Els circuits lògics combinacionals

Montse Peiron Guàrdia
Fermín Sánchez Carracedo

PID_00215616

Universitat Oberta de Catalunya

Índex

Introducció

Un computador és una màquina construïda a partir de dispositius electrònics bàsics, interconnectats adequadament. Podem dir que aquests dispositius són les "peces" o "maons" amb els quals es construeix un computador.

En aquest mòdul coneixerem a fons els diferents components que constitueixen els circuits. Els dispositius electrònics més elementals són les **portes lògiques** i els **blocs lògics**, que formen els **circuits lògics**. Un circuit lògic es pot veure com un conjunt de dispositius que manipulen d'una manera determinada els senyals electrònics que els arriben (els **senyals d'entrada**), i generen com a resultat un altre conjunt de senyals (els **senyals de sortida**).

Hi ha dos grans tipus de circuits lògics:

1) Els **circuits combinacionals**, que es caracteritzen perquè el valor dels senyals de sortida en un moment determinat depèn del valor dels senyals d'entrada en aquest mateix moment.

2) Els **circuits seqüencials**, en els quals el valor dels senyals de sortida en un moment determinat depèn dels valors que han arribat pels senyals d'entrada des de la posada en funcionament del circuit (tenen, per tant, capacitat de memòria).

En aquest mòdul s'estudien els circuits lògics combinacionals. Els circuits seqüencials s'estudiaran en el següent mòdul "Els circuits lògics seqüencials".

Objectius

L'objectiu fonamental d'aquest mòdul és conèixer a fons els circuits lògics combinacionals, és a dir, saber com estan formats i ser capaços d'utilitzar-los amb agilitat, fins al punt d'estar-hi totalment familiaritzats.

Per a arribar a aquest punt caldrà haver assolit els objectius següents:

1. Entendre l'àlgebra de Boole i les diferents maneres d'expressar funcions lògiques.

2. Conèixer les diferents portes lògiques, veure com es poden utilitzar per a sintetitzar funcions lògiques i ser capaços de fer-ho. Entendre per què és desitjable de minimitzar el nombre de portes i de nivells de portes dels circuits, i saber-ho fer.

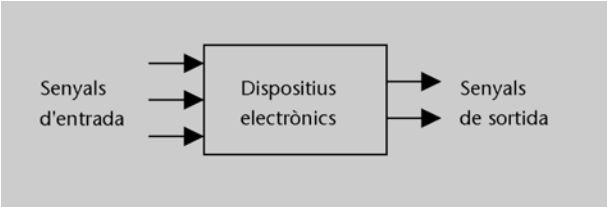
3. Conèixer la funcionalitat d'un conjunt de blocs combinacionals bàsics, i ser capaços d'utilitzar-los en el disseny de circuits.

En definitiva, després de l'estudi d'aquest mòdul hem de ser capaços de construir fàcilment un circuit qualsevol usant els diferents dispositius que s'hauran conegut, i també d'entendre la funcionalitat de qualsevol circuit donat.

1. Fonaments de l'electrònica digital

1.1. Circuits, senyals i funcions lògiques

Entenem per **circuit** un sistema format per un cert nombre de **senyals d'entrada** (cada senyal correspon a un cable), un conjunt de **dispositius electrònics** que fan operacions sobre els senyals d'entrada (els manipulen electrònicament), i que generen un cert nombre de **senyals de sortida**. Els senyals de sortida, doncs, es poden veure com a funcions dels senyals d'entrada, i es pot dir que els dispositius electrònics *computen* aquestes fun-cions.



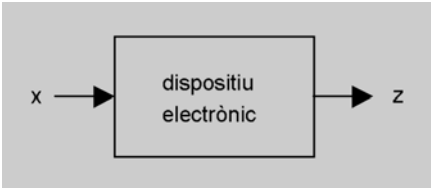
En els circuits, els cables es poden trobar en dos valors de tensió (voltatge): tensió alta o tensió baixa. Aquests dos valors de tensió s'identifiquen normalment pels símbols 1 i 0, respectivament, de manera que es diu que un senyal *val 0* (quan en el cable corresponent hi ha tensió baixa) o *val 1* (quan en el cable hi ha tensió alta, també anomenada *tensió d'alimentació*); quan un senyal val 1 es diu que *està actiu*. Els senyals que poden prendre els valors 0 o 1 s'anomenen **senyals lògics** o **binaris**. Un **circuit lògic** és aquell en el qual els senyals d'entrada i de sortida són lògics. Les funcions que computa un circuit lògic són **funcions lògiques**.

La tensió alta o tensió d'alimentació base pot ser de 3,3, 5 o 12 volts, però actualment els circuits tenen dispositius per a modificar-la segons els requisits de cada component i de cada moment; per exemple, quan el processador està en una fase d'activitat alta el voltatge augmenta. La tensió baixa sempre es de 0 volts.

Hi ha circuits que funcionen amb *lògica inversa*, i en aquest cas es diu que un senyal està actiu quan val 0. En aquest curs, però, quan diem que un senyal està actiu volem dir que val 1.

Com que els senyals només poden prendre dos valors, direm que l'un és el contrari de l'altre. Així, doncs, podem afirmar que quan un senyal no val 1, llavors segur que val 0, i viceversa.

Prenem un circuit que tingui només un senyal d'entrada (que anomenem *x*), un dispositiu electrònic i un senyal de sortida (que anomenem *z*).



Atès que tant *x* com *z* només poden valer 0 o 1, només hi ha quatre dispositius electrònics diferents que puguin interconnectar *x* i *z*:

- Un dispositiu que faci que la sortida *z* valgui sempre 0 (aquest dispositiu consistiria a connectar la sortida a una font de tensió de 0 volts).
- Un dispositiu que faci que la sortida valgui sempre el mateix que l'entrada *x* (aquest dispositiu consistiria a connectar directament la sortida amb l'entrada).
- Un dispositiu que faci que la sortida valgui sempre el contrari del que val l'entrada (aquest dispositiu consistiria en un inversor del nivell de tensió).
- Un dispositiu que faci que la sortida valgui sempre 1 (aquest dispositiu consistiria a connectar la sortida directament a la tensió d'alimentació).

En altres paraules, podem dir que només hi ha quatre funcions lògiques que tinguin una sola variable d'entrada, tal com es mostra en la figura 1:

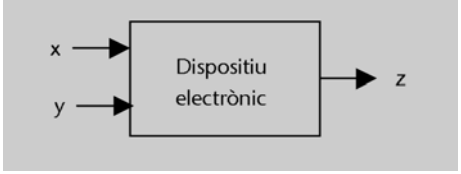
Figura 1

Funció lògica	Descripció	Valor de la funció quan x = 0	Valor de la funció quan x = 1
$z = f_0(x) = 0$	funció constant 0	0	0
$z = f_1(x) = x$	funció identitat	0	1
$z = f_2(x) = x'$	funció contrari	1	0
$z = f_3(x) = 1$	funció constant 1	1	1

Nota

Introduïm aquí la nomenclatura *x'* per a referir-nos a la funció "contrari de *x*". Més endavant, la definirem amb més propietat.

Prenem ara un circuit que tingui dos senyals d'entrada (que anomenem *x* i *y*), un dispositiu electrònic i un senyal de sortida (que anomenem *z*).



En aquest cas, hi ha setze dispositius electrònics diferents que puguin interconnectar les entrades amb la sortida, que corresponen a les funcions lògiques que es mostren en la figura 2.

Figura 2

Funció lògica	Valor de la funció quan			
	x = 0, y = 0	x = 0, y = 1	x = 1, y = 0	x = 1, y = 1
$z = g_0(x,y)$	0	0	0	0
$z = g_1(x,y)$	0	0	0	1
$z = g_2(x,y)$	0	0	1	0
$z = g_3(x,y)$	0	0	1	1
$z = g_4(x,y)$	0	1	0	0
$z = g_5(x,y)$	0	1	0	1
$z = g_6(x,y)$	0	1	1	0
$z = g_7(x,y)$	0	1	1	1
$z = g_8(x,y)$	1	0	0	0
$z = g_9(x,y)$	1	0	0	1
$z = g_{10}(x,y)$	1	0	1	0
$z = g_{11}(x,y)$	1	0	1	1
$z = g_{12}(x,y)$	1	1	0	0
$z = g_{13}(x,y)$	1	1	0	1
$z = g_{14}(x,y)$	1	1	1	0
$z = g_{15}(x,y)$	1	1	1	1

Així, doncs, si tenim en compte els circuits amb una o dues entrades, podem arribar a dissenyar fins a 4 + 16 = 20 dispositius electrònics diferents. Ara bé, a la pràctica només se'n construeixen un subconjunt, concretament els que corresponen a les funcions f_2, g_1 i g_7 (i algunes altres que veurem més endavant), perquè a partir d'aquestes funcions es poden construir totes les altres, tal com veurem en l'apartat 1.2.

A continuació, veurem que hi ha una correspondència entre els elements d'un circuit lògic i la lògica intuïtiva (d'aquí deriva la denominació de circuits "lògics"). La lògica té cinc components bàsics:

– Els valors *fals* i *cert*.
– Les conjuncions *i* i *o*.
– La partícula de negació *no*.

Per exemple, siguin les dues frases següents:

frase_A: "en Joan estudia química",
frase_B: "la Carme estudia piano".

Cadascuna d'aquestes frases pot ser certa o falsa. A partir d'aquestes i de les conjuncions i la negació, construïm ara les tres frases següents:

frase_I: "en Joan estudia química **i** la Carme estudia piano",
frase_O: "en Joan estudia química **o** la Carme estudia piano",
frase_NO: "en Joan **no** estudia química".

Segons la lògica:

Nota

Normalment, quan utilitzem la conjunció *o* en llenguatge natural ho fem de manera exclusiva. És a dir, si diem "X o Y" ens referim al fet que o bé és cert X o bé és cert Y, però no totes dues coses alhora. La *o* lògica, en canvi, inclou també la possibilitat que les dues afirmacions siguin certes.

– La frase_I és certa només si són certes la frase_A i la frase_B, simultàniament.

– La frase_O és certa si ho és la frase_A o bé la frase_B, o bé si ho són totes dues.

– La frase_NO és certa només si la frase_A és falsa.

La correspondència de la lògica amb els elements d'un circuit lògic és la següent:

Lògica	Circuits lògics
fals	0
cert	1
conjunció *i*	funció g_1
conjunció *o*	funció g_7
partícula *no*	funció f_2

En efecte, siguin dos senyals lògics x i y. Si analitzem les taules de les figures 1 i 2, podem veure el següent:

– $g_1(x,y)$ val 1 només si valen 1 totes dues variables x i y simultàniament,
– $g_7(x,y)$ val 1 si x val 1 o bé y val 1 o bé valen 1 totes dues,
– $f_2(x)$ val 1 només si x val 0.

És a dir, es compleix el mateix que en l'exemple de les frases. Per això, la funció g_1 s'anomena AND (la conjunció *i* en anglès), la funció g_7 s'anomena OR (la conjunció *o* en anglès) i la funció f_2 s'anomena NOT (*no* en anglès).

Així, doncs, els circuits lògics es construeixen a partir del mateix fonament que la lògica. En l'apartat següent s'estudia aquest fonament.

1.2. Àlgebra de Boole

Una **àlgebra de Boole** és una entitat matemàtica formada per un conjunt que conté dos elements, unes operacions bàsiques sobre aquests elements i una llista d'axiomes que defineixen les propietats que compleixen les operacions.

George Boole

Va formalitzar matemàticament la lògica el 1854, quan va definir el que avui es coneix per *àlgebra de Boole*. De fet, aquí presentem un cas particular d'àlgebra de Boole, anomenat *àlgebra de Boole binària*; en general, el conjunt d'una àlgebra de Boole pot tenir més de dos elements.

El 1938, Claude Shannon va definir el comportament dels circuits lògics sobre el fonament de l'àlgebra de Boole, i va crear l'*àlgebra de commutació.*

Els dos **elements** d'una àlgebra de Boole es poden anomenar *fals* i *cert* o, més usualment, 0 i 1. Així, una variable booleana o **variable lògica** pot prendre els valors 0 i 1.

Les **operacions booleanes bàsiques** són les següents:

- La **negació** o complementació o NOT, que correspon a la partícula no i es representa amb una cometa simple (’). Així, l’expressió x’ denota la negació de la variable x, i es llegeix “*no x*”.
- **El producte lògic** o **AND**, que correspon a la conjunció *i* de la lògica i es representa amb símbol “·”. Així, si x i y són variables lògiques, l’expressió $x \cdot y$ denota el seu producte lògic, i es llegeix “x i y”.
- La **suma lògica** o **OR**, que correspon a la conjunció *o* i es representa pel símbol “+”. Així, l’expressió “$x + y$” denota la suma lògica de les variables x i y, i es llegeix “x o y”.

L’operació de negació es pot representar també d’altres maneres. Per exemple, $!x$ o, més usualment, $\bar{x}$.

Aquestes operacions booleanes bàsiques es poden definir escrivint el resultat que donen per cada possible combinació de valors de les variables d’entrada, tal com es mostra en la figura 3. En les taules d’aquesta figura hi ha a l’esquerra de la ratlla vertical totes les combinacions possibles de les variables d’entrada, i a la dreta el resultat de l’operació per a cada combinació. Podem comprovar que corresponen a les funcions f_2, g_1 i g_7 que hem vist en l’apartat anterior (figures 1 i 2).

Atenció

És important no confondre els operadors · i + amb les operacions *producte enter i suma entera* a les quals estem acostumats. El significat dels símbols vindrà determinat pel context en el qual ens trobem. Així, si treballem amb enters, 1 + 1 = 2 (‘u més u igual a dos’), mentre que si estem en un context booleà, 1 + 1 = 1 (‘cert o cert igual a cert’, tal com es pot veure a la taula de l’operació OR).

Figura 3

x	x’
0	1
1	0

Operació NOT

x	y	$x \cdot y$
0	0	0
0	1	0
1	0	0
1	1	1

Operació AND

x	y	$x + y$
0	0	0
0	1	1
1	0	1
1	1	1

Operació OR

Els **axiomes** que descriuen el comportament de les operacions booleanes són els següents:

La formulació que es presenta aquí dels axiomes booleans va ser introduïda per Huntington el 1904, a partir de la descripció original de Boole.

Siguin x, y i z variables lògiques. Llavors es compleix el següent:

a) La **propietat commutativa:**

$$x + y = y + x$$
$$x \cdot y = y \cdot x$$

b) La **propietat associativa:**

$$x + (y + z) = (x + y) + z$$
$$x \cdot (y \cdot z) = (x \cdot y) \cdot z$$

c) La **propietat distributiva:**

$$x \cdot (y + z) = x \cdot y + x \cdot z$$
$$x + (y \cdot z) = (x + y) \cdot (x + z)$$

Observem que la segona igualtat de la propietat distributiva no es compleix en el context de l'aritmètica entera.

d) Els **elements neutres:**

$$x + 0 = x$$
$$x \cdot 1 = x$$

e) La **complementació**. Per a qualsevol x, es compleix el següent:

$$x + x' = 1$$
$$x \cdot x' = 0$$

A partir d'aquests axiomes, es poden demostrar una sèrie de lleis o teoremes molt útils per a treballar amb expressions algebraiques booleanes.

Teoremes de l'àlgebra de Boole

Si x, y i z són variables lògiques, es compleixen les lleis següents:

1) Principi de dualitat

Tota identitat deduïda a partir dels axiomes continua essent certa si les operacions + i · i els elements 0 i 1 s'intercanvien en tota l'expressió.

Aquesta llei es pot comprovar, per exemple, examinant les parelles d'expressions que apareixen en la definició dels axiomes.

2) Llei d'idempotència

$$x + x = x$$
$$x \cdot x = x$$

3) Llei d'absorció

$$x + x \cdot y = x$$
$$x \cdot (x + y) = x$$

4) Llei de dominància

$$x + 1 = 1$$
$$x \cdot 0 = 0$$

5) Llei d'involució

$$(x')' = x$$

6) Lleis de De Morgan

$$(x + y)' = x' \cdot y'$$
$$(x \cdot y)' = x' + y'$$

Un bon exercici per a veure que es compleixen aquestes lleis és pensar en la seva correspondència amb la lògica, tal com es fa en l'última part de l'apartat 1.1.

En les expressions algebraiques, utilitzem els parèntesis de la mateixa manera que hi estem acostumats a fer-ho en aritmètica entera; per exemple, l'expressió $x + y \cdot z$ és el mateix que la $x + (y \cdot z)$ i és diferent que $(x + y) \cdot z$.

Per a negar una expressió sencera la posarem entre parèntesis, i, per tant, $x + y'$ és diferent de $(x + y)'$.

Activitats

1. Avalueu el valor de l'expressió $x + y \cdot (z + x')$ per als casos:
$[x\ y\ z] = [0\ 1\ 0]$,
$[x\ y\ z] = [1\ 1\ 0]$,
$[x\ y\ z] = [0\ 1\ 1]$.

2. Demostreu les lleis de De Morgan d'acord amb tots els possibles valors que poden prendre les variables que hi apareixen.

3. Demostreu la llei 4 de l'àlgebra de Boole a partir de la llei 2 i de l'axioma *e*.

1.3. Representació de funcions lògiques

Les funcions lògiques es poden expressar de diverses maneres. Les que usarem nosaltres són les **expressions algebraiques** i les **taules de veritat**.

1.3.1. Expressions algebraiques

Les **expressions algebraiques** estan formades per variables lògiques, els elements 0 i 1, els operadors producte (·), suma (+) i negació ('), i els símbols (,) i =.

Mitjançant aquests elements es pot expressar qualsevol funció lògica. Per exemple, la funció g_4 de la figura 2 es pot expressar així:

$$g_4(x, y) = x' \cdot y.$$

L'expressió $x' \cdot y$ val 1 (és certa) només si valen 1 (són certes) les subexpressions x' i y simultàniament. L'expressió x' val 1 només si x val 0. En la descripció de la funció g_4 (figura 2) es pot comprovar que només val 1 en el cas en què $x = 0$ i $y = 1$.

Una mateixa funció lògica es pot expressar mitjançant infinites expressions algebraiques equivalents.

Per saber si dues expressions algebraiques són equivalents (és a dir, expressen la mateixa funció) podem analitzar si una es pot derivar de l'altra usant els axiomes i lleis de l'àlgebra de Boole.

Per exemple, la primera llei de De Morgan ens diu que les funcions lògiques f i g són la mateixa:

$$f(x, y) = (x + y)'; g(x, y) = x' \cdot y'.$$

A la inversa, a partir de l'expressió algebraica d'una funció podem trobar altres expressions equivalents aplicant-li els axiomes i lleis de l'àlgebra de Boole.

Per exemple, si tenim la funció següent:

$$f(x, y, z) = x \cdot y' \cdot z + x' \cdot y \cdot z + x \cdot y \cdot z,$$

i apliquem l'axioma c (propietat distributiva), obtenim l'expressió equivalent següent:

$$f(x, y, z) = x \cdot y' \cdot z + (x' + x) \cdot y \cdot z.$$

Per l'axioma e (de complementació) i després el d (d'elements neutres), obtenim el que expressem a continuació:

$$f(x, y, z) = x \cdot y' \cdot z + 1 \cdot y \cdot z = x \cdot y' \cdot z + y \cdot z.$$

Per la propietat distributiva, $f(x, y, z) = (x \cdot y' + y) \cdot z$.

I, així, podríem seguir trobant infinites expressions equivalents per la mateixa funció.

Igual que fem habitualment quan treballem amb equacions, podem ometre el signe "·" en els productes lògics, per tal de simplificar l'escriptura de les expressions algebraiques: si escrivim seguits els noms de diverses variables, sobreentendrem que se'n fa el producte lògic.

Així, les dues expressions següents són igualment vàlides:

$$x_1' \cdot x_0 + x_2 \cdot x_1' \cdot x_0', \text{ o bé } x_1' x_0 + x_2 x_1' x_0'.$$

Activitats

4. Trobeu una expressió algebraica per a les funcions g_1, g_3, g_6, g_7 i g_{10} de la figura 2.
5. Trobeu una expressió més senzilla per aquesta funció:
$f = wx + xy' + yz + xz' + xy$.
6. Sigui una funció de tres variables x, y i z. Trobeu-ne una expressió algebraica suposant que la funció ha de valer 1 quan es compleixi alguna de les condicions següents:
– $x = 1$ o $y = 0$,
– $x = 0$ i $z = 1$,
– totes tres variables valen 1.
7. Es disposa de dues caixes fortes electròniques, A i B. Cadascuna de les caixes té un senyal associat, x_A i x_B respectivament, que val 1 quan la caixa és oberta i 0 quan és tancada. Es té també un interruptor general que té un senyal associat ig, que val 0 si l'interruptor està tancat i 1 si està obert.
Es vol construir un sistema d'alarma contra robatoris, que generarà un senyal de sortida s. Aquest senyal ha de valer 1 quan alguna caixa forta estigui oberta i l'interruptor estigui tancat.
Doneu l'expressió algebraica de la funció $s = f(x_A, x_B, ig)$.
8. En Joan s'ha examinat de tres assignatures. Els seus amics han vist els resultat dels exàmens i li han comentat el següent:
– Has aprovat matemàtiques o física, diu el primer.
– Has suspès química o matemàtiques, diu el segon.
– Has aprovat només dues assignatures, diu el tercer.

Entenem que la "o" d'aquestes frases és exclusiva. És a dir, la primera frase es podria substituir per "o bé has aprovat matemàtiques i has suspès física, o bé has suspès matemàtiques i has aprovat física", i de manera similar amb la segona frase.

a) Escriviu les expressions algebraiques de les afirmacions de cadascun dels amics.
b) Utilitzant els axiomes i teoremes de l'àlgebra de Boole, deduïu quines assignatures ha aprovat en Joan i quina ha suspès.

1.3.2. Taules de veritat

Una **taula de veritat** expressa una funció lògica especificant el valor que té la funció per a cada possible combinació de valors de les variables d'entrada.

Taula de veritat

En aritmètica entera és impossible d'escriure la taula de veritat d'una funció, ja que les variables d'entrada poden prendre infinits valors possibles, i, per tant, la llista hauria de ser infinita. En àlgebra de Boole és possible gràcies al fet que les variables poden prendre només dos valors.

Concretament, a l'esquerra de la taula hi ha una llista amb totes les combinacions de valors possibles de les variables d'entrada, i a la dreta el valor de la funció per a cadascuna de les combinacions. Per exemple, les taules que havíem vist en la figura 3 eren, de fet, les taules de veritat de les funcions NOT, AND i OR.

Si una funció té n variables d'entrada, i atès que una variable pot prendre només dos valors, 0 o 1, les entrades poden prendre 2^n combinacions de valors diferents. Per tant, la seva taula de veritat tindrà a l'esquerra n columnes (una per a cada variable) i 2^n files (una per a cada combinació possible). A la dreta, hi haurà una columna amb els valors de la funció.

Les files les escriurem sempre en *ordre lexicogràfic*: és a dir, si interpretem les diferents combinacions com a nombres naturals, escriurem primer la combinació corresponent al 0 (formada per només zeros), després la corresponent a l'1 i successivament en ordre creixent fins a la corresponent al $2^n - 1$ (formada per només uns).

Nota

No obstant això, les combinacions es podrien escriure en qualsevol ordre. Per exemple, les dues taules següents expressen la mateixa funció:

x	y	f
0	0	0
0	1	1
1	0	1
1	1	0

x	y	f
0	1	1
1	1	0
1	0	1
0	0	0

La figura 4 mostra l'estructura de les taules de veritat per a funcions d'una, dues i tres variables d'entrada.

Figura 4

Estructura de la taula de veritat d'una funció de

1 variable

a	f
0	
1	

2 variables

a	b	f
0	0	
0	1	
1	0	
1	1	

3 variables

a	b	c	f
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

Per exemple, la funció g_5 de la figura 2 té la taula de veritat següent:

x	y	g_5
0	0	0
0	1	1
1	0	0
1	1	1

És important de notar que l'ordre en què posem les columnes de les variables és significatiu. Per exemple, podíem haver escrit la taula de veritat de la funció g_5 de la manera següent:

y	x	g_5
0	0	0
0	1	0
1	0	1
1	1	1

És força usual anomenar les variables d'una funció amb una mateixa lletra i diferents subíndexs; per exemple, x_2, x_1, x_0. Per convenció, posarem en la columna de més a l'esquerra la variable de subíndex més alt, i a la de més a la dreta la de subíndex més baix.

Direm que la variable que posem a la columna de més a l'esquerra en una taula de veritat és la variable **de més pes**, i la que posem en la columna de més a la dreta és la **de menys pes**. Una vegada fixat l'ordre de les columnes i les files, la taula de veritat d'una funció és **única**.

Podem expressar el comportament de diverses funcions que tenen les mateixes variables d'entrada en una única taula de veritat. En aquest cas, a la dreta de la línia vertical hi haurà tantes columnes com funcions. No establirem cap convenció per a ordenar les columnes corresponents a les funcions (es poden posar en qualsevol ordre). A continuació, es mostra un exemple.

x_2	x_1	x_0	f_0	f_1	f_2
0	0	0	1	0	0
0	0	1	1	0	1
0	1	0	0	1	1
0	1	1	0	0	0
1	0	0	0	1	1
1	0	1	1	0	0
1	1	0	0	0	1
1	1	1	0	1	1

Activitats

9. Escriviu la taula de veritat de les funcions g_1, g_3, g_6, g_7 i g_{10} de la figura 2.
10. Dibuixeu l'estructura de la taula de veritat de la funció $f(x_3, x_2, x_1, x_0)$.
11. Especifiqueu la taula de veritat d'una funció de quatre variables, x_3, x_2, x_1 i x_0, que val 1 només quan un nombre parell de les variables valen 1 (recordeu que el 0 és un número parell).

1.3.3. Correspondència entre expressions algebraiques i taules de veritat

És important de saber passar amb agilitat d'una expressió algebraica d'una funció a la seva taula de veritat, i viceversa.

Per a obtenir la taula de veritat a partir d'una expressió algebraica, podem fer-ho de dues maneres:

a) Avaluar l'expressió per a cada combinació de les variables, i escriure al lloc corresponent de la taula el resultat de l'avaluació.

Per exemple, amb la funció següent:

$$f(a, b, c) = a + a'\ b'\ c + ac'.$$

Si avaluem l'expressió per al cas de la combinació [$a\ b\ c$] = [0 0 0], obtenim el resultat següent:

$$0 + 1 \cdot 1 \cdot 0 + 0 \cdot 1 = 0 + 0 + 0 = 0.$$

Per tant, en la primera fila de la taula hi anirà un 0.

Per a la combinació [$a\ b\ c$] = [0 0 1], obtenim el resultat que presentem a continuació:

$$0 + 1 \cdot 1 \cdot 1 + 0 \cdot 0 = 0 + 1 + 0 = 1.$$

Per tant, en la segona fila hi anirà un 1. I així successivament fins a haver avaluat la funció per a totes les combinacions.

b) Analitzar l'expressió *a vista*, deduir per quins valors de les variables la funció val 1 o 0, i omplir la taula.

Prenem la mateixa funció de l'exemple anterior. Veiem que sempre que $a = 1$, la funció val 1 (per la llei 4, de dominància). Per tant, podem posar un 1 en totes les files en les quals $a = 1$ (les quatre últimes).

A continuació, estudiem els casos en què $a = 0$ (els que ens falten per omplir). L'expressió de la funció ens queda de la manera següent:

$$f(0, b, c) = 0 + 0' \cdot b' \cdot c + 0 \cdot c' = b' \cdot c.$$

Aquesta expressió val 1 només en el cas [$b\ c$] = [0 1]. Per tant, posarem un 1 en la fila corresponent a la combinació [0 0 1].

Per a les combinacions que encara ens queden per omplir, la funció val 0.

Observem que el pas d'expressió a taula de veritat ens pot ser molt útil a l'hora de determinar si dues expressions algebraiques són equivalents o no ho són. Podem obtenir la taula de veritat a partir de cadascuna de les expressions, i si les taules resultants són iguals podem concloure que les dues expressions corresponen a una mateixa funció.

Activitats

12. Obteniu la taula de veritat de les funcions següents:
a) $f(x, y, z, w) = xy'zw' + yz'w + x'z + xyw + x'yz'w'$
b) $f(x, y, z, w) = xy + z$

13. Mitjançant taules de veritat, estudieu si les dues expressions següents són equivalents:
a) $f = x'y' + yw + x'w$,
b) $g = x'y'z'w' + x'z'w + yz'w + xyw + x'z$.

14. Escriviu la taula de veritat corresponent a l'activitat 7.

15. Escriviu la taula de veritat corresponent a l'activitat 8. Trobeu la solució de l'activitat a partir d'estudiar el contingut d'aquesta taula.

El pas de taula de veritat a expressió algebraica es pot fer "a vista", si tenim prou experiència. Però serà més còmode si ho fem coneixent les expressions en suma de mintermes, que veurem en l'apartat següent.

1.3.4. Expressions en suma de mintermes

Ja hem vist que hi ha infinites expressions algebraiques equivalents per a una funció qualsevol. Ara bé, qualsevol funció lògica es pot expressar amb una expressió en suma de productes. És a dir, una expressió de la forma següent:

$$A + B + C,$$

on A, B i C són termes producte, és a dir, tenen la forma que reproduïm a continuació:

$$X \cdot Y \cdot Z,$$

on X, Y i Z són variables lògiques, bé negades o bé sense negar. Per exemple, les dues expressions següents corresponen a la mateixa funció (ho podeu demostrar), però només la segona té la forma de suma de productes:

$$f(x, y, z) = (x + y\ z')'z,$$
$$f(x, y, z) = x'\ y'\ z + x'\ y\ z.$$

A partir de la taula de veritat d'una funció, és molt senzill d'obtenir-ne una expressió en suma de productes. Vegem les taules de veritat de les quatre funcions següents:

x_2	x_1	x_0	f_0	f_1	f_2	F
0	0	0	0	0	0	0
0	0	1	1	0	0	1
0	1	0	0	0	0	0
0	1	1	0	0	0	0
1	0	0	0	1	0	1
1	0	1	0	0	0	0
1	1	0	0	0	1	1
1	1	1	0	0	0	0

Si examinem f_0, f_1 i f_2, veiem que tenen la particularitat de valer 1 només per a una de les combinacions de les variables d'entrada, de manera que les podem expressar fàcilment amb un sol terme producte, en el qual **apareixen totes les variables**, negades o sense negar. Les variables que valen 0 en la combinació que fa que la funció valgui 1 apareixeran negades en el terme producte; les que valen 1 apareixeran sense negar. Així, obtenim els termes producte següents per a les funcions anteriors:

$$f_0(x_2, x_1, x_0) = x_2' \, x_1' \, x_0.$$
$$f_1(x_2, x_1, x_0) = x_2 x_1' x_0'.$$
$$f_2(x_2, x_1, x_0) = x_2 \, x_1 \, x_0'.$$

S'anomena *terme mínim* o **minterme** el terme producte (únic) que expressa una funció que només val 1 per a una sola combinació de les variables d'entrada.

Nota

En un minterme figuren sempre totes les variables de la funció, tant negades com sense negar.

No és tan senzill d'obtenir "a vista" una expressió per a la funció F, ja que val 1 per a diverses combinacions de les variables. Ara bé, sí que podem veure fàcilment que F val 1 si o bé f_0 o bé f_1 o bé f_2 valen 1. És a dir,

$$F = f_0 + f_1 + f_2.$$

Per tant, obtenim el següent:

$$F(x_2,x_1,x_0) = x_2' \, x_1' \, x_0 + x_2 \, x_1' \, x_0' + x_2 \, x_1 \, x_0',$$

que és una expressió algebraica de F en forma de suma de productes.

Així, doncs, a partir de la taula de veritat d'una funció podem obtenir una expressió en suma de productes fent la suma lògica dels mintermes que la formen. Per això, l'expressió que obtenim d'aquesta manera s'anomena **suma de mintermes**.

Nota

Qualsevol funció lògica es pot expressar també en forma de producte de sumes, però en aquest curs no en parlarem.

Les expressions en suma de mintermes o producte de sumes s'anomenen *formes canòniques* d'una funció. Els mintermes s'anomenen *termes canònics*.

En el capítol següent (apartat 2.3) veurem que a partir d'una expressió en suma de mintermes es pot obtenir una altra expressió en suma de productes més senzilla.

Activitats

16. Obteniu l'expressió en suma de mintermes de les funcions f_0, f_1, f_2 i f_3:

x_2	x_1	x_0	f_0	f_1	f_2	f_3
0	0	0	0	0	0	0
0	0	1	0	0	1	1
0	1	0	1	0	0	1
0	1	1	0	0	1	0
1	0	0	0	1	0	1
1	0	1	0	1	1	0
1	1	0	0	1	0	0
1	1	1	1	1	1	1

17. Obteniu l'expressió en forma de suma de mintermes de les funcions següents:
a) $f(x, y, z) = xy + z$,
b) $f(x, y, z, w) = x'y'zw + x'y'z' + xy'w + yz'w$.

1.4. Altres funcions comunes

Hem vist que hi ha quatre funcions lògiques d'una variable (figura 1) i 16 de dues variables (figura 2). Ens hem fixat, especialment, en les funcions f_2, g_1 i g_7, perquè corresponen a les operacions bàsiques de l'àlgebra de Boole, i les hem anomenat, respectivament, *negació* (NOT), *producte lògic* (AND) i *suma lògica* (OR). Després, hem vist que qualsevol funció lògica es pot construir a partir d'aquestes tres.

D'entre les funcions de dues variables, n'hi ha algunes altres que reben també un nom propi, perquè es fan servir de manera comuna. Són les següents:

Funció O-exclusiva o XOR
Aquesta funció val 1 quan alguna de les dues variables d'entrada val 1, però no quan valen 1 totes dues alhora; per això rep el nom d'*O-exclusiva*. És la funció g_6 de la figura 2.

Es representa amb símbol "⊕", i té aquesta taula de veritat:

a	b	$a \oplus b$
0	0	0
0	1	1
1	0	1
1	1	0

$a \oplus b = a'b + ab'$

La funció XOR de més de dues variables es calcula així:

$$a \oplus b \oplus c \oplus d = (((a \oplus b) \oplus c) \oplus d).$$

Per tant, dóna 0 si un nombre parell de les variables val 1, i dóna 1 si un nombre senar de les variables val 1. Fixem-nos que en el cas de dues variables, aquest fet es tradueix en el següent:

$a \oplus b = 0$ => a i b tenen el mateix valor,
$a \oplus b = 1$ => a i b tenen valors diferents.

Per tant, la funció XOR ens permet de saber si dues variables són iguals o no ho són.

Altres propietats de la XOR:

$$0 \oplus a = a,$$
$$1 \oplus a = a'.$$

Així, si fem la XOR d'una variable amb un 0, la variable queda inalterada, mentre que si fem la XOR amb un 1, a la sortida apareix la variable negada.

Funció NAND

És la negació de l'AND, és a dir:

$$a \text{ NAND } b = (a \cdot b)'.$$

Per tant, val 1 sempre que no es compleixi que les dues variables d'entrada valen 1. És, doncs, la funció g_{14} de la figura 2. Aquesta és la seva taula de veritat:

a	b	$(a \cdot b)'$
0	0	1
0	1	1
1	0	1
1	1	0

Funció NOR

És la negació de l'OR, és a dir:

$$a \text{ NOR } b = (a + b)'.$$

Per tant, val 1 només quan cap de les dues variables d'entrada val 1. És la funció g_8 de la figura 2. Aquesta és la seva taula de veritat:

a	b	$(a+b)'$
0	0	1
0	1	0
1	0	0
1	1	0

1.5. Funcions especificades incompletament

Hi ha funcions per a les quals algunes combinacions de les variables d'entrada no es produiran mai. Per exemple, suposem una funció $f(x_2, x_1, x_0)$ en la qual les variables corresponen a senyals connectats a tres aparells de mesura del so en una sala, de manera que tenim el següent:

$x_2 = 1$ si el so supera els 100 dB, $x_2 = 0$ altrament,
$x_1 = 1$ si el so supera els 200 dB, $x_1 = 0$ altrament,
$x_0 = 1$ si el so supera els 300 dB, $x_0 = 0$ altrament.

La combinació $[x_2\ x_1\ x_0] = [0\ 1\ 0]$ no es produirà mai, ja que és impossible que el so estigui per sota dels 100 dB i per sobre dels 200 dB simultàniament. Tampoc no hi haurà mai les combinacions [0 0 1], [0 1 1] i [1 0 1].

Les combinacions que no es produiran mai es diuen **combinacions no importa** o **combinacions *don't care***.

En la taula de veritat d'una funció, escriurem *x* en les files corresponents a les combinacions no importa. Per exemple, suposem que la funció anterior ha de valer 1 si el so està entre (100 dB, 200 dB] o si el so supera els 300 dB. La seva taula de veritat és la següent:

x_2	x_1	x_0	f
0	0	0	0
0	0	1	x
0	1	0	x
0	1	1	x
1	0	0	1
1	0	1	x
1	1	0	0
1	1	1	1

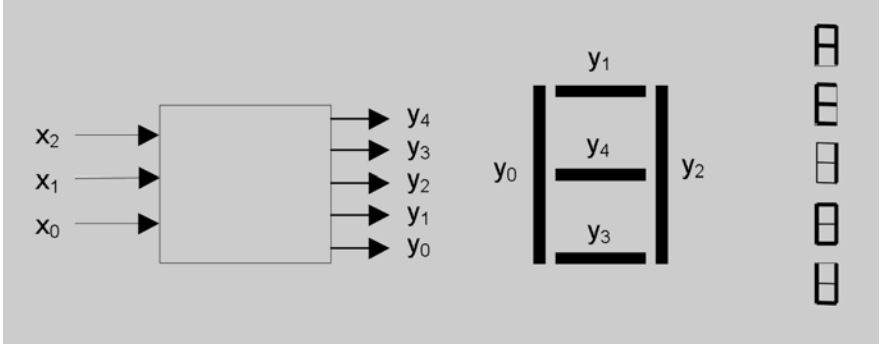
En el capítol següent veurem com es poden obtenir expressions algebraiques de funcions especificades incompletament.

Activitats

18. Es vol construir un sistema que computi cinc funcions de sortida (y_0, y_1, y_2, y_3 i y_4) de tres variables (x_2, x_1 i x_0). Aquestes tres variables codifiquen les cinc vocals, tal com es mostra a la taula següent.

vocal	x_2	x_1	x_0
A	0	0	0
E	0	0	1
I	0	1	0
O	0	1	1
U	1	0	0

Cadascuna de les cinc funcions està associada a un segment d'un *visualitzador de cinc segments* com el que es mostra en la figura. Per exemple, quan y_1 val 1, el segment de dalt el visualitzador s'il·lumina. La figura mostra també quins segments s'han d'il·luminar per a formar les diferents vocals.



En cada moment, el visualitzador ha de formar la vocal que codifiquin les variables d'entrada en aquest moment. Escriviu la taula de veritat de les cinc funcions.

19. Es vol dissenyar un sistema de reg d'una planta amb control de la temperatura i de la humitat de la terra. El sistema té tres senyals d'entrada (variables) i dos de sortida (funcions).

Entrades:

- Un sensor de temperatura (t): es posa a 1 si la temperatura de la terra supera un límit prefixat $T0$.
- Dos sensors d'humitat de la terra (h_0 i h_1): es posen a 1 quan la humitat de la terra supera els límits $H0$ i $H1$, respectivament. El límit $H0$ és inferior al límit $H1$ ($H0 < H1$).

Sortides:

- *Regar* (R): quan es posa a 1, s'activa el reg de la planta.
- *Escalfar* (E): quan es posa a 1, s'activa l'escalfament de la terra.

Les especificacions del sistema són les següents:

- La planta es rega sempre que la terra està seca, és a dir, sempre que no se supera el límit $H0$.
- També es rega quan la temperatura supera el límit $T0$ i la humitat de la terra és inferior a $H1$.
- La terra de la planta s'escalfa quan la temperatura és inferior a $T0$ i la humitat és superior a $H0$.

Escriviu la taula de veritat de les funcions R i E.

2. Implementació de circuits lògics combinacionals

2.1. Portes lògiques. Síntesi i anàlisi

En les figures 1 i 2 havíem vist que es poden construir fins a vint dispositius electrònics diferents per a funcions d'una i dues variables d'entrada. Ara bé, a la pràctica només es construeixen els que calculen les funcions NOT, AND, OR, XOR, NAND i NOR.

Vegeu les figures 1 i 2 en el subapartat 1.1 d'aquest mòdul.

Els dispositius electrònics que calculen funcions lògiques s'anomenen **portes lògiques**. Són dispositius que estan connectats a un cert nombre de senyals d'entrada i un senyal de sortida.

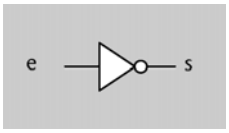
Portes lògiques

Estan formades internament per diferents combinacions de **transistors**, que són els dispositius electrònics més elementals.

A continuació presentem el símbol gràfic amb el qual es representa cada porta lògica. En totes les figures que apareixen a continuació, els senyals d'entrada queden a l'esquerra de les portes i el senyal de sortida queda a la dreta.

Porta NOT o inversor

Aquesta porta es representa gràficament amb aquest símbol:

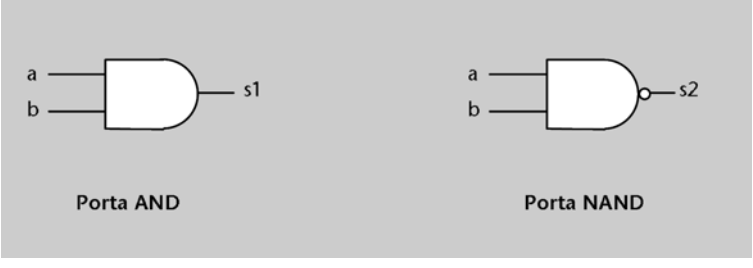


El funcionament és el següent: si en l'entrada *e* hi ha un 0 (tensió baixa), llavors a la sortida *s* hi haurà un 1 (tensió alta). I a la inversa, si hi ha un 1 al punt *e*, llavors hi haurà un 0 al punt *s*.

Per tant, la porta NOT implementa físicament la funció de negació: $s = e'$.

Portes AND i NAND

Es representen gràficament amb aquests símbols:



NAND

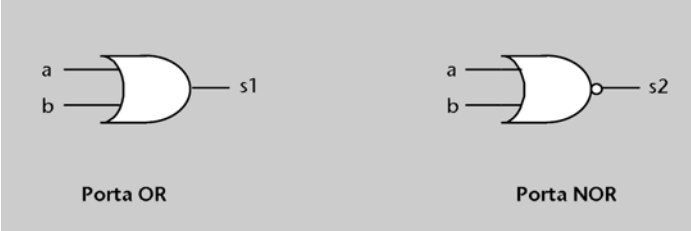
La funció NAND és molt utilitzada en el disseny de circuits reals perquè és molt fàcil implementar-la amb transistors (és a dir, no s'implementa a partir d'una porta AND i una NOT).

La porta AND implementa la funció lògica AND, és a dir: la sortida $s1$ val 1 només si les dues entrades a i b valen 1. Per tant, $s1 = a \cdot b$.

La porta NAND implementa la funció lògica NAND. En el punt *s*2 hi ha un 1 sempre que en alguna de les dues entrades *a* o *b* hi hagi un 0. És a dir, $s2 = (a \cdot b)'$.

Portes OR i NOR

Es representen gràficament amb aquests símbols:



El cercle en un circuit

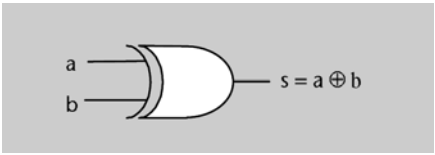
El cercle simbolitza una negació, en senyals de sortida (com en les portes NOT, NAND i NOR). També s'usa un cercle per a negar senyals d'entrada, però en aquest curs no farem servir aquesta possibilitat.

La porta OR implementa la funció lògica OR, és a dir: en la sortida *s*1 hi ha un 1 si qualsevol de les dues entrades és a 1. Per tant, $s1 = a + b$.

La porta NOR implementa la funció lògica NOR. Al punt *s*2 hi trobarem un 1 només quan a totes dues entrades hi hagi un 0, és a dir: $s2 = (a + b)'$.

Porta XOR

Implementa la funció lògica XOR, és a dir: la sortida *s* val 1 si alguna de les dues entrades val 1, però no si valen 1 totes dues alhora. Es representa gràficament amb aquest símbol:



Totes les portes (llevat de la NOT) poden tenir més de dos senyals d'entrada. El seu funcionament és el següent:

- En una porta AND de *n* entrades, la sortida val 1 només quan totes les *n* entrades valen 1.
- En una porta OR de *n* entrades, la sortida val 1 quan una o més d'una de les *n* entrades valen 1.
- En una porta XOR de *n* entrades, la sortida val 1 quan hi ha un nombre senar d'entrades a 1. El 0 es considera un nombre parell, i, per tant, si totes les entrades valen 0 la sortida també val 0.

Síntesi i anàlisi

Qualsevol funció lògica es pot implementar usant aquestes portes, és a dir, es pot construir un circuit que es comporti com la funció.

Dibuix de les portes

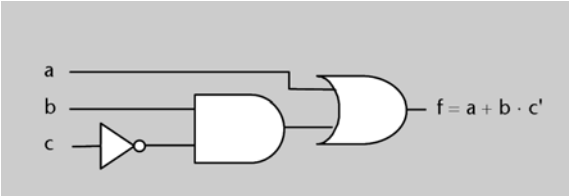
En un circuit les portes es poden dibuixar en qualsevol sentit: amb el senyal de sortida a la dreta o a l'esquerra, i també verticalment amb el senyal de sortida a dalt o a baix, segons convingui per a la claredat del disseny.

El procés d'obtenir el circuit que implementa una funció a partir d'una expressió algebraica s'anomena **síntesi**.

El procés d'obtenir una expressió d'una funció a partir del circuit que la implementa s'anomena **anàlisi**.

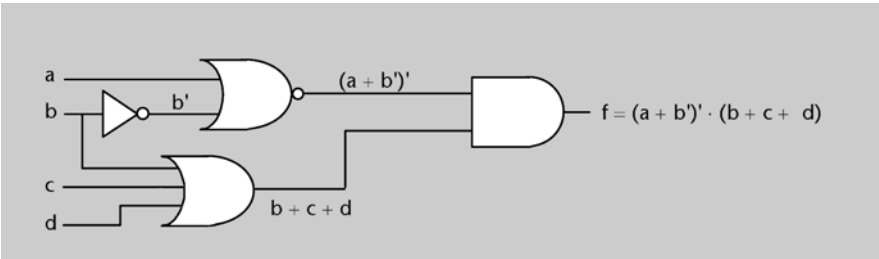
Per a sintetitzar (o implementar) una funció a partir de la seva expressió algebraica n'hi ha prou de substituir cada operador de la funció per la porta lògica adequada. Per exemple, un terme producte de tres variables s'implementarà amb una porta AND de tres entrades. Les portes han d'estar interconnectades entre si i amb les entrades tal com indiqui l'expressió.

Per exemple, el circuit que implementa la funció $f(a, b, c) = a + bc'$ és el següent:

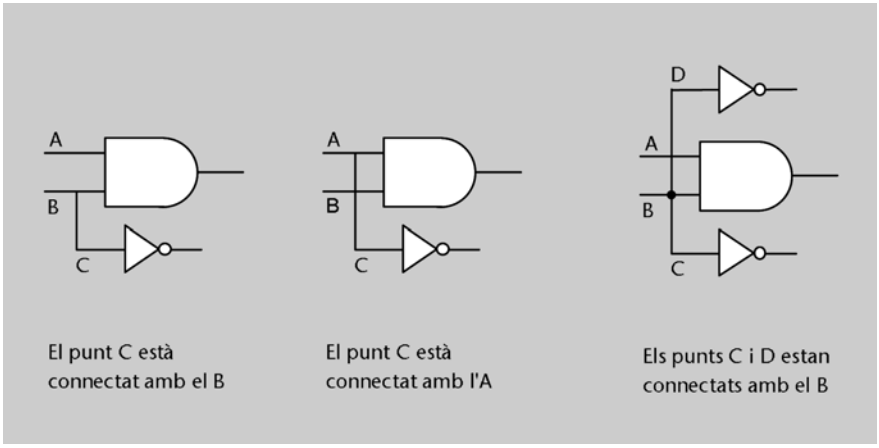


Per a analitzar un circuit, cal escriure les expressions que corresponen a la sortida de cada porta, començant des de les entrades del circuit, fins a obtenir l'expressió corresponent a la línia de sortida del circuit.

Per exemple, el circuit següent implementa la funció $f(a, b, c, d) = (a + b')'(b + c + d)$.



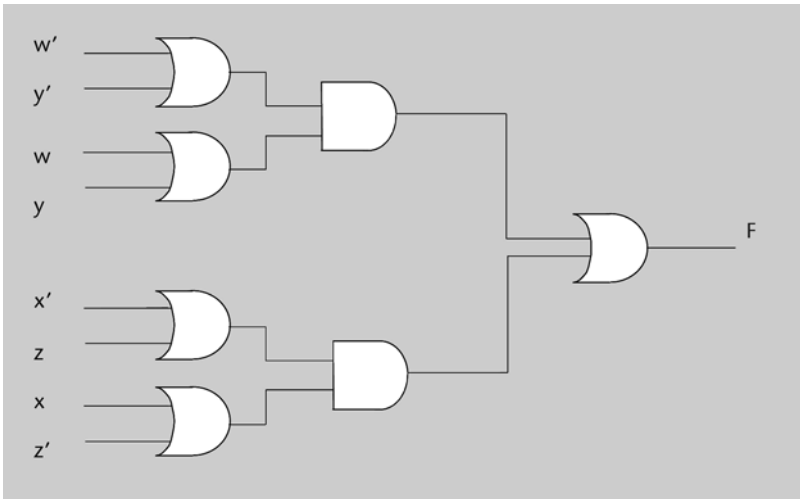
En un circuit, si una línia comença sobre una altra línia perpendicular, s'entén que les línies estan connectades (i, per tant, que tenen el mateix valor lògic). Si dues línies perpendiculars es creuen, s'entén que no estan connectades. Si es posa un punt damunt una línia, s'entén que totes les línies que el toquen estan connectades. A continuació es mostren uns quants exemples.



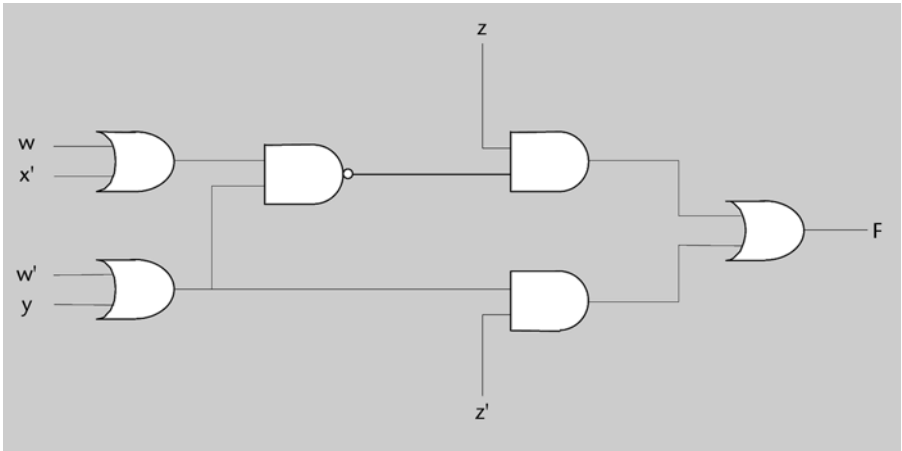
Activitats

20. Analitzeu els circuits següents:

a)



b)



21. Sintetitzeu la funció $f(a, b, c, d) = d \cdot (a' + a \cdot b) \cdot (b' \oplus c)$.

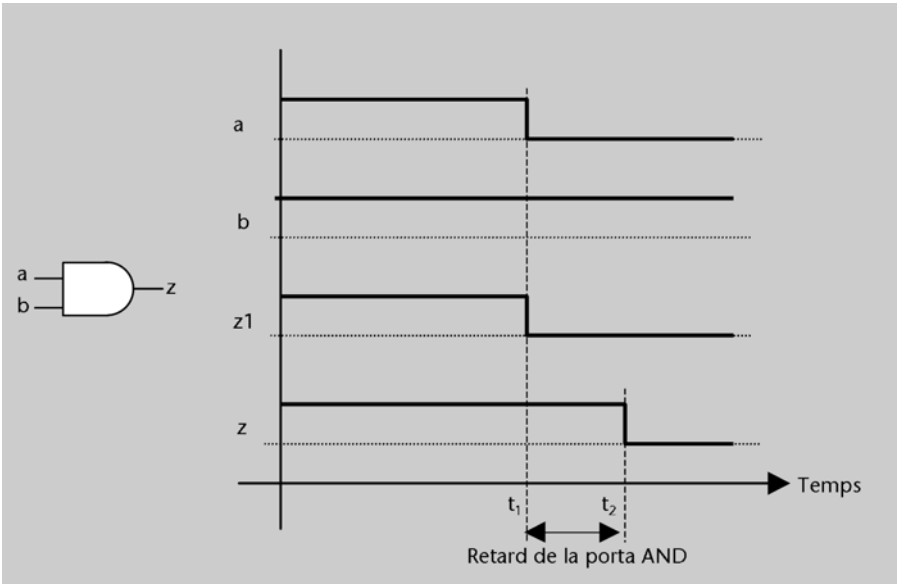
2.2. Disseny de circuits a dos nivells

2.2.1. Retards. Cronogrames. Nivells de portes

Les portes lògiques no responen instantàniament a les variacions en els senyals d'entrada, sinó que tenen un cert **retard**. La millor manera d'entendre aquest concepte és mirant la figura 5, en la qual hi ha un circuit senzill (només una porta AND) i un **cronograma** del seu funcionament.

Un **cronograma** és una representació gràfica de l'evolució dels senyals d'un circuit al llarg del temps.

Figura 5



Cronograma

En un cronograma les línies de punts horitzontals representen el valor lògic 0 per a cada senyal. Les línies contínues gruixudes representen el valor en què es troba cada senyal en cada moment (0 si la línia contínua és sobre la línia de punts, 1 si està més amunt).

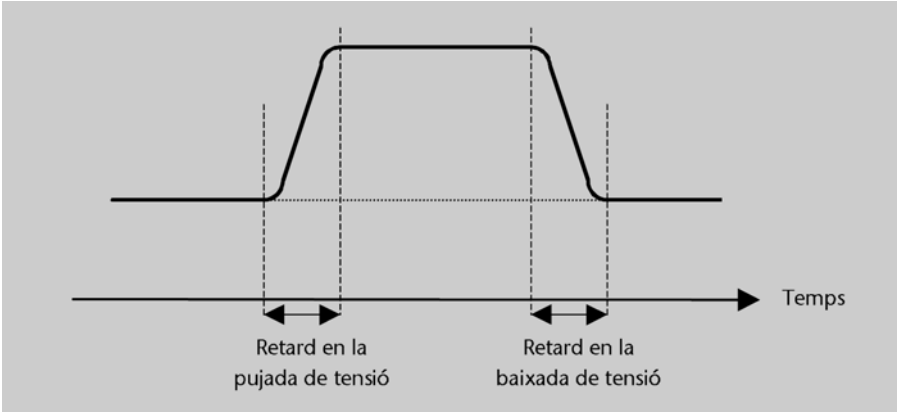
En vertical es poden assenyalar amb línies discontínues instants determinats de temps (com ara t_1 i t_2 en el cas de la figura 5).

Suposem que en les entrades *a* i *b* hi ha connectats dos interruptors que ens permeten en cada moment de posar aquests senyals a 0 o a 1. En l'exemple de la figura 5, hem suposat que inicialment tots dos senyals estan a 1, i que en l'instant t_1 posem el senyal *a* a 0. En aquest moment, el senyal *z* s'hauria de posar a 0, perquè $0 \cdot 1 = 0$. Aquesta situació hipotètica és la que es mostra en el cronograma amb el senyal *z*1. No obstant això, en la realitat *z* no es posa a 0 fins a l'instant t_2, a causa del fet que els dispositius electrònics interns de la porta AND triguen un cert temps a reaccionar. Direm que la porta AND té un retard de $t_2 - t_1$.

Cada porta té un retard diferent, que depèn de la tecnologia que s'hagi usat per a construir-la. Els retards són molt petits, de l'ordre de nanosegons, però cal tenir-los en compte a l'hora de construir físicament un circuit.

De fet, la transició entre diferents nivells de tensió d'un senyal tampoc no és instantània, sinó que es produeix tal com es mostra en la figura 6.

Figura 6



Un dels objectius dels enginyers electrònics que construeixen circuits és que el temps de resposta del circuit sigui el més petit possible. Atès que cada porta té un cert retard, un circuit serà en general més ràpid com menys **nivells de portes** hi hagi entre les entrades i les sortides.

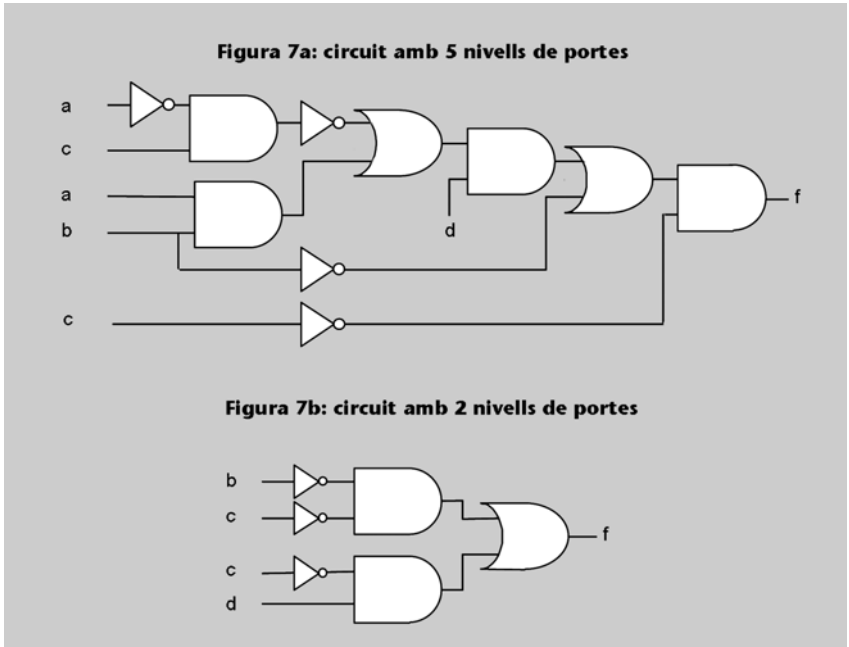
El temps de resposta d'una porta depèn, entre altres coses, del nombre d'entrades de la porta.

El nombre de **nivells de portes** d'un circuit és el màxim nombre de portes que un senyal ha de travessar consecutivament per a generar el senyal de sortida.

En comptabilitzar el nombre de nivells de portes d'un circuit **no es tenen en compte les portes NOT** (per raons que no veurem en aquest curs).

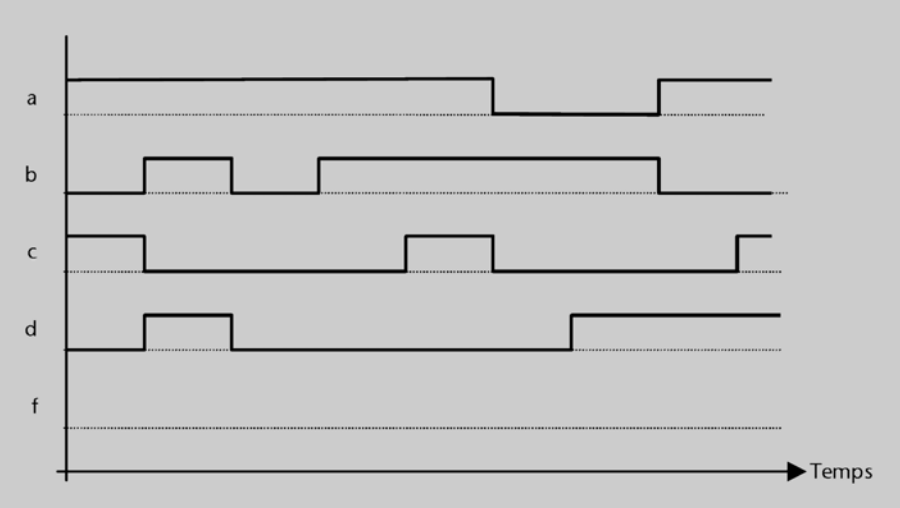
Per exemple, la figura 7 mostra el nombre de nivells de portes de dos circuits diferents (podeu comprovar que la funció que implementen és la mateixa). Un bon enginyer escolliria el circuit de la figura 7*b* per a implementar aquesta funció, ja que és més ràpid.

Figura 7



Activitats

22. Completeu el cronograma següent, suposant que el senyal *f* correspon a la funció de la figura 7*b* i que les entrades prenen els valors dibuixats. Considereu que els retards introduïts per les portes són 0.

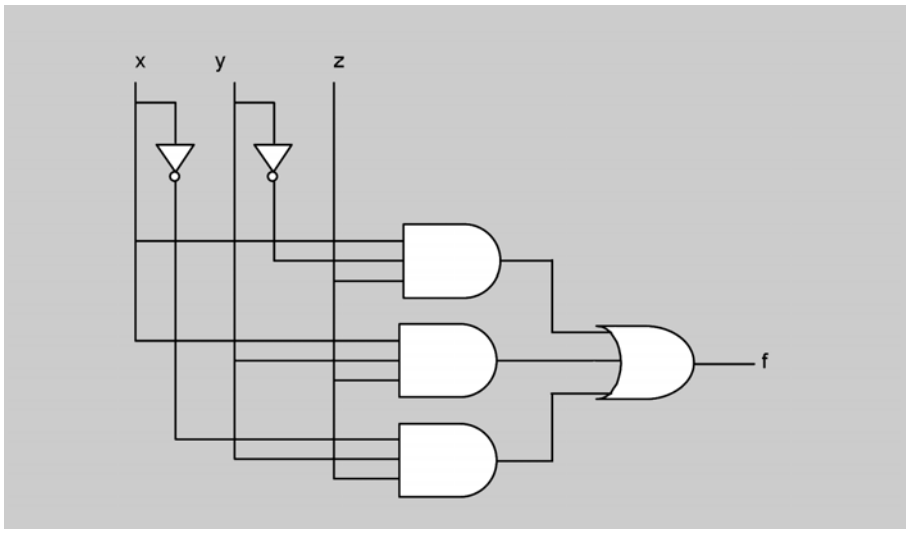


2.2.2. Síntesi a dos nivells

No hi ha una fórmula universal per a trobar l'expressió d'una funció que donarà lloc al circuit més ràpid possible. No obstant això, coneixem una manera de garantir que un circuit no tindrà més de dos nivells de portes: partir de l'expressió de la funció en suma de mintermes. En efecte, el circuit corresponent a una suma de mintermes té, a més de les portes NOT, un primer nivell de portes AND (que computen els diferents mintermes) i un segon nivell en el qual hi haurà una única porta OR, amb tantes entrades com mintermes tingui l'expressió. Per exemple, la figura 8 mostra el circuit que s'obté quan se sintetitza aquesta funció:

$$f = xy'z + xyz + x'yz.$$

Figura 8



Els circuits que s'obtenen a partir de les sumes de mintermes s'anomenen **circuits a dos nivells**. El procés pel qual els obtenim s'anomena **síntesi a dos nivells**.

És possible que una funció es pugui implementar amb un circuit que tingui menys de dos nivells. No obstant això, no hi ha una manera de trobar-lo sistemàticament. Així, doncs, les expressions de les funcions en suma de mintermes ens permeten de construir els circuits en dos nivells més ràpids possibles que podem obtenir de manera sistemàtica.

D'altra banda, a partir de l'expressió d'una funció en suma de mintermes en pot resultar un circuit amb menys de dos nivells: si hi ha un sol minterme no hi haurà el nivell OR.

Tenir en compte els retards de les diferents portes i de les transicions entre nivells de tensió és fonamental a l'hora de construir circuits. Això no obstant, en aquest curs considerarem que els circuits són ideals, de manera que no es tindran mai en compte els retards, és a dir, s'assumirà sempre que són 0.

Hi ha altres mètodes de síntesi que generen circuits amb més de dos nivells però més ràpids que els que s'obtenen a partir de les expressions de les funcions en suma de mintermes. En aquest curs no els veurem.

Activitats

23. Feu la síntesi a dos nivells de les funcions següents:
a) $f(x,y,z,w) = x'y'z'w' + x'yz'w' + xy'zw' + xyzw + x'y'z'w + x'yzw$.
b) $f(x,y,z) = xz' + y'z + x'y$.

24. Es vol dissenyar un circuit combinacional que permeti multiplicar dos nombres naturals de dos bits.
a) Indiqueu el nombre de bits de la sortida.
b) Escriviu la taula de veritat de les funcions de sortida.
c) Implementeu el circuit a dos nivells.

25. Sintetitzeu a dos nivells la funció següent:

x_2	x_1	x_0	f
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

2.3. Minimització de funcions

Ja hem vist que un dels objectius que s'ha d'aconseguir quan construïm un circuit és que sigui tan ràpid com sigui possible. La síntesi a dos nivells ens permet d'aconseguir de manera sistemàtica un grau de rapidesa acceptable.

També és desitjable que un circuit sigui petit i barat. Aquestes dues característiques estan relacionades amb el nombre de portes del circuit: com menys n'hi hagi, més petit i barat serà.

2.3.1. Simplificació d'expressions

En el capítol anterior s'ha vist com es pot obtenir l'expressió d'una funció en suma de mintermes. A vegades, aquestes expressions es poden simplificar, la qual cosa ens permetrà d'implementar la funció amb un circuit més petit.

En concret, les simplificacions que ens seran útils corresponen als casos en què **dos o més mintermes es poden reduir a un sol terme producte**, en el qual apareixeran menys variables.

Per exemple, sigui la funció següent:

$$f(x, y, z) = x'yz' + x'yz + \ldots$$

Aquests dos mintermes ens diuen que la funció val 1 si $x = 0$, $y = 1$ i $z = 0$ o bé si $x = 0$, $y = 1$ i $z = 1$. Però aquesta informació es pot resumir dient que la funció val 1 sempre que $x = 0$ i $y = 1$, independentment del valor de z. Això es pot expressar algebraicament traient factor comú en els dos mintermes és a dir, aplicant les propietats distributiva, de complementació i d'element neutre):

$$f(x,y,z) = x'yz' + x'yz + \ldots$$
$$= x'y(z'+z) + \ldots = x'y \cdot 1 + \ldots = x'y + \ldots$$

Prenem ara una funció que val 1 per a les següents combinacions [x y z] (entre altres):

[1 0 0]
[1 0 1]
[1 1 0]
[1 1 1]

Aquesta funció val 1 sempre que $x = 1$, independentment del valor de y i z. Aplicant el mateix raonament que abans ho podem expressar algebraicament així:

$$f(x,y,z) = xy'z' + xy'z + xyz' + xyz + \ldots =$$
$$xy'(z'+z) + xy\,(z'+z) + \ldots =$$
$$xy' + xy + \ldots = x\,(y'+y) + \ldots = x + \ldots$$

Així, doncs, veiem que podem obtenir un sol terme producte en els casos següents:

1. Si en dos mintermes totes les variables es mantenen constants llevat d'una, llavors podem treure factor comú eliminant la variable que canvia. En el terme producte resultant hi apareixeran $n - 1$ variables, essent n el nombre de variables de la funció.

2. Si en quatre mintermes totes les variables es mantenen constants llevat de dues, llavors podem treure factor comú dues vegades i eliminar les dues variables que canvien. En el terme producte resultant hi apareixeran $n - 2$ variables.

3. En general, si en 2^m mintermes totes les variables es mantenen constants llevat de m, llavors podem treure factor comú m vegades i eliminar les m variables que canvien. En el terme producte resultant hi apareixeran $n - m$ variables.

També podem detectar altres casos en què es pot treure factor comú en una expressió algebraica. Per exemple, $f(x, y, z) = xy' + xz = x \cdot (y' + z)$. Però aquests casos no ens interessen, perquè l'expressió resultant no és una suma de productes.

El fet d'obtenir l'expressió en suma de productes més simplificada possible d'una funció, traient factor comú en els casos que s'acaben de descriure, s'anomena **minimitzar la funció**.

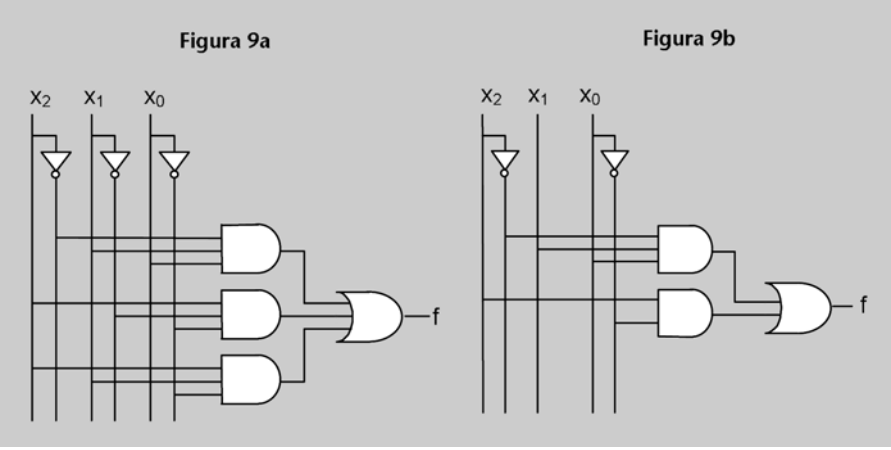
A partir de l'expressió minimitzada d'una funció podrem construir circuits més petits i barats que els que obteníem de l'expressió en suma de mintermes: potser caldran menys portes NOT, algunes de les portes AND seran de menys entrades, tindran menys portes AND i la porta OR serà de menys entrades.

La figura 9 mostra els circuits corresponents a l'expressió en suma de mintermes (figura 9*a*) i a l'expressió minimitzada (figura 9*b*) d'aquesta funció:

$$f(x_2,x_1,x_0) = x_2'x_1x_0 + x_2x_1'x_0' + x_2x_1x_0' =$$
$$= x_2'x_1x_0 + x_2x_0' \ (x_1'+x_1) = x_2'x_1x_0 + x_2x_0'.$$

Es pot veure que el circuit de la figura 9*a* té tres portes NOT, tres portes AND de tres entrades i una porta OR de tres entrades. El circuit simplificat, en canvi, té només dues portes NOT, dues portes AND (una de dues entrades i l'altra de tres) i una porta OR de dues entrades.

Figura 9



2.3.2. Síntesi mínima a dos nivells. Mètode de Karnaugh

Mirant la taula de veritat de la funció podem detectar els casos en què l'expressió en suma de mintermes es pot minimitzar. Ara bé, en alguns casos ho po-

dem veure fàcilment, però en altres és més difícil. Per exemple, la figura 10*a* mostra la taula de veritat de la funció:

$$f(x_2, x_1, x_0) = x_2'x_1.$$

Només mirant la taula és fàcil d'obtenir aquesta expressió, perquè les combinacions per les quals la funció val 1 (els mintermes) estan en files consecutives.

La funció de la figura 10*b* és $f(x_2, x_1, x_0) = x_1'x_0$, perquè val 1 sempre que $x_1 = 0$ i $x_0 = 1$, independentment de quant valgui x_2. Però en aquest cas no és tan senzill de veure-ho a simple vista, perquè els mintermes no estan en files consecutives.

Figura 10

Figura 10a

x_2	x_1	x_0	f
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

Figura 10b

x_2	x_1	x_0	f
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0

El **mètode de Karnaugh** proporciona una mecànica senzilla per a detectar visualment els casos en què es pot minimitzar una expressió en suma de mintermes. Per tant, d'entre tots els circuits a dos nivells que implementen una funció, permet d'obtenir fàcilment el més petit de tots.

El mètode de Karnaugh consta de quatre passos:

1) Traslladar la taula de veritat de la funció a una estructura que s'anomena *mapa de Karnaugh*.
2) Detectar visualment els casos en què es pot treure factor comú.
3) Deduir els termes producte més simples possible.
4) Obtenir l'expressió mínima de la funció fent la suma lògica dels termes producte.

Vegem detalladament com es duu a terme cadascun d'aquests passos.

1) Construcció del mapa de Karnaugh

El **mapa de Karnaugh** és una transcripció de la taula de veritat d'una funció a una estructura formada per caselles en la qual cada casella correspon a una combinació de les variables (i, per tant, a una fila de la taula de veritat).

Es diu que dues caselles del mapa són **adjacents** si corresponen a combinacions en les quals només canvia el valor d'una variable.

La figura 11 mostra l'estructura del mapa de Karnaugh per a funcions de 2, 3 i 4 variables. Per exemple, la casella que està ombrejada amb gris clar correspon a la combinació [*a b c d*] = [1 1 0 1].

També és possible aplicar el mètode de Karnaugh a funcions de més de quatre variables, però en aquest curs no s'estudiarà. De fet, per a aquests casos hi ha altres mètodes més adequats, que no estudiarem.

Figura 11

Estructura dels mapes de Karnaugh per a funcions de

2 variables

a / b	0	1
0		
1		

3 variables

a b / c	00	01	11	10
0				
1				

4 variables

a b / c d	00	01	11	10
00				
01				
11				
10				

Fixem-nos que en les capçaleres de les files i les columnes dels mapes de Karnaugh les combinacions no estan en ordre lexicogràfic.

D'aquesta manera es compleix que les caselles adjacents queden disposades en el mapa de la manera següent:

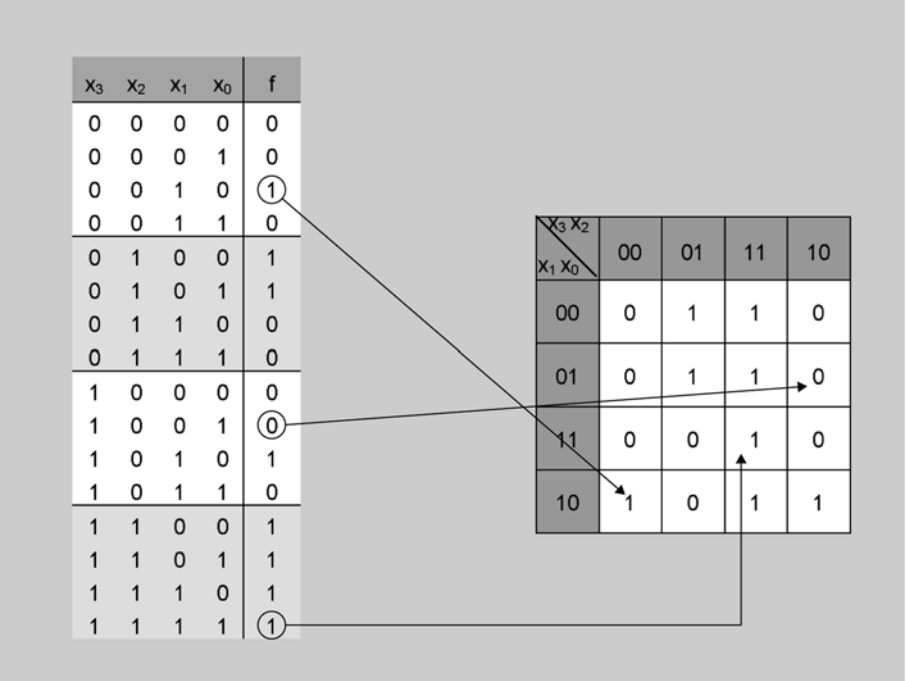
1. Dues caselles on únicament canvia el valor d'una variable són adjacents.

2. En els mapes de tres i quatre variables, les caselles de la columna de més a la dreta també són adjacents amb les de la columna de més a l'esquerra.

3. En els mapes de quatre variables, les caselles de la fila superior també són adjacents amb les de la fila inferior.

Una vegada dibuixat el mapa, posarem dins de cada casella el valor de la funció per a la combinació corresponent de variables, a partir de la taula de veritat. En la figura 12 se'n pot veure un exemple en què es mostra explícitament la posició d'alguns mintermes en la taula.

Figura 12



De fet, n'hi ha prou d'omplir les caselles per a les quals la funció val 1.

2) Detecció dels casos en què es pot treure factor comú

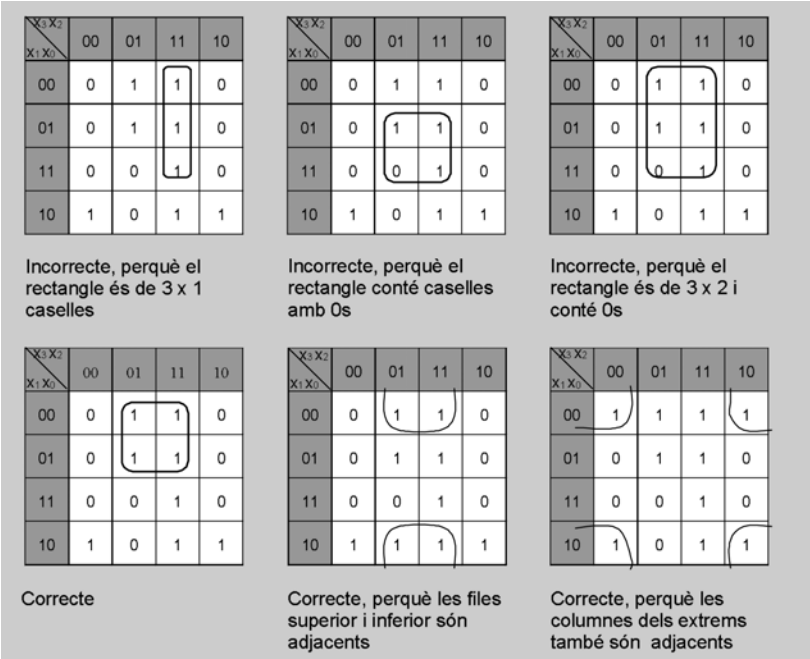
Suposem que dues caselles adjacents contenen uns; corresponen, doncs, a dos mintermes de la funció. Però entre aquestes només canvia el valor d'una variable (per la definició d'adjacència), i, per tant, corresponen a un cas en què es pot treure factor comú dels dos mintermes.

Suposem ara que quatre caselles adjacents contenen uns. Entre aquestes només canvia el valor de dues variables, i, per tant, corresponen a un cas en què es pot treure factor comú dues vegades.

Així, el segon pas del mètode de Karnaugh consisteix a agrupar amb rectangles els uns que estiguin en caselles adjacents, formant grups d'1, 2, 4, 8 o 16 uns. Els costats d'aquests rectangles han de ser d'un nombre de caselles potència de 2, i en el seu interior només hi pot haver uns.

En la figura 13 es mostren diversos exemples de rectangles correctes i incorrectes.

Figura 13

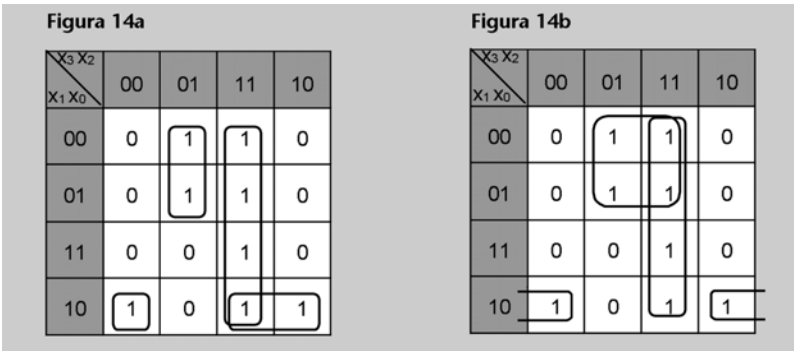


La manera d'agrupar els uns d'un mapa no és única. En fer les agrupacions, cal tenir en compte el següent:

– Tots els uns han de formar part d'algun grup.
– Els grups han de ser com més grans millor (per tal que a cada terme hi aparegui el nombre més petit possible de variables). Per això, començarem buscant els grups més grans que puguem fer (recordeu que dintre d'un grup només n'hi pot haver uns).
– Com menys grups hi hagi, millor (per tal d'obtenir el nombre més petit possible de termes producte). El fet de buscar els grups més grans redueix el nombre total de grups que es faran.
– Un mateix 1 pot formar part de més d'un grup si això ajuda a satisfer els dos objectius anteriors. Si, després de fer els grups grans que veiem a primera vista, queda algun 1 dispers, mirem si el podem agrupar amb altres 1 que ja formin part d'algun grup.

En la figura 14 es mostren dues maneres d'agrupar els uns del mapa de l'exemple anterior. La de la figura 14a no és incorrecta, però la de la figura 14*b* és millor. Sempre cal procurar trobar l'agrupació òptima.

Figura 14



3) Deducció dels termes producte

Prenem el mapa de la figura 14*b*. El grup dels quatre uns de la tercera columna correspon a les combinacions $[x_3\ x_2\ x_1\ x_0]$ =

$$[1\ 1\ 0\ 0]$$
$$[1\ 1\ 0\ 1]$$
$$[1\ 1\ 1\ 0]$$
$$[1\ 1\ 1\ 1]$$

L'expressió en suma de mintermes que s'obtindria d'aquests quatre uns és $x_3x_2x_1'x_0' + x_3x_2x_1'x_0 + x_3x_2x_1x_0' + x_3x_2x_1x_0$. Si traiem factor comú, obtenim x_3x_2, perquè el valor de la funció és independent de x_1 i x_0. Si mirem el mapa, podem veure que les variables x_3 i x_2 no canvien de valor dins el rectangle, mentre que x_1 i x_0 prenen totes les combinacions possibles.

Així, doncs, obtindrem un terme producte de cada grup de la manera següent:

1. Només hi apareixen les variables el valor de les quals és constant per a totes les caselles que formen el grup.
2. Si en totes les caselles del grup una variable val 1, la variable apareix en el terme producte sense negar.
3. Si en totes les caselles del grup una variable val 0, la variable apareix al terme producte negada.

Així, en el mapa de la figura 14*b,* de l'altre grup de quatre uns n'obtenim el terme

$$x_2x_1'.$$

Del grup de dos uns n'obtenim el terme

$$x_2'x_1x_0'.$$

Observeu

A partir d'un rectangle de 2^m uns obtenim un terme producte amb $n - m$ variables, essent n el nombre de variables de la funció. Per exemple, en el terme $x_2'x_1x_0'$, que s'obté a partir d'un rectangle de dos uns ($m = 1$, $n - m = 3$), hi ha tres variables. Mentre que el terme x_2x_1' té només dues variables, perquè s'obté d'un rectangle amb quatre uns ($m = 2$, $n - m = 2$).

4) Obtenció de l'expressió mínima de la funció

Ja sabem que una funció es pot expressar fent la suma lògica de tots els seus mintermes. Els termes producte obtinguts en el pas anterior "resumeixen" els mintermes d'una funció. Per tant, podem expressar la funció fent la suma lògica d'aquests termes producte.

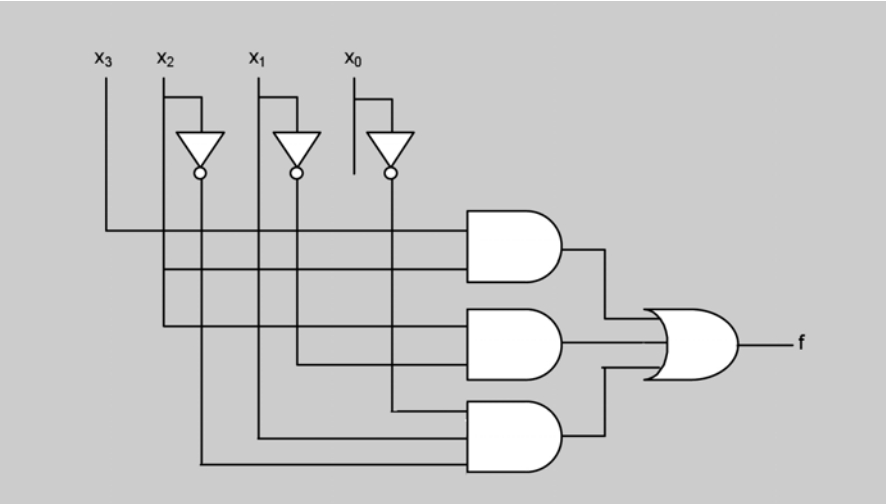
En l'exemple de la figura 14*b*, l'expressió minimitzada de la funció és la següent:

$$f(x_3,x_2,x_1,x_0) = x_3x_2 + x_2x_1' + x_2'x_1x_0'.$$

La figura 15 mostra el circuit a dos nivells que implementa aquesta funció a partir de l'expressió minimitzada. Es pot veure que és més petit i barat que el que obtindríem a partir de l'expressió en suma de mintermes original, ja que

aquest tindria quatre portes NOT, vuit portes AND de tres entrades cadascuna i una porta OR de vuit entrades. De tots els circuits a dos nivells que implementen aquesta funció, el de la figura 15 és el més petit i barat; es diu que és el **circuit mínim a dos nivells**.

Figura 15



Activitats

26. Sintetitzeu de manera mínima a dos nivells la funció descrita en l'activitat 7.

27. Sintetitzeu de manera mínima a dos nivells la funció descrita en l'activitat 23*a*. Compareu la resposta amb la que heu obtingut en l'activitat 23.

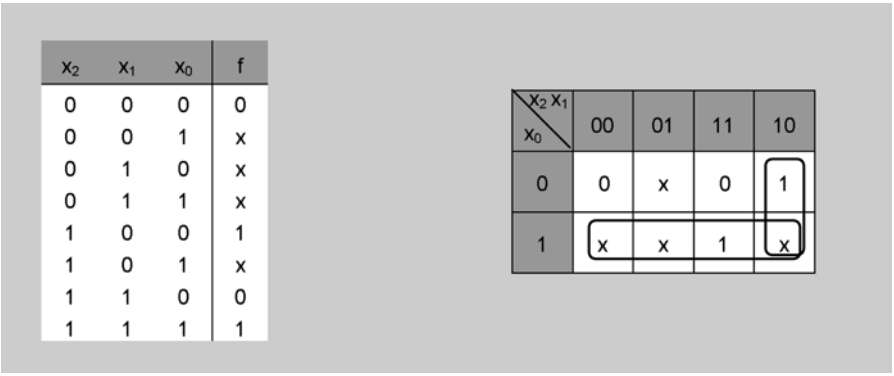
2.3.3. Minimització de funcions especificades incompletament

Tal com hem vist en l'apartat 1.5, el valor de les funcions especificades incompletament és indiferent per algunes combinacions de les variables (les combinacions "no importa"). És a dir, per les combinacions "no importa" tant podem suposar que la funció val 1 com que val 0.

En la taula de veritat de la funció posem *x* en les files corresponents a les combinacions "no importa". En el mapa de Karnaugh també posarem *x* en les caselles corresponents. Atès que el valor de la funció en aquests casos és indiferent, suposarem que les *x* són un 1 o un 0, segons el que ens convingui més, tenint en compte que sempre hem de procurar d'obtenir el nombre més petit possible d'agrupacions i que les agrupacions siguin tan grans com sigui possible.

Prenem la funció d'exemple que hem vist en l'apartat 1.5. La figura 16 mostra la seva taula de veritat i el mapa de Karnaugh, amb les agrupacions de caselles.

Figura 16



x_2	x_1	x_0	f
0	0	0	0
0	0	1	x
0	1	0	x
0	1	1	x
1	0	0	1
1	0	1	x
1	1	0	0
1	1	1	1

x_0 \ $x_2 x_1$	00	01	11	10
0	0	x	0	1
1	x	x	1	x

Ens interessa de suposar que les *x* de la fila inferior valen 1, perquè així obtenim un grup de quatre uns i un grup de dos uns. Per tant, l'expressió mínima de la funció és la següent:

$$f = x_0 + x_2 x_1'.$$

Fixem-nos que la *x* de la casella $[x_2\ x_1\ x_0] = [0\ 1\ 0]$ l'hem presa com a 0, perquè si no ho féssim així obtindríem un grup addicional innecessàriament (els grups han de cobrir tots els uns, però no cal que cobreixin totes les *x*).

Activitats

28. Sintetitzeu de manera mínima a dos nivells la funció descrita en l'activitat 19.

3. Blocs combinacionals

Un **bloc combinacional** és un circuit lògic combinacional amb una funcionalitat determinada. Està construït a partir de portes, com els circuits que hem vist fins ara.

Fins aquest moment, les "peces" que hem fet servir per a sintetitzar circuits han estat portes lògiques. Després d'estudiar aquest capítol, podrem fer servir també els blocs combinacionals com a peces per a dissenyar circuits més complexos, com per exemple un computador, que no es poden pensar a nivell de portes però sí a nivell de blocs.

Hi ha molts blocs combinacionals, en aquest curs només en veurem els més bàsics.

3.1. Multiplexor. Multiplexor de busos. Demultiplexor

Imaginem que en una ciutat hi ha tres carrers que conflueixen en un altre carrer d'un sol carril. Farà falta un urbà o algun tipus de senyalització per a controlar que en cada moment circulin cap al carrer de sortida els cotxes provinents d'un únic carrer confluent.

Un **multiplexor** és un bloc que fa la funció d'urbà en circuits electrònics. Té un cert nombre de senyals d'entrada que "competeixen" per a connectar-se a un senyal únic de sortida, i uns senyals de control que serveixen per a determinar quin senyals d'entrada es connecten en cada moment amb la sortida.

Més concretament, les entrades i sortida d'un multiplexor són les següents:

1. 2^m entrades *de dades*, identificades per la lletra *e* i numerades des de 0 fins a $2^m - 1$. Direm que l'entrada de dades numerada amb el 0 és la *de menys pes*, i la numerada amb el $2^m - 1$, la *de més pes*.

2. Una sortida de dades, *s*.

3. *m* entrades *de control* o *de selecció*, identificades per la lletra *c* i numerades des de 0 fins a $m - 1$. Direm que l'entrada de control numerada amb el 0 és la *de menys pes*, i la numerada amb $m - 1$, la *de més pes*.

4. Una entrada *de validació*, que anomenem VAL.

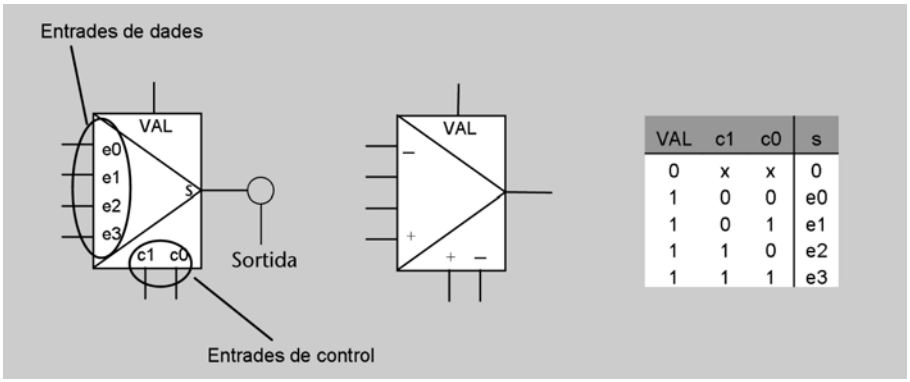
Per a especificar la mida d'un multiplexor, direm que és un multiplexor 2^m-1; per exemple, un multiplexor 4-1 és un multiplexor de quatre entrades de dades i dues de control.

La figura 17 mostra dues possibles representacions gràfiques d'un multiplexor 4-1 , és a dir, amb $m = 2$ (vegeu el primer requadre al marge). La diferència entre aquestes és que en la primera posem els noms de les entrades, mentre que en la segona només indiquem amb els signes "+" i "–" quines són les de més i menys pes (la resta s'assumeix que estan ordenades en ordre de pes). Totes dues representacions són igualment vàlides.

Alguns dels blocs combinacionals que es descriuen en aquest capítol, a més de les portes lògiques descrites al capítol 2, poden trobar-se al mercat en forma de xips. Cadascuna de les entrades o sortides dels circuits es correspon amb una "pota" (anomenada *pin*) del xip. Així doncs, una porta AND de dues entrades necessita 3 pins d'un xip, dos per a les seves entrades i un per a la sortida. D'altra banda, un multiplexor 4-1 necessita 7 pins: 4 per a les entrades de dades, 2 per a les entrades de control i 1 per a la sortida de dades. A més, com que les portes i els blocs són circuits electrònics, necessiten connectar-se a una font d'alimentació (els circuits lògics funcionen amb corrent continu). Per aquest motiu, tots els xips tenen 2 pins addicionals, un per a connectar-se al positiu de la font d'alimentació i un altre per a connectar-se a massa (0 volts).

En general no escriurem el nom de la sortida (tal com es fa en la segona representació) perquè la podem identificar inequívocament perquè està dibuixada en el punt del multiplexor en què s'ajunten les dues línies obliqües interiors. També, en el cas dels multiplexors 2-1, que només tenen una entrada de control, la podem identificar per la lletra c (sense número) o bé no posar-hi el nom, ja que la podem identificar inequívocament pel fet que estarà dibuixada en un costat curt del multiplexor.

Figura 17



VAL	c1	c0	s
0	x	x	0
1	0	0	e0
1	0	1	e1
1	1	0	e2
1	1	1	e3

Nota

En aquesta figura l'entrada de dades de més pes del multiplexor és a la part inferior, i la de menys pes, a la part superior. Podem invertir l'ordre si és més convenient per a la claredat d'un circuit. Igualment, les entrades de control poden girar-se (més pes a la dreta, menys a l'esquerra).

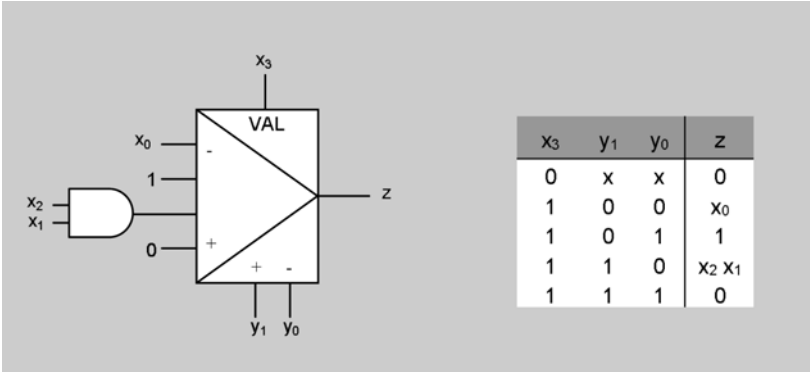
La figura 17 també mostra la taula de veritat que descriu el funcionament del multiplexor (en aquest cas un multiplexor 4-1), que és el següent:

- Quan l'entrada de validació val 0, la sortida del multiplexor es posa a 0 independenment del valor de les entrades de dades. En la taula, això ho reflectim posant *x* en les columnes corresponents a les entrades de dades, en la fila en què VAL = 0.

- Quan l'entrada de validació està activa (val 1), llavors les entrades de control determinen quina de les entrades de dades es connecta amb la sortida, de la manera següent: s'interpreten les variables connectades a les entrades de control (c_1 i c_0 a l'exemple) com un número codificat en binari amb *m* bits (l'entrada de més pes correspon al bit de més pes); si el número codificat és *i*, l'entrada de dades que es connecta amb la sortida és la numerada amb el número *i*.

Si no dibuixem l'entrada de validació en un multiplexor, assumirem que aquesta està a 1.

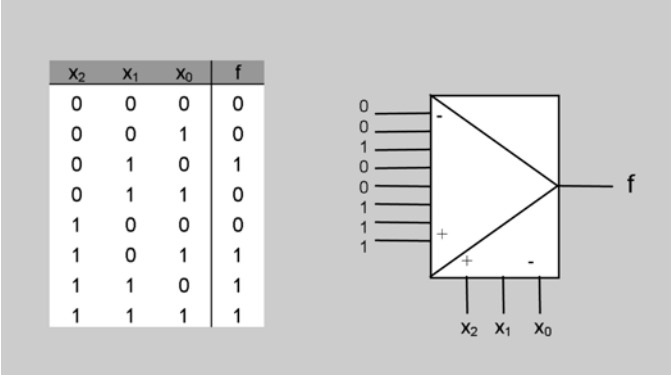
A les entrades i sortides d'un multiplexor hi connectarem els senyals lògics que convinguin per al funcionament dels circuits. Per exemple, si connectem els senyals d'entrada x_3, x_2, x_1, x_0, y_1, y_0 a un multiplexor 4-1 tal com es mostra en la figura següent, el senyal de sortida *z* prendrà els valors que es descriuen en la taula de veritat de la dreta.



x_3	y_1	y_0	z
0	x	x	0
1	0	0	x_0
1	0	1	1
1	1	0	x_2 x_1
1	1	1	0

Una possible aplicació dels multiplexors és la implementació de funcions lògiques. La figura 18 mostra la implementació d'una funció amb un multiplexor que té tantes entrades de control com variables té la funció. La sortida del multiplexor valdrà 1 si les variables connectades als senyals de control construeixen les combinacions 2, 5, 6 o 7, que corresponen als casos en què la funció *f* val 1.

Figura 18



x_2	x_1	x_0	f
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Multiplexor de busos

Un **mot de *n* bits** és una agrupació de *n* bits, usualment amb un significat semàntic conjunt (per exemple, un número codificat en binari amb *n* bits).

Per convenció, usarem lletres majúscules per a donar nom a un mot o variable de *n* bits. Cadascun dels bits que el formen l'identificarem per la mateixa lletra en minúscula, juntament amb un subíndex per a identificar-ne el pes. Per exemple,

$$A = a_7a_6a_5a_4a_3a_2a_1a_0 = a_{7..0}$$

Per a referir-nos a un subconjunt dels bits d'un mot escriurem els subíndexs corresponents. Per exemple, $a_{4..1}$.

Un **bus** és una agrupació d'un cert nombre de cables, per cadascun dels quals circula un bit. Així, un mot de *n* bits circularà per un bus de *n* bits. Direm que *n* és l'*amplada* o *mida* del bus.

Els busos es representen gràficament tal com es mostra a la figura 19*a*. En la línia que representa el senyal hi posem una ratlleta transversal acompanyada d'un número que indica el nombre de bits del bus.

En dibuixar circuits, és fonamental indicar de quants bits són tots els busos que hi apareixen. Una línia que no especifiqui el nombre de bits correspondrà sempre a un senyal d'un sol bit.

A vegades interessa de disgregar els bits que formen un bus o d'agregar-n'hi d'addicionals. Les figures 19*b*, 19*c* i 19*d* mostren uns quants exemples de com ho representarem gràficament.

Figura 19

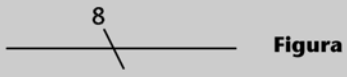


Figura 19a: bus de 8 bits

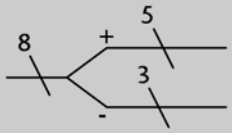


Figura 19b: el bus de 8 bits es disgrega en dos busos, un format pels 5 bits de més pes i un altre format pels 3 bits de menys pes.

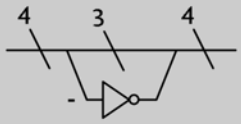


Figura 19c: la línia inferior correspon al bit de menys pes del bus de 4 bits. Aquest bit es nega i després s'agrega de nou al bus original

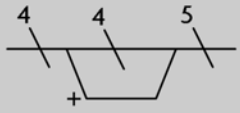
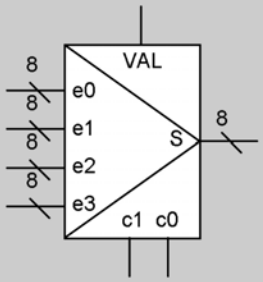


Figura 19d: la línia inferior correspon a una replicació del bit de més pes del bus de 4 bits, perquè el bus original continua essent de 4 bits. Aquest nou bit s'agrega posteriorment al bus original, quedant finalment un bus de 5 bits.

Un **multiplexor de busos** funciona igual que un multiplexor de bits, però en aquest cas les entrades de dades i la sortida són busos d'un determinat nombre de bits. La figura 20 mostra la representació gràfica d'un multiplexor 4-1 de busos de vuit bits.

Figura 20



VAL	c_1	c_0	S
0	x	x	0
1	0	0	e0
1	0	1	e1
1	1	0	e2
1	1	1	e3

Activitats

29. Obteniu l'expressió algebraica de la sortida *s* d'un multiplexor 4-1. Feu la seva implementació interna mitjançant portes lògiques.
30. Implementeu la funció $f(a, b, c) = abc' + a'c$ amb un multiplexor 8-1.
31. Implementeu un circuit combinacional que permeti de multiplicar dos nombres enters de dos bits representats en complement a 2 utilitzant només un multiplexor.
32. Dissenyeu un multiplexor 16-1 utilitzant dos multiplexors 8-1 i les portes lògiques que facin falta.
33. Construïu un circuit amb una entrada *X* de 8 bits per la qual arriba un nombre natural representat en binari i una sortida *Z* també de 8 bits, de manera que $Z = X$ si *X* és parell i $Z = 0$ si *X* és senar.

Desmultiplexor

Un **desmultiplexor** fa la funció inversa d'un multiplexor: donat un senyal d'entrada, determina amb quin senyal de sortida s'ha de connectar.

Els desmultiplexors tenen els senyals següents:

Per especificar la mida d'un desmultiplexor, direm que és un desmultiplexor $1–2^m$; per exemple, un demultiplexor 1-4 té quatre sortides de dades i dues entrades de control.

1. Una entrada de dades de *n* bits, *e* (*n* pot ser igual a 1).

2. 2^m sortides de dades de *n* bits, identificades per la lletra *s* i numerades des de 0 fins a $2^m - 1$. Direm que la sortida de dades numerada amb el 0 és la de menys pes, i la numerada amb el $2^m - 1$, la de més pes.

3. *m* entrades de control o de selecció, d'un bit, identificades per la lletra *c* i numerades des de 0 fins a $m - 1$. Direm que l'entrada de control numerada amb el 0 és la de menys pes, i la numerada amb $m - 1$, la de més pes.

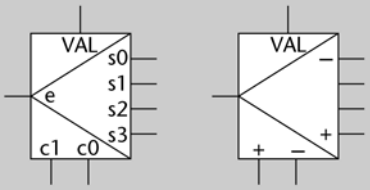
4. Una entrada de validació d'un bit, que anomenem *VAL*.

El funcionament del desmultiplexor és el següent:

- Quan l'entrada de validació val 0, totes les sortides valen 0.

- Quan l'entrada de validació val 1, llavors les entrades de control determinen quina de les sortides de dades es connecta amb l'entrada, de la mateixa manera que en el cas del multiplexor (si el nombre que codifiquen les entrades de control és *i*, l'entrada es connecta amb la sortida numerada amb el nombre *i*). Les sortides de dades no seleccionades es posen a 0.

La figura següent mostra les dues possibilitats per a la representació gràfica d'un desmultiplexor 1-4 i la taula que en descriu el funcionament. Igual que en el cas dels multiplexors, podem usar els signes més (+) i menys (–) per a indicar el pes de les entrades de control i de les sortides de dades. També, igual que en el cas dels multiplexors, en general no escriurem el nom de l'entrada *e*, i si no dibuixem l'entrada VAL assumirem que està activa.



VAL	c1	c0	s3	s2	s1	s0
0	x	x	0	0	0	0
1	0	0	0	0	0	e
1	0	1	0	0	e	0
1	1	0	0	e	0	0
1	1	1	e	0	0	0

Activitats

34. Es vol dissenyar un circuit que controli l'accés a una sala de concerts que té a l'entrada dos llums, un de verd i un de vermell, de manera que en cada moment només està encès un dels dos (el vermell si la sala és plena, el verd si encara hi cap gent). El circuit rep com a entrada un senyal ple que val 1 quan la sala està plena i 0 quan encara no. Per a encendre el llum verd s'ha d'activar el senyal verd, i per a encendre el vermell s'ha d'activar el senyal vermell. Implementeu el circuit usant només un desmultiplexor.

3.2. Codificadors i descodificadors

La funció d'un **codificador** és generar la codificació binària d'un nombre.

Els codificadors tenen els senyals següents:

1. Una entrada de validació, VAL, que funciona igual que en el cas dels multiplexors: si val 0, totes les sortides valen 0 (quan no dibuixem l'entrada de validació, assumirem que val 1).

2. 2^m entrades de dades (d'un bit), identificades per la lletra *e* i numerades de 0 a $2^m - 1$ (la de nombre més alt és la de més pes).

3. m sortides de dades (d'un bit), identificades per la lletra *s* i numerades de 0 a $m - 1$, que s'interpreten com si formessin un nombre codificat en binari amb m bits (la sortida de més pes correspon al bit de més pes).

El funcionament d'un codificador és el següent:

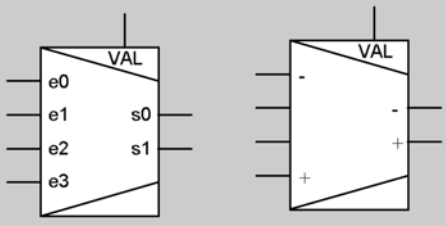
Quan l'entrada de validació val 1, si l'entrada de més pes d'entre les que són a 1 és la numerada amb el número *i*, llavors les sortides codifiquen en binari el número *i*. O, dit d'una altra manera: perquè un codificador generi la representació binària del nombre *i*, cal que l'entrada activa de més pes sigui la numerada amb el nombre *i*.

La figura 21 mostra la representació gràfica d'un codificador 4-2 i la taula de veritat que explica el seu funcionament (observem que, com en el cas dels multiplexors, podem fer servir els símbols "+" i "–" en lloc dels noms de les entrades i sortides). Hi veiem que, per exemple, quan $e3 = 0$ i $e2 = 1$, les sortides

Per a especificar la mida d'un codificador, direm que és un codificador $2^m - m$; per exemple, un codificador 4-2 és un codificador de quatre entrades i dues sortides.

codifiquen el nombre 2, independentment del valor de $e1$ i $e0$, perquè l'entrada de més pes que està activa és $e2$. Es diu que, com més pes té una entrada, més *prioritat* té.

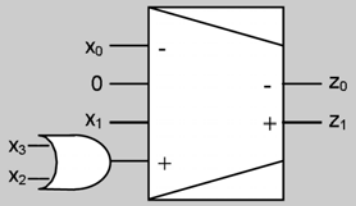
Figura 21



VAL	e3	e2	e1	e0	s1	s0
0	x	x	x	x	0	0
1	0	0	0	0	0	0
1	0	0	0	1	0	0
1	0	0	1	x	0	1
1	0	1	x	x	1	0
1	1	x	x	x	1	1

Fixem-nos que les sortides del codificador valen 0 tant si no hi ha cap entrada a 1 com si està a 1 l'entrada de menys pes (i també quan l'entrada VAL és 0).

La figura següent mostra un exemple d'ús d'un codificador i la taula de veritat que descriu el funcionament del circuit.



x_3	x_2	x_1	x_0	z_1	z_0
0	0	0	0	0	0
0	0	0	1	0	0
0	0	1	x	1	0
0	1	x	x	1	1
1	x	x	x	1	1

Recordem que si en un codificador no hi dibuixem l'entrada de validació, assumirem que aquesta està a 1.

Els **descodificadors** fan la funció inversa als codificadors: donada una combinació binària present en l'entrada, indiquen a quin nombre decimal correspon.

Els descodificadors tenen els senyals següents:

1. Una entrada de validació.
2. m entrades de dades, identificades per la lletra e i numerades de 0 a $m - 1$, que s'interpreten com si formessin un nombre codificat en binari (l'entrada de més pes correspon al bit de més pes).
3. 2^m sortides, identificades per la lletra s i numerades de 0 a $2^m - 1$, de les quals només una val 1 en cada moment (si l'entrada de validació està a 0, llavors totes les sortides valen 0).

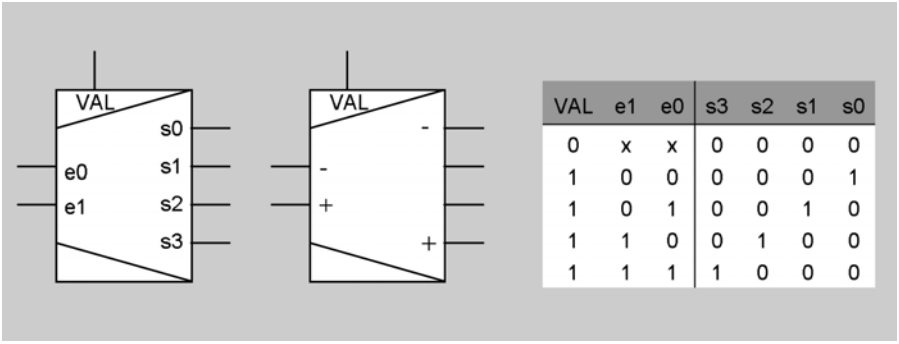
Per a identificar la mida dels descodificadors farem servir la mateixa convenció que en el cas dels codificadors; per exemple, descodificador 3-8.

El funcionament d'un descodificador és el següent:

Quan l'entrada de validació val 1, si les entrades codifiquen en binari el número i, llavors es posa a 1 la sortida numerada com a i.

La figura 22 mostra la representació gràfica d'un descodificador 2-4 i la taula de veritat que descriu el seu funcionament.

Figura 22



VAL	e1	e0	s3	s2	s1	s0
0	x	x	0	0	0	0
1	0	0	0	0	0	1
1	0	1	0	0	1	0
1	1	0	0	1	0	0
1	1	1	1	0	0	0

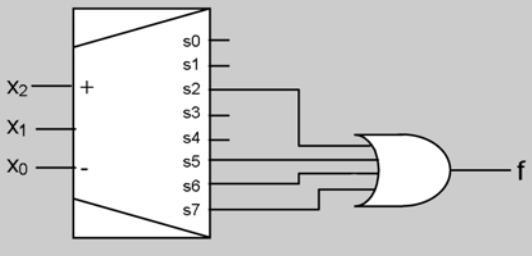
Els descodificadors també es poden fer servir per a implementar funcions lògiques. Si la funció té n variables, usarem un descodificador $n - 2^n$, i connectarem les variables a les entrades en ordre de pes. D'aquesta manera, la sortida i del descodificador es posarà a 1 quan les variables, interpretades com a bits d'un nombre en binari, codifiquin el número i. Per exemple, si connectem les variables $[x_1\ x_0]$ a les entrades $[e1\ e0]$ del descodificador de la figura 22, la sortida $s2$ valdrà 1 quan $[x_1\ x_0] = [1\ 0]$.

Per tant, per a implementar la funció n'hi haurà prou de connectar les sortides corresponents a les combinacions que fan que la funció valgui 1 a una porta OR. Quan alguna d'aquestes combinacions estigui present a l'entrada del descodificador, la sortida corresponent es posarà a 1 i, per tant, de la porta OR sortirà un 1. Quan les variables construeixin una combinació que faci que la funció valgui 0, totes les entrades de la porta OR valdran 0 i, per tant, també la seva sortida.

La figura 23 en mostra un exemple. Podem comprovar que la sortida de la porta OR valdrà 1 quan valgui 1 la sortida $s2$ del descodificador, o la $s5$, la $s6$ o la $s7$. És a dir, quan les variables d'entrada construeixin alguna de les combinacions que facin que la funció valgui 1.

Figura 23

x_2	x_1	x_0	f
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1



Si no dibuixem l'entrada de validació en un descodificador, assumirem que aquesta està a 1.

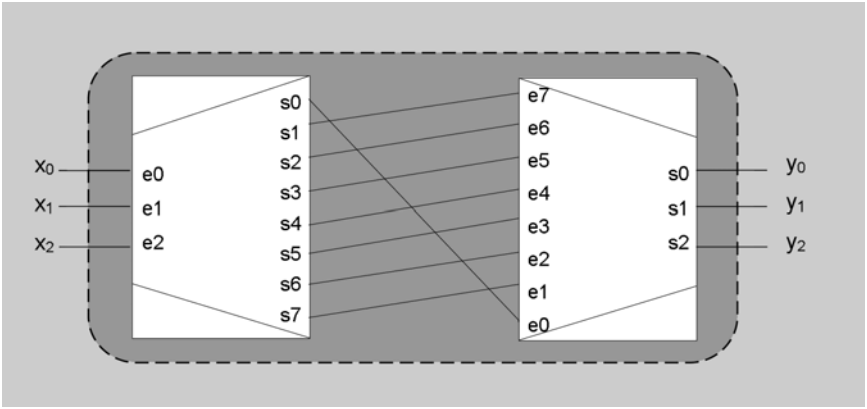
Quan dissenyem un circuit usant blocs, podem deixar una o més sortides dels blocs sense connectar enlloc. Però, no podem deixar **cap** entrada sense connectar, perquè altrament el comportament del circuit seria indeterminat.

Activitats

35. Deduïu la implementació interna (mitjançant portes lògiques) d'un codificador 4-2.
36. Deduïu la implementació interna (mitjançant portes lògiques) d'un descodificador 2-4.
37. Implementeu la funció $f(a, b, c) = abc' + a'c$ amb un descodificador 3-8.
38. Dissenyeu un circuit que generi la representació en signe i magnitud de nombres en el rang [–7, 7]. El circuit ha de tenir aquestes entrades i sortides:

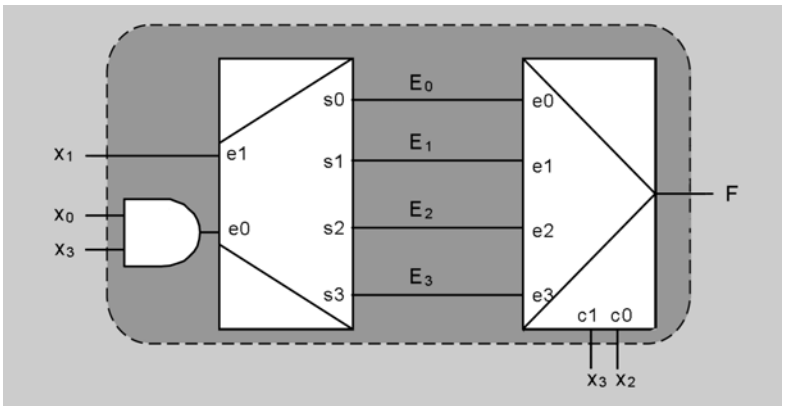
- Vuit entrades d'un bit, $e7$, $e6$, ..., $e0$, de les quals com a molt una estarà activa en cada moment. Si la que està a 1 és ei, la magnitud del nombre que s'ha de representar és i.
- Una altra entrada d'un bit sg, que indica el signe del nombre que s'ha de representar (0 positiu, 1 negatiu).
- Quatre sortides d'un bit, $s3$, ..., $s0$, que contindran la representació en signe i magnitud del nombre que es vol representar. El nombre 0 es representarà sempre amb el bit de signe a 0.
- Una altra sortida d'un bit, *null*, què valdrà 1 només quan no s'indiqui cap magnitud per ser representada. En aquest cas, les sortides $s3$, ..., $s0$ també valdran 0.

39. Què fa el circuit següent suposant que $X = [x_2\, x_1\, x_0]$ i $Y = [y_2\, y_1\, y_0]$ són nombres enters codificats en complement a 2?



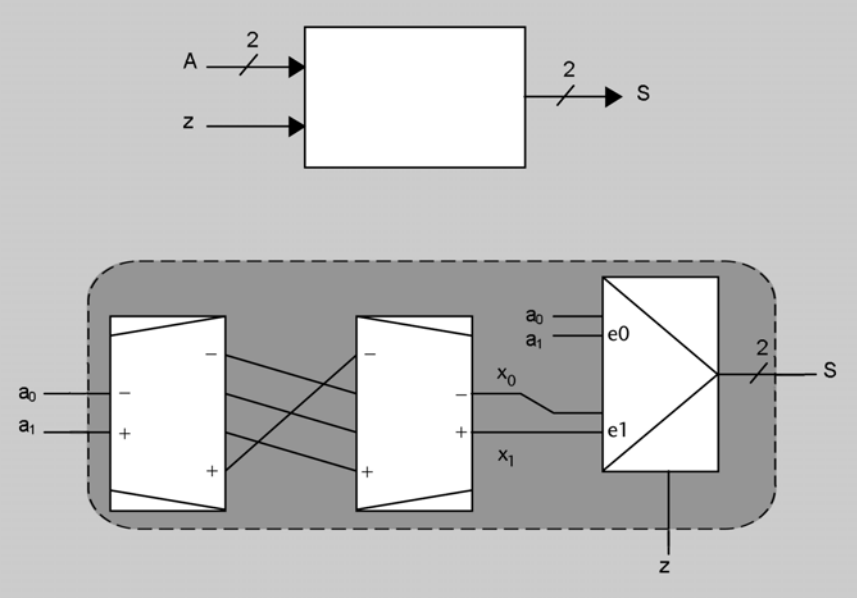
40. Construïu un circuit que implementi la funció $Y = (X + 3) \bmod 4$, en què X i Y són nombres naturals representats en binari amb 2 bits. El circuit només pot contenir dos blocs i cap porta.

41. Donat el circuit següent:



a) Trobeu les expressions algebraiques de E_i (i = 0, 1, 2, 3) en funció de x_3, x_1 i x_0.
b) Escriviu la taula de veritat de $F(x_3, x_2, x_1, x_0)$.

42. Dissenyeu un descodificador 3-8 utilitzant dos descodificadors 2-4 i les portes lògiques que facin falta.
43. Quina funció fa el circuit següent si interpretem les entrades i la sortida com a nombres codificats en binari natural?



3.3. Decaladors lògics i aritmètics

Un **decalador** és un bloc combinacional que té la funció de desplaçar bits cap a l'esquerra o cap a la dreta.

Els decaladors tenen un senyal d'entrada i un senyal de sortida, tots dos de n bits. El senyal de sortida s'obté desplaçant els bits d'entrada m vegades cap a la dreta o cap a l'esquerra.

1. Si el desplaçament és cap a l'esquerra, els m bits de menys pes de la sortida es posen a 0.

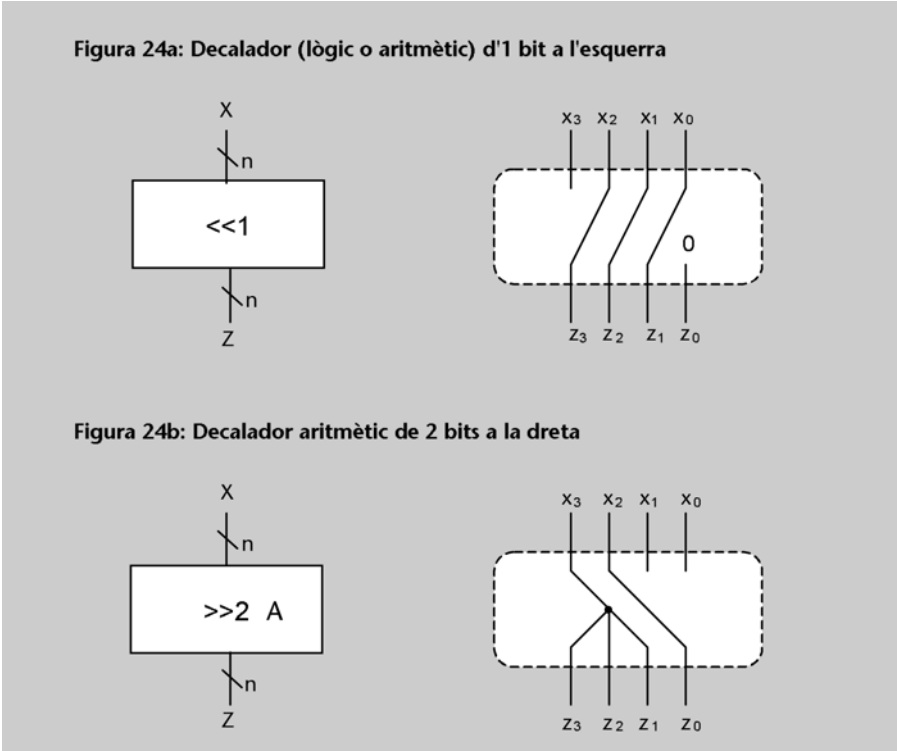
2. Si el desplaçament és cap a la dreta, hi ha dues possibilitats per als m bits de més pes de la sortida:

– En els **decaladors lògics** es posen a 0.

– En els **decaladors aritmètics** prenen el valor del bit de més pes de l'entrada. Fixem-nos que, si interpretem les entrades i sortides com a nombres codificats en complement a 2 amb n bits, els decaladors aritmètics mantenen a la sortida el signe de l'entrada.

Les figures 24*a* i 24*b* mostren la representació gràfica i la implementació interna d'un decalador d'un bit a l'esquerra i d'un decalador aritmètic de dos bits a la dreta, respectivament, de senyals de quatre bits. En el cas dels decaladors a la dreta, usarem la lletra L per a indicar que són lògics i la lletra A per a indicar que són aritmètics.

Figura 24



Si interpretem les entrades i sortides com a codificacions de nombres naturals, podem dir que els decaladors fan les funcions de multiplicar i dividir per potències de 2 (divisió entera). Un decalador de m bits a l'esquerra multiplica per 2^m, i un decalador lògic de m bits a la dreta fa la divisió entera per 2^m. Si interpretem els nombres com a codificats en complement a 2, aleshores per a dividir per 2^m cal usar decaladors aritmètics.

Activitats

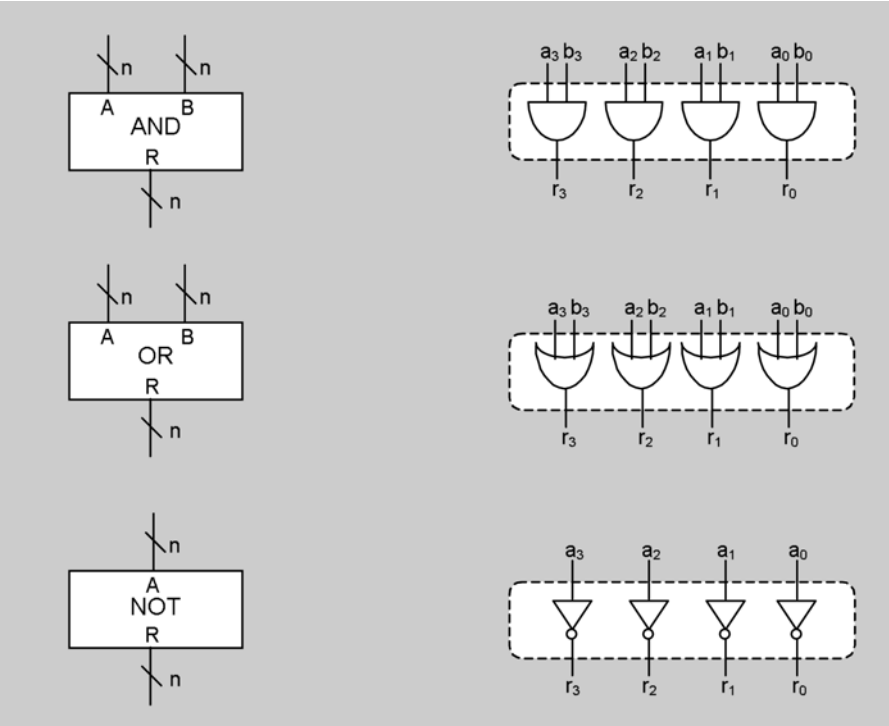
44. Contesteu els apartats següents:
a) Implementeu un circuit que pugui actuar com un decalador d'un bit a l'esquerra de senyals de quatre bits usant un multiplexor 2-1 de busos de quatre bits. Un senyal de control d'un bit, *d*, determina si el número d'entrada s'ha de decalar o no.
b) A quina operació aritmètica equival aquest desplaçament?
c) Indiqueu quan es produiria sobreeiximent en els casos d'interpretar el senyal d'entrada com un nombre natural codificat en binari o bé com un nombre enter codificat en complement a 2.

3.4. Blocs AND, OR i NOT

Aquests blocs fan les operacions AND, OR i NOT bit a bit sobre entrades de n bits.

Tenen dues entrades de dades (només una en el cas del bloc NOT) i una sortida, totes de n bits. La figura 25 mostra la seva representació gràfica i la implementació interna per al cas $n = 4$.

Figura 25



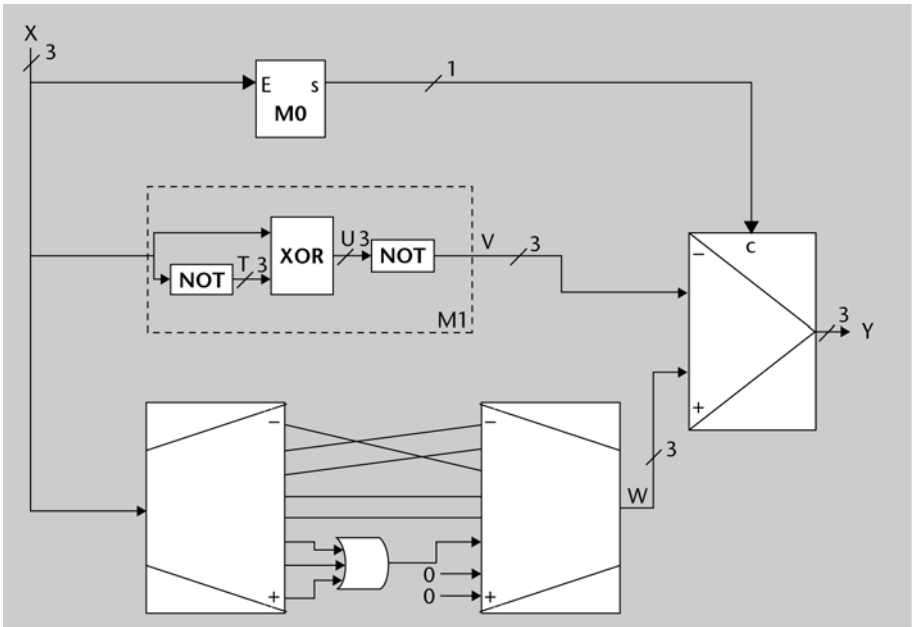
Podem construir blocs anàlegs a aquests, per exemple un bloc XOR seria un bloc que fa la XOR bit a bit de les entrades.

Activitats

45. Dissenyeu un circuit amb una entrada X per la qual arriben nombres naturals representats en binari amb 8 bits i una sortida Z de 8 bits, de manera que $Z = X'$ si X és múltiple de 4, i $Z = x_3x_200$ si no ho és.

46. En el circuit lògic combinacional següent, l'entrada X i la sortida Y codifiquen nombres naturals en binari i 3 bits i el bloc $M0$ té aquest comportament:

$$s = x_2x_1x_0 + x_2x_1x_0' + x_2x_1'x_0' + x_2x_1'x_0 + x_2'x_1x_0$$



a) Implementeu el bloc *M*0 amb el mínim nombre de blocs combinacionals i, si cal, portes lògiques.
b) Quina funció fa el subcircuit identificat com a *M*1?
c) Escriviu la taula de veritat del circuit complet.

3.5. Memòria ROM

La **memòria ROM** és un bloc combinacional que permet de guardar el valor de 2^m mots de n bits.

ROM

La denominació ROM deriva de l'anglès *Read Only Memory*, perquè es refereix a memòries en les quals no es poden fer escriptures, sinó només lectures.

Podem veure una memòria ROM com un arxivador amb calaixos que guarden bits. Cada calaix té capacitat per a guardar un cert nombre de bits (tots tenen la mateixa), i l'arxivador té un nombre determinat de calaixos, que és sempre una potència de dos.

La memòria ROM té els elements següents:

1. 2^m mots o dades de n bits, cadascuna en una posició (calaix) diferent de la memòria ROM. Les posicions que contenen les dades estan numerades des del 0 fins al $2^m - 1$, i aquests nombres s'anomenen **adreces**.

2. Una entrada d'adreces de m bits, que s'identifica amb símbol @. Els m bits de l'entrada d'adreces s'interpreten com a nombres codificats en binari (i, per tant, cal determinar quin és el pes de cada bit).

3. Una sortida de dades de n bits.

El funcionament de la ROM és el següent:

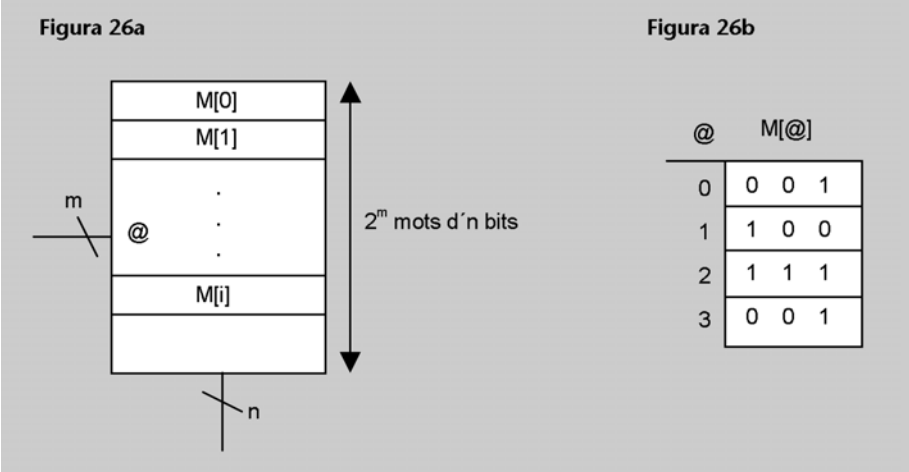
Quan els m bits de l'entrada d'adreces (interpretats en binari) codifiquen el número i, llavors la sortida pren el valor de la dada que hi ha emmagatzemada a l'adreça i. Per a referir-nos a aquesta dada usarem la notació **M[*i*]**, i direm que **llegim** la dada de l'adreça i.

Així, doncs, només es pot accedir al valor d'un mot (llegir-lo) en cada instant (és com si en cada moment només es pogués obrir un calaix).

La figura 26*a* mostra de quina manera representarem la memòria ROM. La figura 26*b* mostra un possible contingut d'una memòria ROM de quatre mots de tres bits. En aquest exemple,

M[0] = 001
M[1] = 100
M[2] = 111
M[3] = 001.

Figura 26

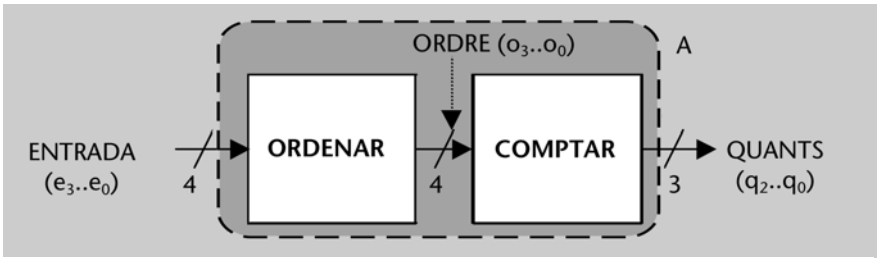


La memòria ROM també es pot fer servir per a sintetitzar funcions lògiques. Per exemple, si connectem les variables x_1 i x_0 (en ordre de pes) a les entrades d'adreces de la ROM de la figura 26*b*, llavors podem interpretar els tres bits de sortida com la implementació de les tres funcions següents:

x_1	x_0	f_2	f_1	f_0
0	0	0	0	1
0	1	1	0	0
1	0	1	1	1
1	1	0	0	1

Activitats

47. Indiqueu la grandària i el contingut d'una memòria ROM que implementi la funció descrita en l'activitat 24.
48. Indiqueu la grandària i el contingut d'una memòria ROM que implementi la funció descrita en l'activitat 19.
49. Utilitzant només una memòria ROM, es vol dissenyar un circuit que, donat un nombre *X* representat en signe i magnitud amb 8 bits, doni a la sortida el mateix nombre representat en complement a 2 i 8 bits. Descriviu el contingut que ha de tenir la memòria ROM (sense escriure tot el contingut) i indiqueu la grandària mínima que ha de tenir. Escriviu el contingut dels mots de les adreces 50, 100 i 150.
50. El circuit combinacional de la figura compta el nombre d'uns que té un mot d'entrada de quatre bits.



El bloc ORDENAR ordena el mot ENTRADA, i col·loca tots els uns a la dreta i tots els zeros a l'esquerra. Per exemple:

Si ENTRADA = 0101, llavors ORDRE = 0011.
Si ENTRADA = 0000, llavors ORDRE = 0000.

El bloc COMPTAR genera a la sortida QUANTS la codificació binària de la quantitat d'uns del mot ORDRE. Per exemple:

Si ORDRE = 0011, llavors QUANTS = 010 (2).
Si ORDRE = 0000, llavors QUANTS = 000 (0).

Responeu els apartats següents:

a) Feu la taula de veritat del bloc ORDENAR.
b) Dissenyeu el bloc COMPTAR utilitzant només blocs combinacionals (exceptuant memòria ROM), de la grandària que faci falta.
c) Dissenyeu el circuit combinacional complet A amb una memòria ROM, i indiqueu-ne la grandària i el contingut.

3.6. Comparador

El **comparador** és un bloc combinacional que compara dos nombres codificats en binari i indica quina relació hi ha entre aquests.

Un comparador té els senyals següents:

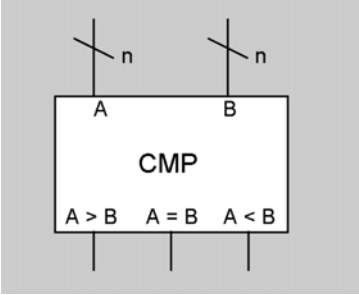
1. Dues entrades de dades de n bits, que reben els noms A i B. S'interpreten com a nombres naturals codificats en binari.

2. Tres sortides d'un bit, de les quals només una val 1 en cada moment:

- La sortida $A > B$ val 1 si el número que arriba per l'entrada A és més gran que el que arriba per l'entrada B.
- La sortida $A = B$ val 1 si els dos números d'entrada són iguals.
- la sortida $A < B$ val 1 si el número que arriba per l'entrada A és més petit que el que arriba per l'entrada B.

La figura 27 mostra la representació gràfica d'un comparador.

Figura 27



Activitats

51. Deduïu les expressions algebraiques de les funcions de sortida $A = B$ i $A < B$ d'un comparador de nombres de dos bits i feu-ne la implementació mitjançant portes lògiques.

52. Dissenyeu un circuit que obtingui en la seva sortida el màxim de dos nombres naturals, X i Y, de quatre bits.

53. Utilitzant qualssevol dels blocs i portes que s'han vist, dissenyeu un circuit amb dues entrades X i Y que codifiquen nombres naturals en binari i 3 bits i una sortida S de 2 bits, que funcioni de la manera següent:

- Si $X > Y$, S ha valer 01.
- Si $X < Y$, S ha valer 10.
- Si $X = Y$ i són diferents de 0, S ha valer 00.
- Si $X = Y$ i són iguals a 0, S ha valer 11.

54. Es disposa de dos forns, cadascun dels quals està equipat amb un termòmetre digital per a mesurar la temperatura interior. Els termòmetres generen un senyal de 8 bits que codifica la temperatura en binari (els forns sempre estan entre 0° C i 255° C).

Utilitzant qualssevol dels blocs i portes que s'han vist, dissenyeu un circuit lògic combinacional que, rebent per les entrades *A* i *B* la temperatura dels dos forns, informi mitjançant la sortida *R*, de 2 bits, en quin rang de temperatures es troba el forn que està més calent de tots dos; si tots dos estan a la mateixa temperatura, ha d'indicar en quin rang es troba aquesta. La correspondència entre rangs de temperatures i valor del senyal *R* és com segueix:

Temperatura del forn que està més calent (o igual)	*R*
[11000000, 11111111]	11
[10000000, 10111111]	10
[01000000, 01111111]	01
[00000000, 00111111]	00

3.7. Sumador

El **sumador** és un bloc combinacional que fa la suma de dos nombres codificats en binari o bé en complement a 2.

Recordeu

Tal com s'ha estudiat en el mòdul "Representació de la informació numèrica", la representació dels enters en complement a 2 permet de fer les sumes fent servir la mateixa mecànica que en les sumes de nombres codificats en binari. Per tant, si un bloc realitza la suma de nombres codificats en binari, la seva sortida també serà la suma correcta si interpretem els nombres d'entrada com a enters codificats en complement a 2. Per això diem que el sumador suma nombres codificats en binari o bé en complement a 2.

La figura 28 mostra la representació gràfica d'un sumador de *n* bits. Els senyals que té són els següents:

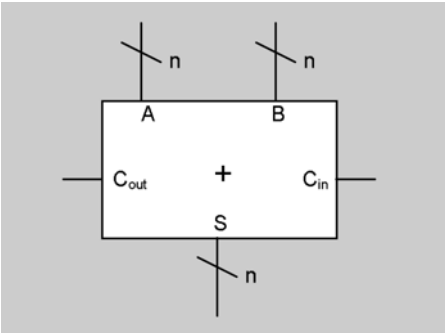
1. Dues entrades de dades de *n* bits, que anomenarem *A* i *B*, per on arribaran els nombres que s'han de sumar.

2. Una sortida de *n* bits, anomenada *S*, que prendrà el valor de la suma dels números *A* i *B*.

3. Una sortida d'un bit, C_{out}, que valdrà 1 si quan es fa la suma es produeix ròssec en el bit de més pes.

4. Una entrada d'un bit, C_{in}, per on arriba un ròssec d'entrada.

Figura 28



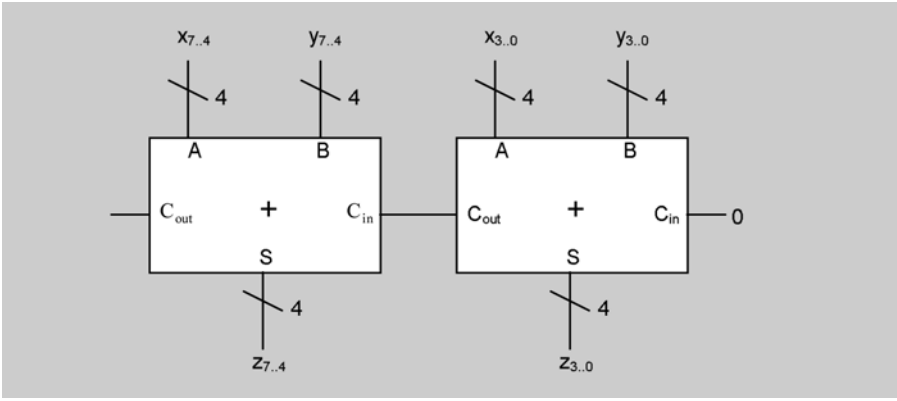
La nomenclatura dels senyals d'entrada i sortida de ròssec (*C*) deriva de la paraula *Carry*, que és el mot anglès que s'utilitza per a ròssec.

L'entrada C_{in} és útil quan es volen sumar nombres de $2 \cdot n$ bits i només es disposa de sumadors de *n* bits. En aquest cas s'encadenen dos sumadors: el primer suma

els n bits més baixos dels nombres i el segon els n bits més alts. La sortida C_{out} del primer sumador es connecta amb l'entrada C_{in} del segon, per tal que el resultat sigui correcte. La figura 29 mostra un exemple per al cas $n = 4$, en què fem la suma $Z = X + Y$ essent X, Y i Z nombres de 8 bits. Fixem-nos que el sumador que suma els bits més baixos el dibuixem a la dreta, perquè així els bits queden ordenats de la manera en què estem acostumats a veure'ls.

Si no necessitem tenir en compte un ròssec d'entrada, connectarem un 0 a l'entrada C_{in}.

Figura 29



Com ja sabem, el fet de limitar la longitud dels nombres a un determinat nombre de bits (n) té com a conseqüència que el resultat d'una suma no sigui sempre correcte (és incorrecte quan es produeix sobreeiximent, és a dir, quan el resultat requereix més de n bits per a ser codificat). En les sumes binàries, sabem que si es produeix ròssec en el bit de més pes llavors el resultat és incorrecte. En canvi, en les sumes en complement a 2 no hi ha cap relació entre el ròssec i el sobreeiximent.

Recordeu

El concepte de sobreeiximent que s'ha estudiat en el mòdul "Representació de la informació numèrica".

Per tant, la sortida C_{out} d'un sumador ens indica que s'ha produït sobreeiximent si interpretem les entrades en binari, però no ens diu res sobre la correctesa del resultat si les interpretem en complement a 2.

Atès que $X - Y = X + (-Y)$, els sumadors es poden utilitzar també per a fer la resta de dos nombres, si canviem el signe del segon operand abans de connectar-lo al sumador.

Recordeu

L'operació de canvi de signe que s'ha estudiat en el mòdul "Representació de la informació numèrica".

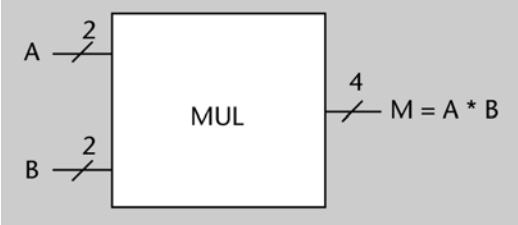
Activitats

55. Suposeu que X és un nombre natural codificat amb tres bits, i implementeu la funció $Z = (3 \cdot X) \bmod 8$ usant només un sumador de quatre bits i un decalador.

56. Dissenyeu un circuit que faci l'operació $Z = X - Y$ essent X, Y i Z nombres codificats en complement a 2 amb n bits.

57. Usant blocs combinacionals, dissenyeu un circuit que, donat un nombre enter X codificat en complement a 2 i 3 bits, n'obtingui el valor absolut, també en complement a 2 i 3 bits. El circuit ha d'indicar si es produeix sobreeiximent en el càlcul posant la sortida V (d'un bit) a 1.

58. Es disposa d'un circuit combinacional que implementa un multiplicador de dos nombres enters, representats en complement a 2 amb dos bits. La sortida d'un circuit multiplicador de nombres enters sempre té el doble de bits que les entrades.



a) Escriviu la taula de veritat del bloc MUL.
b) Implementeu amb blocs combinacionals un circuit que calculi l'operació següent:

$$Z = A^4 + A^3 + A^2,$$

on A és un nombre enter de dos bits representat en complement a 2. Podeu usar blocs de qualsevol nombre de bits (especifiqueu de quants i justifiqueu la resposta), inclòs el de l'apartat *a*, que no cal dissenyar. Una possible solució només requereix un sumador i dos multiplicadors.

59. A, B, C i D són senyals de 8 bits que codifiquen nombres naturals. Es vol dissenyar un circuit que implementi la funció Y, que es descriu així:

$Y = 2 \times C$	si	$C = D$
$Y = (A + B + C) / 4$	si	$C \neq D$

Els busos d'entrada i sortida de cada bloc han de tenir la menor amplada possible per a aconseguir que en cap moment no es produeixin sobreeiximents.

3.8. Unitat aritmètica i lògica (UAL)

Una **unitat aritmètica i lògica** és un aparell capaç de fer un determinat conjunt d'operacions aritmètiques i lògiques sobre dos nombres d'entrada codificats en binari o en complement a 2.

Les unitats aritmètiques i lògiques s'anomenen *UAL* o, també, *ALU* (de l'anglès *Arithmetic and Logic Unit*).

Per a dissenyar una UAL cal especificar el conjunt d'operacions que volem que faci. Per exemple, una UAL pot fer la suma, la resta, l'AND i l'OR dels operands d'entrada. En cada moment, s'especificarà a la UAL quina és l'operació que ha de fer.

En aquest curs només estudiarem UAL que operin amb nombres naturals i enters. Els processadors tenen, a més, UAL per a operar amb nombres reals codificats en coma flotant.

Els senyals que té una UAL són els següents:

1. Dues entrades de dades de n bits, A i B, per on arribaran els nombres sobre els quals s'han de fer les operacions.

2. Una sortida de dades de n bits, R, on s'obtindrà el resultat de l'operació.

3. Un cert nombre d'entrades de control, c_i, d'un bit cadascuna. Si la UAL és capaç de fer 2^m operacions diferents, ha de tenir m entrades de control. Cada combinació de les entrades de control indicarà a la UAL que faci una operació concreta. Per exemple, la UAL de la figura 30 pot fer quatre operacions.

4. Un cert nombre de sortides d'un bit, que s'anomenen ***bits d'estat***, i tenen la funció d'indicar algunes circumstàncies que es poden haver produït durant el càlcul.

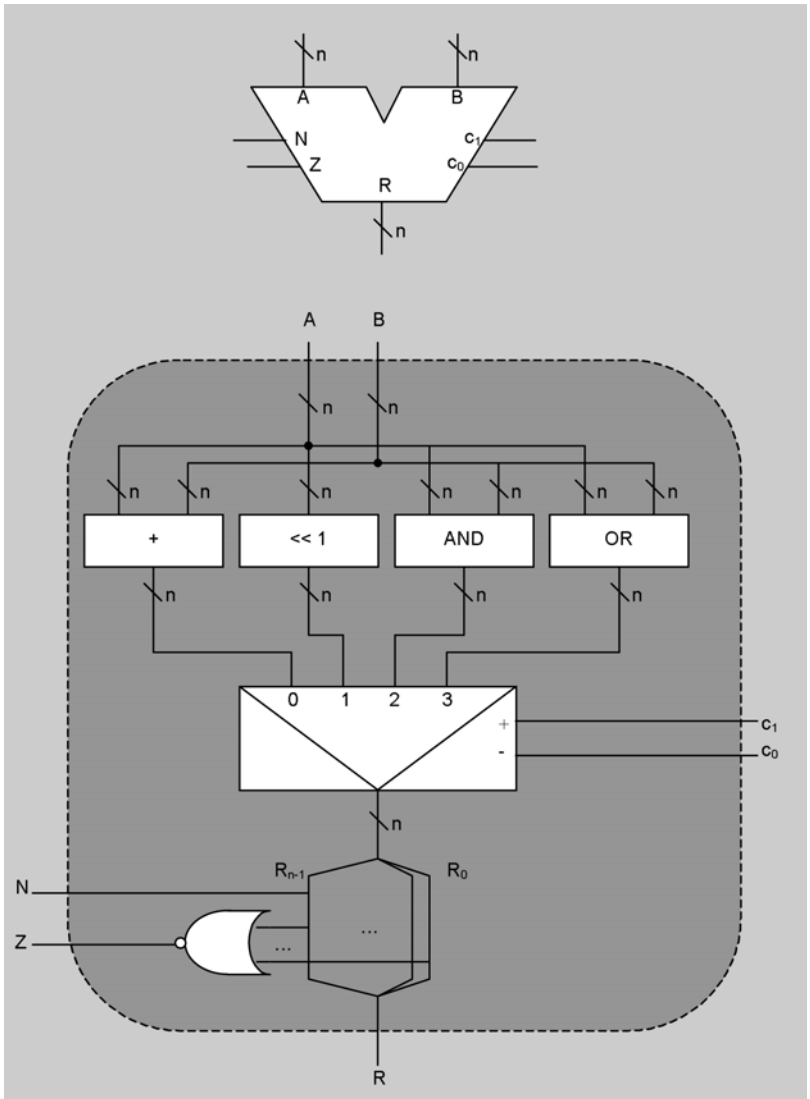
Els bits d'estat més habituals són els que indiquen si s'ha produït ròssec en l'últim bit (*C*), si s'ha produït sobreeiximent (*V*), si el resultat de l'operació ha estat negatiu (*N*) o si el resultat de l'operació ha estat 0 (*Z*).

El bit de sobreeiximent se sol identificar amb les lletres *O* o *V*, que deriven de la paraula anglesa ***overflow***, que significa 'desbordament'.

La figura 30 mostra la representació gràfica i la implementació interna d'una UAL que genera els bits d'estat *N* i *Z* i que fa les operacions següents:

c_1	c_0	*R*
0	0	*A* + *B*
0	1	2 * *A*
1	0	*a* AND *b*
1	1	*a* OR *b*

Figura 30



Nota

En aquesta figura hem disgregat els bits del bus corresponent al senyal de sortida *R* per a poder dibuixar la implementació dels bits *N* i *Z*. S'ha d'interpretar que tots els bits de *R* es connecten a la porta NOR (de *n* entrades) que computa *Z*.

Activitats

60. Contesteu els apartats següents:

a) Dissenyeu una UAL que, a partir de dues entrades de control c_1 i c_0, faci les operacions següents sobre dos números A i B de quatre bits:

- $[c_1\ c_0] = [0\ 0] : R = A + B$.
- $[c_1\ c_0] = [0\ 1] : R = A - B$.
- $[c_1\ c_0] = [1\ 0] : R = A$.
- $[c_1\ c_0] = [1\ 1] : R = -A$.

b) Afegiu a la UAL dissenyada en l'apartat anterior els circuits necessaris per a calcular els bits de condició següents:

- *Vb*: sobreeiximent si s'interpreta que els nombres d'entrada estan codificats en binari.
- *V*: sobreeiximent si s'interpreta que els nombres d'entrada estan codificats en complement a 2.
- *N*: $N = 1$ si el resultat de l'operació interpretat en complement a 2 és negatiu, i 0 en el cas contrari.
- *Z*: $Z = 1$ si el valor de la sortida és 0 , i $Z = 0$ en el cas contrari.

Resum

En aquest mòdul s'han estudiat els fonaments dels circuits electrònics digitals. S'ha vist que es construeixen a partir dels mateixos elements que la lògica natural, formalitzada matemàticament per l'àlgebra de Boole: els elements bàsics són els valors 0 i 1 i les operacions suma lògica, producte lògic i negació. A partir d'aquí s'han introduït les variables i funcions lògiques.

S'han conegut dues maneres de representar funcions lògiques: les expressions algebraiques i les taules de veritat. S'ha vist que són especialment còmodes les expressions en suma de productes.

Després s'ha vist de quina manera s'implementen físicament les operacions lògiques bàsiques per a construir circuits, mitjançant les portes lògiques. S'ha après la manera de minimitzar circuits a dos nivells, mitjançant el mètode de minimització de Karnaugh.

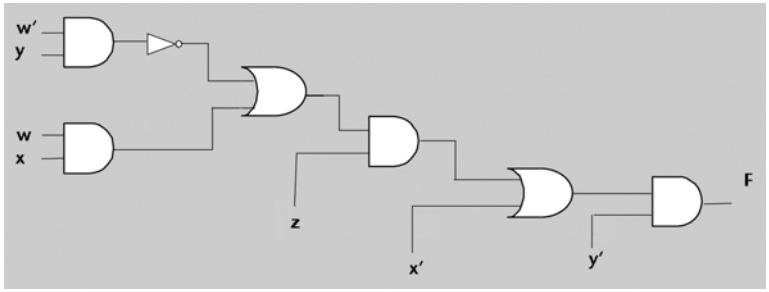
Finalment, s'han conegut diversos blocs combinacionals, i s'ha après a utilitzar la funcionalitat de cadascun d'aquests per a dissenyar i entendre circuits complexos.

Exercicis d'autoavaluació

Entre parèntesis s'indica l'apartat dels apunts que s'ha d'haver estudiat per tal de poder resoldre cada exercici.

1. (1.2) Demostreu que les lleis de De Morgan també es compleixen per a tres i quatre variables. N'hi ha prou de demostrar que $(xyz)' = x' + y' + z'$ i que $(xyzw)' = x' + y' + z' + w'$, perquè les altres lleis s'obtenen directament aplicant el principi de dualitat.

2. (2.1) Analitzeu el circuit combinacional següent i expresseu algebraicament en forma de suma de productes la funció que implementa:



3. (2.2) Dissenyeu a dos nivells un circuit combinacional que permeti de multiplicar dos nombres enters de dos bits representats en complement a 2.

4. (2.3) Sintetitzeu de manera mínima a dos nivells el circuit descrit en l'activitat 24.

5. (3.1) Dissenyeu un multiplexor 16-1 utilitzant únicament multiplexors 4-1 (no feu servir cap porta lògica addicional).

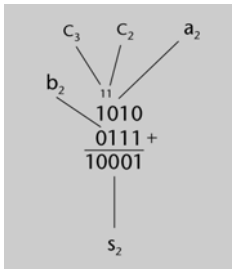
6. (3.2) Dissenyeu un descodificador 4-16 utilitzant únicament cinc descodificadors 2-4.

7. (3.2) Suposeu que X és un nombre natural codificat amb tres bits. Implementeu la funció $(3 \cdot X)$ mod 8 usant només un descodificador 3-8 i un codificador 8-3.

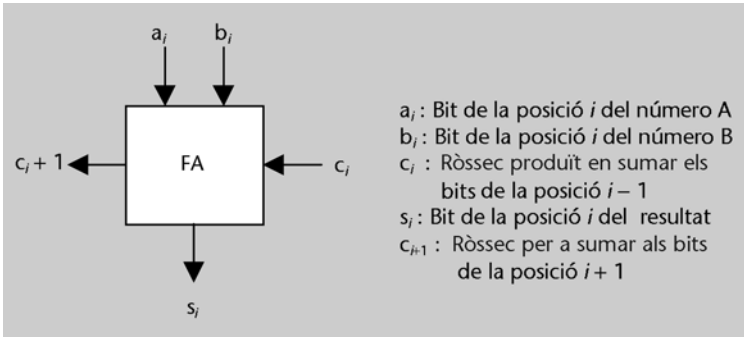
8. (3.3) Contesteu els apartats següents:
a) Implementeu un *decalador programable a l'esquerra* de nombres de vuit bits. El decalador té una entrada de control de tres bits, C, que indica el nombre de bits del decalatge.
b) Si C codifica en binari el valor n, a quina operació aritmètica equival aquest desplaçament?
c) Suposeu que $C = 010$ i indiqueu quan es produiria sobreeiximent en els casos d'interpretar l'entrada com un nombre natural o bé com un enter representat en complement a 2.
Suposem ara que volem fer l'operació $X / 2^n$ usant un decalador a la dreta.
d) Quin tipus de decalador hem de fer servir si X és un nombre natural representat en binari? I si és un nombre enter representat en complement a 2?
e) En quins casos el resultat de la divisió no és exacte?

9. (3.6) Sintetitzeu un comparador de nombres enters de quatre bits codificats en complement a 2, utilitzant comparadors de nombres naturals de quatre bits i les portes lògiques necessàries.

10.(3.7) Quan fem una suma de nombres binaris, la fem bit a bit (de la mateixa manera que en decimal la fem dígit a dígit). Per a obtenir el bit de la posició i necessitem conèixer els bits que són en la posició i en els dos operands i el ròssec que es genera en la suma dels bits de la posició $i - 1$.
La suma d'aquests tres bits dóna com a resultat dos bits: el bit s_i, que correspon al bit de la posició i de la suma, i el bit c_{i+1}, que és el ròssec que es genera per a la suma dels bits de la posició $i + 1$.
A continuació, es mostra un exemple per a $A = 1010$ i $B = 0111$. L'exemple mostra concretament la suma dels bits de la posició dos (recordeu que el bit de més a la dreta és el bit de la posició 0).



El sumador de nombres d'un bit té tres entrades (a_i, b_i i c_i) i dues sortides (s_i i c_{i+1}). Aquest sumador rep el nom de *full adder* (sumador complet), i s'abreuja FA. La figura següent en mostra l'estructura.



a) Escriviu la taula de veritat i feu la implementació interna (mitjançant portes lògiques) d'un FA com el descrit.

b) Utilitzeu quatre FA per tal de construir un sumador de dos nombres de quatre bits. Què s'ha de connectar a l'entrada que correspon al bit c_0?

11. (3.7) Dissenyeu un circuit que sigui capaç de sumar o restar dos nombres codificats en complement a 2 amb quatre bits d'acord amb el que valgui el senyal d'entrada *s'/r*: si val 0, el circuit ha de fer la suma, i si val 1, ha de fer la resta. El circuit ha de generar a més un senyal de sortida *C* que indiqui si s'ha produït ròssec en l'últim bit.

12.(3.7) Volem dissenyar un circuit que controli els ascensors d'un edifici de cinc plantes i planta baixa. L'edifici té dos ascensors, *A* i *B*, cadascun amb dos senyals associats:

- Un senyal que indica en binari a quina planta és l'ascensor en cada moment (*plantaA* i *plantaB*).
- Un senyal d'activació (*actA*, *actB*).

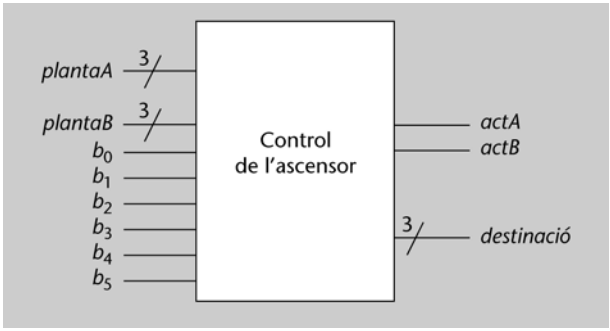
Quan un senyal d'activació es posa a 1, l'ascensor corresponent es posa en moviment; el senyal *destinació* li indica, en binari (nombre natural), cap a quina planta ha d'anar. Després que l'ascensor es posi en moviment, el senyal d'activació torna a 0 automàticament (no cal que ens preocupem de fer-ho).

A cada planta *i* (i = 0, 1, ..., 5) hi ha un botó per a cridar l'ascensor, connectat al senyal b_i. Quan algú el crida des de la planta *i*-èssima, el senyal b_i es posa a 1 i torna a 0 quan l'ascensor ha arribat a la planta corresponent.

Mentre no hi hagi cap crida els ascensors han de continuar aturats. Quan algú cridi l'ascensor des d'una planta, s'hi ha de moure el que estigui més a prop. Si tots dos estan a la mateixa distància, hi anirà l'ascensor *A*.

Per facilitar el disseny suposarem algunes simplificacions en el sistema:

- Mai no es premerà més d'un botó de crida alhora.
- En el moment de prémer un botó de crida tots dos ascensors estan aturats.



Dissenyeu el circuit de control dels ascensors fent servir els blocs i portes que facin falta, i tenint cura de donar a tots els busos l'amplada mínima necessària. Si voleu podeu fer servir,

com a blocs, els circuits que es dissenyen en les activitats 56 i 57, dimensionant l'amplada de les entrades i sortides segons us calgui.

13. (3.8) Dissenyeu una UAL amb sortida R de quatre bits que, a partir de tres entrades de control c_2, c_1 i c_0, faci les operacions següents sobre dos números A i B de quatre bits:

- $[c_2\, c_1\, c_0] = [0\ x\ x] : R = B$.
- $[c_2\, c_1\, c_0] = [1\ 0\ 0] : R = A + B$.
- $[c_2\, c_1\, c_0] = [1\ 0\ 1] : R = A - B$.
- $[c_2\, c_1\, c_0] = [1\ 1\ 0] : R = B >> 1$.
- $[c_2\, c_1\, c_0] = [1\ 1\ 1] : R = A$ AND B.

Solucionari

Activitats

1. Per a avaluar l'expressió $x + y \cdot (z + x')$ substituirem les variables per cada combinació de valors:

Per al cas $[x\ y\ z] = [0\ 1\ 0]$, tenim el següent:

$$0 + 1 \cdot (0 + 1) = 0 + 1 \cdot 1 = 0 + 1 = 1.$$

Per al cas $[x\ y\ z] = [1\ 1\ 0]$, tenim el següent:

$$1 + 1 \cdot (0 + 0) = 1 + 1 \cdot 0 = 1 + 0 = 1.$$

Per al cas $[x\ y\ z] = [0\ 1\ 1]$, tenim el següent:

$$0 + 1 \cdot (1 + 1) = 0 + 1 \cdot 1 = 0 + 1 = 1.$$

2. Per a fer la demostració, substituirem les variables d'entrada per totes les combinacions de valors que puguin prendre. Les igualtats s'han de complir per a tots els casos.

- Igualtat $(x + y)' = x' \cdot y'$
  - Cas $x = 0$ i $y = 0$:
    $(x + y)' = (0 + 0)' = 0' = 1$,
    $x' \cdot y' = 0' \cdot 0' = 1 \cdot 1 = 1$.
  - Cas $x = 0$ i $y = 1$:
    $(x + y)' = (0 + 1)' = 1' = 0$,
    $x' \cdot y' = 0' \cdot 1' = 1 \cdot 0 = 0$.
  - Cas $x = 1$ i $y = 0$:
    $(x + y)' = (1 + 0)' = 1' = 0$,
    $x' \cdot y' = 1' \cdot 0' = 0 \cdot 1 = 0$.
  - Cas $x = 1$ i $y = 1$:
    $(x + y)' = (1 + 1)' = 1' = 0$,
    $x' \cdot y' = 1' \cdot 1' = 0 \cdot 0 = 0$.

Com que les dues expressions valen el mateix per a tots els casos possibles, concloem que són equivalents.

- Igualtat $(x \cdot y)' = x' + y'$
  - Cas $x = 0$ i $y = 0$:
    $(x \cdot y)' = (0 \cdot 0)' = 0' = 1$,
    $x' + y' = 0' + 0' = 1 + 1 = 1$.
  - Cas $x = 0$ i $y = 1$:
    $(x \cdot y)' = (0 \cdot 1)' = 0' = 1$,
    $x' + y' = 0' + 1' = 1 + 0 = 1$.
  - Cas $x = 1$ i $y = 0$:
    $(x \cdot y)' = (1 \cdot 0)' = 0' = 1$,
    $x' + y' = 1' + 0' = 0 + 1 = 1$.
  - Cas $x = 1$ i $y = 1$:
    $(x \cdot y)' = (1 \cdot 1)' = 1' = 0$,
    $x' + y' = 1' + 1' = 0 + 0 = 0$.

Com que les dues expressions valen el mateix per a tots els casos possibles, concloem que són equivalents.

3. La llei 4 de l'àlgebra de Boole diu que $x + 1 = 1$ i que $x \cdot 0 = 0$.

Demostrarem només la primera igualtat; la segona quedarà demostrada pel principi de dualitat.

L'axioma *e* diu que $x + x' = 1$. Per tant,

$$x + 1 = x + x + x'.$$

La llei 2 (d'idempotència) diu que $x + x = x$. Per tant,

$$x + x + x' = x + x' = 1.$$

Hem obtingut l'última igualtat aplicant de nou l'axioma *e*. Per tant, $x + 1 = 1$, tal com volíem demostrar.

4.

- La funció g_1 val 1 només quan x val 1 i y val 1. Per tant, una possible expressió lògica per aquesta funció és $g_1(x, y) = x \cdot y$.
- La funció g_3 val 1 quan x val 1 i y val 0, o bé quan x val 1 i y val 1. Una possible expressió per la funció és, per tant, $x \cdot y' + x \cdot y$.
  També podem veure que la funció val 1 quan la variable x val 1, independentment del valor que prengui la variable y. Per tant, una possible expressió lògica per a aquesta funció és $g_3(x, y) = x$.
- La funció g_6 val 1 quan x val 1 i y val 0, o bé quan x val 0 i y val 1. Per tant, una altra possible expressió lògica per a aquesta funció és $g_6(x, y) = xy' + x'y$.
- La funció g_7 val 1 quan x val 0 i y val 1, quan x val 1 i y val 0, o bé quan x val 1 i y val 1. Per tant, una possible expressió lògica per aquesta funció és $g_7(x, y) = x'y + xy' + xy$.
  També podem dir que la funció val 1 sempre que no es compleixi que $x = 0$ i $y = 0$. Per tant, una altra expressió per a la funció és $g_7(x, y) = (x'y')'$. Si apliquem la segona llei de De Morgan sobre aquesta expressió, i després la llei d'involució, obtenim l'expressió $g_7(x, y) = x'' + y'' = x + y$.
- La funció g_{10} val 1 quan x val 0 i y val 0, o bé quan x val 1 i y val 0. És a dir, la funció val 1 quan la variable y val 0, independentment del valor de x. Per tant, la funció val el contrari del que valgui y, i una possible expressió lògica per a aquesta funció és $g_{10}(x, y) = y'$.

5. Tenim l'expressió:

$$wx + xy' + yz + xz' + xy.$$

Per a simplificar-la aplicarem la propietat distributiva. Agrupem el segon terme amb l'últim, i obtenim el següent:

$$wx + x(y' + y) + yz + xz'.$$

Per l'axioma de complementació sabem que $y + y' = 1$. Si fem la substitució en l'expressió i apliquem l'axioma dels elements neutres, obtenim el següent:

$$w \cdot x + x \cdot 1 + yz + xz' = wx + x + yz + xz'.$$

Per la llei d'absorció, $wx + x = x$. Per tant,

$$wx + x + yz + xz' = x + yz + xz'.$$

Apliquem una altra vegada la llei d'absorció a $x + xz'$, i obtenim el següent:

$$x + yz + xz' = x + yz.$$

Per tant, concloem amb el que reproduïm a continuació:

$$wx + xy' + yz + xz' + xy = x + yz.$$

6.

- Per la primera condició hem de trobar una expressió que valgui 1 quan $x = 1$ o $y = 0$. Això és el mateix que dir que $x = 1$ o $y' = 1$. L'expressió que val 1 quan es compleix aquesta condició és $x + y'$.
- La segona condició és $x = 0$ i $z = 1$. Això és el mateix que $x' = 1$ i $z = 1$, i l'expressió que val 1 en aquest cas és $x' \cdot z$.
- La tercera condició, $x = 1$, $y = 1$ i $z = 1$, només es compleix per l'expressió $x \cdot y \cdot z$.
- Com que la funció ha de valer 1 en qualsevol dels tres casos, és a dir, quan es compleix la condició 1, la condició 2 o la condició 3, tenim que l'expressió de la funció és la següent:

$$F = x + y' + x'z + xyz.$$

7. La funció ha de valer 1 quan s'hagi d'activar l'alarma. Segons ens diu l'enunciat, l'alarma s'activa quan l'interruptor general està tancat ($ig = 0$) i alguna de les dues caixes fortes és oberta (és a dir, $x_A = 1$ o $x_B = 1$).

Una expressió per a la funció S que val 1 quan $ig = 0$ i ($x_A = 1$ o $x_B = 1$) és la següent:

$$s = ig'(x_A + x_B).$$

8. Per a saber quines assignatures ha aprovat en Joan i quina ha suspès, plantejarem les afirmacions dels seus tres amics de manera algebraica.

El resultat de cada assignatura es pot representar amb una variable booleana. Aquesta variable valdrà 1 si l'assignatura està aprovada i 0 en el cas contrari.

Les variables que utilitzarem per a cadascuna de les assignatures seran m per a matemàtiques, f per a física i q per a química.

Podem expressar les tres frases de la manera següent:

– Has aprovat matemàtiques o física: $mf' + m'f$.
– Has suspès química o matemàtiques: $qm' + q'm$.
– Has aprovat dues assignatures: $m'fq + mf'q + mfq'$.

Com que s'han de complir les tres afirmacions a la vegada, tenim el següent:

$$(mf' + m'f)\ (qm' + q'm)\ (m'fq + mf'q + mfq') = 1.$$

Simplificarem l'expressió aplicant els axiomes i teoremes de l'àlgebra de Boole fins que obtinguem la resposta:

$$(mf' + m'f)(qm' + q'm)(m'fq + mf'q + mfq') =$$

(propietat distributiva sobre els dos primers parèntesis)

$$(mf'qm' + mf'q'm + m'fqm' + m'fq'm)(m'fq + mf'q + mfq') =$$

(complementació i idempotència)

$$(0 + mf'q' + m'fq + 0)\ (m'fq + mf'q + mfq') =$$

(elements neutres i distributiva sobre els dos parèntesis)

$$mf'q'm'fq + mf'q'mf'q + mf'q'mfq' + m'fqm'fq + m'fqmf'q + m'fqmfq' =$$

(complementació i idempotència)

$$0 + 0 + 0 + m'fq + 0 + 0 = m'fq.$$

Hem obtingut, per tant, l'expressió següent:

$$m'fq = 1.$$

Aquesta expressió només val 1 quan m val 0 i f i q valen 1, és a dir, que en Joan ha suspès matemàtiques i ha aprovat física i química.

9. Les taules de veritat d'aquestes funcions són les següents:

x	y	g_1
0	0	0
0	1	0
1	0	0
1	1	1

x	y	g_3
0	0	0
0	1	0
1	0	1
1	1	1

x	y	g_6
0	0	0
0	1	1
1	0	1
1	1	0

x	y	g_7
0	0	0
0	1	1
1	0	1
1	1	1

x	y	g_{10}
0	0	1
0	1	0
1	0	1
1	1	0

10. Posarem les variables a l'esquerra de la taula, en ordre decreixent de subíndexs. Les quatre variables poden prendre setze combinacions de valors diferents; les escriurem en ordre lexicogràfic (si les interpretéssim com a nombres naturals, correspondrien als números des del 0 fins al 15). A la dreta de la taula hi haurà la columna corresponent a f.

x_3	x_2	x_1	x_0	f
0	0	0	0	
0	0	0	1	
0	0	1	0	
0	0	1	1	
0	1	0	0	
0	1	0	1	
0	1	1	0	
0	1	1	1	
1	0	0	0	
1	0	0	1	
1	0	1	0	
1	0	1	1	
1	1	0	0	
1	1	0	1	
1	1	1	0	
1	1	1	1	

11. La taula de veritat és la següent:

x_3	x_2	x_1	x_0	f
0	0	0	0	1
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	0
1	0	0	0	0
1	0	0	1	1
1	0	1	0	1
1	0	1	1	0
1	1	0	0	1
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

12. Per a obtenir aquestes taules, escriurem una columna per a cadascun dels termes producte, i després obtindrem la columna corresponent a f fent l'OR de les columnes parcials.

a)

x	y	z	w	$x \cdot y' \cdot z \cdot w'$	$y \cdot z' \cdot w$	$x' \cdot z$	$x \cdot y \cdot w$	$x' \cdot y \cdot z' \cdot w'$	f
0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0
0	0	1	0	0	0	1	0	0	1
0	0	1	1	0	0	1	0	0	1
0	1	0	0	0	0	0	0	1	1
0	1	0	1	0	1	0	0	0	1
0	1	1	0	0	0	1	0	0	1
0	1	1	1	0	0	1	0	0	1
1	0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0	0	0
1	0	1	0	1	0	0	0	0	1
1	0	1	1	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0	0
1	1	0	1	0	1	0	1	0	1
1	1	1	0	0	0	0	0	0	0
1	1	1	1	0	0	0	1	0	1

b)

x	y	z	w	$x \cdot y$	$x \cdot y + z$
0	0	0	0	0	0
0	0	0	1	0	0
0	0	1	0	0	1
0	0	1	1	0	1
0	1	0	0	0	0
0	1	0	1	0	0
0	1	1	0	0	1
0	1	1	1	0	1
1	0	0	0	0	0
1	0	0	1	0	0
1	0	1	0	0	1
1	0	1	1	0	1
1	1	0	0	1	1
1	1	0	1	1	1
1	1	1	0	1	1
1	1	1	1	1	1

13. Escriurem les taules de veritat de les dues funcions:

x	y	z	w	$x' \cdot y'$	$y \cdot w$	$x' \cdot w$	f
0	0	0	0	1	0	0	1
0	0	0	1	1	0	1	1
0	0	1	0	1	0	0	1
0	0	1	1	1	0	1	1
0	1	0	0	0	0	0	0
0	1	0	1	0	1	1	1
0	1	1	0	0	0	0	0
0	1	1	1	0	1	1	1
1	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0
1	0	1	0	0	0	0	0
1	0	1	1	0	0	0	0
1	1	0	0	0	0	0	0
1	1	0	1	0	1	0	1
1	1	1	0	0	0	0	0
1	1	1	1	0	1	0	1

x	y	z	w	$x' \cdot y' \cdot z' \cdot w'$	$x' \cdot z' \cdot w$	$y \cdot z' \cdot w$	$x \cdot y \cdot w$	$x' \cdot z$	g
0	0	0	0	1	0	0	0	0	1
0	0	0	1	0	1	0	0	0	1
0	0	1	0	0	0	0	0	1	1
0	0	1	1	0	0	0	0	1	1
0	1	0	0	0	0	0	0	0	0
0	1	0	1	0	1	1	0	0	1
0	1	1	0	0	0	0	0	1	1
0	1	1	1	0	0	0	0	1	1
1	0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0	0
1	0	1	1	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0	0
1	1	0	1	0	0	1	1	0	1
1	1	1	0	0	0	0	0	0	0
1	1	1	1	0	0	0	1	0	1

Deduïm que les funcions no són equivalents perquè les seves taules de veritat són diferents.

14. La taula de veritat és la següent:

ig	x_A	x_B	s
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

15. Per a resoldre aquest problema, assignarem una funció a cadascuna de les afirmacions dels amics d'en Joan. La funció valdrà 1 per les combinacions que fan certa l'afirmació, i 0 en el cas contrari. Les funcions que representen les afirmacions dels amics corresponen a les funcions F_1, F_2 i F_3, respectivament.
Anomenarem m, f i q les variables que representen les assignatures. Aquestes variables valdran 1 quan l'assignatura estigui aprovada, i 0 si està suspesa.

La solució, representada per la funció F, l'obtindrem fent una AND de les funcions F_1, F_2 i F_3, ja que totes tres frases són certes simultàniament. La combinació per la qual F val 1 correspondrà a la solució del problema.

m	f	q	F_1	F_2	F_3	F
0	0	0	0	0	0	0
0	0	1	0	1	0	0
0	1	0	1	0	0	0
0	1	1	1	1	1	1
1	0	0	1	1	0	0
1	0	1	1	0	1	0
1	1	0	0	1	1	0
1	1	1	0	0	0	0

La funció F indica que en Joan ha aprovat física i química i ha suspès les matemàtiques.

16.
$f_0 = x_2'x_1x_0' + x_2x_1x_0$,
$f_1 = x_2x_1'x_0' + x_2x_1'x_0 + x_2x_1x_0' + x_2x_1x_0$,
$f_2 = x_2'x_1'x_0 + x_2'x_1x_0 + x_2x_1'x_0 + x_2x_1x_0$,
$f_3 = x_2'x_1'x_0 + x_2'x_1x_0' + x_2x_1'x_0' + x_2x_1x_0$.

17. Escriurem la taula de veritat i a partir d'aquesta obtindrem l'expressió en suma de mintermes.

a) La taula de veritat de la funció és la següent:

x	y	z	f
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

L'expressió de f en suma de mintermes és aquesta:

$$f = x'y'z + x'yz + xy'z + xyz' + xyz.$$

b) La taula de veritat de la funció és la següent:

x	y	z	w	f
0	0	0	0	1
0	0	0	1	1
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	1
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	0
1	1	1	1	0

L'expressió de f en suma de mintermes és la que reproduïm a continuació:

$$f = x'y'z'w' + x'y'z'w + x'y'zw + x'yz'w + xy'z'w + xy'zw + xyz'w.$$

18. Les combinacions d'entrada 101, 110 i 111 no es poden produir, i, per tant, no ens importarà el valor que prenguin les sortides en aquests casos.
La taula de veritat de les funcions d'aquest sistema és la següent:

x_2	x_1	x_0	y_4	y_3	y_2	y_1	y_0
0	0	0	1	0	1	1	1
0	0	1	1	1	0	1	1
0	1	0	0	0	1	0	0
0	1	1	0	1	1	1	1
1	0	0	0	1	1	0	1
1	0	1	x	x	x	x	x
1	1	0	x	x	x	x	x
1	1	1	x	x	x	x	x

19.
a) A partir de l'enunciat, obtenim que les expressions de les funcions R i E són les següents:

$$R = h_0' + h_1't$$
$$E = t'h_0$$

Ara bé, les combinacions amb $h_1 = 1$ i $h_0 = 0$ no es poden produir (hem de suposar que els sensors funcionen bé). Per tant, la taula de veritat del sistema és la següent:

t	h_1	h_0	R	E
0	0	0	1	0
0	0	1	0	1
0	1	0	x	x
0	1	1	0	1
1	0	0	1	0
1	0	1	1	0
1	1	0	x	x
1	1	1	0	0

20. Per a fer l'anàlisi, cal obtenir l'expressió de la funció a partir del circuit. A partir de l'expressió obtindrem la taula de veritat. Per a omplir-la còmodament, trobarem prèviament una expressió simplificada de la funció.

a) L'expressió que obtenim directament a partir del circuit és la següent:

$$F = (y' + w')\ (y + w) + (x' + z)\ (x + z').$$

Si simplifiquem aquesta expressió, obtenim el següent:

$$F = y'w + yw' + x'z' + xz.$$

La taula de veritat és la següent:

x	y	z	w	f
0	0	0	0	1
0	0	0	1	1
0	0	1	0	0
0	0	1	1	1
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1
0	1	1	1	0
1	0	0	0	0
1	0	0	1	1
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	0
1	1	1	0	1
1	1	1	1	1

b) L'expressió que obtenim directament a partir del circuit és la següent:

$$F = ((x' + w)\ (y + w'))'z + (y + w')z'.$$

Si simplifiquem aquesta expressió, obtenim el que reproduïm a continuació:

$$F = ((x' + w)' + (y + w')')z + (y + w')z' =$$
$$=(xw' + y'w)z + (y + w')z'=$$
$$=xzw' + y'zw + yz' + z'w'.$$

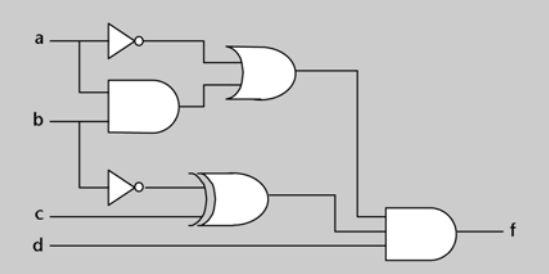
La taula de veritat és la següent:

x	y	z	w	f
0	0	0	0	1
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	1
0	1	0	1	1
0	1	1	0	0
0	1	1	1	0
1	0	0	0	1
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0

21. Per a fer la síntesi, simplement dibuixem una línia per a cada variable de la funció i substituïm les operacions lògiques de l'expressió per la porta lògica corresponent. La funció és aquesta:

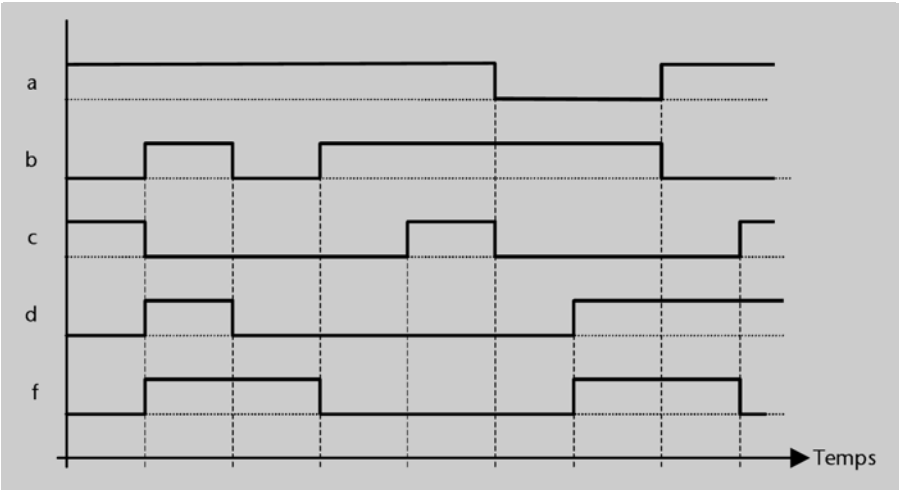
$$f(a, b, c, d) = d\cdot(a' + a \cdot b)\cdot(b' \oplus c).$$

La figura següent mostra el circuit.



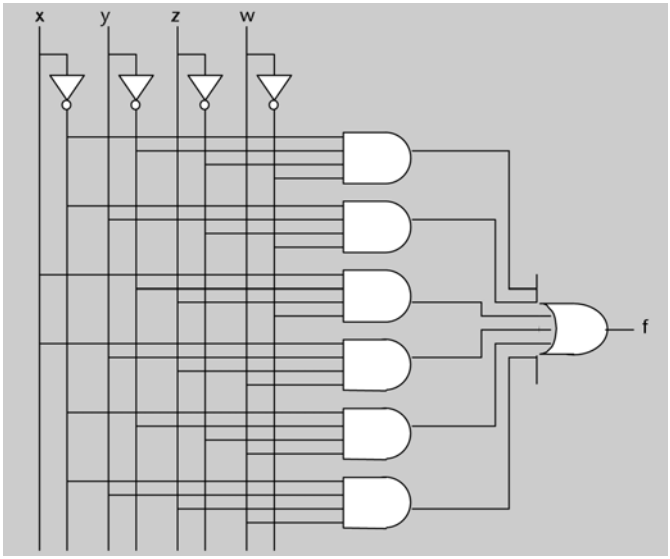
22. En la figura es mostra el cronograma. Com que es tracta d'un circuit combinacional en el qual no es consideren els retards (tal com farem habitualment en aquesta assignatura), les variacions en la sortida es produeixen en el mateix moment en què es produeix alguna variació de les entrades (línies puntejades verticals). No obstant això, no totes les variacions de les entrades produeixen una variació de la sortida, com es pot veure en el cronograma.

La funció que implementa el circuit és $f = b'c' + c'd$.

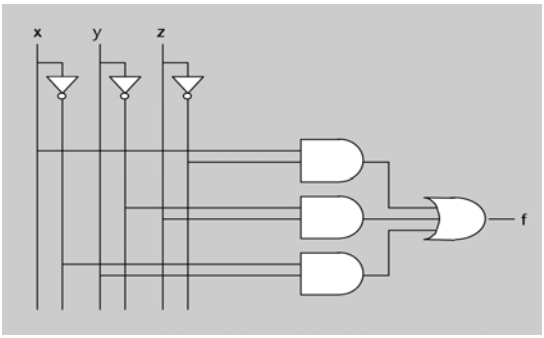


23. En les figures es mostra la síntesi a dos nivells de cada circuit. Cada terme producte està implementat per una porta AND de tantes entrades com variables té el terme producte. La suma lògica està implementada per una porta OR de tantes entrades com termes producte té l'expressió.

a)



b)



24.

a) El rang de valors que pot prendre un nombre natural de dos bits és 0..3. Per tant, el rang de la multiplicació de dos números serà 0..9. Per a representar el 9 necessitem quatre bits. Per tant, la sortida tindrà quatre bits.

b) Anomenarem A i B, respectivament, els nombres d'entrada, i C la seva multiplicació. Per a obtenir la taula de veritat passem A i B a decimal, calculem $C = A * B$ en decimal, i finalment codifiquem C en binari, amb els bits $c_3 \ldots c_0$. Per exemple, prenem la fila $[a_1\ a_0\ b_1\ b_0] = [1\ 0\ 1\ 1]$. Tenim que $A = 2$, $B = 3$. Per tant, $C = 2 * 3 = 6$, que es codifica [0 1 1 0] en binari. La taula de veritat completa és la següent:

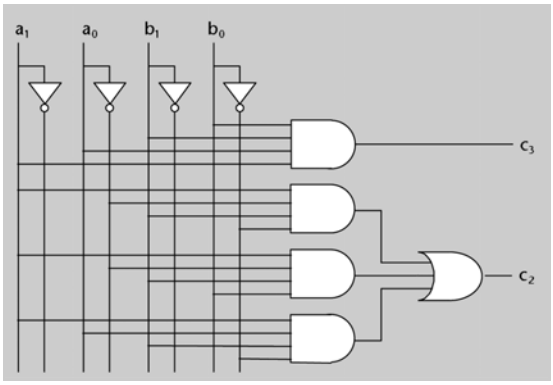
a_1	a_0	b_1	b_0	c_3	c_2	c_1	c_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0
0	0	1	0	0	0	0	0
0	0	1	1	0	0	0	0
0	1	0	0	0	0	0	0
0	1	0	1	0	0	0	1
0	1	1	0	0	0	1	0
0	1	1	1	0	0	1	1
1	0	0	0	0	0	0	0
1	0	0	1	0	0	1	0
1	0	1	0	0	1	0	0
1	0	1	1	0	1	1	0
1	1	0	0	0	0	0	0
1	1	0	1	0	0	1	1
1	1	1	0	0	1	1	0
1	1	1	1	1	0	0	1

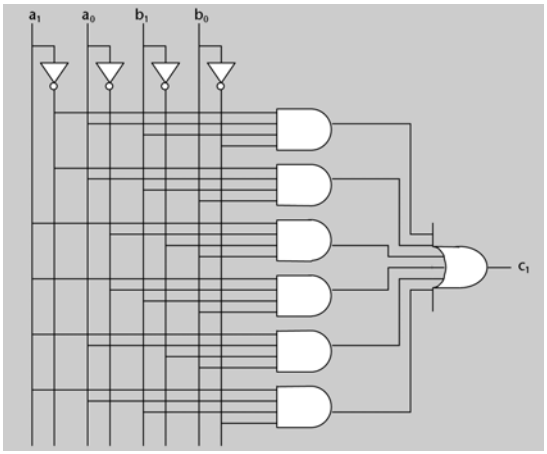
c) Les expressions en suma de productes de les quatre funcions de sortida són les següents:

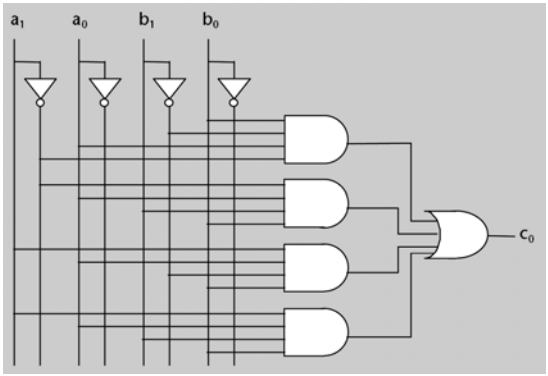
$c_3 = a_1a_0b_1b_0$,
$c_2 = a_1a_0'b_1b_0' + a_1a_0'b_1b_0 + a_1a_0b_1b_0'$,

$c_1 = a_1'a_0b_1b_0' + a_1'a_0b_1b_0 + a_1a_0'b_1'b_0 + a_1a_0'b_1b_0 + a_1a_0b_1'b_0 + a_1a_0b_1b_0'$,
$c_0 = a_1'a_0b_1'b_0 + a_1'a_0b_1b_0 + a_1a_0b_1'b_0 + a_1a_0b_1b_0$.

A continuació, es mostren els circuits a dos nivells corresponents a les quatre funcions de sortida.



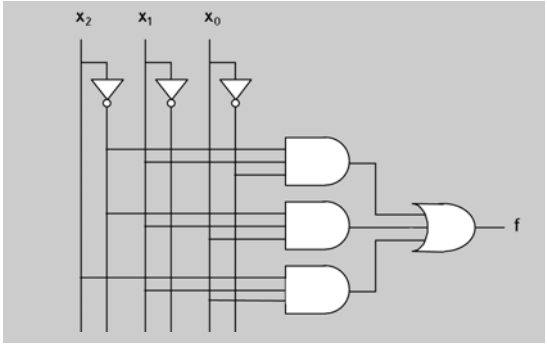




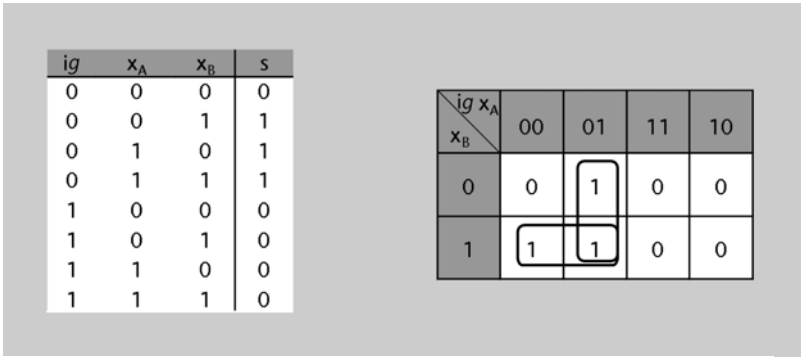
25. L'expressió en suma de mintermes de la funció és la següent:

$$f = x_2'x_1x_0' + x_2'x_1x_0 + x_2x_1x_0$$

A continuació, es mostra la síntesi a dos nivells de la funció.



26. Tot seguit, es mostra la taula de veritat de la funció i la seva minimització utilitzant el mètode de Karnaugh.



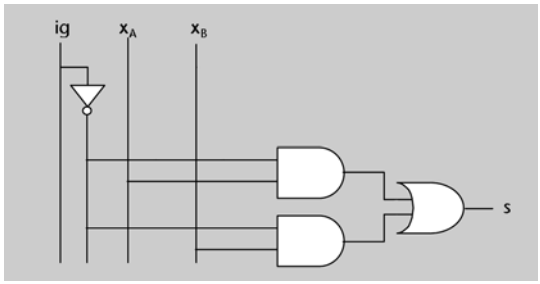
ig	x_A	x_B	s
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

x_B \ ig x_A	00	01	11	10
0	0	1	0	0
1	1	1	0	0

Del rectangle vertical se n'obté el terme producte $ig' \cdot x_A$, i del rectangle horitzontal el terme $ig' \cdot x_B$. Per tant, l'expressió de la funció és aquesta:

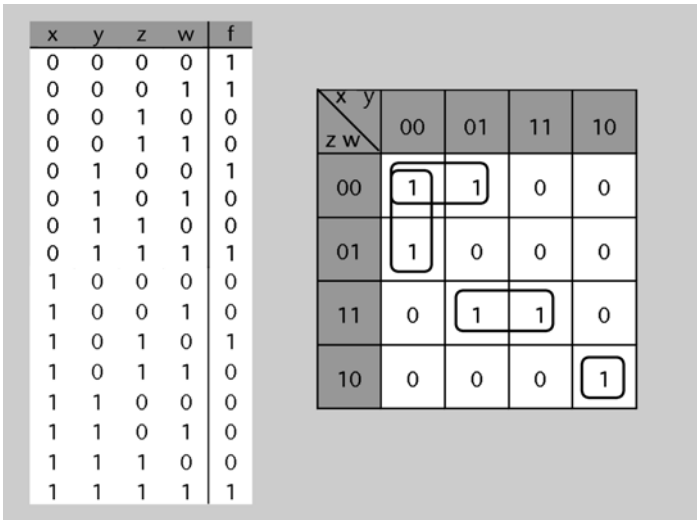
$$s = ig'x_A + ig'x_B.$$

A continuació, es mostra la síntesi mínima a dos nivells de la funció.



Fixeu-vos que si traiem ig' factor comú en l'expressió mínima de s, obtenim $s = ig'(x_A + x_B)$, que és l'expressió que hem deduït en l'activitat 7 (però no és una suma de productes).

27. Tot seguit, es mostren la taula de veritat de la funció i la seva minimització per Karnaugh.



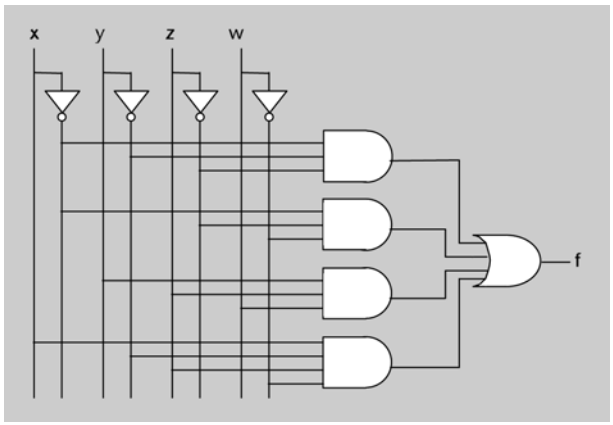
x	y	z	w	f
0	0	0	0	1
0	0	0	1	1
0	0	1	0	0
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

z w \ x y	00	01	11	10
00	1	1	0	0
01	1	0	0	0
11	0	1	1	0
10	0	0	0	1

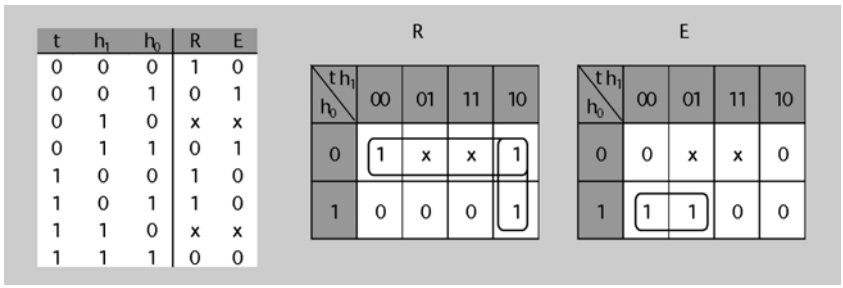
Del mapa de Karnaugh se'n dedueix aquesta expressió:

$$f = x'y'z' + x'z'w' + yzw + xy'zw'.$$

El circuit mínim a dos nivells es presenta en la figura següent. Es pot comprovar que és més senzill que el que s'ha obtingut en l'activitat 23.



28. En la figura següent es mostra la taula de veritat de les funcions i la seva minimització per Karnaugh.



t	h_1	h_0	R	E
0	0	0	1	0
0	0	1	0	1
0	1	0	x	x
0	1	1	0	1
1	0	0	1	0
1	0	1	1	0
1	1	0	x	x
1	1	1	0	0

R

h_0 \ th_1	00	01	11	10
0	1	x	x	1
1	0	0	0	1

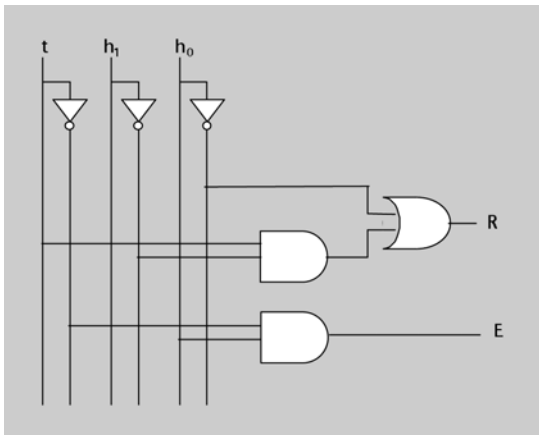
E

h_0 \ th_1	00	01	11	10
0	0	x	x	0
1	1	1	0	0

Les funcions obtingudes són les següents:

$$R = h_0' + th_1',$$
$$E = t'h_0.$$

En la figura següent es mostra el circuit minimitzat.

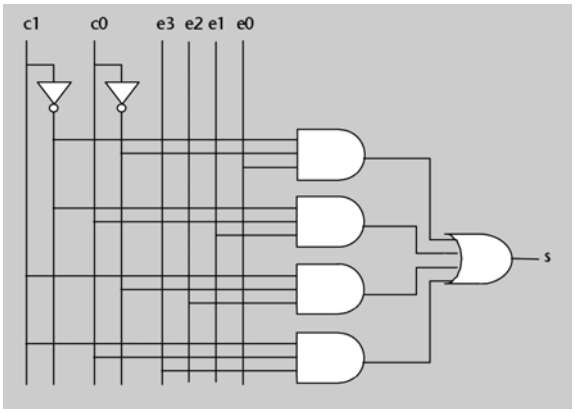


29. El valor de la sortida *s* serà el valor de *e*3, *e*2, *e*1 o *e*0 d'acord amb el que valguin les entrades de control *c*1 i *c*0.

Per exemple, si [*c*1 *c*0] = [1 0], la sortida *s* valdrà 1 si *e*2 = 1, i valdrà 0 si *e*2 = 0 (és a dir, valdrà *e*2). En aquest cas podríem escriure que l'expressió de la sortida és *c*1*c*0'*e*2. En cada moment, les entrades *c*1 i *c*0 valen alguna d'aquestes quatre combinacions: [0 0], [0 1], [1 0] o bé [1 1]. Per tant, només un dels productes *c*1'*c*0', *c*1*c*0', *c*1'*c*0 i *c*1*c*0 pot valer 1 en cada moment. Podem concloure que una expressió vàlida per a *s* és la següent:

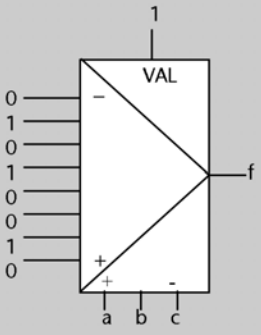
$$s = c1'c0'e0 + c1'c0e1 + c1c0'e2 + c1c0e3.$$

La implementació amb portes d'aquesta expressió és la següent:



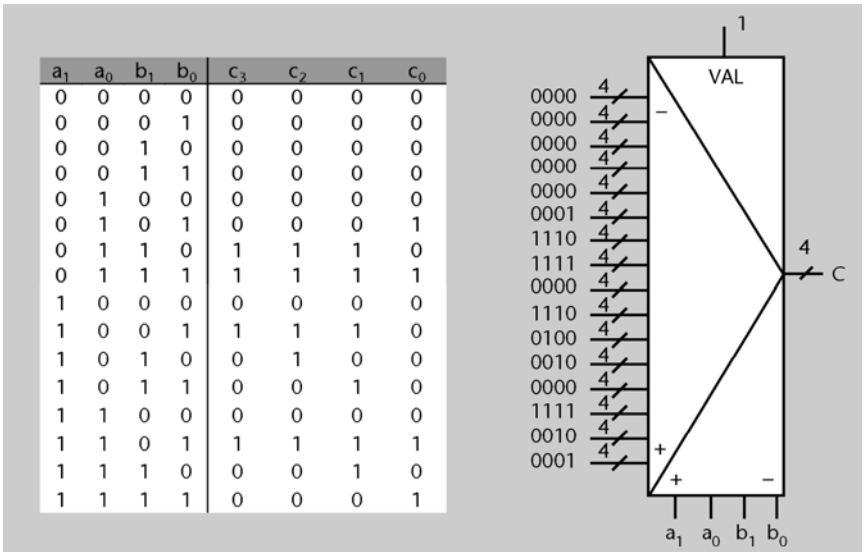
30. Per a implementar aquesta funció construirem la seva taula de veritat. A partir d'aquesta taula sabrem què hem de connectar a les entrades del multiplexor per tal que a la sortida ens aparegui un 0 o un 1 segons indiqui la funció.
En definitiva, es tracta de traspassar la columna *f* de la taula de veritat a les entrades del multiplexor.

a	b	c	f
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0



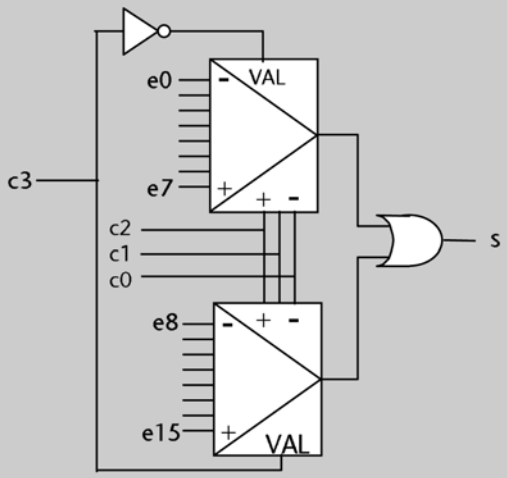
31. En primer lloc, construirem la taula de veritat del circuit.
El rang d'un nombre enter representat en complement a 2 amb dos bits és [–2..1]. Per tant, el rang del producte de dos d'aquests nombres és [–2..4]. Per a representar nombres dins aquest rang n'hi ha prou amb quatre bits, ja que amb quatre bits podem representar nombres en el rang [–8..7].
Per tant, la taula de veritat té quatre entrades i quatre sortides. Les entrades són els dos bits de cadascun dels números *A* i *B*, i les sortides són els quatre bits del resultat *C*.
Per a implementar el circuit utilitzarem un multiplexor de busos de quatre bits de setze entrades de dades. La figura mostra aquesta implementació.

a_1	a_0	b_1	b_0	c_3	c_2	c_1	c_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0
0	0	1	0	0	0	0	0
0	0	1	1	0	0	0	0
0	1	0	0	0	0	0	0
0	1	0	1	0	0	0	1
0	1	1	0	1	1	1	0
0	1	1	1	1	1	1	1
1	0	0	0	0	0	0	0
1	0	0	1	1	1	1	0
1	0	1	0	0	1	0	0
1	0	1	1	0	0	1	0
1	1	0	0	0	0	0	0
1	1	0	1	1	1	1	1
1	1	1	0	0	0	1	0
1	1	1	1	0	0	0	1

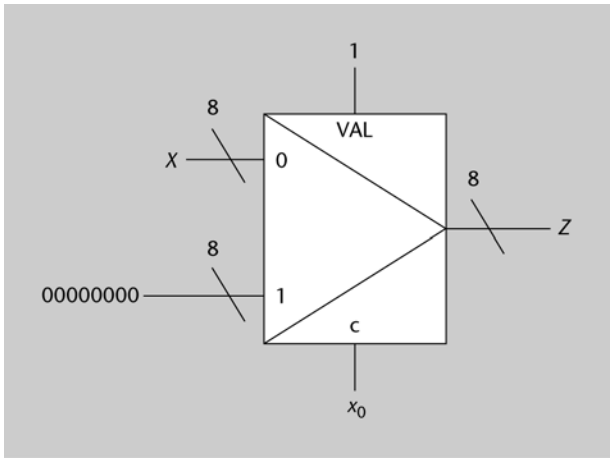


32. Escrivim primer la taula de veritat corresponent a un multiplexor 16-1. Aquest multiplexor té quatre entrades de control (*c*3.. *c*0) i setze entrades de dades (*e*15..*e*0).
La línia discontínua a la taula mostra que es pot descompondre el funcionament del circuit com si es tractés de dos multiplexors 8-1, tots dos controlats pels senyals *c*2, *c*1 i *c*0. El senyal *c*3 determina quin dels dos multiplexors funciona en cada moment.

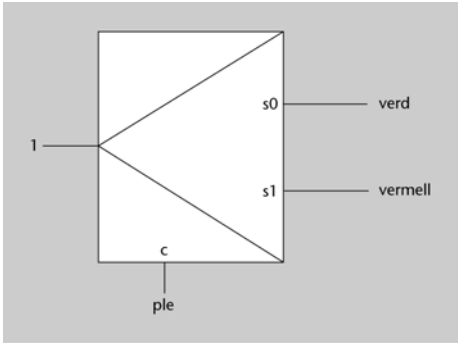
c3	c2	c1	c0	f
0	0	0	0	e0
0	0	0	1	e1
0	0	1	0	e2
0	0	1	1	e3
0	1	0	0	e4
0	1	0	1	e5
0	1	1	0	e6
0	1	1	1	e7
1	0	0	0	e8
1	0	0	1	e9
1	0	1	0	e10
1	0	1	1	e11
1	1	0	0	e12
1	1	0	1	e13
1	1	1	0	e14
1	1	1	1	e15



33. Com que *Z* ha de valer un de dos possibles valors, la podem implementar mitjançant un multiplexor 2-1. El bit de menys pes de *X* ens indica si és parell (0) o senar (1). Podem doncs connectar aquest bit a l'entrada de control del multiplexor, i això ens duu a connectar *X* a l'entrada de dades 0 i 00000000 a l'entrada de dades 1.



34. N'hi ha prou de posar un 1 a l'entrada del desmultiplexor, i fer que el senyal *ple* controli cap a quin dels dos llums arriba aquest 1.



35. Deduirem les expressions lògiques de cadascuna de les sortides i construirem el circuit a partir d'aquestes. Per a resoldre l'activitat, és convenient de tenir al davant la representació gràfica del codificador i la seva taula de veritat (figura 21 dels apunts).

Com que la sortida del codificador ha de codificar en binari l'entrada de més pes que valgui 1, podem fer el raonament següent:

1) La sortida $s1$ es posarà a 1 quan estiguin a 1 les entrades $e3$ o $e2$ (en aquests casos s'ha de codificar un 11 o un 10, respectivament).

2) La sortida $s0$ es posarà a 1 quan estigui a 1 l'entrada $e3$ (en aquest cas s'ha de codificar un 11, independentment dels valors de les altres entrades) o bé quan ho estigui l'entrada $e1$ i no ho estigui ni la $e2$ ni la $e3$ (en aquest cas, $[e3\ e2\ e1] = [0\ 0\ 1]$, s'ha de codificar un 01).

3) D'altra banda, l'entrada de validació ha d'estar activa perquè el codificador funcioni com a tal.

Per tant, les expressions algebraiques de les sortides són les següents:

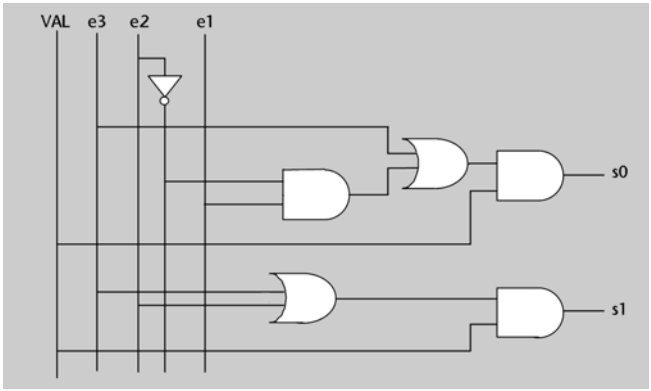
– $s1$ valdrà 1 quan VAL = 1 i $e3$ o $e2$ siguin 1 (apartats 1 i 3):

$$s1 = \text{VAL}\ (e3 + e2).$$

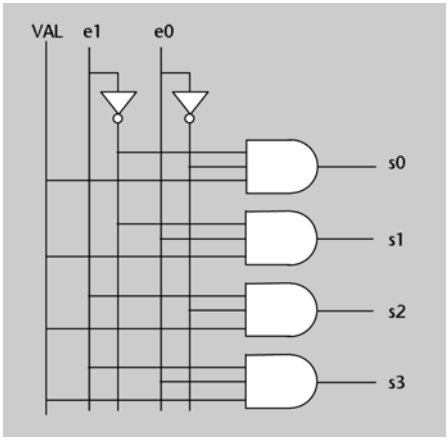
– $s0$ valdrà 1 quan VAL = 1 i $e3$ sigui 1, o quan $[e3\ e2\ e1] = [0\ 0\ 1]$ (apartats 2 i 3):

$$s0 = \text{VAL}\ (e3 + e3'e2'e1) = \text{VAL}\ (e3 + e2'e1).$$

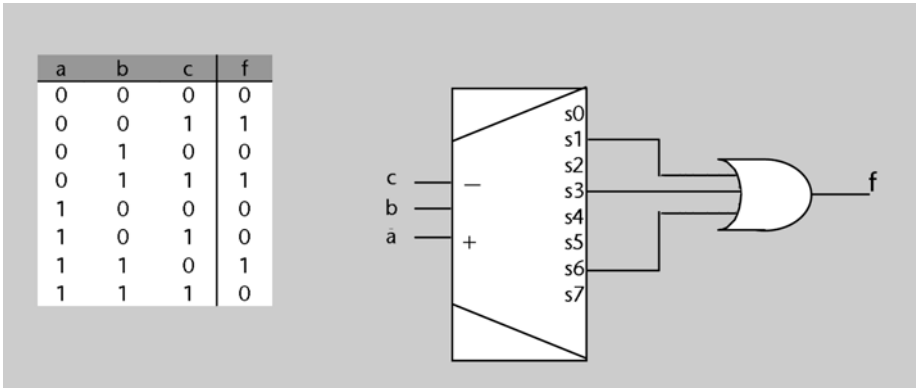
A continuació, es mostra una possible implementació d'aquest codificador. Com es pot veure en la figura, el valor de l'entrada $e0$ no té influència en les sortides. Aquest és el motiu que fa que un codificador tingui un 0 a la seva sortida tant quan ha de codificar un 0 com quan cap de les seves entrades té un 1.



36. Quan l'entrada de validació està activa, cada sortida d'un descodificador es posa a 1 només quan es produeix una combinació determinada de les entrades. Per tant, la implementació de cadascuna de les sortides serà un circuit que detecti si aquesta combinació està present a l'entrada, tal com es mostra en la figura.



37. Primer farem la taula de veritat, i a partir de la taula utilitzarem un descodificador per a sintetitzar el circuit. La porta OR fa la suma lògica dels casos en què la funció val 1.



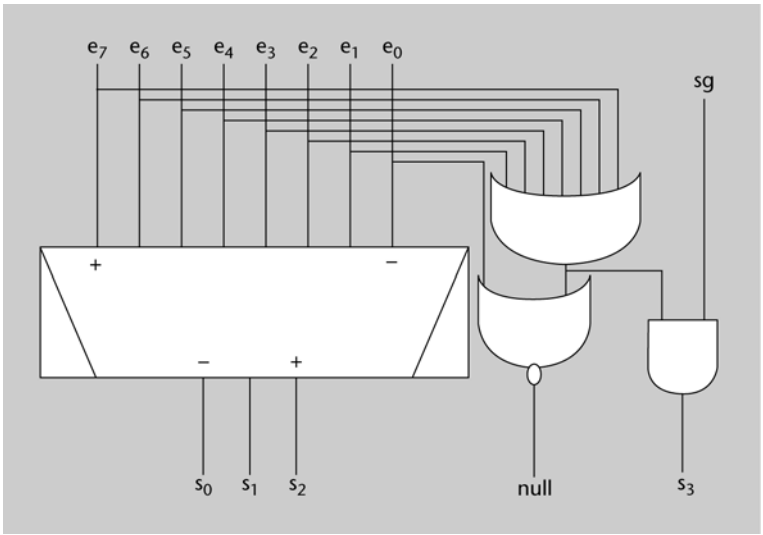
38. Podem generar la magnitud del nombre usant un codificador 8-3, a les entrades del qual connectarem *e*7, *e*6, ..., *e*0. Les sortides d'aquest codificador, per tant, seran directament les sortides *s*2, *s*1 i *s*0. Fixem-nos que això també farà que [*s*2 *s*1 *s*0] = [0 0 0] quan no hi hagi cap entrada *ei* a 1 (tal com ens demana l'enunciat).

A primera vista podríem pensar que el bit *s*3 (signe del nombre que s'ha de representar) el podem obtenir directament de l'entrada *sg*, però cal tenir en compte aquestes dues excepcions:

- quan *e*0 = 1, aleshores *s*3 ha de valer 0 independentment del valor de *sg*.
- quan no hi hagi cap entrada *ei* a 1, *s*3 també ha de valdre 0 independentment del valor de *sg*.

Per tant, *s*3 ha de valer 1 quan alguna de les entrades *e*7, ..., e1 sigui 1 i *sg* = 1.

La sortida *null* ha de valer 1 només quan no hi hagi cap entrada *ei* a 1.



39. Per tal de deduir què fa el circuit, construirem la seva taula de veritat. La taula mostra també el valor de l'entrada i la sortida quan les interpretem com a nombres representats en complement a 2 (columnes *X* i *Y*).

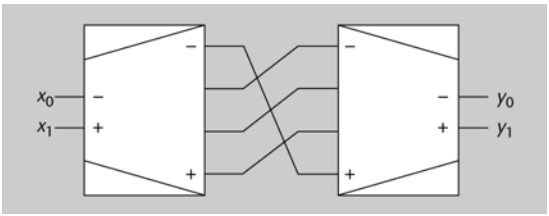
X	x_2	x_1	x_0	y_2	y_1	y_0	*Y*
0	0	0	0	0	0	0	0
1	0	0	1	1	1	1	–1
2	0	1	0	1	1	0	–2
3	0	1	1	1	0	1	–3
–4	1	0	0	1	0	0	–4
–3	1	0	1	0	1	1	3
–2	1	1	0	0	1	0	2
–1	1	1	1	0	0	1	1

Com es pot veure en la taula, allò que fa el circuit és canviar el signe del nombre de l'entrada. En el cas particular $X = -4$ això no és possible, perquè el 4 no es pot representar en complement a 2 amb només tres bits (el resultat és erroni en aquest cas: es produeix sobreeiximent).

40. A partir de la codificació en binari d'un nombre, hem de generar la codificació en binari d'un altre nombre. Ho podem aconseguir connectant les sortides d'un descodificador amb les entrades d'un codificador, de manera que la sortida i del descodificador es connecti amb l'entrada *j* del codificador, essent *j* el valor que ha de tenir la sortida quan l'entrada val *i* (és a dir, aplicant la mateixa tècnica que en l'activitat anterior).
La correspondència entre valors de l'entrada *X* i valors de la sortida *Y* l'expressa aquesta taula:

X	*Y*
0	3
1	0
2	1
3	2

Per tant, les connexions s'han de fer tal com es mostra en la figura següent.



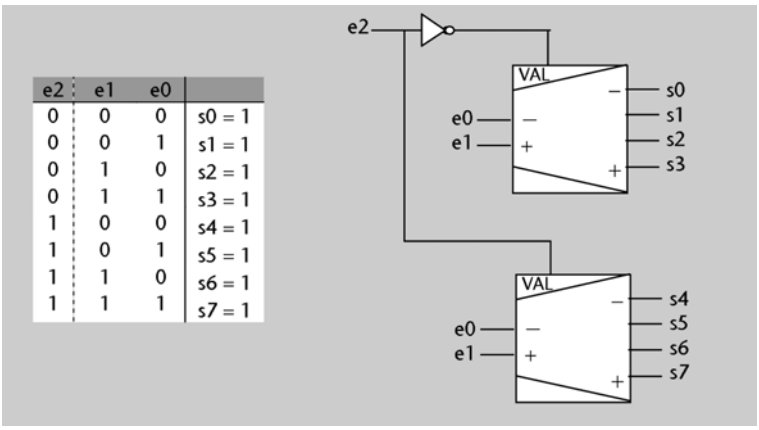
41.
a) E_0 és 1 quan les dues entrades del descodificador són 0, és a dir, quan $x_1 = 0$ i $x_0x_3 = 0$. Per tant, l'expressió algebraica per a E_0 és $E_0 = x_1'(x_0x_3)'$. Si seguim el mateix raonament per a la resta de sortides, obtenim el següent:
$E_0 = x_1'(x_0x_3)'$,
$E_1 = x_1'(x_0x_3) = x_1'x_0x_3$,
$E_2 = x_1(x_0x_3)'$,
$E_3 = x_1(x_0x_3) = x_1x_0x_3$.

b) La taula de veritat es troba en la figura següent. Per a obtenir la columna corresponent a *F* hem fet un pas intermedi: hem posat en una columna el valor de *F* en termes de les entrades de dades del multiplexor (E_i), i en la columna final hem substituït les variables E_i pel valor que prenen per a cada combinació de x_i, d'acord amb les expressions obtingudes en l'apartat **a**.

x_3	x_2	x_1	x_0	*F*	*F*
0	0	0	0	E_0	1
0	0	0	1	E_0	1
0	0	1	0	E_0	0
0	0	1	1	E_0	0
0	1	0	0	E_1	0
0	1	0	1	E_1	0
0	1	1	0	E_1	0
0	1	1	1	E_1	0
1	0	0	0	E_2	0
1	0	0	1	E_2	0
1	0	1	0	E_2	1
1	0	1	1	E_2	0
1	1	0	0	E_3	0
1	1	0	1	E_3	0
1	1	1	0	E_3	0
1	1	1	1	E_3	1

42. Aquesta activitat és molt semblant a l'activitat 32. En la figura següent es mostra la taula de veritat del descodificador 3-8, que té tres entrades (*e*2..*e*0), i vuit sortides (*s*0..*s*7). Per a simplificar

la taula, s'ha posat una única columna de sortida en la qual hem indicat quina de les sortides val 1; totes les altres valen 0. Com es pot veure en la taula de veritat, l'entrada *e*2 determina quin dels dos descodificadors 2-4 funciona en cada moment (i, per tant, controla la seva entrada de validació). A cada descodificador 2-4, les entrades *e*1 i *e*0 determinen quina de les sortides ha de valer 1.



43. Per a resoldre aquest exercici construirem la seva taula de veritat i després interpretarem els nombres com a naturals.
Anomenem X el nombre que surt del codificador: $X = [x_1\ x_0]$.

A	a_1	a_0	z	x_1	x_0	X	s_1	s_0	S
0	0	0	0	0	1	1	0	0	0
1	0	1	0	1	0	2	0	1	1
2	1	0	0	1	1	3	1	0	2
3	1	1	0	0	0	0	1	1	3
0	0	0	1	0	1	1	0	1	1
1	0	1	1	1	0	2	1	0	2
2	1	0	1	1	1	3	1	1	3
3	1	1	1	0	0	0	0	0	0

Tal com podem veure en la taula,

$$X = (A + 1) \bmod 4.$$

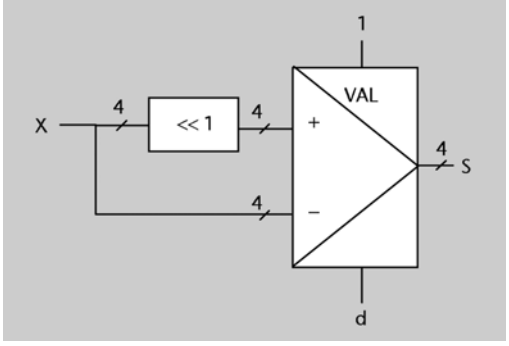
La variable z selecciona entre les entrades A i X, i, per tant, la sortida es pot expressar amb la taula següent:

z	S
0	A
1	X = (A+1) mod 4

Això es pot resumir en aquesta expressió:

$$S = (A + z) \bmod 4.$$

44.
a) En la figura es mostra la implementació del circuit. El senyal d determina si la sortida S val el mateix que l'entrada X ($d = 0$) o X decalat un bit a l'esquerra ($d = 1$).



b) Aquest desplaçament d'un bit a l'esquerra és equivalent a multiplicar l'entrada per dos. Per tant, $S = 2 * X$.
c) Es produeix sobreeiximent quan el resultat no es pot representar amb quatre bits.

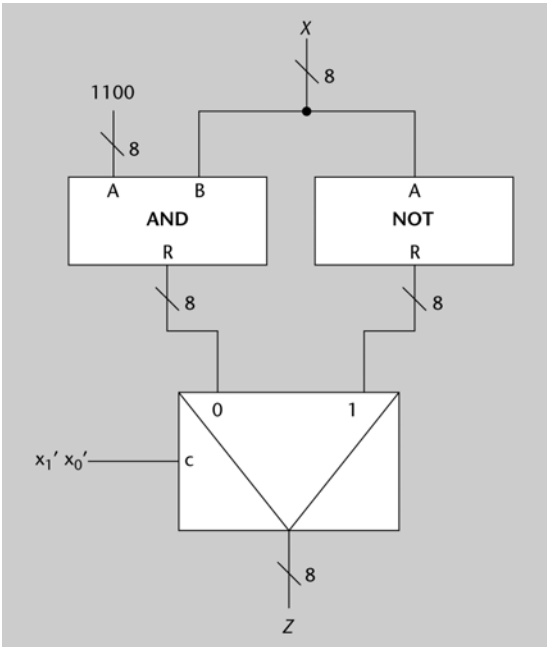
– Si interpretem els números X i S com a números naturals codificats en binari, el rang de X i S és [0..15]. Per tant, quan X sigui més gran que 7, S no podrà representar el resultat i es produirà sobreeiximent.

Sobreeiximent si $X > 7$.

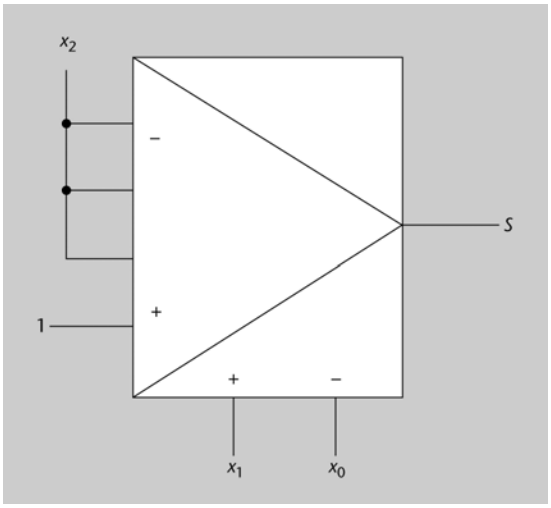
– Si interpretem els nombres com a enters codificats en complement a 2, el rang de X i S és [–8..7]. Per tant, es produirà sobreeiximent quan X sigui més petit que –4 o més gran que 3.

Sobreeiximent si $X < -4$ o $X > 3$.

45. La sortida Z ha de prendre un de dos valors; això ho podem aconseguir mitjançant un multiplexor 2-1. Quin dels dos valors prendrà depèn de si X és múltiple de 4 o no. Un nombre representat en binari és múltiple de 4 si els dos bits de menys pes valen 0; per tant, a l'entrada de control del multiplexor hi hem de connectar $x_1'x_0'$.
Per obtenir els valors que hem de connectar a cadascuna de les entrades de dades del multiplexor podem usar els blocs que es descriuen en l'apartat 3.4, tal com es mostra en la figura.



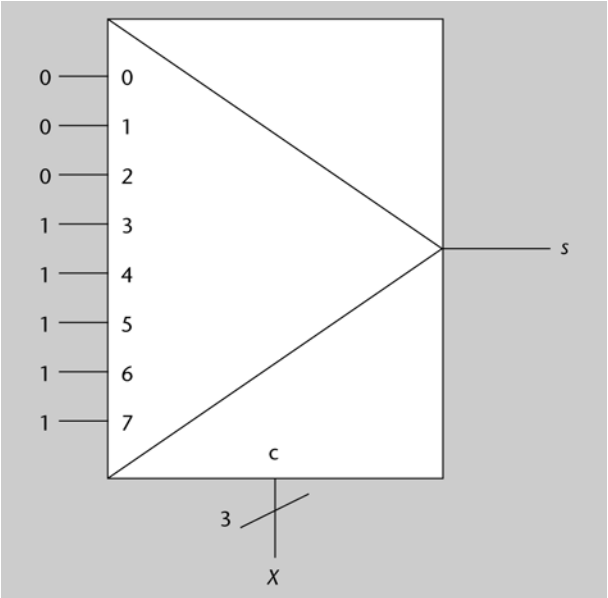
46.
a) Veiem que per qualsevol combinació de valors de x_1 i x_0 el senyal de sortida s val x_2, excepte en el cas $x_1 = x_0 = 1$ en què s val 1 ($x_2 + x_2'$). Per tant, podem implementar $M0$ amb un multiplexor 4-1 i sense cap porta addicional, per exemple:



Una altra opció és implementar la funció a partir de la seva taula de veritat, tal com es fa en la figura 18 de l'apartat 3.1:

$x_2x_1x_0$	s
000	0
001	0
010	0
011	1
100	1
101	1
110	1
111	1

En aquest cas farem servir un multiplexor 8-1, a les quals connectarem les variables de la funció, i copiarem els 1*s* i 0*s* a les entrades de dades del multiplexor.



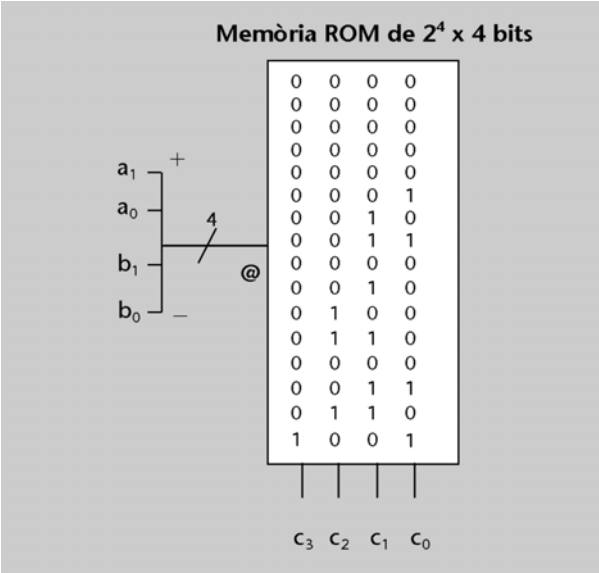
b) El valor *U* és una XOR de *X* amb la negació de si mateixa. Atès que a XOR $a' = 1$, obtenim que *U* valdrà sempre 111, i per tant *V* valdrà sempre 000. Deduïm, doncs, la funció del circuit *M*1 és generar la combinació 000.
c) Analitzem els valors que pren *Y* en funció dels valors que pren *X*.
Com hem vist en l'apartat *a)* analitzant la sortida del bloc *M*0, *s* val 0 si $X = 0$, 1 o 2 i val 1 sempre que $X > 2$. Per tant, si $X \leq 2$ la sortida *Y* del circuit val 0 (que és el que val *V* en tot moment) i si $X > 2$ val *W*.
Ara analitzem el valor de *W*:

a) Si $X \leq 2$ no cal que l'analitzem perquè sabem que en aquest cas *Y* val 0.
b) Si *X* val 3 o 4, *W* val 3 i 4 respectivament.
c) Si $X \geq 5$, *W* val 5.

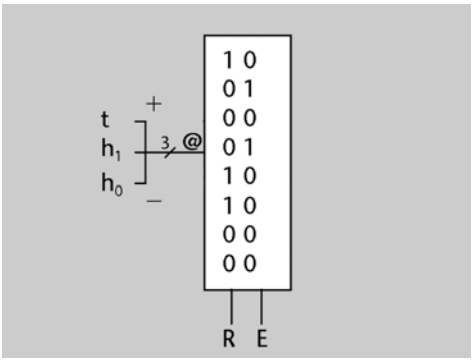
Ajuntant-ho tot obtenim aquesta taula de veritat:

$x_2x_1x_0$	$y_2y_1y_0$
000	000
001	000
010	000
011	011
100	100
101	101
110	101
111	101

47. El circuit multiplicador de dos nombres naturals de dos bits té quatre entrades ($a_1\ a_0\ b_1\ b_0$) i quatre sortides ($c_3\ c_2\ c_1\ c_0$), tal com s'explica en l'activitat 24.
Quatre variables poden prendre $2^4 = 16$ combinacions diferents; per tant, la grandària de la ROM és de setze mots de quatre bits (un per a cada funció de sortida).
En la figura següent es mostra el seu contingut, que coincideix amb la part dreta de la taula de veritat de l'activitat 24. La variable a_1 està connectada al bit de més pes del bus d'adreces, i la variable b_0 al bit de menys pes (és molt important indicar en el bus d'adreces a quin senyal d'entrada es connecta el bit de més pes per tal que el resultat sigui correcte. Si els bits d'entrada estiguessin en un altre ordre, la taula de veritat seria diferent i, per tant, també el contingut de la ROM).



48. Tal com s'explica en l'activitat 19, aquest circuit té tres senyals d'entrada i dos de sortida, i, per tant, la grandària de la memòria ROM és de $2^3 \times 2$ bits (vuit mots de dos bits).
En el contingut de la memòria s'han substituït les *x* de la taula de veritat per zeros, ja que el valor de qualsevol bit en una memòria ROM ha d'estar definit. Això no obstant, les *x* es podrien haver substituït per qualsevol valor binari.



49. Connectarem l'entrada *X* a l'entrada d'adreces de la memòria ROM, per tal que a la sortida ens doni la representació en complement a 2 desitjada. Per tant, l'entrada d'adreces serà de 8 bits i la memòria tindrà 256 mots de 8 bits cadascun.
En el mot de l'adreça *i* posarem la representació en complement a 2 del nombre que en signe i magnitud es codifica amb la seqüència de bits corresponent a la representació binària del nombre *i*. Així, s'accedirà a les adreces 00000000..01111111 quan el nombre *X* sigui positiu, que en complement a 2 es codifiquen igual que en signe i magnitud, de manera que en el mot de cadascuna d'aquestes adreces hi haurà la mateixa combinació de bits que la de l'adreça corresponent. Les adreces 10000000..11111111 s'accediran quan el nombre *X* sigui negatiu, de manera que en el mot de cadascuna d'aquestes adreces hi haurà la representació en complement a 2 del nombre –*A*, essent *A* el nombre que codifiquen els 7 bits més baixos de l'adreça.
L'adreça 50 és, expressada en binari, l'adreça 00110010. Si la llegim com un nombre expressat en signe i magnitud veiem que és positiu, i per tant el contingut d'aquesta adreça serà 00110010.
L'adreça 150 és, expressada en binari, l'adreça 10010110. Si la llegim com un nombre expressat en signe i magnitud veiem que és negatiu. La magnitud del nombre és 0010110. Abans

de canviar-li el signe l'hem d'expressar en 8 bits, ja que estem buscant una representació en complement a 2 i 8 bits: 00010110. Ara canviem 1*s* per 0*s* i viceversa i sumem 1 al resultat, i obtenim 11101010. Aquest serà el contingut de l'adreça 150.
L'adreça 200 és, expressada en binari, l'adreça 11001000. Si la llegim com un nombre expressat en signe i magnitud veiem que és negatiu. La magnitud del nombre és 1001000. De nou, abans de canviar-li el signe l'hem d'expressar en 8 bits: 01001000. Ara canviem 1*s* per 0*s* i viceversa i sumem 1 al resultat, i obtenim 10111000. Aquest serà el contingut de l'adreça 200.

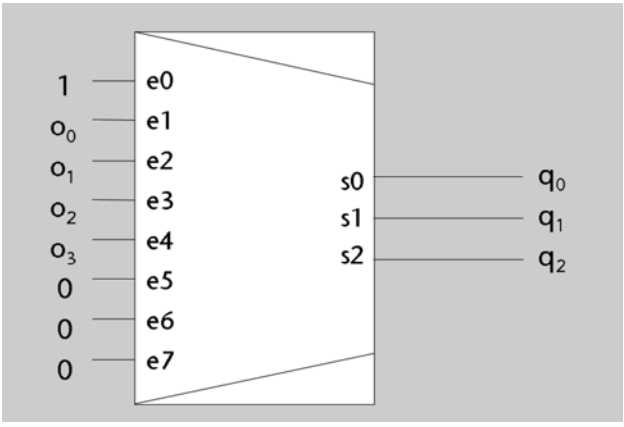
50.
a) La taula de veritat del circuit és la següent:

e_3 e_2 e_1 e_0	o_3 o_2 o_1 o_0
0 0 0 0	0 0 0 0
0 0 0 1	0 0 0 1
0 0 1 0	0 0 0 1
0 0 1 1	0 0 1 1
0 1 0 0	0 0 0 1
0 1 0 1	0 0 1 1
0 1 1 0	0 0 1 1
0 1 1 1	0 1 1 1
1 0 0 0	0 0 0 1
1 0 0 1	0 0 1 1
1 0 1 0	0 0 1 1
1 0 1 1	0 1 1 1
1 1 0 0	0 0 1 1
1 1 0 1	0 1 1 1
1 1 1 0	0 1 1 1
1 1 1 1	1 1 1 1

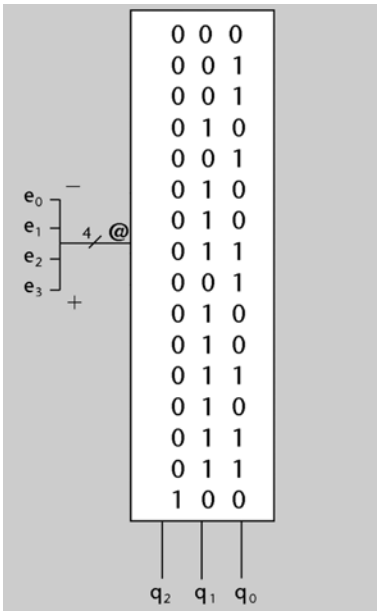
b) Les sortides del bloc ORDENAR tenen tots els uns a la dreta, tal com es mostra en la taula següent (recordem que QUANTS = [q_2 q_1 q_0]):

o_3 o_2 o_1 o_0	**QUANTS**	q_2 q_1 q_0
0 0 0 0	0 (no hi ha 1s a l'entrada)	0 0 0
0 0 0 1	1 (hi ha un 1s a l'entrada)	0 0 1
0 0 1 1	2 (hi ha dos 1s a l'entrada)	0 1 0
0 1 1 1	3 (hi ha tres 1s a l'entrada)	0 1 1
1 1 1 1	4 (hi ha quatre 1s a l'entrada)	1 0 0

Fixem-nos que la sortida QUANTS ha de formar la codificació binària d'un dels números 0, 1, 2, 3 o 4. Per a sintetitzar-la, per tant, podem usar un codificador 8-3 (calen tres bits per a codificar el quatre en binari). Concretament, tenim que si el bit més alt d'ORDRE que està a 1 és o_i, llavors QUANTS ha de valer $i + 1$. Per tant, connectarem el senyal o_i a l'entrada $i + 1$ del codificador, tal com es mostra en la figura següent.
En l'entrada 0 del codificador hi pot haver tant un 1 com un 0 (en tots dos casos, QUANTS valdrà 0 si no hi ha cap bit d'ORDRE a 1).



c) La grandària de la ROM ha de ser $2^4 \times 3$ bits, i el seu contingut és el següent:



51. Primer deduirem l'expressió algebraica de cadascuna de les funcions. Anomenarem a_1, a_0, b_1 i b_0 als bits dels nombres d'entrada. Perquè els nombres siguin iguals, ho han de ser bit a bit. Per tant, s'ha de complir que $a_0 = b_0$ i $a_1 = b_1$.

La porta XOR permet de detectar la desigualtat entre dos bits (recordeu la seva taula de veritat). Per tant, una negació de la seva sortida detectarà la igualtat. Així, doncs, l'expressió per a la sortida $A = B$ és la següent:

$$(a_1 \oplus b_1)'(a_0 \oplus b_0)'.$$

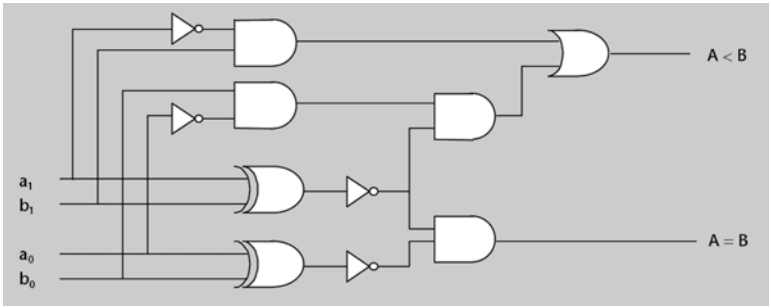
Per a fer la sortida $A < B$ tindrem en compte el següent:

– A és més petit que B si ho és el seu bit de més pes: $b_1 = 1$ i $a_1 = 0$.
– Si els dos bits de més pes de A i B són iguals, llavors A és més petit que B si ho és el seu bit de menys pes: $a_1 = b_1$ i $b_0 = 1$ i $a_0 = 0$.

Per tant, l'expressió per la sortida $A < B$ és la següent:

$$b_1 a_1' + (b_1 \oplus a_1)' b_0\, a_0'.$$

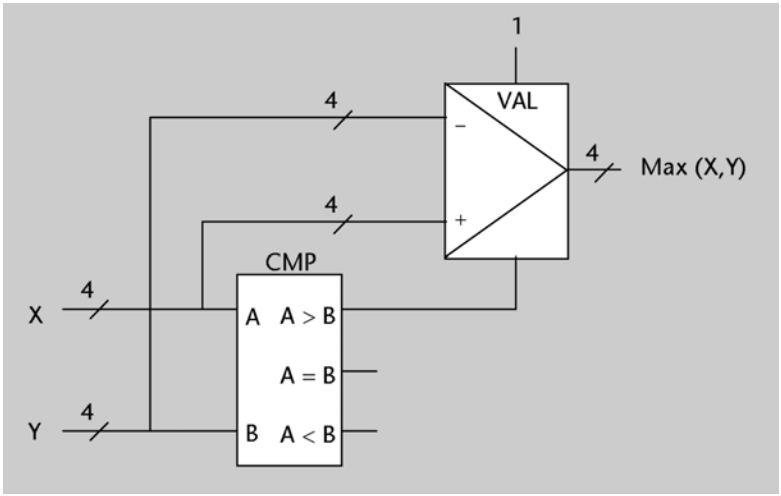
En la figura següent es troba la implementació dels dos circuits.



52. Per a decidir quin del dos nombres és més gran utilitzarem un comparador de quatre bits. Connectarem el número X a l'entrada A i el número Y a l'entrada B del comparador. La sortida $A > B$ d'aquest comparador valdrà 1 si $X > Y$, i 0 en el cas contrari.

Aquesta sortida controlarà un multiplexor de busos que seleccionarà entre X i Y. Així, quan $A > B$ valdrà 1 seleccionarem el número X, i en qualsevol altre cas ($X <= Y$) seleccionarem el número Y.

La figura següent mostra aquest disseny.

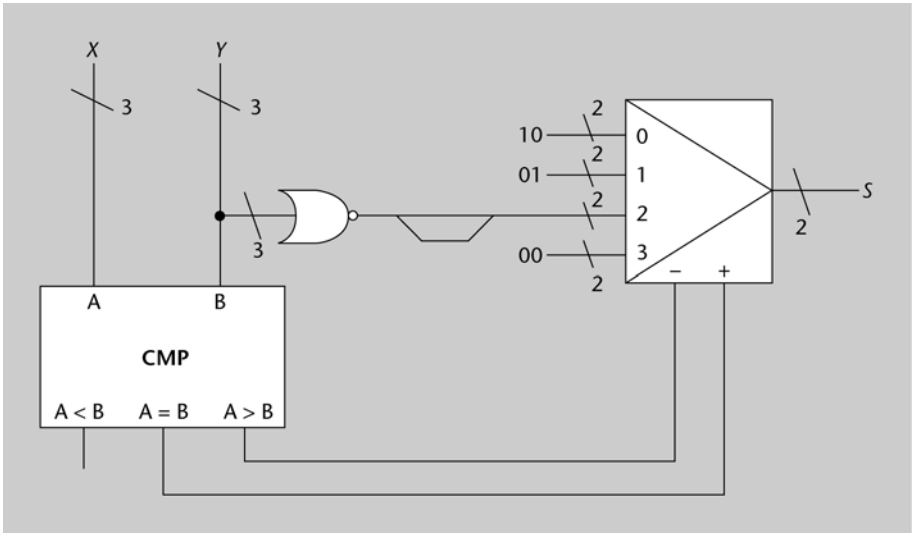


53. La sortida *S* val en tot moment algun dels quatre valors. Una manera d'implementar-la és, doncs, mitjançant un multiplexor 4-1.

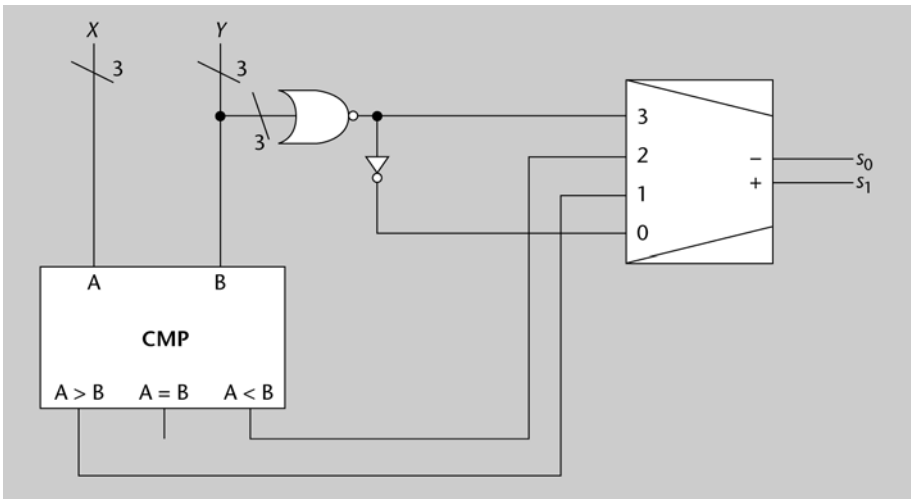
Quin d'aquests valors s'haurà d'escollir en cada moment depèn del resultat de la comparació entre *X* i *Y*, cosa que ens porta a usar un comparador, i, en cas que $X = Y$, de si *X* (o *Y*) = 0; això ho podem saber usant una porta NOR a les entrades de la qual haurem de connectar els 3 bits d'*X* o d'*Y*.

Hi ha moltes possibilitats per a controlar quin valor surt en cada moment del multiplexor a partir de les sortides del comparador i de la porta NOR. Una és la que es mostra en la figura, segons la qual:

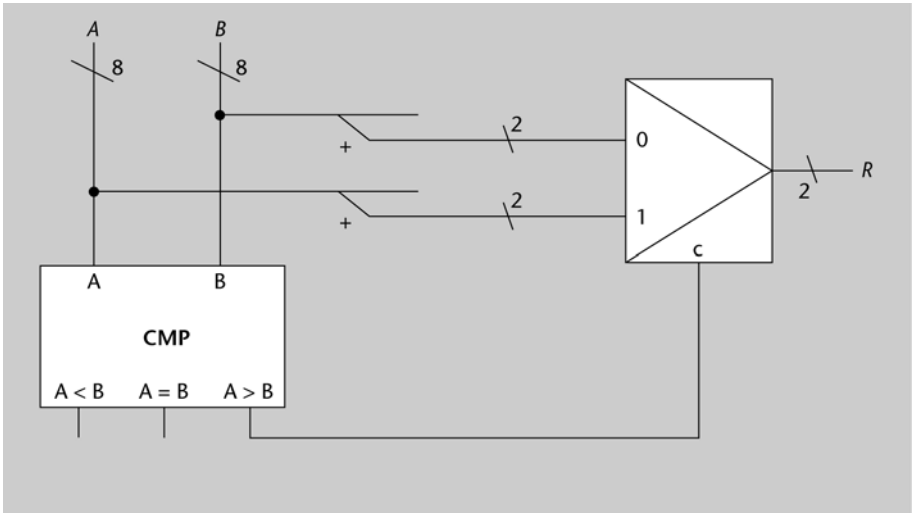
- L'entrada de dades 0 passarà cap a la sortida quan $X < Y$ (ni $X = Y$ ni $X > Y$).
- L'entrada de dades 1 passarà cap a la sortida quan $X > Y$.
- L'entrada de dades 2 passarà cap a la sortida quan $X = Y$. En aquest cas, si la sortida de la porta NOR és 1 la sortida *S* ha de valer 11, i si és 0 *S* ha de valer 00. Per tant, podem duplicar la sortida de la porta NOR per obtenir el bus de dos bits que connectem a l'entrada de dades 2.
- L'entrada de dades 3 no passarà mai cap a la sortida perquè mai no es donarà la combinació [1 1] en les entrades de control. Hi podem connectar, doncs, qualsevol cosa, per exemple 00.



També podem dissenyar el circuit sense multiplexor, generant els valors que pot prendre la sortida amb un codificador 4-2, ja que els valors que pot prendre són les quatre combinacions que es poden fer amb dos bits. Observem que si $X > Y$ o $X < Y$ l'entrada 0 del codificador estarà a 1, però a la sortida no s'hi generarà la combinació [0 0] perquè hi haurà una altra entrada del codificador també a 1.

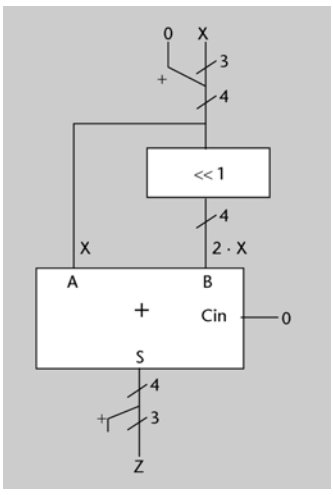


54. Per a saber quin dels dos forns està més calent usarem un comparador. Fixem-nos que el valor de *R* coincideix en tot moment amb els dos bits més alts de la temperatura del forn que està més calent (o igual), de manera que podem generar *R* a partir d'aquests bits. Un multiplexor selecciona si s'han d'agafar de la temperatura *A* o de la *B*.



55. Fer l'operació X mod 2^n en binari consisteix a quedar-se amb els n bits de menys pes de X. Per tant, per a implementar aquesta funció n'hi haurà prou de sumar $X + 2 * X$ (X decalat un bit a l'esquerra) i quedar-se amb els tres bits de menys pes de la sortida del sumador ($[z_2\ z_1\ z_0]$).

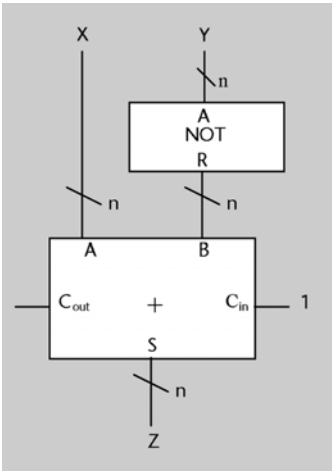
La figura següent mostra aquest disseny.



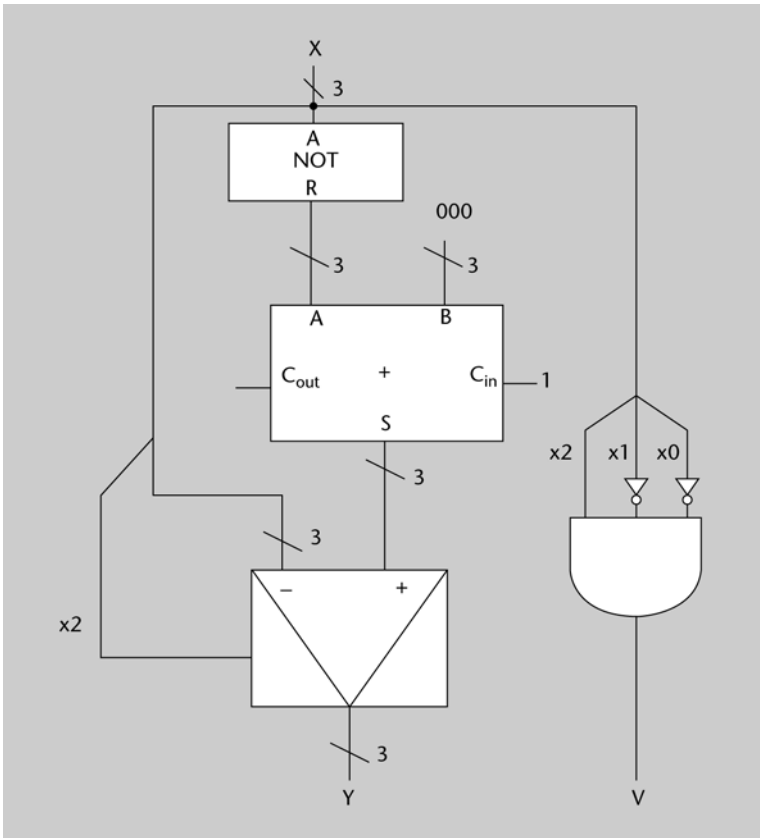
56. Per a fer aquesta operació aprofitarem la igualtat següent, vàlida quan els nombres estan representats en complement a 2:

$$Z = X - Y = X + (-Y) = X + Y' + 1.$$

Per tant, per a implementar el circuit ens cal un bloc NOT de n bits per a obtenir Y', i un sumador de n bits per a fer la suma $X + Y'$. L'1 que encara queda per sumar es pot connectar a l'entrada de ròssec del sumador.
La figura següent mostra el disseny proposat.



57. En cas que el nombre sigui positiu la sortida ha de ser igual a l'entrada, i en cas que sigui negatiu cal canviar el signe de l'entrada; això ho aconseguim negant tots els bits i sumant 1 al resultat. Per a seleccionar una opció o l'altra fem servir el bit de més pes de l'entrada.
Només es pot produir sobreeiximent en cas que l'entrada valgui –4, ja que en aquest cas el valor de la sortida (4) no és representable amb 3 bits en complement a 2. Per tant, $V = x_2x_1'x_0'$.



58.
a) La taula de veritat és la següent (hi escrivim també la interpretació dels nombres en complement a 2 per a fer-la més entenedora). Per a calcular el resultat de la multiplicació passem les entrades a decimal com en l'activitat 24.

A	B	a_1	a_0	b_1	b_0	m_3	m_2	m_1	m_0	$M = A*B$
0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	–2	0	0	1	0	0	0	0	0	0
0	–1	0	0	1	1	0	0	0	0	0
1	0	0	1	0	0	0	0	0	0	0
1	1	0	1	0	1	0	0	0	1	1
1	–2	0	1	1	0	1	1	1	0	–2
1	–1	0	1	1	1	1	1	1	1	–1
–2	0	1	0	0	0	0	0	0	0	0
–2	1	1	0	0	1	1	1	1	0	–2
–2	–2	1	0	1	0	0	1	0	0	4
–2	–1	1	0	1	1	0	0	1	0	2
–1	0	1	1	0	0	0	0	0	0	0
–1	1	1	1	0	1	1	1	1	1	–1
–1	–2	1	1	1	0	0	0	1	0	2
–1	–1	1	1	1	1	0	0	0	1	1

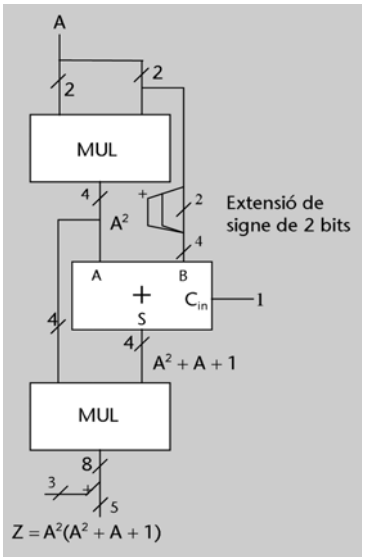
b) Per a obtenir la solució que es descriu en l'enunciat, amb un sumador i dos multiplicadors, s'ha d'utilitzar la igualtat següent:

$$A^4 + A^3 + A^2 = A^2\,(A^2 + A + 1).$$

A continuació, farem un estudi del rang que poden tenir cadascuna de les expressions per tal de decidir el nombre de bits de cada bloc.

$a_1\ a_0$	A	A^2	A^2+A+1	$A^2\cdot(A^2 + A + 1) = A^4 + A^3 + A^2$
00	0	0	1	0
01	1	1	3	3
10	–2	4	3	12
11	–1	1	1	1
calen:		4 bits	3 bits	5 bits

A^2 s'obtindrà mitjançant un bloc MUL de dos bits, connectant A a les dues entrades. A^2, doncs, tindrà quatre bits (el doble de bits que les entrades).
$A^2 + A + 1$ s'obtindrà mitjançant un sumador, connectant l'1 a l'entrada C_{in}. Tot i que si ens fixem en el rang del resultat d'aquesta suma amb tres bits n'hi hauria prou, el sumador ha de ser de quatre bits perquè A^2 té quatre bits. Per tant, caldrà ampliar a quatre bits l'entrada A (fent una extensió de signe) per a connectar-la a l'altra entrada del sumador, tal com es mostra en la figura.
Un altre bloc MUL de quatre bits calcularà $A^2\,(A^2 + A + 1)$. La sortida del multiplicador serà, doncs, de vuit bits, tot i que el rang del resultat només en necessita cinc. El resultat està format, per tant, pels cinc bits de menys pes de la sortida d'aquest segon bloc MUL.



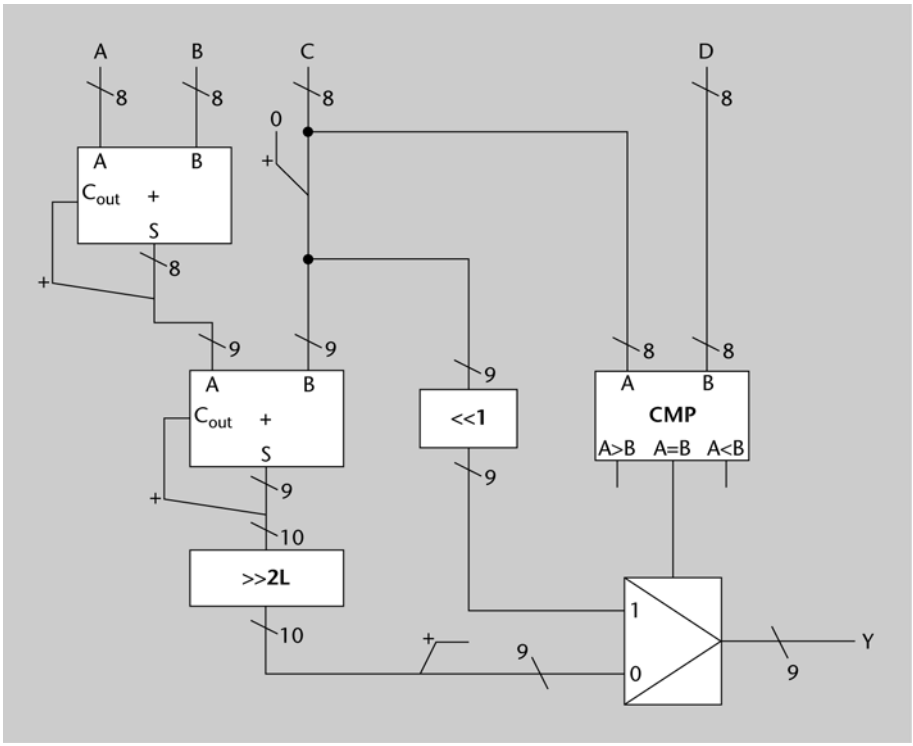
59. Com que A, B, C i D codifiquen nombres naturals amb 8 bits, poden valer entre 0 i 255.

Multiplicarem C per 2 mitjançant un decalador a l'esquerra de 9 bits, perquè el resultat serà un valor entre 0 i 510. Per tant, estenem C a 9 bits abans de connectar-lo a l'entrada del decalador.

D'altra banda, per obtenir la suma $A + B + C$ calcularem primer $A + B$ amb un sumador de 8 bits. El resultat valdrà entre 0 i 510, i per tant s'ha de representar amb 9 bits; els assolirem agregant el C_{out} del sumador al resultat. Després sumem a aquest resultat el valor de C estès a 9 bits, i obtindrem un valor entre 0 i 765, que s'ha de representar amb 10 bits. De nou els obtindrem agregant el C_{out} del sumador al resultat.

Per a dividir la suma entre 4 s'ha de desplaçar 2 bits a la dreta, cosa que podem fer amb un decalador de 10 bits. Com que estem treballant amb nombres naturals, el decalador ha de ser lògic. El resultat de la divisió és un valor entre 0 i 191, que necessita 8 bits per a ser representat. Però, com que Y ha de tenir 9 bits per a representar qualsevol dels dos possibles resultats (ja hem vist que $2 \times C$ necessita 9 bits), descartarem només un dels bits de sortida del decalador.

Finalment, comparem C i D per saber quin dels dos càlculs hem de donar com a resultat. Escollim entre ells mitjançant un multiplexor.



60.

a) Hem de dissenyar una UAL que pugui fer quatre operacions. Algunes d'aquestes es poden fer directament usant blocs que ja hem estudiat en la teoria d'aquest mòdul, però d'altres requereixen la utilització de circuits que combinin blocs amb portes. Analitzarem una per una les operacions, per tal de decidir de quina manera en farem la implementació. Tal com hem vist en la teoria, una vegada s'han implementat els circuits que fan totes les operacions, seleccionarem l'operació que s'ha de fer mitjançant un multiplexor de busos de quatre bits de quatre entrades.

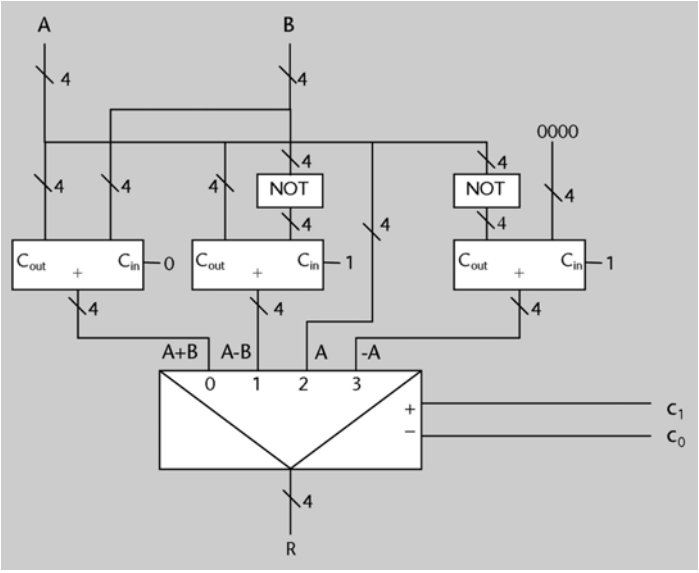
– $R = A + B$ Aquesta suma es pot fer directament amb un sumador de quatre bits.

– $R = A - B$ Tal com s'ha vist en l'activitat 56, $A - B = A + B' + 1$. Per tant, per a implementar aquesta resta farà falta un sumador i un bloc NOT de quatre bits.

– $R = A$ L'entrada A, de quatre bits, estarà connectada directament a una de les entrades del multiplexor.

– $R = -A$ Com hem fet en el cas de la resta, podem usar la igualtat $-A = A' + 1$, i per tant per a implementar aquesta operació necessitarem un sumador i un bloc NOT de quatre bits.

En la figura següent es pot veure la implementació descrita.



b) Analitzem com s'han de calcular cadascun d'aquests bits. Fixem-nos que l'operació $R = A$ no generarà mai sobreeiximent.

- *Vb*: per a estudiar el valor d'aquest bit cal interpretar les entrades (i, per tant, també la sortida) com a nombres naturals codificats en binari.
  - Operació $R = A + B$: en aquest cas el sobreeiximent és el ròssec generat en l'últim bit.
  - Operació $R = A - B$: es produirà sobreeiximent quan el resultat sigui negatiu (no es podrà codificar en binari). Tenint en compte que per a restar sumem el complementari més 1, el resultat és negatiu quan no es produeix ròssec (podeu verificar aquesta afirmació provant uns quants exemples).
  - Operació $R = -A$: es produeix sobreeiximent sempre, perquè el resultat és sempre un nombre negatiu.

  La taula següent resumeix el valor de *Vb*:

c_1	c_0	*Vb*
0	0	C_{out}
0	1	C_{out}'
1	0	0
1	1	1

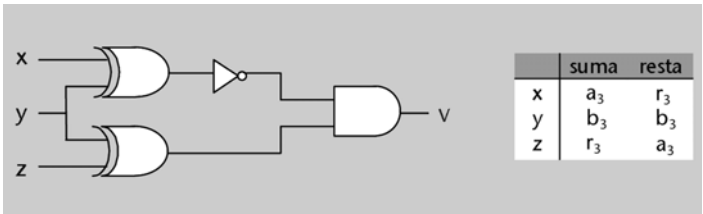
- *V*: per a estudiar el valor d'aquest bit cal que ens posem en el cas d'interpretar les entrades, i, per tant, també la sortida, com a nombres enters codificats en complement a 2.
  - Operació $R = A + B$: es produeix sobreeiximent si els dos sumands són del mateix signe i el resultat és de signe contrari. El signe dels operands i del resultat ve determinat pel seu bit de més pes. Com en l'activitat 51, podem usar l'operació XOR per a detectar igualtat o desigualtat entre aquests bits, i obtenir l'expressió següent:

$$V = (a_3 \oplus b_3)' \, (b_3 \oplus r_3).$$

  - Operació $R = A - B$: es produeix sobreeiximent quan els dos operands són de signe contrari i el resultat és del mateix signe que B. Per tant,

$$V = (a_3 \oplus b_3) \, (r_3 \oplus b_3)'.$$

  Com es pot veure en les dues expressions anteriors, es pot utilitzar el mateix circuit per a calcular el sobreeiximent per a $A + B$ i per a $A - B$, tot i que les entrades s'han de connectar de manera diferent. Aquest circuit té tres entrades, *x*, *y* i *z*, i calcula l'expressió $s = (x \oplus y)' \cdot (y \oplus z)$. L'hem anomenat *detector V*, i el seu disseny intern es presenta a continuació.



	suma	resta
x	a_3	r_3
y	b_3	b_3
z	r_3	a_3

– Operació $R = -A$: es produeix sobreeiximent només en el cas $A = -8$, i, per tant:

$$V = a_3\ a_2'\ a_1'\ a_0'.$$

La taula següent resumeix el valor del bit V:

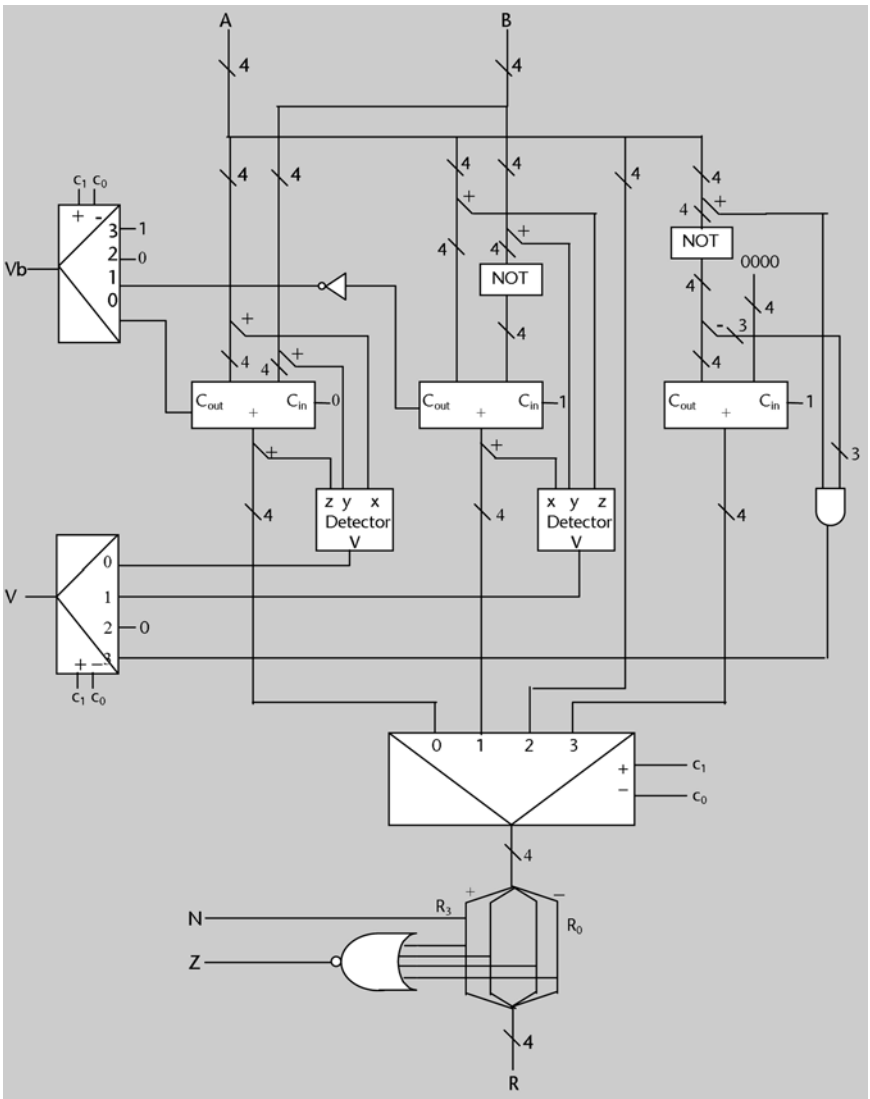
c_1	c_0	V
0	0	$(a_3 \oplus b_3)'(b_3 \oplus r_3)$
0	1	$(r_3 \oplus b_3)'(b_3 \oplus a_3)$
1	0	0
1	1	$a_3\ a_2'\ a_1'\ a_0'$

Per a implementar els dos bits de sobreeiximent farem servir dos multiplexors, tots dos controlats pels bits c_1 i c_0. Els multiplexors seleccionaran en cada moment el valor correcte de V i Vb segons l'operació que es faci.

- N: tal com s'ha vist en la teoria, el signe del resultat és el bit de més pes.

- Z: tal com s'ha vist en la teoria, per a calcular si el resultat és 0 es fa amb una porta NOR de quatre entrades.

A continuació, es mostra el circuit.



Exercicis d'autoavaluació

1. Hem de demostrar que $(xyz)' = x' + y' + z'$. Apliquem en primer lloc la propietat associativa i després el teorema de De Morgan per a dues variables, dues vegades:

$$(xyz)' = ((xy)z)' = (xy)' + z' = x' + y' + z'.$$

Farem el mateix per a quatre variables, havent demostrat ja que es compleix per a tres variables:

$$(xyzw)' = ((xyz)w)' = (xyz)' + w' = x\ '+ y' + z' + w'.$$

2.
$F = (((w'\ y)' + wx)\ z + x')\ y' = ((w'' + y' + wx)\ z + x')\ y' = ((w + y' + wx)\ z + x')\ y' = ((w + y')\ z + x')\ y' =$
$= (wz + y'z + x')\ y' = wzy' + y'y'z + x'y' = wzy' + y'z + x'y'.$

3. Primer farem la taula de veritat. Després implementarem el circuit a dos nivells.

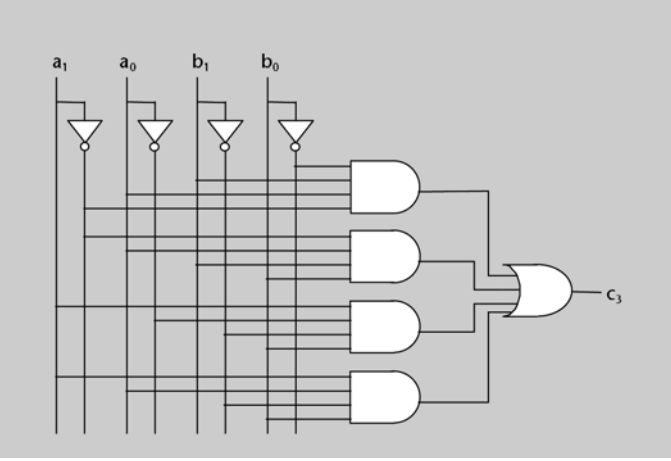
El rang de valors que pot prendre un nombre enter de dos bits representat en complement a dos és [–2..1]. Per tant, el rang de la multiplicació de dos nombres serà [–2..4]. Per a representar el 4 en complement a 2 necessitem quatre bits. Per tant, la sortida tindrà quatre bits.

Anomenarem $[a_1\ a_0]$ i $[b_1\ b_0]$ respectivament als bits dels dos nombres d'entrada, i $[c_3\ c_2\ c_1\ c_0]$ als bits de la seva multiplicació. La taula de veritat d'aquest multiplicador és la següent:

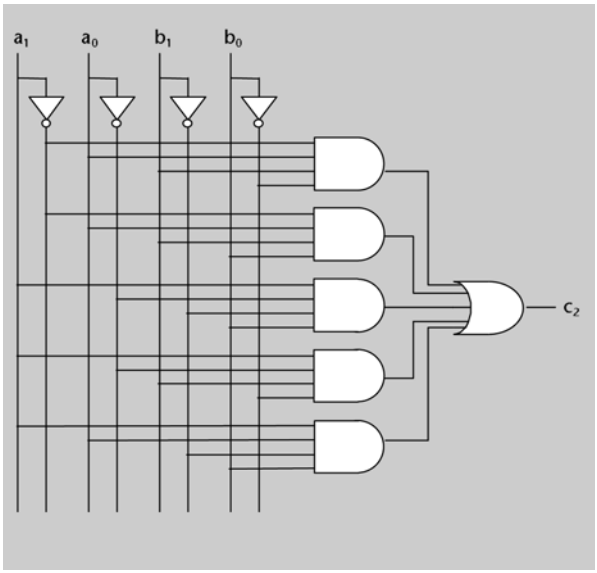
a_1	a_0	b_1	b_0	c_3	c_2	c_1	c_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0
0	0	1	0	0	0	0	0
0	0	1	1	0	0	0	0
0	1	0	0	0	0	0	0
0	1	0	1	0	0	0	1
0	1	1	0	1	1	1	0
0	1	1	1	1	1	1	1
1	0	0	0	0	0	0	0
1	0	0	1	1	1	1	0
1	0	1	0	0	1	0	0
1	0	1	1	0	0	1	0
1	1	0	0	0	0	0	0
1	1	0	1	1	1	1	1
1	1	1	0	0	0	1	0
1	1	1	1	0	0	0	1

A continuació, es mostren els circuits a dos nivells corresponents a les quatre funcions de sortida.

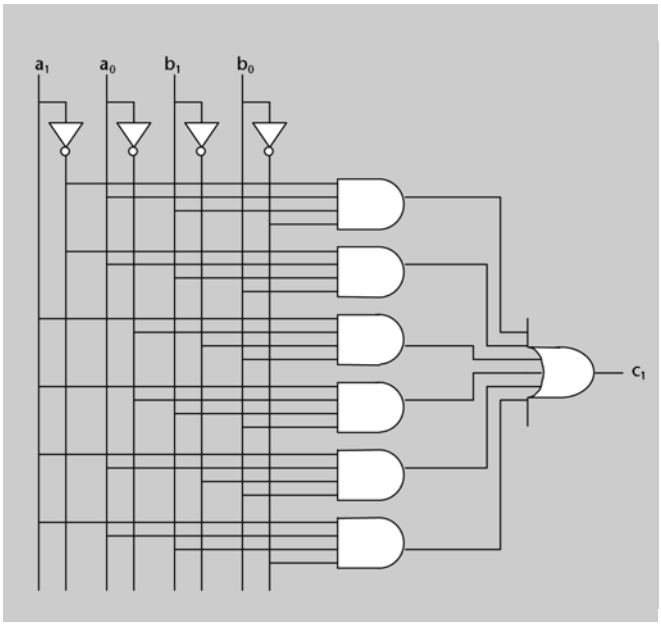
$c_3 = a_1'\ a_0\ b_1\ b_0' + a_1'\ a_0\ b_1\ b_0 + a_1\ a_0'\ b_1'\ b_0 + a_1\ a_0\ b_1'\ b_0.$



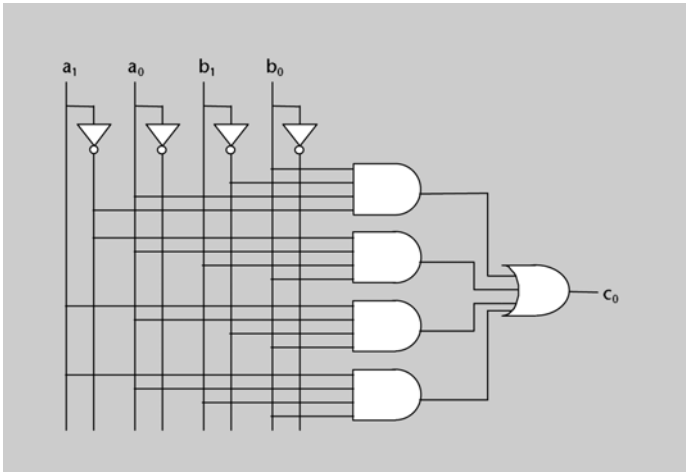
$c_2 = a_1' \, a_0 \, b_1 \, b_0' + a_1' \, a_0 \, b_1 \, b_0 + a_1 \, a_0' \, b_1' \, b_0 + a_1 \, a_0' \, b_1 \, b_0' + a_1 \, a_0 \, b_1' \, b_0$.



$c_1 = a_1' \, a_0 \, b_1 \, b_0' + a_1' \, a_0 \, b_1 \, b_0 + a_1 \, a_0' \, b_1' \, b_0 + a_1 \, a_0' \, b_1 \, b_0 + a_1 \, a_0 \, b_1' \, b_0 + a_1 \, a_0 \, b_1 \, b_0'$.



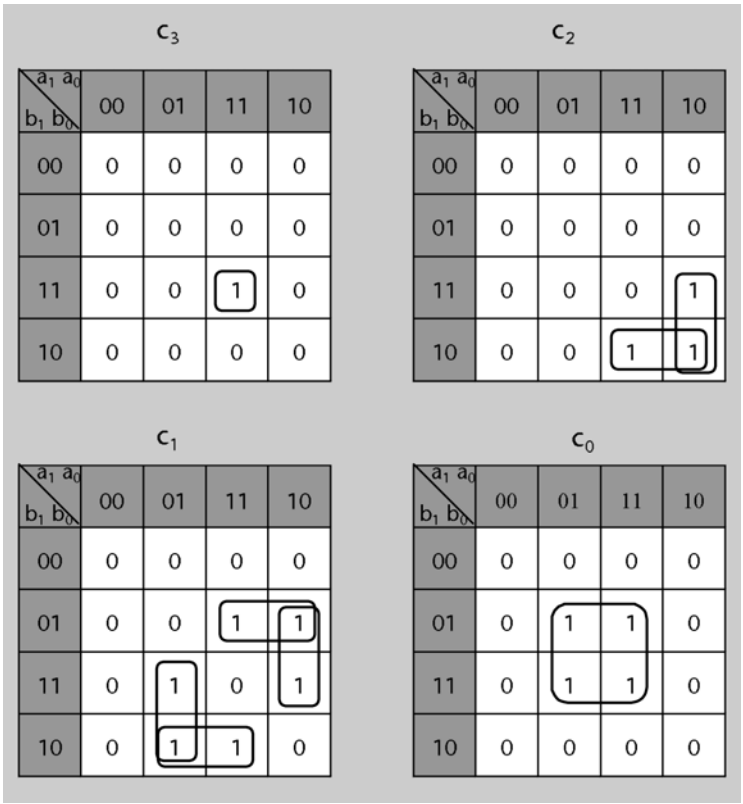
$c_0 = a_1' \, a_0 \, b_1' \, b_0 + a_1' \, a_0 \, b_1 \, b_0 + a_1 \, a_0 \, b_1' \, b_0 + a_1 \, a_0 \, b_1 \, b_0$.



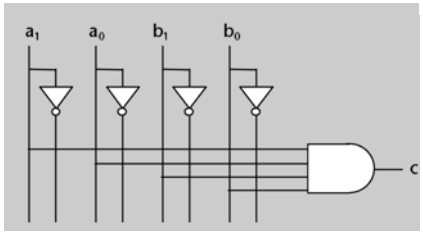
4. Tenim la taula de veritat següent:

a_1	a_0	b_1	b_0	c_3	c_2	c_1	c_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0
0	0	1	0	0	0	0	0
0	0	1	1	0	0	0	0
0	1	0	0	0	0	0	0
0	1	0	1	0	0	0	1
0	1	1	0	0	0	1	0
0	1	1	1	0	0	1	1
1	0	0	0	0	0	0	0
1	0	0	1	0	0	1	0
1	0	1	0	0	1	0	0
1	0	1	1	0	1	1	0
1	1	0	0	0	0	0	0
1	1	0	1	0	0	1	1
1	1	1	0	0	1	1	0
1	1	1	1	1	0	0	1

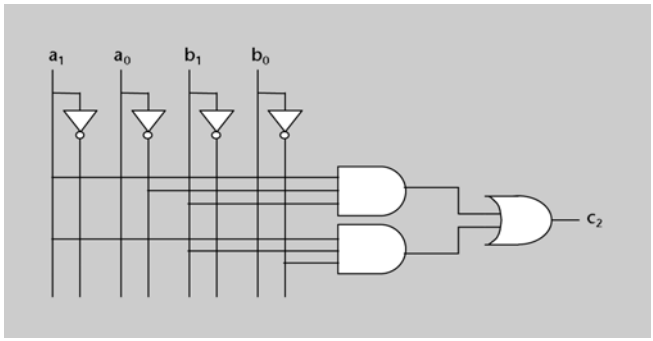
A continuació, simplificarem per Karnaugh cadascuna de les funcions de sortida, i després implementarem el circuit.



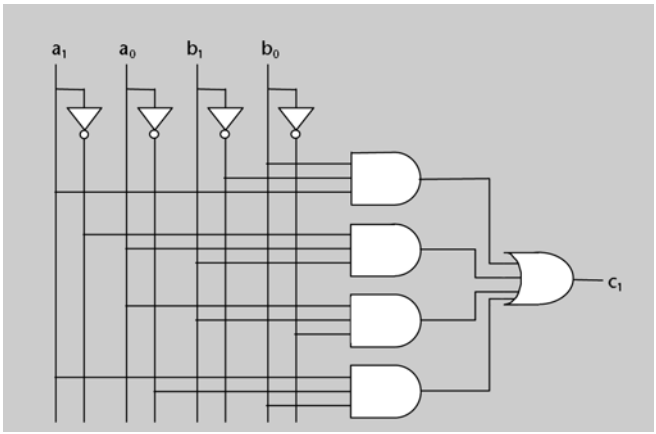
$c_3 = a_1\, a_0 b_1 b_0$.



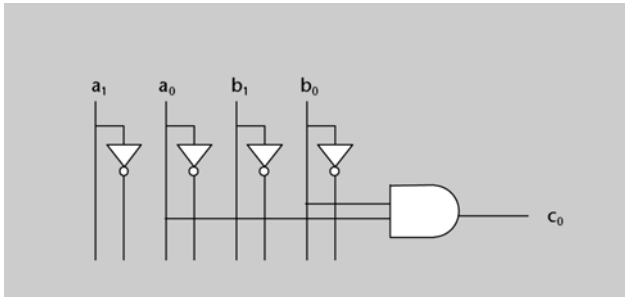
$c_2 = a_1a_0'b_1 + a_1b_1b_0'$.



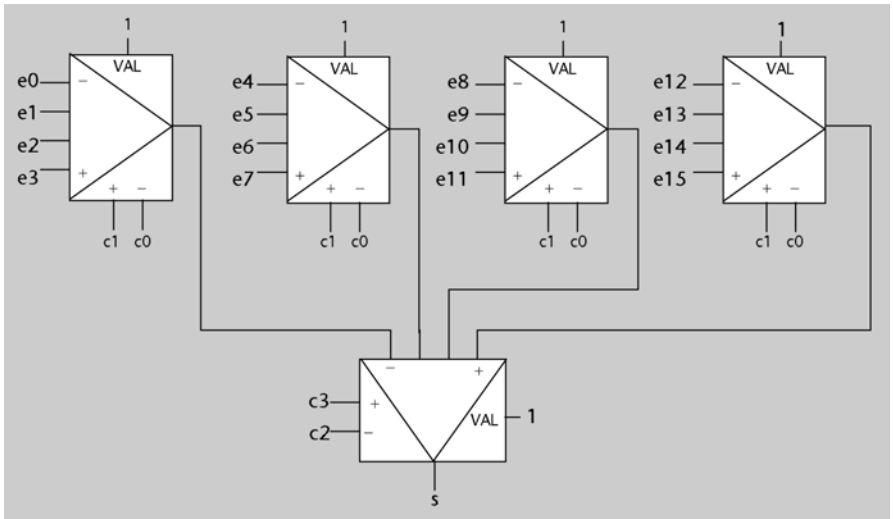
$c_1 = a_1b_1'b_0 + a_1'a_0b_1 + a_0b_1b_0' + a_1a_0'b_0$.



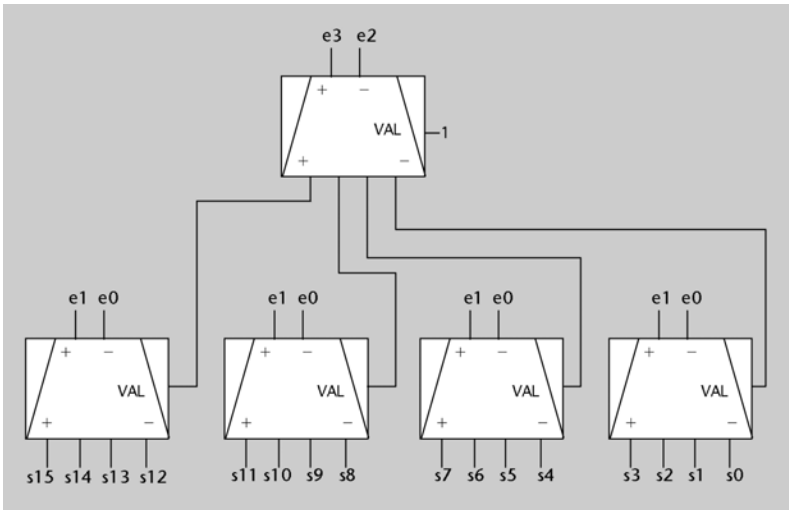
$c_0 = a_0b_0$.



5.



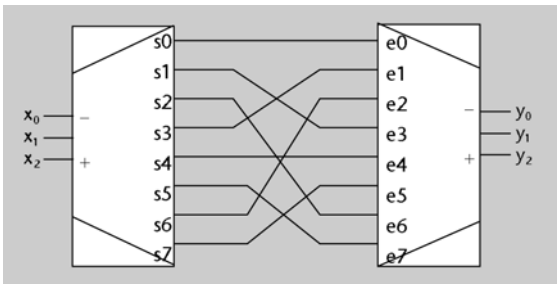
6.



7. La taula de veritat és la següent:

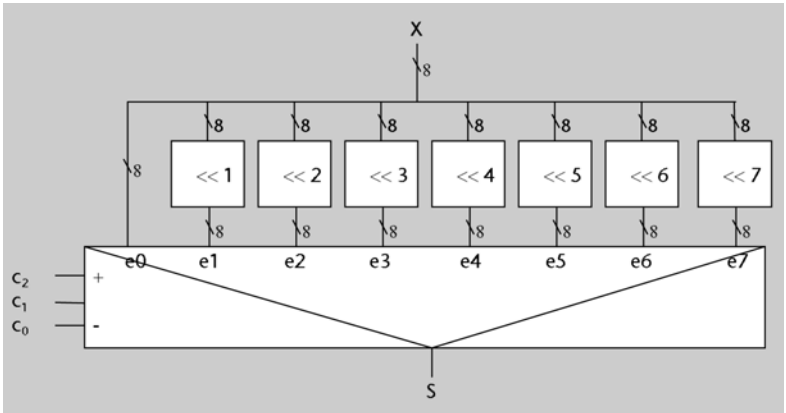
X	x_2	x_1	x_0	y_2	y_1	y_0	Y=3X mod 8
0	0	0	0	0	0	0	0
1	0	0	1	0	1	1	3
2	0	1	0	1	1	0	6
3	0	1	1	0	0	1	1
4	1	0	0	1	0	0	4
5	1	0	1	1	1	1	7
6	1	1	0	0	1	0	2
7	1	1	1	1	0	1	5

A continuació se'n mostra el disseny.



8.

a)



b) Tenint en compte que una operació de decalatge a l'esquerra d'un bit és equivalent a multiplicar per 2, si decalem X n bits, aquesta operació equival a $S = X \cdot 2^n$.

c) Si $C = 010$, el decalatge correspon a l'operació de multiplicar per quatre. Si X és natural, el rang representable amb vuit bits és [0..255]. Per tant, es produeix sobreeiximent quan $X \geq 256 / 4 = 64$.

Si X és enter, el rang de nombres representables és [–128..127]. Per tant, es produeix sobreeiximent quan $X >= 128 / 4 = 32$ o bé quan $X < -128 / 4 = -32$.

d) Si X és un nombre natural, els bits de més pes que s'afegeixen al nombre quan aquest es desplaça a la dreta han de valer 0. Per tant, s'ha d'usar un decalador lògic.

Si X és un nombre enter, els bits de més pes que s'afegeixen al nombre quan aquest es desplaça a la dreta han de valer el mateix que el bit de més pes del nombre, per tal de conservar-ne el signe. Per tant, s'ha d'usar un decalador aritmètic.

e) Un decalador a la dreta no computa la funció $X / 2^n$, sinó la funció $\lfloor X / 2^n \rfloor$. Per tant, si s'usa per a calcular $X / 2^n$, el resultat no serà exacte si algun dels bits que es perden en desplaçar a la dreta val 1. Dit d'una altra manera, si el desplaçament és de k bits ($C = k$), el resultat és inexacte quan tenim el següent:

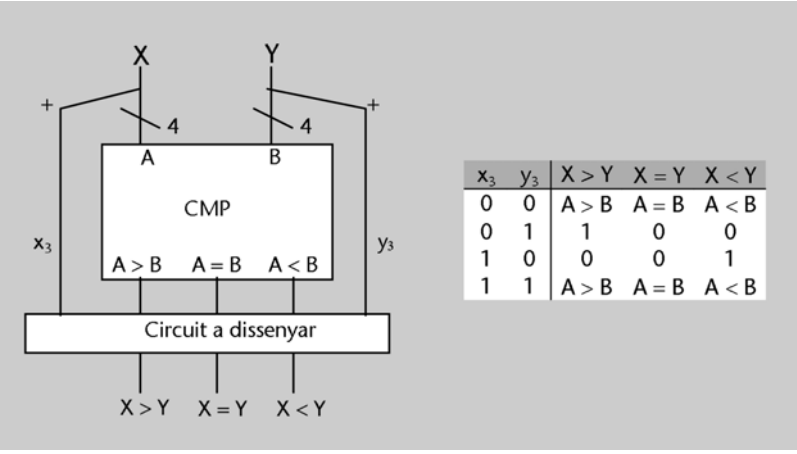
$$X \bmod 2^k \neq 0.$$

9. El resultat de la comparació depèn dels signes dels dos nombres (que vénen determinats pel seu bit de més pes: 1 per als negatius, 0 per als positius o zero).

Si els dos nombres són de signes diferents, llavors el més gran és el positiu.

Quan els dos nombres són del mateix signe, llavors els podem comparar usant un comparador de nombres naturals. Fixem-nos que el resultat serà correcte també en el cas que tots dos nombres siguin negatius, perquè si $X > Y$, interpretant X i Y en complement a dos, llavors també es compleix que $X > Y$ si els interpretem en binari. Per exemple, si comparem $X = -1$ (1111) amb $Y = -2$ (1110) el comparador trauria un 1 per la sortida $X > Y$.

A continuació, es mostra un primer esquema del circuit, i la taula de veritat que descriu el comportament de la part del circuit que queda per implementar, d'acord amb el signe de tots dos nombres.



x_3	y_3	X > Y	X = Y	X < Y
0	0	A > B	A = B	A < B
0	1	1	0	0
1	0	0	0	1
1	1	A > B	A = B	A < B

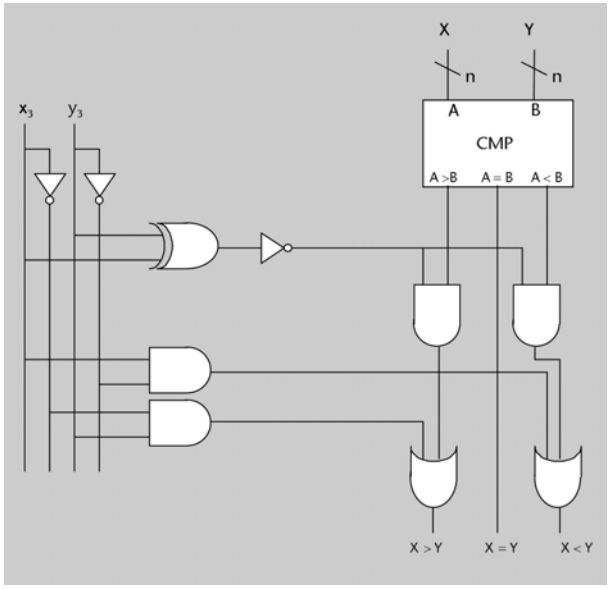
D'aquestes taules es dedueixen les igualtats següents:

$[X > Y] = x_3' y_3 + (x_3 \oplus y_3)' \cdot [A > B]$,

$[X < Y] = x_3 y_3' + (x_3 \oplus y_3)' \cdot [A < B]$,

$[X = Y] = [A = B]$.

A continuació, es mostra el circuit complet:



10. a) La taula de veritat d'un *full adder* és la següent:

a_i	b_i	c_i	c_{i+1}	s_i
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

L'expressió en suma de productes de c_{i+1} és la següent:

$$c_{i+1} = a_i' \, b_i \, c_i + a_i \, b_i' \, c_i + a_i \, b_i \, c_i' + a_i \, b_i \, c_i.$$

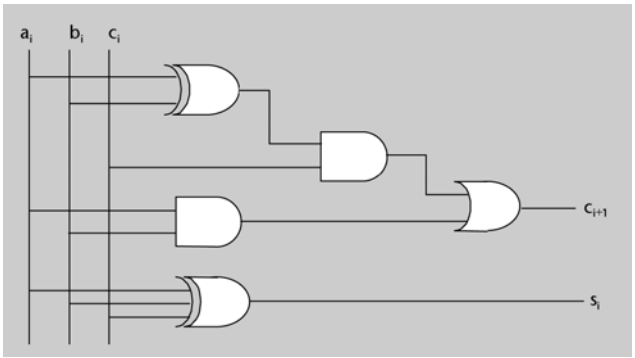
Podem treure factor comú en els dos primers i els dos últims termes, i obtenim el següent:

$$c_{i+1} = (a_i' \, b_i + a_i \, b_i') \, c_i + a_i \, b_i \, (c_i' + c_i) = (a_i \oplus b_i) \, c_i + a_i \, b_i.$$

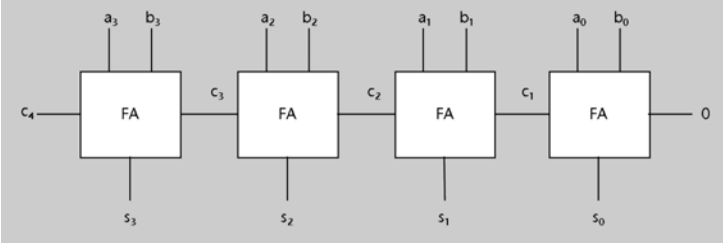
Pel que fa a s_i, veiem que quan $a_i = 0$ val $(b_i \oplus c_i)$ i quan $a_i = 1$ val $(b_i \oplus c_i)'$. Això es pot expressar de la manera següent:

$$s_i = a_i' \, (b_i \oplus c_i) + a_i \, (b_i \oplus c_i)' = a_i \oplus b_i \oplus c_i.$$

El circuit intern d'un *full adder*, que implementa aquestes expressions, es mostra en la figura següent.



b) A continuació, es mostra com s'encadenen quatre *full adders* per tal d'aconseguir un sumador de nombres de quatre bits. Com es pot veure en la figura, en el bit de ròssec d'entrada, c_0, s'hi ha de connectar un 0 per tal que la suma sigui correcta.



11. Es tracta de dissenyar un circuit que faci el següent:

Si $s'/r = 0$, llavors $R = A + B$.
Si $s'/r = 1$, llavors $R = A - B = A + B' + 1$.

Podem utilitzar un sumador de nombres de quatre bits per a fer aquest circuit.

– Connectarem el nombre A a una entrada.

– En l'altra entrada connectarem el número B o el número B', segons si el senyal s'/r val 0 o 1 respectivament. Recordem aquesta propietat de la funció XOR:

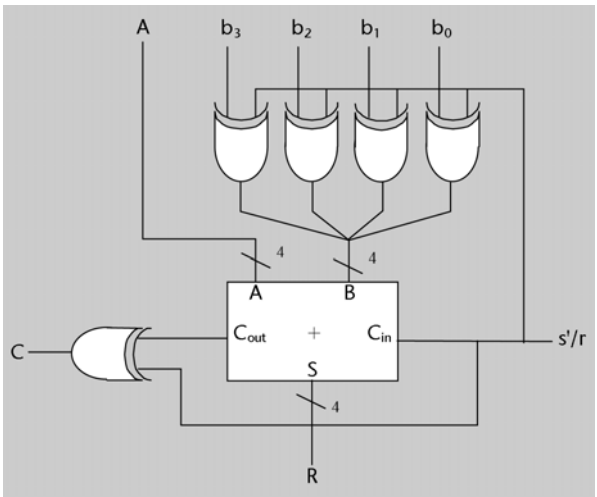
$$0 \oplus x = x,\ 1 \oplus x = x'.$$

Per tant, per a obtenir B o B' segons s'/r, podem fer una XOR de s'/r amb cadascun dels bits de B.

– Connectarem un 0 a l'entrada de ròssec si $s'/r = 0$, o bé un 1 si $s'/r = 1$. Per tant, hi podem connectar directament el senyal s'/r.

Una altra opció per a seleccionar B o B' amb s'/r seria utilitzar un multiplexor.

El circuit sumador/restador que es descriu en els paràgrafs anteriors es mostra a continuació.



Quan es fa una suma, la sortida C coincideix amb l'últim bit de ròssec.

Quan es fa la resta mitjançant la suma del complementari més 1, el ròssec de l'últim bit és sempre la negació del que obtindríem si s'hagués fet directament la resta.

Per tant, quan $s'/r = 0$ es compleix que $C = C_{out}$, i quan $s'/r = 1$ es compleix que $C = C_{out}'$. Així, doncs, si utilitzem la mateixa propietat de la funció XOR que hem fet servir abans, obtenim el següent:

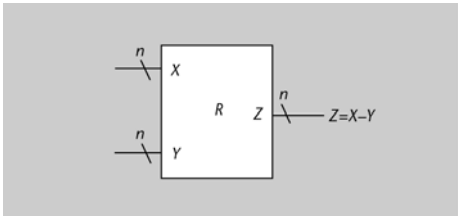
$$C = s'/r \oplus C_{out}.$$

12. Per a saber cap a quina planta s'han de moure els ascensors (és a dir, quin valor hem de donar al senyal *destinació*) necessitem codificar en binari (nombre natural) la planta des de la qual s'ha fet la crida. Això ho podem aconseguir mitjançant un codificador de 8 entrades i 3 sortides, amb b_i connectat a l'entrada e_i, les entrades e_7 i e_6 connectades a zero i amb els 3 bits de sortida ajuntats formant el bus *destinació*.

Per saber quin ascensor està més a prop de la planta de destinació podem restar la planta on hi ha cadascun dels ascensors (*plantaA* i *plantaB*) de la planta *destinació*. Per a fer la resta podem fer servir un sumador i aplicar la igualtat següent:

$$X - Y = X + (-Y) = X + Y' + 1$$

tal com es fa en l'activitat 56. Com que X i Y s'han de representar en complement a 2, afegirem un 0 com a bit de més pes a tots els operands de les restes. Representem el circuit que es dissenya en aquesta activitat mitjançant el bloc *R*, de la manera següent:



La resta de dos valors en el rang [0, 5] donarà un valor en el rang [–5, 5]. En tenim prou amb 4 bits per a codificar aquest rang en complement a 2, per tant n pot ser 4.

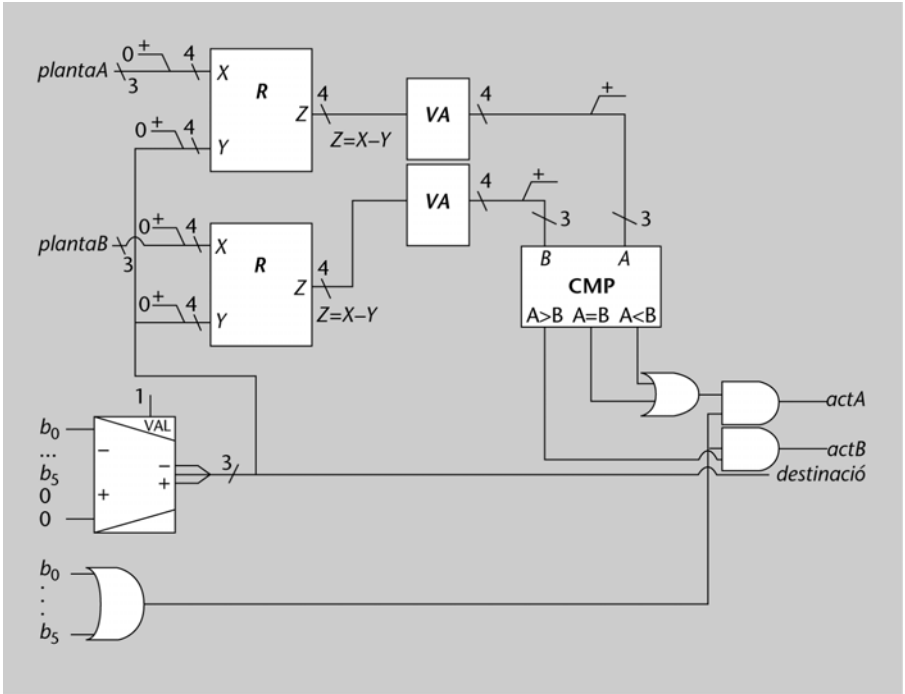
Les diferències *plantaX – destinació* poden ser tant positives com negatives, perquè l'ascensor *X* pot estar més amunt o més avall que la planta des d'on es fa la crida. La distància que busquem serà el valor absolut de la resta. Ho farem usant un bloc *VA*, que contindrà el circuit que s'ha dissenyat en l'activitat 57 (però de 4 bits). Aquest valor absolut no serà mai més gran que 5, de manera que es pot representar amb 3 bits. Per tant, no es produirà mai sobreeiximent en aquest bloc *VA*, és més, descartem un dels seus bits de sortida.

Un cop tenim ja les dues distàncies, podem saber quina de les dues és més gran mitjançant un comparador. N'hi ha prou amb un comparador de nombres de 3 bits.

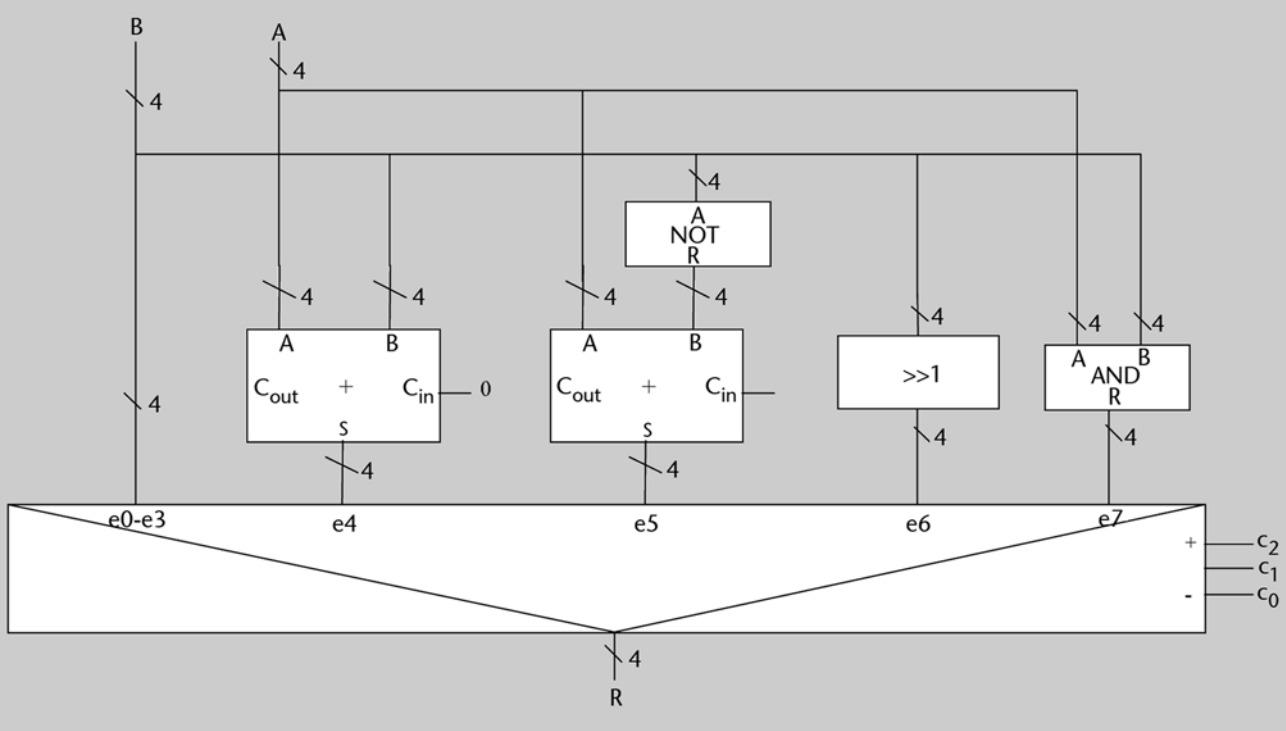
Si posem la distància a la qual es troba l'ascensor *A* a l'entrada *A* del comparador i l'altra distància a l'entrada *B*, sabem que quan la sortida $A < B$ del comparador sigui 1 haurem de posar el senyal *actA* a 1. Anàlogament, quan $A > B$ sigui 1, haurem de posar a 1 el senyal *actB*. En cas que $A = B$ sigui 1, és a dir, que tots dos ascensors estiguin a la mateixa distància, hem de posar a 1 *actA* (és a dir, $actA = (A < B) + (A = B)$.

Com que els senyals *actA* i *actB* s'han de mantenir a 0 mentre no es premi cap botó, cal fer un producte lògic (AND) entre aquestes sortides del comparador i un senyal que valgui 0 sempre que no s'hagi premut cap botó. Per exemple, el podem obtenir fent una suma lògica (OR) amb tots els bits b_i.

El circuit complet queda de la manera següent:



13. El circuit d'aquesta UAL es mostra en la figura següent. Observem que l'entrada *B* es connecta a les quatre entrades de menys pes del multiplexor.



Bibliografia

Gajsky, D. D. (1997). *Principios de Diseño Digital*. Prentice Hall.

Hermida, R.; Corral, A. del; Pastor, E.; Sánchez, F. (1998). *Fundamentos de Computadores*. Madrid: Síntesis.