UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

Álgebra de Boole

Compuertas Lógicas

Los dispositivos físicos que implementan una función booleana simple, es decir un operador booleano, se denominan compuertas lógicas o simplemente **compuertas.**

En lenguaje técnico también se las denomina *«gates»*

UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

Compuertas lógicas

Proposiciones		Conjuntos	Boole
∧	Conjunción	∩	Producto lógico (.) AND
∨	Disyunción	∪	Suma lógica (+) OR

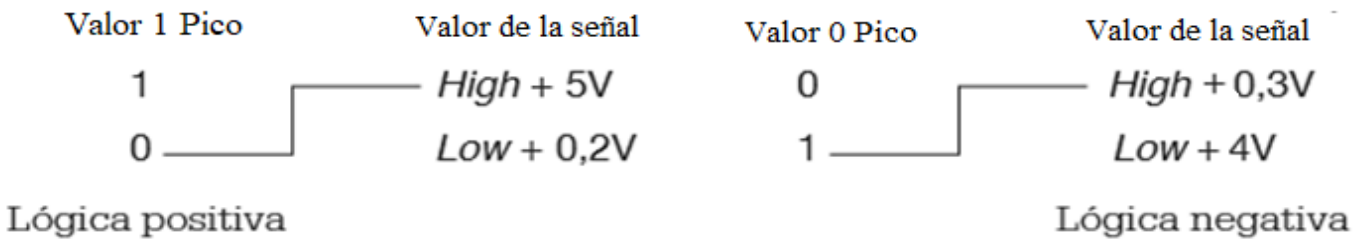
Lógica positiva

En el Algebra de Boole las variables son binarias y sólo pueden tomar dos valores que son complementarios entre si.

Estos valores se designan como:

1 SI / ALTO / **V**ERDADERO / ON

0 NO / BAJO/ **F**ALSO / OFF

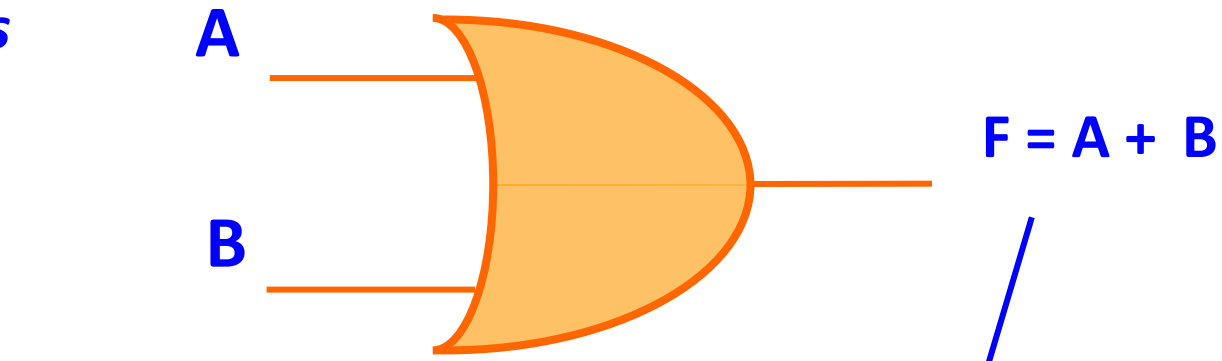


Compuertas Lógicas: OR

Realiza la Suma Lógica

La función lógica OR es Falsa sólo cuando todas las variables de entrada están en "0"

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



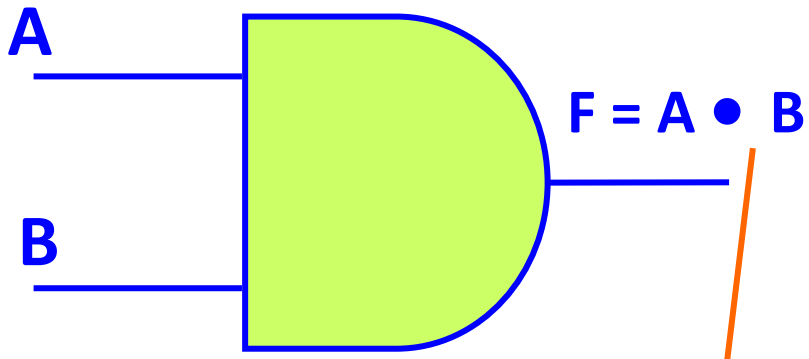
Se lee A "o" B (OR)

Compuertas Lógicas: AND

Realiza el Producto Lógico

La función lógica AND es verdadera sólo cuando todas las variables de entrada están en "1"

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



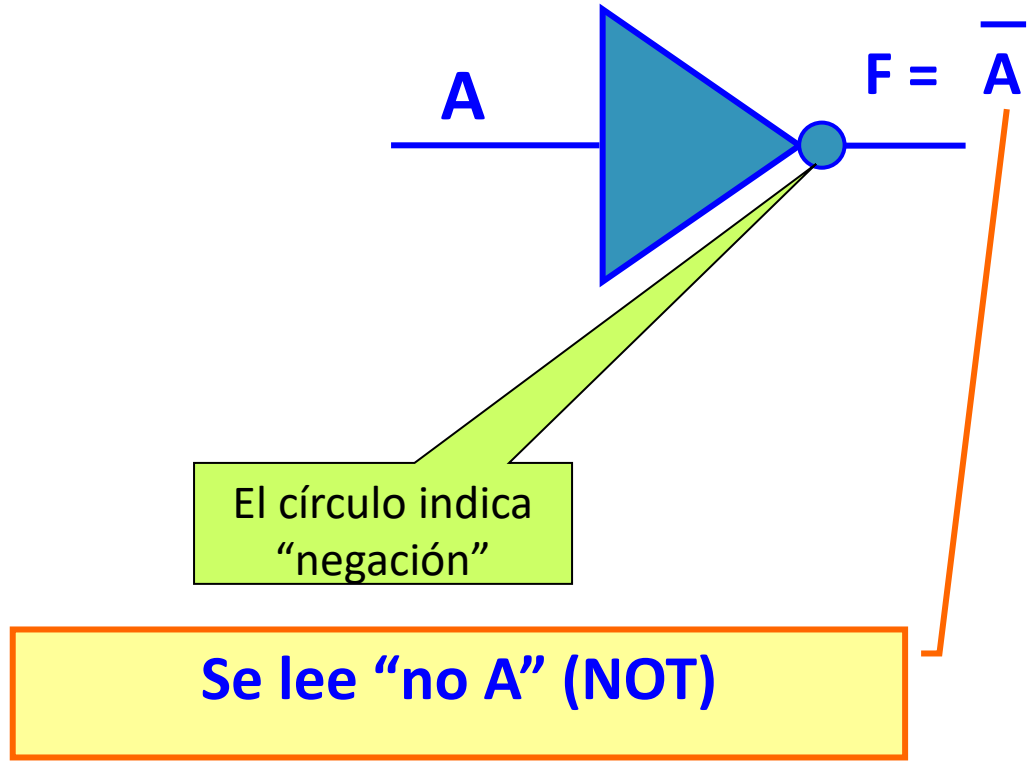
Se lee A "y" B (AND)

Compuertas Lógicas: NOT

Realiza la Complementación

Este operador "invierte" el valor lógico de la entrada.

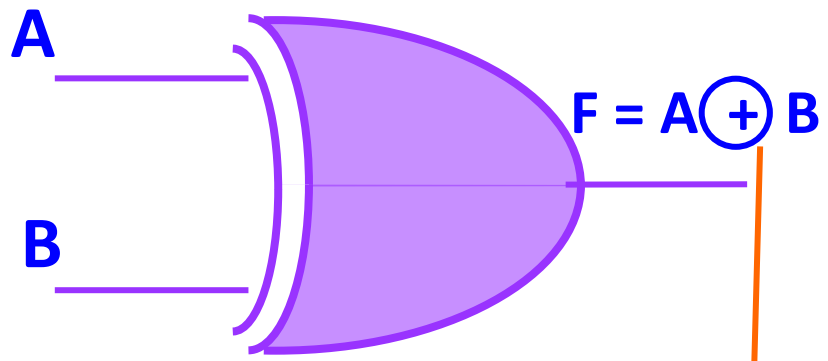
A	F
0	1
1	0



Compuertas Lógicas: XOR

Esta función lógica especial, OR-Exclusiva es verdadera sólo cuando es IMPAR la cantidad de variables de entrada que están en "1"

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

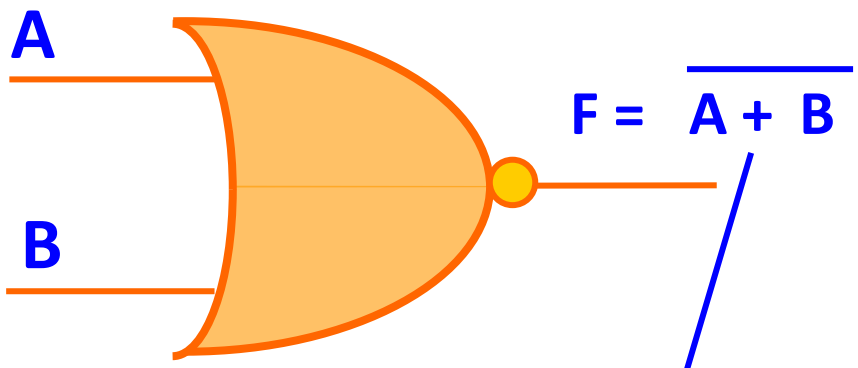


Se lee A "exclusive-or" B (OR-EX)

Compuertas Negadas: NOR

La función lógica NOR es Verdadera sólo cuando todas las variables de entrada están en "0"

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

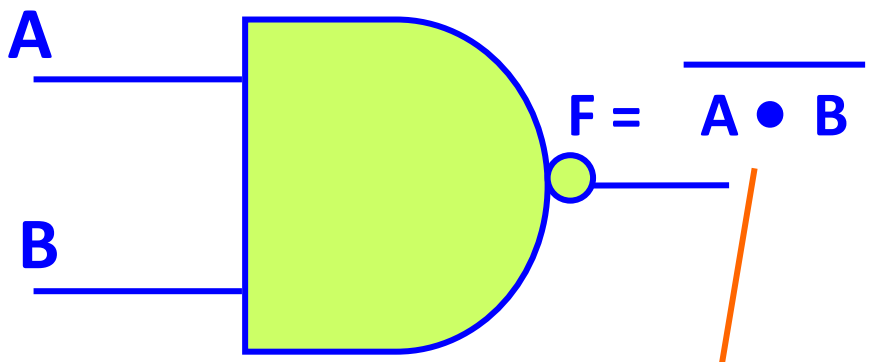


Se lee A o B "negado" (NOR)

Compuertas Negadas: NAND

La función lógica NAND es Falsa sólo cuando todas las variables de entrada están en "1"

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

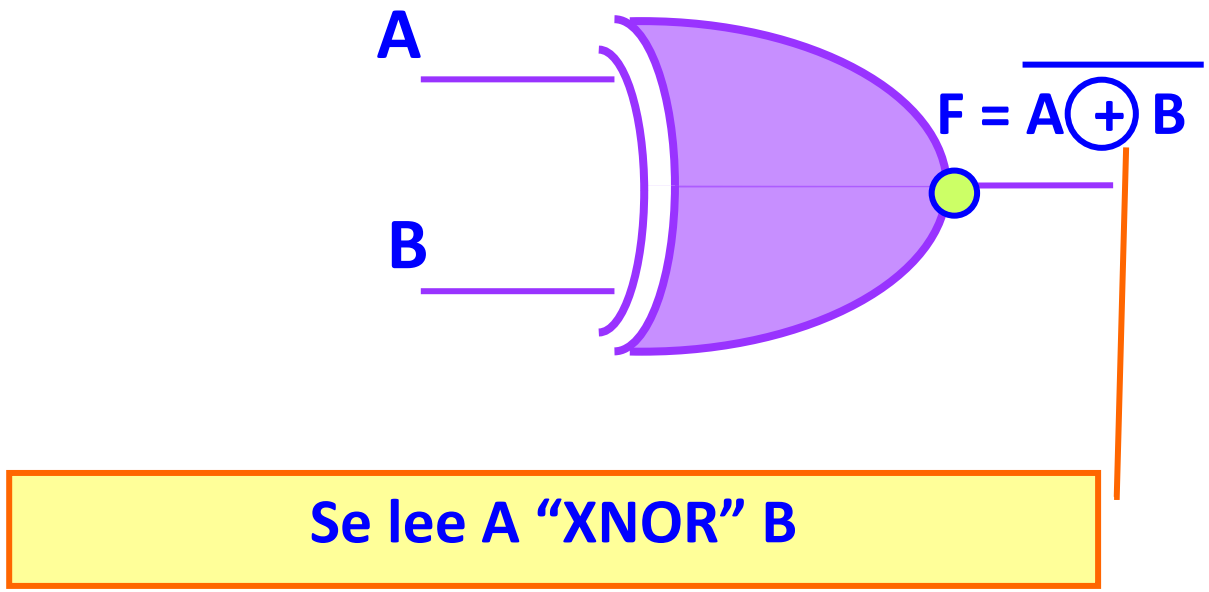


Se lee A y B "negado"(NAND)

Compuertas Lógicas: XNOR

Esta función lógica especial, XNOR-Exclusiva es verdadera, es decir "1", cuando las variables de entrada son iguales.

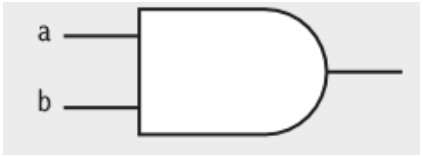
A	B	F
0	0	1
0	1	0
1	0	0
1	1	1



Resumen de compuertas

AND

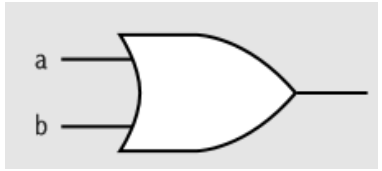
A	B	A.B
0	0	0
0	1	0
1	0	0
1	1	1



$f(a, b) = a \cdot b$

OR

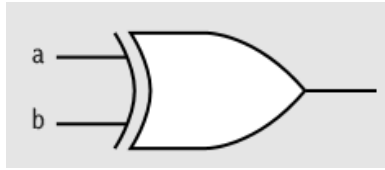
A	B	A+B
0	0	0
0	1	1
1	0	1
1	1	1



$f(a, b) = a + b$

XOR

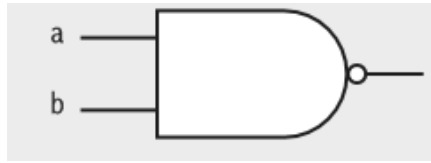
A	B	A+B
0	0	0
0	1	1
1	0	1
1	1	0



$f(a, b) = a \oplus b$

NAND

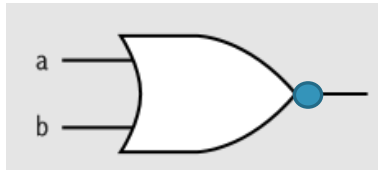
A	B	$\overline{A.B}$
0	0	1
0	1	1
1	0	1
1	1	0



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

NOR

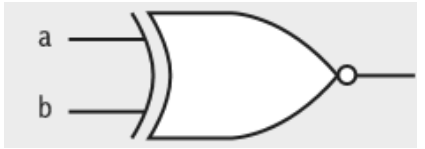
A	B	$\overline{A+B}$
0	0	1
0	1	0
1	0	0
1	1	0



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

XNOR

A	B	$\overline{A+B}$
0	0	1
0	1	0
1	0	0
1	1	1



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

UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

Funciones booleanas

Un conjunto de variables booleanas vinculadas entre sí mediante los operadores de suma lógica, producto lógico y complementación constituye una ***función booleana***.

- La Tabla de Verdad es una de las formas de expresar una función booleana.
- También se usan expresiones literales (***polinómica y factorial***) y expresiones simbólicas.
- La ***forma canónica*** de un función booleana es una expresión cuyos términos contienen la totalidad de las variables del problema.

Tablas de verdad

Una tabla de verdad es: “Una lista ordenada de las 2n combinaciones distintas de ceros y unos, que se pueden obtener de la combinación del valor de n variables binarias”. Para 3 variables, y considerando que los valores que puede tomar cada una son sólo 0 o 1, la cantidad de combinaciones binarias distintas es de $2^3 = 8$, para 4 variables la cantidad de combinaciones es, entonces, 24 = 16. En términos generales, con n variables se pueden obtener 2n combinaciones diferentes.

A	B	C	Z
0	0	0	**0**
0	0	1	**1**
0	1	0	**1**
0	1	1	**0**
1	0	0	**1**
1	0	1	**0**
1	1	0	**0**
1	1	1	**1**

Formas normales o canónicas de una función

Una función se puede representar por una serie de términos canónicos

Se llama término canónico de una función lógica a “todo producto o toda suma en los que aparecen todas las variables que componen una función, en su forma directa o inversa”. O sea que hay productos canónicos y sumas canónicas.

Un minitérmino o producto canónico es: “el producto de las variables en juego, o sus negaciones individuales que hacen que el producto valga 1 (uno)”, o, expresado de otro modo, “es la representación de una de las combinaciones de la tabla de verdad por medio del producto lógico”.

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

Formas normales o canónicas de una función

"Si se expresa una función como la suma lógica de aquellos minitérminos que en su tabla de verdad tengan valor 1, se obtiene la expresión de su forma normal disyuntiva (FND) ".

Un maxitérmino o suma canónica es: "la suma de las variables en juego, o sus negaciones individuales que hacen que la suma valga 0 (cero)", o, expresado de otro modo, "es la representación de una de las combinaciones de la tabla de verdad por medio de la suma lógica, pero teniendo en cuenta que los 0 en la combinación se expresan con la variable correspondiente sin negar y los 1, con la variable correspondiente negada".

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

Formas normales o canónicas de una función

Forma normal conjuntiva

“Si se expresa una función como el producto lógico de aquellos maxitérminos que en su tabla de verdad tengan valor 0, se obtiene la expresión de su forma normal conjuntiva (FNC) ”. La FND y la FNC se denominan formas normales o canónicas de una función.

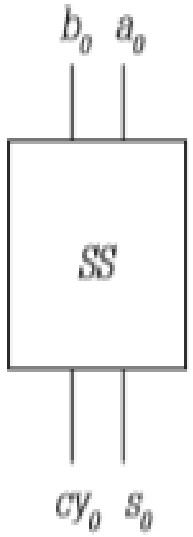
Ejemplos:

Sea un circuito semisumador, a partir de su tabla de verdad podremos obtener en primer lugar los minitérminos y los maxitérminos de la función y luego expresar la FND y la FNC de ésta.

Circuito sumador-binario en paralelo

Circuito semi-sumador (SS) o Half Adder (HA)

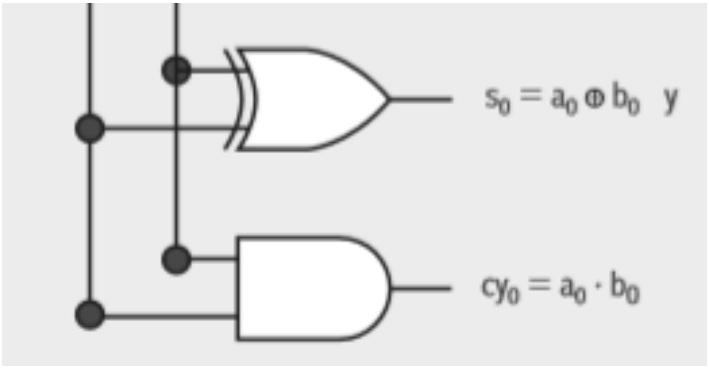
Es un circuito cuya función es sumar 2 bits sin tener en cuenta el acarreo anterior. Tiene dos salidas, una representa la suma y la otra, el valor del acarreo (carry).



donde:

$$\begin{array}{cccc} {}^{0}0 & {}^{0}0 & {}^{0}1 & {}^{1}1 \\ + & + & + & + \\ \underline{0} & \underline{1} & \underline{0} & \underline{1} \\ 0 & 1 & 1 & 0 \end{array}$$

a_0	b_0	s_0	c_0
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1



Formas normales o canónicas de una función

En la tabla se observan las formas canónicas de las funciones.

$$s_0 = a_0 \oplus b_0 \qquad y \qquad cy_0 = a_0 \cdot b_0$$

Para s_0:

$FND(s_0) = (\overline{a}_0 \cdot b_0) + (a_0 \cdot \overline{b}_0)$

$FNC(s_0) = (a_0 + b_0) \cdot (\overline{a}_0 + \overline{b}_0)$

y para cy_0:

$FND(cy_0) = a_0 \cdot b_0$

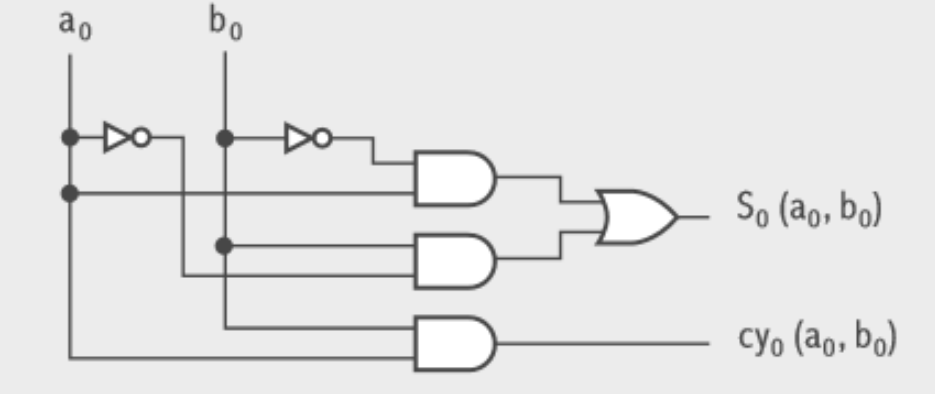
$FNC(cy_0) = (a_0 + b_0) \cdot (a_0 + \overline{b}_0) \cdot (\overline{a}_0 + b_0)$

a_0	b_0	s_0	c_0
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

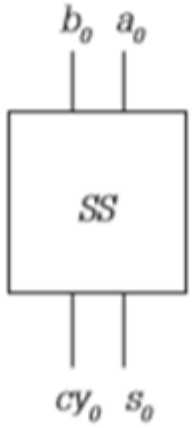
Formas normales o canónicas de una función

Forma normal conjuntiva

El circuito semi-sumador, podemos deducir las funciones de suma y acarreo correspondientes, ya sea por intermedio de la forma normal disyuntiva (FND) o por la forma normal conjuntiva (FNC). Elegiremos siempre la que más nos convenga, teniendo en cuenta la menor cantidad de términos que se han de representar, para favorecer, así, la implementación del circuito.



Paso a paso



b_0	a_0	s_0	cy_0	Miniterminos	Maxiterminos
0	0	0	0	$\overline{b_0} \cdot \overline{a_0}$	$b_0 + a_0$
0	1	1	0	$\overline{b_0} \cdot a_0$	$b_0 + \overline{a_0}$
1	0	1	0	$b_0 \cdot \overline{a_0}$	$\overline{b_0} + a_0$
1	1	0	1	$b_0 \cdot a_0$	$\overline{b_0} + \overline{a_0}$

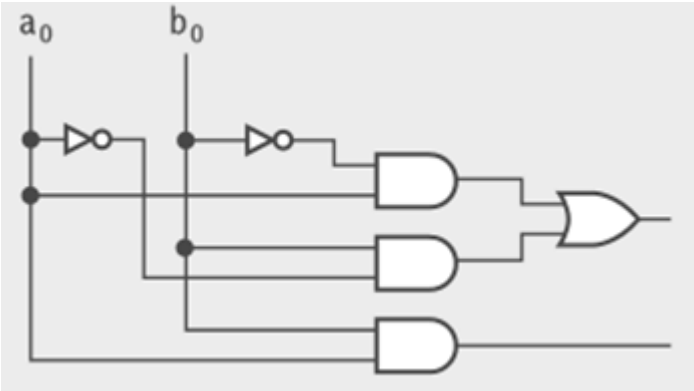
Defino las funciones: $f(a_0,b_0) = s_0$
$g(a_0,b_0) = cy_0$

FND: Suma de minitérminos que en la tabla de verdad de la función sean 1.

FNC: Producto de maxitérminos que en la tabla de verdad de la función sean 0

FND $f(a_0,b_0) = (\overline{b_0} \cdot a_0) + (b_0 \cdot \overline{a_0})$
$g(a_0,b_0) = (b_0 \cdot a_0)$

FNC $f(a_0,b_0) = (b_0 + a_0) \cdot (\overline{b_0} + \overline{a_0})$
$g(a_0,b_0) = (b_0 + a_0) \cdot (b_0 + \overline{a_0}) \cdot (\overline{b_0} + a_0)$



$f(a_0,b_0) = s_0$

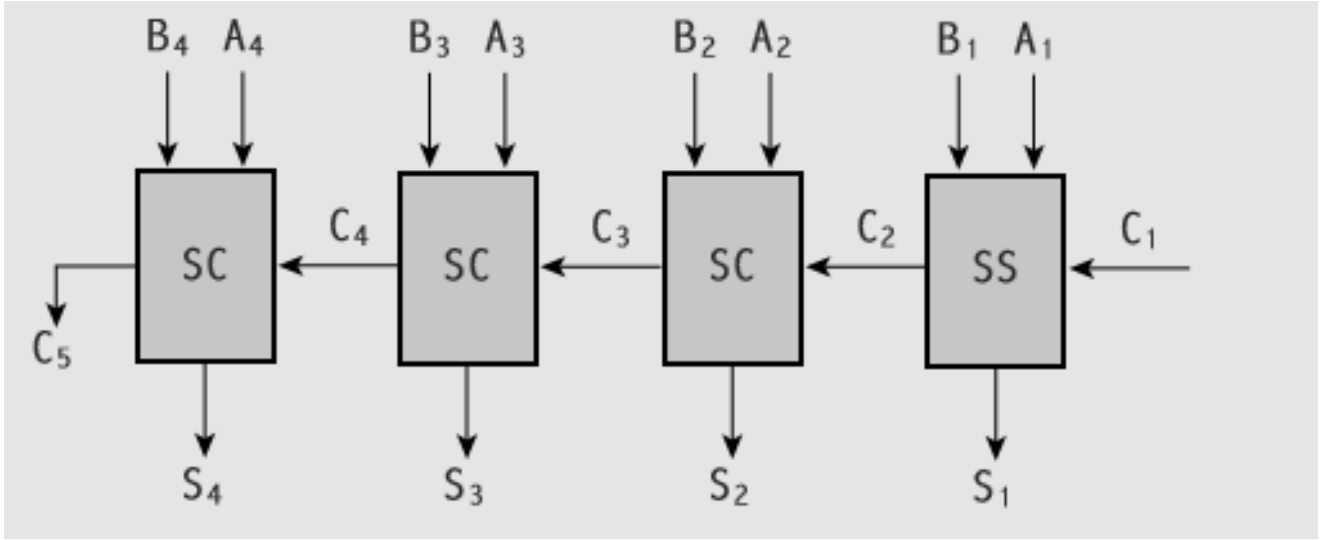
$g(a_0,b_0) = cy_0$

Circuito sumador-binario en paralelo

Ahora analizaremos cómo a partir de la expresión de una función booleana se puede confeccionar el circuito que lleve a cabo la operación que describe.

Supongamos que queremos sumar dos números de 8 dígitos cada uno:

Cy		*0*	*0*	*1*	*1*	*0*	*0*	*0*	*0*
A			*1*	*0*	*1*	*1*	*0*	*1*	*1*	*0*
+B	*+*		*0*	*0*	*0*	*1*	*1*	*0*	*0*	*1*
S			*1*	*1*	*0*	*0*	*1*	*1*	*1*	*1*



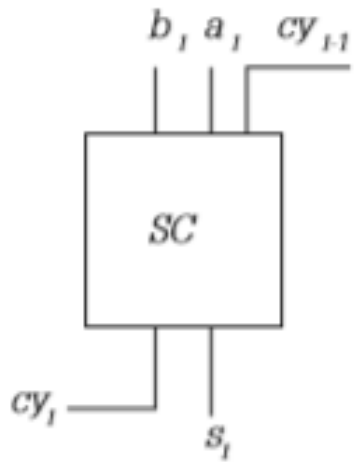
Sumador binario paralelo con acarreo serie

Circuito sumador-binario en paralelo

Circuito sumador completo (SC) o Full Adder (FA)

Este circuito tiene como finalidad sumar 2 bits y el acarreo anterior, generando como salidas el resultado de la suma y el nuevo acarreo.

Genéricamente:



donde:

$0\,0$	$0\,1$
+	+
0	0
0	1
a)	b)

b_i = b (variable que se ha de sumar)

a_i = a (variable que se ha de sumar)

cy_{i-1} = c (acarreo anterior)

s_i = S (resultado de la suma)

cy_i = C (nuevo acarreo)

a_i	b_i	cy_{i-1}	S_i	Cy_i
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

Formas normales o canónicas de una función

Ahora hallamos las formas normales para el sumador completo.

Recordemos que:

b_1 = b (variable que se ha de sumar) s_1 = S (resultado de la suma)

a_1 = a (variable que se ha de sumar) cy_1 = C (nuevo acarreo)

cy_1-1 = c (acarreo anterior)

a	b	c	S	C
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

$$FND_{(S)} = (\bar{a} \cdot \bar{b} \cdot c) + (\bar{a} \cdot b \cdot \bar{c}) + (a \cdot \bar{b} \cdot \bar{c}) + (a \cdot b \cdot c)$$

$$FNC_{(S)} = (a + b + c) \cdot (a + \bar{b} + \bar{c}) \cdot (\bar{a} + b + \bar{c}) \cdot (\bar{a} + \bar{b} + c)$$

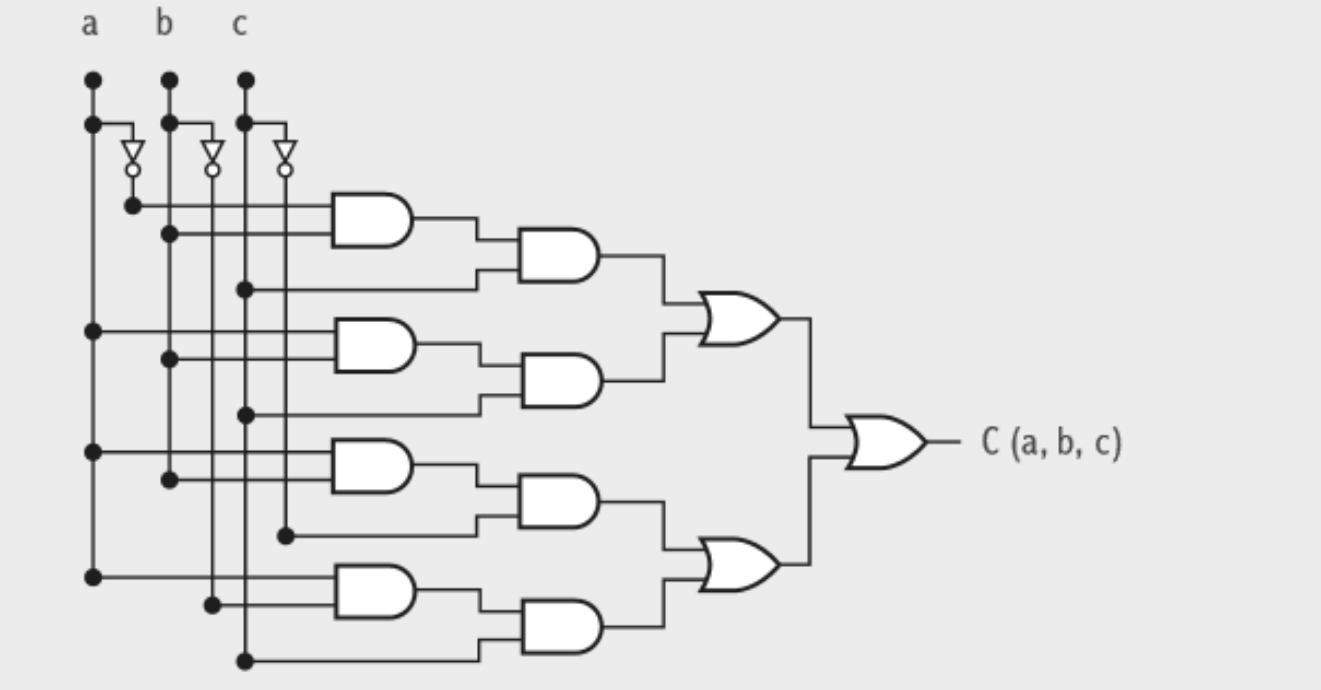
$$FND_{(C)} = (\bar{a} \cdot b \cdot c) + (a \cdot \bar{b} \cdot c) + (a \cdot b \cdot \bar{c}) + (a \cdot b \cdot c)$$

$$FNC_{(C)} = (a + b + c) \cdot (a + b + \bar{c}) \cdot (a + \bar{b} + c) \cdot (\bar{a} + b + c)$$

Formas normales o canónicas de una función

Forma normal conjuntiva

Si graficamos el circuito con las formas normales disyuntivas de S y C, obtendremos el diagrama de lógica del circuito sumador-completo o full adder.



Actividad

Circuito lógico de la función en Forma Normal Conjuntiva:

$$Z = (A+B+C)\cdot (A+\overline{B}+\overline{C})\cdot (\overline{A}+B+\overline{C})\cdot (\overline{A}+\overline{B}+C)$$

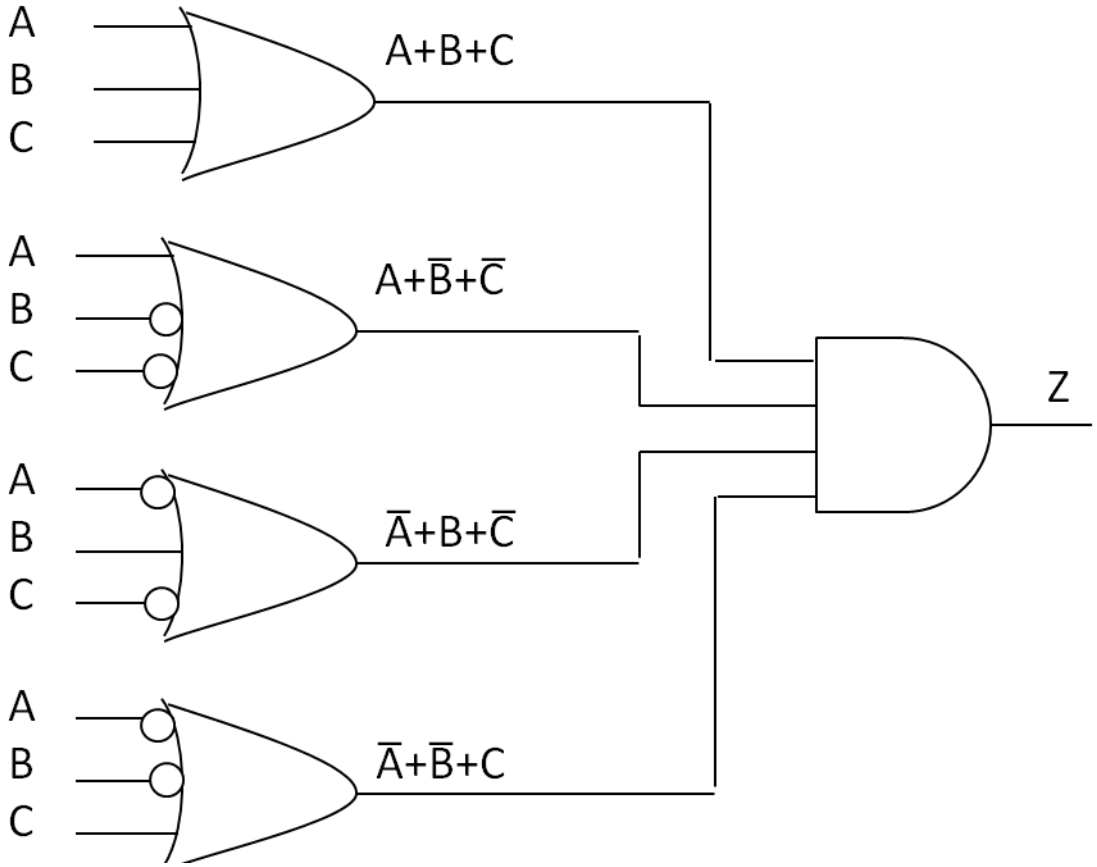
¿Cómo quedaría el esquema utilizando compuertas lógicas?

Actividad

Circuito lógico de la función en Forma Normal Conjuntiva:

$$Z = (A+B+C)\cdot(A+\overline{B}+\overline{C})\cdot(\overline{A}+B+\overline{C})\cdot(\overline{A}+\overline{B}+C)$$

¿Cómo quedaría el esquema utilizando compuertas lógicas?



Actividad

a) Calcular los miniterminos para la siguiente tabla

b) Realizar su simplificación con Algebra de Boole

c) Corroborar la tabla de verdad final con inicial

d) Hacer el circuito con compuerta lógicas

A	B	C	Z	m
0	0	0	**0**	$\bar{A}\cdot\bar{B}\cdot\bar{C}$
0	0	1	**1**	$\bar{A}\cdot\bar{B}\cdot C$
0	1	0	**1**	$\bar{A}\cdot B\cdot\bar{C}$
0	1	1	**0**	$\bar{A}\cdot B\cdot C$
1	0	0	**1**	$A\cdot\bar{B}\cdot\bar{C}$
1	0	1	**0**	$A\cdot\bar{B}\cdot C$
1	1	0	**0**	$A\cdot B\cdot\bar{C}$
1	1	1	**1**	$A\cdot B\cdot C$

Actividad

a) Calcular los miniterminos para la siguiente tabla

b) Realizar su simplificación con Algebra de Boole

c) Corroborar la tabla de verdad final con inicial

d) Hacer el circuito con compuerta lógicas

A	B	C	Z	m
0	0	0	**0**	$\bar{A}\cdot\bar{B}\cdot\bar{C}$
0	0	1	**1**	$\bar{A}\cdot\bar{B}\cdot C$
0	1	0	**1**	$\bar{A}\cdot B\cdot\bar{C}$
0	1	1	**0**	$\bar{A}\cdot B\cdot C$
1	0	0	**1**	$A\cdot\bar{B}\cdot\bar{C}$
1	0	1	**0**	$A\cdot\bar{B}\cdot C$
1	1	0	**0**	$A\cdot B\cdot\bar{C}$
1	1	1	**1**	$A\cdot B\cdot C$

En este ejemplo, los minitérminos a sumar serían:

$\bar{A}\cdot\bar{B}\cdot C$ $\quad A\cdot\bar{B}\cdot\bar{C}$

$\bar{A}\cdot B\cdot\bar{C}$ $\quad A\cdot B\cdot C$

La función quedaría descrita por:

$$Z = \bar{A}\cdot\bar{B}\cdot C + \bar{A}\cdot B\cdot\bar{C} + A\cdot\bar{B}\cdot\bar{C} + A\cdot B\cdot C$$

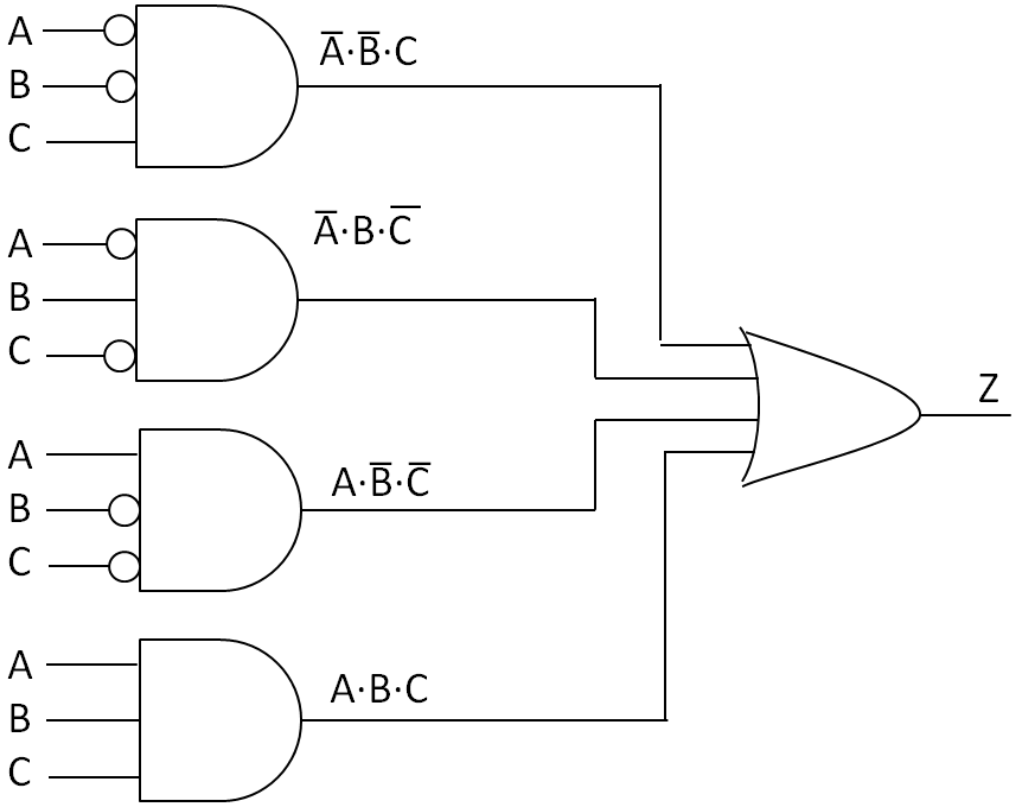
Se puede verificar reemplazando los valores para cada caso.

Actividad

Circuito lógico de la función en Forma Normal Disyuntiva:

$$Z = \overline{A}\cdot\overline{B}\cdot C + \overline{A}\cdot B\cdot\overline{C} + A\cdot\overline{B}\cdot\overline{C} + A\cdot B\cdot C$$

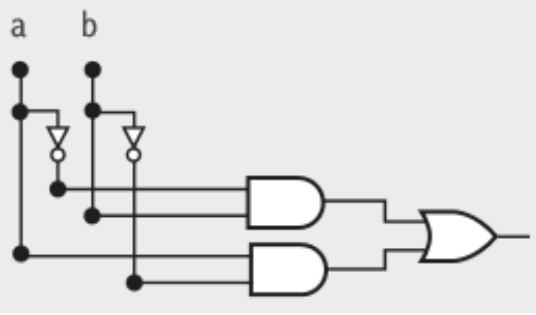
¿Cómo quedaría el esquema utilizando compuertas lógicas?



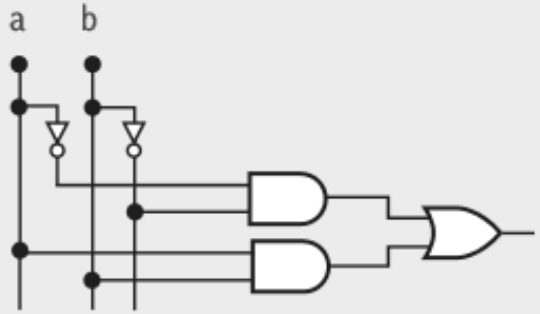
Circuitos equivalentes

Dos circuitos son equivalentes cuando son representados con distintas formas algebraicas pero responden a la misma tabla de verdad.

Ejemplos:



		(1)	(2)	
a	*b*	$\bar{a} \cdot b$	$a \cdot \bar{b}$	(1) + (2)
0	0	0	0	0
0	1	1	0	1
1	0	0	1	1
1	1	0	0	0



		(3)	(4)	
a	*b*	$a + b$	$\bar{a} + \bar{b}$	(3) · (4)
0	0	0	1	0
0	1	1	1	1
1	0	1	1	1
1	1	1	0	0

Los circuitos *A* y *B* representan las formas normales de la función a ⊕ b. Como ambas son expresiones diferentes obtenidas de la misma tabla, generan circuitos equivalentes.

Al circuito *A* le corresponde la función FND = ($\bar{a} \cdot b + a \cdot \bar{b}$), y al circuito *B* le corresponde la función FNC = (a + b) · ($\bar{a} + \bar{b}$).

UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

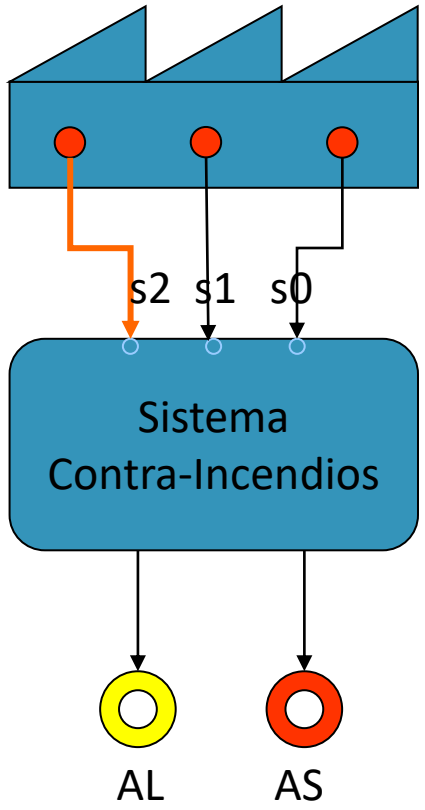
Aplicaciones

UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

Ejemplo de Funciones Booleanas

El sistema de seguridad contra incendios de un depósito funciona en base a tres sensores S0, S1 y S2. Cuando dos de estos sensores están activados (en “1”) se enciende una alarma luminosa [AL]. Además, si S2 se activa, también se enciende la alarma sonora [AS].

Ejemplo: Tabla de Verdad



s2	s1	s0	AL	AS
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	1	0
1	0	0	0	**1**
1	0	1	1	**1**
1	1	0	1	**1**
1	1	1	1	**1**

Ejemplo: Formas Literales

Expresión Literal Completa –Suma de Productos

AL = $\overline{S2}$.S1.S0 + S2.$\overline{S1}$.S0 + S2.S1.$\overline{S0}$ + S2.S1.S0

AS = S2.$\overline{S1}$.$\overline{S0}$ + S2.$\overline{S1}$.S0 + S2.S1.$\overline{S0}$ + S2.S1.S0

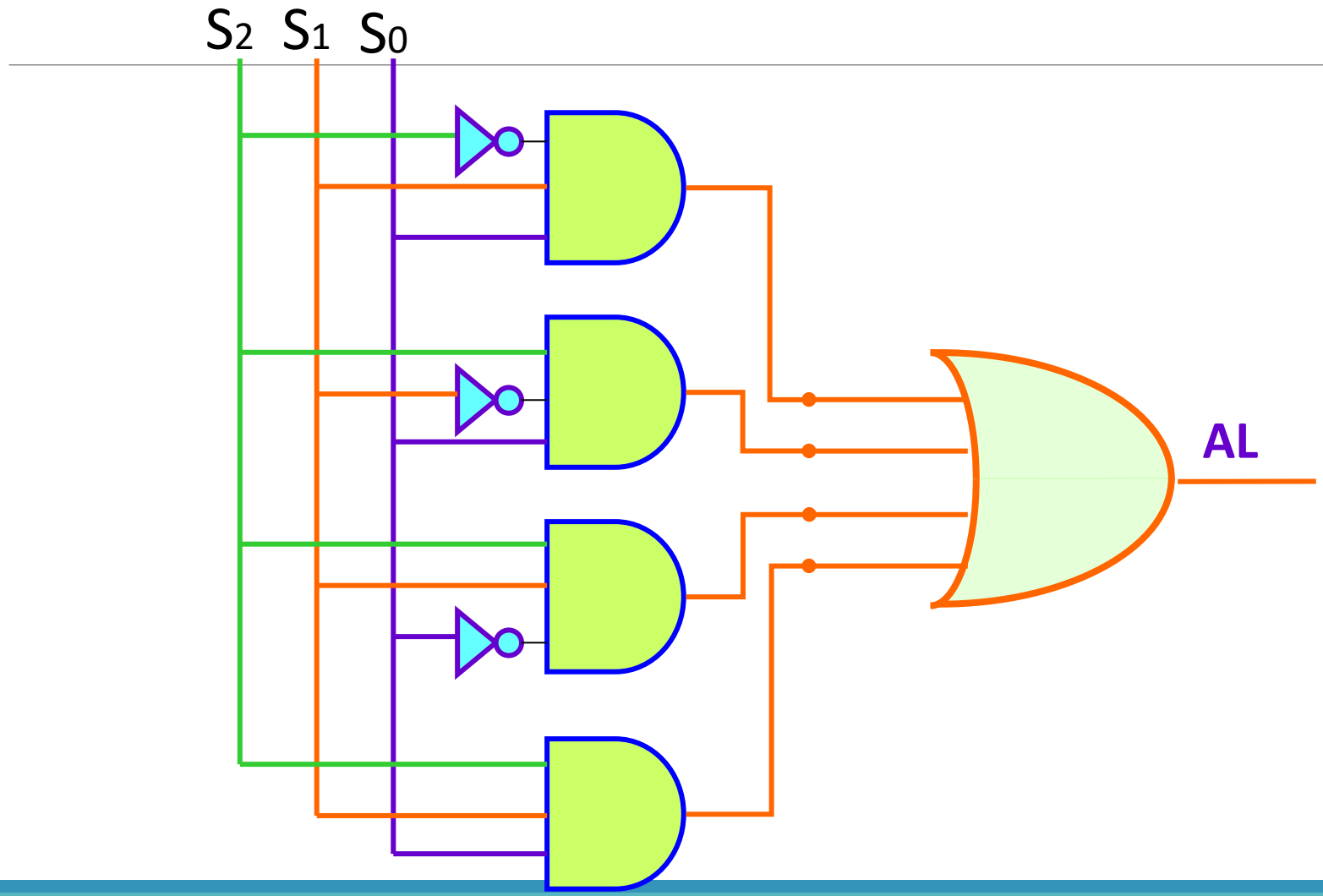
Expresión Literal en Minitérminos

AL = $m_3 + m_5 + m_6 + m_7$

AS = $m_4 + m_5 + m_6 + m_7$

Dado que todos los términos de esta forma POLINÓMICA contienen todas las variables, esta expresión se denomina CANÓNICA

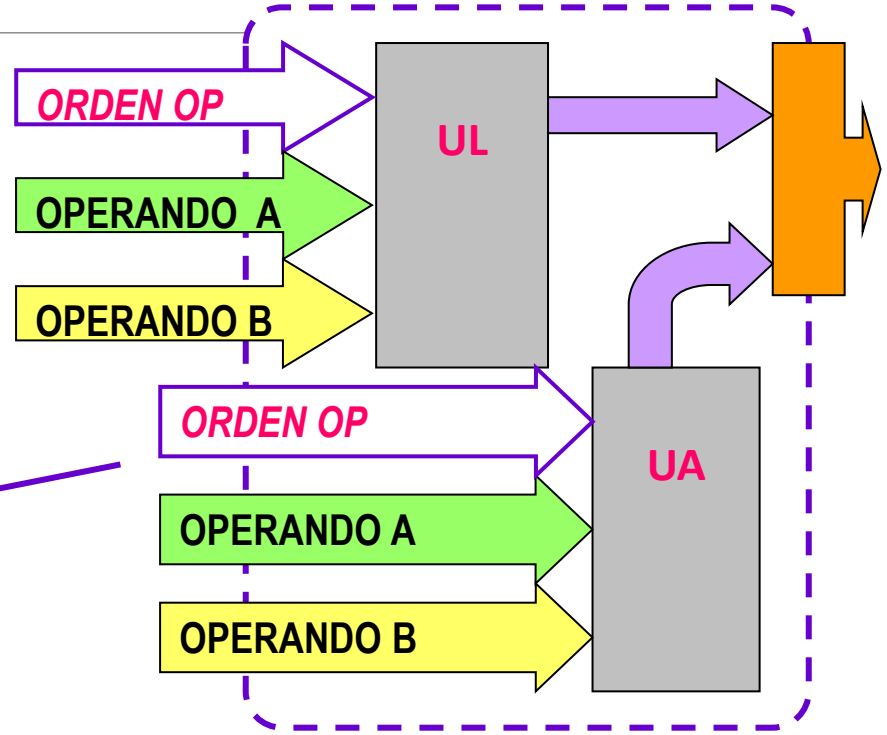
Ejemplo: Circuitos



Concepto de UAL

La Unidad Aritmética y Lógica es el componente de la CPU que realiza las operaciones lógicas (AND, OR, XOR, etc.) y aritméticas (en principio suma y resta).

Podemos pensar entonces en una estructura con dos operadores básicos, uno lógico –**UL** y uno aritmético –**UA**, como muestra la figura adjunta

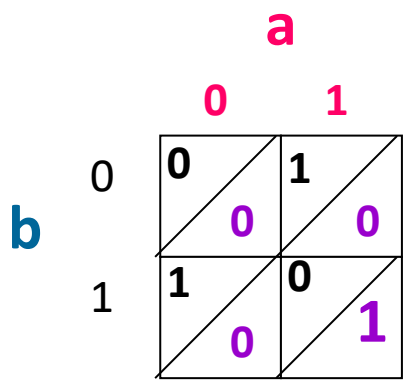


Como se observa en esta figura, los operadores reciben los operandos (datos) y una orden de operación (para indicar QUE operación deben hacer).

Sumador Elemental: Semisumador

El sumador elemental será capaz de sumar un par de bits “a” y “b”.

La Tabla de Verdad se deriva a partir de la Tabla de Suma Aritmética.



a	b	S suma	C acarreo
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

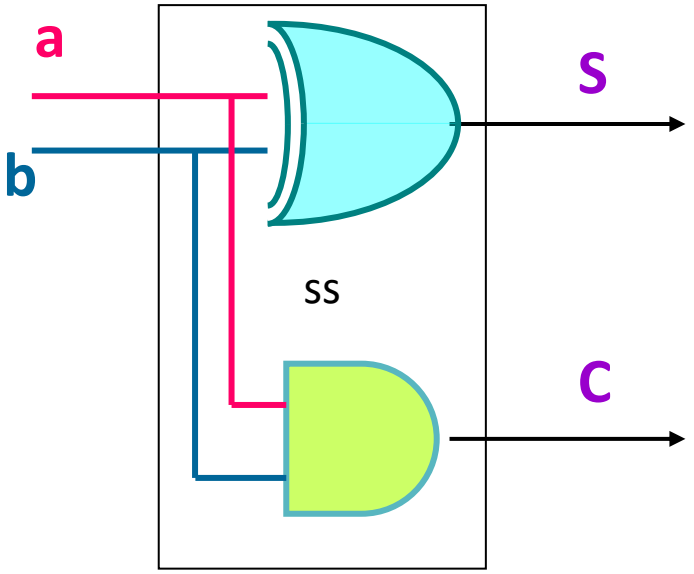
$S = a \oplus b$

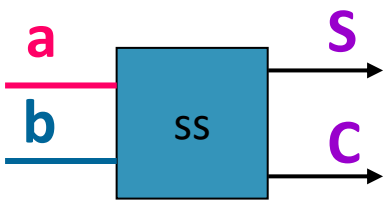
$C = a \bullet b$

Sumador Elemental: Semisumador

Para sumar dos cantidades de *n* bits es necesario disponer de *n* bloques funcionales como el obtenido.

Si bien permiten sumar bit a bit, al no tener en cuenta el acarreo no implementan correctamente la suma. Por este motivo este bloque se denomina **semisumador** o *half adder*.





Sumador Elemental: Sumador Completo

Para realizar la suma de dos cadenas de bits es necesario tener en cuenta el acarreo que cada "etapa" le pasa a la siguiente. La Tabla de Verdad adjunta tiene en cuenta esta situación.

Ahora las funciones son

$$S = a \oplus b \oplus c_{-1}$$

$$C = (a \bullet b) + (a \oplus b) \bullet c_{-1}$$

El bloque funcional que las implementa se denomina **sumador completo** *o full adder*.

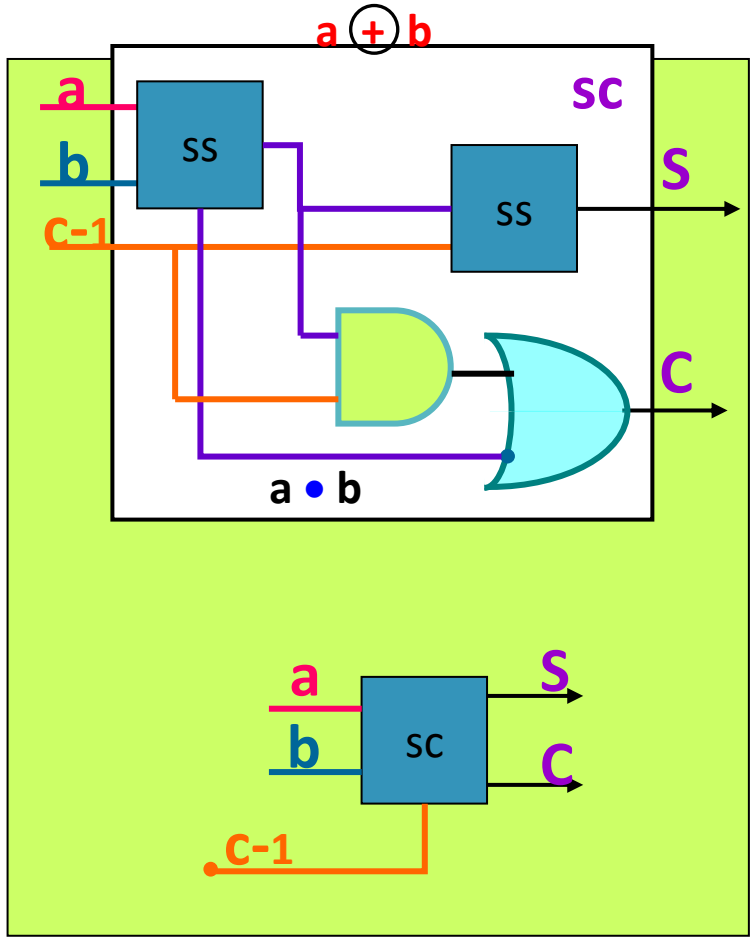
a	b	c_{-1}	S	C
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	**1**
1	0	0	1	0
1	0	1	0	**1**
1	1	0	0	**1**
1	1	1	1	**1**

Sumador Elemental: Sumador Completo

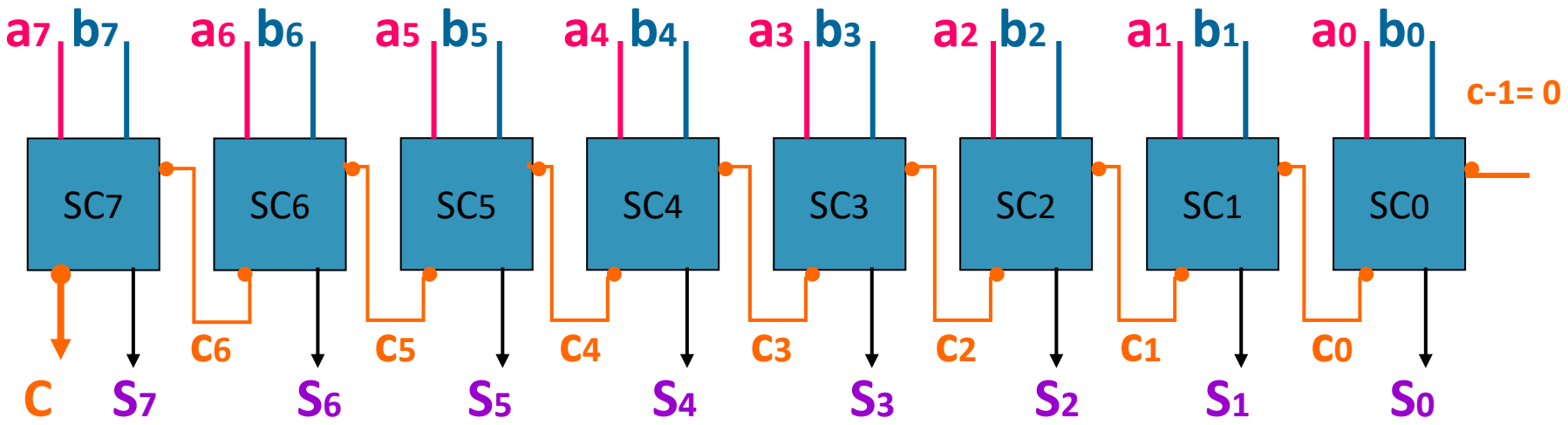
Este bloque se denomina **sumador completo** o *full adder*.

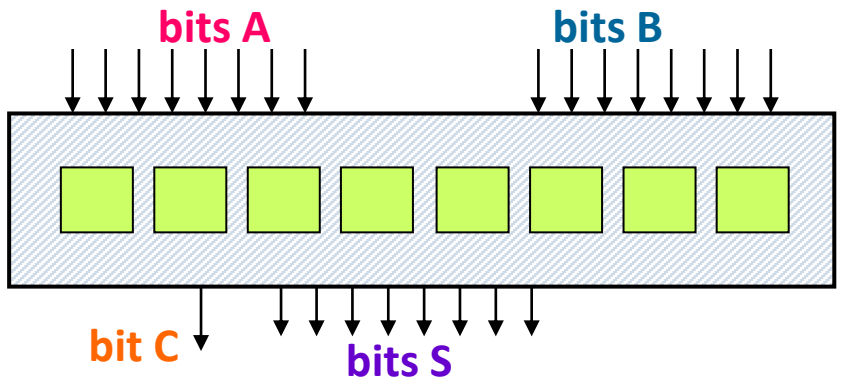
Permite sumar bit a bit, y dado que tiene en cuenta el acarreo posibilita la implementación correcta de una suma de n bits.

- Justificar la implementación propuesta para el sumador completo.



Sumador de 8 bits





La Resta en la UAL

La resta se implementa utilizando el mismo dispositivo sumador, gracias a la propiedad de la notación posicional de cantidades denominada COMPLEMENTO.

Para un número N de “p” dígitos expresado en base “b” se definen:

- COMPLEMENTO DIRECTO (a la base menos uno)

  $$CD(N) = (b^p - 1) - N$$

- COMPLEMENTO AUTENTICO (a la base)

  $$CA(N) = (b^p) - N$$

Y de allí

$$CA(N) = CD(N) + 1$$

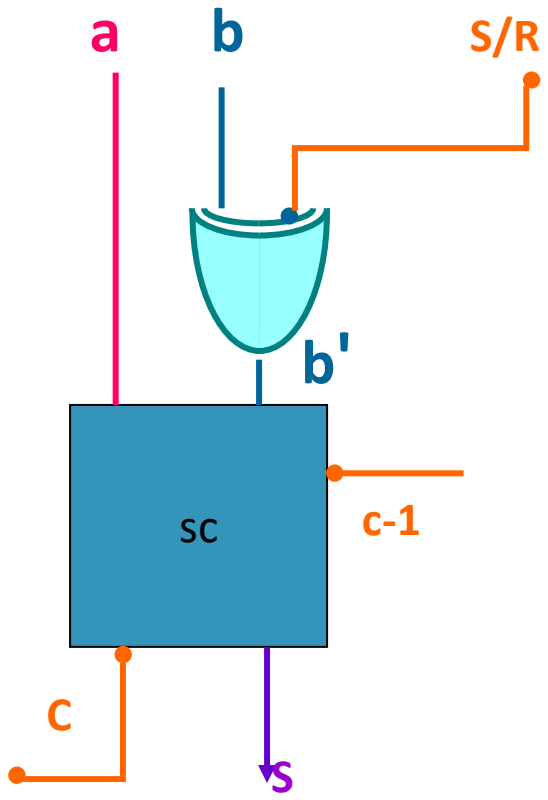
Según se ha visto, la ***resta*** puede obtenerse “***sumando***” al minuendo el ***complemento auténtico*** del sustraendo.

En binario el complemento directo se obtiene invirtiendo todos los bits del registro. Por lo tanto, es sencillo deducir el esquema del sumador-restador elemental que realice las operaciones (a + b) y (a - b), según se indique.

Sumador – Restador en Complemento Auténtico

S/R	b	b'
0	0	0
0	1	1
1	**0**	**1**
1	**1**	**0**

Si S/R es cero (0) el bit “b” se transfiere sin cambios; si es uno (1), el bit transferido es el complemento directo del recibido



Sumador-Restador de n bits