Índex

1 Sistemes de numeració

Les xifres d'un nombre en qualsevol sistema de numeració tenen un cert valor posicional. Quan escrivim per exemple, el nombre 234, en el sistema de numeració de base 10, el que volem dir és

$$234 = 2 \cdot 10^2 + 3 \cdot 10^1 + 4 \cdot 10^0 = 200 + 30 + 4$$

diem que aquest nombre està en base 10 perquè en el seu desenvolupament en potències aquestes són el nombre 10. Hem de tenir clar que quan treballem en una certa base n, disposem només d'n símbols per construir els nombres.

També podem expressar nombres decimals d'aquesta manera

$$37,819 = 3 \cdot 10^1 + 7 \cdot 10^0 + 8 \cdot 10^{-1} + 1 \cdot 10^{-2} + 9 \cdot 10^{-3}$$

En cada base el nombre de símbols a disposició és diferent, quan $n = 10$ els que podem fer servir per construir els nombres s'han de triar d'entre el conjunt

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

és important veure que el nombre 10, la base en sí, no hi és en aquesta llista.

1.1 Bases diferents de 10

Considerem ara la base 2 de numeració que té únicament dos símbols $\{1, 0\}$. Prenem per exemple el nombre binari,

$$100101,11_2$$

fem servir el subíndex per indicar en quina base està el nombre. Quin nombre en base 10 representa? Per passar un nombre (des de qualsevol base) a base 10, l'expressem com a suma de potències i calculem

$$\begin{aligned} 100101,11_2 &= 1 \cdot 2^5 + 0 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 + 1 \cdot 2^{-1} + 1 \cdot 2^{-2} \\ &= 32 + 0 + 0 + 4 + 1 + \frac{1}{2} + \frac{1}{4} = 37,75_{10} \end{aligned}$$

La base 8, també anomenada *octal*, es fa servir molt. Per exemple, el nombre 765_8, equival en base 10 a 501_{10}, veiem-ho

$$765_8 = 7 \cdot 8^2 + 6 \cdot 8^1 + 5 \cdot 8^0 = 448 + 48 + 5 = 501_{10}$$

Un altre sistema que s'utilitza encara més és l'hexadecimal, de base 16. Com que només hi ha 10 símbols diferents per les xifres i en necessitem 6 més, es fan servir les lletres (majúscules) A, B, C, D, E, F de forma que $F = 15$ i $A = 10$, per exemple

$$\begin{aligned} 6E4AB2_{16} &= 6 \cdot 16^5 + E \cdot 16^4 + 4 \cdot 16^3 + A \cdot 16^2 + B \cdot 16^1 + 2 \cdot 16^0 \\ &= 6 \cdot 1048576 + 14 \cdot 65536 + 4 \cdot 4096 + 10 \cdot 256 + 11 \cdot 16 + 2 \cdot 1 \\ &= 7\,228\,082_{10} \end{aligned}$$

Segurament us heu adonat que quantes menys xifres té un sistema de numeració més en tenen els nombres que representen, per exemple abans hem vist

$$100101_2 = 37_{10}$$

en part, aquesta és la utilitat de treballar en hexadecimal, podem representar nombres molt grans amb menys xifres, i això, a nivell d'informàtica, suposa un estalvi important, a banda, 16 és divisible entre 8 i entre 2, cosa que fa, com veurem, molt fàcil el pas de binari i octal a hexadecimal.

1.2 Pas de base 10 a qualsevol altra

Amb aquests exemples hem vist com passar un nombre que es troba en un sistema de numeració arbitrari a decimal. Per fer el pas invers, farem servir un mètode de divisions successives.

Per exemple, per passar el nombre 379_{10} a binari farem

3 7 9	2										
1 7	1 8 9	1 8 9	2	9 4	2	4 7	2				
1 9		0 9	9 4	1 4	4 7	0 7	2 3				
1		1		0		1					

2 3	2										
0 3	1 1	1 1	2	5	2	2	2				
1		1	5	1	2	0	1				

aturem l'algorisme en el moment que el quocient és més petit que el divisor. Llavors, prenem *el darrer quocient* i tots els residus que ha aparegut *en ordre invers* per obtenir

$$379_{10} = 101111011_2$$

Per passar de decimal a octal fem el mateix però ara dividint per 8. Per exemple, per passar el mateix nombre d'abans, 379 a base 8,

$$\begin{array}{rl|l} 3\ 7\ 9 & 8 & \\ \hline 5\ 9 & 4\ 7 & 4\ 7 \mid 8 \\ \hline 3 & & 7 \mid 5 \end{array}$$

prenem el darrer quocient i els residus en ordre invers a com s'han obtingut

$$379_{10} = 573_8$$

Com ho faríem per passar per exemple 43251_6 a base 3? El millor en aquests casos és passar a base 10 i després a la nova base.

1.3 El problema de la precisió i els nombres binaris

Tornant a la representació en binari, com ho fem pels nombres racionals? Per exemple, com podem representar en binari el nombre $9,7_{10}$?

Per la part entera fem el mateix d'abans, divisions successives i ordenar residus en ordre invers a com s'obtenen

$$\begin{array}{r|l|r|l|r|l} 9 & 2 & 4 & 2 & 2 & 2 \\ \hline 1 & 4 & 0 & 2 & 0 & 1 \end{array}$$

per tant, de moment sabem $9_{10} = 1001_2$

Ara considerem l'algorisme següent;

$$\begin{array}{c} 0,7 \times 2 = 1,4 \geq 1 \Rightarrow 1 \\ 0,4 \times 2 = 0,8 < 1 \Rightarrow 0 \\ 0,8 \times 2 = 1,6 \geq 1 \Rightarrow 1 \\ 0,6 \times 2 = 1,2 \geq 1 \Rightarrow 1 \\ 0,2 \times 2 = 0,4 < 1 \Rightarrow 0 \\ - - - - - - - - - - \\ 0,4 \times 2 = 0,8 < 1 \Rightarrow 0 \end{array}$$

prenem la part decimal i multipliquem per 2, si el resultat és més gran o igual a 1, prenem 1. Si el resultat és més petit que 1, prenem 0. Seguim amb la part decimal del nombre resultat del primer producte. Arribats a cert punt, pot ser que els resultats es comencin a repetir cíclicament, és a dir, el procés no acaba. Això representa els nombres racionals. Si el procés no s'acaba i no es repeteix, tenim entre mans un nombre irracional.

Associem a $0,7_{10}$ el nombre binari decimal obtingut prenent els uns i zeros de dalt a baix

$$0,7_{10} \rightarrow 0.10110\,(0110)$$

finalment

$$9,7_{10} = 1001,10110$$

En contrast amb el que hem vist en aquest exemple, per qualsevol nombre racional que provingui d'una divisió per una potència de 2, l'algorisme serà finit, i diem que la representació de tal nombre decimal en binari és exacta. Per exemple el nombre $\frac{19}{8} = 2,375_{10}$ en binari es troba com

$$2_{10} = 10_{10}$$

i

$$\begin{gathered} 0,375 \times 2 = 0,75 < 1 \Rightarrow 0 \\ 0,75 \times 2 = 1,5 \geq 1 \Rightarrow 1 \\ 0,5 \times 2 = 1 \geq 1 \Rightarrow 1 \end{gathered}$$

de forma que

$$2,375_{10} = 10,011$$

La manca de precisió que es pot donar al treballar amb nombres binaris pot ser molt important. Prenem com exemple el nombre $3,00390625_{10}$ que es pot representar en forma exacta per $11,000000001_2$, si augmentem aquest nombre decimal mínimament, convertint-lo en $11,000000010_2$ aquest nombre en base 10 és $3,0078125_{10}$, és a dir que tots els nombres reals entre $3,00390625_{10}$ i $3,0078125_{10}$ es representen pel mateix nombre, per tant, hi ha 390625 nombres que no es poden representar fent servir la mateixa quantitat de dígits a la part decimal.

1.4 Més sobre les bases binaria, octal i hexadecimal

Considerem la taula

Binari-Octal	
Dígit octal	Dígit binari
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

Aquesta taula ens permet passar un nombre en binari directament a octal. Sigui per exemple el nombre 1100110_2 per passar-lo a octal l'escrivim fent grups de tres amb les seves xifres

$$1\,100\,110$$

afegim zeros *no significatius* per poder tenir tots els grups complets

$$001\,100\,110$$

ara llegim a la taula i *traduïm* directament cada grup de tres en binari per la corresponent xifra en octal

$$1\,4\,6$$

i ja el tenim, $1100110_2 = 146_8$. Per passar de base 8 a binari fem el mateix procés al revés. Traduïm cada xifra del nombre en base 8 al triplet corresponent de la taula. Per exemple, el nombre 45_8 s'escriu en base 2 com

$$100\,101 = 100101_2$$

Una cosa semblant succeeix amb la base 16.

Binari-Hexadecimal	
Dígit hexadecimal	Dígit binari
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
A	1010
B	1011
C	1100
D	1101
E	1110
F	1111

Sigui ara el nombre 110110110_2. Per passar-lo a hexadecimal l'escrivim fent grups de quatre amb les seves xifres

$$1\,1011\,0110$$

afegim zeros *no significatius* per poder tenir tots els grups complets

$$0001\,1011\,0110$$

ara llegim a la taula i *traduïm* directament cada grup de quatre en binari per la corresponent xifra en hexadecimal

$$1\,B\,6$$

i ja el tenim, $110110110_2 = 1B6_{16}$.

El canvi invers és fa de forma semblant al cas de la conversió d'octal a binari abans discutida.

1.5 El codi BCD

Anteriorment hem convertit nombres binaris a decimal. Recordeu que cada bit tenia un valor segons la posició que ocupava. Així, el nombre 10001_2

$$10001_2 = 1 \cdot 2^4 + 0 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 16 + 1 = 17_{10}$$

ens referirem a aquests nombres com a *binaris purs*. Amb el BCD (binary-coded-decimal) un nombre decimal no es converteix com un tot a binari, sino xifra a xifra. Per exemple, el nombre 1 de 17 és 0001 en binari (prenent 4 bits) i el 7 és 0111, llavors

$$17_{10} = 10001_2$$

com a binari pur, però

$$17_{10} = 0001\ 0111$$

en BCD.

1.6 Exercicis

1. Convertiu els següents nombres binaris a base 10
   (a) 100110
   (b) 110011
   (c) 110111
   (d) 1001,10
   (e) 101010110,001
2. Convertiu els següents nombres decimals a binari
   (a) 93
   (b) 647
   (c) 310
   (d) 131
   (e) 258,75
   (f) 1,625
   (g) 19,3125
3. Convertiu els següents nombres hexadecimals a decimal
   (a) 13
   (b) 65
   (c) 3F0
   (d) D0CE
   (e) 0,2
   (f) 12,9
   (g) F1,A
   (h) C8,D
4. Convertiu els següents nombres hexadecimals a binari, octal i decimal
   (a) 3,A2
   (b) 1B1,9
   (c) 6416213A,17B
5. Convertiu els següents nombres a hexadecimal

(a) $204231,134_5$

(b) 165433_7

6. Convertiu cada decimal a BCD.

(a) 62

(b) 25

(c) 274

(d) 284

(e) 42,91

(f) 5,014

7. Convertiu cada BCD a decimal

(a) 1001

(b) 101

(c) 110 0001

(d) 100 0111

(e) 11 0110, 1000

(f) 11 1000, 1000 1

2 Introducció als circuits lògics

2.1 Àlgebra de Boole

Un àlgebra de Boole és un conjunt $\mathcal{B}$ en el que s'han definit unes operacions *negació*, *suma lògica*, *producte lògic* que compleixen unes propietats

- Les operacions són internes

$$\overline{a} \in \mathcal{B} \qquad a+b \in \mathcal{B} \qquad a \cdot b \ \in \mathcal{B} \qquad \forall a, b \in \mathcal{B}$$

- Existeix element neutre

$$a+0=a \qquad a \cdot 1 = a \qquad \forall a \in \mathcal{B}$$

- Existeix element invers

$$a \cdot \overline{a} = 0 \qquad a + \overline{a} = 1 \qquad \forall a \in \mathcal{B}$$

- Se satisfà la propietat commutativa

$$a+b=b+a \qquad a \cdot b = b \cdot a \qquad \forall a, b \in \mathcal{B}$$

- Se satisfà la propietat distributiva

$$a \cdot (b+c) = (a \cdot b) + (a \cdot c) \qquad a + (b \cdot c) = (a+b) \cdot (a+c) \qquad \forall a, b \in \mathcal{B}$$

Els elements de $\mathcal{B}$ s'anomenen variables lògiques i només poden adoptar dos valors 1 ó 0, *cert* - *fals*.

2.2 Operacions lògiques

Definirem les operacions lògiques amb l'ajut de taules de veritat.

- Negació o complement (NOT)

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

- Suma lògica (OR)

a	b	$a+b$
0	0	0
0	1	1
1	0	1
1	1	1

- Producte lògic (AND)

a	b	$a \cdot b$
0	0	0
0	1	0
1	0	0
1	1	1

A banda d'aquestes que hem vist, a partir d'elles es construeixen d'altres

- Suma exclusiva (XOR)

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

- *NAND*

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

- *NOR*

a	b	$\overline{a + b}$
0	0	1
0	1	0
1	0	0
1	1	0

- *XNOR*

a	b	$\overline{a \oplus b}$
0	0	1
0	1	0
1	0	0
1	1	1

2.3 Lleis de De Morgan

Poden ser útils les expressions següents

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

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

2.4 Funcions lògiques

Una funció lògica és qualsevol expressió que lligui variables lògiques a través de les operacions que hem vist. Les funcions lògiques tenen un *valor de veritat* en funció dels valors d'entrada de les variables. La forma d'obtenir aquest valor és a través d'una taula de veritat.

Com a exemple trobem la taula de veritat de la funció

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

a	b	c	$\overline{c}$	ab	$ab + \overline{c}$
0	0	0	1	0	1
0	0	1	0	0	0
0	1	0	1	0	1
1	0	0	1	0	1
0	1	1	0	0	0
1	0	1	0	0	0
1	1	0	1	1	1
1	1	1	0	1	1

2.5 Exercicis

1. Construïu la taula de la veritat de les següents funcions lògiques.
   (a) $f(a,b) = a \cdot b + a$
   (b) $f(a,b) = (a \oplus b) \cdot \overline{b}$
   (c) $f(a,b) = \overline{(\overline{a} + b)} \oplus (a \cdot \overline{b})$
   (d) $f(a,b,c) = (a \cdot b) + c$
   (e) $f(a,b,c) = \overline{(a \cdot b) \oplus c}$
   (f) $f(a,b,c,d) = \overline{\overline{(\overline{a} + b)} \oplus (c \cdot \overline{d})}$

2.6 Mètode de Karnaugh

Sovint obtindrem la taula de la veritat d'una funció lògica *desconeguda* que descriu el comportament d'un circuit lògic. Una vegada escrita la funció, s'haurà d'intentar simplificar. Per fer això l'àlgebra de Boole disposa d'una col·lecció de teoremes que es podrien fer servir. Aquesta manera de procedir és complexa i pot arribar a ser molt farragosa. En casos particulars de funcions fins a quatre variables és més senzill fer servir l'anomenat *mètode de Karnaugh*.

A partir de l'exemple de la secció **2.4** podem escriure la funció lògica que resulta de dues formes diferents. Hem de tenir en compte que el que obtindrem serà precisament la funció sobre la que treballem.

- A través de *minterms*. Ens fixarem en les combinacions de variables que donen 1 a la sortida i els escriurem com a suma de productes de les variables tenint en compte en quin estat es troben

$$f(a,b,c) = \overline{a}\overline{b}\overline{c} + \overline{a}b\overline{c} + a\overline{b}\overline{c} + ab\overline{c} + abc$$

- A través de *maxterms*. Ens fixarem en les combinacions de variables que donen 0 a la sortida i els escriurem com a productes de sumes de les variables tenint en compte en quin estat es troben

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

Les dues formes de procedir són equivalents i convé fer servir sempre la mateixa, típicament la primera. En ocasions molt especials (quan hi molts zeros i pocs 1 a la sortida de la funció) pot ser convenient fer servir la segona.

Prenent doncs la versió de *minterms*

$$f(a,b,c) = \overline{a}\overline{b}\overline{c} + \overline{a}b\overline{c} + a\overline{b}\overline{c} + ab\overline{c} + abc$$

El que farem és situar els uns i zeros en un diagrama com el següent

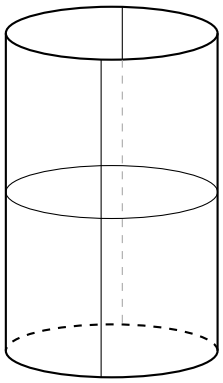
c \ ab	00	01	11	10
0	1	1	1	1
1	0	0	1	0

Noteu *l'ordre* en el que posem els valors de les parelles ab, ho hem fet en codi *Gray*, segons el qual només podem canviar un bit en cada pas

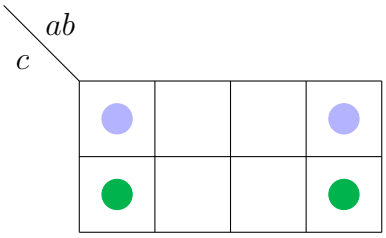
$$00 \rightarrow 01 \rightarrow 11 \rightarrow 10$$

Això és important per tal que funcioni el mètode que serveix per simplificar un cop hem omplert la taula amb els 1 i 0.

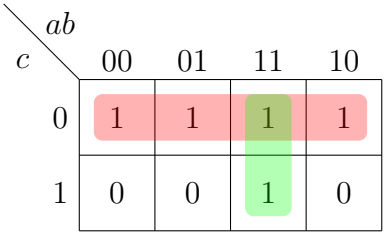
El següent pas és fer grups d'uns, el més grans possible tenint en compte que el nombre d'uns agrupats ha de ser una potència de 2. Els grups es poden solapar, han de tenir forma rectangular i finalment hem de tenir en compte la *topologia* associada al diagrama. En l'exemple que ens ocupa, de tres variables, el diagrama té la topologia d'un cilindre,



de forma que es poden considerar en contacte *extern* les cel·les assenyalades

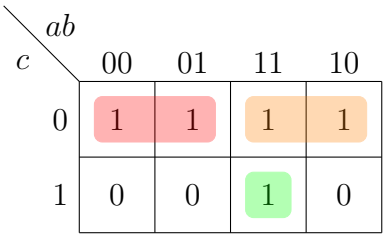


Aleshores, la forma òptima de fer els grups en aquest cas és

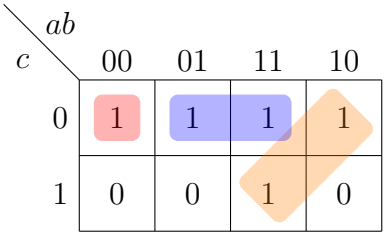


Hem pogut fer un grup de quatre 1 i un de dos. Recordem que el fet que es solapin no és problema.

Abans de seguir veiem un exemple d'agrupament no òptim



i un altre exemple d'un agrupament il·legal,



Reprenent el nostre cas, quin és el significat que tenen els grups que hem fet? La idea és que en cada grup hi ha variables que no canvien. El mètode de simplificació degut a Karnaugh ens diu que cada grup es pot representar precisament per l'estat de les variables que no canvien en el grup. És a dir, al grup de quatre que hem fet les variables a i b canvien i passen per tots els estats, però la variable c es manté, això si, negada. A l'altre grup, de dos 1 que hem fet les variables que no canvien són a i b. Podem escriure doncs

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

que és exactament la funció de partida. Recordem que las exercicis que haurem de fer la funció **no** serà coneguda, la única informació que tindrem serà la taula de la veritat i per tant, tindrà sentit l'esforç fet per a simplificar-la.

Ens podríem demanar a què corresponen casos extrems com que la taula estigui plena d'1 o zeros. En aquest cas es diu que la funció que volem simplificar és una *tautologia*, no es pot representar simplificada i té sempre valor veritat (1) o fals (0), independentment del valor de les variables que integren la funció.

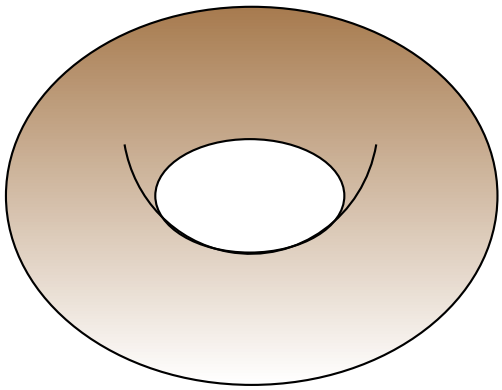
Veiem ara un altre exemple amb una taula ja donada, aquest cop amb quatre variables,

cd \ ab	00	01	11	10
00	1	0	0	1
01	0	1	1	1
11	0	1	1	1
10	1	0	0	1

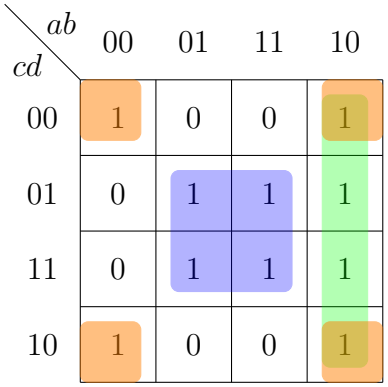
A priori podríem considerar els grups

cd \ ab	00	01	11	10
00	1	0	0	1
01	0	1	1	1
11	0	1	1	1
10	1	0	0	1

Però hem de tenir en compte que en el cas de quatre variables, el diagrama de Karnaugh corresponent té la topologia d'un tor



de forma que les vores esquerra i dreta estan "en contacte" de la mateixa manera que ho estan la vora superior i la inferior. De fet, les quatre cantonades del diagrama es troben en contacte. Amb això, la tria òptima de grups és



De forma que l'expressió simplificada de la funció és

$$f(a,b,c,d) = bd + a\overline{b} + \overline{b}\,\overline{d}$$

Aquesta relació entre el nombre de variables i la topologia associada a la taula resultant és el que fa que el mètode de Karnaugh no sigui gaire útil per cinc o més variables. En aquests casos, les superfícies que en resulten tenen autointerseccions, cosa que es tradueix en identificacions complicades de columnes interiors a la taula.

2.6.1 Els "Don't care"

A l'hora de resoldre un exercici d'electrònica digital que demani trobar la funció que codifica el comportament d'un dispositiu, podem trobar que algunes de les combinacions de les variables del problema que ens apareixen a la taula de veritat siguin irrealitzables, per exemple, si volem controlar una aixeta que s'ha d'obrir si el nivell d'aigua d'un dipòsit baixa fins a un cert valor, o tancar si en sobrepassa un altre, és clar que la condició "el nivell d'aigua es troba per sobre del límit superior i al mateix temps per sota de l'inferior" no es donarà mai, ja que és físicament impossible. En aquestes circumstàncies assignarem un valor X a la funció lògica per aquesta combinació de les variables. Aquest símbol es trasllada a la taula de veritat i es fa servir estratègicament com a 1 o 0 segons convingui, per poder fer una simplificació més eficaç. Per exemple, al mapa de Karnaugh que tenim a continuació

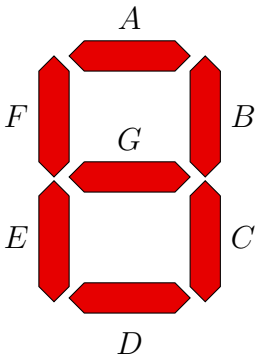
cd \ ab	00	01	11	10
00	X	1	1	1
01	0	1	1	0
11	0	1	X	0
10	0	X	1	X

cd \ ab	00	01	11	10
00	1	1	1	1
01	0	1	1	0
11	0	1	1	0
10	0	1	1	0

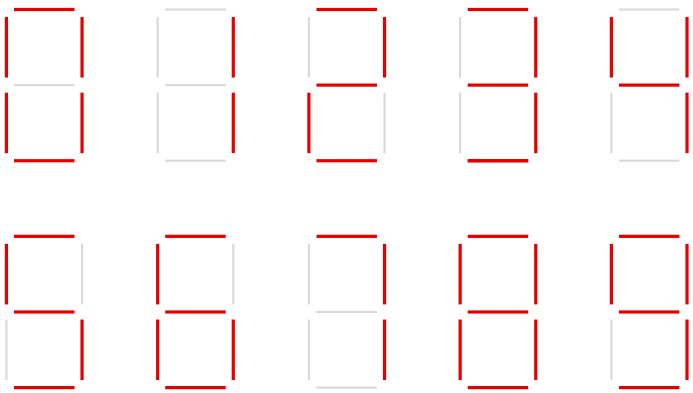
hem fet la tria que s'assenyala, ja que d'aquesta manera podem generar un grup de vuit 1's, i el 1 de la cantonada superior dreta pot quedar agrupat en un conjunt de 4 1's. Noteu que no té sentit activar a 1 el *don't care* de la cantonada inferior dreta, ja que només farem aparèixer un terme més a la funció, cosa que no té sentit. Més endevant veurem que l'expressió de la funció lògica simplificada té associada un circuit físic format per diverses portes lògiques. És en aquest sentit, que si dos circuits resolen el mateix problema i un fa servir menys portes lògiques que un altre, s'ha de considerar millor.

2.6.2 El decodificador BCD a display de 7 segments

Mitjançant un display de leds de set segments es poden reproduir els nombres del zero al nou.



Suposem que els diferents nombres tindran l'aparença següent



A partir d'aquí es vol implementar un codificador (es deixa com a exercici) que permeti mostrar al display un nombre enter entre 0 i 9.

El primer que s'ha de fer és una taula amb la representació en BCD de les xifres a codificar. Després, s'ha de completar la taula tenint en compte quins segments del display estaran activats i quins no. Finalment fem un diagrama de Karnaugh per cada segment i simplifiquem la funció obtinguda. Podem suposar que les variables BCD són *abcd* on *a* correspon al dígit de més pes i *d* al dígit de menys pes.

#	*abcd*	A	B	C	D	E	F	G
0	0000							
1	0001							

⋮

⋮

⋮

⋮

Verifiqueu les funcions lògiques obtingudes per cada segment a

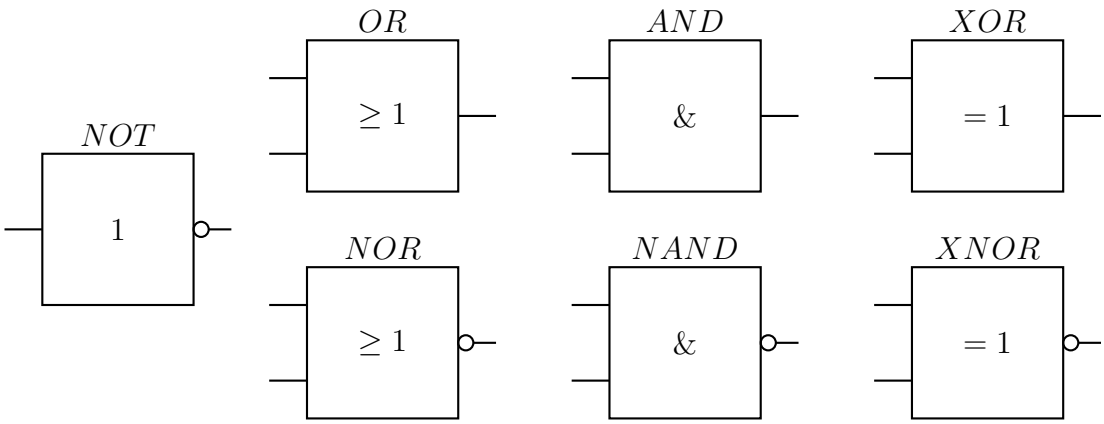
https://logic.ly/demo

2.7 Portes lògiques

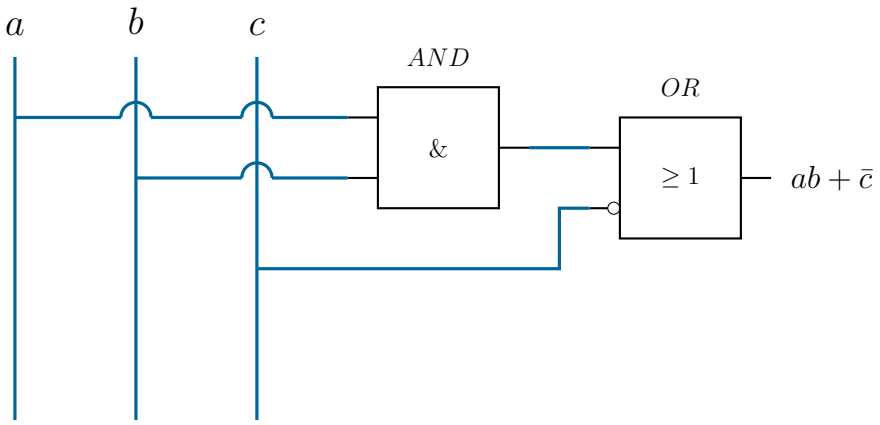
Les funcions lògiques que hem estudiat es poden implementar en la pràctica fent servir portes lògiques electròniques. Sense entrar en detall, comentem una sèrie de factors que s'han de tenir en compte a nivell pràctic quan es treballa amb elles,

- Les portes lògiques necessiten estar connectades a una font d'alimentació per funcionar.
- *Les entrades* de les portes s'activen amb dos valors diferents de la tensió, típicament $0\,V$ i $5\,V$, que representen el 0 i 1 lògics respectivament.
- *La sortida* d'una porta lògica pot prendre dos valors de la tensió, típicament $0\,V$ i $5\,V$, que representen el 0 i 1 lògics respectivament. En general cada porta lògica ofereix només una sortida.
- Sempre hi ha un cert retard entre el senyal d'entrada a una porta lògica i la seva resposta o sortida.

Les portes lògiques corresponen a les operacions lògiques que hem vist anteriorment. La simbologia usada a nivell europeu és



Reprenent la funció lògica de l'exemple de la secció **2.4**, l'esquema implementat amb portes lògiques és

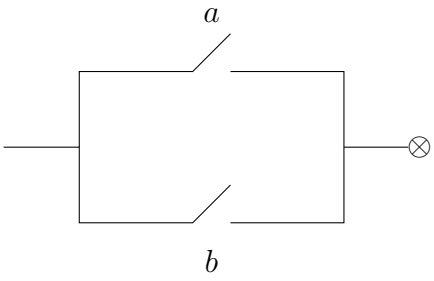


Noteu com hem *negat* la variable c just abans de connectar-la a la porta OR.

2.8 Diagrama de contactes

Els diagrames de contactes es basen en l'ús de circuits amb interruptors per codificar funcions lògiques. En tots ells obviarem la representació de la font d'alimentació (tal i com hem fet amb les portes lògiques).

- Funció lògica OR.

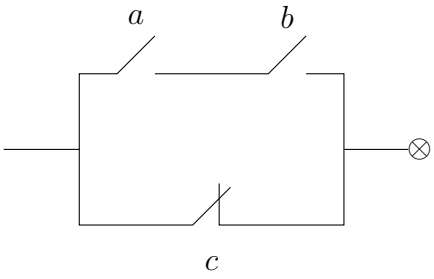


La *bombeta* ($\otimes$) s'encendrà sempre que l'interruptor a o el b o ambdós, estiguin tancats.

- Funció lògica AND.

En aquest cas cal que els dos contactes estiguin tancats per que passi corrent.

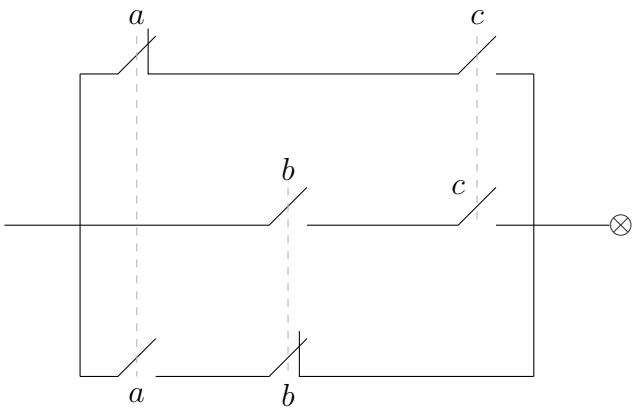
- Representem ara la funció $f(a, b, c) = ab + \bar{c}$



Noteu el símbol emprat pel contacte assignat a la variable negada $\bar{c}$.

- Representem finalment un cas més general, per exemple la funció

$$f(a, b, c) = \bar{a}c + bc + a\bar{b}$$



Noteu ara com es representen les variables lligades, encara que una d'elles es trobi negada.