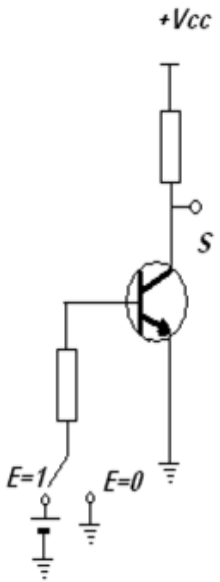
Electrónica Digital

De las puertas lógicas y su implementación con transistores BJT. Algo de la composición de circuitos lógicos a partir de las tablas de verdad de las funciones de Boole y cómo se puede sumar dos números decimales en expresión binaria Comparador binario, comparador de cuatro bits acoplamiento de dos comparadores para comparar números de ocho bits .

Implementación con transistores BJT de las puertas:

- NOT 
- NOR 
- AND 

INVERSOR
PUERTA **NOT**



esquema del circuito inversor con un transistor

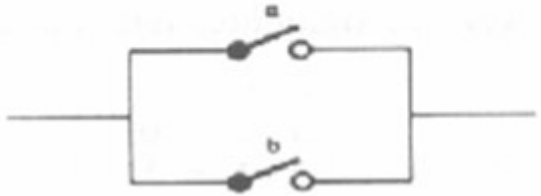
representación con símbolos americano y europeo

Recordemos que los valores de tensión para la entrada sabemos que sólo pueden ser 0 y 1 por lo que de aquí en adelante no se dibujarán ni la pila ni la conexión a tierra.

E	S
0	1
1	0

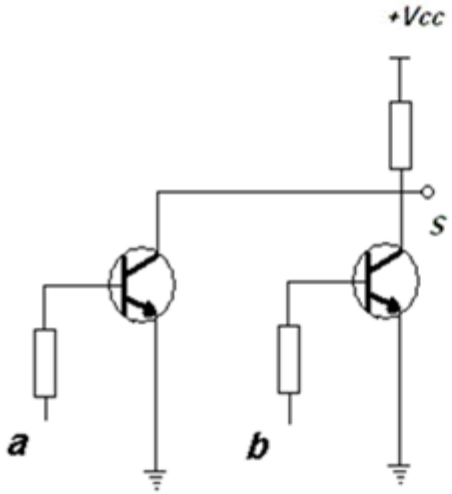
Puertas Lógicas NOR con transistores.

Dado que la suma de llaves se definió como:



= (a+b)

Resulta totalmente comprensible presentar la puerta OR como dos transistores en paralelo. Pero colocando dos puertas inversoras en paralelo se obtiene la puerta NOR .

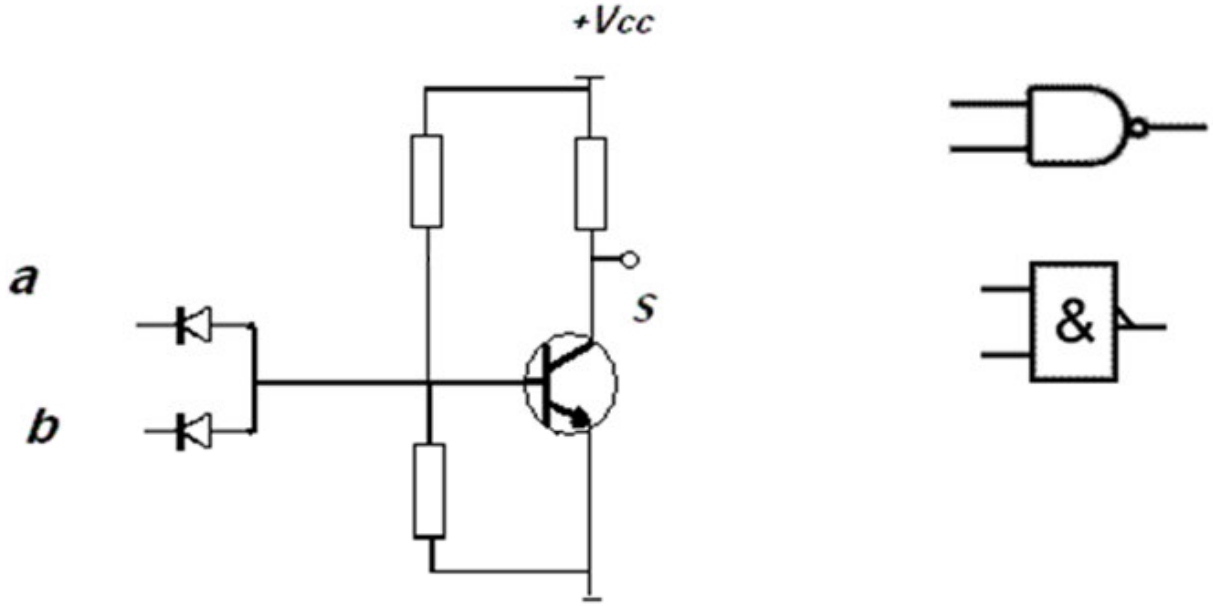


a	b	s	$\bar{s} = (a + b)$
0	0	1	0
0	1	0	1
1	0	0	1
1	1	0	1

esquema del circuito de una puerta NOR hecho con dos transistores.
Tabla de verdad de dicho circuito

Puertas Lógicas NAND con transistores.

Además se puede armar una puerta NAND, en base a lo aprendido con el transistor en modo de inversor



esquema del circuito de una puerta NAND hecho con un transistor y dos diodos y símbolos en versión americana y europea

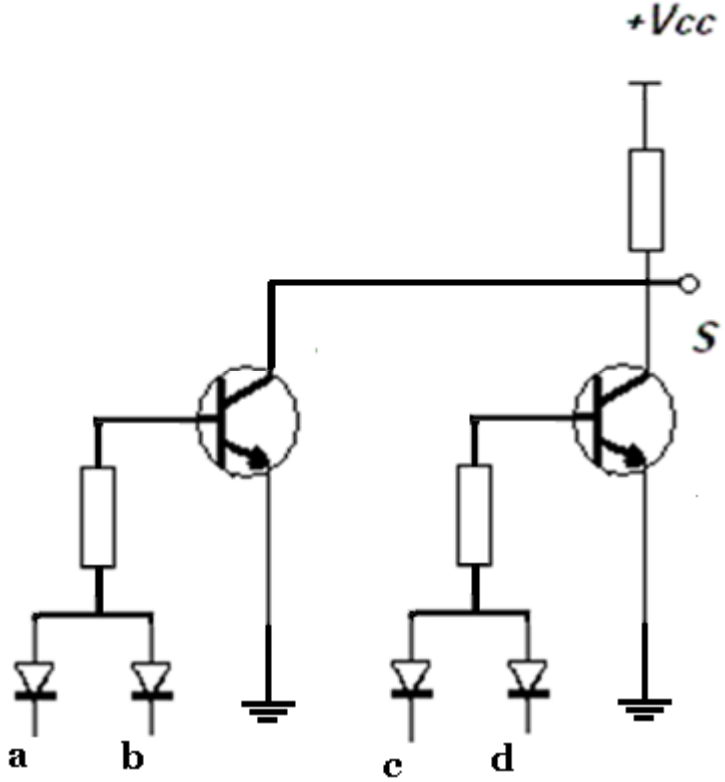
a	b	S	$\overline{S} = a * b$
0	0	1	0
0	1	1	0
1	0	1	0
1	1	0	1

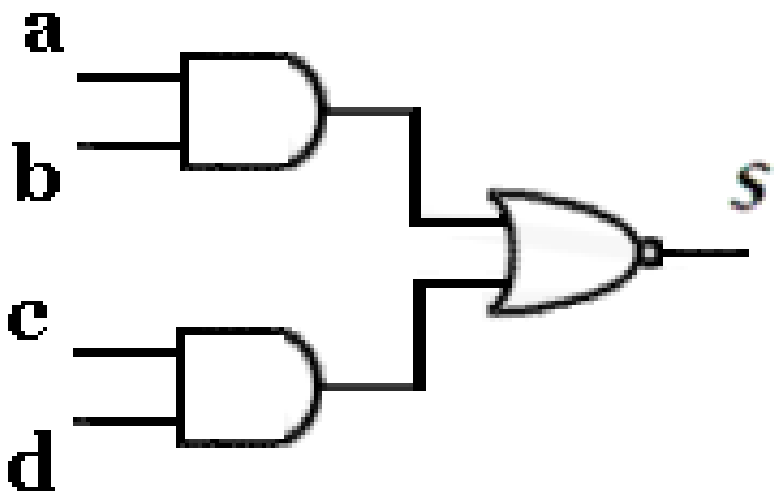
Cualquier función dentro del álgebra de Boole se puede obtener a partir de la tabla de verdad, usando las operaciones de inversión, producto y suma.

a	b	c	f(a,b,c)	Aporta con el término
0	0	0	0	
0	0	1	1	$\bar{a} * \bar{b} * c$
0	1	0	0	
0	1	1	0	
1	0	0	0	
1	0	1	0	
1	1	0	1	$a * b * \bar{c}$
1	1	1	1	$a * b * c$

$$f(a,b,c) = \bar{a} * \bar{b} * c + a * b * \bar{c} + a * b * c$$

Es así que adopta mucha importancia la puerta ANDORI que se puede implementar fácilmente





$$S = f(a,b,c,d) = a * b + c * d$$

Notar que para hacer varios productos sólo hay que agregar diodos.

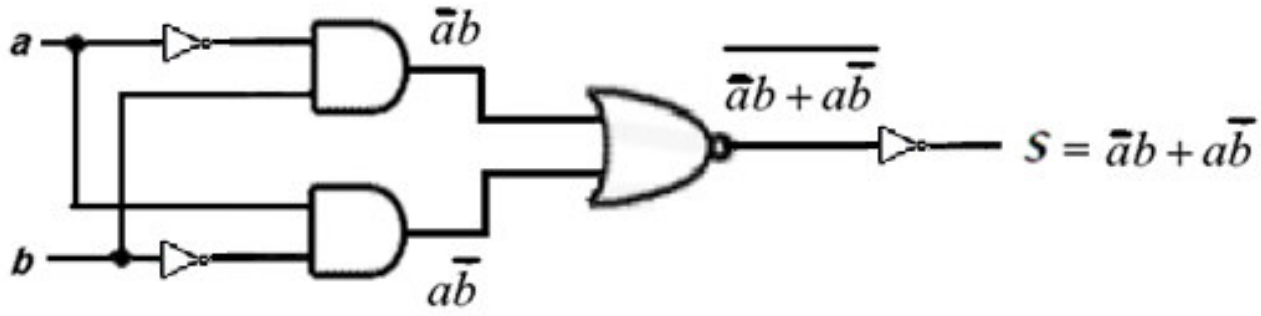
Puerta XOR usando la ANDORI

a	b	S
0	0	0
0	1	1
1	0	1
1	1	0

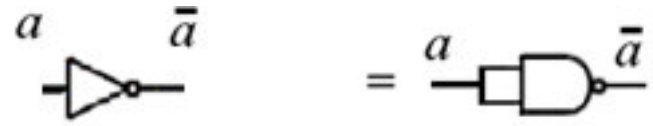
Se puede ver que la función que representa a esta tabla es:

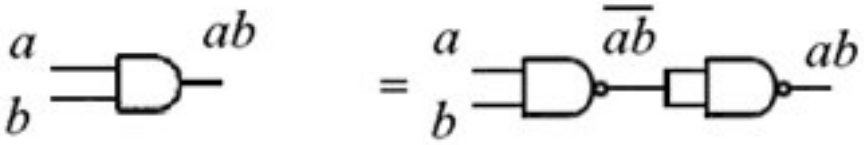
$$f(a,b) = \breve{a}b + a\breve{b}$$

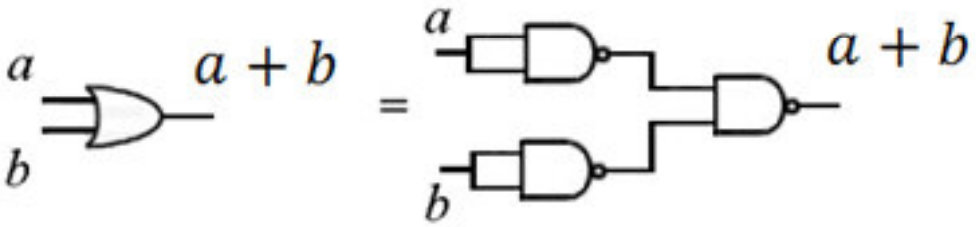
y cuyo símbolo es:



Capacidad de desarrollar cualquier función sólo con puertas NAND







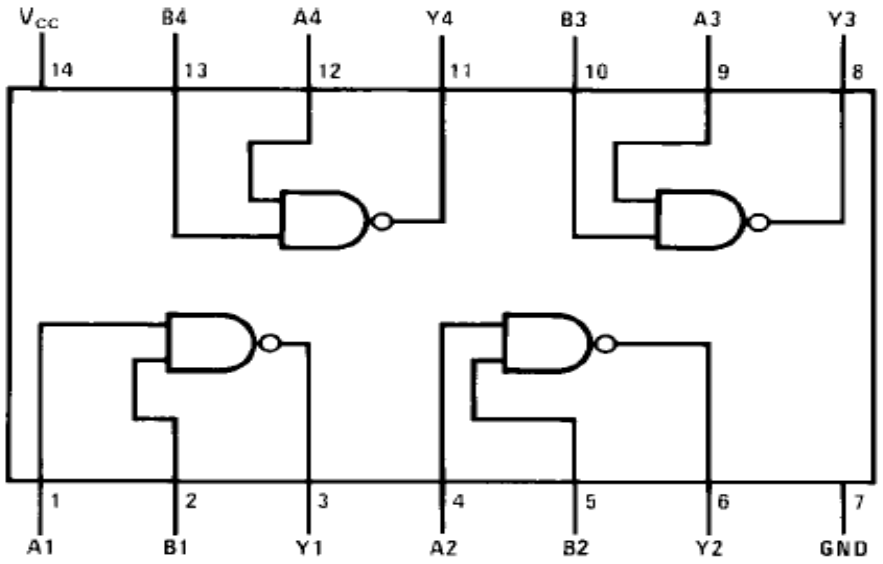
$$a + b = \overline{\overline{a + b}} = \bar{a} * \bar{b}$$

Esta misma propiedad la tiene la puerta NOR

54LS00/DM54LS00/DM74LS00
2-Input NAND Gates

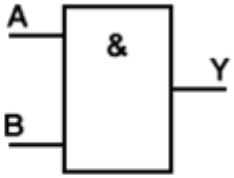
General Description

This device contains four independent gates each of which performs the logic NAND function.

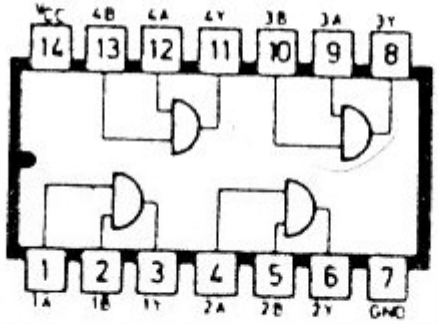


Se justifica plenamente la existencia de chips de pequeña escala de integración (SSI) con los que cualquier persona puede desarrollar el dispositivo lógico electrónico que desee (incluso una computadora) desarrollando las funciones a partir de la tabla de verdad y uniendo los terminales de manera adecuada

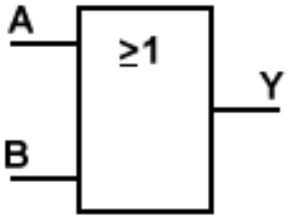
AND

	
Símbolo Americano	Símbolo Europeo

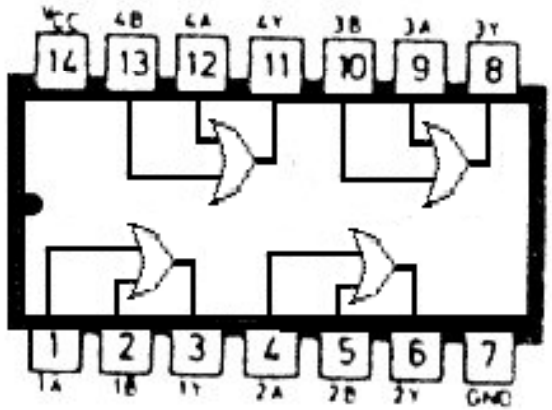
A	B	S
0	0	0
0	1	0
1	0	0
1	1	1



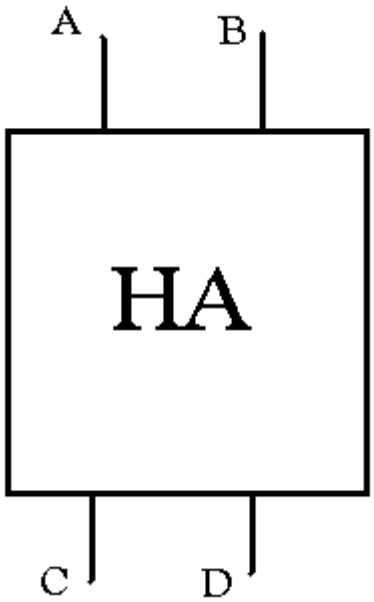
7408

OR 	
Símbolo Americano	Símbolo Europeo

A	B	S
0	0	0
0	1	1
1	0	1
1	1	1

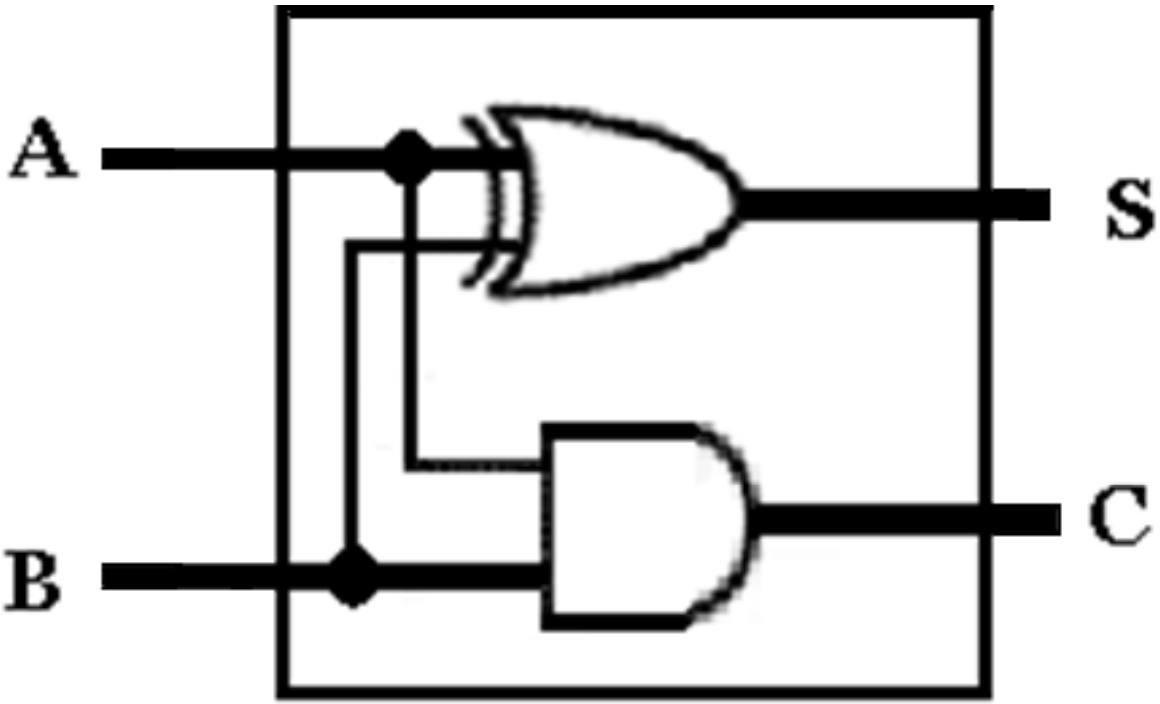


7432



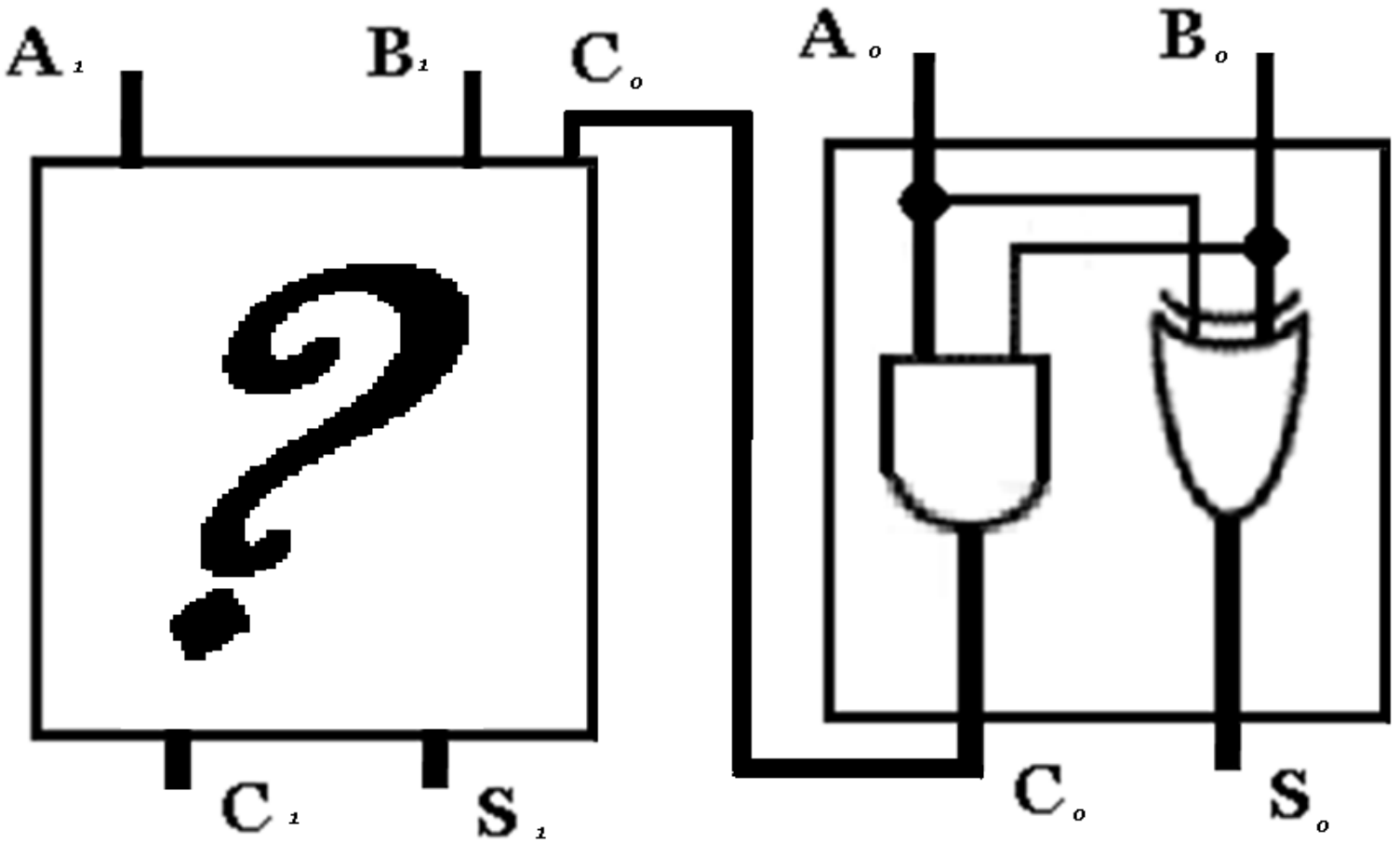
Entrada		Salida		
A	B	Suma	C	D
0	0	00	0	0
0	1	01	0	1
1	0	01	0	1
1	1	11	1	0

Semisumador

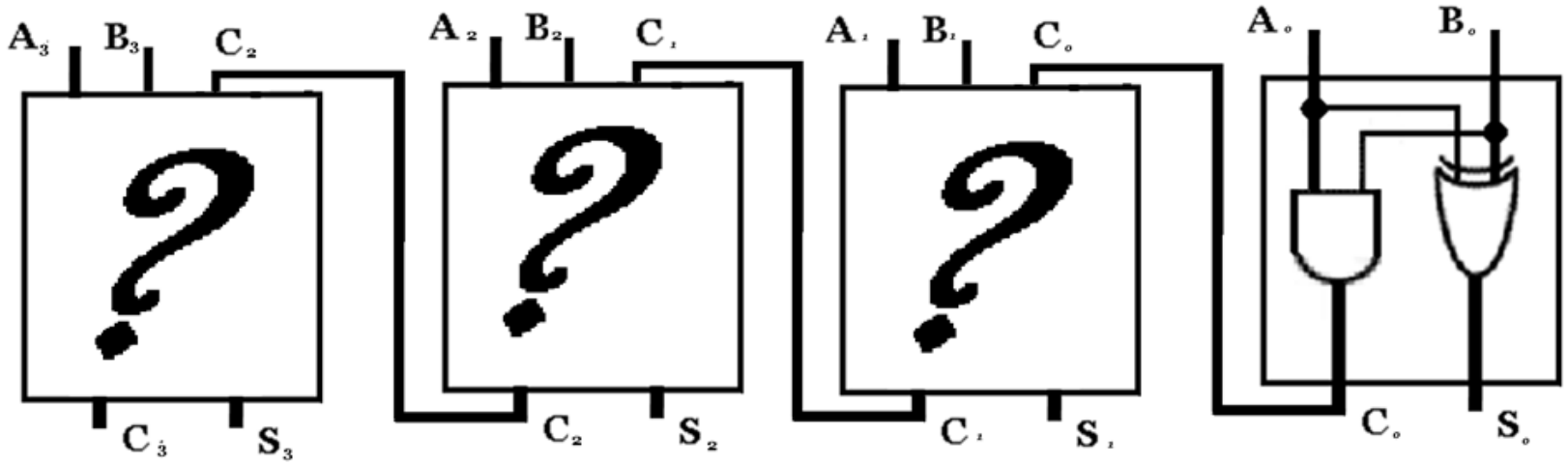


Si el dispositivo sumador para el bit de menor peso exige como resultado dos bits de salida, para sumar el bit de segundo orden en el peso tendré que tener en cuenta el acarreo de la suma del bit de menor peso

Nos queda entonces averiguar cómo diseñamos el sumador de dos bits que tenga en cuenta el acarreo del bit anterior



Y un sumador de cuatro bits
$A+B = A_3 A_2 A_1 A_0 + B_3 B_2 B_1 B_0$
tendría esta forma



La tabla de verdad generalizada para cualquier bit de orden superior al más bajo deberá ser:

	A_1	B_1	C_0	S_1	C_1
Linea	A_n	B_n	C_{n-1}	S_n	C_n
0	0	0	0	0	0
1	0	0	1	1	0
2	0	1	0	1	0
3	0	1	1	0	1
4	1	0	0	1	0
5	1	0	1	0	1
6	1	1	0	0	1
7	1	1	1	1	1

$$S_n = \overline{A_n} * \overline{B_n} * C_{n-1} + \overline{A_n} * B_n * \overline{C_{n-1}} + A_n * \overline{B_n} * \overline{C_{n-1}} + A_n B_n C_{n-1}$$

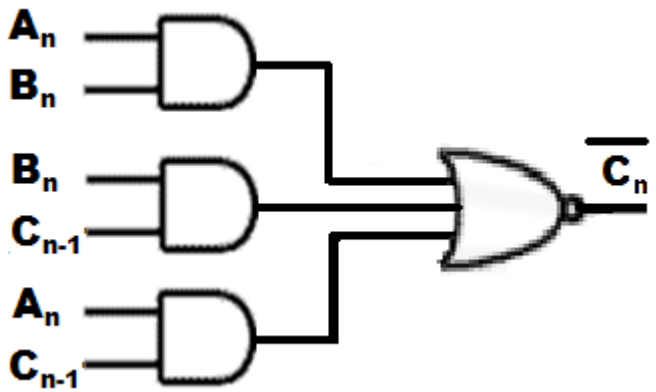
$$C_n = \overline{A_n} B_n C_{n-1} + A_n \overline{B_n} C_{n-1} + A_n B_n \overline{C_{n-1}} + A_n B_n C_{n-1}$$

$$C_n = \overline{A_n} B_n C_{n-1} + A_n \overline{B_n} C_{n-1} + A_n B_n \overline{C_{n-1}} + A_n B_n C_{n-1}$$

$$C_n = \overline{A_n} B_n C_{n-1} + A_n \overline{B_n} C_{n-1} + A_n B_n \overline{C_{n-1}} +$$

$$+ A_n B_n C_{n-1} + A_n B_n C_{n-1} + A_n B_n C_{n-1}$$

$$= B_n C_{n-1} + A_n C_{n-1} + A_n B_n$$



Linea	A_n	B_n	C_{n-1}	$\overline{A_n}$	$\overline{B_n}$	$\overline{C_{n-1}}$	S_n	C_n	$\overline{S_n}$	$\overline{C_n}$	$\overline{\text{Linea}}$
0	0	0	0	1	1	1	0	0	1	1	7
1	0	0	1	1	1	0	1	0	0	1	6
2	0	1	0	1	0	1	1	0	0	1	5
3	0	1	1	1	0	0	0	1	1	0	4
4	1	0	0	0	1	1	1	0	0	1	3
5	1	0	1	0	1	0	0	1	1	0	2
6	1	1	0	0	0	1	0	1	1	0	1
7	1	1	1	0	0	0	1	1	0	0	0

Los valores de $\overline{S_n}$ y $\overline{C_n}$ se obtienen por conclusión

$$\overline{C_n} = (\overline{B_n})(\overline{C_{n-1}}) + (\overline{A_n})(\overline{C_{n-1}}) + (\overline{A_n})(\overline{B_n})$$

Definiendo $D_n = (A_n + B_n + C_{n-1})(\overline{C_n})$

$$D_n = (A_n + B_n + C_{n-1})\{(\overline{B_n})(\overline{C_{n-1}}) + (A_n)(\overline{C_{n-1}}) + (\overline{A_n})(\overline{B_n})\}$$

$$= A_n(\overline{B_n})(\overline{C_{n-1}}) + A_n(\overline{A_n})(\overline{C_{n-1}}) + A_n(\overline{A_n})(\overline{B_n}) +$$

$$+ B_n(\overline{B_n})(\overline{C_{n-1}}) + B_n(\overline{A_n})(\overline{C_{n-1}}) + B_n(\overline{A_n})(\overline{B_n}) +$$

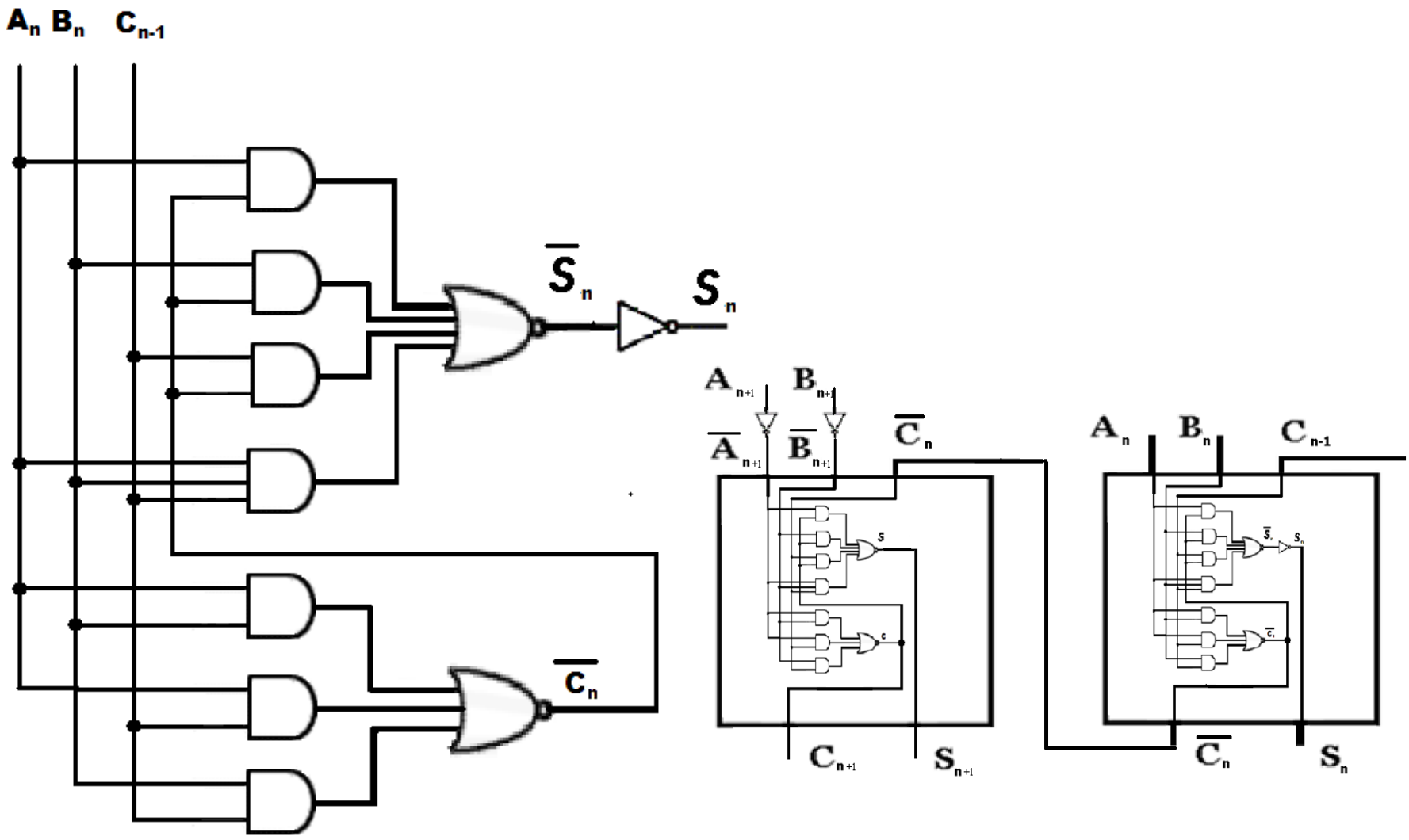
$$+ C_{n-1}(\overline{B_n})(\overline{C_{n-1}}) + C_{n-1}(\overline{A_n})(\overline{C_{n-1}}) + C_{n-1}(\overline{A_n})(\overline{B_n})$$

y comparando con la expresión para S_n

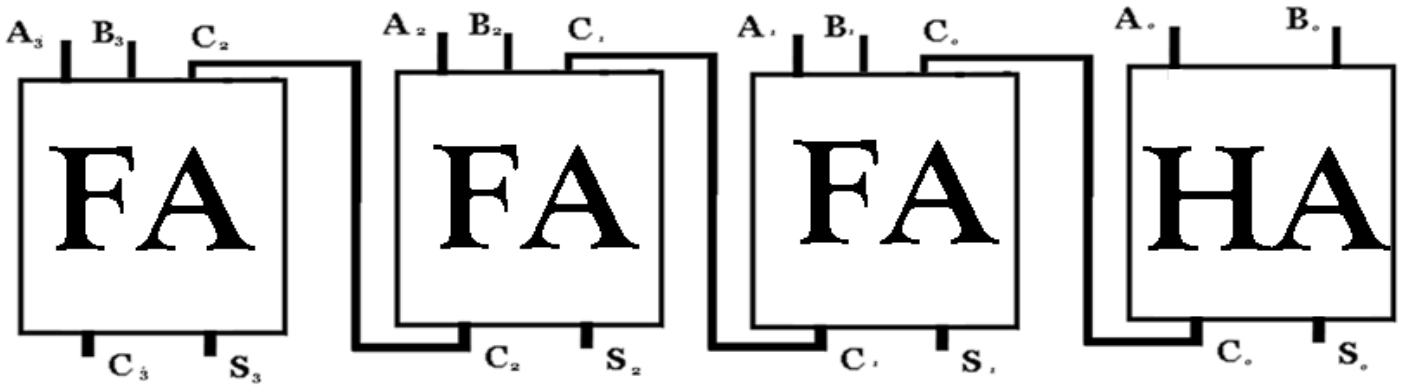
$$S_n = \overline{A_n} * \overline{B_n} * C_{n-1} + \overline{A_n} * B_n * \overline{C_{n-1}} + A_n * \overline{B_n} * \overline{C_{n-1}} + A_n B_n C_{n-1}$$

O sea

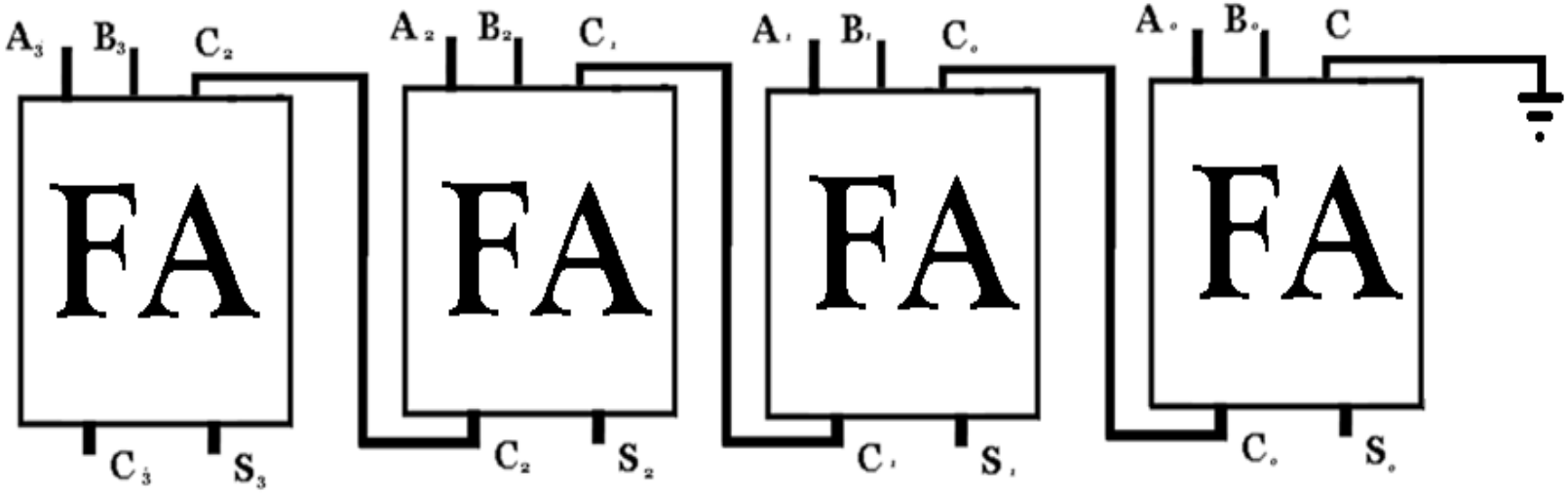
$$S_n = A_n\,\overline{C_n} + B_n\,\overline{C_n} + C_{n-1}\,\overline{C_n} + A_n\,B_n\,C_{n-1}$$


A_n B_n C_{n-1}
$\overline{S}_n$
S_n
$\overline{C}_n$
A_{n+1} B_{n+1}
$\overline{A}_{n+1}$ $\overline{B}_{n+1}$
$\overline{C}_n$
C_{n+1} S_{n+1}
A_n B_n C_{n-1}
$\overline{C}_n$ S_n

Full Adder



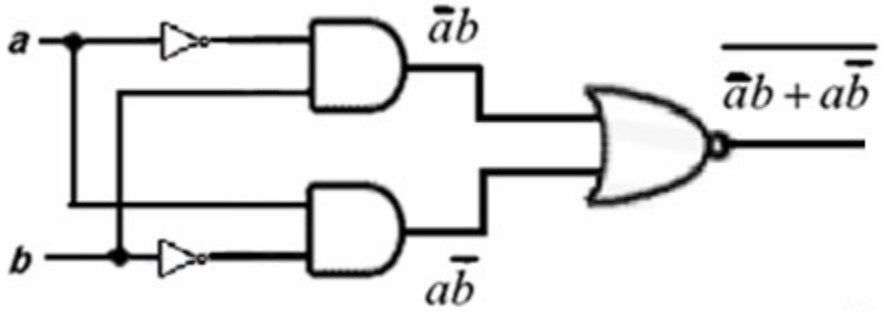
Sumador de cuatro bits

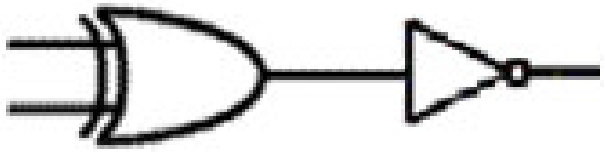


COMPARADOR DIGITAL DE UN BIT

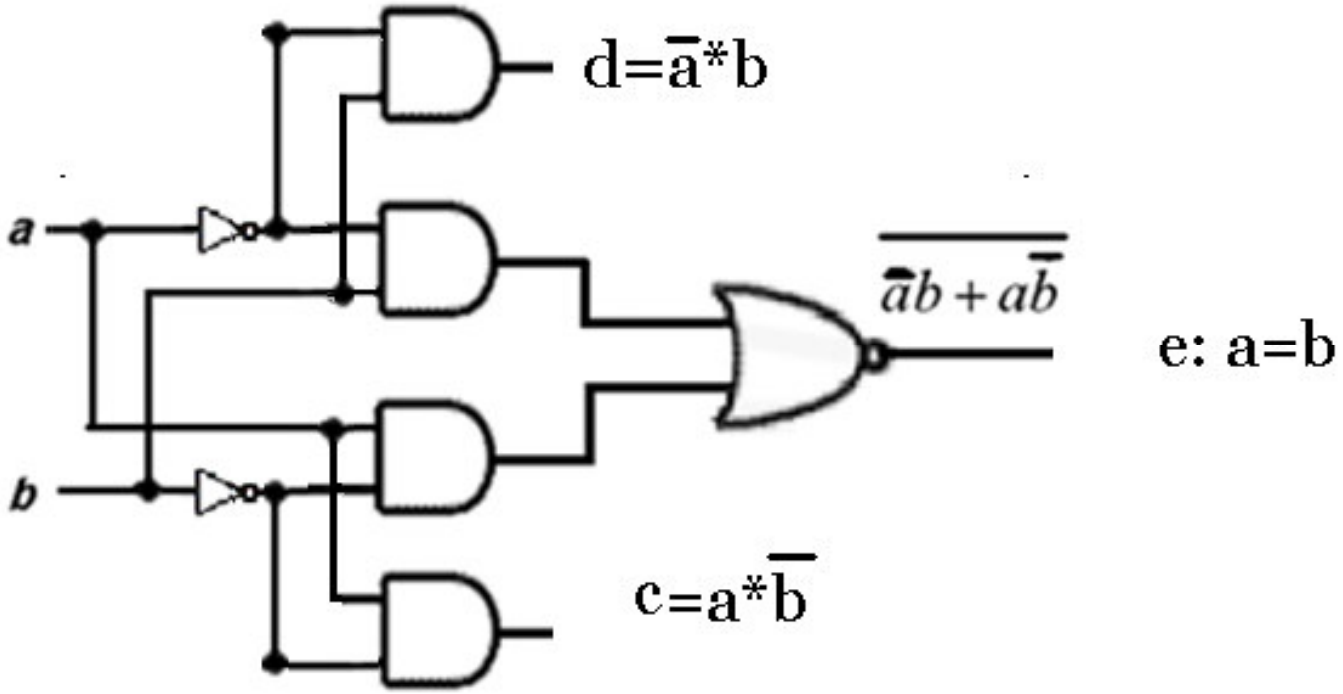
a	b	$\bar{a}$	$\bar{b}$	c = a*$\bar{b}$	d = $\bar{a}$*b	[($\bar{a}$*b)+(a*$\bar{b}$)]	$\overline{[(\bar{a}*b)+(a*\bar{b})]}$
0	0	1	1	0	0	0	1
0	1	1	0	0	1	1	0
1	0	0	1	1	0	1	0
1	1	0	0	0	0	0	1

a>b a<b





COMPARADOR DE UN BIT CON PUERTA ANDORI



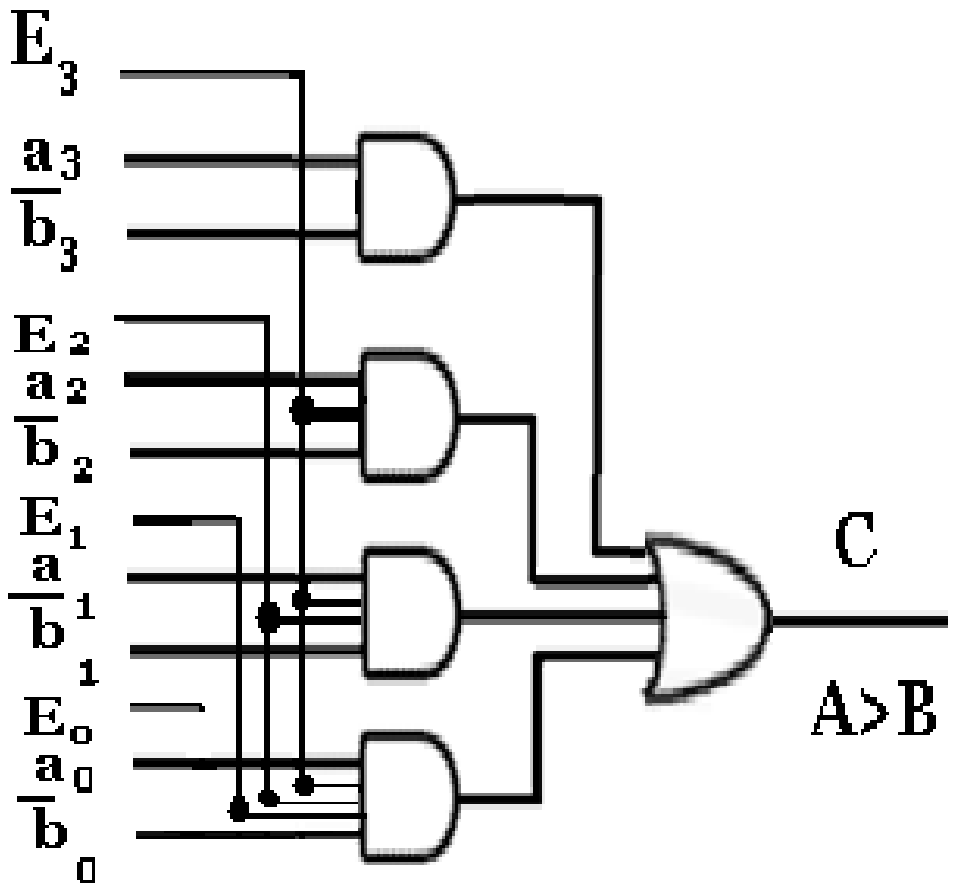
- C: a > b
- E: a = b
- D: a < b

COMPARADOR DE CUATRO BITS

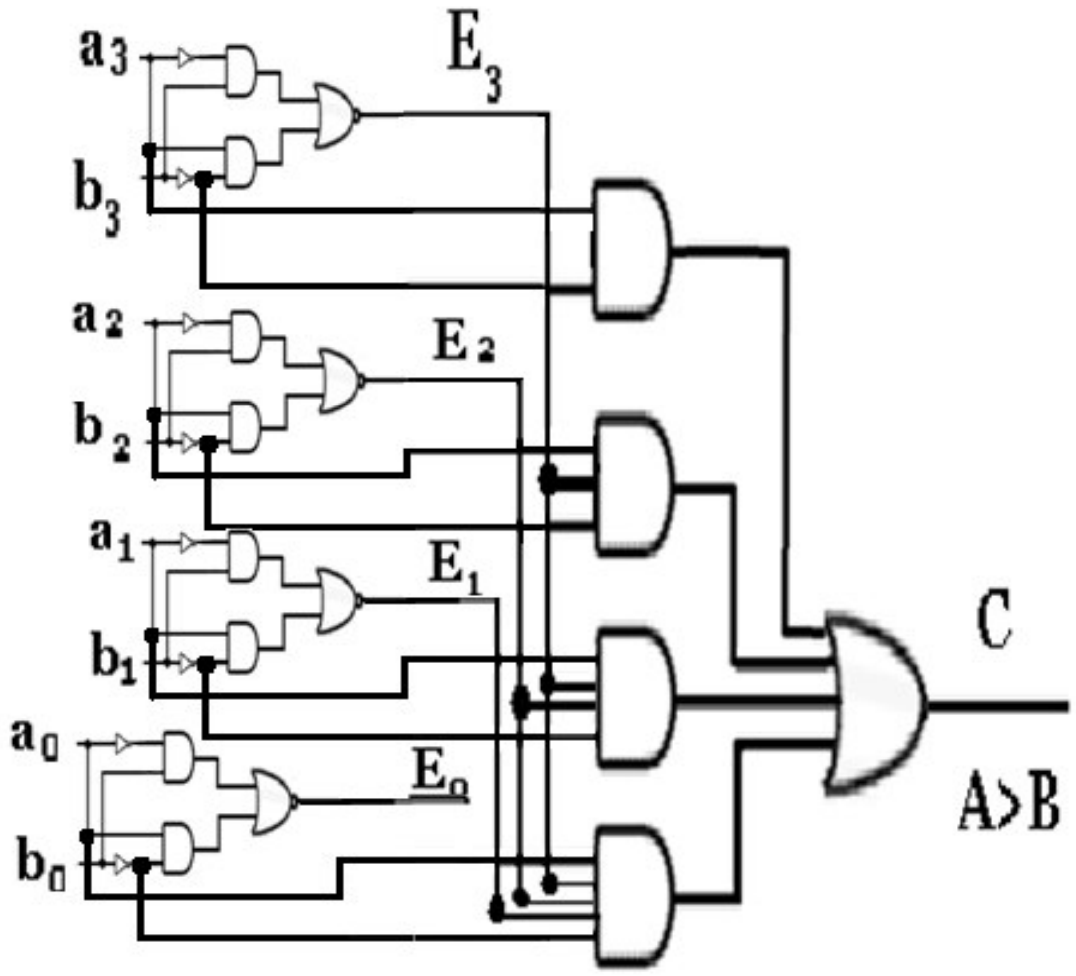
- A= $a_3\ a_2\ a_1\ a_0$
- B= $b_3\ b_2\ b_1\ b_0$
- A = B sólo si $a_3 = b_3 \wedge a_2 = b_2 \wedge a_1 = b_1 \wedge a_0 = b_0$
- O sea $E = E_3 * E_2 * E_1 * E_0$
- A > B si $a_3 > b_3$ **o**
- si $a_3 = b_3 \wedge a_2 > b_2$ **o**
- si $a_3 = b_3 \wedge a_2 = b_2 \wedge a_1 > b_1$ **o**
- si $a_3 = b_3 \wedge a_2 = b_2 \wedge a_1 = b_1 \wedge a_0 > b_0$
- O sea $C = C_3 + E_3*C_2 + E_3*E_2*C_1 + E_3*E_2*E_1*C_0$

A>B LO OBTENGO CON

$C = C_3 + E_3 * C_3 + E_3 * E_2 * C_2 + E_3 * E_2 * C_1 + E_3 * E_2 * E_1 * C_0$

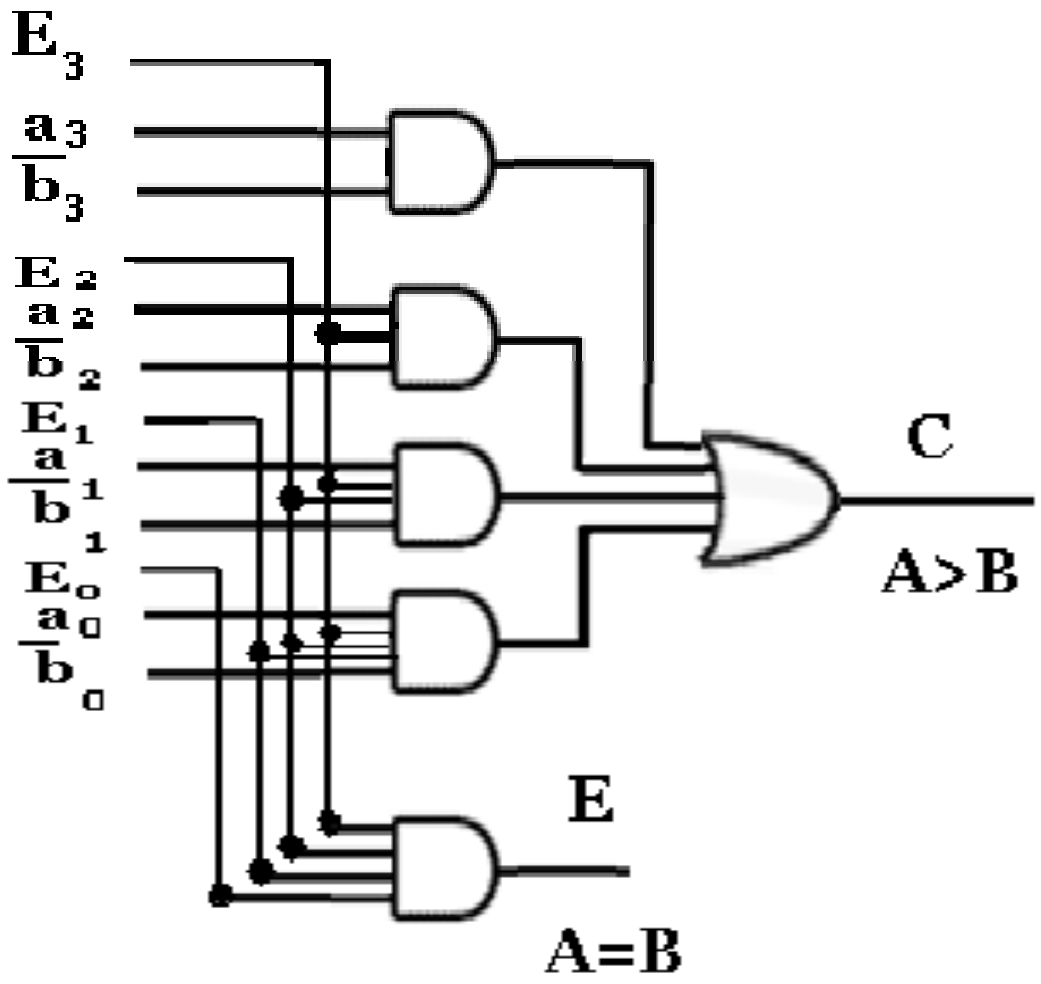


Los Iguales de cada bit los obtenemos usando otras puertas ANDORI

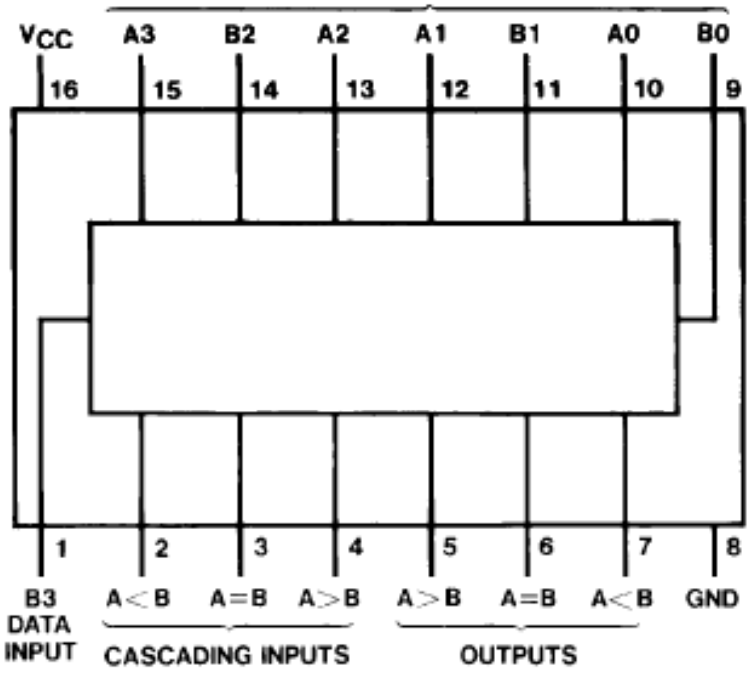


- Para hacer un comparador completo necesito tener la salida de la igualdad.

EL COMPARADOR COMPLETO CON EL IGUAL TENDRÍA LA FORMA



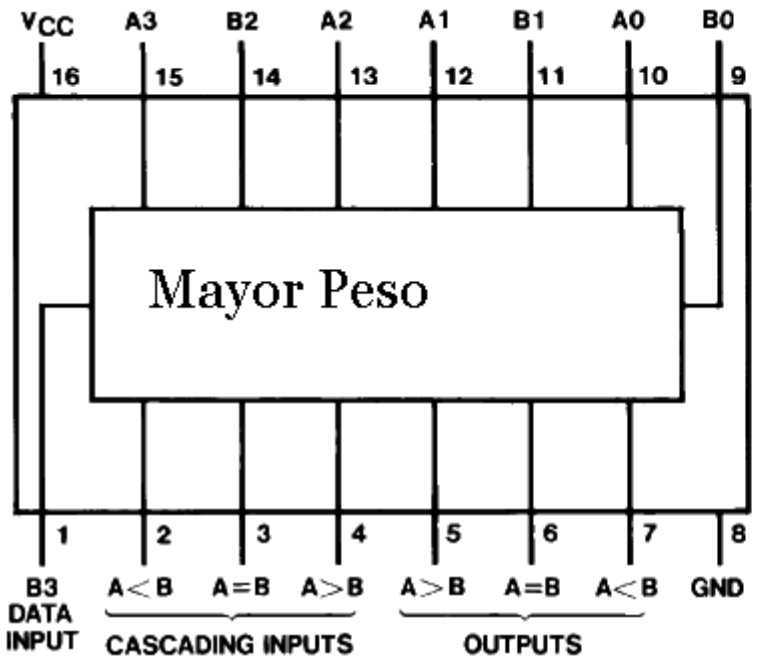
COMPARADOR BINARIO DE CUATRO BIT

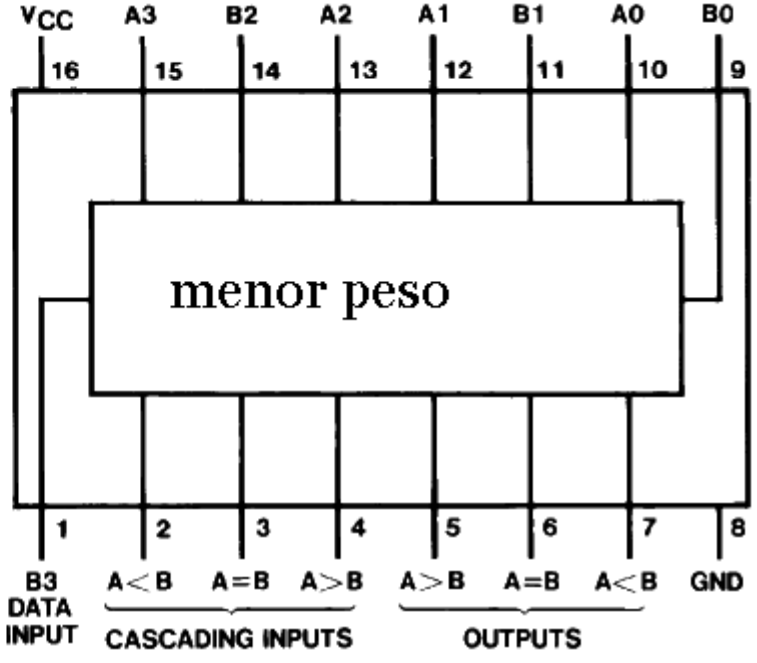


Order Number 54LS85DMQB, 54LS85FMQB, 54LS85LMQB, DM54LS85J, DM54LS85W, DM74LS85M or DM74LS85N
See NS Package Number E20A, J16A, M16A, N16E or W16A

- 54LS85/DM54LS85/DM74LS85
- De National Semiconductor
- En realidad viene preparado para acoplarlo con otros para hacer comparaciones de mas de cuatro bits

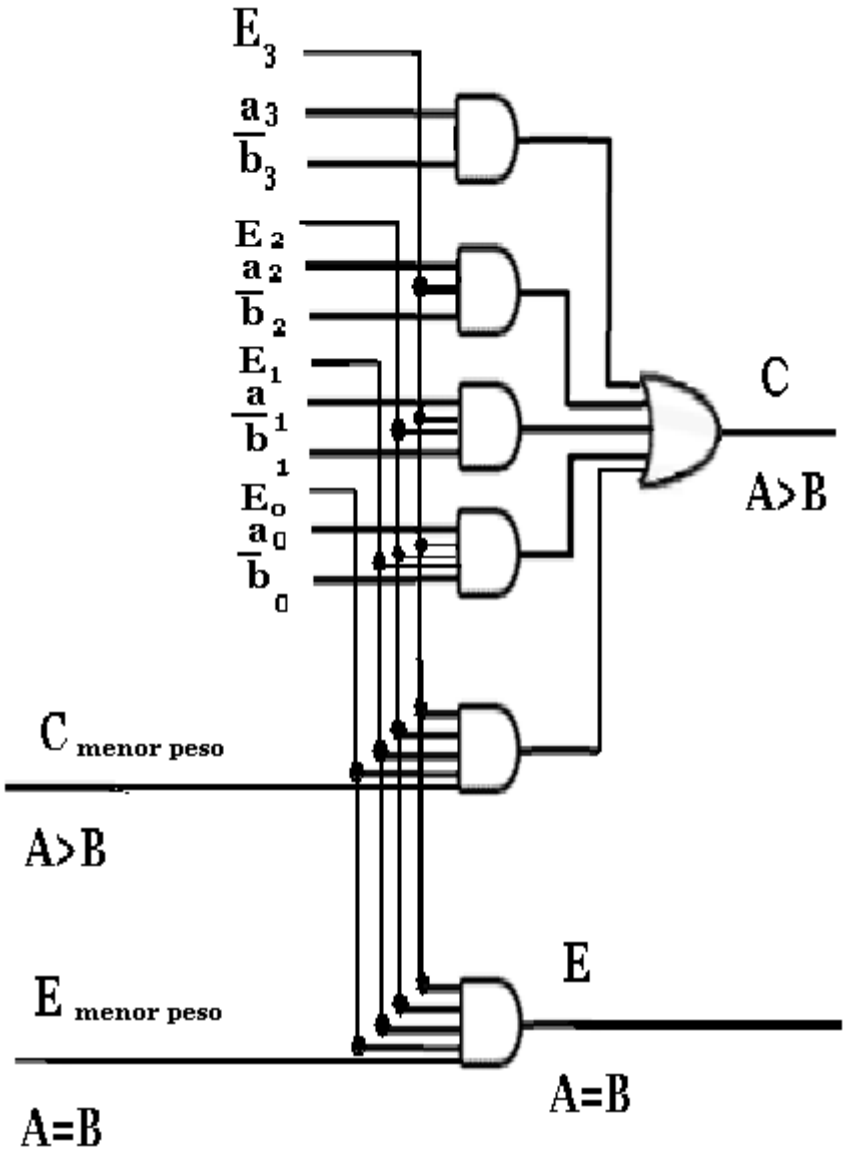
PARA COMPARAR UN NÚMERO DE OCHO BIT





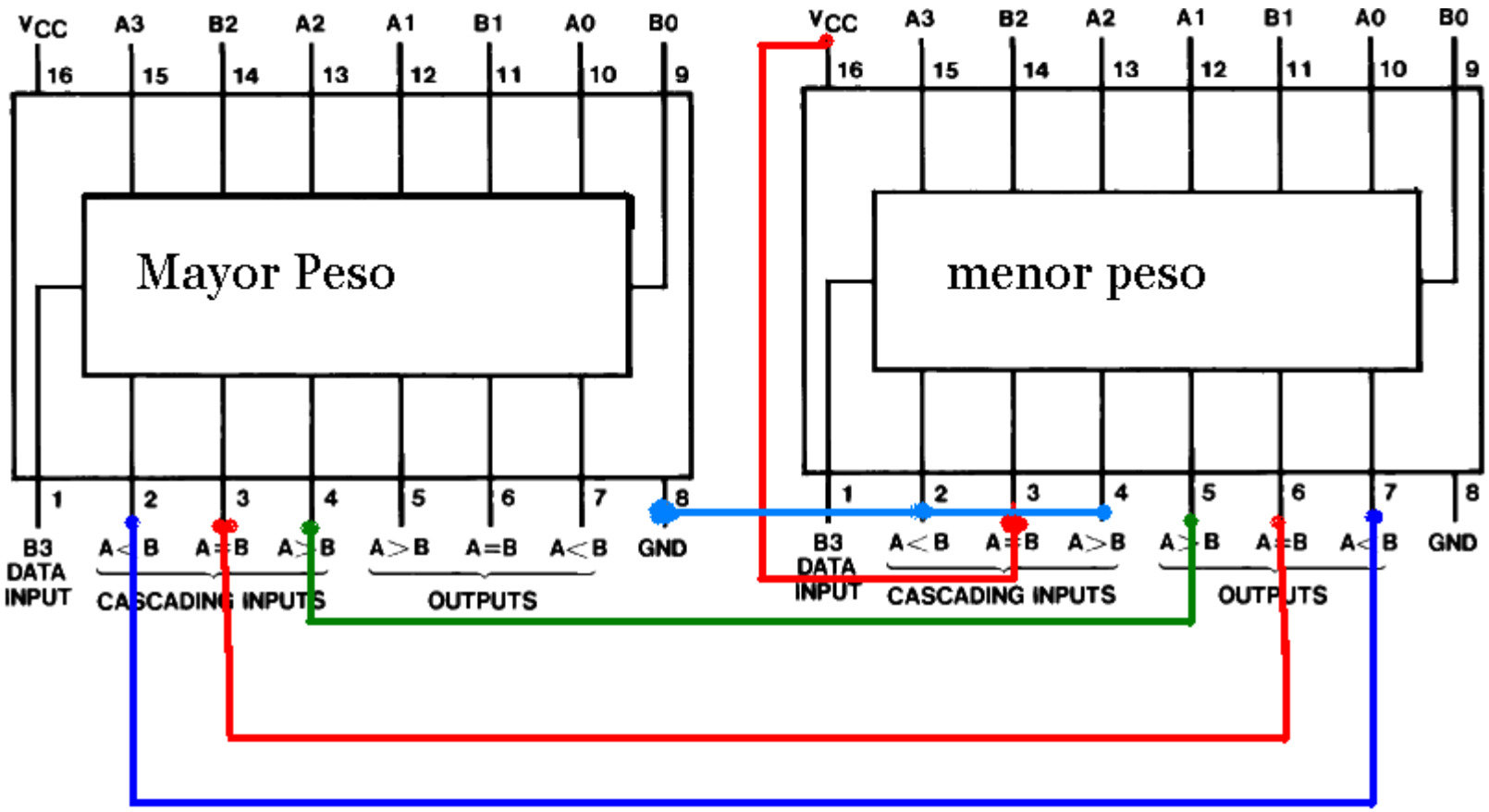
- El criterio es que para aser si un numero es mayor que otro alcanza con saber si el bit de mayor peso lo es

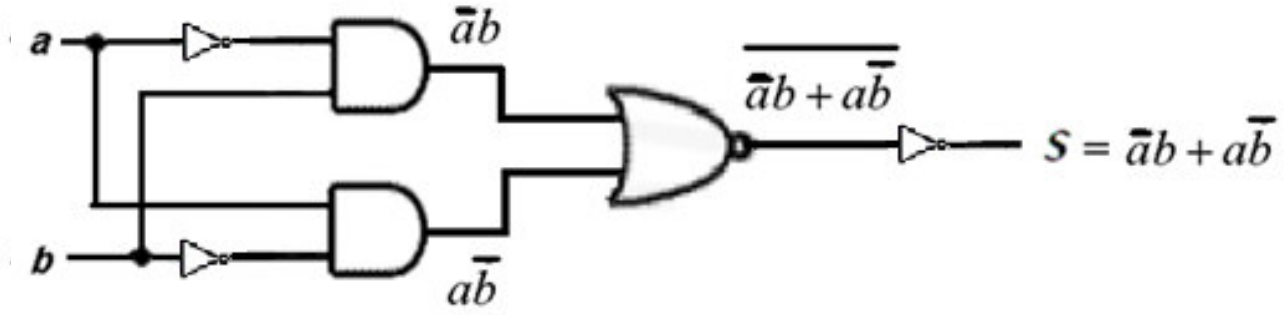
COMPARADOR DE MAS DE CUATRO BITS



- Se pone un chip comparador de los cuatro bits de menor peso, como entrada del comparador de cuatro bits de mayor

QUEDA ENTONCES




a
b
$\bar{a}b$
$a\bar{b}$
$\overline{\bar{a}b + a\bar{b}}$
$S = \bar{a}b + a\bar{b}$