Organización del Computador 1
Lógica Digital 1: álgebra de Boole y compuertas

Dr. Marcelo Risk

Departamento de Computación
Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires

2017

Lógica digital

- La computadoras necesitan almacenar datos e instrucciones en memoria.
- Sistema binario: solo dos estados posibles.
- Porqué?
  - Es mucho más sencillo identificar entre sólo dos estados.
  - Es menos propenso a errores

Diseño de circuitos

- Circuitos que operan con valores lógicos:
  - Verdadero = 1
  - Falso = 0
- Idea: realizar diferentes operaciones lógicas y matemáticas combinando circuitos.

Álgebra de Boole

Figura: George Boole (1815-1864).

- George Boole, desarrolló un sistema algebraico para formular proposiciones con símbolos.
- Su álgebra consiste en un método para resolver problemas de lógica que recurre solamente a los valores binarios:
  - verdadero y falso.
  - on y off.
  - 1 y 0.
- tres operadores:
  - AND (y).
  - OR (y).
  - NOT (no).

Álgebra de Boole

- Las variables Booleanas sólo toman los valores binarios: 1 ó 0.
- Una variable Booleana representa un bit que quiere decir:
  - Binary digIT

Álgebra de Boole

- Las variables Booleanas sólo toman los valores binarios: 1 ó 0.
- Una variable Booleana representa un bit que quiere decir:
  - Binary digIT

Operadores básicos: **AND**

- Un operador booleano puede ser completamente descripto usando tablas de verdad.
- El operador **AND** es conocido como producto booleano (.):

X	Y	X AND Y
0	0	0
0	1	0
1	0	0
1	1	1

Operadores básicos: **OR**

- El operador **OR** es conocido como producto booleano (+):

X	Y	X OR Y
0	0	0
0	1	1
1	0	1
1	1	1

Operadores básicos: **NOT**

- El operador NOT se nota con una barra $\overline{X}$:

X	$\overline{X}$
0	1
1	0

Funciones booleanas

- Tabla de verdad de esta función $F(x, y, z) = x\bar{z} + y$
- El **NOT** tiene mayor precedencia que todos
- El **AND** mayor que el **OR**

x	y	z	$\bar{z}$	$x\bar{z}$	$x\bar{z}+y$
0	0	0	1	0	0
0	0	1	0	0	0
0	1	0	1	0	1
0	1	1	0	0	1
1	0	0	1	1	1
1	0	1	0	0	0
1	1	0	1	1	1
1	1	1	0	0	1

Identidades

Identidad	$1.A = A$	$0 + A = A$
Nula	$0.A = 0$	$1 + A = 1$
Idempotencia	$A.A = A$	$A + A = A$
Inversa	$A.\overline{A} = 0$	$A + \overline{A} = 1$
Conmutativa	$A.B = B.A$	$A + B = B + A$
Asociativa	$(A.B).C = A.(B.C)$	$(A + B) + C = A + (B + C)$
Distributiva	$A + B.C = (A + B).(A + C)$	$A.(B + C) = A.B + A.C$
Absorción	$A.(A + B) = A$	$A + A.B = A$
de Morgan	$\overline{A.B} = \overline{A} + \overline{B}$	$\overline{A + B} = \overline{A}.\overline{B}$

Identidades: ejemplo de aplicación

- Usando identidades booleanas podemos reducir esta función:
  $F(x,y,z) = (x+y).(x+\overline{y}).\overline{(x.\overline{z})}$

$(x+y)(x+\overline{y})(\overline{x}+z)$	de Morgan
$(xx+x\overline{y}+yx+y\overline{y})(\overline{x}+z)$	Distributiva
$(x+x\overline{y}+yx+0)(\overline{x}+z)$	Idempotencia e inversa
$(x+x(\overline{y}+y))(\overline{x}+z)$	Nula y distributiva
$(x)(\overline{x}+z)$	Inversa, identidad y nula
$x\overline{x}+xz$	Distributiva
xz	Inversa e identidad

Fórmulas equivalentes

- Varias fórmulas pueden tener la misma tabla de verdad:
  - Son lógicamente equivalentes.
- En general se suelen elegir las formas **canónicas**
  - Suma de productos:
    - $F_1(x, y, z) = xy + zx + yz$
  - Producto de sumas:
    - $F_2(x, y, z) = (x + y)(z + x)(y + z)$

Suma de productos

- Es fácil convertir una función a una suma de productos usando la tabla de verdad.
- Elegimos los valores que dan 1 y hacemos un producto (AND) de la fila (negando si aparece un 0)?.
- Luego sumamos todo (OR):

	x	y	z	$x\bar{z}+y$	
	0	0	0	0	
	0	0	1	0	
→	0	1	0	1	←
→	0	1	1	1	←
→	1	0	0	1	←
	1	0	1	0	
→	1	1	0	1	←
→	1	1	1	1	←

$$F(x,y,z) = (\bar{x}y\bar{z}) + (\bar{x}yz) + (x\bar{y}\bar{z}) + (xy\bar{z}) + (xyz)$$

Circuitos booleanos

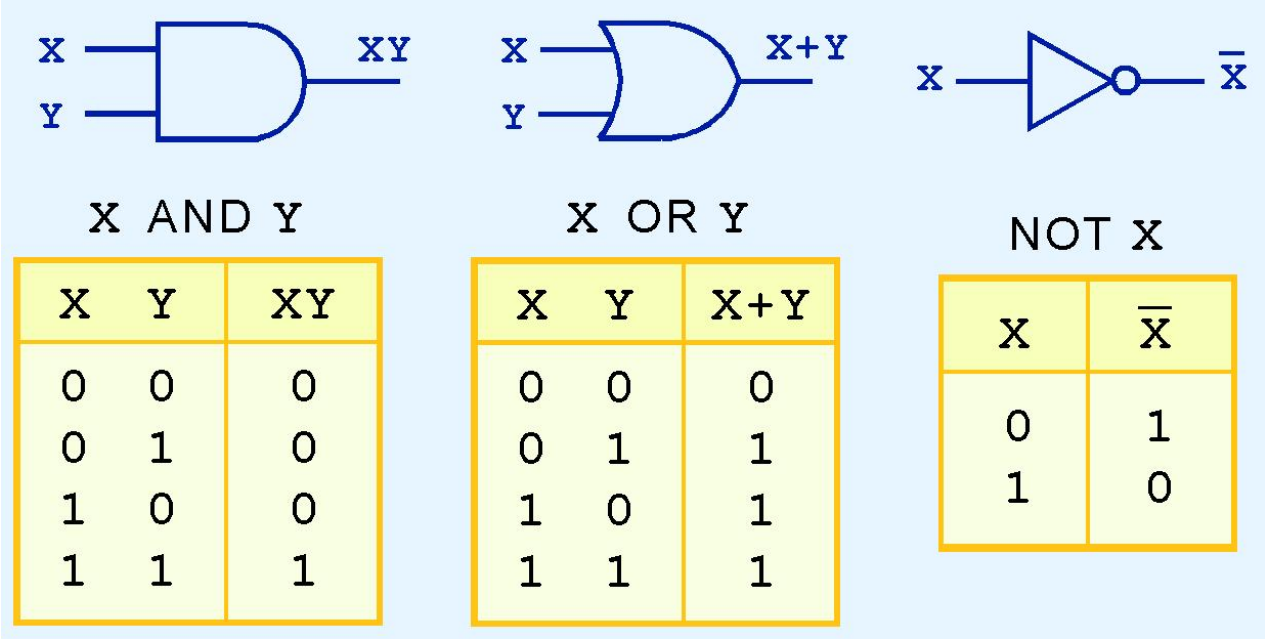
- Las computadores digitales contienen circuitos que implementan funciones booleanas.
- Cuando más simple la función más chico el circuito:
  - Son más baratos, consumen menos, y son mas rápidos!
- Podemos usar las identidades del algebra de Boole para reducir estas funciones.

Compuertas lógicas

- Una compuerta es un dispositivo electrónico que produce un resultado en base a un conjunto de valores de valores de entrada:
  - En realidad, están formadas por uno o varios transitores, pero lo podemos ver como una unidad.
  - Los circuitos integrados contienen colecciones de compuertas conectadas con algún propósito.

Compuertas lógicas

- Las más simples: AND, OR, y NOT:



X AND Y

X	Y	XY
0	0	0
0	1	0
1	0	0
1	1	1

X OR Y

X	Y	X+Y
0	0	0
0	1	1
1	0	1
1	1	1

NOT X

X	$\overline{X}$
0	1
1	0

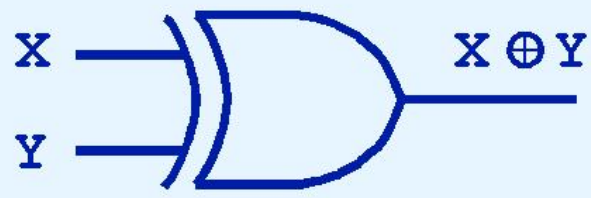
- Se corresponden exactamente con las funciones booleanas que vimos.

Compuertas lógicas

- Una compuerta muy útil: el OR exclusivo => XOR
- La salida es 1 cuando los valores de entrada difieren.

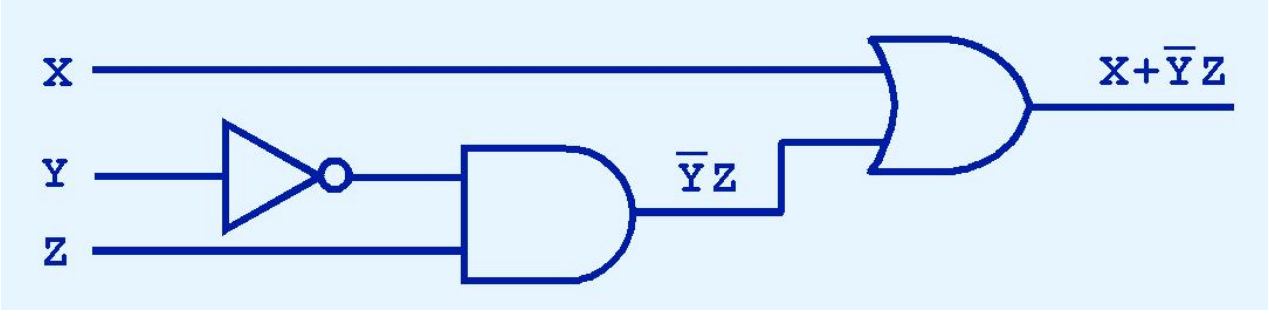
X XOR Y

X	Y	X⊕Y
0	0	0
0	1	1
1	0	1
1	1	0



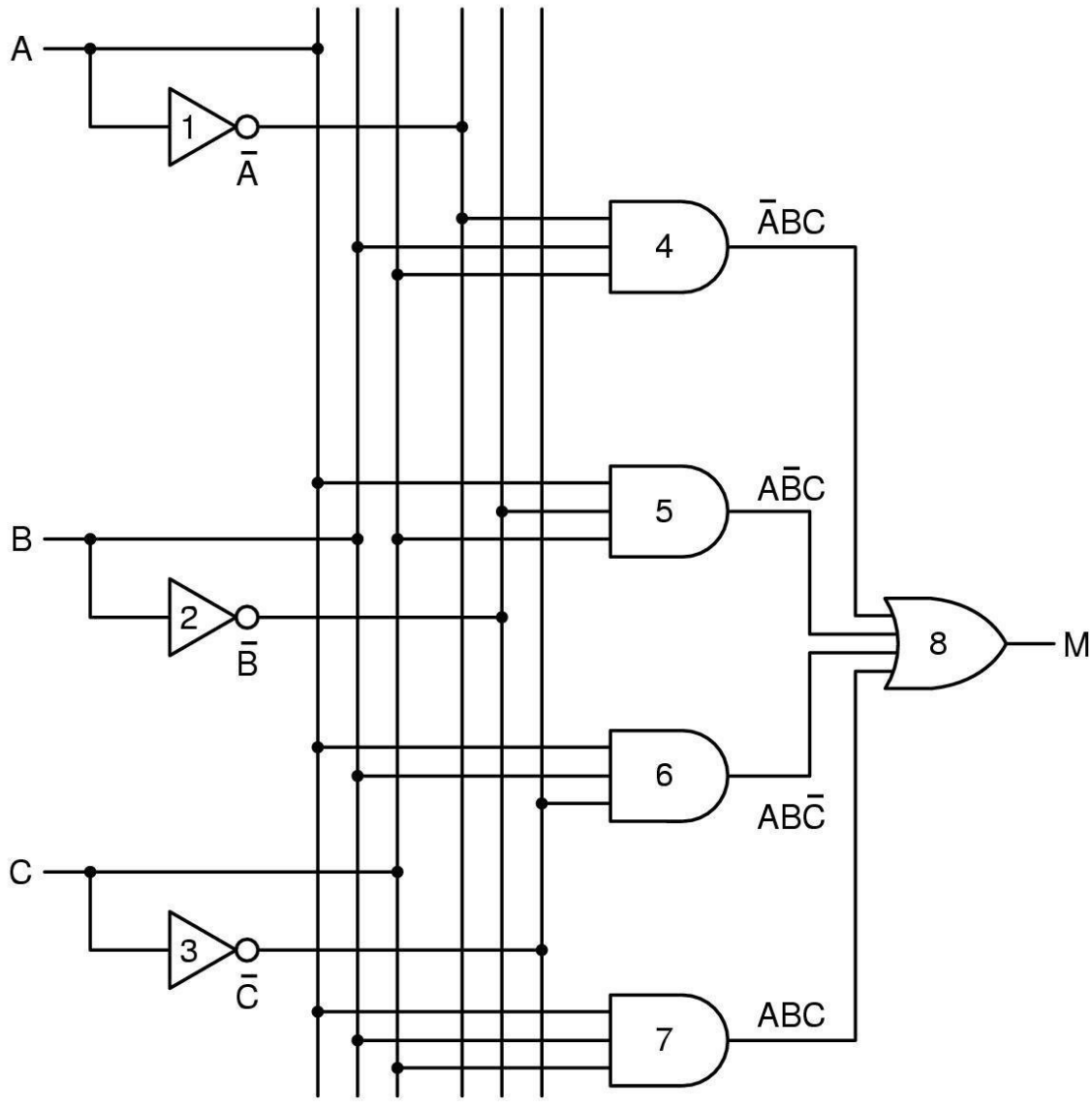
Implementación de funciones booleanas

- Combinando compuertas se pueden implementar funciones booleanas.
- Este circuito implementa la siguiente función: $F(x, y, z) = x + \overline{y}z$



Ejemplo: función mayoría

A	B	C	M
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

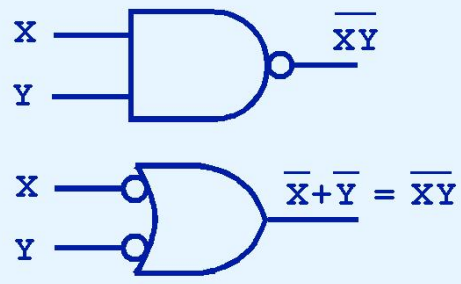


Compuertas lógicas combinadas

- NAND y NOR son dos compuertas lógicas combinadas.
- Con la identidad de Morgan se pueden implementar con AND u OR.
- Son más baratas y cualquier operación básica se puede representar usándolas cualquiera de ellas (sin usar la otra)?.

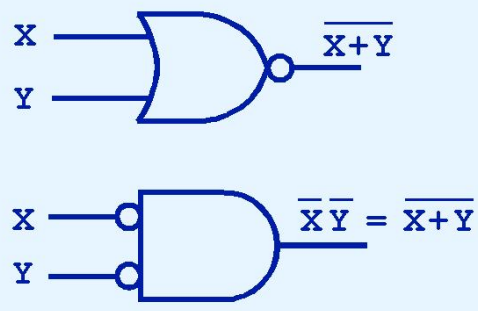
X NAND Y

X	Y	X NAND Y
0	0	1
0	1	1
1	0	1
1	1	0

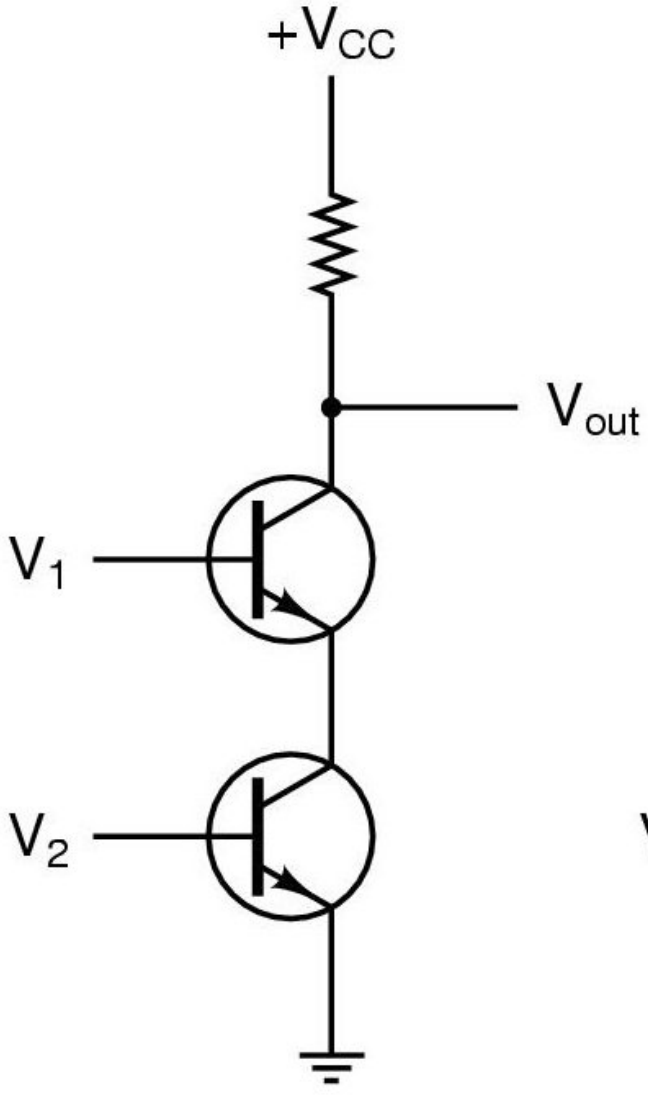


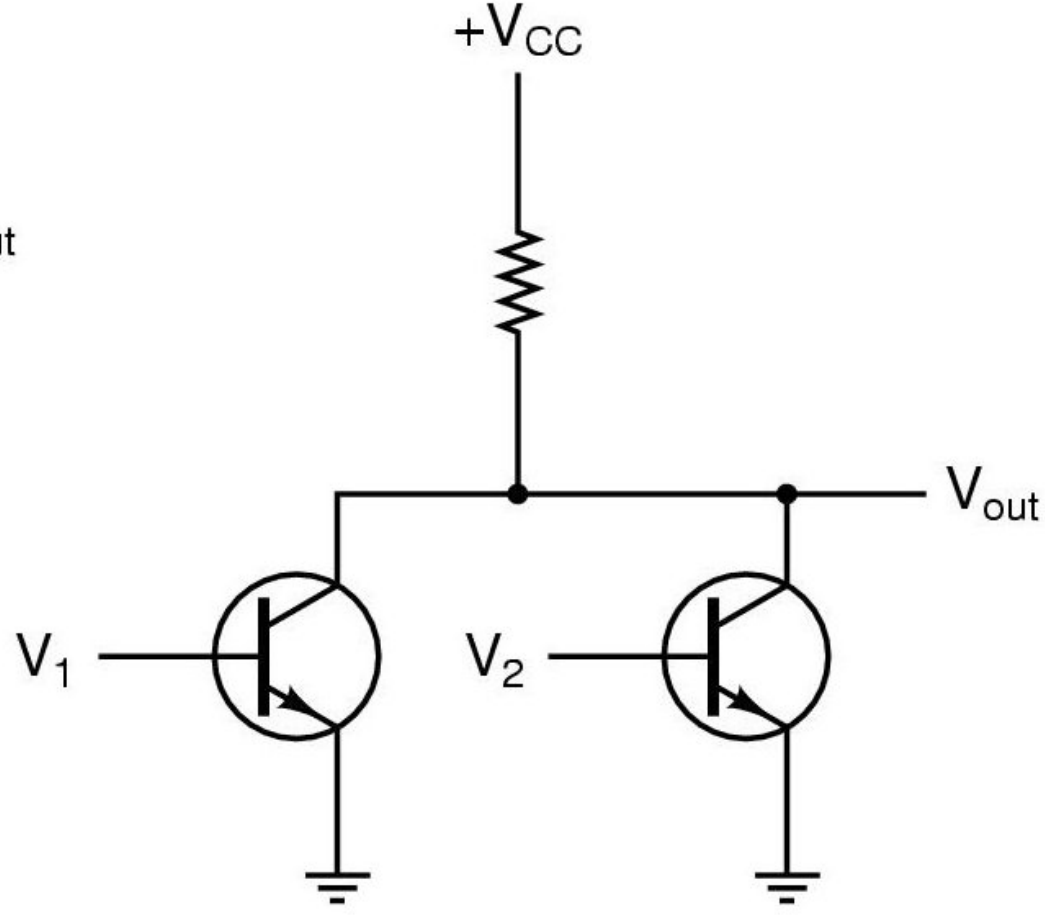
X NOR Y

X	Y	X NOR Y
0	0	1
0	1	0
1	0	0
1	1	0



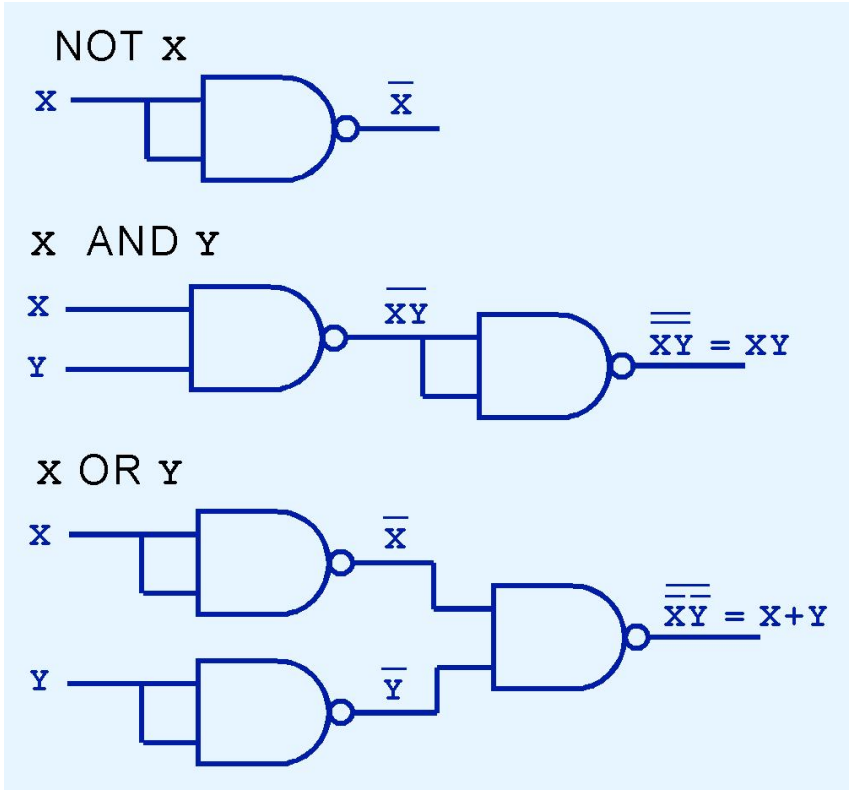
NAND y NOR





Ejemplos

- NOT usando NAND: simplemente unir las dos entradas.
- Utilizando solo NAND o NOR realizar circuitos con la misma funcionalidad que el AND y OR.



Circuitos combinatorios

- Producen una salida específica al (casi) instante que se le aplican valores de entrada.
- Implementan funciones booleanas.
- La aritmética y la lógica de la CPU se implementan con estos circuitos.

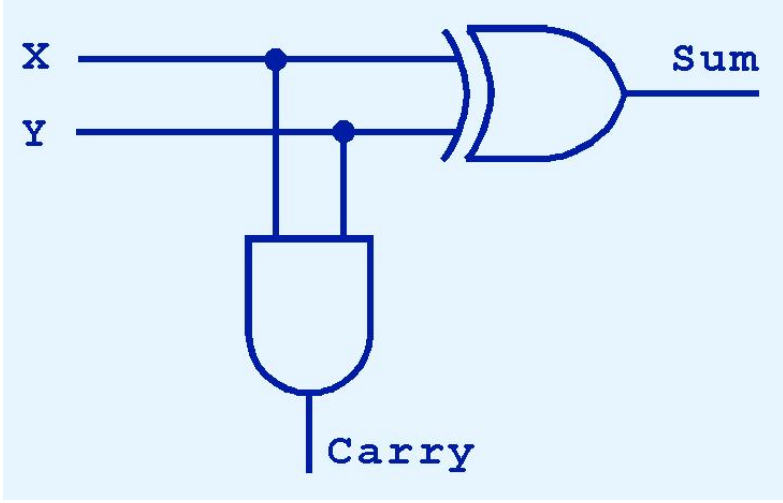
Sumador

- Como podemos construir un circuito que sume dos bits X e Y?
- $F(X, Y) = X + Y$ (suma aritmética)
- Que pasa si $X = 1$ e $Y = 1$?

X	Y	*Suma*	*carry*
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

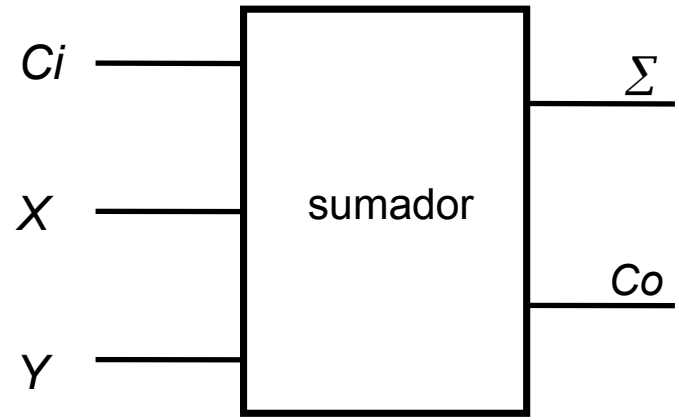
Semi-Sumador

- Podemos usar un XOR para la suma y un AND para el carry.
- A este circuito se lo llama semi-sumador.

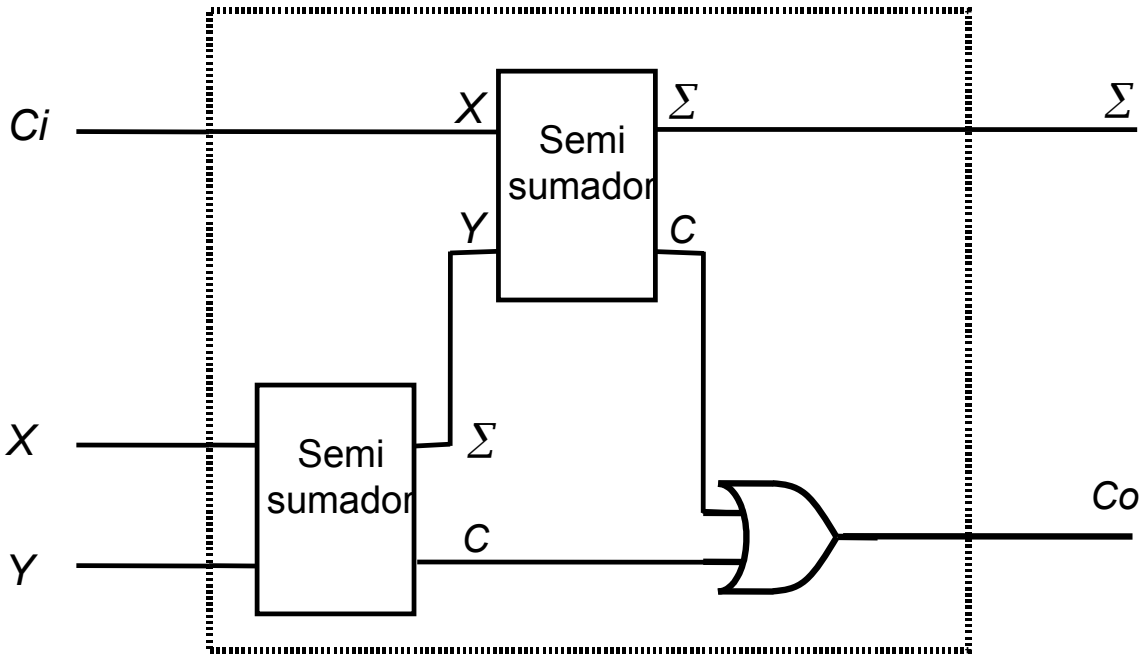


Sumador

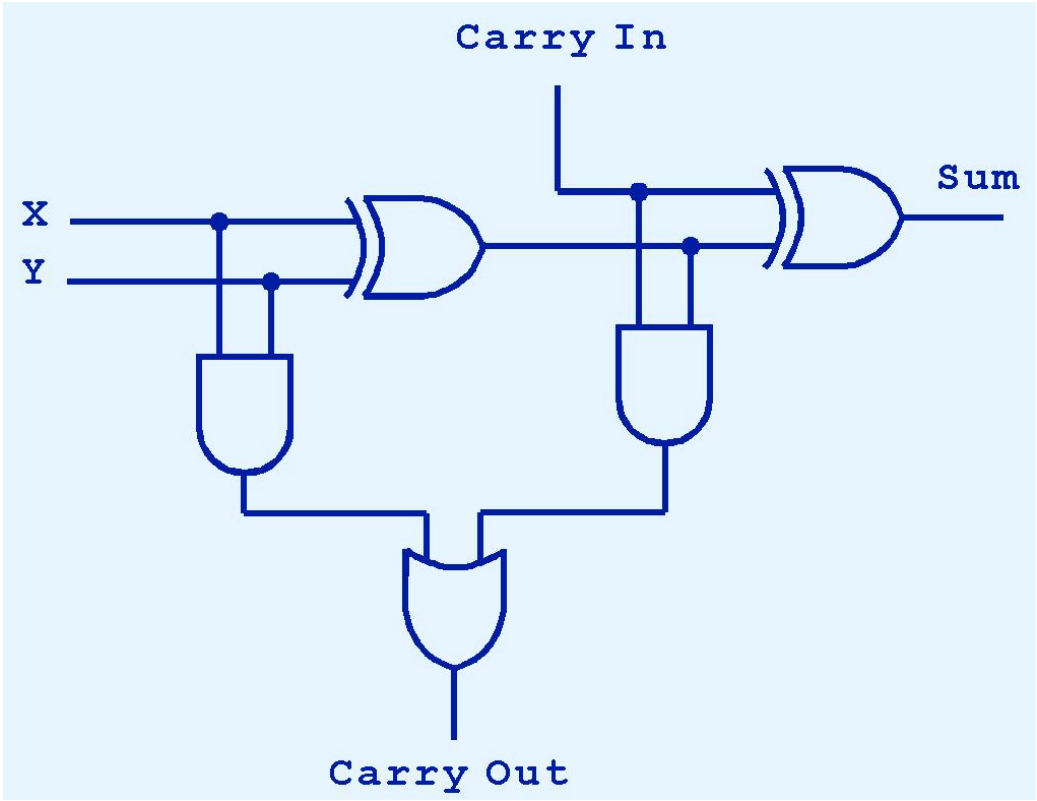
Co	1	
X		1
Y		1
	1	0



Sumador: estructura interna



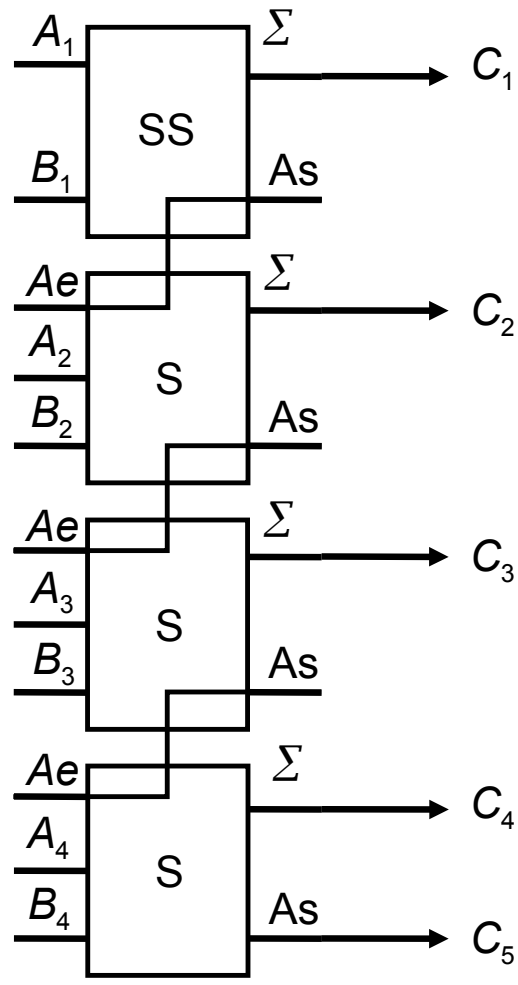
Sumador: estructura interna y tabla de verdad



X	*Y*	*Ci*	*Suma*	*Co*
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

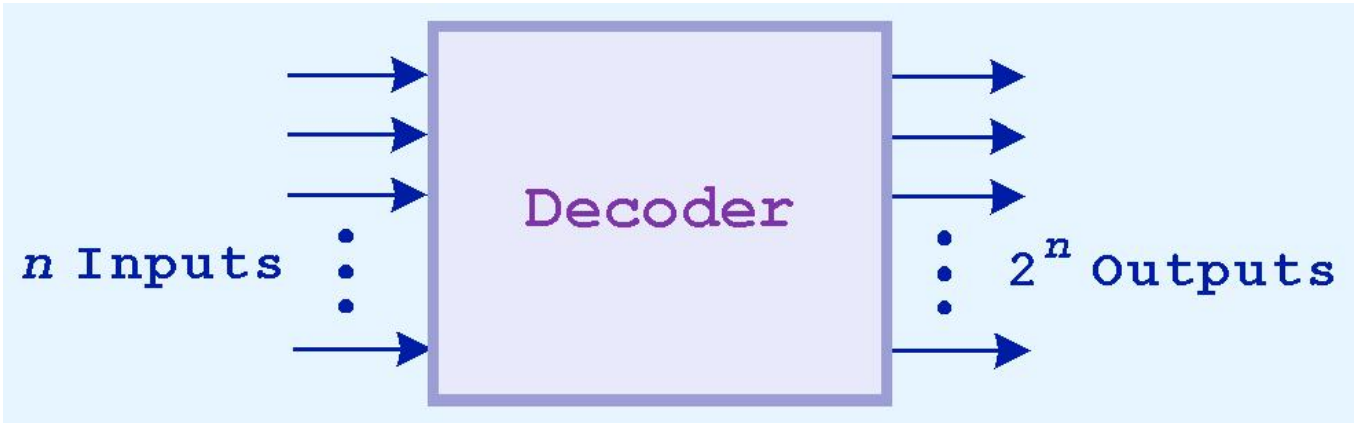
Sumador de 4 bits

	A_4	A_3	A_2	A_1
+	B_4	B_3	B_2	B_1
C_5	C_4	C_3	C_2	C_1



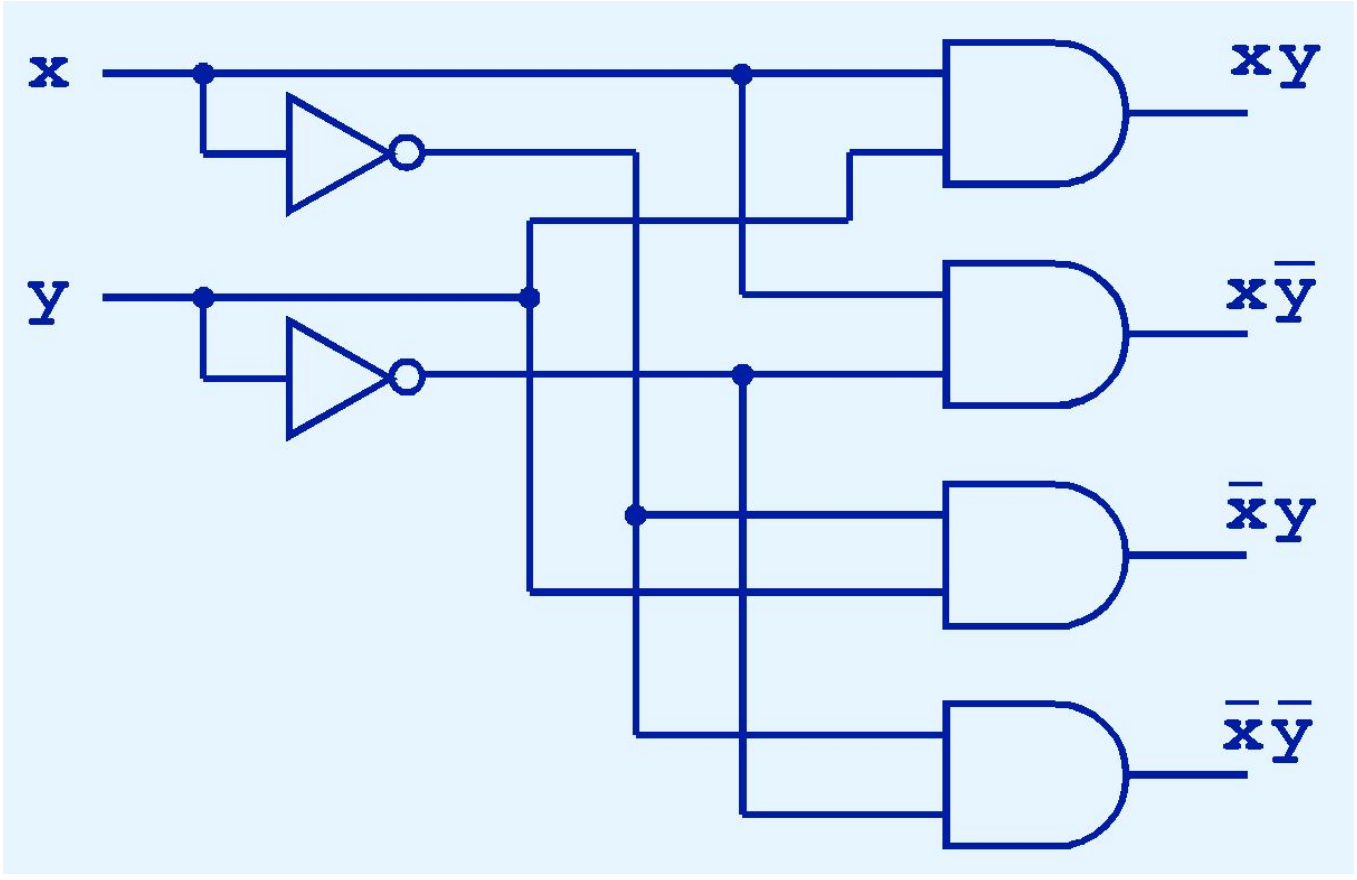
Decodificadores

- Los decodificadores de n entradas pueden seleccionar una de 2^n salidas.
- Son ampliamentes utilizados.
- Por ejemplo:
  - Seleccionar una locación en una memoria a partir de una dirección colocada en el bus memoria.



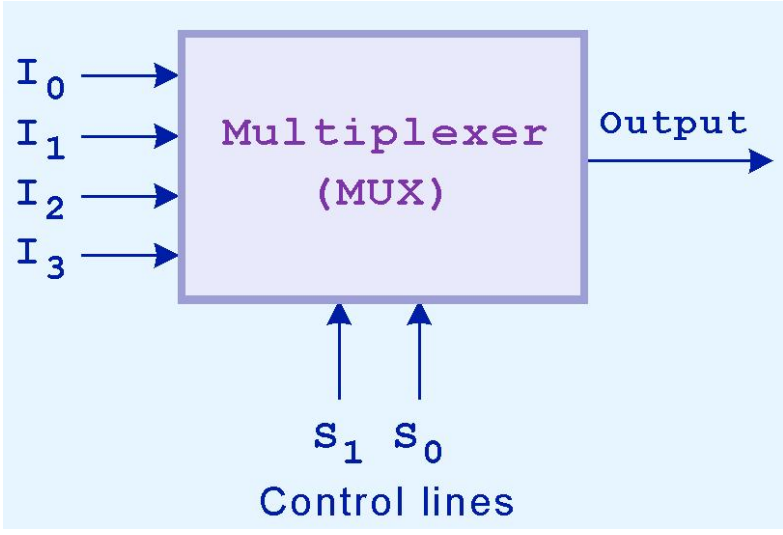
Decodificadores: ejemplo

- Decodificador 2-a-4:



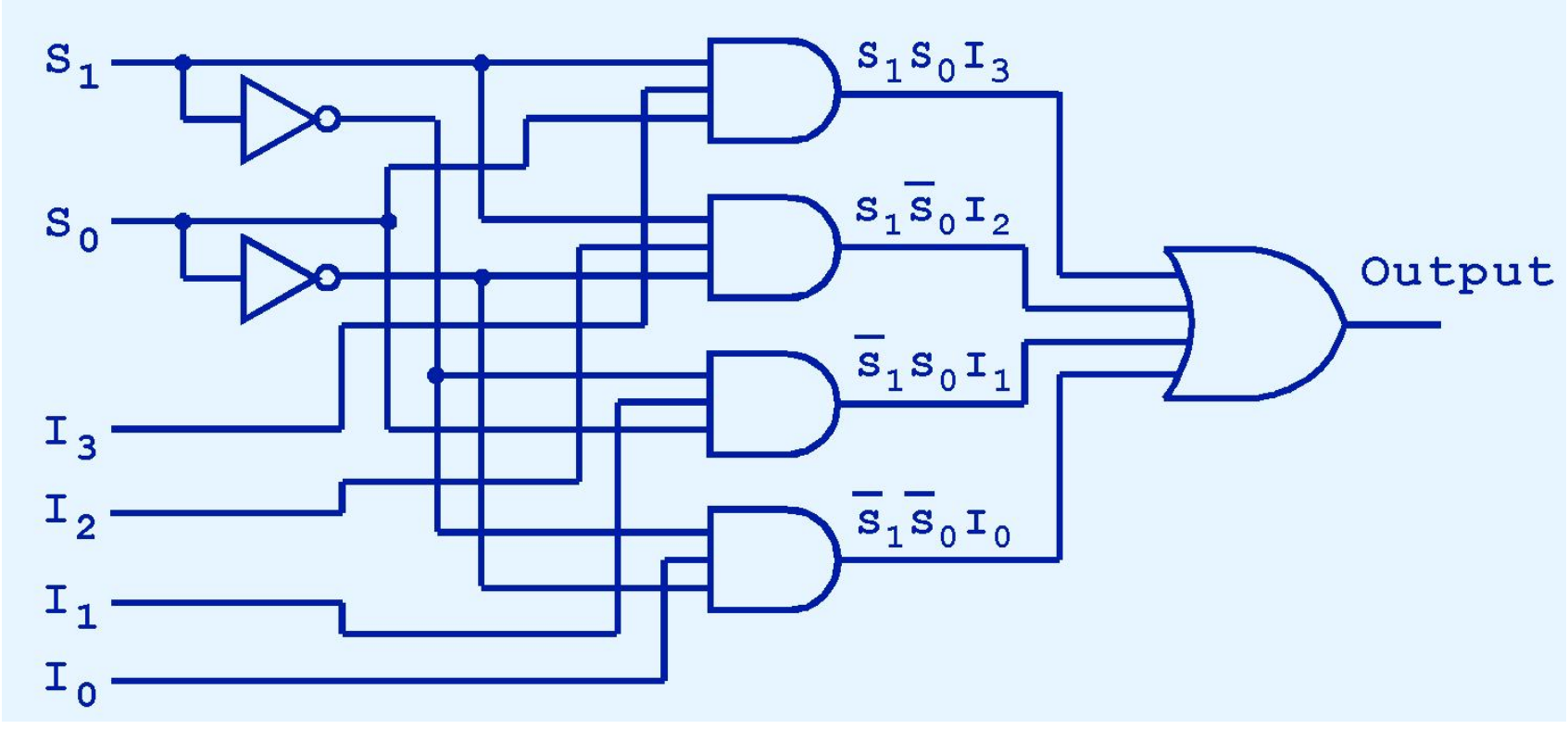
Multiplexores

- Selecciona una salida a de varias entradas.
- La entrada que es seleccionada como salida es determinada por las líneas de control.
- Para seleccionar entre n entradas, se necesitan $log_2 n$ líneas de control.
- Demultiplexor
  - Exactamente lo contrario al multiplexor.
  - Dada una entrada la direcciona entre n salidas, usando $log_2 n$ líneas de control.



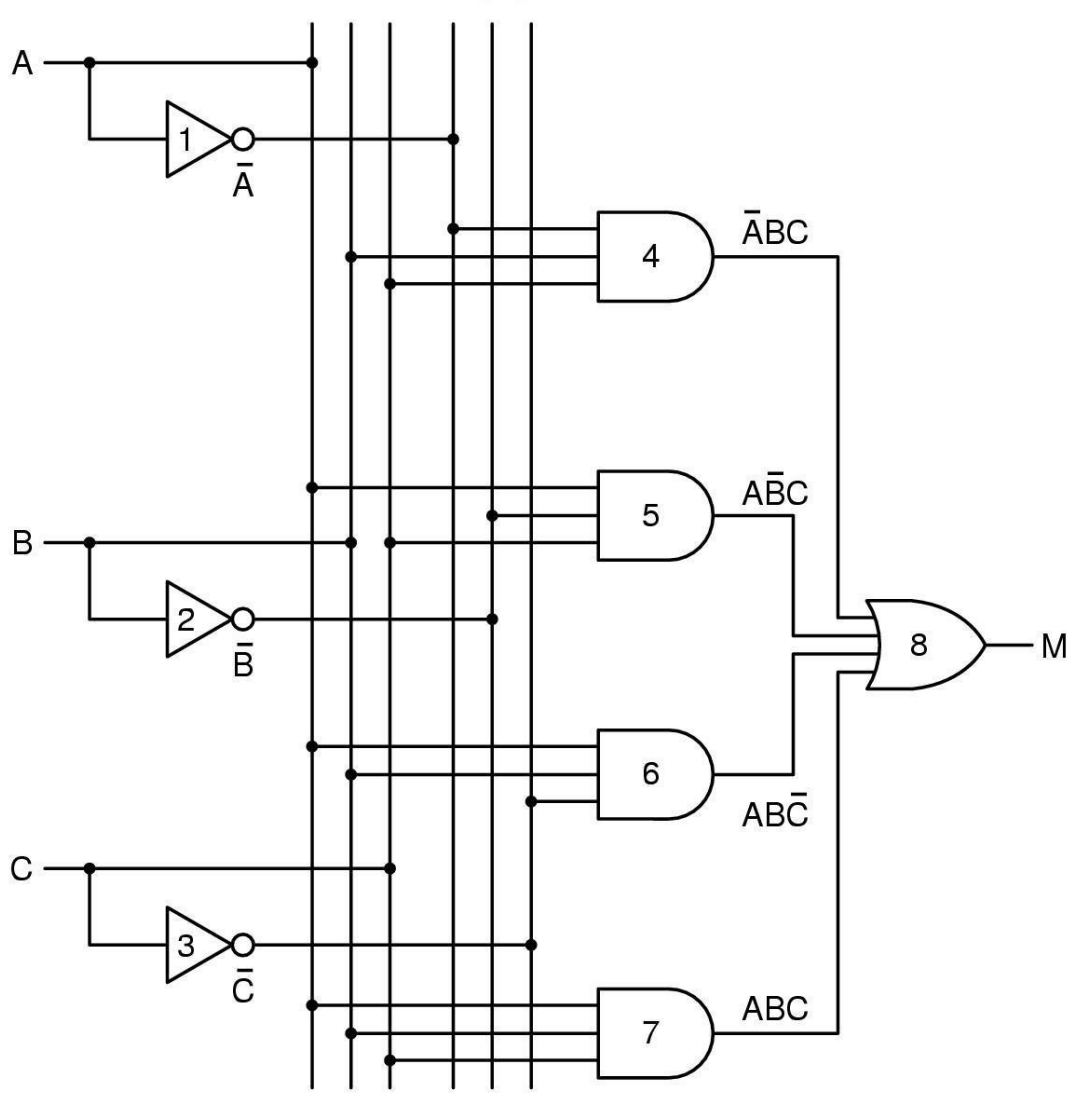
Multiplexor: ejemplo

- Multiplexor 4-a-1:

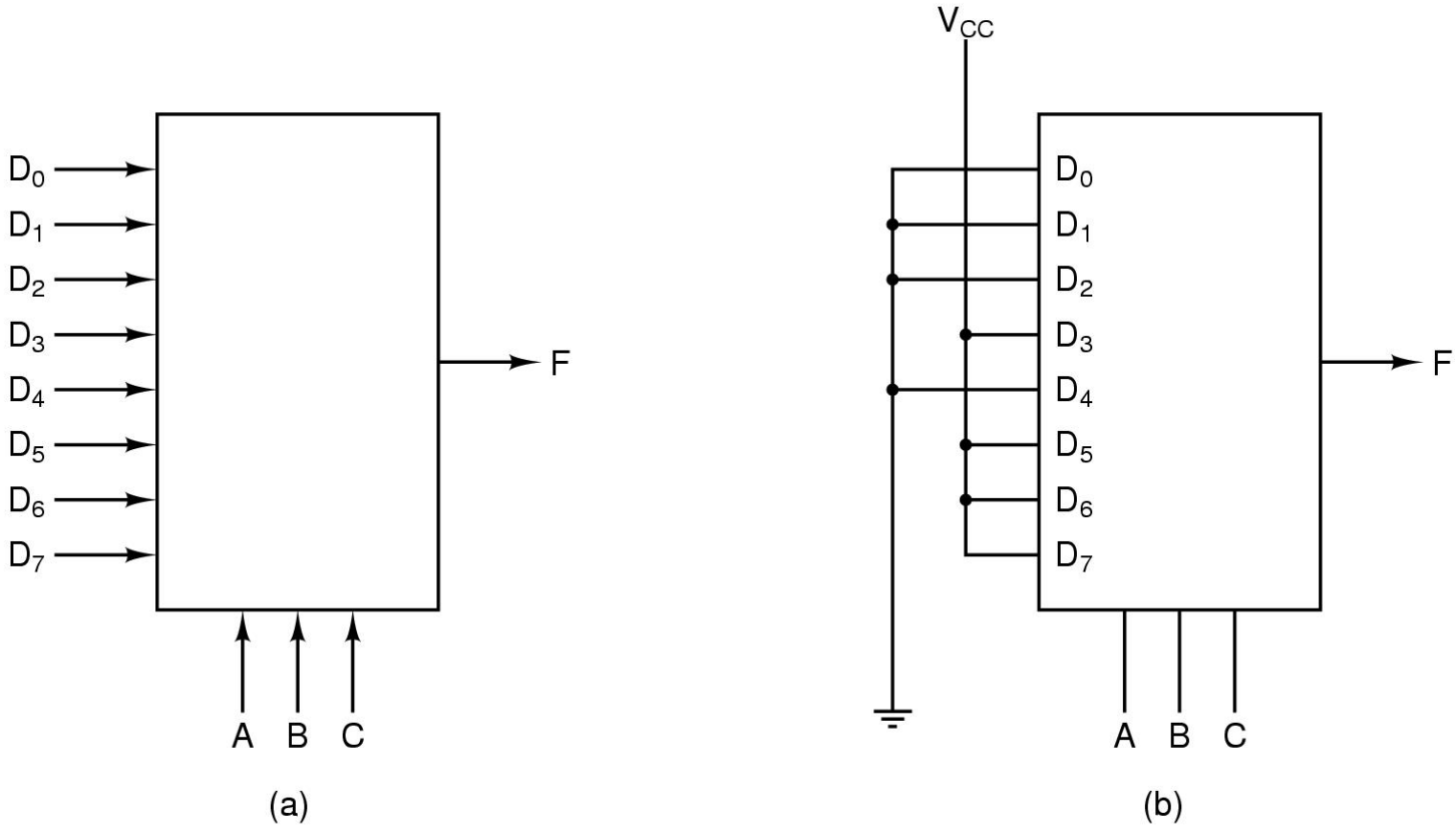


Función mayoría

A	B	C	M
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1



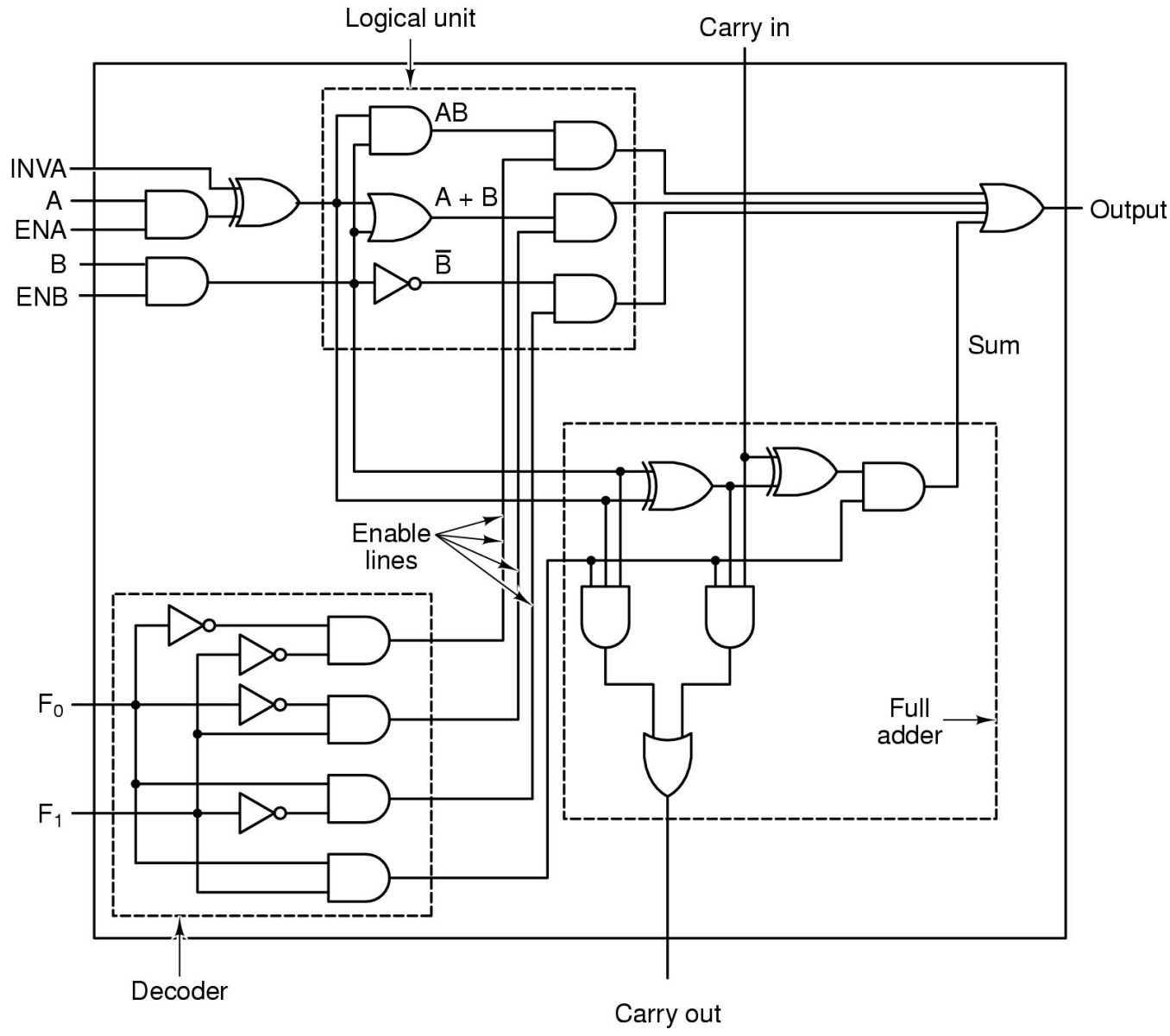
Función mayoría con multiplexor



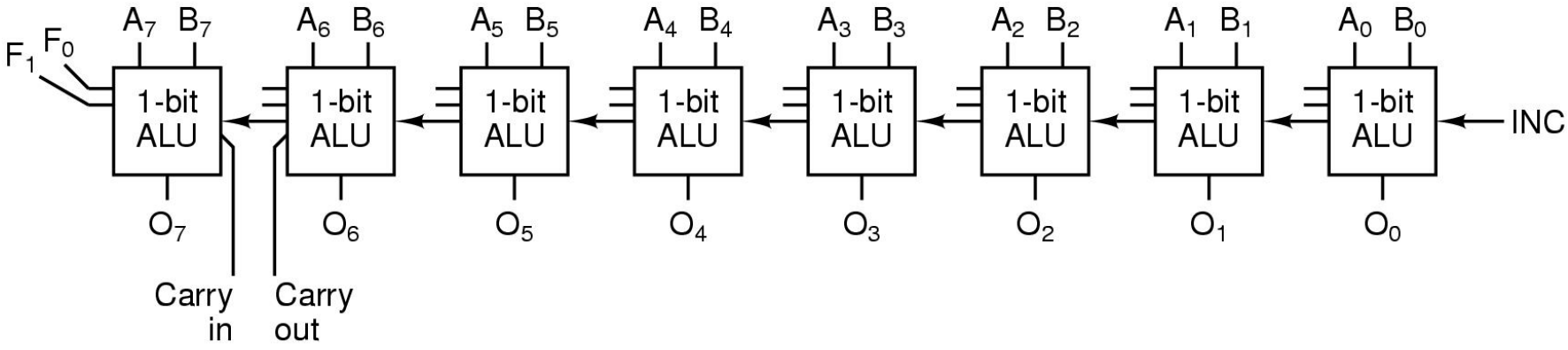
Ejemplo

- Construir una ALU de 1 bit.
- 3 entradas:
  - A, B, carry.
- 4 operaciones:
  - A.B, A+B, NOT B, Suma(A,B,Carry).
- Salidas:
  - Resultado, Carry out.

ALU de 1 bit


Logical unit
Carry in
AB
INVA
A
ENA
A + B
B
ENB
$\overline{B}$
Output
Sum
Enable lines
F_0
F_1
Full adder
Decoder
Carry out

ALU de 8 bits


A_7 B_7
A_6 B_6
A_5 B_5
A_4 B_4
A_3 B_3
A_2 B_2
A_1 B_1
A_0 B_0
F_1 F_0
1-bit ALU
INC
O_7
O_6
O_5
O_4
O_3
O_2
O_1
O_0
Carry in
Carry out

Memoria ROM

Entradas					Salidas							
I_4	I_3	I_2	I_1	I_0	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0
0	0	0	0	0	1	0	1	1	0	1	1	0
0	0	0	0	1	0	0	0	1	1	1	0	1
0	0	0	1	0	1	1	0	0	0	1	0	1
0	0	0	1	1	1	0	1	1	0	0	1	0
		.							.			
		.							.			
		.							.			
1	1	1	1	0	0	1	1	1	0	1	0	1
1	1	1	1	1	0	0	1	1	0	0	1	1

ROM con decoder