

Diseño de Sistemas Digitales

FCHE 2011-2

Bloques Combinacionales
Fundamentales

2 Circuitos combinacionales

Objetivo: El alumno conocerá los componentes electrónicos básicos involucrados en los circuitos combinacionales, así como los bloques funcionales combinacionales más utilizados en el diseño de sistemas digitales tanto en descripción estructural como por comportamiento usando HDL.

Contenido:

- 2.1 Compuertas lógicas AND, OR, NOT, NAND, NOR, XOR, XNOR
- 2.2 Formas canónicas, estándar, mintérminos y maxtérminos
- 2.3 Minimización de funciones booleanas con mapas de Karnaugh y Quine-McCluskey
- 2.4 Circuitos integrados, familias lógicas
- 2.5 Interpretación de parámetros en las hojas de datos de las compuertas lógicas
- 2.6 Convenciones de lógica positiva y lógica negativa
- 2.7 Representación estructural y por comportamiento de las compuertas lógicas en algún lenguaje de descripción de hardware (HDL)
- 2.8 Implementación estructural y por comportamiento de Funciones Booleanas usando algún HDL
- 2.9 Universalidad de las compuertas NAND y NOR
- 2.10 Propiedades de la XOR y NXOR para la implementación de generadores y detectores de paridad
- 2.11 Análisis de tiempo en la implementación de funciones booleanas
- 2.12 Descripción estructural y por comportamiento, usando HDL, de los bloques combinacionales fundamentales: Medio sumador, sumador completo, sumador/restador de "n" bits, comparadores de "n" bits, sumadores BCD, multiplicadores de "NxM" bits, decodificadores, codificadores, decodificadores BCD – 7 SEG, multiplexores, demultiplexores
- 2.13 Dispositivos lógicos programables elementales: ROM, PLA, PAL
- 2.14 Implementación de funciones booleanas con decodificadores, multiplexores, ROM, PLA y PAL

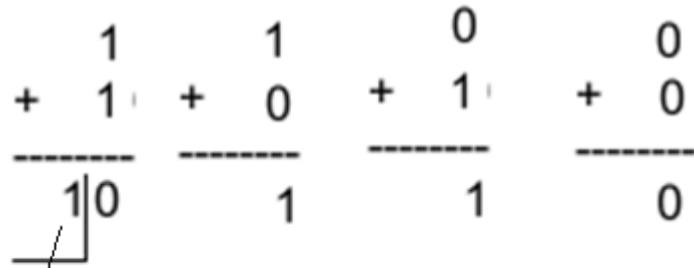
PROCEDIMIENTO DE DISEÑO

El diseño de circuitos combinacionales comienza desde el enunciado del problema y termina con el diagrama de circuito lógico, o con un conjunto de funciones de Boole de los cuales se puede obtener el diagrama lógico fácilmente. El procedimiento cubre los siguientes pasos:

LOGICA COMBINACIONAL

1. Se enuncia el problema.
2. Se determina el número requerido de variables de entrada y el número requerido de variables de salida.
3. Se le asignan letras a las variables de entrada y salida.
4. Se deduce la tabla de verdad que define las relaciones entre las entradas y las salidas.
5. Se obtiene la función de Boole simplificada para cada salida.
6. Se dibuja el diagrama lógico.

Medio Sumador

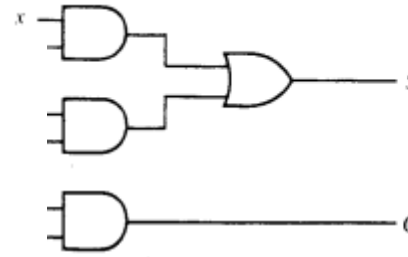


Carry o Acarreo

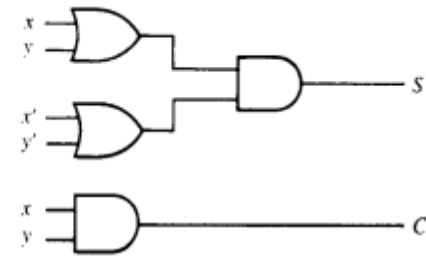
x	y	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

$$S = x'y + xy'$$

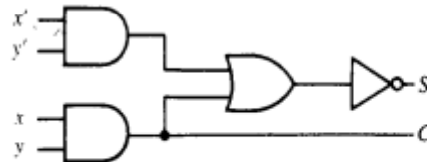
$$C = xy$$



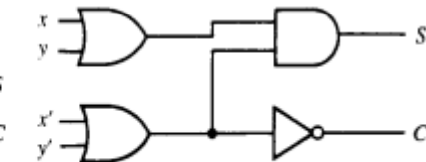
(a) $S = xy' + x'y$
 $C = xy$



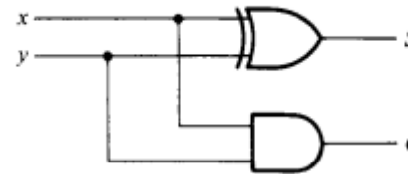
(b) $S = (x+y)(x'+y')$
 $C = xy$



(c) $S = (C + x'y)$
 $C = xy$



(d) $S = (x+y)(x'+y')$
 $C = (x'+y')$



(e) $S = x \oplus y$
 $C = xy$

$$S' = xy + x'y' \quad C = xy$$

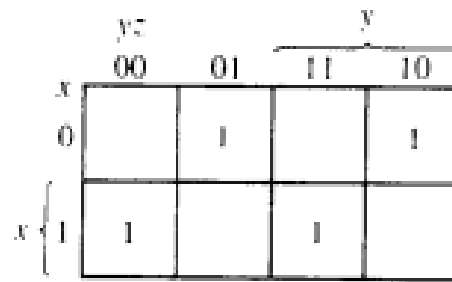
$$S = (C + x'y)'$$

$$C = xy = (x' + y')$$

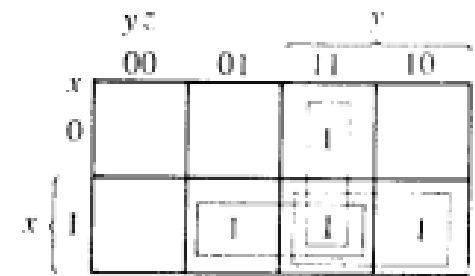
$$\begin{array}{r}
 \\
 + \quad 00 \\
 \quad 01 \\
 \hline
 \quad 01
 \end{array}
 \quad
 \begin{array}{r}
 \\
 + \quad 11 \\
 \quad 01 \\
 \hline
 \quad 100
 \end{array}
 \quad
 \begin{array}{r}
 \\
 + \quad 11 \\
 \quad 11 \\
 \hline
 \quad 110
 \end{array}$$

x $c2$	y	z	C	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Sumador Completo



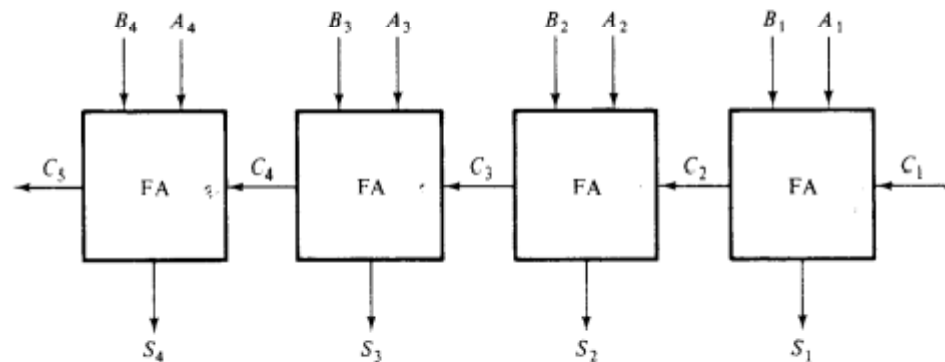
$$S = x'y'z + x'yz' + xy'z' + xyz$$



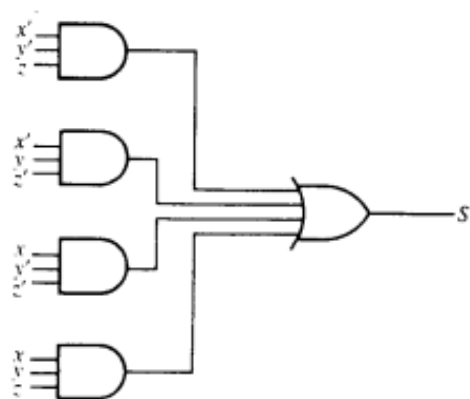
$$C = xy + xz + yz$$

$$S = x'y'z + x'yz' + xy'z' + xyz$$

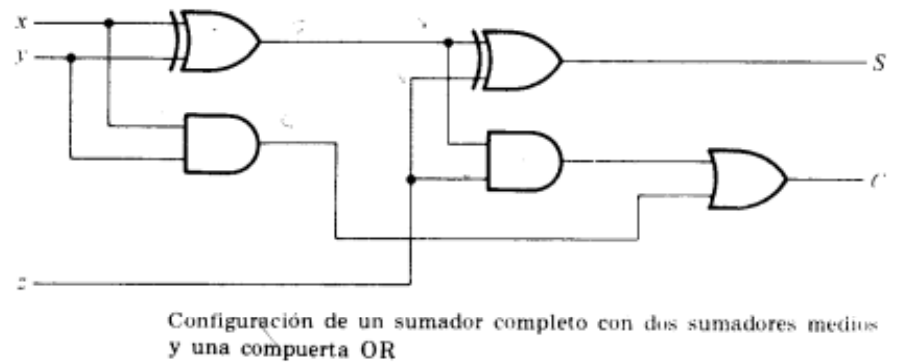
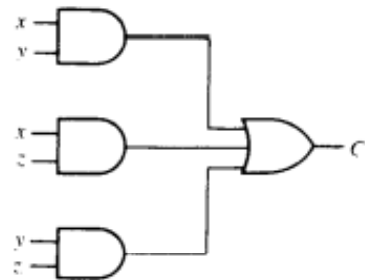
$$C = xy + xz + yz$$



Sumadores completos de 4 bits



Configuración de un sumador completo en suma de productos



Configuración de un sumador completo con dos sumadores medios y una compuerta OR

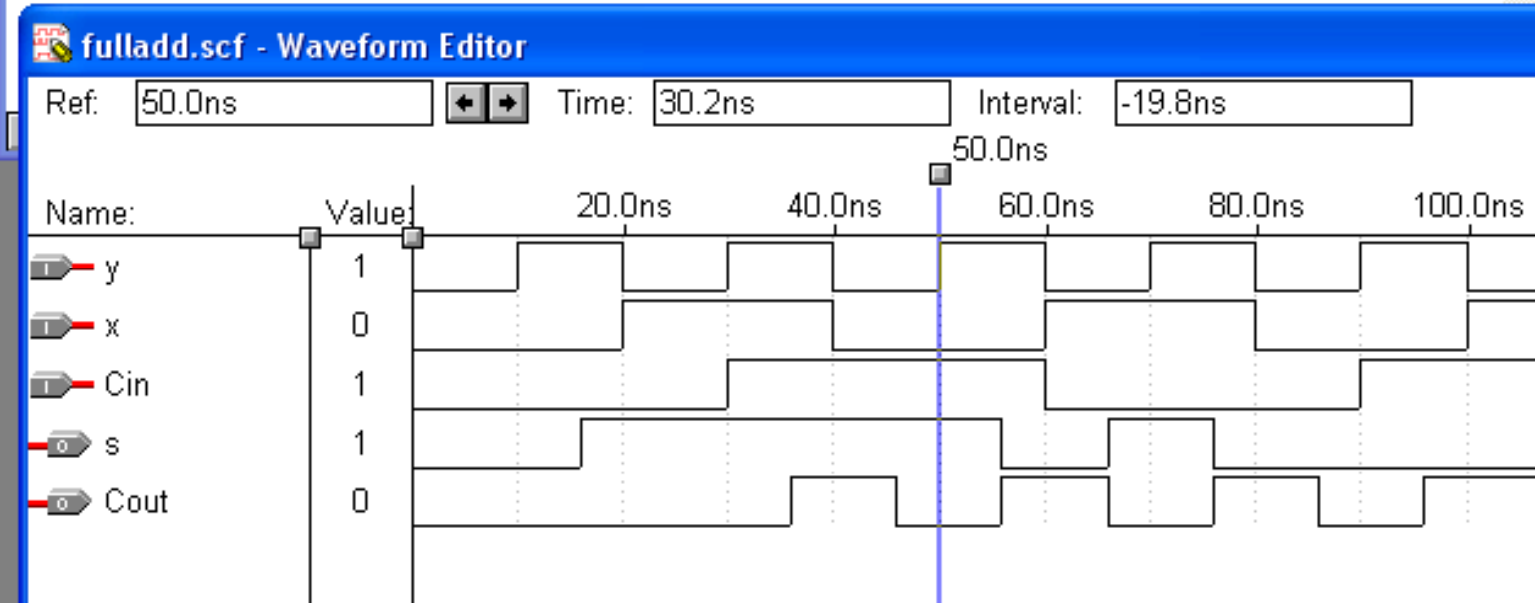
$$\begin{aligned}
 S &= z \oplus (x \oplus y) \\
 &= z'(xy' + x'y) + z(xy' + x'y)' \\
 &= z'(xy' + x'y) + z(xy + x'y') \\
 &= xy'z' + x'yz' + xyz + x'y'z
 \end{aligned}$$

$$C = z(xy' + x'y) + xy = xy'z + x'yz + xy$$

```
fulladd.vhd - Text Editor
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;

ENTITY fulladd IS
    PORT (
        Cin, x, y      : IN    STD_LOGIC ;
        s, Cout        : OUT   STD_LOGIC ) ;
END fulladd ;

ARCHITECTURE LogicFunc OF fulladd IS
BEGIN
    s <= x XOR y XOR Cin ;
    Cout <= (x AND y) OR (Cin AND x) OR (Cin AND y) ;
END LogicFunc ;
```



Restador

$$\begin{array}{r}
 1 \\
 - 1 - 0 - 1 \\
 \hline
 0 1 - 1
 \end{array}$$

$$\begin{aligned}
 D &= x'y + xy' \\
 B &= x'y
 \end{aligned}$$

x	y	B	D
0	0	0	0
0	1	1	1
1	0	0	1
1	1	0	0

$$\begin{array}{r}
 00 \\
 - 01 - 01 \\
 \hline
 01 10
 \end{array}$$

\downarrow prestado
 \downarrow Borrow

x	yz		y	
	00	01	11	10
0		1		1
1	1		1	

$$D = x'y'z + x'yz + xy'z' + xyz$$

x	yz		y	
	00	01	11	10
0		1	1	1
1			1	

$$B = x'y + x'z + yz$$

$$D = x'y'z + x'yz' + xy'z' + xyz$$

$$B = x'y + x'z + yz$$

x	y	debe z	B	D
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	1	0
1	0	0	0	1
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

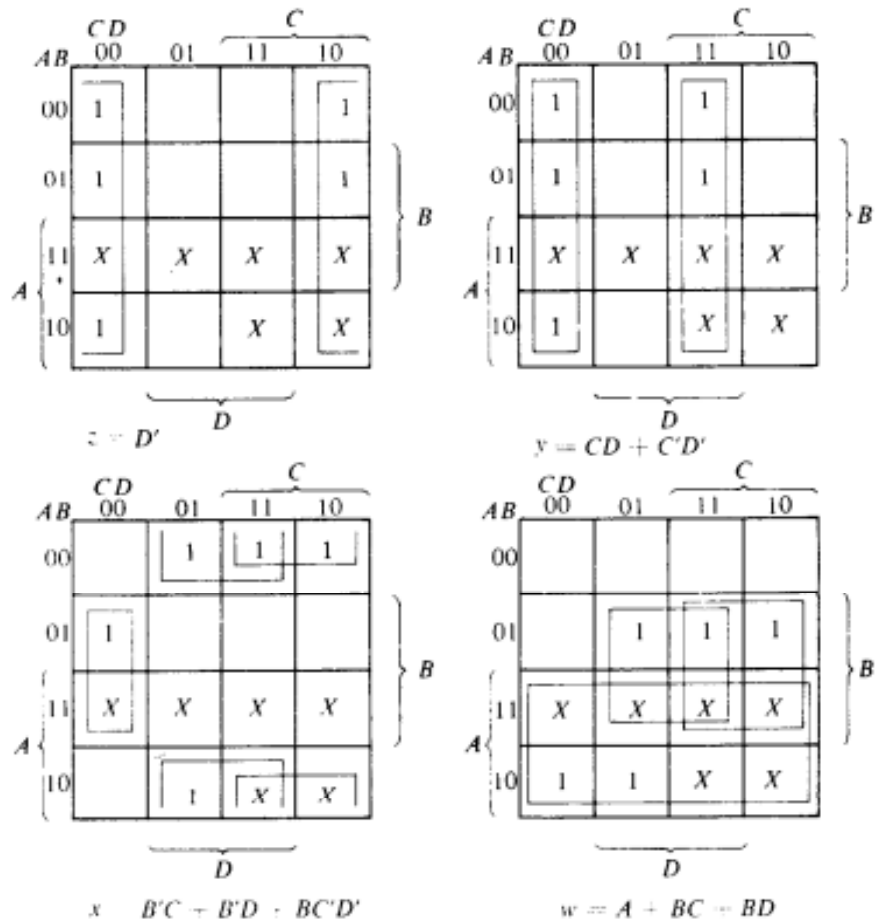
debe x	y	z	B	D
0	0	0	0	0
0	0	1	1	1
0	1	0	0	1
0	1	1	0	0
1	0	0	1	1
1	0	1	1	0
1	1	0	0	0
1	1	1	1	1

Conversión entre códigos

un conversor de código es un circuito que hace compatibles dos sistemas a pesar de que ambos tengan diferente código binario.

Tabla de verdad para el ejemplo de conversión de código

Entrada BDC				Salida código exceso 3			
A	B	C	D	w	x	y	z
0	0	0	0	0	0	1	1
0	0	0	1	0	1	0	0
0	0	1	0	0	1	0	1
0	0	1	1	0	1	1	0
0	1	0	0	0	1	1	1
0	1	0	1	1	0	0	0
0	1	1	0	1	0	0	1
0	1	1	1	1	0	1	0
1	0	0	0	1	0	1	1
1	0	0	1	1	1	0	0

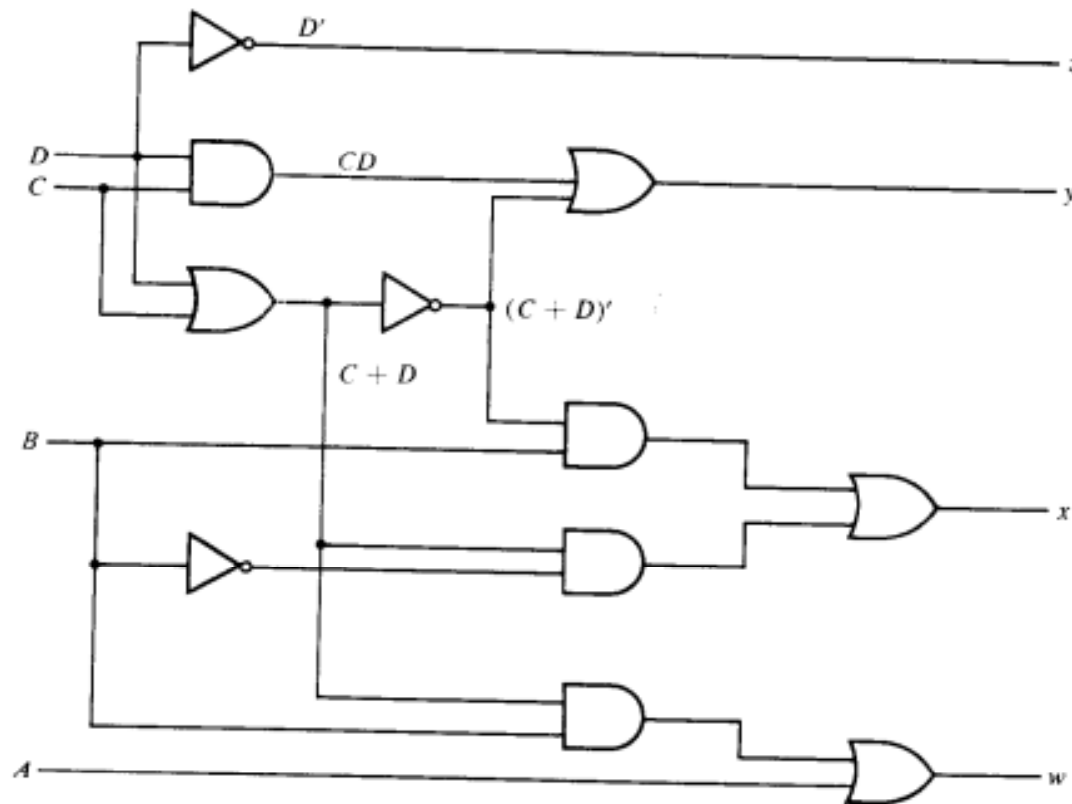


$$z = D'$$

$$y = CD + C'D' = CD + (C + D)'$$

$$x = B'C + B'D + BC'D' = B'(C + D) + BC'D'$$
$$= B'(C + D) + B(C + D)'$$

$$w = A + BC + BD = A + B(C + D)$$



¿Como verificamos que si es?

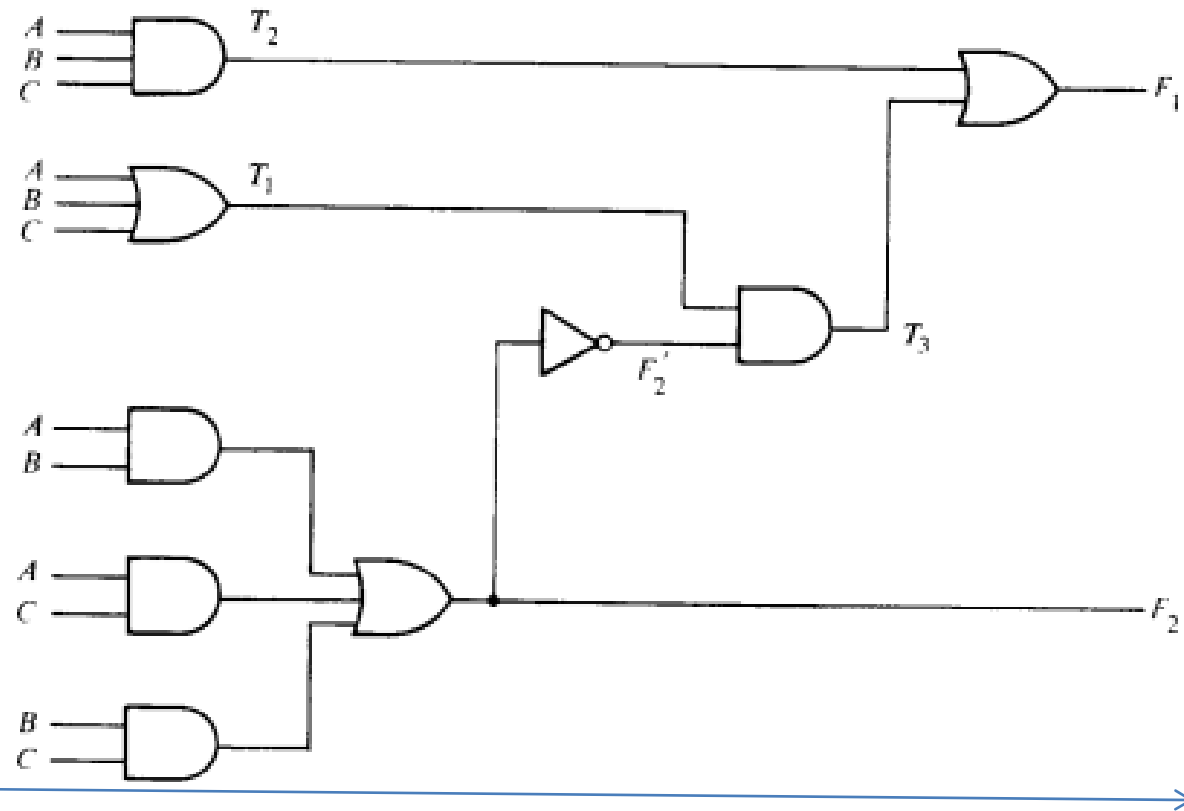
PROCEDIMIENTO DE ANALISIS

Para obtener las funciones de Boole de salida de un diagrama lógico, se procede de la siguiente manera:

1. Señálese con símbolos arbitrarios todas las salidas de las compuertas que son función de las variables de entrada. Obténgase las funciones de Boole para cada compuerta.
2. Márquese con otros símbolos arbitrarios aquellas compuertas que son una función de las variables de entrada y las compuertas marcadas anteriormente. Encuéntrese las funciones de Boole para ellas.
3. Repítase el proceso esbozado en el paso 2 hasta que se obtengan las salidas del circuito.
4. Obténgase las funciones de Boole de salida en términos de las variables de entrada solamente, por sustitución repetida de las funciones definidas anteriormente.

Ejemplo de análisis

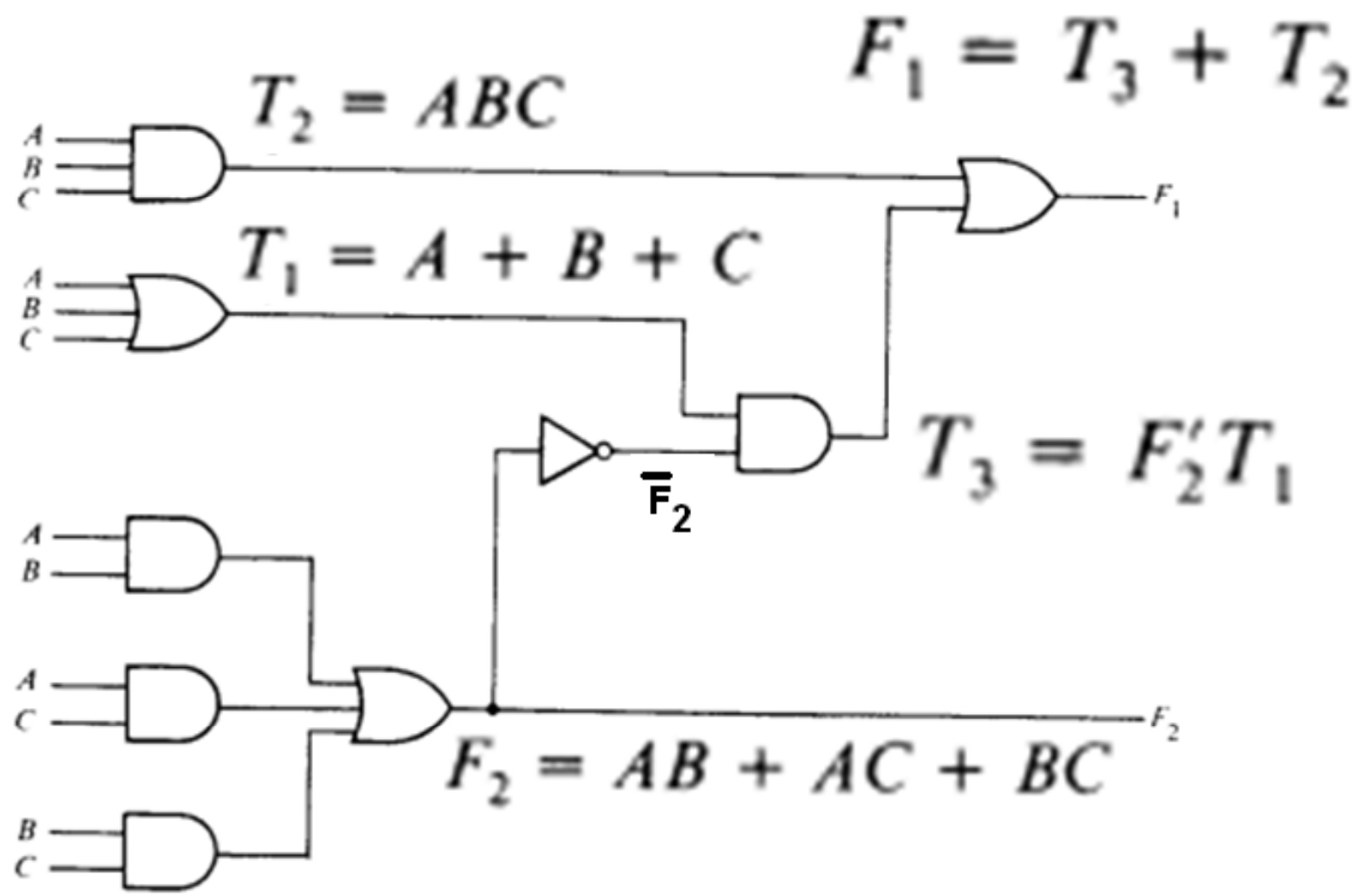
Dadas las compuertas



$$F_2 = AB + AC + BC$$

$$T_1 = A + B + C$$

$$T_2 = ABC$$



$$F_2 = AB + AC + BC$$

$$T_1 = A + B + C$$

$$T_2 = ABC$$

$$T_3 = F_2' T_1$$

$$F_1 = T_3 + T_2$$

Sustituimos los datos:

$$\begin{aligned} F_1 &= T_3 + T_2 = F_2' T_1 + ABC = (AB + AC + BC)'(A + B + C) + ABC \\ &= (A' + B')(A' + C')(B' + C')(A + B + C) + ABC \\ &= (A' + B'C')(AB' + AC' + BC' + B'C) + ABC \\ &= A'BC' + A'B'C + AB'C' + ABC \end{aligned}$$

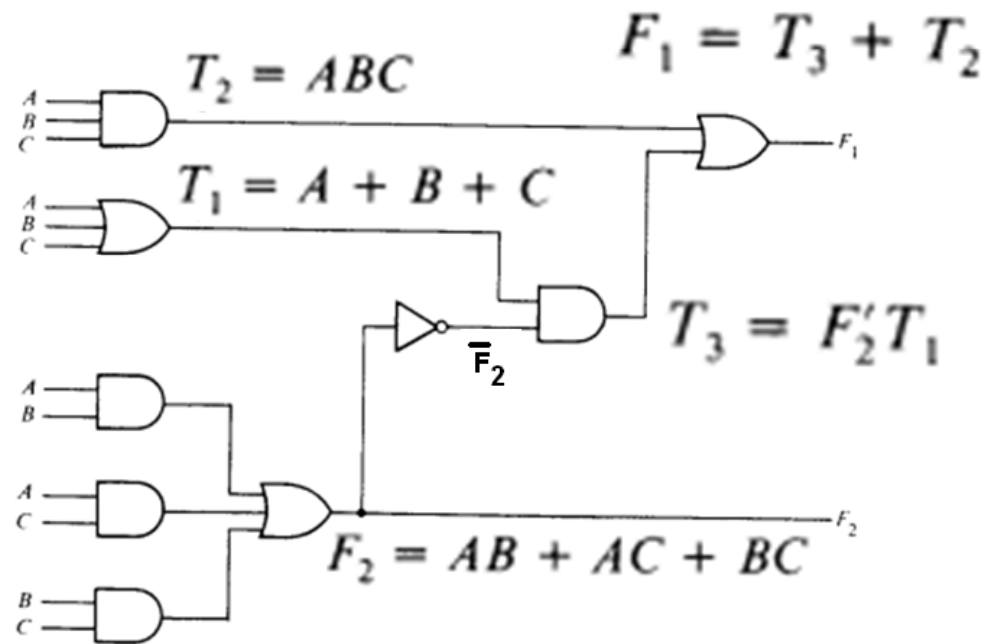
¿Como sacar la tabla de verdad?

¿Cuántas entradas existen sin repetir variables?

1. **Determinése el número de variables de entrada del circuito. Para n entradas, fórmese las 2^n posibles combinaciones de entrada de unos y ceros listando los números binarios desde 0 hasta $2^n - 1$.**
2. **Márquese las salidas de las compuertas seleccionadas con símbolos arbitrarios.**
3. **Obténgase la tabla de verdad para las salidas de aquellas compuertas que son una función de las variables de entrada solamente.**
4. **Procédase a obtener la tabla de verdad para las salidas de aquellas compuertas que son una función de los valores definidos previamente hasta que se determinen las columnas para todas las salidas.**

¿Cuántas entradas existen sin repetir variables?

<i>A</i>	<i>B</i>	<i>C</i>	<i>AB</i>	<i>AC</i>	<i>BC</i>	<i>F₂</i>	<i>F₂'</i>	<i>T₁</i>	<i>T₂</i>	<i>T₃</i>	<i>F₁</i>
0	0	0	0	0	0	0	1	0	0	0	0
0	0	1	0	0	0	0	1	1	0	1	1
0	1	0	0	0	0	0	1	1	0	1	1
0	1	1	0	0	1	1	0	1	0	0	0
1	0	0	0	0	0	0	1	1	0	1	1
1	0	1	0	1	0	1	0	1	0	0	0
1	1	0	1	0	0	1	0	1	0	0	0
1	1	1	1	1	1	1	0	1	1	0	1



Comparador de Magnitudes

es un circuito combinacional que compara dos números, A y B y determina sus magnitudes relativas. El resultado de la comparación se especifica por medio de tres variables binarias que indican cuando $A > B$, $A = B$, ó $A < B$.

$$A = A_3A_2A_1A_0$$

$$B = B_3B_2B_1B_0$$

dos números son iguales si todos los pares de números significativos son iguales, es decir si $A_3 = B_3$ y $A_2 = B_2$ y $A_1 = B_1$ y $A_0 = B_0$.

A	B	$A = B$	$A > B$	$A < B$
0	0	0	0	0
0	1	0	0	1
1	0	0	1	0
1	1	1	0	0

Multiplicadores

La multiplicación de dos números binarios se realiza con lápiz y papel efectuando adiciones sucesivas y acarreos. Para ilustrar lo anterior:

1011	multiplicando (11)
× 1101	multiplicador (13)
1011	
0000	
1011	
1011	
<hr/>	
10001111	producto (143)

Este proceso consiste en examinar los bits sucesivos del multiplicador, empezando con el LSB. Si el bit multiplicador es un 1, el multiplicando se copia; si se trata de un 0, se escriben ceros.

multiplicando:

1011

multiplicador:

1101

1011

LSB del multiplicador = 1; escriba el multiplicando; corra el multiplicando una posición a la izquierda (10110).

1011

segundo bit multiplicador = 0; escriba el resultado anterior; corra el multiplicando a la izquierda de nuevo (101100)

+ 101100

tercer bit multiplicador = 1; escriba el multiplicando (101100)

110111

sume; corra de nuevo el multiplicando a la izquierda (1011000)

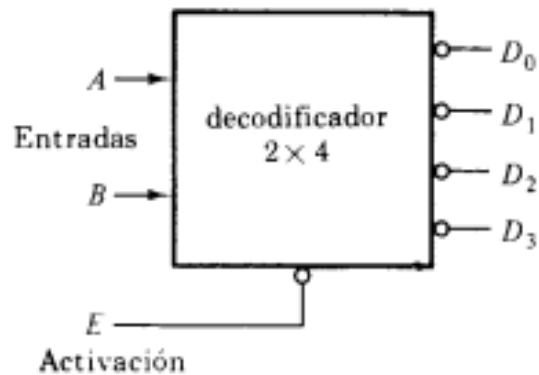
+ 1011000

cuarto bit multiplicador = 1; escriba el nuevo multiplicando (1011000)

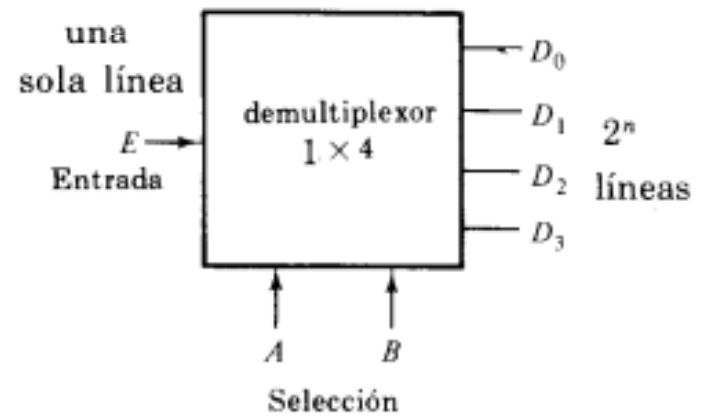
10001111

sume para obtener el producto final

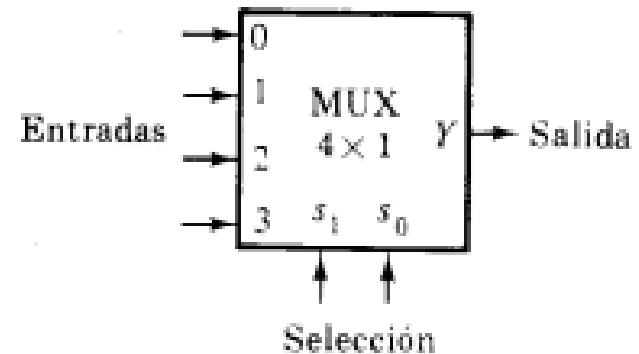
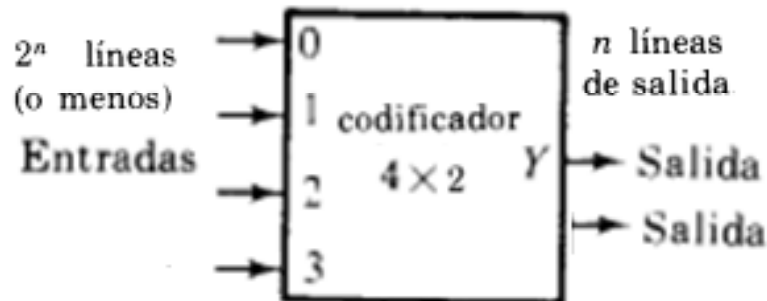
Bloques Combinacionales

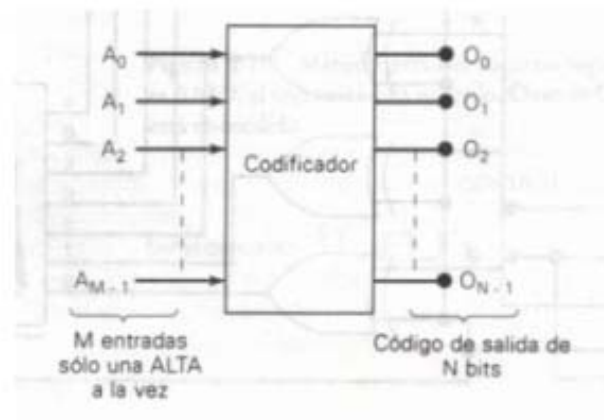
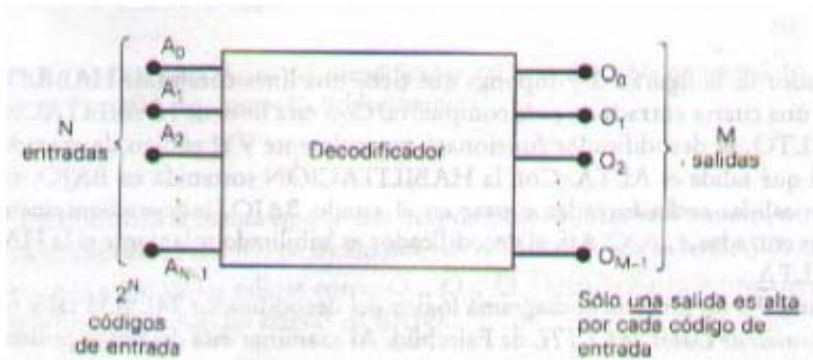
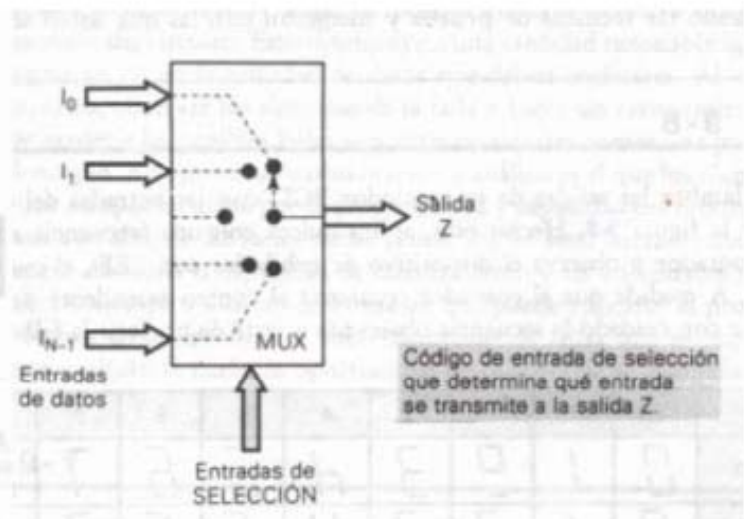
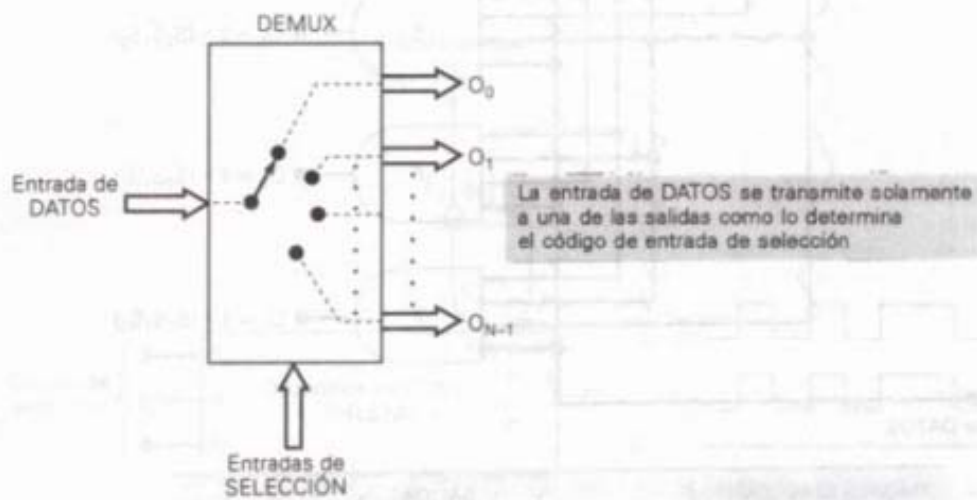


(a) Decodificador con activador



(b) Demultiplexor





Deco/codif

Tabla de verdad del decodificador de línea de 3 a 8

Entradas			Salidas							
<i>x</i>	<i>y</i>	<i>z</i>	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

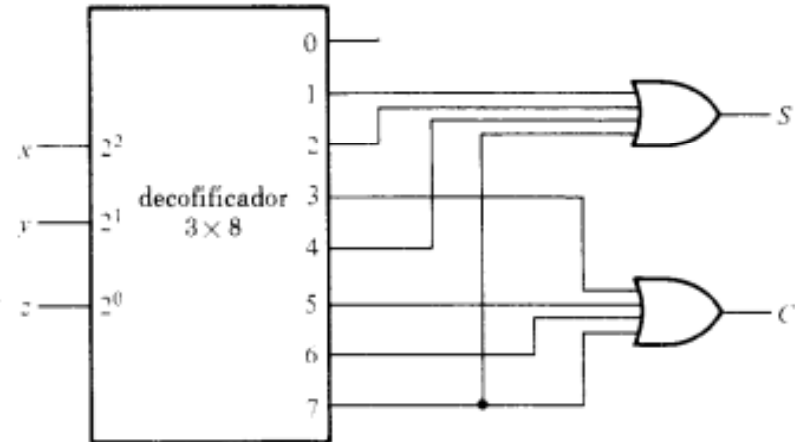


Tabla de verdad de codificador octal a binario

Entradas								Salidas		
D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	<i>x</i>	<i>y</i>	<i>z</i>
1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	1	0	0	1	0	1
0	0	0	0	0	0	1	0	1	1	0
0	0	0	0	0	0	0	1	1	1	1

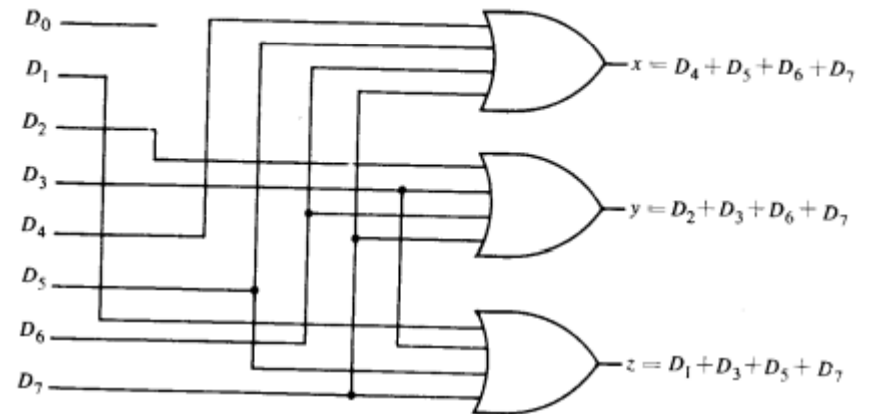
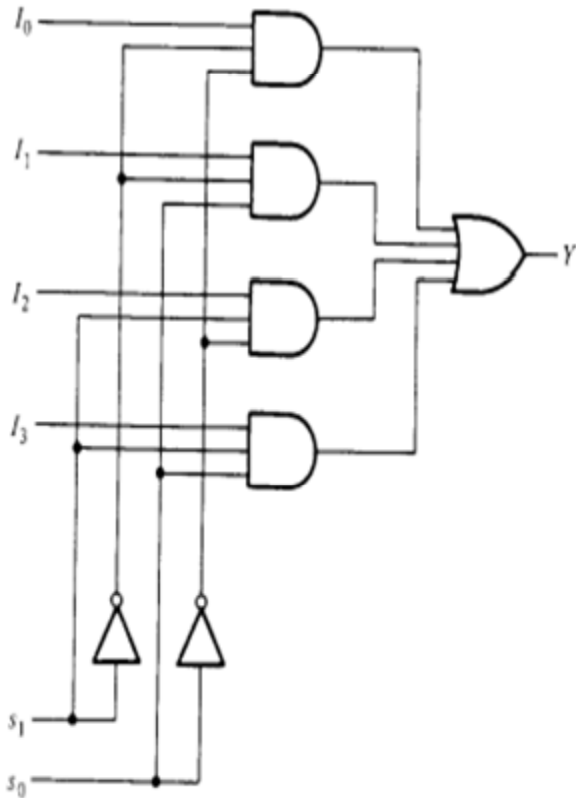


Figura 5-15 Codificador octal a binario

*Por ejemplo el CI tipo 74148.

Multiplexores



mux4to1.vhd - Text Editor

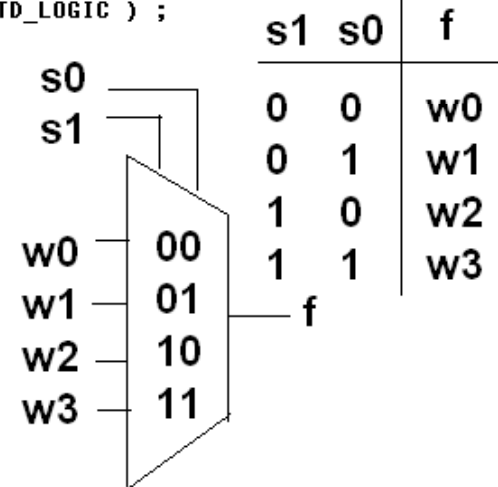
```

LIBRARY ieee ;
USE ieee.std_logic_1164.all ;

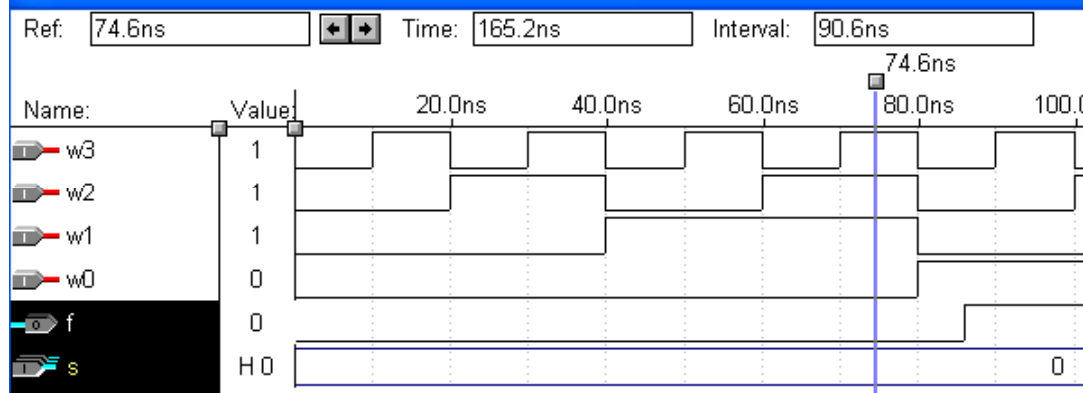
ENTITY mux4to1 IS
    PORT ( w0, w1, w2, w3 : IN    STD_LOGIC ;
          s                : IN    STD_LOGIC_VECTOR(1 DOWNTO 0) ;
          f                : OUT   STD_LOGIC ) ;
END mux4to1 ;

ARCHITECTURE Behavior OF mux4to1 IS
BEGIN
    WITH s SELECT
        f <=  w0 WHEN "00",
              w1 WHEN "01",
              w2 WHEN "10",
              w3 WHEN OTHERS ;
END Behavior ;

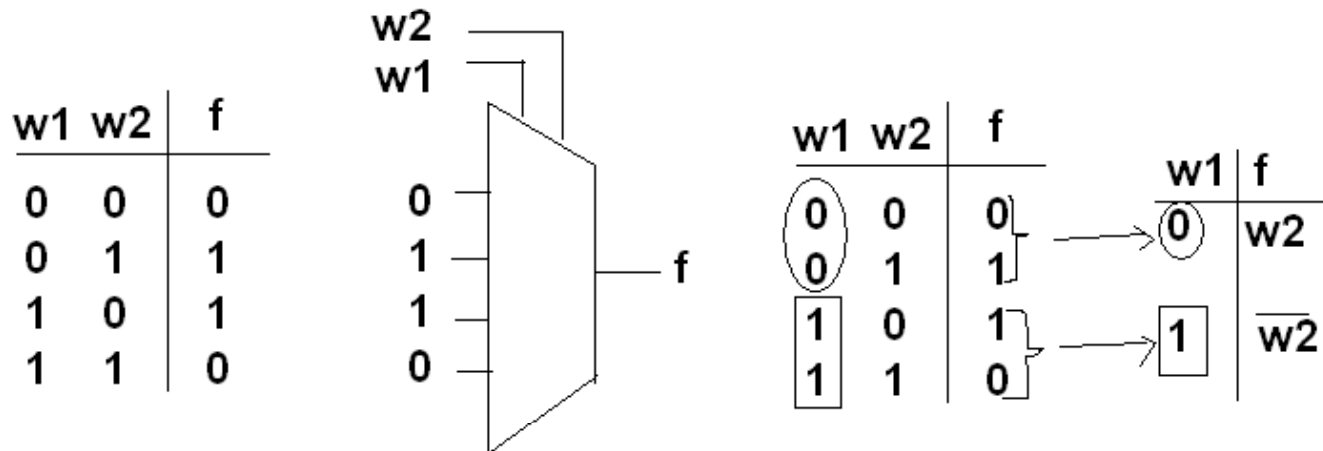
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;
PACKAGE mux4to1_package IS
    COMPONENT mux4to1
        PORT ( w0, w1, w2, w3 : IN    STD_LOGIC ;
              s                : IN    STD_LOGIC_VECTOR(1 DOWNTO 0) ;
              f                : OUT   STD_LOGIC ) ;
    END COMPONENT ;
END mux4to1_package ;
    
```



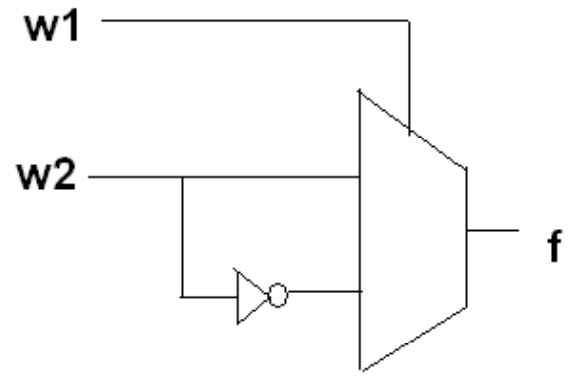
mux4to1.scf - Waveform Editor



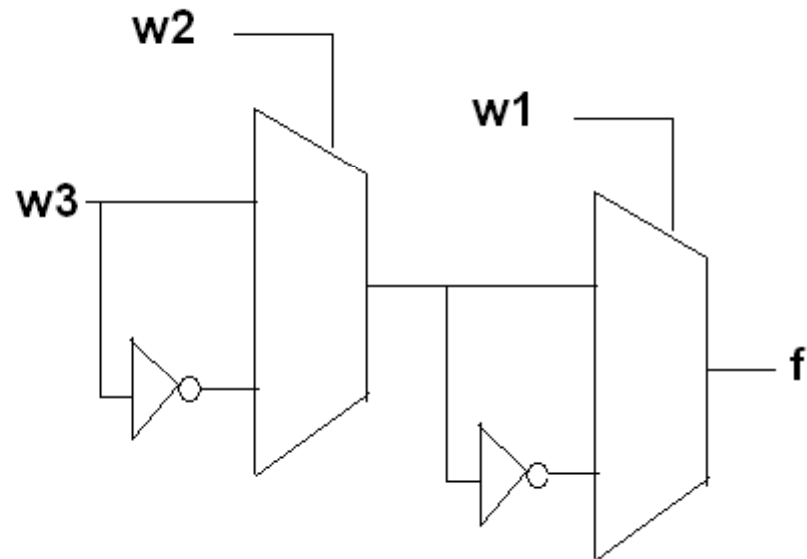
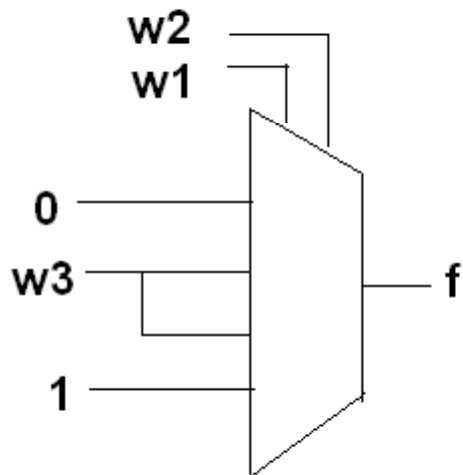
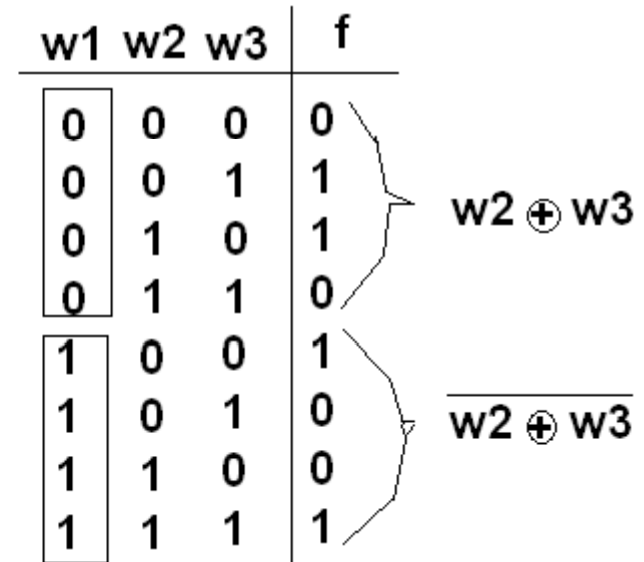
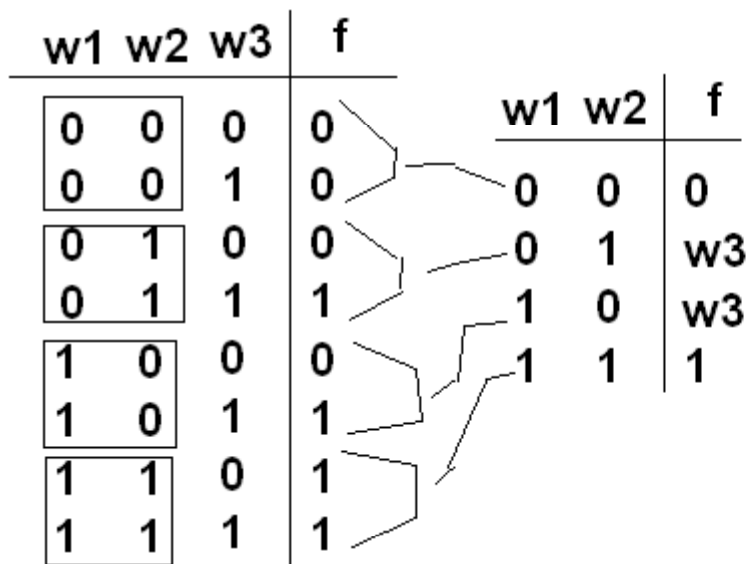
Síntesis con MUX



Se elige una de las dos entradas como entrada de selección(w1).
 Se escribe la tabla de verdad respecto a la entrada(w1). Obsérvese que cuando w1=0, w2 es igual que la salida f. y cuando w1=1 la salida respecto a w2 esta invertida.



Sintetizando funciones Booleanas..



Síntesis de multiplexores mediante la expansión de Shannon

- Teorema: cualquier función booleana $f(w_1, w_2, \dots, w_n)$ puede escribirse

$$\overline{w_1} * f(0, w_2, \dots, w_n) + w_1 * f(1, w_2, \dots, w_n)$$

Ejemplo:

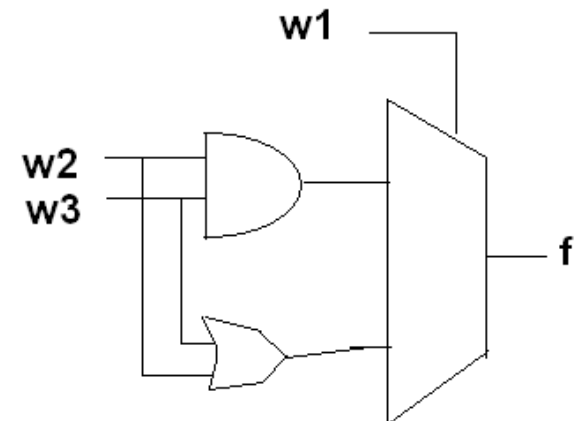
$$F(w_1, w_2, w_3) = w_1 w_2 + w_1 w_3 + w_2 w_3$$

En términos de w_1

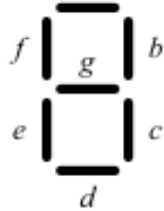
$$F = \overline{w_1}(w_2 w_3) + w_1(w_2 + w_3)$$

w1	w2	w3	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

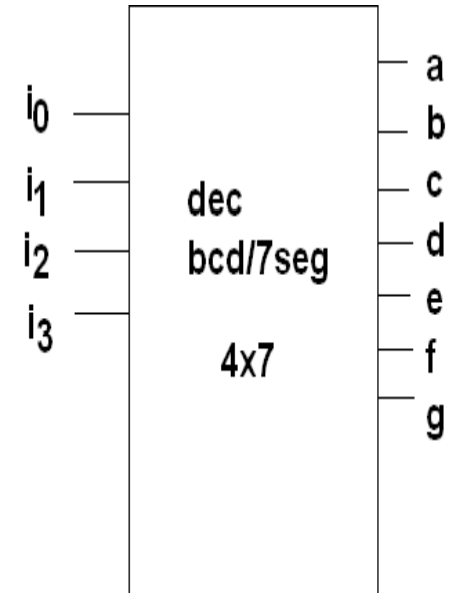
w1	f
0	$w_2 w_3$
1	$w_2 + w_3$



Decodificador BCD a 7 segmentos



Inputs				Decimal Digit	Display	a	b	c	d	e	f	g
i_3	i_2	i_1	i_0									
0	0	0	0	0		1	1	1	1	1	1	0
0	0	0	1	1		0	1	1	0	0	0	0
0	0	1	0	2		1	1	0	1	1	0	1
0	0	1	1	3		1	1	1	1	0	0	1
0	1	0	0	4		0	1	1	0	0	1	1
0	1	0	1	5		1	0	1	1	0	1	1
0	1	1	0	6		1	0	1	1	1	1	1
0	1	1	1	7		1	1	1	0	0	0	0
1	0	0	0	8		1	1	1	1	1	1	1
1	0	0	1	9		1	1	1	0	0	1	1
Rest of the Combinations						×	×	×	×	×	×	×



$$a = i_3'i_2'i_1'i_0' + i_3'i_2'i_1i_0' + i_3'i_2'i_1i_0 + i_3'i_2i_1'i_0 + i_3'i_2i_1i_0' + i_3'i_2i_1i_0 + i_3i_2'i_1'i_0' + i_3i_2'i_1'i_0$$