Introducció als sistemes combinacionals

En els **sistemes digitals** les operacions es fan en codi binari, perquè el disseny de circuits per a operacions binàries és molt més senzill que per altres representacions

Els dispositius electrònics amb funcions booleanes s'anomenen **comportes lògiques**. Les comportes lògiques són circuits electrònics que operen amb una o més senyals d'entrada per produir una senyal de sortida.

En els sistemes combinacionals la sortida és únicament funció de la entrada, i per ser sistemes discrets, el seu comportament queda completament determinat definint la sortida resultant per a cada una de les combinacions de les dades d'entrada, en forma de les anomenades taules de veritat.

Altres sistemes digitals més complexos són els sistemes seqüencials, que tenen estats interns i la sortida no depèn únicament de les entrades, però només veurem sistemes combinacionals.

Algebra de Boole

Matemàtiques bàsiques per al disseny de sistemes digitals: Formulisme matemàtic per operar les funcions de commutació:

- Desenvolupada per Georges Boole al 1847 per problemes de lògica matemàtica
- Claude Shannon al 1939 l'aplica per primer cop a funcions de commutació

Definicions

- **Variable lògica:** variable que pot assolir únicament dos valors {(0,1), (L,H), (F,V)}
- **Funció lògica:** funció definida amb variables lògiques el resultat de la qual només pot assolir dos valors {(0,1), (L,H), (F,V)}
- **Àlgebra:** conjunt d'elements, S, format, com a mínim, per dos elements diferents, amb dues operacions internes, suma (+) i producte (•) (que anomenarem suma lògica i producte lògic). Els elements satisfan el principi de substitució.

Sistemes digitals seqüencials: Portes lògiques

NOT: Realitza una inversió a nivell d'entrada. Si a l'entrada hi ha un nivell alt (1), a la sortida hi haurà un nivell baix (0), i viceversa.

OR: La porta OR implementa la suma lògica de les entrades. És a dir, sempre que almenys a una de les entrades hi hagi un 1, a la sortida hi haurà un 1. Només quan a les dues entrades hi hagi un 0 a la sortida hi haurà un 0 també.

NOR: La porta NOR implementa la negació de la suma lògica de les entrades. És a dir, sempre que almenys a una de les entrades hi hagi un 1, a la sortida hi haurà un 0. Només quan a les dues entrades hi hagi un 0 a la sortida hi haurà un 1.

AND: La porta AND implementa la multiplicació lògica de les entrades. És a dir, sempre que almenys a una de les entrades hi hagi un 0, a la sortida hi haurà un 0. Només quan a les dues entrades hi hagi un 1 a la sortida hi haurà un 1 també.

NAND: La porta NAND implementa la negació de la multiplicació lògica de les entrades. És a dir, sempre que almenys a una de les entrades hi hagi un 0, a la sortida hi haurà un 1. Només quan a les dues entrades hi hagi un 1 a la sortida hi haurà un 0.

XOR: La porta XOR funciona de forma que donarà 1 a la sortida sempre que el valor de les entrades sigui **diferent**, 0 en cas contrari.

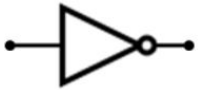
XNOR: La porta XOR funciona de forma que donarà 1 a la sortida sempre que el valor de les entrades sigui **iguals**, 0 en cas contrari.

Sistemes digitals seqüencials: Portes lògiques

AND		
Entrada A	Entrada B	Sortida
0	0	0
0	1	0
1	0	0
1	1	1

OR		
Entrada A	Entrada B	Sortida
0	0	0
0	1	1
1	0	1
1	1	1

XOR		
Entrada A	Entrada B	Sortida
0	0	0
0	1	1
1	0	1
1	1	0



NOT	
Entrada A	Entrada B
0	1
1	0

NAND		
Entrada A	Entrada B	Sortida
0	0	1
0	1	1
1	0	1
1	1	0

NOR		
Entrada A	Entrada B	Sortida
0	0	1
0	1	0
1	0	0
1	1	0

XNOR		
Entrada A	Entrada B	Sortida
0	0	1
0	1	0
1	0	0
1	1	1

Taula de veritat

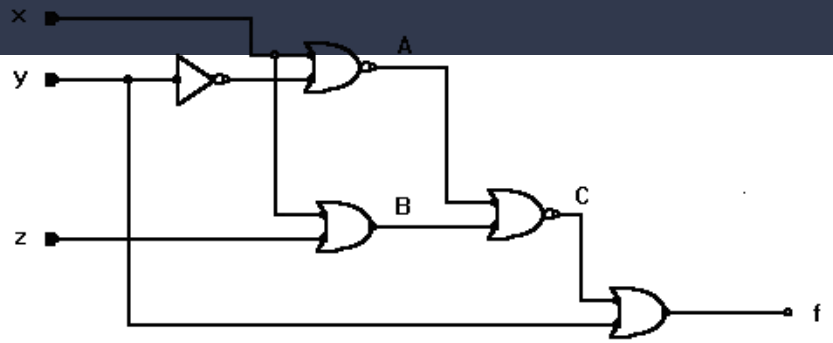
Taula de veritat: per una funció de n variables tenim una columna amb les 2^n combinacions d'1 i 0 que es poden formar i un altre columna amb el valor de la funció per aquestes entrades).

Dues funcions diferents tenen taules de veritat diferents.

- Per N variables hi ha 2^{2N} funcions de commutació.
  - Així per a 1 variable tenim 4 funcions possibles
  - Per 2 variables, 16 funcions possibles
  - Per 3 variables, 256 funcions possibles

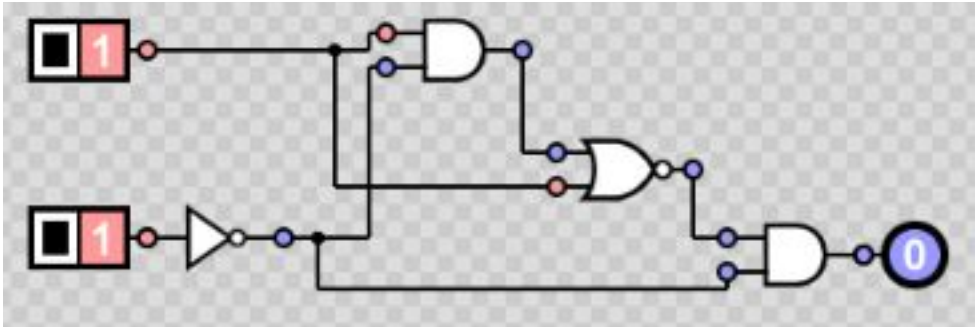
Aquest programari us servirà per simular taules i circuits lògics → AQUÍ

Exemple



X	Y	Z	/Y	X OR /Y	A=X NOR /Y	B = X OR Z	A OR B	C=A NOR B	F=C OR Y
0	0	0	1	1	0	0	0	1	1
0	0	1	1	1	0	1	1	0	0
0	1	0	0	0	1	0	1	0	1
0	1	1	0	0	1	1	1	0	1
1	0	0	1	1	0	1	1	0	0
1	0	1	1	1	0	1	1	0	0
1	1	0	0	1	0	1	1	0	1
1	1	1	0	1	0	1	1	0	1

Exemple



((A AND /B) NOR A) AND /B

A	B	/B	P1= A AND /B	P1 OR A	P2= /(P1 OR A)	P2 AND /B
0	0	1	0	0	1	1
0	1	0	0	0	1	0
1	0	1	1	1	0	0
1	1	0	0	1	0	0