquest cas és inferior que en el cas del dispositiu IEEE 802.15.4. Tot i que la potència de transmissió és més gran, la freqüència de treball també ho és i per aquest motiu la potència de senyal decau de manera més pronunciada amb la distància.

Si s'analitza ara el cas de classe 1, es veu que la potència rebuda és:

$$P_r = P_t\left(\frac{c}{4\pi f}\right)^2 \frac{1}{d^3} = 100\cdot 10^{-3}\left(\frac{3\cdot 10^8}{4\pi 2400\cdot 10^6}\right)^2 \frac{1}{60^3} = 45.8\cdot 10^{-12}\text{W}$$

que en transformar-ho a dBm dóna:

$$P_r(\text{dBm}) = 10log_{10}\left(45.8\cdot 10^{-12}\text{W}\right) + 30\text{dB} = -73.39\ \text{dBm}$$

I, per tant, aquesta sí és una alternativa adient per a l'escenari del problema, ja que el senyal rebut és per sobre de la sensibilitat del receptor.

8. Per a dissenyar el subsistema d'alimentació, el primer que s'ha de tenir en compte és que el sistema encastat de l'enunciat és alimentat per la xarxa elèctrica. Per tant, s'ha de fer servir un convertidor ca i a la sortida del mateix es posarà un pont de díodes per a assegurar que el sistema serà robust a canvis de polaritat.

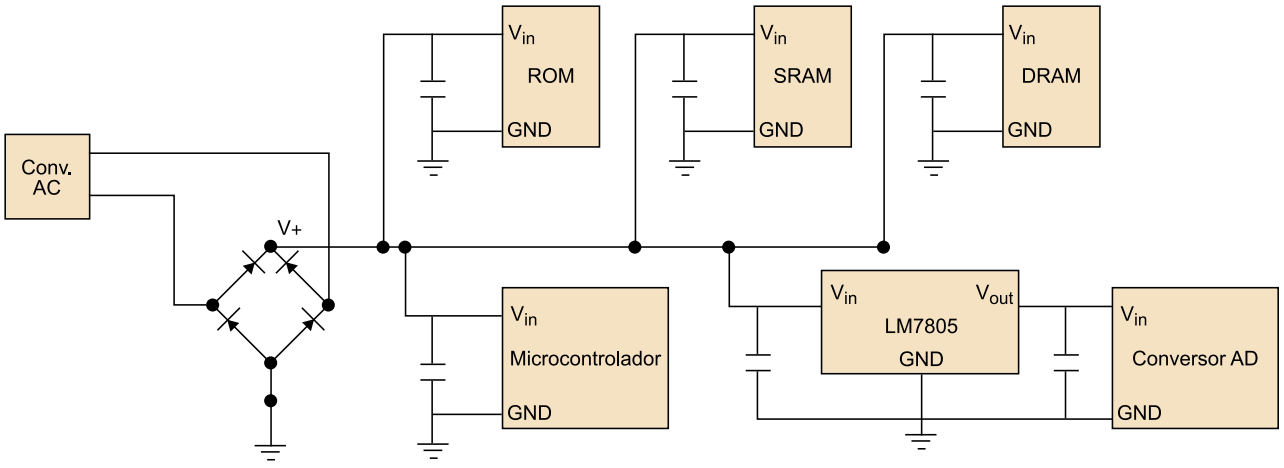
Fet això, el pas següent és connectar el microcontrolador a la línia d'alimentació. Per tal de fer el sistema robust a soroll electromagnètic, s'inclourà un condensador desacoblador a la línia. D'altra banda, es farà servir un pla de massa per a connectar totes les masses del circuit i d'aquesta manera es reduirà el bucle de corrent.

Per a alimentar les diferents memòries es farà servir el mateix procediment que l'utilitzat en el cas del microcontrolador.

En el cas del convertidor analògic-digital, es farà servir un regulador per a assegurar que el component s'alimenta amb una tensió a 5 V. Per aquest motiu, es pot fer servir el regulador comercial LM7805. D'altra banda, per a evitar el soroll electromagnètic, s'utilitzarà l'esquema basat en el regulador lineal amb dos condensadors desacobladors.

En la figura següent es mostra l'esquema del disseny resultant, en què les entrades V_{in} i GND es refereixen a les entrades d'alimentació i massa, respectivament. Observeu que només s'han dibuixat les línies d'alimentació del sistema, ja que el problema es referia només al disseny del subsistema d'alimentació.

Disseny resultant de l'activitat 8



Connexió al pla de massa

Exercicis d'autoavaluació

1. d

2. a

3. a

4. d

5. b

6. d

7. c

8. b

9. a

10. c

11. c

12. c

13. d

14. c

15. c

16. a

17. b

18. c

19. b

20. d

21. c

22. a

23. b

24. c

Bibliografia

Alcubilla, R.; Pons, J.; Bardés, D. (2004). *Diseño digital. Una perspectiva VLSI-CMOS*. Barcelona: UPC.

Catsoulis, J. (2005). *Designing Embedded Hardware*. Sebastopol, Califòrnia: O'Reilly and Associates.

Peckol, J. K. (2008). *Embedded Systems: A Contemporary Design Tool*. Hoboken, Nova Jersey: Wiley.

Yaghmour, K. (2003). *Building Embedded Linux Systems*. Sebastopol, Califòrnia: O'Reilly and Associates.