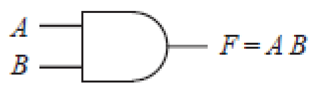
Lógica Combinacional

No existe memoria, la salida depende de la entrada en todo momento, a igual entrada igual salida

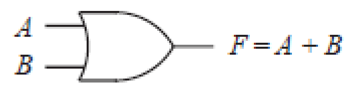
Compuertas básicas

A	B	F
0	0	0
0	1	0
1	0	0
1	1	1



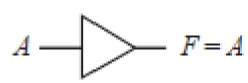
AND

A	B	F
0	0	0
0	1	1
1	0	1
1	1	1



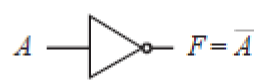
OR

A	F
0	0
1	1



Buffer

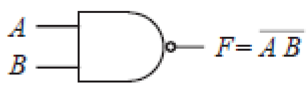
A	F
0	1
1	0



NOT (Inverter)

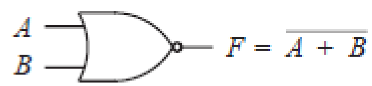
Compuertas básicas

A	B	F
0	0	1
0	1	1
1	0	1
1	1	0



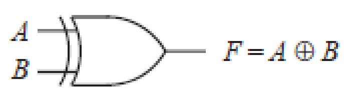
NAND

A	B	F
0	0	1
0	1	0
1	0	0
1	1	0



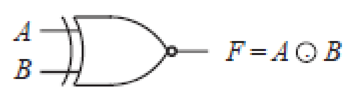
NOR

A	B	F
0	0	0
0	1	1
1	0	1
1	1	0



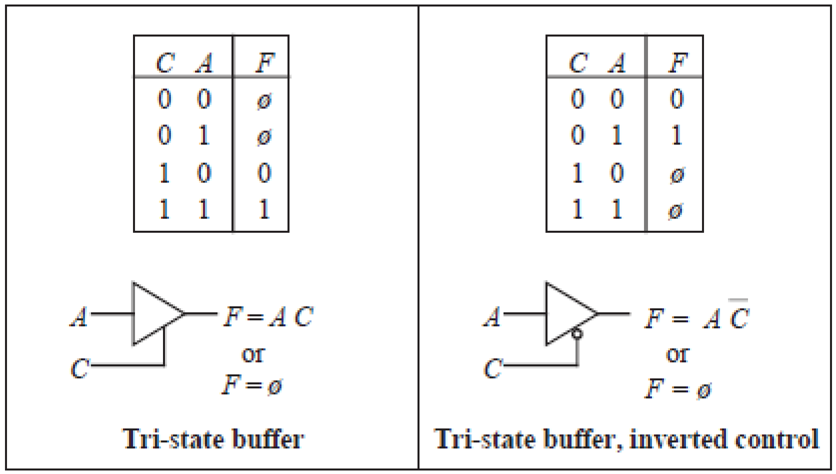
Exclusive-OR (XOR)

A	B	F
0	0	1
0	1	0
1	0	0
1	1	1



Exclusive-NOR (XNOR)

Compuertas básicas



C	A	F
0	0	ø
0	1	ø
1	0	0
1	1	1

Tri-state buffer

C	A	F
0	0	0
0	1	1
1	0	ø
1	1	ø

Tri-state buffer, inverted control

Algebra de Boole

Propiedad	**Producto**	**Suma**
Conmutativa	AB = BA	A+B = B+A
Distributiva	A(B+C) = AB+AC	A+BC = (A+B)(A+C)
Identidad	1.A = A	0+A = A
Complemento	AA’ = 0	A+A’ = 1
Teorema del cero y uno	0.A = 0	1+A = 1
Idempotencia	AA = A	A+A = A
Asociativa	A(BC) = (AB)C	A+(B+C) = (A+B)+C
Involución	A’’ = A	
Teorema DeMorgan	(AB)’ = A’+B’	(A+B)’ = A’B’
Teorema del concenso	AB+A’C+BC = AB+A’C	(A+B)(A’+C)(B+C) = (A+B)(A’+C)
Teorema de absorción	A(A+B) = A	A+AB = A

Primera Forma Canónica

- Suma de M productos de N variables
- M nro de productos = nro de ‘1’ a la salida
- N nro de términos de cada producto = nro de variables de entrada
- Cada producto forma un ‘1’ de la salida
- La suma de todos los productos forman todos los ‘1’ de la salida
- La salida es binaria, si no es ‘1’ es ‘0’
- Ideal cuando hay pocos ‘1’
- El resultado puede ser simplificado

Segunda Forma Canónica

- Producto de M sumas de N variables
- M nro de sumas = nro de ‘0’ a la salida
- N nro de términos de cada cada = nro de variables de entrada
- Cada suma forma un ‘0’ de la salida
- El producto de todos las sumas forman todos los ‘0’ de la salida
- La salida es binaria si no es ‘0’ es uno
- Ideal cuando hay pocos ‘0’
- El resultado puede ser simplificado

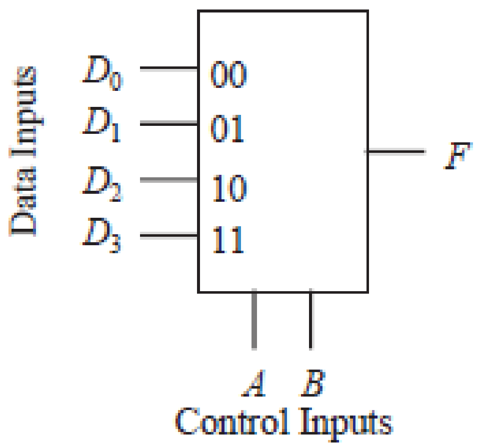
Simplificación

- Se intenta minimizar la complejidad del sistema
- Menor cantidad de componentes
- Menor cantidad de conexiones
- Menor costo
- Menor tamaño
- Menor producción de calor
- Mayor velocidad

Karnaugh

- Simplificación, menor cantidad de términos y mas chicos
- Reescribo la Tabla de Verdad con el código de Gray en el mapa
- Tomo grupos de 1 o 0 contiguos (1er o 2da Forma Canónica) bajo las premisas:
- Rectángulos o cuadrados
- Cantidad de elementos n = 2x con x entero
- Cantidad de elementos la mayor posible, mas simplificación
- Puedo tomar un termino varias veces (idempotencia)
- No tomar grupos redundantes
- Los extremos son contiguos (los términos que difieren en 1 bit)

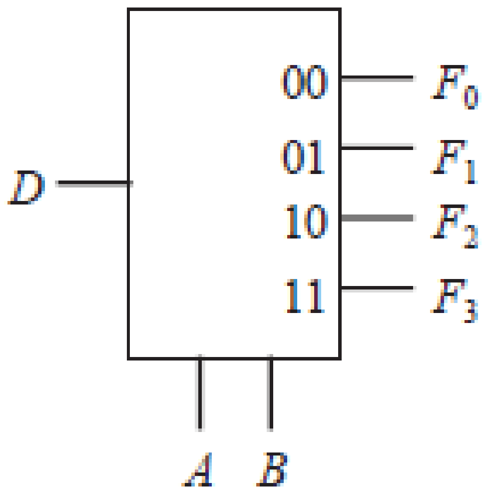
Multiplexores



A	B	F
0	0	D_0
0	1	D_1
1	0	D_2
1	1	D_3

$$F = \overline{A}\,\overline{B}\,D_0 + \overline{A}\,B\,D_1 + A\,\overline{B}\,D_2 + A\,B\,D_3$$

Demultiplexores

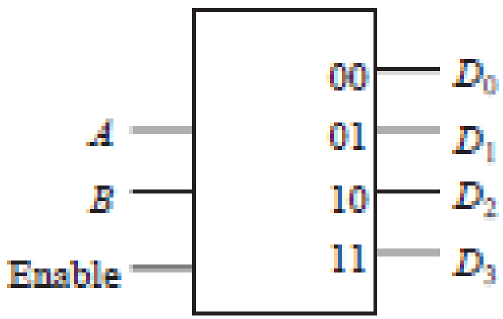


$$F_0 = D\,\overline{A}\,\overline{B} \qquad F_2 = D\,A\,\overline{B}$$

$$F_1 = D\,\overline{A}\,B \qquad F_3 = D\,A\,B$$

D	A	B	F_0	F_1	F_2	F_3
0	0	0	0	0	0	0
0	0	1	0	0	0	0
0	1	0	0	0	0	0
0	1	1	0	0	0	0
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1

Decodificadores

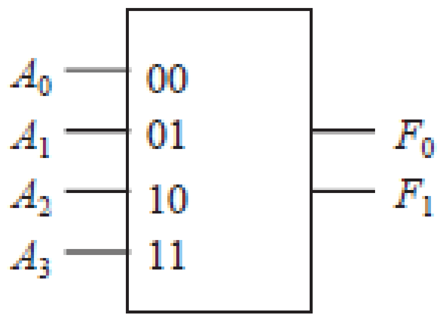


Enable = 1					
A	B	D_0	D_1	D_2	D_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

Enable = 0					
A	B	D_0	D_1	D_2	D_3
0	0	0	0	0	0
0	1	0	0	0	0
1	0	0	0	0	0
1	1	0	0	0	0

$D_0 = \overline{A}\,\overline{B}$ $\quad D_1 = \overline{A}\,B$ $\quad D_2 = A\,\overline{B}$ $\quad D_3 = A\,B$

Codificadores de prioridad

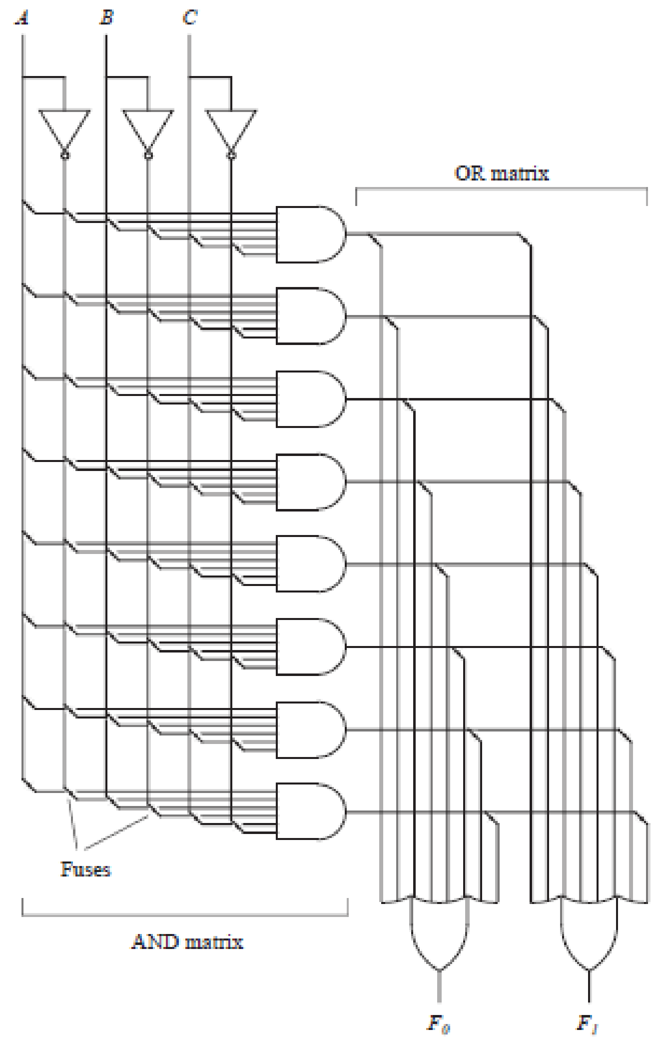


$$F_0 = \overline{A_0}\,\overline{A_1}\,A_3 + \overline{A_0}\,\overline{A_1}\,A_2$$

$$F_1 = \overline{A_0}\,\overline{A_2}\,A_3 + \overline{A_0}\,A_1$$

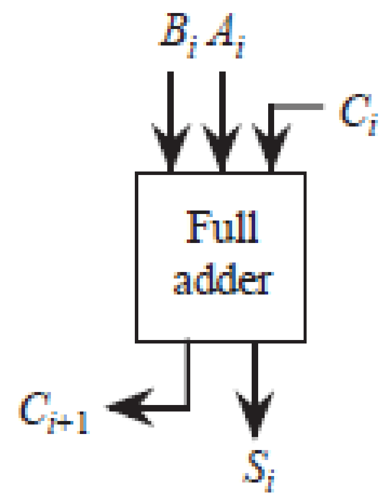
A_0	A_1	A_2	A_3	F_0	F_1
0	0	0	0	0	0
0	0	0	1	1	1
0	0	1	0	1	0
0	0	1	1	1	0
0	1	0	0	0	1
0	1	0	1	0	1
0	1	1	0	0	1
0	1	1	1	0	1
1	0	0	0	0	0
1	0	0	1	0	0
1	0	1	0	0	0
1	0	1	1	0	0
1	1	0	0	0	0
1	1	0	1	0	0
1	1	1	0	0	0
1	1	1	1	0	0

PROGRAMMABLE LOGIC ARRAYS (PLA)



Sumador

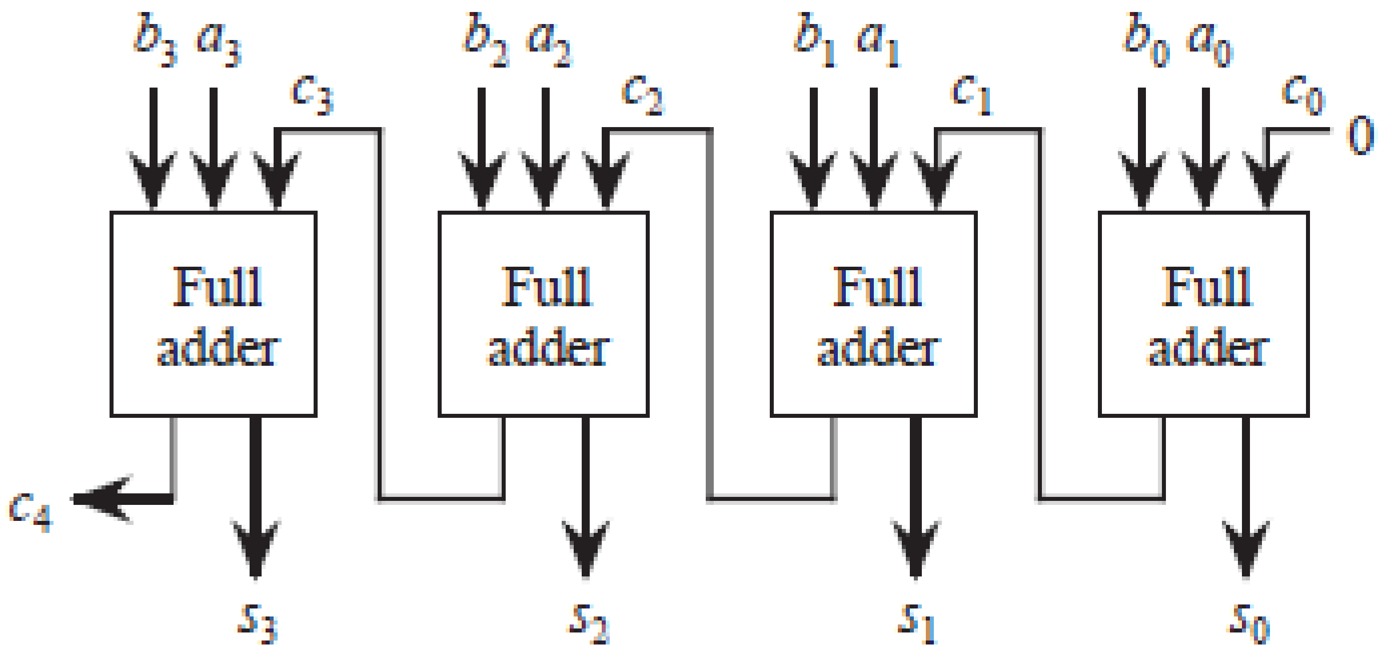
A_i	B_i	C_i	S_i	C_{i+1}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1



$$S_i = A_i'B_i'C_i + A_i'B_iC_i' + A_iB_i'C_i' + A_iB_iC_i = A \text{ xor } B \text{ xor } C$$

$$C_i = A_i'B_iC_i + A_iB_i'C_i + A_iB_iC_i' + A_iB_iC_i = AB + BC + AC$$

Sumador



b3 a3
c3
b2 a2
c2
b1 a1
c1
b0 a0
c0
0
Full adder
Full adder
Full adder
Full adder
c4
s3
s2
s1
s0