Clase 07 - Compuertas lógicas

IIC1001 - Algoritmos y Sistemas Computacionales

Cristian Ruz – cruz@uc.cl

Martes 31-Marzo-2026

Departamento de Ciencia de la Computación
Escuela de Ingeniería
Pontificia Universidad Católica de Chile

Contenidos

Contacto

Temas

Representación numérica

Contenidos

Contacto

Temas

Representación numérica

Contacto

🐐

vicente.cabra@uc.cl

Vicente Cabra

Ayudante jefe



🐸

alejandro.tapia@uc.cl

Alejandro Tapia

Ayudante jefe



🛼

gabriel.sandoval@uc.cl

Gabriel Sandoval

Ayudante jefe



Contenidos

Contacto

Temas

Representación numérica

Temas del curso

Sistemas computacionales

- Representación datos, números y codificación
- Funcionamiento hardware, procesadores y memoria.
- Funcionamiento de sistemas operativos: ejemplo scheduling
- Funcionamiento de Internet
- Herramientas computacionales: github + latex

Algoritmos

- Algoritmos y resolución de problemas
- Eficiencia algorítmica
- Estructuras secuenciales y ordenamiento
- Grafos y árboles

Próximos eventos

Jueves 9-Abril: Actividad 3. En grupos aleatorios

- Compuertas lógicas
- Circuitos digitales simples

Martes 14-Abril: Interrogación 1.

- Hasta circuitos sumadores.

Contenidos

Contacto

Temas

Representación numérica

¿Qué sabemos?

Representaciones numéricas, operatoria y codificación

- Bases numéricas: 2, 10, 16
- Bits y bytes
- Codificación de caracteres (mapas, *encodings*)
- Operaciones aritméticas básicas

Sabemos como el computador los representa.

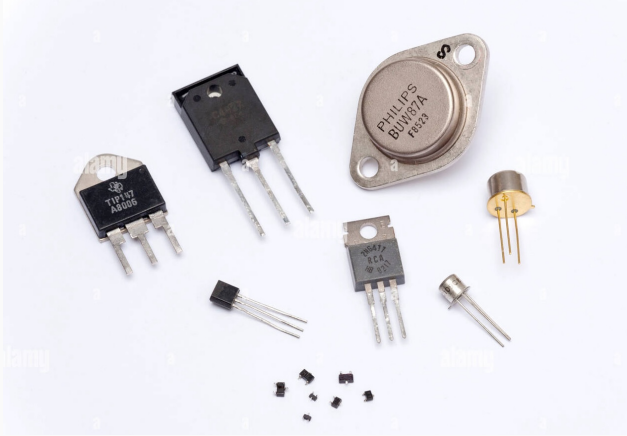
¿Cómo las implementa y opera con ellas?

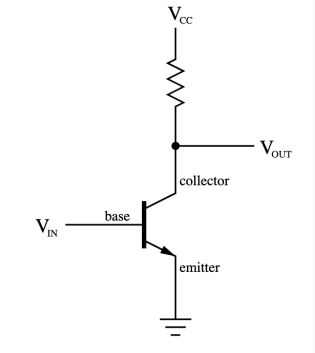
La magia física

Representación de 0 y 1

Transistores

- Dispositivo semiconductor
- Un voltaje aplicado a una entrada, permite controlar el voltaje de la salida
- Cierto nivel de voltaje puede interpretarse como “hay señal”, y la ausencia de ese voltaje indica que “no hay señal”.





La magia teórica

S.XIX, George Boole

- Desarrolla sistema formal para analizar proposiciones lógicas
- Proposiciones lógicas se basan en “valores de verdad”: Verdadero (V) ó Falso (F)
- Lógica booleana: operaciones `not`, `and`, `or`
- Álgebra booleana: operaciones algebraicas entre valores lógicos

1937, Claude Shannon

- Álgebra booleana puede usarse para representar valores numéricos (binarios)
- Valor Verdadero (`True`): 1
- Valor Falso (`False`): 0

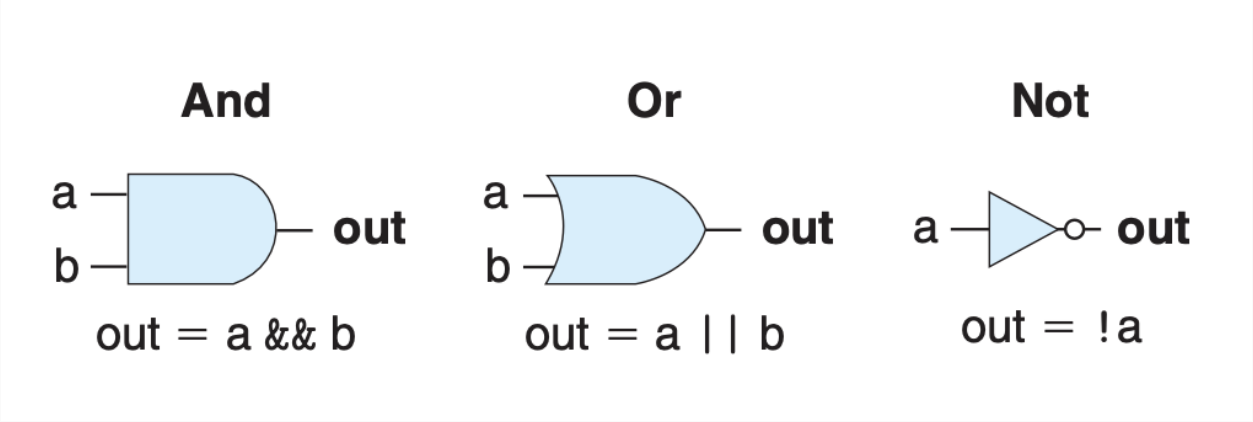
Implementación se basa en compuertas lógicas:

`not`, `and`, `or`, `xor` ... (hay más)

Compuertas lógicas

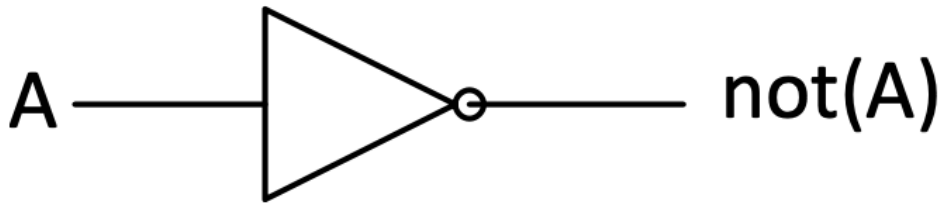
Componentes electrónicas que implementa las condiciones lógicas de Boole.

Poseen una cantidad de valores (bit) de entrada y entregan como resultado uno o más valores (bits) de salida de acuerdo a una tabla.



Compuerta NOT

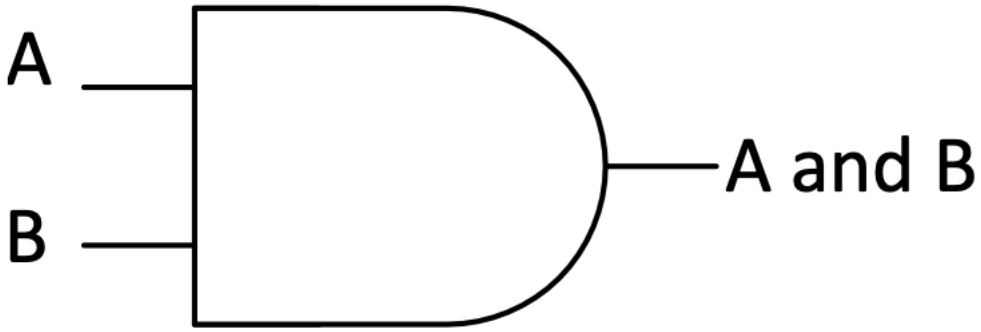
Entrada es lo opuesto de la salida: $\text{not}(A) = \neg A$



A	not(A)
0	1
1	0

Compuerta AND

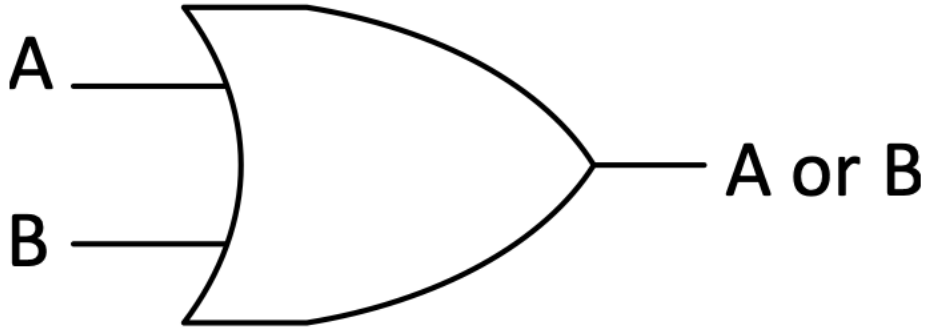
Entrada es verdadera (1) sólo cuando **ambas** entradas son verdaderas: A **and** $B = A \wedge B$



A	B	A and B
0	0	0
0	1	0
1	0	0
1	1	1

Compuerta AND

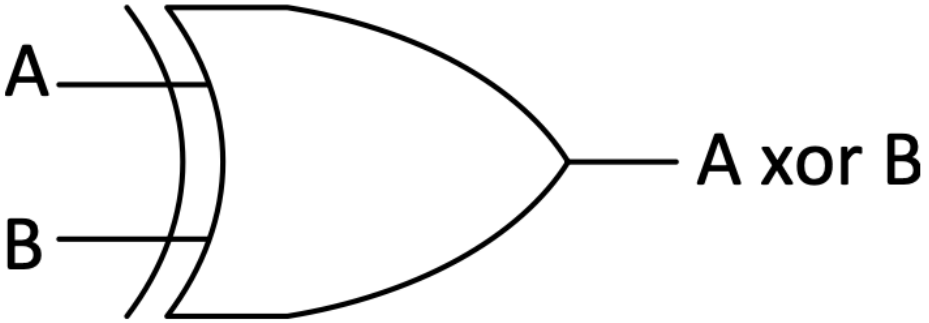
Entrada es verdadera (1) cuando **al menos una** de las entradas son verdaderas: $A \texttt{ or } B = A \vee B$



A	B	A or B
0	0	0
0	1	1
1	0	1
1	1	1

Compuerta XOR

Entrada es verdadera (1) cuando **exactamente una** de las entradas son verdaderas: $A \texttt{ xor } B = A \bigoplus B$

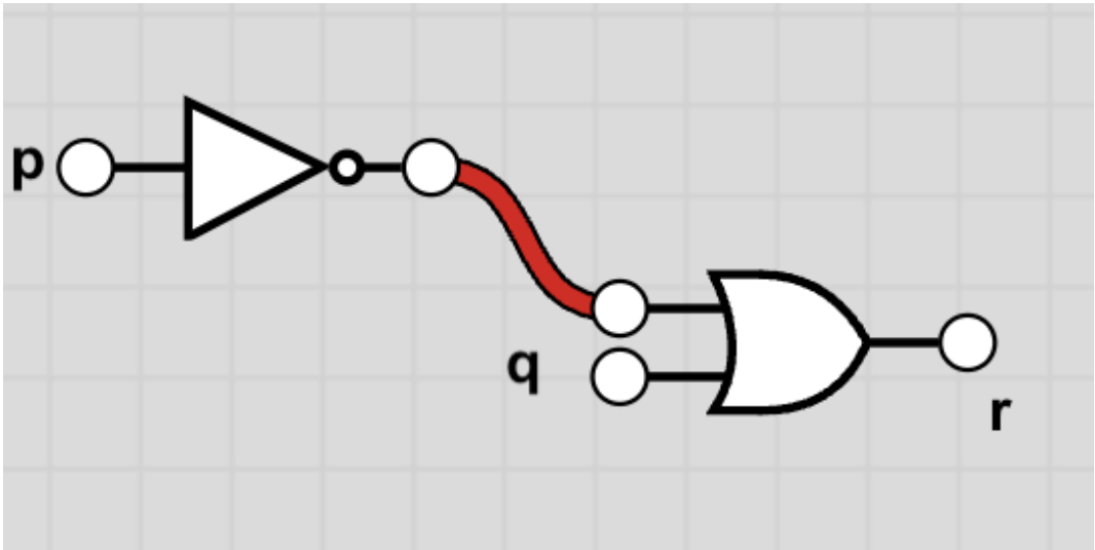


A	B	A xor B
0	0	0
0	1	1
1	0	1
1	1	0

Circuitos binarios

Mezclando entradas y salidas de diferentes compuertas podemos construir circuitos que implementan operaciones lógicas.

¿Qué salidas se obtienen del siguiente circuito? (I2, 2023-1)



Circuitos binarios

Construir un circuito cuya salida sea 1 solamente si ambos bit de entrada son iguales.

a	b	a **eq** b
0	0	1
0	1	0
1	0	0
1	1	1

¿Cómo se escribiría en álgebra booleana?

¿Cómo se escribiría un circuito lógico?

Circuitos sumadores

Podemos empezar a construir circuitos que implementen operaciones aritméticas.

¿Cómo sería relación entrada/salida? $S(A, B) = A + B$

A	B	$A + B$
0	0	00
0	1	01
1	0	01
1	1	10

Circuitos sumadores

Podemos empezar a construir circuitos que implementen operaciones aritméticas.

¿Cómo sería relación entrada/salida? $S(A, B) = A + B$

A	B	$A + B$
0	0	00
0	1	01
1	0	01
1	1	10

¡Pero la salida de una suma de dos números de un 1 bit son dos bit!

Circuitos sumadores

INPUT: bit A y bit B

OUTPUT: dos bit. Podemos separarlos: S_1, S_0

A	B	$A + B = S_1S_0$	S_1	S_0
0	0	00	0	0
0	1	01	0	1
1	0	01	0	1
1	1	10	1	0

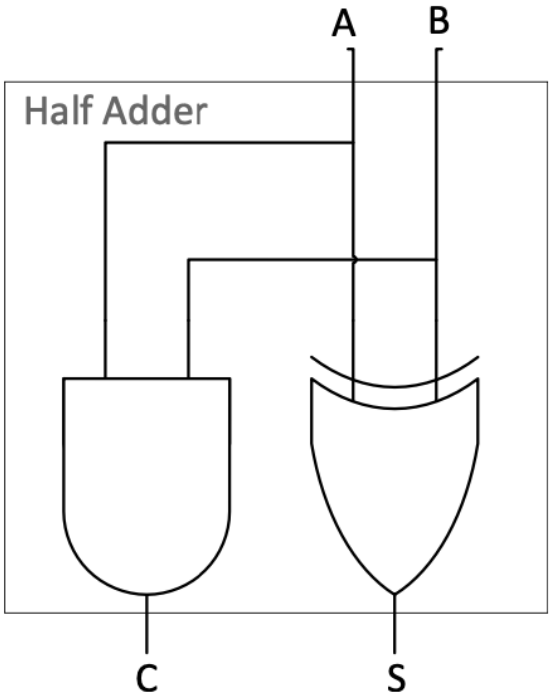
Circuitos sumadores

INPUT: bit A y bit B

OUTPUT: dos bit. Podemos separarlos: S_1, S_0

A	B	$A + B = S_1S_0$	S_1	S_0
0	0	00	0	0
0	1	01	0	1
1	0	01	0	1
1	1	10	1	0

Half-adder



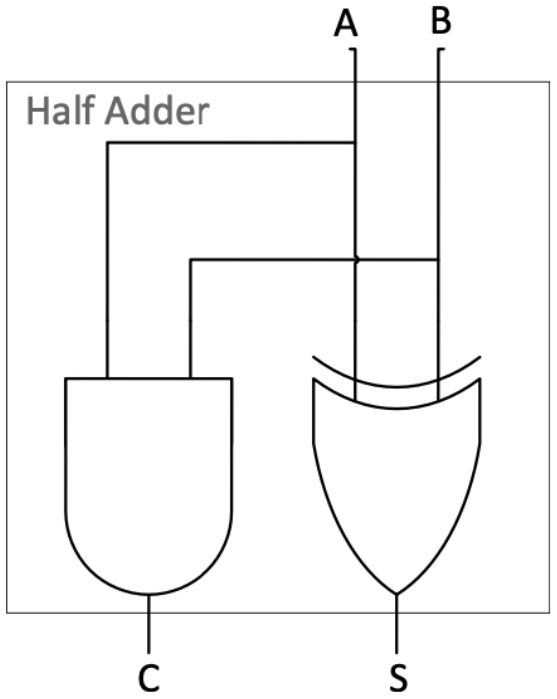
Circuitos sumadores

INPUT: bit A y bit B

OUTPUT: dos bit. Podemos separarlos: S_1, S_0

A	B	$A + B = S_1S_0$	S_1	S_0
0	0	00	0	0
0	1	01	0	1
1	0	01	0	1
1	1	10	1	0

Half-adder



Con esto podemos dos números de 1 bit.

¿Cómo sumar números más grandes?

Circuitos sumadores

¿Cómo sumar números, por ejemplo, de 2-bit?

INPUT: dos números de dos bit: $A = A_1A_0$, $B = B_1B_0$

OUTPUT: $A + B = S$ ¿cuántos bit?

Hay que ver cómo funciona la suma

$$\begin{array}{cccccc} & & {\scriptstyle 1} & {\scriptstyle 1} & & \\ & 1 & 0 & 0 & 1 & \\ + & 0 & 0 & 1 & 1 & \\ \hline & 1 & 1 & 0 & 0 & \end{array}$$

Circuitos sumadores

¿Cómo sumar números, por ejemplo, de 2-bit?

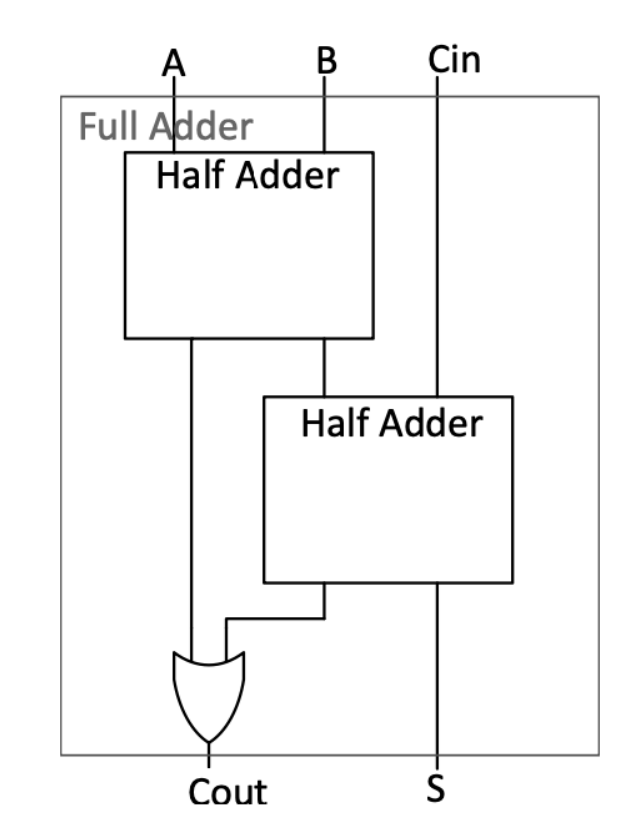
INPUT: dos números de dos bit: $A = A_1A_0$, $B = B_1B_0$

OUTPUT: $A + B = S$ ¿cuántos bit? $S = S_2S_1S_0$

Hay que ver cómo funciona la suma

$$\begin{array}{cccccc} & & {\scriptstyle 1} & {\scriptstyle 1} & & \\ & 1 & 0 & 0 & 1 & \\ + & 0 & 0 & 1 & 1 & \\ \hline & 1 & 1 & 0 & 0 & \end{array}$$

Full-adder



Circuitos sumadores

¿Cómo sumar números, por ejemplo, de 2-bit?

INPUT: dos números de dos bit: $A = A_1A_0$, $B = B_1B_0$

OUTPUT: $A + B = S$ ¿cuántos bit?

A_1A_0	B_1B_0	$S = S_2S_1S_0$
00	00	000
00	01	001
00	10	010
00	11	011
01	00	001
01	01	010
01	10	011
01	11	100
10	00	010
10	01	011
10	10	100
10	11	101
11	00	011
11	01	100
11	10	101
11	11	110

Circuitos sumadores

¿Cómo sumar números, por ejemplo, de 2-bit?

INPUT: dos números de dos bit: $A = A_1A_0$, $B = B_1B_0$

OUTPUT: $A + B = S$ ¿cuántos bit? $S = S_2S_1S_0$

A_1A_0	B_1B_0	$S = S_2S_1S_0$
00	00	000
00	01	001
00	10	010
00	11	011
01	00	001
01	01	010
01	10	011
01	11	100
10	00	010
10	01	011
10	10	100
10	11	101
11	00	011
11	01	100
11	10	101
11	11	110

Full-adder

