ESCUELA POLITÉCNICA SUPERIOR DE CÓRDOBA
Universidad de Córdoba

SISTEMAS DIGITALES

Tema 4-Análisis y Síntesis de Sistemas Combinacionales. Circuitos MSI

José Luis Ávila Jiménez

UNIVERSIDAD DE CÓRDOBA

Objetivos.

- Definir el concepto de sistema digital combinacional.
- Comprender la metodología clásica de Análisis y Diseño de Sistemas Combinacionales.
- Asimilar las diversas formas de implementación en dos niveles de los Sistemas Combinacionales: puertas básicas y universales.
- Comprender el funcionamiento de algunos bloques funcionales lógicos combinacionales MSI .
- Comprender la implementación de funciones lógicas mediante decodificadores y multiplexores.
- Comprender las operaciones aritméticas básicas con los números enteros binarios sin signo.
- Comprender las características básicas de los sistemas de representación de los números binarios enteros con signo.
- Comprender el funcionamiento y el diseño de los circuitos aritméticos básicos.

TEMA 4: Análisis y síntesis de sistemas combinacionales. Circuitos MSI

1. – Definición de sistema combinacional.

2.– Análisis de circuitos combinacionales.

4.3.– Síntesis de circuitos combinacionales.

   1. – Etapas del diseño.
   2. – Implementación en dos niveles.

      1.– Con puertas básicas AND, OR y NOT.

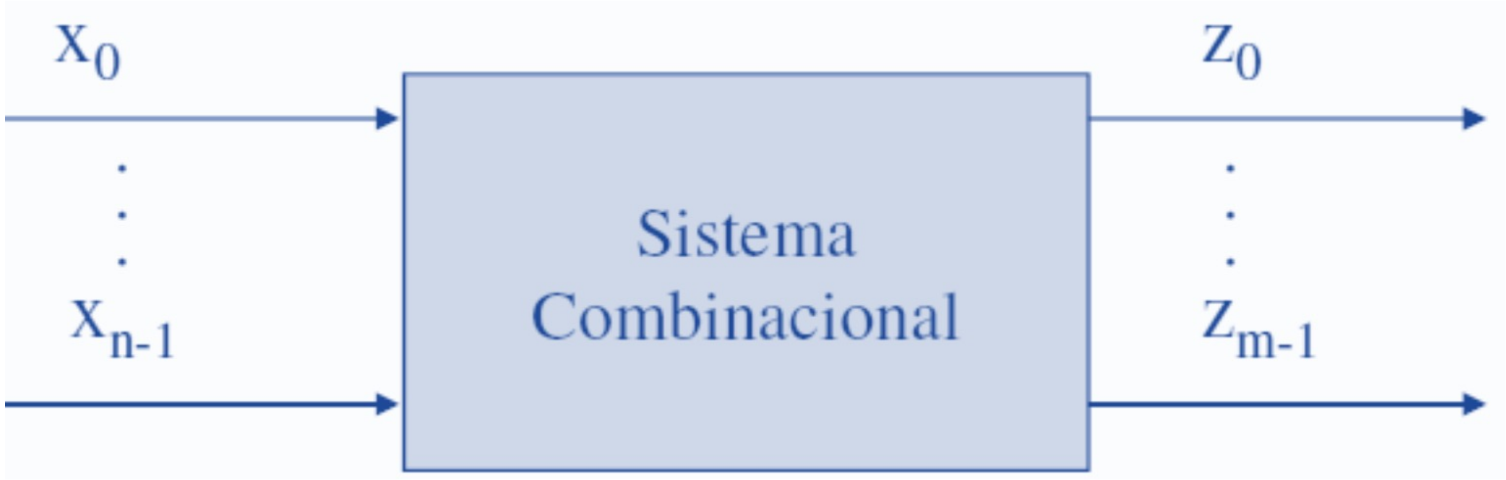
      4.3.2.2.– Solamente con puertas NAND.

      4.3.2.3.– Solamente con puertas NOR.

4. – Decodificadores. Codificadores.
5. – Multiplexores. Demultiplexores.
6. – Aplicaciones de los decodificadores y multiplexores.
7. – Aritmética binaria básica: suma binaria; resta mediante el complemento a 2.
8. – Circuitos aritméticos binarios: Sumador completo; sumador paralelo con acarreo serie; restador.

Un sistema digital **combinacional** es aquel en el que, en cada instante, el estado lógico de sus salidas depende únicamente del valor de sus entradas, salvo retardos en la propagación de las señales.

Un sistema **combinacional** puede ser representado mediante una tabla de verdad, con expresiones algebraicas o representación a nivel de bloque.



Las señales x_i y z_i se representan mediante variables lógicas o de conmutación, y sólo pueden tomar los valores lógicos 0 y 1.

En un sistema combinacional el valor de cada una de las variables de salida se expresa mediante una función lógica de las variables de entrada.

Un circuito combinacional consiste en n variables de entrada, m variables de salida, puertas lógicas e interconexiones.

Un circuito combinacional tiene puertas lógicas sin realimentación ni elementos de almacenamiento.

Análisis de circuitos

Consiste en determinar las funciones lógicas que implementa el circuito a partir del diagrama lógico. También podremos obtener su tabla de verdad.

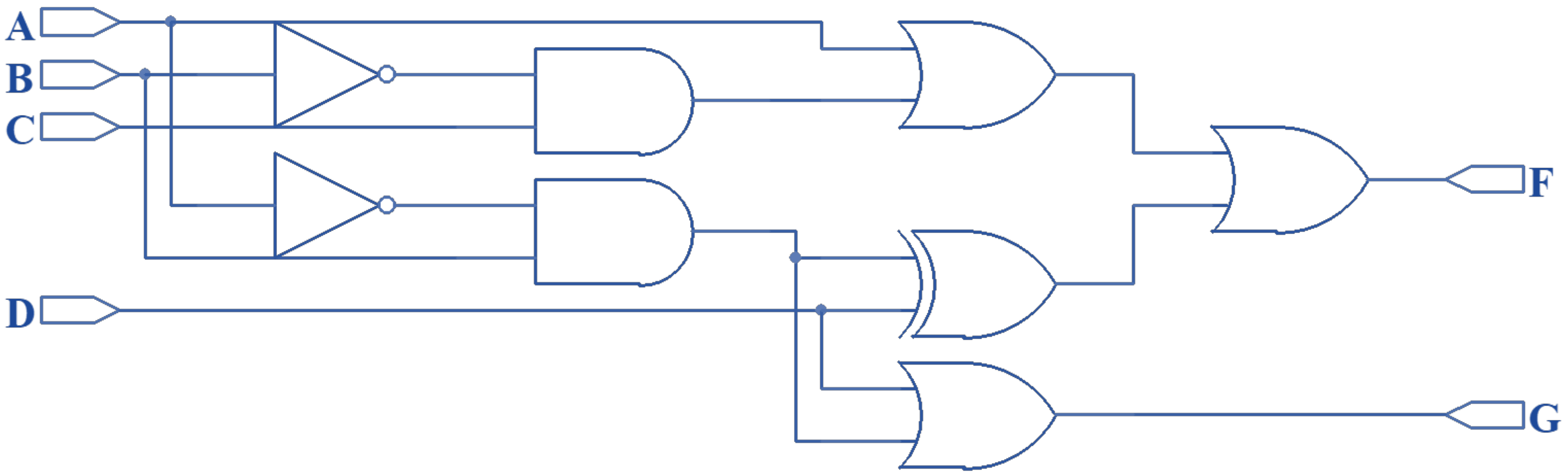
Coincide con el proceso de conversión del formato de diagrama lógico a expresión lógica de cada una de las funciones lógicas de las variables de salida.

En otras palabras, obtener el funcionamiento de un sistema digital combinacional a partir de su estructura.

Vamos a ver varios ejemplos:

4.2.– Análisis de circuitos combinacionales.

Ejemplo 1:



Funciones lógicas de las funciones de salida:

$$F = A + B\cdot C + (A\cdot B) \oplus D = A + \overline{B}\cdot C + A\cdot B\cdot D + A\cdot B\cdot D = A + \overline{B}\cdot C + A\cdot B\cdot D + (A+\overline{B})\cdot D = A + \overline{B}\cdot C +$$
$$\overline{A}\cdot B\cdot \overline{D} + A\cdot D + \overline{B}\cdot D = (A + A\cdot D) + \overline{B}\cdot C + \overline{A}\cdot B\cdot \overline{D} + \overline{B}\cdot D = A + \overline{B}\cdot C + \overline{A}\cdot B\cdot \overline{D} + \overline{B}\cdot D$$

$$G = \overline{A}\cdot B + D$$

4.2.– Análisis de circuitos

Ejemplo 2:

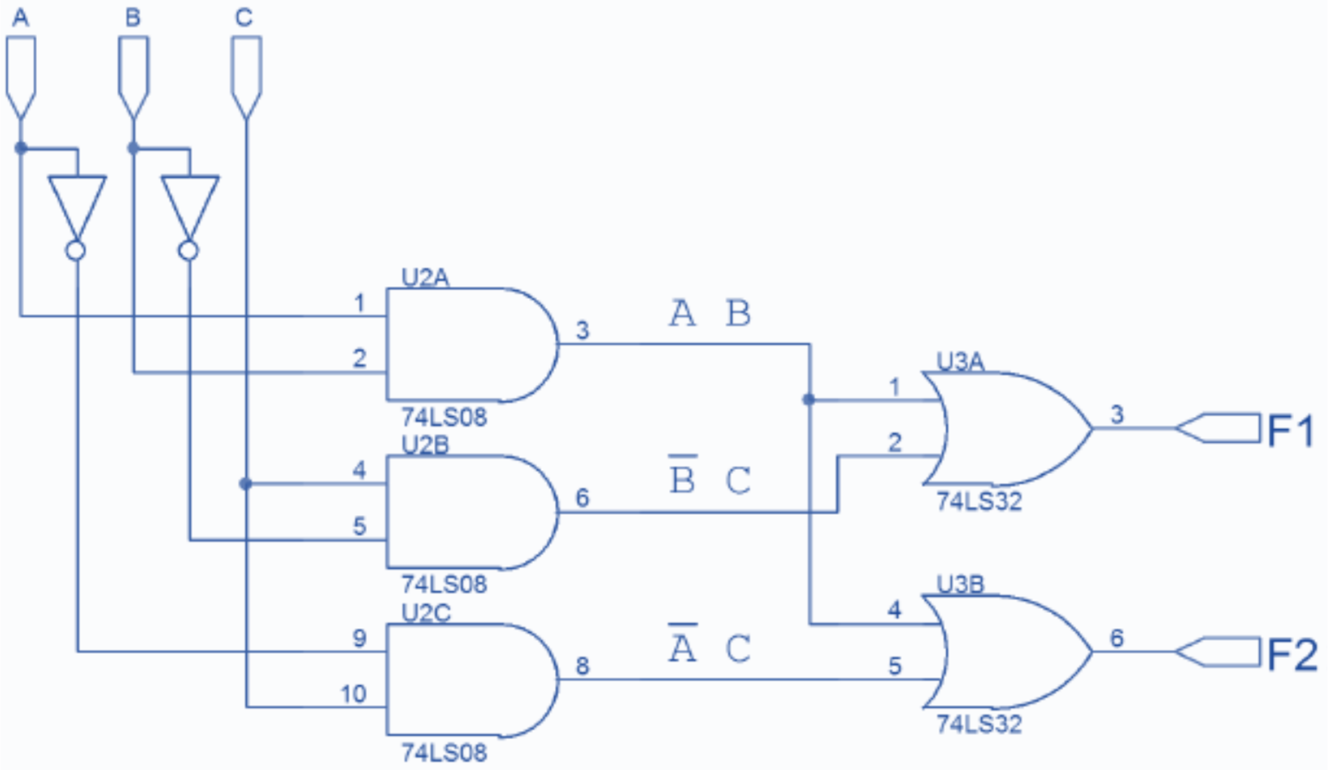


Tabla de verdad:

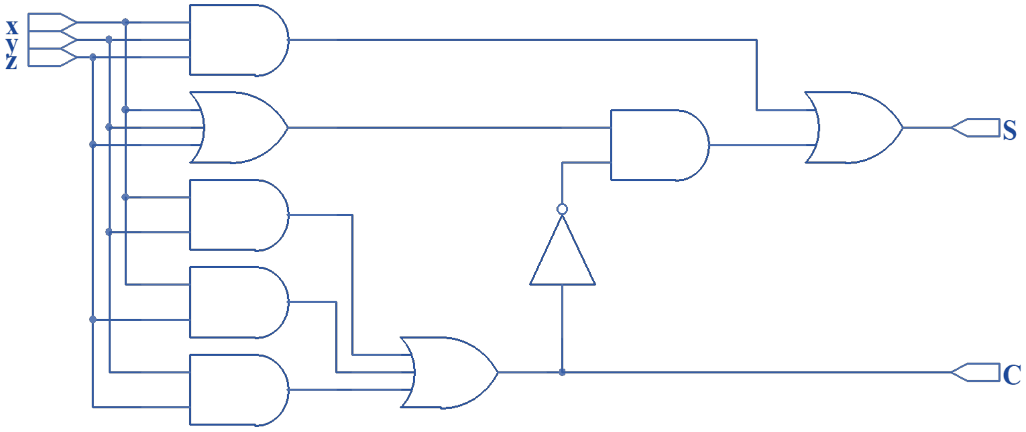
m_i	A B C	F1	F2
0	0 0 0	0	0
1	0 0 1	1	1
2	0 1 0	0	0
3	0 1 1	0	1
4	1 0 0	0	0
5	1 0 1	1	0
6	1 1 0	1	1
7	1 1 1	1	1

Funciones lógicas de las funciones de salida:

$$F1(A,B,C) = AB + \overline{B}C$$
$$F2(A,B,C) = AB + \overline{A}C$$

4.2.– Análisis de circuitos combinacionales.

Ejemplo 3: Podemos obtener la tabla de verdad a partir del circuito.



x	y	z	x·y	x·z	y·z	x·y·z	x+y+z	C=xy+xz+yz	$\overline{C}$	I=$\overline{C}$ (x+y+z)	S=xyz+I
0	0	0	0	0	0	0	0	0	1	0	0
0	0	1	0	0	0	0	1	0	1	1	1
0	1	0	0	0	0	0	1	0	1	1	1
0	1	1	0	0	1	0	1	1	0	0	0
1	0	0	0	0	0	0	1	0	1	1	1
1	0	1	0	1	0	0	1	1	0	0	0
1	1	0	1	0	0	0	1	1	0	0	0
1	1	1	1	1	1	1	1	1	0	0	1

x	y	z	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

Análisis de circuitos

Ejemplo 3: En la tabla anterior se puede comprobar que se trata de un sumador binario de 3 bits.

Otra forma de representar el funcionamiento de un circuito es a través de la **simulación lógica**, que es un método rápido y exacto de análisis de circuitos combinacionales, si el resultado deseado son formas de onda lógicas.

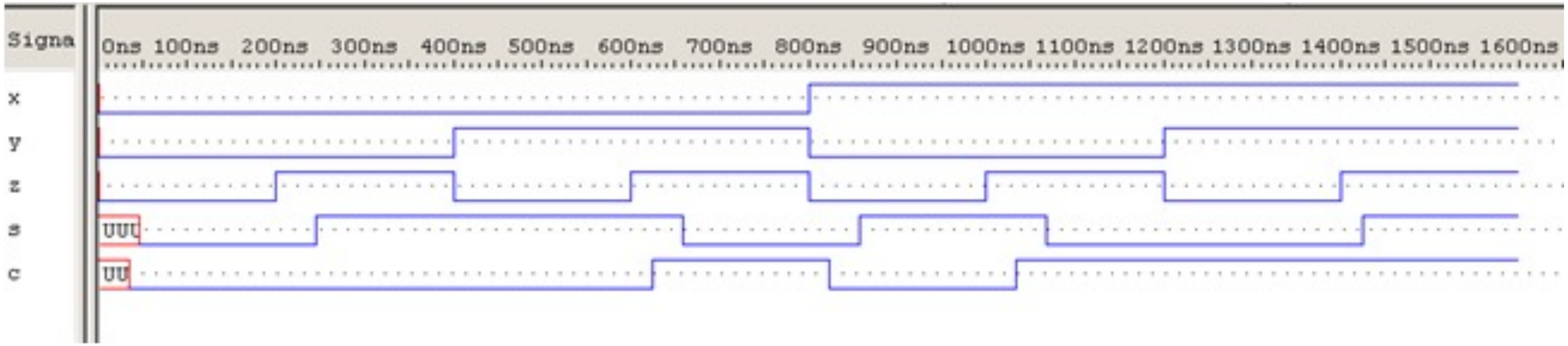
Existen paquetes de software (OrCAD) que ofrecen unas bibliotecas de donde pueden obtenerse símbolos para las puertas, entradas, salidas, etc. Estos programas nos permiten representar el diagrama lógico mediante un esquema lógico. Posteriormente se usa una herramienta de captura de esquemáticos para representar el circuito.

Análisis de circuitos.

Para efectuar la simulación, tenemos que introducir las entradas del circuito. En el ejemplo anterior, serían las ocho combinaciones posibles de las entradas.

Las entradas se especifican como el contenido de un archivo que puede leer el simulador, o bien se introducen interactivamente en el simulador (PSPICE).

Con los estímulos apropiados la salida en el programa OrCAD sería la siguiente:



TEMA 4: Análisis y síntesis de sistemas combinacionales. Circuitos MSI

4.1.– Definición de sistema combinacional.

4.2.– Análisis de circuitos combinacionales.

3. – Síntesis de circuitos combinacionales.

1. – Etapas del diseño.
2. – Implementación en dos niveles.
    1.– Con puertas básicas AND, OR y NOT.
    4.3.2.2.– Solamente con puertas NAND.
    4.3.2.3.– Solamente con puertas NOR.

4. – Decodificadores. Codificadores.
5. – Multiplexores. Demultiplexores.
6. – Aplicaciones de los decodificadores y multiplexores.
7. – Aritmética binaria básica: suma binaria; resta mediante el complemento a 2.
8. – Circuitos aritméticos binarios: Sumador completo; sumador paralelo con acarreo serie; restador.

4.3.– Síntesis de circuitos combinacionales.

El diseño o síntesis de un circuito combinacional consiste en, a partir de la especificación de un problema, obtener el diagrama lógico que lo cumpla.

4.3.1.- Etapas del diseño:

Enunciado del problema.

Determinación de la cantidad de variables de entrada y variables de salida necesarias.

Se le asignan nombre simbólicos a estas variables.

Obtener la tabla de verdad que define la relación entre las entradas y las salidas.

Obtención de las funciones booleanas de cada salida como función de las variables de entrada, y se simplifican.

Se representa el diagrama lógico o esquema eléctrico.

Comprobación de la corrección del diseño (simulación y montaje).

4.3.– Síntesis de circuitos combinacionales.

4.3.2.- Implementación en dos niveles: niveles de implementación:

El objetivo principal de la minimización de funciones es sintetizar la función de la forma más económica posible.

Consideraciones a tener en cuenta cuando se trata de evaluar las prestaciones de un circuito digital: la **velocidad de respuesta**, y el **coste**.

La **velocidad de respuesta** disminuye al aumentar el retardo de propagación de la señales de salida. El retardo de propagación aumenta conforme crece el nº de niveles de puertas (nº de etapas por las que tiene que pasar una señal desde la entrada hasta la salida).

El **coste** viene determinado por el nº de puertas lógicas y el nº de entradas de las puertas lógicas : "Dados dos circuitos de conmutación, se considerará de menor coste el que tenga menos puertas lógicas, y a igualdad de puertas, el que necesite menos conexiones, es decir puertas con menor número de entradas".

4.3.– Síntesis de circuitos combinacionales.

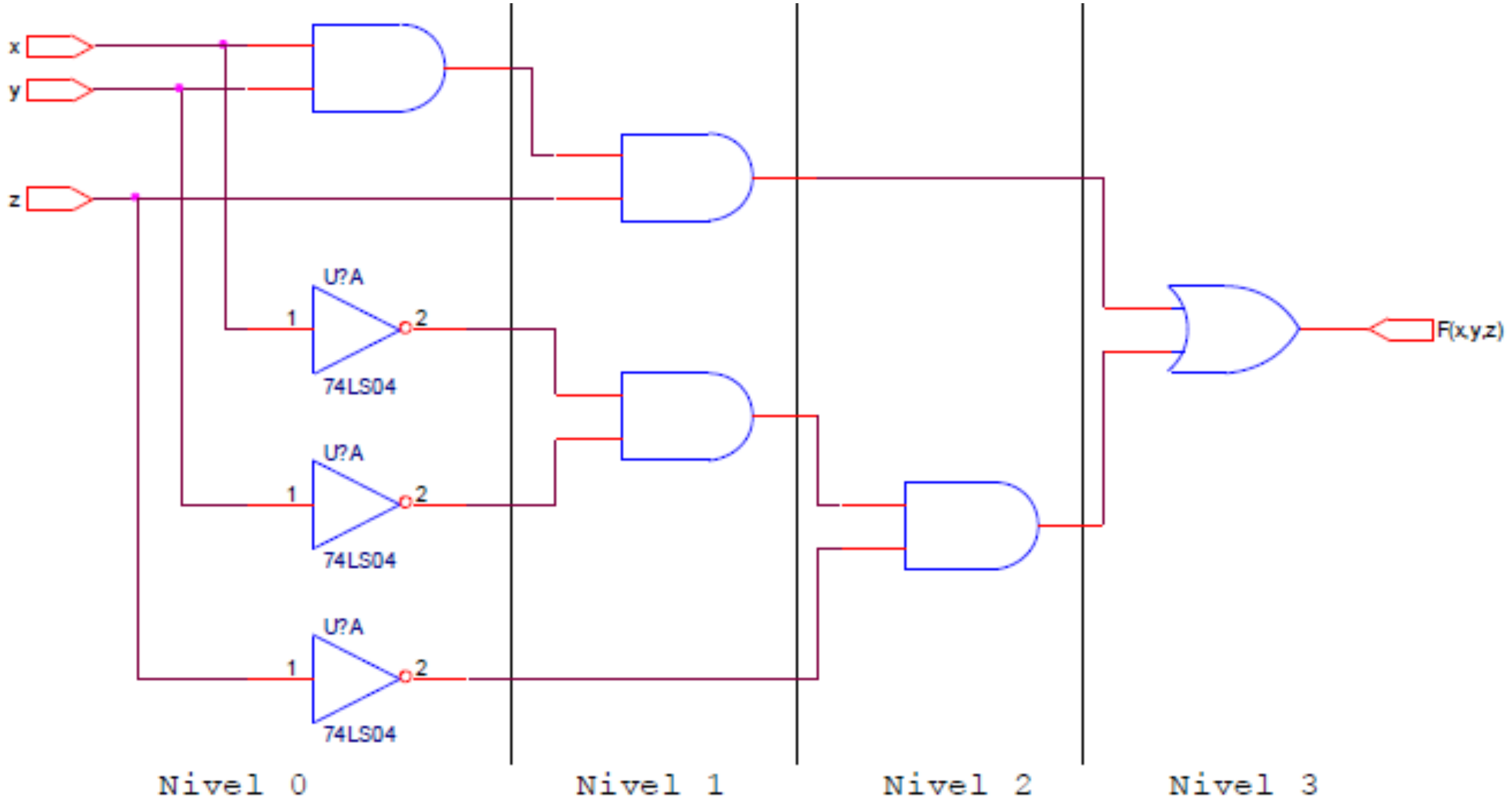
4.3.2.- Implementación en dos niveles: niveles de implementación:

Las puertas NOT que complementan las variables de entrada no se suelen considerar como un nivel de implementación.

Si se usan circuitos integrados SSI para implementar las puertas lógicas, habría que considerarlas como un nivel adicional en el cálculo de los retardos.

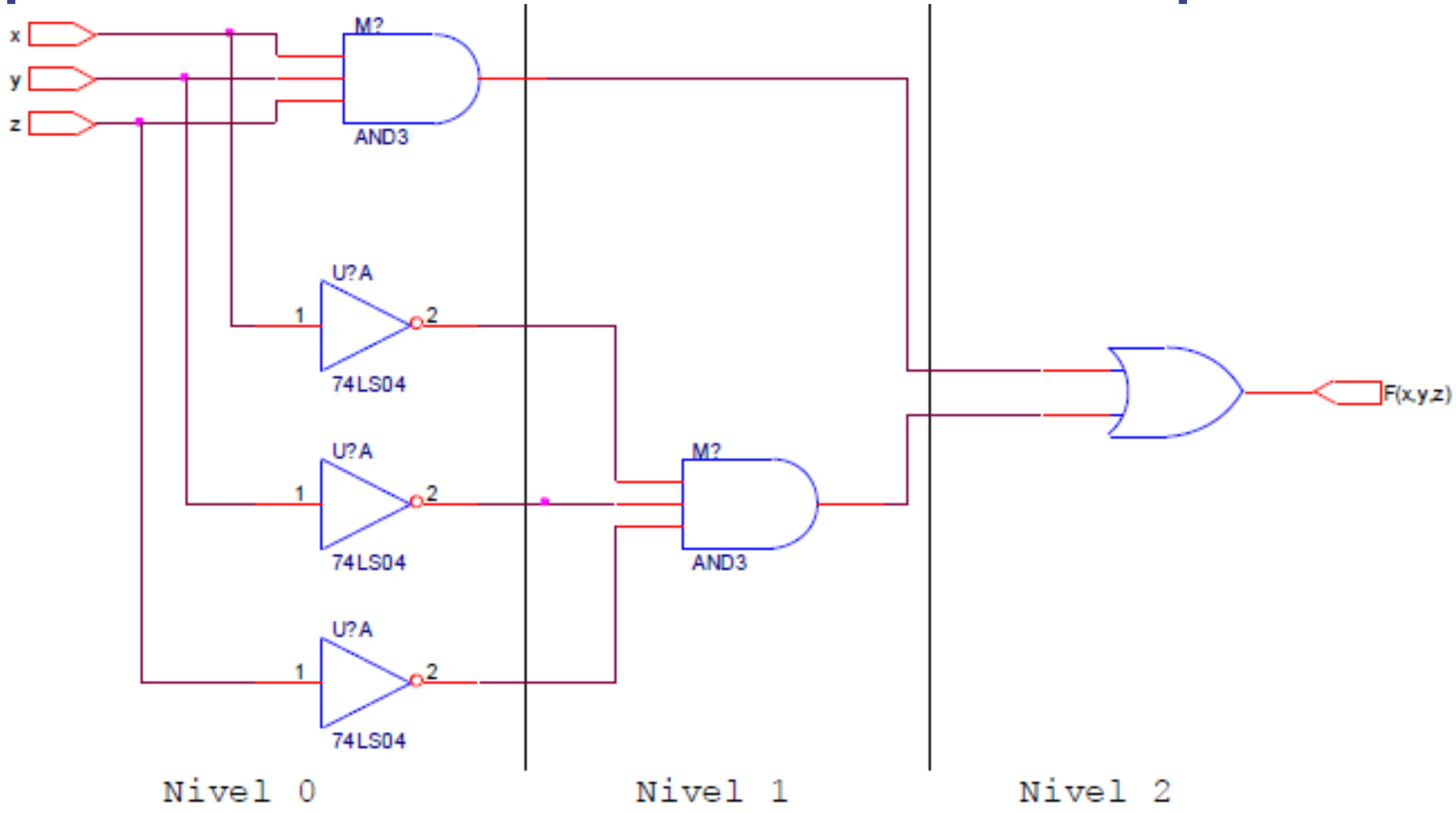
4.3.– Síntesis de circuitos combinacionales.

4.3.2.- Implementación en dos niveles: niveles de implementación:



4.3.– Síntesis de circuitos combinacionales.

4.3.2.- Implementación en dos niveles: niveles de implementación:



4.3.– Síntesis de circuitos combinacionales.

4.3.2.- Implementación en dos niveles: niveles de implementación:

Los criterios de minimización suelen ser los siguientes:

- Mínimo número de puertas.
- Mínimo número de entradas a las puertas.
- Mínimo tiempo de propagación.
- Limitación de *fan–out* de cada puerta. El *fan–out* (o factor de carga) de u puerta es el número de puertas que se pueden conectar a la salida de ot puerta igual.

4.3.– Síntesis de circuitos combinacionales.

4.3.2.- Implementación en dos niveles: niveles de implementación:

Las ventajas de una implementación en dos niveles como suma de productos lógicos o como producto de sumas son las siguientes:

- El retardo de propagación es mínimo.
- Fácil implementación mediante circuitos lógicos.
- Las funciones lógicas se pueden implementar fácilmente mediante un Dispositivo Lógico Programable (PLD).

4.3.– Síntesis de circuitos combinacionales.

4.3.2.- Implementación en dos niveles: ejemplo de diseño:

Enunciado del problema: Diseño de un circuito que controle el resultado de una votación donde votan tres personas.

- Votan tres personas.
- Hay dos opciones de voto: SI y NO. No existe abstención.
- Generar las salidas necesarias para expresar el resultado por mayoría simple.

Variables de entrada y salida:

- Variables de entrada: a, b, c.
- Variable de salida: f.

Síntesis de circuitos combinacionales.

4.3.2.- Implementación en dos niveles: ejemplo de diseño:

Tabla de verdad: Los valores de la variable de salida se determinan según el problema planteado

a	b	c	f
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

Síntesis de circuitos combinacionales.

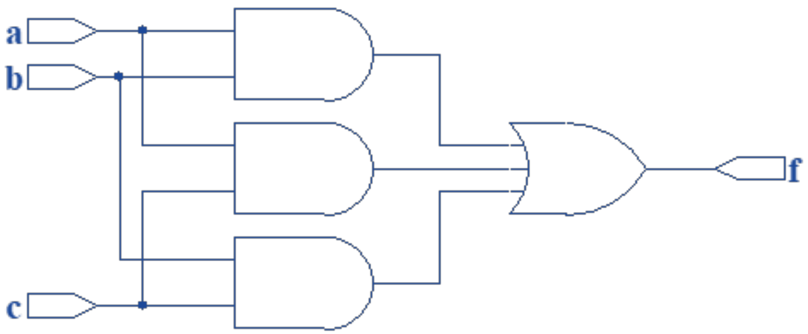
4.3.2.- Implementación en dos niveles: ejemplo de diseño:

Se simplifica la función como suma de productos y como producto de sum

a	b	c	f
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

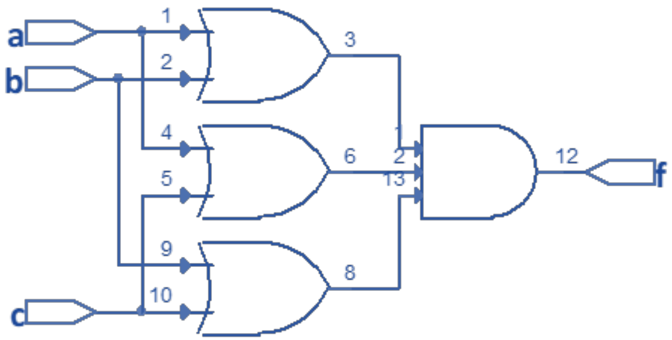
a\bc	00	01	11	10
0			1	
1		1	1	1

$f = a \cdot b + a \cdot c + b \cdot c$



a\bc	00	01	11	10
0	0	0		0
1	0			

$f = (a+b) \cdot (a+c) \cdot (b+c)$

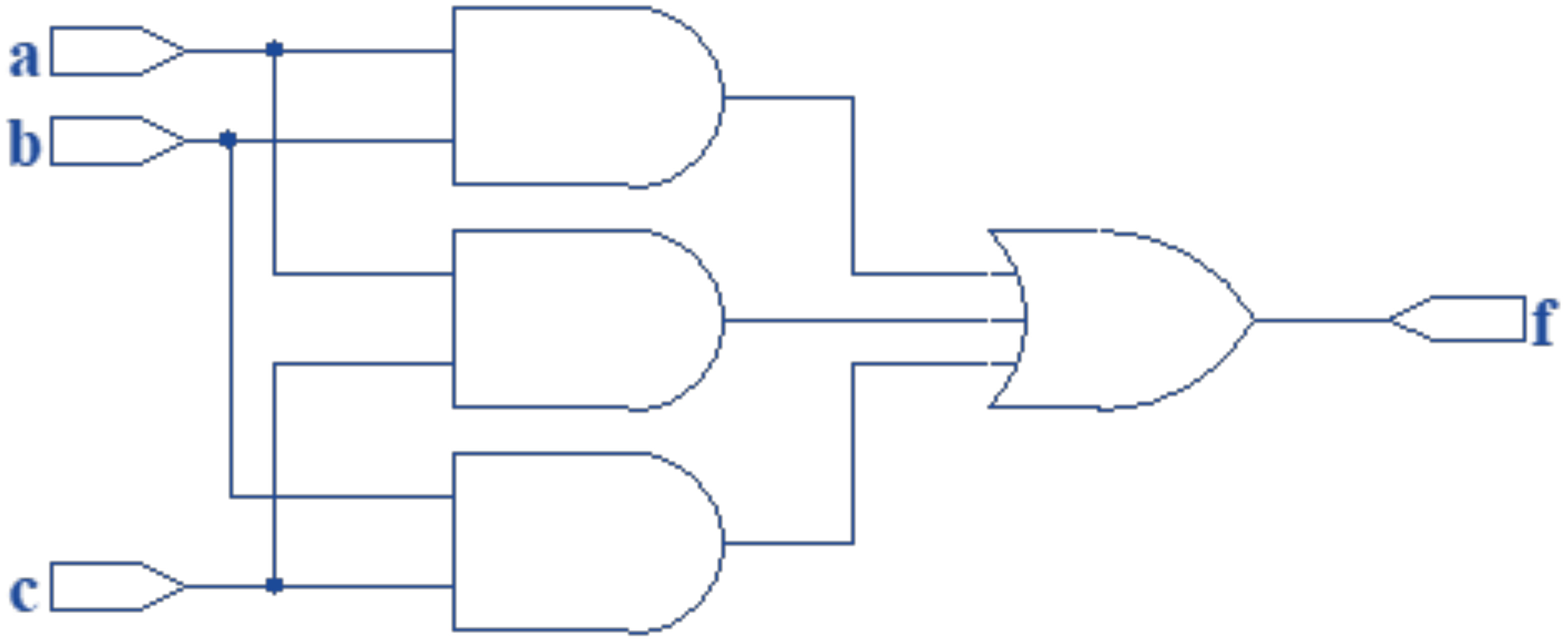


Síntesis de circuitos combinacionales.

4.3.2.- Implementación en dos niveles con puertas lógicas básicas AND, OR y NOT:

Implementación mediante puertas lógicas básicas partiendo de la suma de productos.

$f = a \cdot b + a \cdot c + b \cdot c$

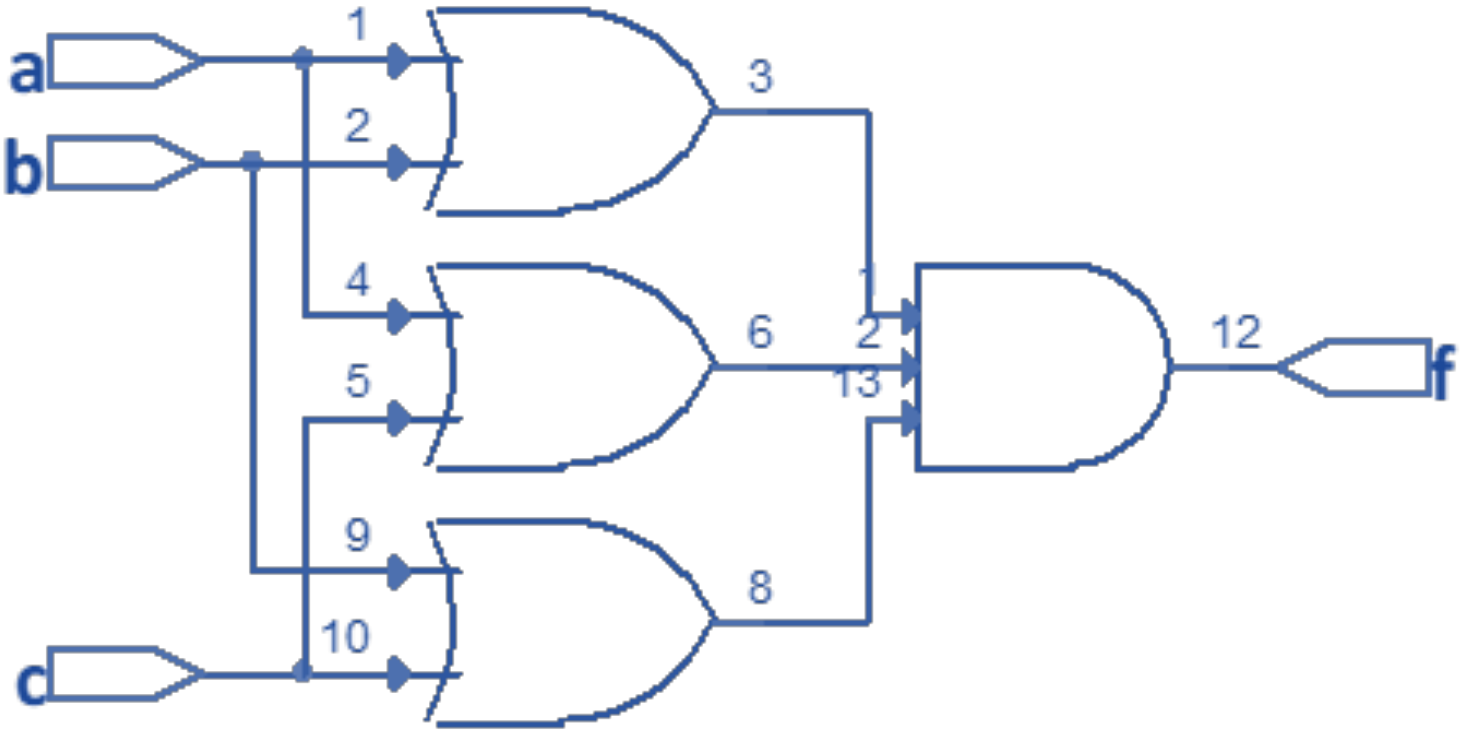


Síntesis de circuitos combinacionales.

4.3.2.- Implementación en dos niveles con puertas lógicas básicas AND, OR y NOT:

Implementación mediante puertas lógicas básicas partiendo del producto de sumas.

$f = (a+b)\cdot(a+c)\cdot(b+c)$



Síntesis de circuitos combinacionales.

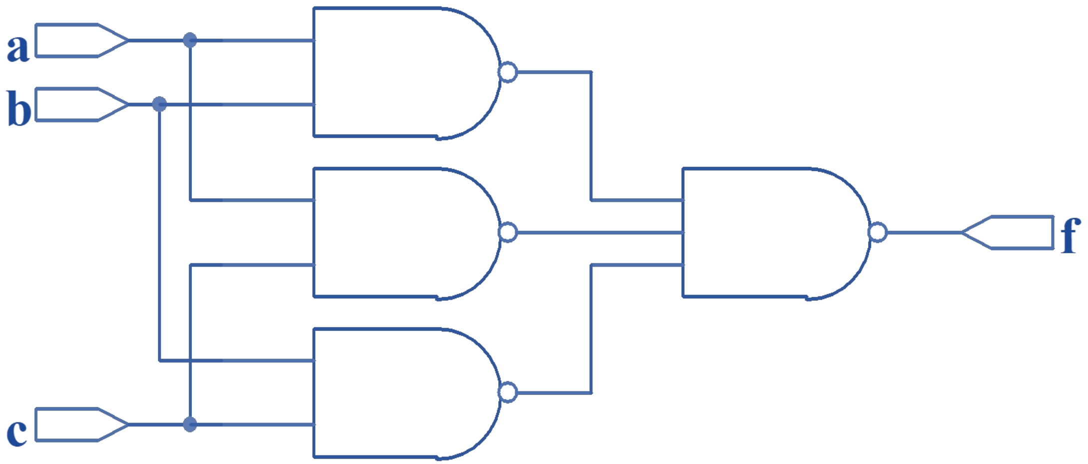
4.3.2.- Implementación en dos niveles: Solamente con puertas NAND:

Partimos de la expresión simplificada de suma de productos. $f = a \cdot b + a \cdot c + b \cdot c$

Se aplican dos complementos a la función global, en virtud del teorema de involución. $f = \overline{\overline{a \cdot b + a \cdot c + b \cdot c}}$

Aplicamos el teorema de De Morgan a un complemento. $f = \overline{\overline{(a \cdot b)} \cdot \overline{(a \cdot c)} \cdot \overline{(b \cdot c)}}$

Representamos el circuito:



4.3.– Síntesis de circuitos combinacionales.

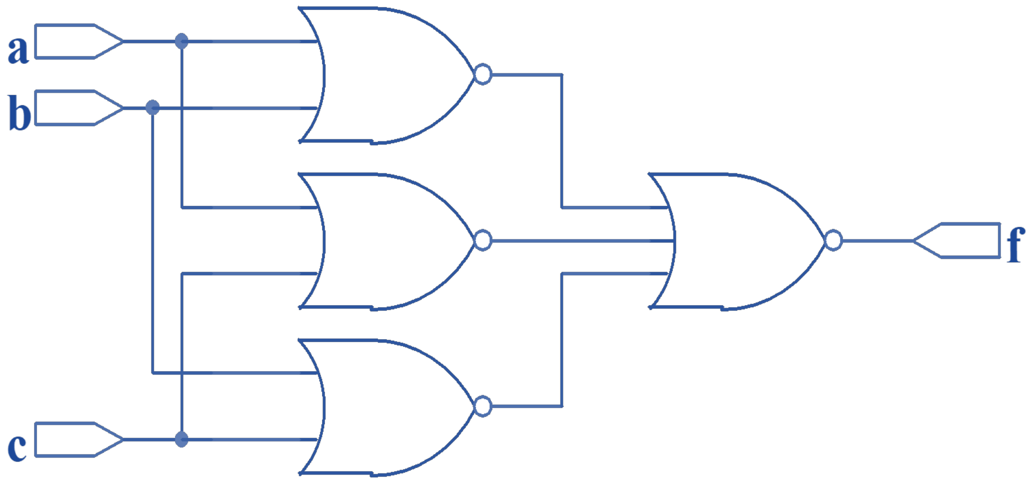
4.3.2.- Implementación en dos niveles: Solamente con puertas NOR:

Partimos de la expresión como producto de sumas. $f=(a+b)\cdot(a+c)\cdot(b+c)$

Se aplican dos complementos a la función global, en virtud del teorema de involución. $f=\overline{\overline{(a+b)\cdot(a+c)\cdot(b+c)}}$

Aplicamos de De Morgan a un complemento. $f=\overline{\overline{(a+b)}+\overline{(a+c)}+\overline{(b+c)}}$

Representamos el circuito:



Decodificadores.

Un **decodificador** es un circuito lógico combinacional cuya función básica es detectar la presencia de una determinada combinación de bits (código) en sus entradas, y señalar la presencia de este código activando solamente la salida correspondiente.

Un decodificador tiene n señales de entrada y M señales de salida, siendo $M \leq 2^n$, y una señal de habilitación del chip: *chip enable* (E).

No tiene porqué ser un de codificador de n a 2^n (binario), sino que M puede ser menor o igual que 2^n. Es el caso de un decodificador BCD a siete segmentos.

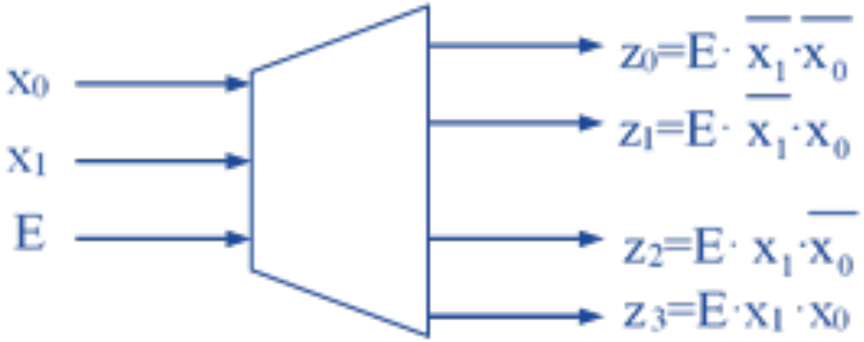
Decodificadores.

Un **decodificador binario de 2 a 4** con salidas activas a nivel alto, que tiene n = 2 entradas, una entrada de habilitación del chip (E, activa a nivel alto), y $2^2 = 4$ señales de salida, de forma que se habilita (a nivel alto) solamente la salida cuyo subíndice corresponde al equivalente decimal de la combinación binaria de entrada.

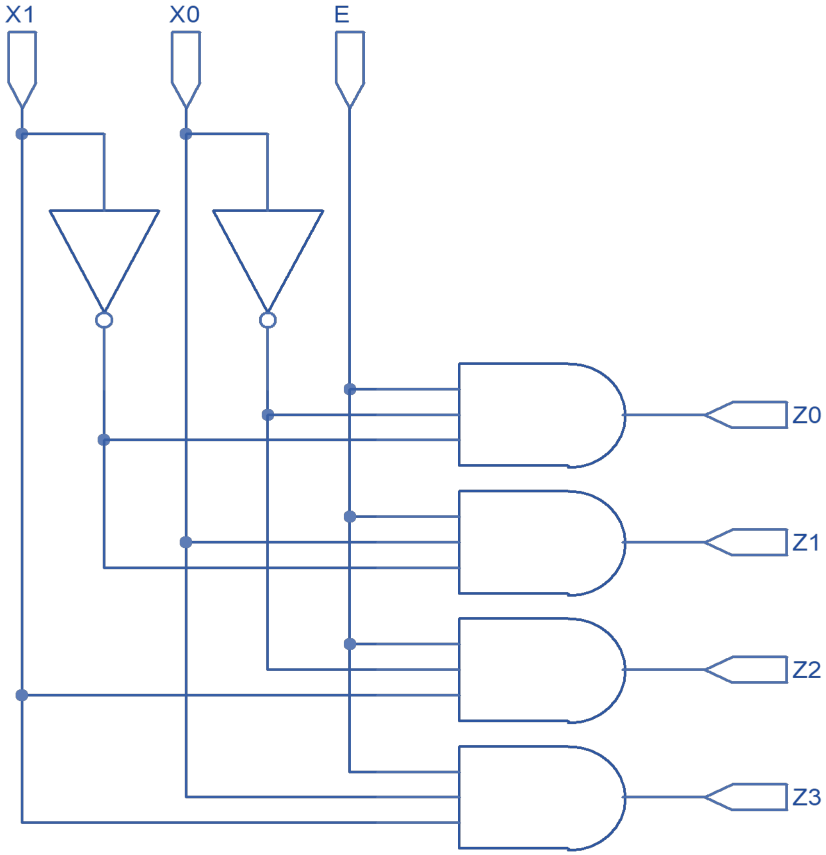
Mostramos a continuación, el símbolo lógico, expresiones algebráicas de las salidas, tabla de verdad y circuito equivalente construido con puertas lógicas.

Decodificadores.

Decodificador 2:4 con salidas activas a nivel alto.

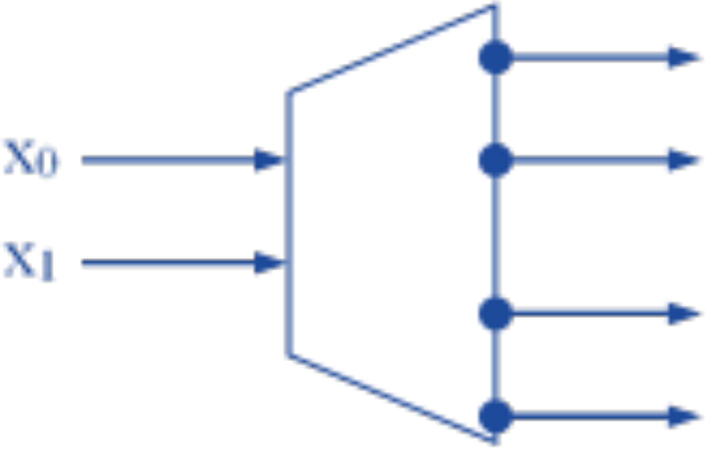


E	x_1	x_0	z_0	z_1	z_2	z_3
0	–	–	0	0	0	0
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1



Decodificadores.

74LS139: Decodificador 2:4 con salidas activas a nivel bajo.



$Z_0 = \overline{\overline{x_1} \cdot \overline{x_0}} = x_1 + x_0$

$Z_1 = \overline{\overline{x_1} \cdot x_0} = x_1 + \overline{x_0}$

$Z_2 = \overline{x_1 \cdot \overline{x_0}} = \overline{x_1} + x_0$

$Z_3 = \overline{x_1 \cdot x_0} = \overline{x_1} + \overline{x_0}$

74LS138: Decodificador 3:8 con salidas activas a nivel bajo.

74LS154 de 4 a 16 con 24 pines: 2 (Vcc y GND), 4 + 16 + 2 entradas de selección G1 y G2 que realizan la función de habilitación del chip (*chip enable*).

Codificadores.

En general se llama codificación al proceso de convertir símbolos comunes o números a un formato codificado.

Un **codificador** es un circuito lógico combinacional que, esencialmente, realiza la función inversa del decodificador: un codificador permite que se introduzca en una de sus M, ($2^n \leq$ M) entradas, un nivel activo que representa un dígito (por ejemplo decimal u octal), y lo convierte en una salida codificada (por ejemplo BCD o binario).

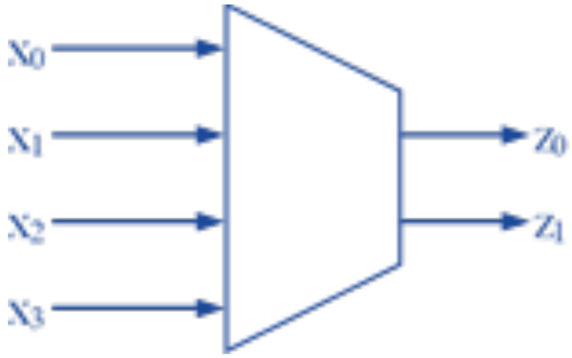
Dependiendo del número de entradas que se puedan activar al mismo tiempo, pueden ser codificadores sin prioridad o codificadores con prioridad.

Codificadores.

En un **codificador sin prioridad** en cada instante tiene que haber siempre una sola entrada activa.

Este circuito genera a su salida la representación en binario del subíndice de la señal de entrada activa.

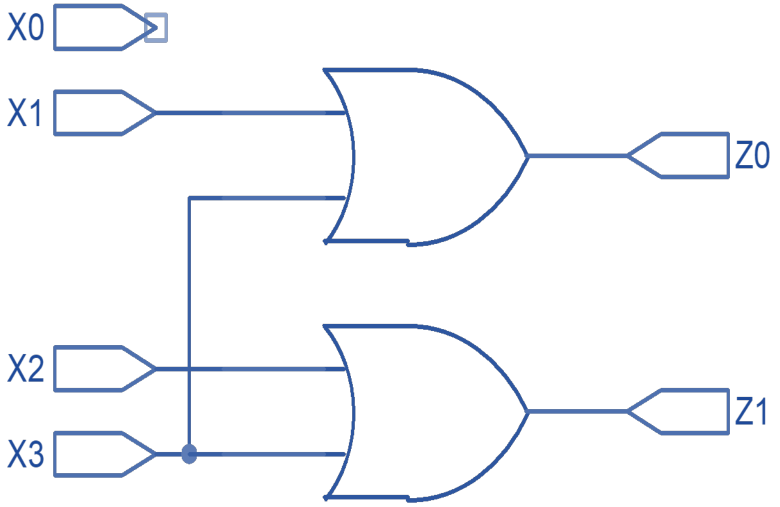
Ejemplo: Codificador binario sin prioridad de 4 a 2.



x0	x1	x2	x3	z1	z0
1	0	0	0	0	0
0	1	0	0	0	1
0	0	1	0	1	0
0	0	0	1	1	1

$$z0 = x1 + x3$$

$$z1 = x2 + x3$$



Codificadores

En un **codificador con prioridad** se ordenan las entradas según un criterio de prioridad y no hay restricción en cuanto al número de entradas que puedan estar activas simultáneamente.

Este circuito genera a su salida la representación en binario del subíndice de la señal de entrada activa con mayor prioridad, que suele ser la más significativa.

Ejemplo: Codificador binario con prioridad de 4 a 2, y que le asignamos la prioridad 3, 2, 1, 0.

Codificador binario prioridad 4 a 2

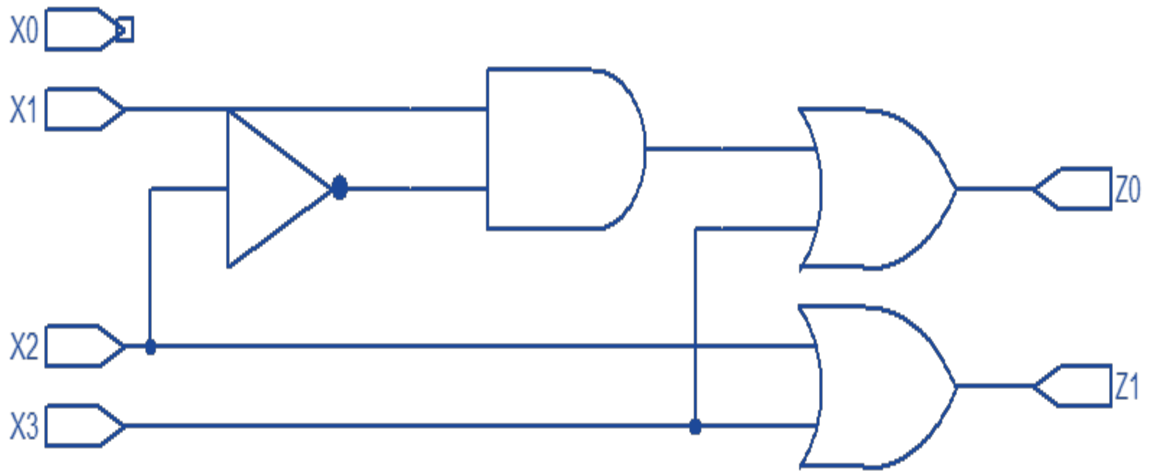
x3	x2	x1	x0	z1	z0
0	0	0	–	0	0
0	0	1	–	0	1
0	1	–	–	1	0
1	–	–	–	1	1

x3	x2	x1	x0	z1	z0
0	0	0	0	0	0
0	0	0	1	0	0
0	0	1	0	0	1
0	0	1	1	0	1
0	1	0	0	1	0
0	1	0	1	1	0
0	1	1	0	1	0
0	1	1	1	1	0
1	0	0	0	1	1
1	0	0	1	1	1
1	0	1	0	1	1
1	0	1	1	1	1
1	1	0	0	1	1
1	1	0	1	1	1
1	1	1	0	1	1
1	1	1	1	1	1

x3x2\x1x0	00	01	11	10
00				
01	1	1	1	1
11	1	1	1	1
10	1	1	1	1

$z_1 = x_2 + x_3$

x3x2\x1x0	00	01	11	10
00			1	1
01				
11	1	1	1	1
10	1	1	1	1

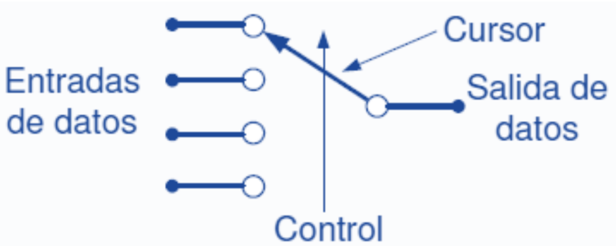


$z_0 = x_3 + x_1 \cdot \overline{x_2}$

Multiplexores. Demultiplexores.

Multiplexores.

Un **multiplexor** realiza la función de un conmutador de múltiples posiciones. Tiene unas entradas de control que indican la entrada que se conecta a la salida.

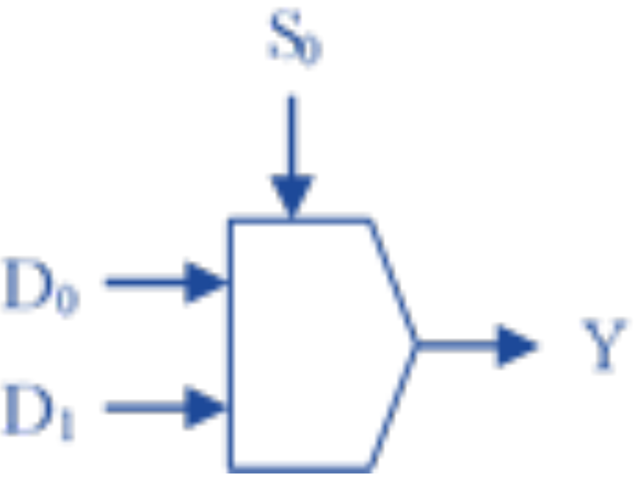


Un multiplexor **MUX**, también llamado selector de datos, es un componente combinacional para la conducción de la información que dispone de **p** entradas de control, $\mathbf{2^p}$ entradas de datos, y solamente **una** salida de datos.

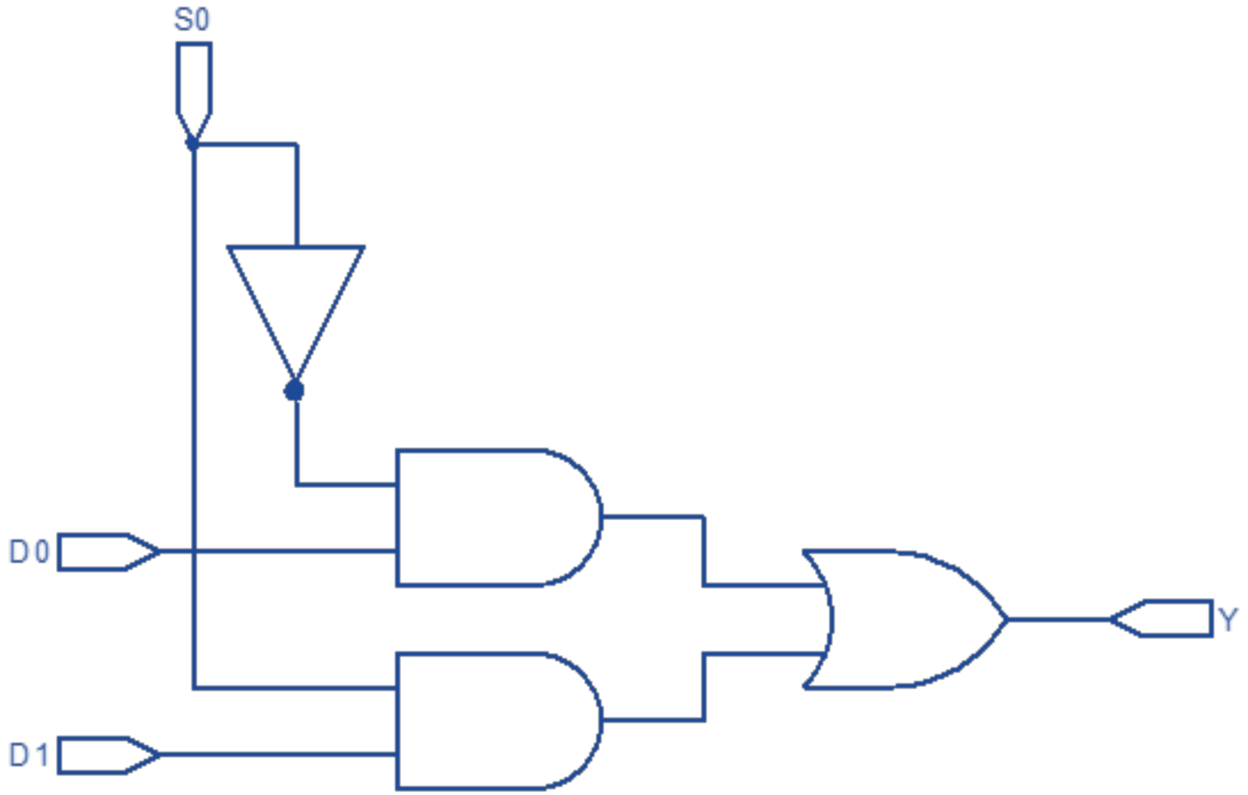
Su función es conectar a la única salida de datos una y sólo una de las entradas, viniendo decidida ésta por los valores que tomen las entradas de control.

Multiplexores.

MUX 2:1: Si tenemos una sola entrada de control S_0 (p = 1), tendremos $2^1 = 2$ entradas de datos (D_0 y D_1), y estamos en el caso de un multiplexor de 2 a 1.



S_0	Y
0	D_0
1	D_1

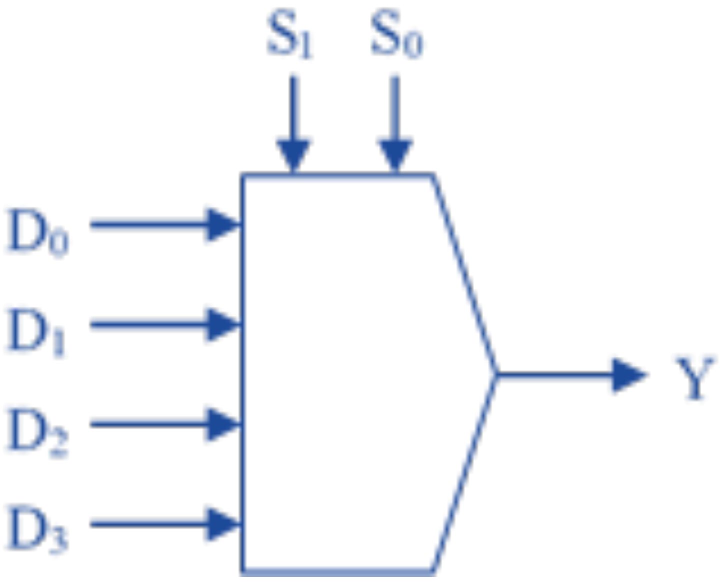


$$Y=\overline{S_0}\cdot D_0+S_0\cdot D_1$$

4.5.1.– Multiplexores.

MUX 4:1: Si tenemos dos entradas de control (p=2) S_1 y S_0 tendremos $2^2 = 4$ entradas de datos (D_3, D_2, D_1 y D_0) y es el caso de un multiplexor de 4 a 1.

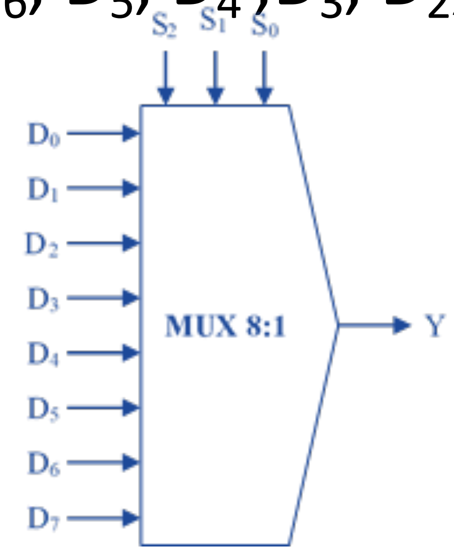
$$Y=\overline{S_1}\,\overline{S_0}\cdot D_0+\overline{S_1}S_0\cdot D_1+S_1\overline{S_0}\cdot D_2+S_1S_0\cdot D_3$$



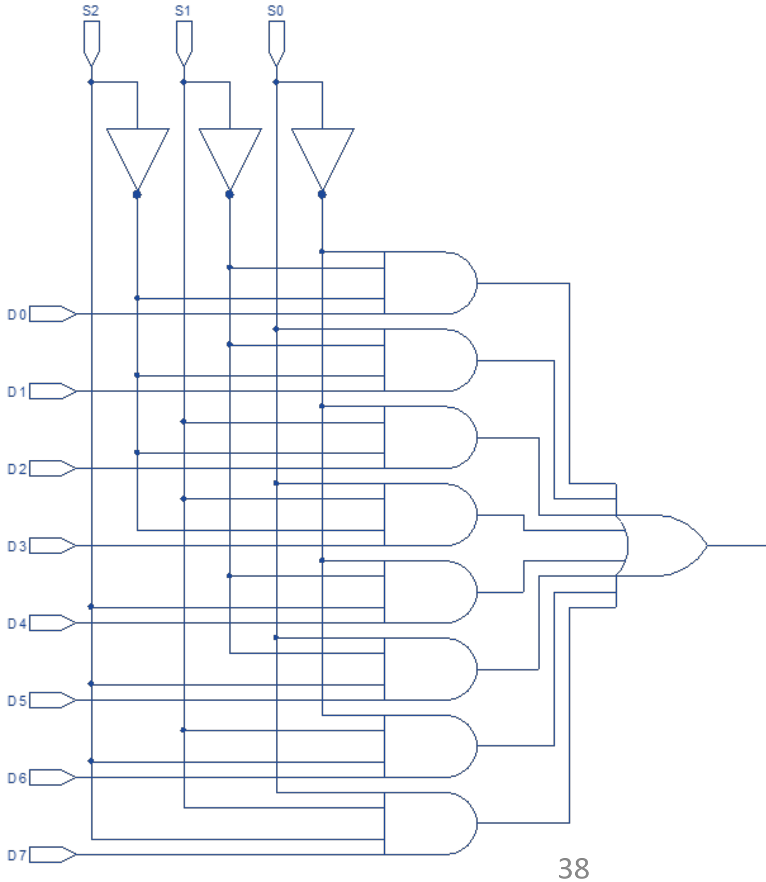
S_1	S_0	Y
0	0	D_0
0	1	D_1
1	0	D_2
1	1	D_3

Multiplexores.

MUX 8:1: Si tenemos tres entradas de control (p=3) S_2, S_1 y S_0 tendremos $2^3 = 8$ entradas de datos (D_7, D_6, D_5, D_4 ,D_3, D_2, D_1 y D_0) y es el multiplexor de 8 a 1.



S_2	S_1	S_0	Y
0	0	0	D_0
0	0	1	D_1
0	1	0	D_2
0	1	1	D_3
1	0	0	D_4
1	0	1	D_5
1	1	0	D_6
1	1	1	D_7



$$Y = \overline{S_2}\,\overline{S_1}\,\overline{S_0}\cdot D_0 + \overline{S_2}\,\overline{S_1}S_0\cdot D_1 + \overline{S_2}S_1\overline{S_0}\cdot D_2 + \overline{S_2}S_1S_0\cdot D_3 + $$
$$+ S_2\overline{S_1}\,\overline{S_0}\cdot D_{44} + S_2\overline{S_1}S_0\cdot D_5 + S_2S_1\overline{S_0}\cdot D_{66} + S_2S_1S_0\cdot D_{77}$$

Demultiplexores.

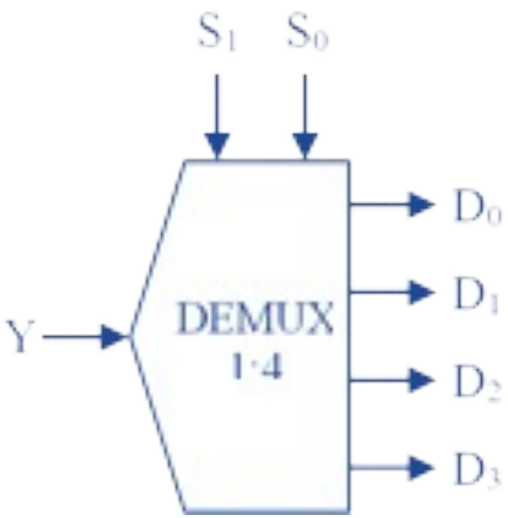
Un **demultiplexor** (DEMUX) básicamente realiza la función contraria al multiplexor encauzando los datos de una fuente común a diversos destinos.

También recibe el nombre de distribuidor de datos. Toma información de una sola línea y la distribuye a una de las $n = 2^p$ salidas de datos.

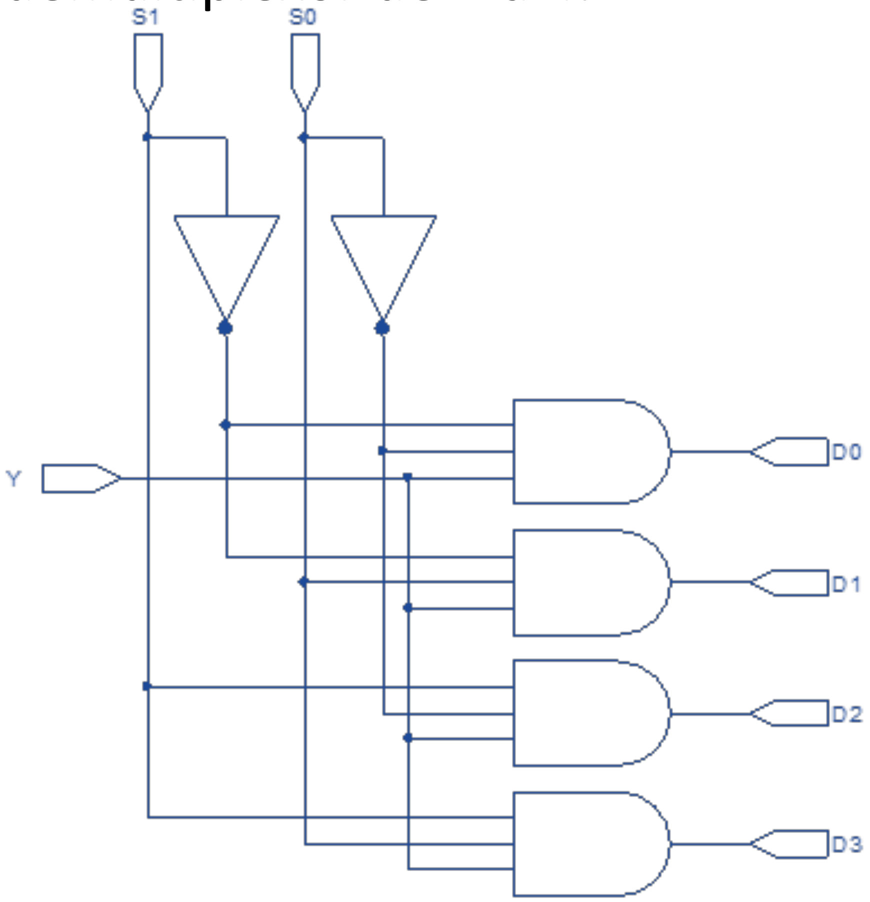
Un demultiplexor conecta su única entrada de datos con una y solo una de las salidas de datos, y en concreto, aquella cuyo subíndice coincide con el equivalente decimal del número binario aplicado en las entradas de control de selección.

Demultiplexores.

DEMUX 1:4: Tiene una única entrada, Y, dos entradas de control (p=2) S_1 y S_0, y $2^2 = 4$ salidas de datos (D_3, D_2, D_1 y D_0), es el caso de un demultiplexor de 1 a 4.



S_1	S_0	D_3	D_2	D_1	D_0
0	0	0	0	0	Y
0	1	0	0	Y	0
1	0	0	Y	0	0
1	1	Y	0	0	0



$D_0 = \overline{S_1}\,\overline{S_0} \cdot Y$

$D_1 = \overline{S_1}\,S_0 \cdot Y$

$D_2 = S_1\,\overline{S_0} \cdot Y$

$D_3 = S_1\,S_0 \cdot Y$

6. – Aplicaciones de los decodificadores y multiplexores.

Aplicaciones de los multiplexores.

Se distinguen varias aplicaciones, entre las que cabe destacar:

- Selector de datos.
- Multiplexación en el tiempo.
- Conversión paralelo–serie. La conversión serie–paralelo se haría con un demultiplexor.
- Implementación de funciones lógicas.

Como selector de datos, un multiplexor se puede usar para conectar dos o más fuentes de información a un único destino. Permite construir sistemas de bus común.

4.6.1.– Aplicaciones de los multiplexores.

Implementación de funciones lógicas mediante multiplexores:

Con un multiplexor de n variables de control podemos implementar cualquier función lógica de n variables sin más que conectar las variables de la función a las entradas de control y en las entradas de datos, conectarlas a la tensión de alimentación (“1” lógico) o a masa (“0” lógico) según sea el valor deseado de la función para cada combinación de entrada.

Por ejemplo sea la función dada como suma de mínterm siguiente:

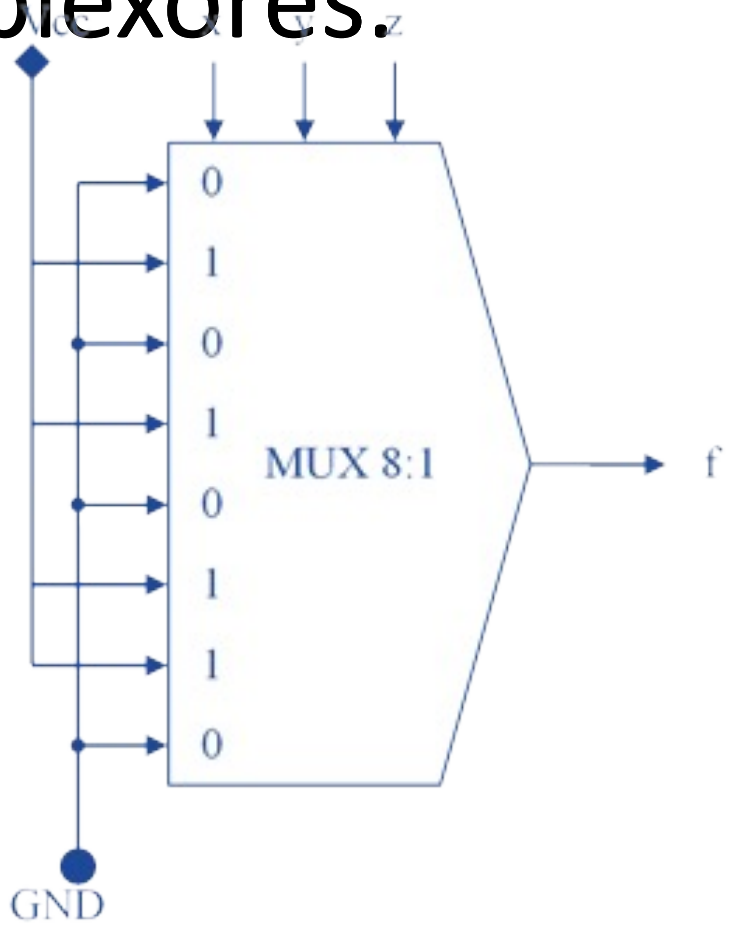
$$f(x,y,z) = \Sigma\, m\,(1, 3, 5, 6) = m_1 + m_3 + m_5 + m_6 = x'\cdot y'\cdot z + x'\cdot y\cdot z + x\cdot y'\cdot z + x\cdot y\cdot z'$$

4.6.1.– Aplicaciones de los multiplexores.

Implementación de funciones lógicas mediante multiplexores.

$f(x,y,z) = \Sigma\, m\,(1, 3, 5, 6) = m_1 + m_3 + m_5 + m_6 = x'\cdot y'\cdot z + x'\cdot yz + xy'\cdot z + xyz'$

x	y	z	f
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0



multiplexores.

- Si en las entradas de datos no sólo se aplican constantes (0 ó 1) sino también variables, cualquier multiplexor de n variables de control se puede utilizar para sintetizar cualquier función lógica de n+1 variables.

- Para ello se selecciona una de las variables, normalmente la menos significativa, y se desarrolla la función con respecto a las demás variables, agrupando las salidas de dos en dos.

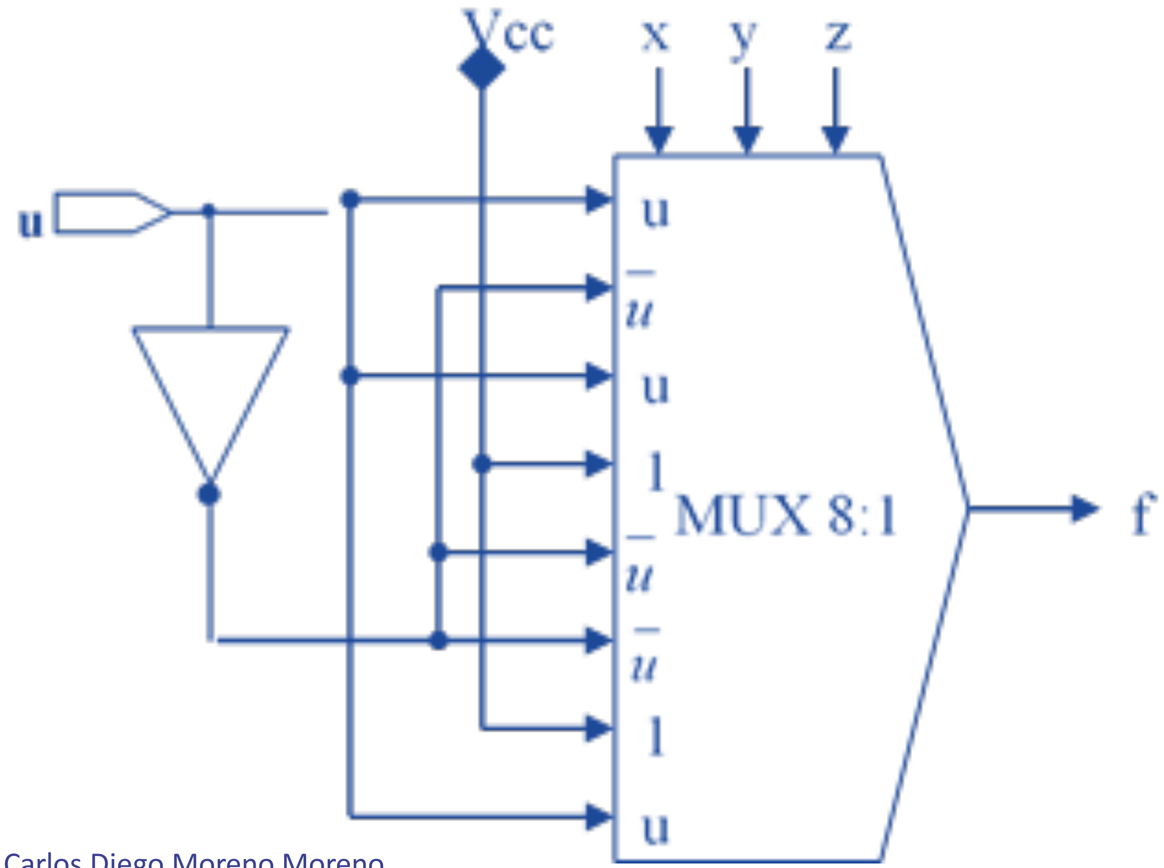
- Por ejemplo sea la función siguiente:

$$f = \Sigma\, m(1, 2, 5, 6, 7, 8, 10, 12, 13, 15)$$

4.6.1.– Aplicaciones de los multiplexores.

x	y	z	u	f	g
0	0	0	0	0	u
0	0	0	1	1	
0	0	1	0	1	$\bar{u}$
0	0	1	1	0	
0	1	0	0	0	u
0	1	0	1	1	
0	1	1	0	1	1
0	1	1	1	1	
1	0	0	0	1	$\bar{u}$
1	0	0	1	0	
1	0	1	0	1	$\bar{u}$
1	0	1	1	0	
1	1	0	0	1	1
1	1	0	1	1	
1	1	1	0	0	u
1	1	1	1	1	

$$f = \Sigma\, m(1, 2, 5, 6, 7, 8, 10, 12, 13, 15)$$



4.6.2.– Aplicaciones de los decodificadores.

- Se distinguen dos aplicaciones fundamentales:
  - Función de decodificación.
  - Implementación de funciones lógicas.
- Función de decodificación:
  - Usando un decodificador del código apropiado, se determina el carácter codificado en cada combinación binaria de dicho código aplicada en sus entradas.
  - Se suele usar para decodificar las direcciones de una Unidad Central de Proceso o Microprocesador, para determinar a qué dispositivo se accede.

4.6.2.– Aplicaciones de los decodificadores.

Implementación de funciones lógicas mediante decodificadores.

- Con un decodificador binario de n variables de entrada y 2^n variables de salida podremos implementar cualquier función de conmutación de n variables, utilizando además una puerta lógica adicional.

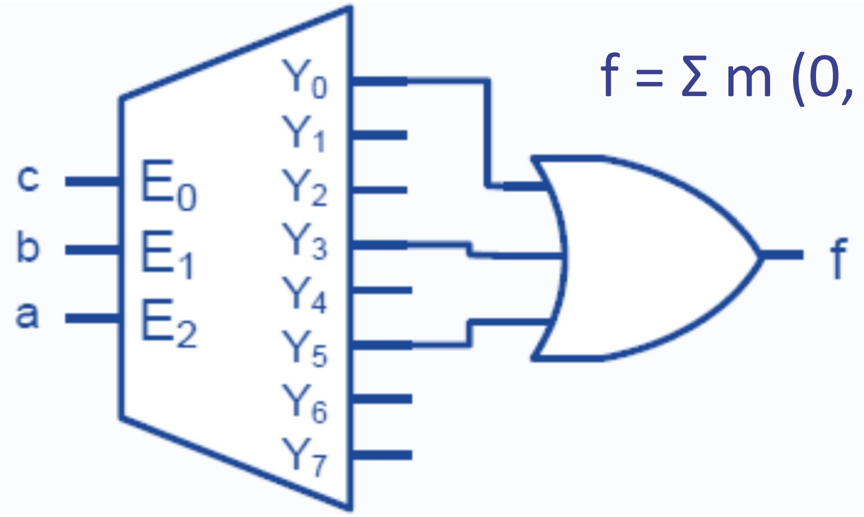
- Se nos pueden presentar los siguientes casos de implementación:
  - Decodificador con salidas activas a nivel alto y puertas OR.
  - Decodificador con salidas activas a nivel bajo y puertas AND.
  - Decodificador con salidas activas a nivel bajo y puertas NAND.
  - Decodificador con salidas activas a nivel alto y puertas NOR.

4.6.2.– Aplicaciones de los decodificadores.

Decodificador con salidas activas a nivel alto y puertas OR.

- Dado que un decodificador de n a 2^n con salidas activas a nivel lógico alto genera en sus salidas los 2^n minterms de sus n entradas, conectamos las variables de la función a las entradas del decodificador y **sumamos los mínterms (los “1”)**.

- Ejemplo: $f(a,b,c) = \sum m(0, 3, 5) + d(2,6) = \prod M(1,4,7) \cdot d(2,6)$.

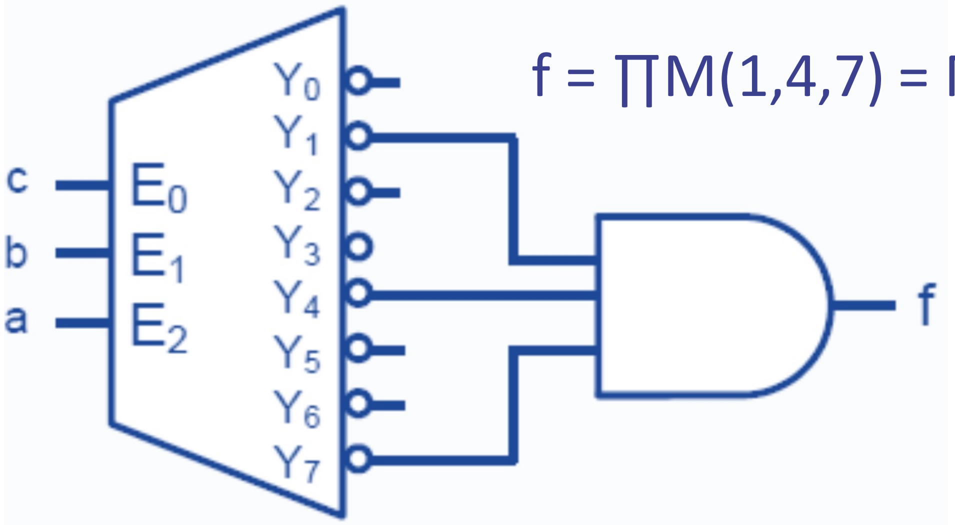


$f = \Sigma\, m(0, 3, 5) = m_0+m_3+m_5 = Y_0+Y_3+Y_5 = a'b'c' + a'bc + ab'c$

4.6.2.– Aplicaciones de los decodificadores.

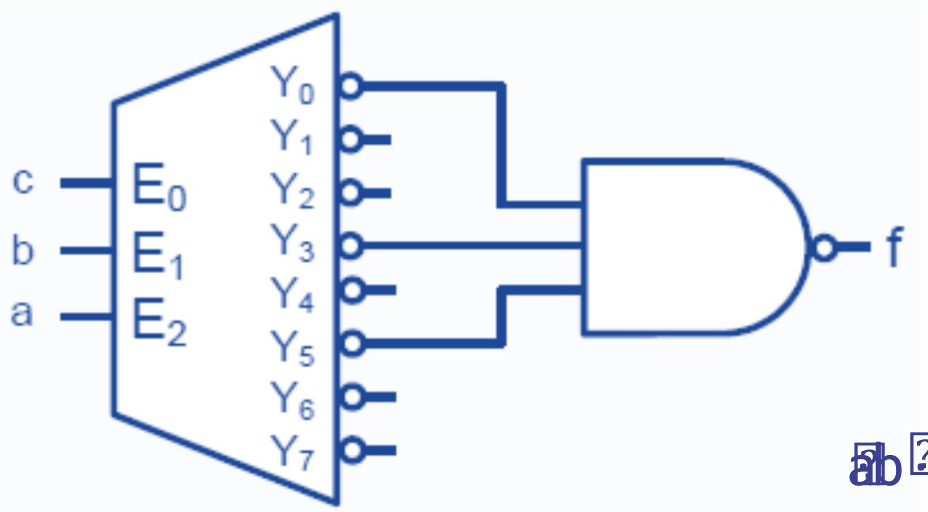
- Decodificador con salidas activas a nivel bajo y puertas AND.
- Dado que un decodificador de n a 2^n con salidas activas a nivel lógico bajo genera en sus salidas los 2^n maxterms de sus n entradas, conectamos las variables de la función a las entradas del decodificador y **multiplicamos los máxterms (los "0")**.
- Ejemplo: $f(a,b,c) = \sum m(0, 3, 5) + d(2,6) = \prod M(1,4,7) \cdot d(2,6)$.

$$f = \prod M(1,4,7) = M_1 \cdot M_4 \cdot M_7 = Y_1 \cdot Y_4 \cdot Y_7 = (a+b+\bar{c})\cdot(\bar{a}+b+c)\cdot(\bar{a}+\bar{b}+\bar{c})$$



4.6.2.– Aplicaciones de los decodificadores.

- **Decodificador con salidas activas a nivel bajo y puertas NAND.**
- En este caso conectamos las variables de la función a las entradas del decodificador, **multiplicamos los mínterms (los "1")** y los **complementamos**. Se obtiene la función como suma de mínterm.
- Ejemplo: f (a,b,c) = ∑ m(0, 3, 5) + d(2,6) = ∏M(1,4,7) · d(2,6).



$$f = \overline{\prod M(0,3,5)} = \overline{M_0 \cdot M_3 \cdot M_5} = \overline{Y_0 \cdot Y_3 \cdot Y_5}$$

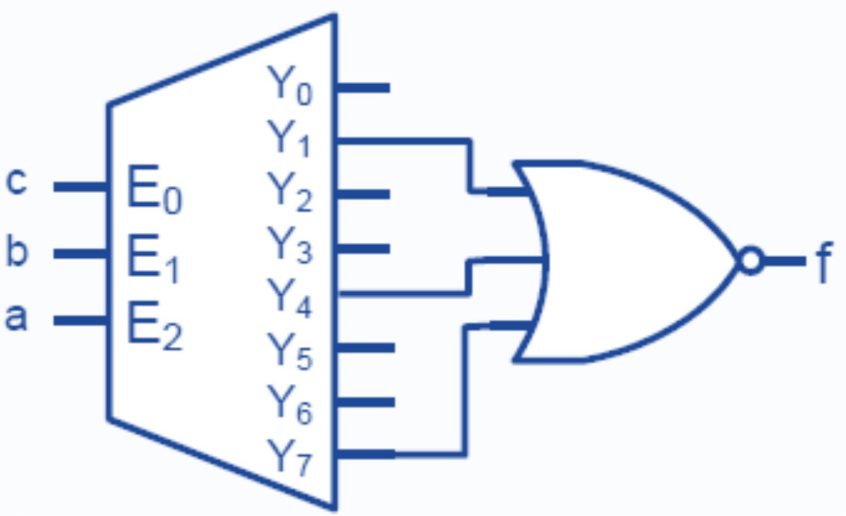
$$= \overline{(a+b+c)\cdot(a+\bar{b}+\bar{c})\cdot(\bar{a}+b+\bar{c})} =$$

$$\overline{(a+b+c)} + \overline{(a+\bar{b}+\bar{c})} + \overline{(\bar{a}+b+\bar{c})} =$$

$$\bar{a}\bar{b}\bar{c} + \bar{a}\cdot b\cdot c + a\cdot\bar{b}\cdot c = m_0+m_3+m_5 = \Sigma\, m\,(0, 3, 5)$$

4.6.2.– Aplicaciones de los decodificadores.

- Decodificador con salidas activas a nivel alto y puertas NOR.
- En este caso conectamos las variables de la función a las entradas del decodificador, **sumamos los máxterms (los "0")** y los **complementamos**. Se obtiene la función como producto de máxterm.
- Ejemplo: f (a,b,c) = ∑ m(0, 3, 5) + d(2,6) = ∏M(1,4,7) · d(2,6).



$$f = \overline{\Sigma\, m\,(1, 4, 7)} = \overline{m_1+m_4+m_7} = \overline{Y_1+Y_4+Y_7} =$$

$$\overline{\bar{a}\bar{b}c+a\cdot\bar{b}\cdot\bar{c}+a\cdot b\cdot c} = \left(\overline{\bar{a}\bar{b}c}\right)\cdot\left(\overline{a\cdot\bar{b}\cdot\bar{c}}\right)\cdot\left(\overline{a\cdot b\cdot c}\right) =$$

$$\left(a+b+\bar{c}\right)\cdot\left(\bar{a}+b+c\right)\cdot\left(\bar{a}+\bar{b}+\bar{c}\right) =$$

$$M_1\cdot M_4\cdot M_7 = \prod M(1,4,7)$$

TEMA 4: Análisis y síntesis de sistemas combinacionales. Circuitos MSI

4.1.– Definición de sistema combinacional.
4.2.– Análisis de circuitos combinacionales.
4.3.– Síntesis de circuitos combinacionales.
1. – Etapas del diseño.
2. – Implementación en dos niveles.
   1.– Con puertas básicas AND, OR y NOT.
   4.3.2.2.– Solamente con puertas NAND.
   4.3.2.3.– Solamente con puertas NOR.
4. – Decodificadores. Codificadores.
5. – Multiplexores. Demultiplexores.
6. – Aplicaciones de los decodificadores y multiplexores.

7. –Aritmética binaria básica: suma binaria; resta mediante el complemento a 2.

8. – Circuitos aritméticos binarios: Sumador completo; sumador paralelo con acarreo serie; restador.

4.7.– Aritmética binaria básica.

4.7.1.– Operaciones aritméticas con números binarios enteros sin signo.

- **Suma binaria:** Para sumar dos números binarios se siguen las siguientes reglas:
  - 0 + 0 = 0
  - 0 + 1 = 1
  - 1 + 0 = 1
  - 1 + 1 = 0 y "llevamos" 1

- Ejemplos:

Acarreo	0	0	0	1	1	0			1	1
	1	**0**	**1**	**0**	**0**	**1**	**1**			**8**	**3**
+		**1**	**0**	**0**	**1**	**1**	**0**	+		**3**	**8**
Resultado	1	1	1	1	0	0	1		1	2	1

Acarreo	1	1	1	1	1	0			1	
		1	**1**	**0**	**1**	**1**	**0**		**5**	**4**
+			**1**	**1**	**0**	**1**	**1**	+	**2**	**7**
Resultado	1	0	1	0	0	0	1		8	1

4.7.– Aritmética binaria básica.

4.7.1.– Operaciones aritméticas con números binarios enteros sin signo.

- **Resta binaria:** Para restar dos números binarios se siguen las siguientes reglas:
  - 0 – 0 = 0
  - 0 – 1 = 1 y "llevamos" 1
  - 1 – 0 = 1
  - 1 – 1 = 0

	1	**0**	**1**	**0**	**0**	**1**	**1**		**8**	**3**
–	^{1}	**1**	**0^1**	**0^1**	**1**	**1**	**0**	–	**3^1**	**8**
Resultado	0	1	0	1	1	0	1		4	5

- Ejemplos:

	1	**1**	**0**	**1**	**1**	**0**		**5**	**4**
–	^{1} **1^1**	**1**	**0^1**	**1^1**	**1**		–	**2^1**	**7**
Resultado	0	1	1	0	1	1		2	7

4.7.– Aritmética binaria básica.

4.7.1.– Operaciones aritméticas con números binarios enteros sin signo.

- **Multiplicación binaria:** Coincide con la multiplicación en decimal y equivale a la función lógica AND. Es el método de suma y desplazamientos.

x	y	x·y
0	0	0
0	1	0
1	0	0
1	1	1

- Ejemplo:

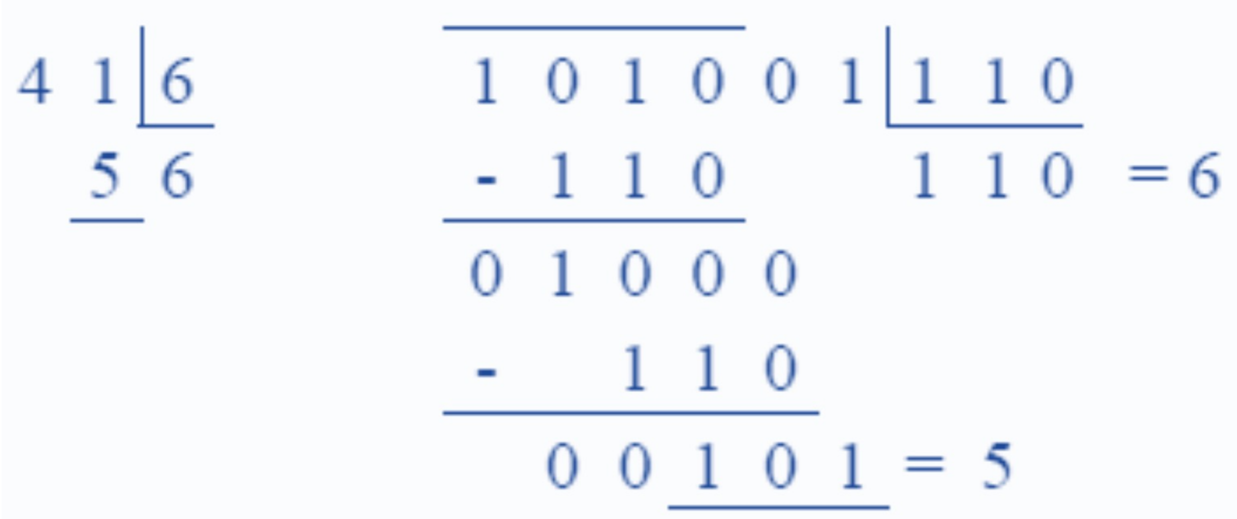
		1	1	Mdo
x		1	3	Mdor
		3	3	
	1	1		
	1	4	3	

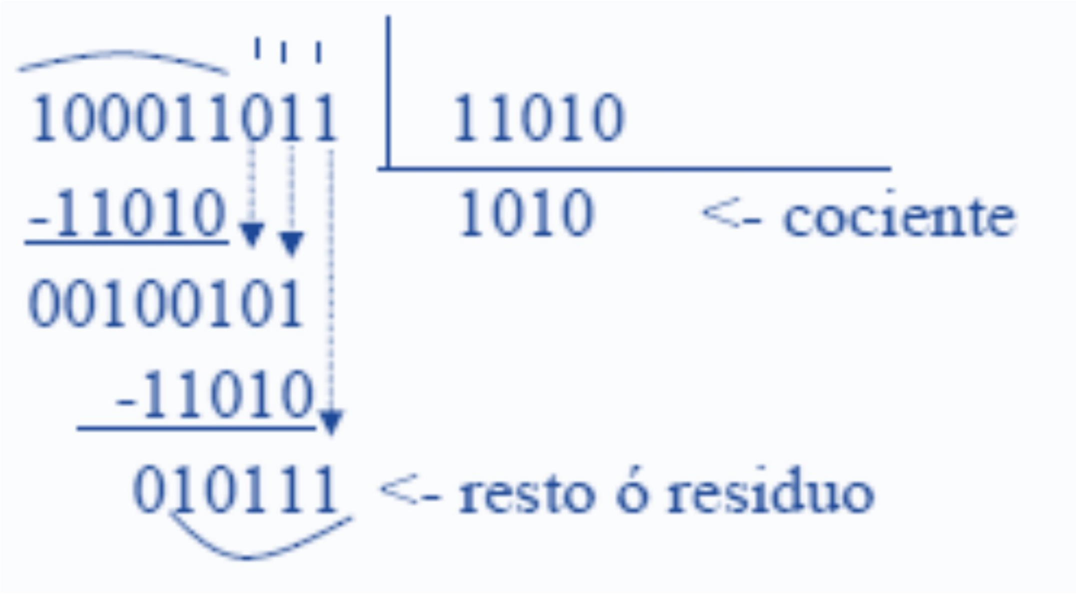
					1	0	1	1
				x	1	1	0	1
					1	0	1	1
				0	0	0	0	
			1	0	1	1		
		1	0	1	1			
	1	0	0	0	1	1	1	1

4.7.– Aritmética binaria básica.

4.7.1.– Operaciones aritméticas con números binarios enteros sin signo.

- **División binaria:** Coincide con la división en decimal basada en multiplicación y restas sucesivas:
- Ejemplos:





4.7.– Aritmética binaria básica.

4.7.2.– Representación y aritmética de los números binarios enteros con signo.

Concepto de complemento a la base y a la base menos uno:

- Dado un número entero positivo X en base b con n dígitos, se define el complemento a la base como **$X(Cb) = b^n - X$**.
  - Si b = 10 $\Rightarrow$ C10, Complemento a 10 $\Rightarrow$ $X(C10) = 10^n - X$
  - Si b = 2 $\Rightarrow$ C2 **, Complemento a 2** $\Rightarrow$ **$X(C2) = 2^n - X$**
- Dado un número entero positivo X en base b con n dígitos, se define el complemento a la base menos uno como **$X(Cb) = b^n - X - 1$**.
  - Si b = 10 $\Rightarrow$ C9, Complemento a 9 $\Rightarrow$ $X(C9) = 10^n - X - 1$
  - Si b = 2 $\Rightarrow$ C1 , **Complemento a 1** $\Rightarrow$ **$X(C1) = 2^n - X - 1$**
- .

4.7.– Aritmética binaria básica.

4.7.2.– Representación y aritmética de los números binarios enteros con signo.

Algoritmos rápidos de cálculo del C1 y C2:

- Para obtener el **complemento a 1** de un número en un sistema de representación de n bits **se cambian todos los bits**.

○ Ejemplo: $11010(C1) = 2^5 - 1 - 11010_2 = 100000_2 - 1 - 11010_2 = 11111_2 - 11010_2 = 00101_{C1}$

$$\mathbf{11010(C1) = 00101_{C1}}$$

- Para obtener el **complemento a 2** de un número en un sistema de representación de n bits **se cambian todos los bits y se le suma 1**.

○ Ejemplo: $11010(C2) = 2^5 - 11010_2 = 100000_2 - 11010_2 = 00110_{C2}$

$$\mathbf{11010(C2) = 00101_{C1} + 1 = 00110_{C2}}$$

4.7.– Aritmética binaria básica.

4.7.2.– Representación y aritmética de los números binarios enteros con signo.

Algoritmos rápidos de cálculo del C1 y C2:

- Otro algoritmo para obtener el **complemento a 2** de un número en un sistema de representación de n bits consiste en que comenzando por la derecha se dejan intactos todos los bits con valor cero y el primero con valor 1, y a partir de éste se complementan todos los bits.

- Ejemplo: $11010(C2) = 2^5 - 11010_2 = 100000_2 - 11010_2 = 00110_{C2}$

$11010(C2) = 00110_{C2}$

4.7.– Aritmética binaria básica.

4.7.2.– Representación y aritmética de los números binarios enteros con signo.

Resta mediante el complemento a 2:

- La representación en complemento a 2 se ideó para hacer más sencilla la suma y resta de números binarios enteros, con o sin signo, realizándola con el mismo circuito lógico.
- Realizar una resta utilizando el complemento a 2 consiste en sumarle al minuendo el complemento a 2 del sustraendo, despreciando el acarreo de la última cifra.

$$X - Y = X + Y_{C2}$$

- Los dos números deben tener el mismo número de bits, o sea que se utiliza la representación en complemento a 2 de un determinado número de bits.

4.7.– Aritmética binaria básica.

4.7.2.– Representación y aritmética de los números binarios enteros con signo.

Resta mediante el complemento a 2: (Utilizamos n = 4 bits)

- Ejemplos: $10 - 7 = 3 \qquad C2(-7) = C2\,(0111) = 1000 + 1 = 1001$ $\qquad 13 - 11 = 2; \qquad C2(-11) = C2(1011) = 0100 + 1 = 0101$

$$\begin{array}{ccccc} & 1 & 0 & 1 & 0 \\ & 1 & 0 & 0 & 1 \\ \hline 1 & 0 & 0 & 1 & 1 \end{array} \qquad\qquad \begin{array}{ccccc} & 1 & 1 & 0 & 1 \\ & 0 & 1 & 0 & 1 \\ \hline 1 & 0 & 0 & 1 & 0 \end{array}$$

- Si el minuendo es menor que el sustraendo, obtenemos el complemento a 2 del resultado de la operación.

$3 - 9 = -6\ ; \qquad C2(-9) = C2(1001) = 0110 + 1 = 0111$

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

$C2(-6) = C2(0110) = 1001 + 1 = 1010$

TEMA 4: Análisis y síntesis de sistemas combinacionales. Circuitos MSI

4.1.– Definición de sistema combinacional.
4.2.– Análisis de circuitos combinacionales.
4.3.– Síntesis de circuitos combinacionales.
1. – Etapas del diseño.
2. – Implementación en dos niveles.
    1.– Con puertas básicas AND, OR y NOT.
    4.3.2.2.– Solamente con puertas NAND.
    4.3.2.3.– Solamente con puertas NOR.
4. – Decodificadores. Codificadores.
5. – Multiplexores. Demultiplexores.
6. – Aplicaciones de los decodificadores y multiplexores.
7. – Aritmética binaria básica: suma binaria; resta mediante el complemento a 2.
8. – Circuitos aritméticos binarios: Sumador completo; sumador paralelo con acarreo serie; restador.

4.8.– Circuitos aritméticos binarios.

- Un circuito aritmético es un circuito combinacional que lleva a cabo operaciones aritméticas con números binarios o números decimales en código binario.
- El circuito aritmético básico es el sumador, ya que:
    - La resta se convierte en una suma mediante el C2.
    - La multiplicación supone la suma de los productos parciales.
    - La división implica la resta del divisor al dividendo parcial si el bit del cociente es 1.
- Vamos a desarrollar estos circuitos comenzando por lo más sencillo, y lo utilizaremos como bloque en circuitos más complejos, utilizando un diseño jerárquico.

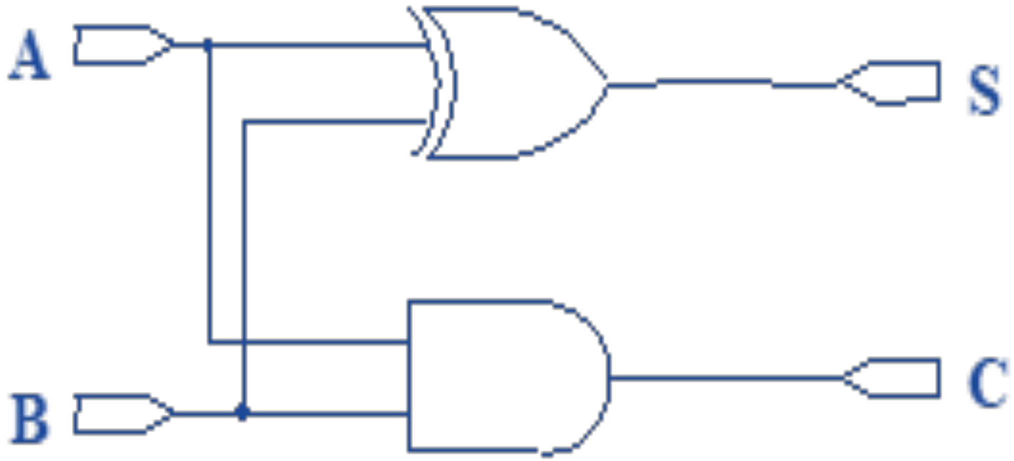
4.8.– Circuitos aritméticos binarios.

- **Semisumador (SS o HA):** Es un circuito aritmético que genera la suma de dos dígitos binarios. Dispone de dos entradas que son los sumandos (**A**, **B**) y dos salidas: una para la suma (**S**) y otra para el acarreo (**C**).

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

$$S = A\cdot\bar{B} + \bar{A}\cdot B = A \oplus B$$

$$C = A\cdot B$$



4.8.– Circuitos aritméticos binarios.

- **Sumador completo (SC o FA):** El sumador completo es un circuito combinacional que forma la suma aritmética de tres bits de entrada, y tiene dos salidas. Dos de las variables de entrada son los bits significativos a sumar (A y B), y la tercera entrada Ci–1 es el acarreo de la posición menos significativa anterior. Las dos variables de salida son el resultado de la suma (S) y el acarreo (Ci) que se propagará a la etapa siguiente.

- Vamos a ver dos implementaciones: una simplificando el acarreo de salida de forma independiente a la suma y, la otra, aprovechando una puerta XOR de la suma para implementar el sumador completo a partir de dos semisumadores.

4.8.– Circuitos aritméticos binarios.

Sumador completo (SC o FA): Primera implementación.

A	B	C_{i-1}	S	C_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

$A\backslash BC_{i-1}$	00	01	11	10
0		1		1
1	1		1	

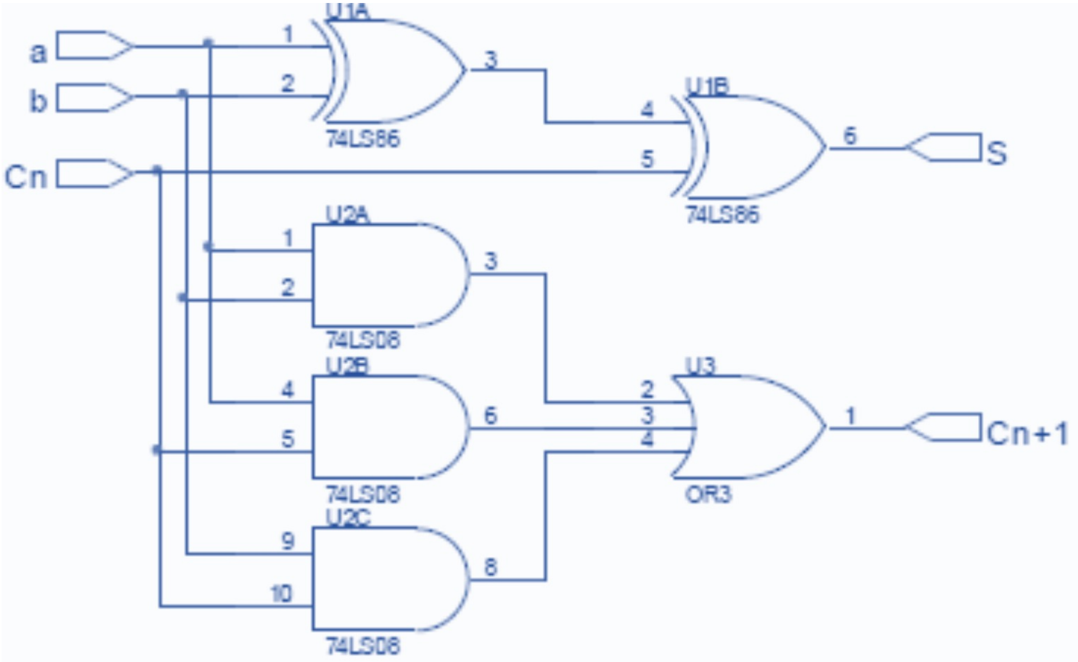
$$S=A'\cdot B'\cdot C_{i-1}+A'\cdot\overline{B\cdot C_{i-1}}+A\cdot B'\cdot\overline{C_{i-1}}+A\cdot B\cdot C_{i-1}=$$

$$(A'\cdot B+A\cdot B')\cdot\overline{C_{i-1}}+(A'\cdot B'+A\cdot B)\cdot C_{i-1}=$$

$$=(A\oplus B)\cdot\overline{C_{i-1}}+(\overline{A\oplus B})\cdot C_{i-1}=A\oplus B\oplus C$$

$A\backslash BC_{i-1}$	00	01	11	10
0			1	
1		1	1	1

$$C_i=A\cdot B+A\cdot C_{i-1}+B\cdot C_{i-1}$$



4.8.– Circuitos aritméticos binarios.

Sumador completo (SC o FA): Segunda implementación.

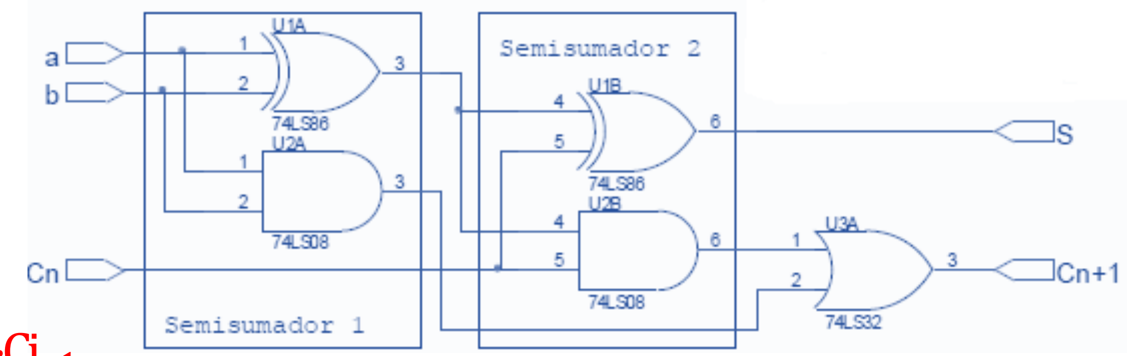
A	B	C_{i-1}	S	C_i
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

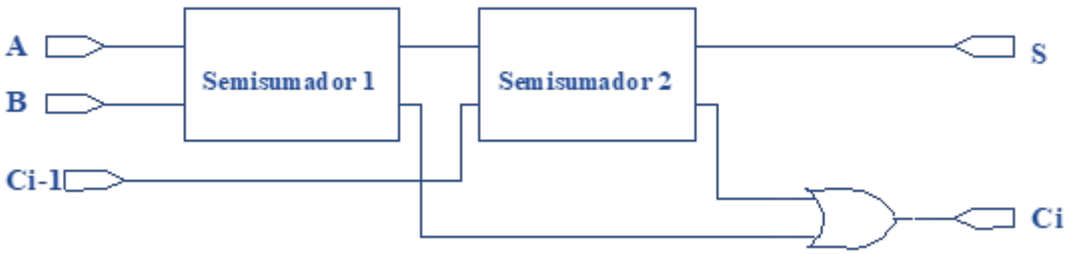
A\B$C_{i-1}$	00	01	11	10
0		1		1
1	1		1	

$$S=A'\cdot B'\cdot C_{i-1}+A'\cdot B\cdot\overline{C_{i-1}}+A\cdot B'\cdot\overline{C_{i-1}}+A\cdot B\cdot C_{i-1}=$$

$$(A'\cdot B+A\cdot B')\cdot\overline{C_{i-1}}+(A'\cdot B'+A\cdot B)\cdot C_{i-1}=$$

$$=(A\oplus B)\cdot\overline{C_{i-1}}+\overline{(A\oplus B)}\cdot C_{i-1}=A\oplus B\oplus C$$



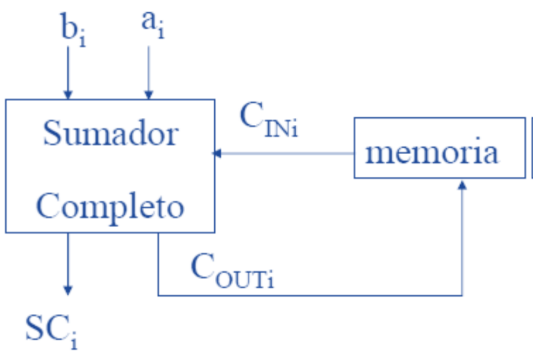


A\B$C_{i-1}$	00	01	11	10
0			1	
1		1	1	1

$$C_i=A\cdot B+A\cdot B'\cdot C_{i-1}+A'\cdot B\cdot C_{i-1}=A\cdot B+(A\cdot B'+A'\cdot B)\cdot C_{i-1}=A\cdot B+(A\oplus B)\cdot C_{i-1}$$

4.8.– Circuitos aritméticos binarios.

- **Sumador serie:** Realiza simultáneamente la suma de dos bits, uno de cada número, y el acarreo procede de la suma anterior. Por lo tanto, está formado básicamente por un solo sumador completo, pero ha de poseer un elemento que memorice el acarreo.



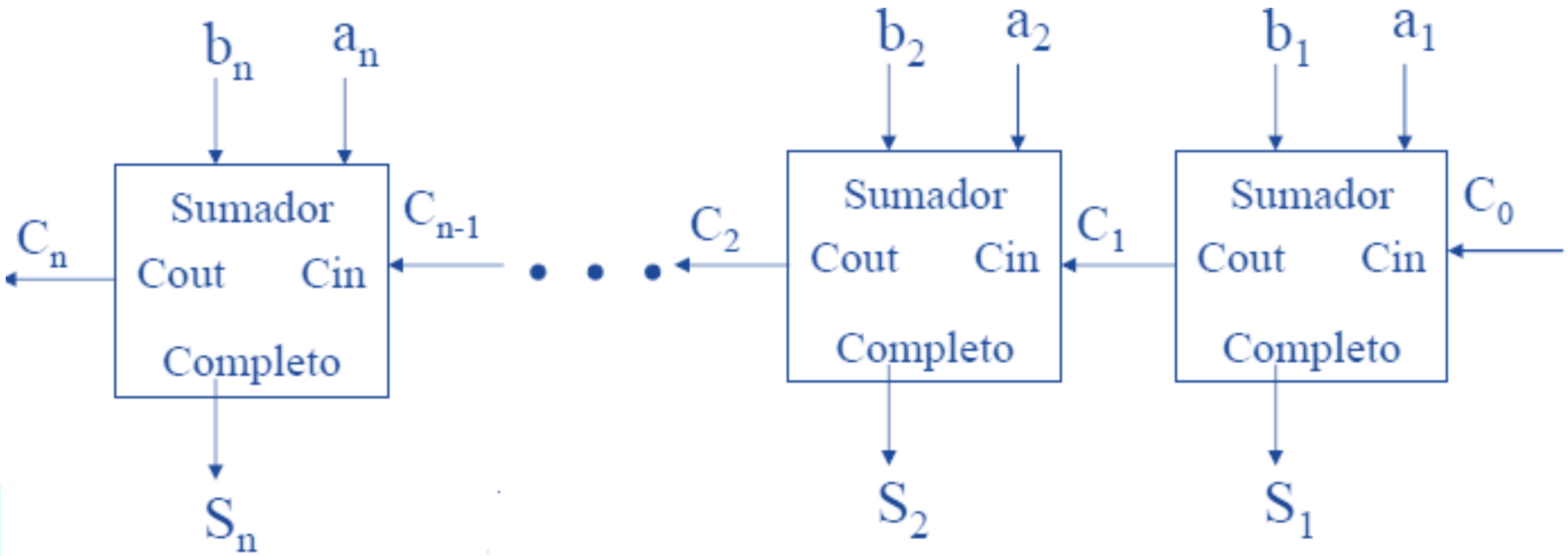
- **Sumador paralelo:** Realiza simultáneamente la suma de dos números de n bits, y para ello utiliza n sumadores completos. El acarreo puede generarse en serie o en paralelo (anticipado). Vamos a ver sólo el paralelo con acarreo serie.

4.8.– Circuitos aritméticos binarios.

- **Sumador paralelo con acarreo serie:** Cada sumador completo realiza la suma de dos bits y el acarreo procedente del sumador de la etapa anterior.
- El acarreo se propaga en serie de un sumador al siguiente, y por lo tanto, el tiempo que tarda en producirse el resultado de la suma es n veces el tiempo que tarda en generarse el acarreo en un sumador completo.
- Ventajas: sencillez y bajo coste.
- Inconvenientes: lentitud. Aunque todos los bits de los dos números se apliquen al mismo tiempo en las entradas, la suma no es correcta hasta que no se hayan propagado todos los acarreos.

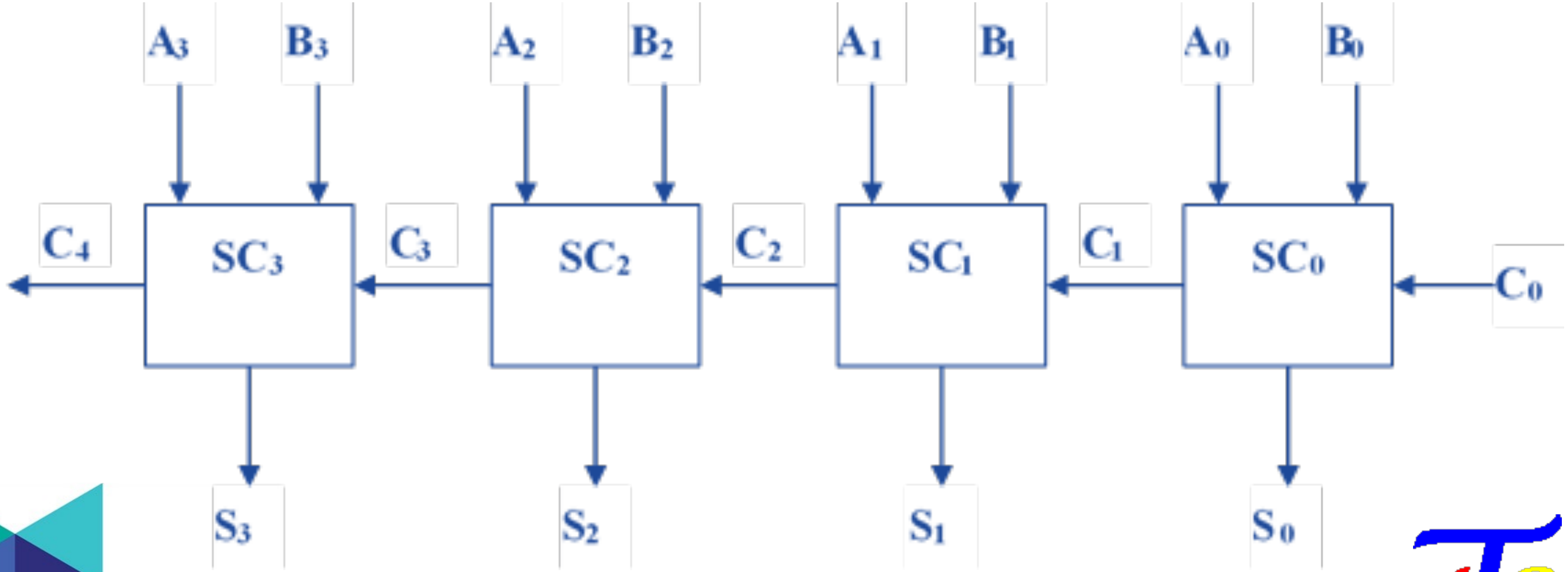
4.8.– Circuitos aritméticos binarios.

- **Sumador paralelo con acarreo serie:**
- Sean dos números $A \equiv a_n, a_{n-1}, \ldots, a_1, a_0$ y $B \equiv b_n, b_{n-1}, \ldots, b_1, b_0$.



4.8.– Circuitos aritméticos binarios.

- **Sumador paralelo con acarreo serie de 4 bits:**
- Sean dos números de 4 bits $A \equiv A_3, A_2, A_1, A_0$ y $B \equiv B_3, B_2, B_1, B_0$.



4.8.– Circuitos aritméticos binarios.

- **Sumador/restador en complemento a 2:**
- La resta se realiza a partir de la suma, y por este motivo se construye más usualmente una unidad funcional que realice tanto la suma como la resta. Son los circuitos sumadores/ restadores.
- La resta binaria puede realizarse sumando al minuendo el complemento a dos del sustraendo.
- El complemento a dos se obtiene complementando cada bit del sustraendo y sumando uno. Esta suma del 1 se lleva a cabo poniendo a 1 el acarreo de entrada C_0, mientras se suma al minuendo el complemento del sustraendo.

4.8.– Circuitos aritméticos binarios.

- **Sumador/restador en complemento a 2:**
- Este sumador/ restador en complemento a dos, tendrá dos entradas de 4 bits, $\mathbf{A=A_3A_2A_1A_0}$ y $\mathbf{B=B_3B_2B_1B_0}$, y una salida $\mathbf{S=S_3S_2S_1S_0}$, además de una señal de selección **S** que cuando esté a 0 realizará la suma, y cuando esté a 1 realizará la resta.
- Esta operación se consigue incluyendo una puerta **XOR** en cada sumador completo que tiene como entradas la señal de selección S y cada uno de los bits de entrada del sustraendo B_i.

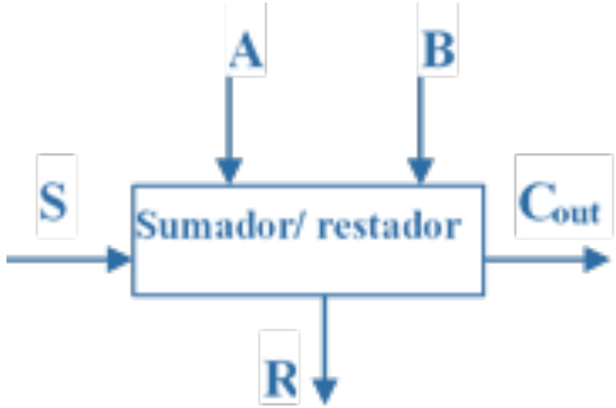
4.8.– Circuitos aritméticos binarios.

Sumador/restador en complemento a 2:

- Cuando $S = 0 \Rightarrow B_i \oplus 0 = B_i \cdot 1 + \overline{B}_i \cdot 0 = B_i$ y se realiza la suma **A + B**.
- Cuando $S = 1 \Rightarrow B_i \oplus 1 = B_i \cdot 0 + \overline{B}_i \cdot 1 = \overline{B}_i$ y se realiza la resta **$A - B = A + \overline{B} + 1$**.

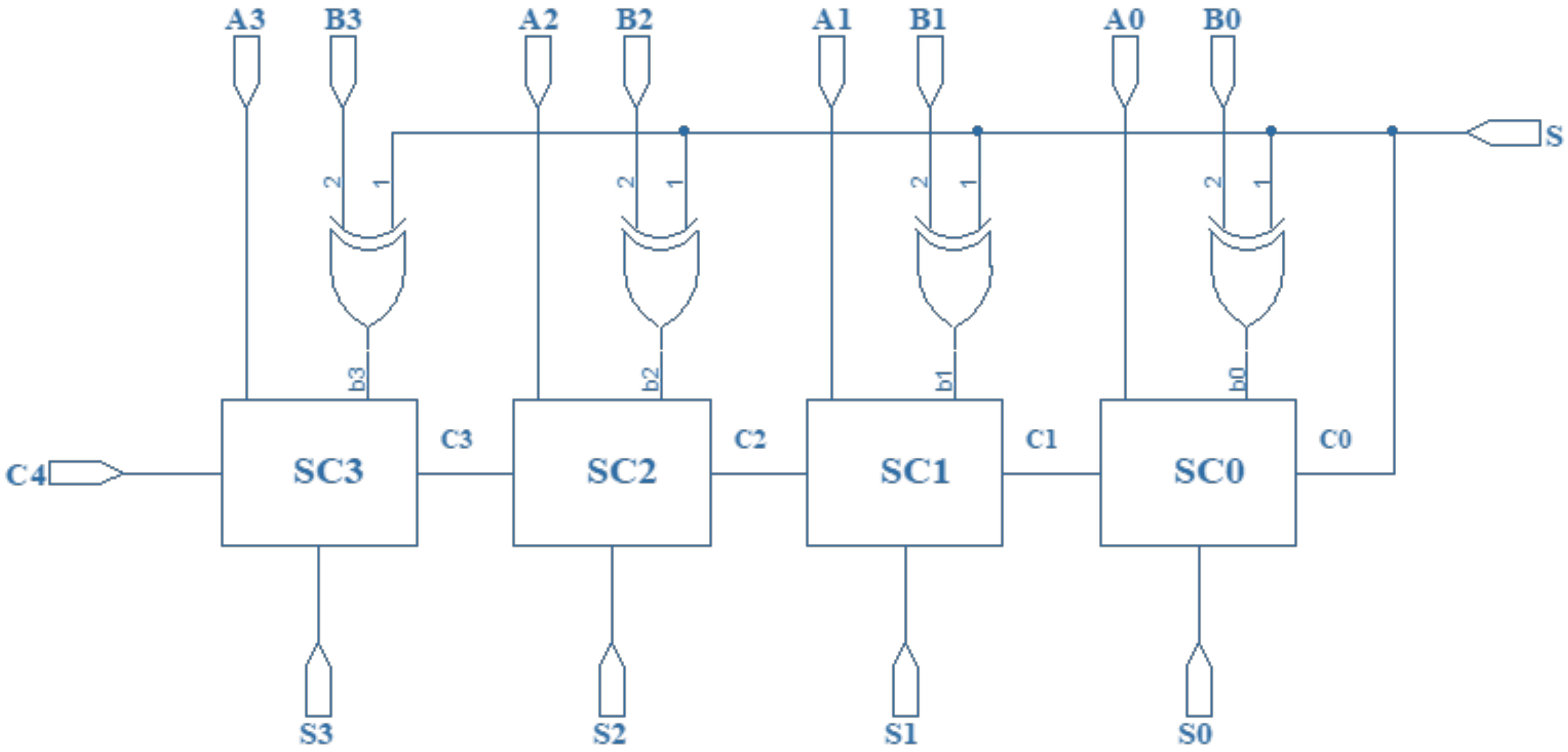
Además la señal **S** de selección se conecta al acarreo de entrada del primer sumador completo, con lo cual sumamos 1.

Operación	S	Función
Suma	**0**	**R=A+B**
Resta	**1**	**R=A+$\overline{B}$+1**



4.8.– Circuitos aritméticos binarios.

Sumador/restador en complemento a 2:

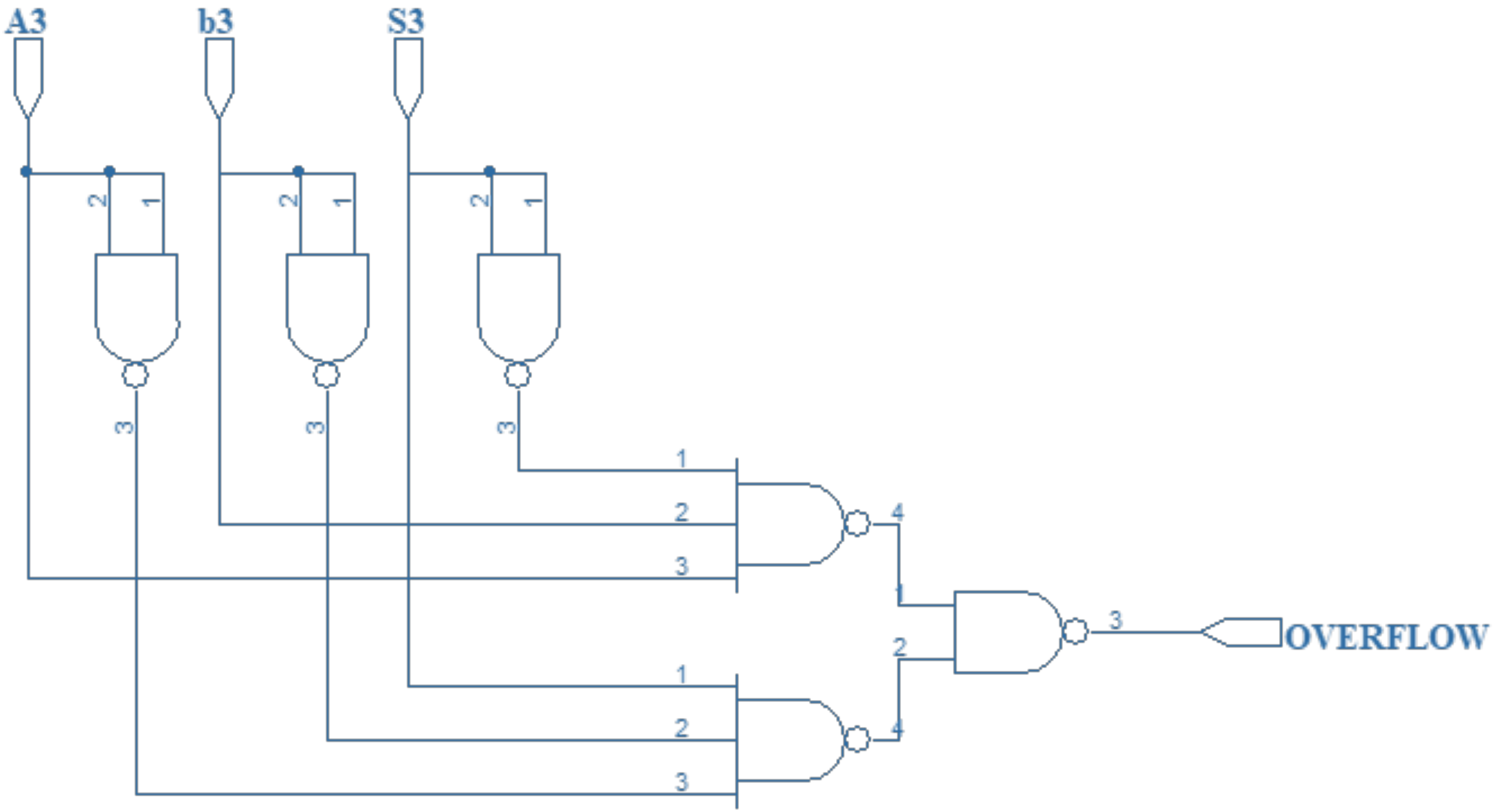


4.8.– Circuitos aritméticos binarios.

- **Sumador/restador en complemento a 2: Desbordamiento (*overflow*):**
- Al circuito anterior se le podría añadir un circuito lógico combinacional que detecte la condición de desborde de suma y resta. Estas condiciones son:
    - Si ambos operandos son positivos (bit de signo es cero) y el resultado es negativo (bit de signo es uno).
    - Si ambos operandos son negativos (bit de signo es uno) y el resultado es positivo (bit de signo es cero).
- **OVERFLOW**$=A3\cdot B3\cdot\overline{S3}+\overline{A3}\cdot\overline{B3}\cdot S3=\overline{\overline{(A3\cdot B3\cdot\overline{S3})+(\overline{A3}\cdot\overline{B3}\cdot S3)}}=\overline{\overline{(A3\cdot B3\cdot\overline{S3})}\cdot\overline{(\overline{A3}\cdot\overline{B3}\cdot S3)}}$

4.8.– Circuitos aritméticos binarios.

Sumador/restador en complemento a 2: Desbordamiento (overflow):



Bibliografía consultada.

- Antonio Lloris y Alberto Prieto. Sistemas Digitales 2º ed. Mc Graw-Hill. 2003.
- Daniel D. Gajski. Principios de Diseño Digital. Ed. Prentice Hall. 1997.
- Enrique Mandado. Sistemas Electrónicos Digitales. Ed. Marcombo. 1998.
- M. Morris Mano, Charles R. Kime. Fundamentos del Diseño Lógico y Computadoras. Ed. Prentice Hall. 1998.

- ¡Muchas gracias por su atención!

- Carlos Diego Moreno Moreno

Área de Arquitectura y Tecnología de Computadores
Departamento de Ingeniería Electrónica y de Computadores.
Escuela Politécnica Superior. Universidad de Córdoba

UNIVERSIDAD D CORDOBA