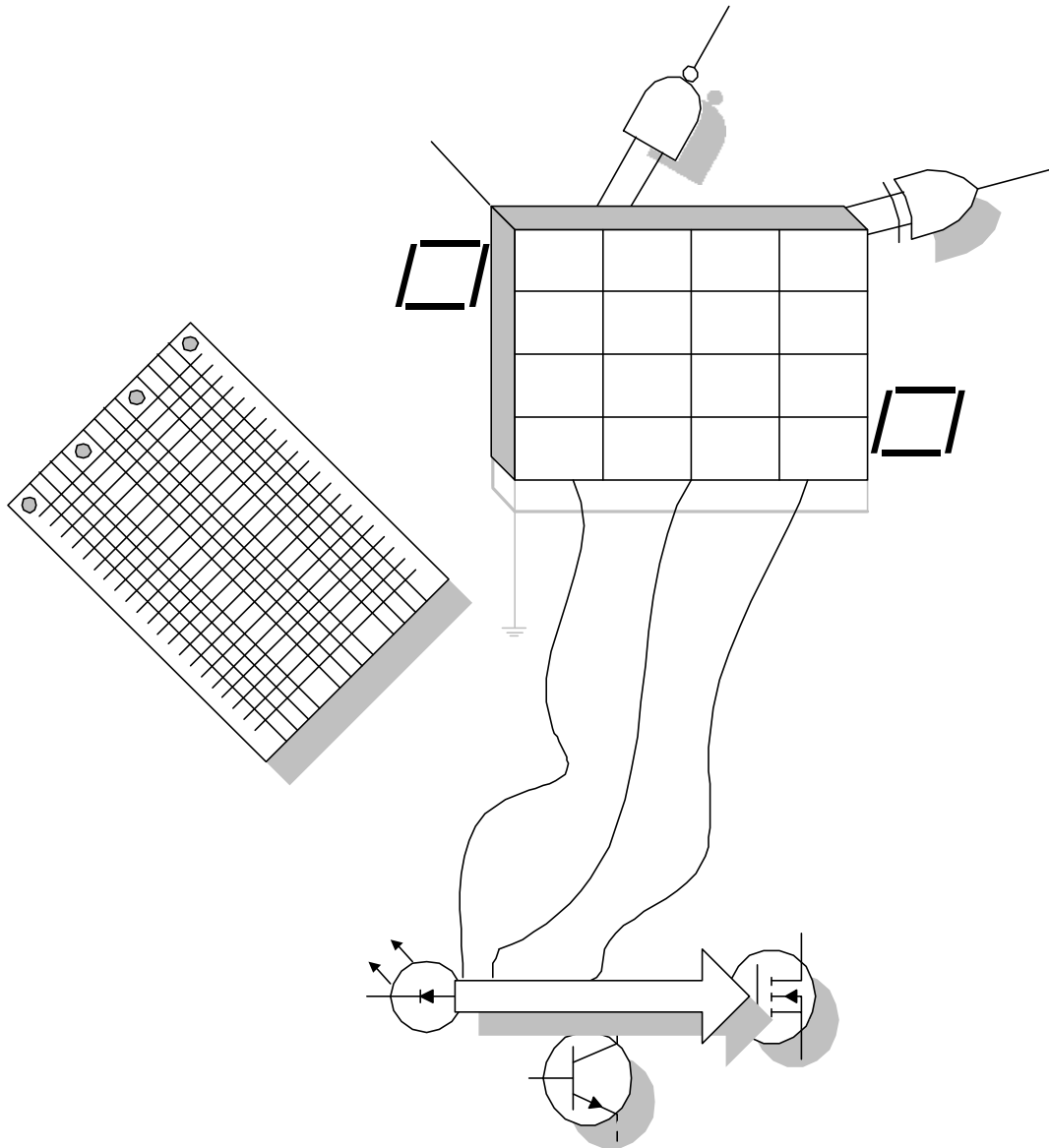


ELECTRÓNICA DIGITAL COMBINACIONAL

Diseño, Teoría y práctica



Angel Agustín Olivier

Mayo de 2002

ÍNDICE GENERAL

	Pág.
RESUMEN	vi
INTRODUCCIÓN	vii
CAPÍTULO	
I	
SISTEMAS Y CÓDIGOS DE NUMERACIÓN.	1
1.1 Sistemas numéricos de cualquier base.....	2
1.1.1 Sistema decimal.....	2
1.1.2 Sistema binario.....	2
1.1.3 Sistema octal.....	2
1.1.4 Sistema Hexadecimal.....	3
1.2 Transformación entre los sistemas numéricos.....	4
1.2.1 Número de base cualquiera a sistema decimal.....	4
1.2.2 Número decimal a cualquier base.....	6
1.2.3 Directas entre sistemas no decimales.....	9
1.3 Operaciones aritméticas de los distintos sistemas.....	15
1.3.1 Suma binaria, octal y hexadecimal.....	15
1.4 Complemento con respecto a la base del sistema.....	20
1.4.1 Disminuido en uno a la base del sistema.....	21
1.4.1.1 Complemento a uno.	
1.4.1.2 Complemento a dos.	
1.4.2 Operaciones aritméticas en complemento a dos.....	26
1.4.3 Representación numérica en coma fija y flotante.....	31
1.5 Códigos de numeración, alfanuméricos y de errores.....	35
1.5.1 Códigos numéricos.....	36
1.5.2 Códigos alfanuméricos.....	41
1.5.3 Códigos detectores y correctores de errores.....	44
II	
ÁLGEBRA DE BOOLE Y COMPUERTAS.	55
2.1 Teoremas y leyes del álgebra de Boole.....	55

CAPÍTULO	Pág.
2.2 Compuertas básicas y universales.....	61
2.3 Simplificación de funciones de conmutación.....	65
2.3.1 Formas canónicas de las funciones de conmutación.....	68
2.4 Diseño, simulación y síntesis de circuitos digitales.....	75
2.5 Aplicaciones de los circuitos digitales.....	82
PRÁCTICA DE LABORATORIO #1.....	87
III MÉTODOS DE SIMPLIFICACIÓN DE FUNCIONES LÓGICAS.	91
3.1 Minimización de funciones mediante mapas de Karnaugh.....	91
3.1.1 Simplificación de funciones aplicando mapas K.....	100
3.1.2 Términos y entradas indiferentes.....	104
3.2 Implicantes primos.....	106
3.3 Minimización mediante el método de Quine - McCluskey.....	110
3.4 Funciones multiterminales.....	116
3.5 Aplicaciones.....	119
IV CARACTERÍSTICAS INTERNAS DE LAS FAMILIAS LÓGICAS.	124
4.1 Parámetros eléctricos de un circuito digital.....	125
4.2 Lógica TTL.....	131
4.3 Lógica CMOS.....	154
4.4 Lógica ECL.....	175
PRÁCTICA DE LABORATORIO #2.....	182
PRÁCTICA DE LABORATORIO #3.....	187
V CIRCUITOS DIGITALES COMBINACIONALES MSI.	191
5.1 Decodificadores.....	191
5.1.1 Salidas y entradas en nivel bajo.....	193
5.1.2 Decodificadores integrados MSI.....	193
5.1.3 Aplicaciones de los decodificadores.....	196
PRÁCTICA DE LABORATORIO #4.....	206
5.2 Codificadores.....	210
5.2.1 Codificadores de prioridad.....	211

CAPÍTULO	Pág.
5.3 Multiplexores.....	216
5.3.1 Aplicaciones de los multiplexores.....	217
5.3.2 El multiplexor como generador de funciones lógicas.....	220
PRÁCTICA DE LABORATORIO #5.....	228
PRÁCTICA DE LABORATORIO #6.....	231
5.4 Circuitos digitales sumadores.....	236
5.4.1 Sumador completo de un bit.....	237
5.4.2 Sumador paralelo.....	239
5.4.3 Aplicaciones de los circuitos sumadores 7483 y 74182.....	244
PRÁCTICA DE LABORATORIO #7.....	251
5.5 Circuitos digitales comparadores.....	254
5.5.1 Circuito integrado comparador 7485.....	256
5.5.2 Aplicaciones de los circuitos comparadores.....	257
PRÁCTICA DE LABORATORIO #8.....	260
5.6 Circuitos generadores y detectores de paridad.....	263
5.6.1 Método de generación y chequeo de paridad de un bit.....	263
5.6.2 Generador y detector de paridad 74180 y 74280.....	265
5.6.3 Circuitos detectores y correctores Hamming.....	269
PRÁCTICA DE LABORATORIO #9.....	272
VI CIRCUITOS DIGITALES COMBINACIONALES VLSI.	276
6.1 Circuitos integrados de memoria ROM.....	280
6.2 Dispositivos lógicos programables (PLD).....	281
6.3 Arreglos lógicos programables combinacionales PAL y PLA.....	282
6.4 Arreglos de compuertas lógicas programables GAL	286
PRÁCTICA DE LABORATORIO #10.....	292
BIBLIOGRAFÍA	294
ANEXO	296

CAPÍTULO 1.

1 SISTEMAS Y CÓDIGOS DE NUMERACIÓN

INTRODUCCIÓN.

Una de las necesidades primordiales del hombre primitivo fue sin duda, la de contar y numerar objetos, utensilios, animales, plantas, etc. Esto lo solía hacer incrustando marcas y símbolos en madera y piedra. Primero, utilizó marcas o rayas para indicar las cantidades; por ejemplo, marcaba cinco rayas para señalar la caza de cinco animales. Sin embargo, con el tiempo, se dio cuenta de la necesidad de usar un método de numeración más compacto y resumido, eran demasiadas marcas para indicar grandes cantidades y por lo tanto la posibilidad de perder el control del conteo. Algunas tribus Suramericanas utilizan los dedos de las manos y pies para contar; de esta forma se repite la cuenta cada veinte veces (diez dedos de las manos y diez de los pies). Otro sistema de numeración son los números Romanos que utilizan los símbolos {I, V, X, L, C, D, M} para denotar las cantidades con valores posicionales y repetición máxima de tres símbolos consecutivos e iguales.

Los avances de la tecnología han creado la dependencia de los sistemas informáticos y de las computadoras; las cuales operan internamente con sistemas de numeración distintos a los conocidos por el hombre cotidiano. Sistemas numéricos de dos símbolos son suficientes para realizar diseños y modelos de circuitos digitales de computadoras. La electrónica digital es el resultado de la acción de variables discretas que pueden representarse e interpretarse, utilizando un sistemas con dos símbolos o dígitos 0 y 1 llamado binario; también se utilizan otros sistemas derivados de éste como lo son, el sistema de ocho símbolos octal y el sistema de 16 símbolos llamado hexadecimal. Estos últimos permiten representar números binarios de forma más pequeña.

1.1 Sistemas numéricos de cualquier base.

Las cantidades se caracterizan por tener dígitos enteros y fraccionarios, cada uno de estas poseen un valor dado por la cantidad de símbolos que maneja el sistema y otro valor que depende de la posición que ocupe el dígito en la cifra. Si a_j indica cualquier dígito de la cifra, b la base del sistema de numeración y además de esto la cantidad de dígitos enteros y fraccionarios son n y k respectivamente, entonces el número representado en cualquier base se puede expresar de la siguiente forma:

$$N_b = [a_{n-1}.a_{n-2}.a_{n-3}.....a_3.a_2.a_1.a_0.a_{-1}.a_{-2}.a_{-3}.....a_{-k}]_b$$

Donde: $j = \{n-1, n-2, \dots, 2, 1, 0, -1, -2, \dots, -k\}$ y $n + k$ indica la cantidad de dígitos de la cifra.

Por ejemplo, el número $31221, 32_4$ en base cuatro tiene $n=5$ y $k=2$ con la parte entera: $a_{n-1}=a_4=3$; $a_3=1$; $a_2=2$; $a_1=2$; $a_0=1$ y parte fraccionaria $a_{-1}=3$; $a_{-2}=2$

1.1.1 Sistema decimal.

Este es el sistema que manejamos cotidianamente, está formado por diez símbolos $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$ por lo tanto la base del sistema es diez (10).

1.1.2 Sistema binario.

Utiliza dos símbolos para representar las cantidades, estos son: el cero "0" y el uno "1"; la base del sistema es dos (2). Este sistema tiene aplicación directa en los circuitos de conmutación y compuertas lógicas digitales. También se le asocia niveles de tensión alta y baja respectivamente. Por lo general, se establecen relaciones de la siguiente forma: el nivel alto se puede denotar con las expresiones 1, High, True, verdadero; y el nivel bajo con 0, Low, False, falso.

1.1.3 Sistema octal.

El sistema numérico octal utiliza ocho símbolos o dígitos para representar cantidades y cifras numéricas. Los dígitos son: $\{0, 1, 2, 3, 4, 5, 6, 7\}$; la base de éste

es ocho (8) y es un sistema que se puede convertir directamente en binario como se verá más adelante.

1.1.4 Sistema hexadecimal.

El sistema numérico hexadecimal utiliza dieciséis dígitos y letras para representar cantidades y cifras numéricas. Los símbolos son: {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F}; la base del sistema es dieciséis (16). También se puede convertir directamente en binario como se verá más adelante. En la tabla 1.1 se muestran los primeros veintiuno números decimales con su respectiva equivalencia binaria, octal y hexadecimal.

DECIMAL	BINARIO	OCTAL	HEXADECIMAL
0	0000	0	0
1	0001	1	1
2	0010	2	2
3	0011	3	3
4	0100	4	4
5	0101	5	5
6	0110	6	6
7	0111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F
16	10000	20	10
17	10001	21	11
18	10010	22	12
19	10011	23	13
20	10100	24	14

Tabla 1.1. Equivalencia entre sistemas de los primeros veintiuno números decimales.

1.2 Transformación entre los sistemas numéricos.

Los sistemas numéricos pueden transformarse aplicando fórmulas ponderadas que establecen relaciones entre los mismos con respecto al sistema decimal que manejamos cotidianamente. También puede realizarse la operación inversa de transformar un número dado en cualquier base al sistema decimal. Las transformaciones que se manejan normalmente son binarias, octal y hexadecimal. Además, existen transformaciones directas como lo son: binario-octal y binario-hexadecimal. A continuación se explican dichas transformaciones.

1.2.1 Transformación de un número de base cualquiera a sistema decimal.

El valor de un número en el sistema decimal depende de los dígitos enteros y fraccionarios que posea, conjuntamente con el peso posicional de la base del sistema numérico dado. Por ejemplo, el número de base diez (decimal) $492,86_{10}$ tiene valores posicionales por cada dígito en correspondencia con el producto de la base de dicho sistema. En la figura 1.1 se observa que el valor ponderado no es más que la **suma de los productos** de los dígitos por la base elevada al exponente según corresponda la posición de dicho dígito.

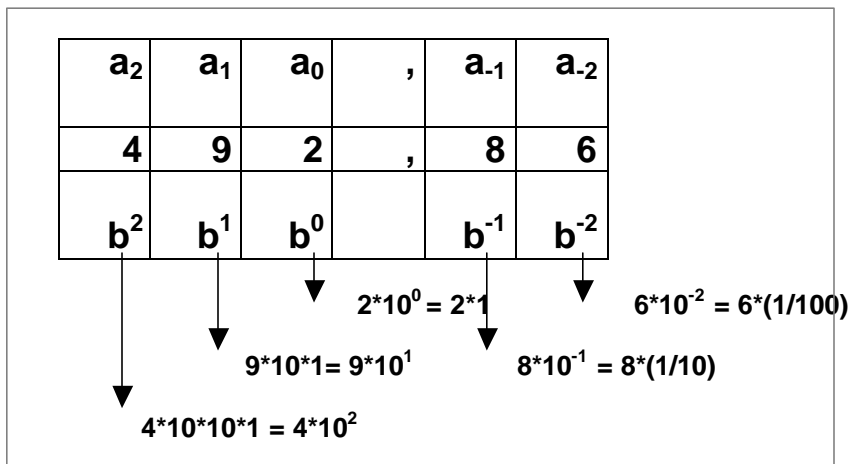


Figura 1.1. Valor ponderado de un número decimal.

La parte entera tiene un valor dado en unidades que se obtiene de la forma siguiente:

$$4 \cdot 10^2 \text{ unidades} + 9 \cdot 10^1 \text{ unidades} + 2 \cdot 10^0 \text{ unidades} = 492 \text{ unidades.}$$

La parte fraccionaria tiene un valor dado en centésimas que se obtiene de la forma siguiente:

$$8 \cdot 10^{-1} = \text{ocho décimas} = \text{ochenta centésimas.}$$

$$6 \cdot 10^{-2} = \text{seis centésimas.}$$

Total 86 centésimas.

La transformación al sistema decimal de un número dado en cualquier base se obtiene con la ecuación:

$$N_{10} = a_{n-1}b^{n-1} + a_{n-2}b^{n-2} + a_{n-3}b^{n-3} + a_{n-4}b^{n-4} + \dots + a_3b^3 + a_2b^2 + a_1b^1 + a_0b^0 + a_{-1}b^{-1} + a_{-2}b^{-2} + \dots + a_{-k}b^{-k} \quad (\text{EC 1.1})$$

$$N_{10} = \sum_{j=-k}^{n-1} a_j b^j \quad (\text{EC 1.1})$$

Donde **k** indica la cantidad de dígitos fraccionarios, **n** la cantidad de dígitos enteros, **a_j** el *i*ésimo dígito y **b** la base del sistema de numeración. Con **j** desde **-k** hasta **n-1**.

Ejemplo 1.1. Transformar a decimal el número binario 11011001,1101₂

Solución: La cantidad de dígitos enteros **n = 8**; la base del sistema **b = 2** y la cantidad de dígitos fraccionarios **k = 4**. Aplicando la ecuación 1.1 tenemos:

$$N_{10} = 1.(2)^7 + 1.(2)^6 + 0.(2)^5 + 1.(2)^4 + 1.(2)^3 + 0.(2)^2 + 0.(2)^1 + 1.(2)^0 + 1.(2)^{-1} + 1.(2)^{-2} + 0.(2)^{-3} + 1.(2)^{-4}$$

$$N_{10} = 128 + 64 + 0 + 16 + 8 + 0 + 0 + 1 + 0,5 + 0,25 + 0 + 0,0625$$

$$N_{10} = 217,8125_{10}$$

Ejemplo 1.2. Transformar a decimal el número hexadecimal 3F06A,AD₁₆

Solución: La cantidad de dígitos enteros **n = 5**; la base del sistema **b = 16** y la cantidad de dígitos fraccionarios **k = 2**. Aplicando la ecuación 1.1 tenemos:

$$N_{10} = 3.(16)^4 + F.(16)^3 + 0.(16)^2 + 6.(16)^1 + A.(16)^0 + A.(16)^{-1} + D.(16)^{-2}$$

$$N_{10} = 3.(16)^4 + 15.(16)^3 + 0.(16)^2 + 6.(16)^1 + 10.(16)^0 + 10.(16)^{-1} + 13.(16)^{-2}$$

$$N_{10} = 196608 + 61440 + 0 + 96 + 10 + 0,625 + 0,05078$$

$$N_{10} = 258154,6758_{10}$$

Ejemplo 1.3. Transformar a decimal el número octal 764321,367₈

Solución: La cantidad de dígitos enteros **n = 6**; la base del sistema **b = 8** y la cantidad de dígitos fraccionarios **k = 3**. Aplicando la ecuación 1.1 tenemos:

$$N_{10} = 7.(8)^5 + 6.(8)^4 + 4.(8)^3 + 3.(8)^2 + 2.(8)^1 + 1.(8)^0 + 3.(8)^{-1} + 6.(8)^{-2} + 7.(8)^{-3}$$

$$N_{10} = 229376 + 24576 + 2048 + 192 + 16 + 1 + 0,375 + 0,09375 + 0,013672$$

$$N_{10} = 256209,4824_{10}$$

1.2.2 Transformación de un número decimal a cualquier base.

Se puede hallar un procedimiento para transformar un número decimal en otro de base **b** con una cantidad **n** de dígitos enteros. Para lograr esto se parte de la **EC 1.1**, donde se obtienen particiones sucesivas hasta llegar a la partición **n**:

$$N_{10} = a_{n-1}.b^{n-1} + a_{n-2}.b^{n-2} + a_{n-3}.b^{n-3} + \dots + a_3.b^3 + a_2.b^2 + a_1.b + a_0$$

$$N_{10} = (a_{n-1}.b^{n-2} + a_{n-2}.b^{n-3} + a_{n-3}.b^{n-4} + \dots + a_3.b^2 + a_2.b + a_1).b + a_0 = (N_{10}^1).b + a_0$$

$$N_{10}^1 = (a_{n-1}.b^{n-3} + a_{n-2}.b^{n-4} + a_{n-3}.b^{n-5} + \dots + a_3.b + a_2).b + a_1 = (N_{10}^2).b + a_1$$

$$N_{10}^2 = (a_{n-1}.b^{n-4} + a_{n-2}.b^{n-5} + a_{n-3}.b^{n-6} + \dots + a_4.b + a_3).b + a_2 = (N_{10}^3).b + a_2$$

$$N_{10}^3 = (a_{n-1}.b^{n-5} + a_{n-2}.b^{n-6} + a_{n-3}.b^{n-7} + \dots + a_5.b + a_4).b + a_3 = (N_{10}^4).b + a_3$$

$$N_{10}^4 = (N_{10}^5).b + a_4$$

$$\cdot \quad \cdot \quad \cdot$$

$$\cdot \quad \cdot \quad \cdot$$

$$\cdot \quad \cdot \quad \cdot$$

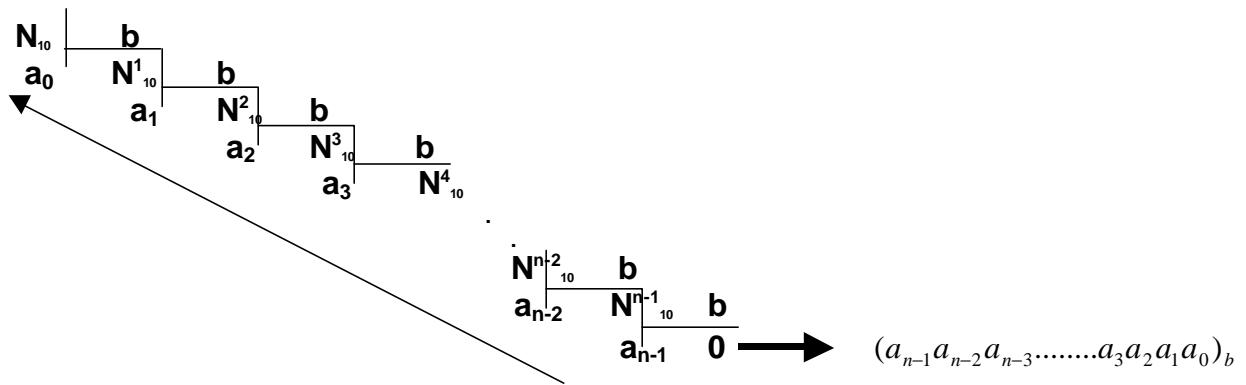
$$N_{10}^{n-1} = (N_{10}^n).b + a_{n-1} \quad \text{Enésima partición que se realiza con el número decimal } N_{10}.$$

El resultado de todo este procedimiento es una división sucesiva donde **b** es el divisor del número desconocido;

a_j : Son los residuos de la división con **j** desde **n-1** hasta **0**.

N_{10}^{n-1} : Es el dividendo, N_{10}^n : Es el cociente de las sucesivas particiones.

La cifra resultante de la transformación de un número decimal, en otro sistema numérico; se construye tomando los residuos en orden inverso a las divisiones o particiones sucesivas. Esto significa que la cifra será: $(a_{n-1}a_{n-2}a_{n-3}\dots\dots a_3a_2a_1a_0)_b$



La parte fraccionaria se transforma multiplicando esta última por la base del sistema y tomando como resultado el dígito entero que resulta del producto. Luego se resta el entero absoluto y el resultado se toma para la conversión; se repite de nuevo el procedimiento multiplicando por la base. En este tipo de conversión se debe limitar la cantidad de dígitos necesarios después de la coma.

$$\begin{aligned}
 (0, q_1 q_2 q_3 q_4 \dots)_{10} x b &= (a_{-1}, p_1 p_2 p_3 \dots) - a_{-1} = (0, p_1 p_2 p_3 \dots)_{10} \\
 (0, p_1 p_2 p_3 \dots)_{10} x b &= (a_{-2}, r_1 r_2 r_3 \dots) - a_{-2} = (0, r_1 r_2 r_3 \dots)_{10} \\
 (0, r_1 r_2 r_3 \dots)_{10} x b &= (a_{-3}, s_1 s_2 s_3 \dots) - a_{-3} = (0, s_1 s_2 s_3 \dots)_{10} \\
 &\vdots \\
 &\vdots \\
 (0, z_1 z_2 z_3 \dots)_{10} x b &= (a_{-k}, z_1 z_2 z_3 \dots) - a_{-k}
 \end{aligned}$$

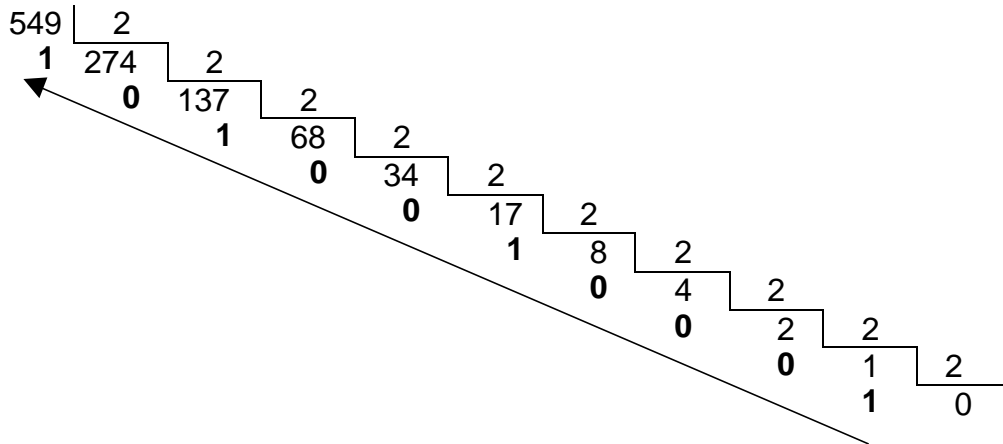
$$= 0, a_{-1} a_{-2} a_{-3} a_{-4} a_{-5} \dots a_{-k} \rightarrow \text{Conversión} \rightarrow \text{fraccionaria.}$$

La transformación completa de la parte entera y la fraccionaria da como resultado la cifra de base cualquiera y tiene la siguiente forma:

$$(a_{n-1}. a_{n-2}. a_{n-3} \dots a_3. a_2. a_1. a_0, a_{-1}. a_{-2}. a_{-3} \dots a_{-k})_b$$

Ejemplo 1.4. Transformar el número $549,28_{10}$ en: a) binario, b) octal y c) hexadecimal respectivamente; con tres dígitos significativos.

Solución a: El problema se resuelve en dos partes; primero convertimos la parte entera y luego la parte fraccionaria.



Parte entera: Se toman los dígitos binarios desde el último residuo hacia el primero en la dirección que indica la flecha.

$$549_{10} = 1000100101_2$$

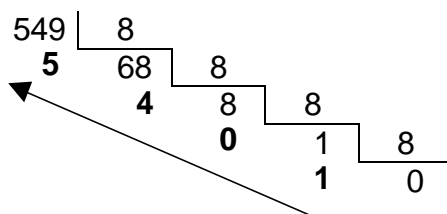
$$0,28 \times 2 = 0,56 - 0 = 0,56 \longrightarrow a_{-1} = 0$$

$$0,56 \times 2 = 1,12 - 1 = 0,12 \longrightarrow a_{-2} = 1$$

$$0,12 \times 2 = 0,24 - 0 = 0,24 \longrightarrow a_{-3} = 0$$

La Parte entera, más la parte fraccionaria, da como resultado: $549,28_{10} = 1000100101,010_2$

Solución b:



Parte entera: Se toman los dígitos binarios desde el último residuo hacia el primero en la dirección que indica la flecha.

$$549_{10} = 1045_8$$

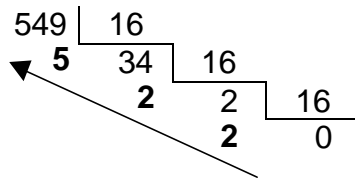
$$0,28 \times 8 = 2,24 - 2 = 0,24 \longrightarrow a_{.1} = 2$$

$$0,24 \times 8 = 0,48 - 0 = 0,48 \longrightarrow a_{.2} = 1$$

$$0,48 \times 8 = 3,84 - 3 = 0,84 \longrightarrow a_{.3} = 3$$

La Parte entera más la parte fraccionaria da como resultado: $549,28_{10} = 1045,213_8$

Solución c:



Parte entera: Se toman los dígitos binarios desde el último residuo hacia el primero en la dirección que indica la flecha.

$$549_{10} = 225_{16}$$

$$0,28 \times 16 = 4,88 - 4 = 0,88 \longrightarrow a_{.1} = 4$$

$$0,88 \times 16 = 14,08 - 14 = 0,08 \longrightarrow a_{.2} = E$$

$$0,08 \times 16 = 1,28 - 1 = 0,28 \longrightarrow a_{.3} = 1$$

La Parte entera más la parte fraccionaria da como resultado: $549,28_{10} = 225,4E1_{16}$

1.2.3 Transformaciones directas entre sistemas octal y hexadecimal.

Las conversiones directas de éstos sistemas, sin pasar por el sistema decimal, se fundamentan en la equivalencia que existe entre la base del sistema binario con respecto a la base del sistema octal y hexadecimal. Las equivalencias se realizan en grupos de dígitos binarios (bits), por ejemplo, para el sistema octal se necesitan tres bits y para el sistema hexadecimal se requieren cuatro bits.

Binario \longleftrightarrow **Octal**: Se hacen las conversiones con grupos de 3 bits. Esto se debe a la equivalencia matemática de la base binaria y octal $2^X = 2^3 = 8$; por lo tanto $X=3$. Los grupos se toman de dos formas: partiendo de la coma, hacia la izquierda, para la parte entera y de este mismo sitio, hacia la derecha, para la parte fraccionaria.

Ejemplo 1.5 Convertir al sistema binario los siguientes números dados en octal:

$$54721,44_8; \quad 263014,021_8$$

Solución:

5	4	7	2	1	,	4	4
101	100	111	010	001	,	100	100

Sentido de la conversión

$$54721,44_8 = 101\ 100\ 111\ 010\ 001,100\ 100_2$$

2	6	3	0	1	4	,	0	2	1
010	110	011	000	001	100	,	000	010	001

$$263014,021_8 = 10\ 110\ 011\ 000\ 001\ 100,000\ 010\ 001_2$$

Binario \longleftrightarrow **Hexadecimal**: Se hacen las conversiones con grupos de 4 bits. Esto se debe a la equivalencia matemática de la base binaria y hexadecimal $2^X = 2^4 = 16$; por lo tanto $X=4$. Los grupos se toman de dos formas: partiendo de la coma, hacia la izquierda, para la parte entera y de este mismo sitio, hacia la derecha, para la parte fraccionaria.

Ejemplo 1.6. Convertir al sistema binario los siguientes números dados en hexadecimal: $A5F729C, B7CD_{16}$; $3BC88A93FFF_{16}$

Solución:

A	5	F	7	2	9	C	,	B	7	C	D
1010	0101	1111	0111	0010	1001	1100	,	1011	0111	1100	1101

$$A5F729C, B7CD_{16} = 1010\ 0101\ 1111\ 0111\ 0010\ 1001\ 1100,1011\ 0111\ 1100\ 1101_2$$

3	B	C	8	8	A	9	3	F	F	F
0011	1011	1100	1000	1000	1010	1001	0011	1111	1111	1111

$$3BC88A93FFF_{16} = 11\ 1011\ 1100\ 1000\ 1000\ 1010\ 1001\ 0011\ 1111\ 1111\ 1111_2$$

Del mismo modo se realizan las transformaciones inversas.

Ejemplo 1.7. Realizar las transformaciones a los sistemas octal y hexadecimal de los siguientes números binarios:

$$a = 1110\ 1010\ 1101\ 0111\ 0000\ 10111, 0001101_2;$$

$$b = 11\ 0010\ 1110\ 0101\ 0101\ 1101\ 0101, 11111_2;$$

$$c = 11\ 0000\ 1010\ 1100\ 1010\ 1111\ 1011\ 1100\ 1101\ 1010\ 1100_2.$$

Solución (a):

001	110	101	011	010	111	000	010	111	,	000	110	100
1	6	5	3	2	7	0	2	7	,	0	6	4

$$1\ 110\ 101\ 011\ 010\ 111\ 000\ 010\ 111, 000\ 110\ 100_2 = 165327027,064_8$$

0001	1101	0101	1010	1110	0001	0111	,	0001	1010
1	D	5	A	E	1	7	,	1	A

$$1\ 1101\ 0101\ 1010\ 1110\ 0001\ 0111, 0001\ 1010_2 = 1D5AE17,1A_{16}$$

Solución (b):

011	001	011	100	101	010	111	010	101	,	111	110
3	1	3	4	5	2	7	2	5	,	7	6

$$11\ 001\ 011\ 100\ 101\ 010\ 111\ 010\ 101, 111\ 11_2 = 313452725,76_8$$

0011	0010	1110	0101	0101	1101	0101	,	1111	1000
3	2	E	5	5	D	5	,	F	8

11 0010 1110 0101 0101 1101 0101,1111 $1_2 = 32E55D5,F8_{16}$

Solución (c):

110	000	101	011	001	010	111	110	111	100	110	110	101	100
6	0	5	3	1	2	7	6	7	4	6	6	5	4

110 000 101 011 001 010 111 110 111 100 110 110 101 100 $_2 = 60531276746654_8$

0011	0000	1010	1100	1010	1111	1011	1100	1101	1010	1100
3	0	A	C	A	F	B	C	D	A	C

11 0000 1010 1100 1010 1111 1011 1100 1101 1010 1100 $_2 = 30ACAFBCDAC_{16}$

Octal \longleftrightarrow **Hexadecimal**: Este tipo de transformación debe ser realizada con un paso previo de conversión binaria. Luego, se pasa del sistema binario al correspondiente octal tomando grupos de tres bits, o se transforma a hexadecimal formando grupos de cuatro bits. La parte entera se agrupa desde la coma hacia la izquierda y la parte fraccionaria desde la coma hacia la derecha, de ser necesario, se rellena con cero en la última posición menos significativa de la cifra.

Ejemplo 1.8. Realizar las transformaciones hexadecimal y octal de los siguientes números:

$a=45674012,3_8 \longrightarrow$ hexadecimal; $b=8F42ABC,D07_{16} \longrightarrow$ octal;

Solución (a): El grupo hexadecimal fraccionario se debe completar con cero.

4	5	6	7	4	0	1	2	,	3 ₈
100	101	110	111	100	000	001	010	,	011 ₂

1001	0111	0111	1000	0000	1010	,	0110 ₂
9	7	7	8	0	A	,	6 ₁₆

$$45674012,3_8 = 97780,6_{16}$$

Solución (b):

8	F	4	2	A	B	C	,	D	0	7 ₁₆
1000	1111	0100	0010	1010	1011	1100	,	1101	0000	0111 ₂

001	000	111	101	000	010	101	010	111	100	,	110	100	000	111 ₂
1	0	7	5	0	2	5	2	7	4	,	6	4	0	7 ₈

$$8F42ABC,D07_{16} = 1075025274,6407_8$$

Ejercicios propuestos 1.1

1.1.1 Transformar al sistema binario, octal y hexadecimal los siguientes números decimales:

- 8879,482₁₀
- 6824,81₁₀
- 4095₁₀
- 699,2₁₀
- 11011,01₁₀
- 2467,42₁₀
- 65468,932₁₀
- 2047,33₁₀
- 4456,2₁₀
- 28079,83₁₀
- 1000,55₁₀
- 789,19₁₀

1.1.2 Transformar al sistema decimal los siguientes números:

- 5A79,C8₁₆
- 6724,61₈
- 10010101,1₂
- 4ED,6F2₁₆
- 1111011,011₂
- 2467,423₁₆
- 1111000,001₂
- 10000,01₈
- 77425,26₈
- 5A79,C8₁₆
- 62666,03₈
- 1111000,001₂
- 10101110,11₂
- 13444,27₈
- 443221,77₈
- 9988,62₁₆
- 11001,1101₈
- 3FFFF₁₆
- ABCD,7F₁₆
- 111111,11₂
- ABCD,7F₁₆
- 28079,7₈
- 4ED,6F2₁₆
- 222457,3₈

1.1.3 Construir una secuencia numérica, desde cero hasta sesenta, equivalente con el sistema decimal.

Se deben tomar grupos de seis símbolos que correspondan con los siguientes: □, ∽, ⊙, ●; los valores posicionales son continuos y se incrementan de uno en uno. El equivalente decimal es el siguiente:

- Cero unidades.
- Una unidad.
- Dos unidades.
- Tres unidades.

1.1.4 Transformar al sistema requerido los siguientes números:

- 3FFCD,4AB2₁₆ —————> Octal
- 64202513₈ —————> Hexadecimal
- 1237650,771₈ —————> Hexadecimal
- 10001,101₁₆ —————> Octal
- 334156,2₈ —————> Hexadecimal
- ABCD6,2₁₆ —————> Octal

1.3 Operaciones aritméticas de los distintos sistemas.

Al igual que en el sistema decimal, también en otros sistemas de numeración, se pueden realizar operaciones aritméticas, tales como: suma, resta, multiplicación y división tomando como referencia la base del sistema dado.

1.3.1 Suma binaria, octal y hexadecimal.

En general, para realizar la suma se procede de la misma forma como se hace en el sistema decimal. Por ejemplo, si $a_{n-1}a_{n-2}.....a_2a_1a_0, a_{-1}a_{-2}....a_{-k}$ es un número dado en una base **b** y $h_{n-1}h_{n-2}.....h_2h_1h_0, h_{-1}h_{-2}....h_{-k}$ es otro dado en la misma base entonces la suma se debe realizar de la siguiente forma:

$$\begin{array}{cccccccc}
 a_{n-1} & a_{n-2} & \dots & a_1 & a_0, & a_{-1} & \dots & a_{-k} & + \\
 h_{n-1} & h_{n-2} & \dots & h_1 & h_0, & h_{-1} & \dots & h_{-k} & \\
 \hline
 (a_{n-1} + h_{n-1} + c_{n-2}) & (a_{n-2} + h_{n-2} + c_{n-3}) & \dots & (a_1 + h_1 + c_0) & (a_0 + h_0 + c_{-1}), & (a_{-1} + h_{-1} + c_{-2}), & \dots & (a_{-k} + h_{-k})
 \end{array}$$

Los dígitos $m_j=(a_j+h_j+c_{j-1})$ pertenecientes al resultado se forman sumando los dígitos de cada columna de los cosumandos, más el acarreo c_{j-1} que viene de la columna anterior. Cada unidad de acarreo tiene el mismo valor de la base del sistema, por ejemplo, en la suma binaria es dos, en octal ocho y en hexadecimal dieciséis. Por ejemplo, llevar 2 en hexadecimal significa que el acarreo es el doble de la base y vale exactamente 32; de este mismo modo, en binario equivale a 4 veces y 16 en octal. Los acarreos aparecen cuando las semisumas de las columnas superan la base del sistema numérico.

Suma binaria: Las operaciones de suma binaria se realizan de la siguiente forma;

$$\begin{array}{ccccccc}
 0 + & 0 + & 1 + & 1 + & 1 + & 1 + & 1 + \\
 \underline{0} & \underline{1} & \underline{0} & \underline{1} & 1 & 1 & 1 \\
 0 & 1 & 1 & 10 & \underline{1} & 1 & 1 \\
 & & & & 11 & \underline{1} & 1 \\
 & & & & & 100 & \underline{1} \\
 & & & & & & 101
 \end{array}$$

Multiplicación hexadecimal:

				5	F	F	A,	1 ₁₆	X
							D	2	C ₁₆
<hr/>									
				4	7	F	B	8	C
				B	F	F	4	2	+
				4	D	F	B	2	D
<hr/>									
				4	F	A	3	1	C
							A,	C ₁₆	

1.3.3 División binaria, octal y hexadecimal.

La operación aritmética de dividir se realiza del mismo modo que en el sistema numérico decimal.

División binaria:

1	1	0'	1'	1'	1'	0'	1' ₂	1	0	1 ₂	
-1	0	1						1	0	1	1
		1	1	1							
		-1	0	1							
			1	0	1						
			-1	0	1						
					0	0	1				

Residuo

División octal y hexadecimal: La división se efectúa del mismo modo que en el sistema decimal y se realiza directamente en la misma base del sistema octal o hexadecimal. Sin embargo, también se puede obtener previamente la conversión en binario y proceder, como en el caso anterior, a realizarla en binario; y después el resultado transformarlo de nuevo al sistema numérico original.

1.4 Complemento de un número con respecto a la base del sistema.

Las representaciones de los números en los distintos sistemas son hechas por convenciones y acuerdos. La finalidad de esto es buscar formas sencillas de manejar universalmente operaciones y representaciones numéricas, representar números fraccionarios, números negativos, etc. El complemento de un número sirve para normalizar y reglamentar las operaciones aritméticas con signo, de forma que puedan ser procesadas por los circuitos internos de una calculadora o computadora.

El complemento a la base de un número se define por la siguiente fórmula:

$N_b^C = b^n - N_b$ (**Ec.1.3**) donde N_b^C es el número complementado a la base b del sistema, n la cantidad de dígitos y N_b es el número dado.

Ejemplo 1.12. Hallar el complemento a diez del número 897324_{10}

Solución: El número está dado en el sistema decimal y la cantidad de dígitos es seis

$$N_{10}^C = 10^6 - 897324_{10} = 102676_{10}$$

Ejemplo 1.13. Hallar el complemento a dieciséis del número $A9EFC21_{16}$

Solución: El número está dado en el sistema hexadecimal y la cantidad de dígitos es siete.

$$N_{16}^C = 16^7 - A9EFC21_{16} = 10000000_{16} - A9EFC21_{16} = 56103DF_{16}$$

Ejemplo 1.14. Hallar el complemento a ocho del número 60472_8

Solución: El número está dado en el sistema octal y la cantidad de dígitos es cinco.

$$N_8^C = 8^5 - 60472_8 = 100000_8 - 60472_8 = 17306_8$$

Ejemplo 1.15. Hallar el complemento a dos del número 100111011101_2

Solución: El número está dado en el sistema binario y la cantidad de dígitos es doce.

$$N_2^C = 2^{12} - 100111011101_2 = 1000000000000_2 - 100111011101_2 = 011000100011_2$$

1.4.1 Complemento disminuido en uno a la base del sistema.

Existe otra forma de hallar el complemento a la base del sistema, ésta es, obteniendo el complemento disminuido a uno y luego sumando uno. Para obtener esta fórmula se procede con un artificio en la **Ec.1.3** de la siguiente forma:

$$N_b^C = (b^n - N_b) + 1 - 1 = [(b^n - 1) - N_b] + 1 \quad \text{(Ec.1.3.1)}. \text{ El valor } N_b^{C-1} = (b^n - 1) - N_b \quad \text{(Ec.1.4)}$$

se conoce como el complemento de la base disminuido a uno. También se le denomina **complemento a uno** del sistema numérico correspondiente y por lo tanto, para hallar el complemento a la base solamente se le debe sumar uno a la **(Ec.1.4)**.

1.4.1.1 Complemento disminuido a uno del sistema binario, octal y hexadecimal.

El complemento disminuido a uno se obtiene aplicando la **Ec.1.4** en cualquiera de los sistemas numéricos. La expresión **(bⁿ-1)** se debe usar como minuendo en el tope de la potencia **bⁿ menos uno**, lo que significa tener una cifra compuesta por los dígitos más significativos y de mayor valor del sistema numérico. Por ejemplo, para hallar el minuendo de 56437₈, en el sistema octal, se procede de la siguiente forma:

n=5; entonces 8⁵ -1=100000₈ -1=77777₈. Ahora, para hallar el complemento disminuido a uno se resta el número dado: $N_b^{C-1} = 77777_8 - 56437_8 = 21340_8$.

Ejemplo 1.16. Hallar el complemento disminuido a uno de los siguientes números:

a) 24BCA0F7₁₆; b) 10011101101₂; c) 1265730₈

Sol. (a): $N_{16}^{C-1} = (16^8 - 1) - 24BCA0F7_{16} = FFFFFFFF_{16} - 24BCA0F7_{16} = DB435F08_{16}$

Sol. (b): $N_2^{C-1} = (2^{11} - 1) - 10011101101_2 = 11111111111_2 - 10011101101_2 = 01100010010_2$

Sol. (c): $N_8^{C-1} = (8^7 - 1) - 1265730_8 = 7777777_8 - 1265730_8 = 6512047_8$

En cualquier sistema de numeración el complemento disminuido a uno se puede hallar con la fórmula resultante de la Ec.1, Ec.2 y Ec.3 de la siguiente forma:

$$[(b^n - 1) - N_b] = [(b - 1)(b - 1) \dots (b - 1)(b - 1) - (a_{n-1})(a_{n-2}) \dots (a_1)(a_0)] \quad \text{Donde cada } (b-1)$$

corresponde al dígito de mayor peso en el sistema de numeración de base **b**. Los **a_j** son los **n** dígitos del número que se va complementar, con **j=0,1,.....,n-2,n-1**. El complemento disminuido a uno se halla, en forma directa, de la siguiente manera:

$$N_b^{C-1} = [(b-1) - a_{n-1}][(b-1) - a_{n-2}] \dots [(b-1) - a_2][(b-1) - a_1][(b-1) - a_0] \quad (\text{Ec.1.4.1}).$$

Ejemplo 1.17. Hallar el complemento disminuido a uno de los siguientes números:

a) $FCBC40_{16}$; b) 101011011_2

Solución (a): $N_{16}^{C-1} = FFFFFFF_{16} - FCBC40_{16} = 0343BF_{16}$

Solución (b): $N_2^{C-1} = 111111111_2 - 101011011_2 = 010100100_2$

1.4.1.1 Complemento a uno.

Es un caso particular del complemento disminuido a uno de la base binaria, tiene muchas aplicaciones en los circuitos digitales y sistemas de computación. Sirven para representar tablas numéricas de cantidades positivas y negativas, invertir los estados de los bits que conforman el dato binario y es utilizado como paso previo para hallar el complemento a dos. De la **Ec.1.4** se puede determinar que el complemento a uno se obtiene invirtiendo el estado o nivel de los bits que conforman la cifra.

Ejemplo 1.18. Hallar el complemento a uno de los siguientes números binarios:

a) 110001010101111010_2 ; b) 101011010101_2

Solución (a): $N_2^{C-1} = 001110101010000101_2$

Solución (b): $N_2^{C-1} = 010100101010_2$

1.4.1.2 Complemento a dos.

Es un caso particular del complemento a la base del sistema binario, tiene muchas aplicaciones en los circuitos digitales y sistemas de computación. Sirven para representar tablas numéricas de cantidades positivas y negativas, invertir los estados de los bits que conforman el dato binario y realizar operaciones aritméticas con signo en el sistema binario. Con la **Ec.1.3** se puede determinar el complemento a dos de un número binario; no obstante, con la misma ecuación se puede hallar un método directo para obtener también el complemento a dos. Este método consiste en ir seleccionando y colocando de derecha a izquierda los dígitos binarios hasta conseguir el primer bit en uno, de allí en adelante se cambian de estado todos los bits restantes.

El otro método para hallar el complemento a dos consiste en obtener el complemento a uno de la cifra y luego sumarle uno; esto último está reflejado en la **(Ec.1.3.1)**.

Ejemplo 1.19. Hallar el complemento a dos de los siguientes números binarios:

a) 101100101010111_2 ; b) 10001101000100_2 ; c) 10111001110000_2

Aplicando el método con la **(Ec.2.1)**;

Solución (a): $N_2^C = 010011010101000_2 + 1 = 010011010101001_2$

Solución (b): $N_2^C = 01110010111011_2 + 1 = 01110010111100_2$

Solución (c): $N_2^C = 01000110001111_2 + 1 = 01000110010000_2$

También, se aplica un método directo (algoritmo), buscando de derecha a izquierda hasta conseguir el primer bit en uno; se escribe(n) el(los) cero(s) anteriores (si los hay); y en los bits restantes, se cambia el estado de los mismos.

a) 101100101010111_2
Solución (a): $N_2^C = 010011010101001_2$ (Se deja igual y los demás cambian)

Solución (b): b) 10001101000100_2 $N_2^C = 01110010111100_2$
 (Se deja igual y los demás cambian)

Solución (c): $N_2^C = 01000110010000_2$

1.4.1.2.1 Representación numérica en complemento a dos.

En el sistema binario, la forma más utilizada para representar los números enteros con signo es la de complemento a dos. Los circuitos microprocesadores poseen internamente unidades de procesamiento aritmético que trabajan bajo éste formato, el cual puede estar constituido por n bits múltiplos de la potencia de base dos. Por ejemplo, para representar los números positivos y negativos se definen datos con tamaño estándar: ocho bits, 16 bits, 32 bits, etc.

En este formato, el bit más significativo (MSB) del dato se utiliza para indicar el signo y los bits restantes representan la magnitud del número. En la figura 1.2 se puede

apreciar la representación del formato utilizado para 16 bits, donde el más significativo (B15) indica que el signo es negativo si vale uno o positivo si vale cero. Las cantidades positivas se encuentran en binario normal mientras que los números negativos están en complemento a dos, esto significa que estos últimos, se deben complementar para poder hallar su verdadero valor.

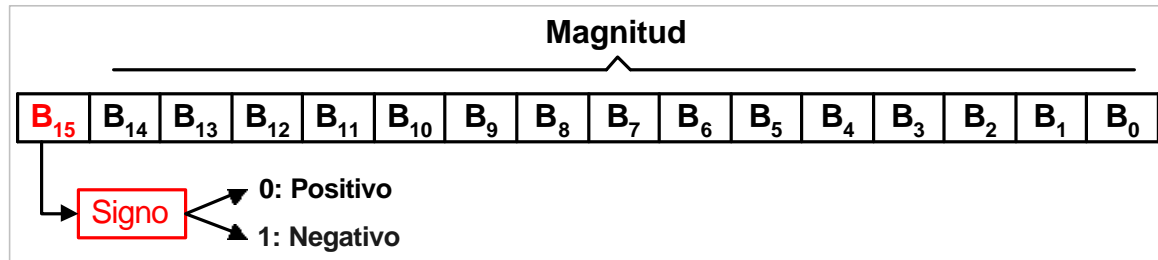


Figura 1.2. Formato de 16 bits para representación numérica con signo.

Número entero	Formato de 16 bits
+ 32767	0111111111111111
.	.
.	.
+5	000000000000101
+4	000000000000100
+3	000000000000011
+2	000000000000010
+1	000000000000001
0	000000000000000
-1	1111111111111111
-2	1111111111111110
-3	1111111111111101
-4	1111111111111100
-5	1111111111111011
.	.
.	.
-32767	100000000000001

P
O
S
I
T
I
V
O
S

B
N
O
R
M
A
L
I
O

N
E
G
A
T
I
V
O
S

C
A
D
O
S
C
O
M
P
L
E
M
E
N
T
O

Tabla 1.2. Representación de números enteros con 16 bits.

El complemento de un número, en éste formato, es igual que cambiar el signo del mismo. Por otra parte, el complemento del complemento da como resultado el mismo número. $N_2^C(N_2^C(X)) = X$

Ejemplo 1.20. Determinar el valor de los siguientes números dados en representación con signo de 16 bits (Formato de 16 bits):

- a) 1100101010111000_2 ; b) $7FA8_{16}$; c) 1111110000011100_2 ;
d) 176102_8 ; e) $FA8_{16}$;

Solución (a): El bit 15 del dato vale uno; esto significa que el número es negativo y está dado en complemento a dos. Primero se debe complementar el dato para hallar su verdadero valor en binario y después se transforma a decimal.

$$N_2^C = 0011010101001000_2 = -14640_{10}$$

Solución (b): Se debe transformar hexadecimal a binario y completar con ceros a la izquierda en caso de que el dato no tenga los 16 bits completos. Luego se hace la transformación a decimal.

$$7FA8_{16} = 0111111110101000_2 = +32680_{10}$$

Solución (c): El bit 15 del dato vale uno; esto significa que el número es negativo y está dado en complemento a dos. Primero se debe complementar el dato para hallar su verdadero valor en binario y después se transforma a decimal.

$$N_2^C = 0000001111100100_2 = -996_{10}$$

Solución (d): Se debe transformar octal a binario y completar con ceros a la izquierda en caso de que el dato no tenga los 16 bits completos. Luego se hace la transformación a decimal.

$$176102_8 = 1111110001000010_2$$

$$N_2^C = 0000001110111110_2 = -958_{10}$$

Solución (e): Se debe transformar hexadecimal a binario y completar con ceros a la izquierda en caso de que el dato no tenga los 16 bits completos. Luego se hace la transformación a decimal.

$$FA8_{16} = 111110101000_2 = 0000111110101000_2 = +4008_{10}$$

1.4.2. Operaciones aritméticas en complemento a dos.

La suma y resta son las operaciones básicas realizadas por los microprocesadores, cualquiera otra operación, es consecuencia **recursiva** de éstas. A continuación se describen estas dos operaciones aritméticas, realizadas con números binarios en complemento a dos utilizando formato de signo y magnitud de 16 bits.

1.4.2.1 Suma en complemento a dos.

Son cuatro casos que se presentan al sumar dos datos en formato con signo de complemento a dos:

I) Suma de dos números positivos. El resultado debe ser positivo, y el bit más significativo de la suma, siempre dará cero.

$$A = 100011111000100_2; \quad B = 10010110111011_2$$

$$\begin{array}{r} 0\ 1\ 0\ 0\ 0\ 1\ 1\ 1\ 1\ 1\ 0\ 0\ 0\ 1\ 0\ 0_2\ + \\ 0\ 0\ 1\ 0\ 0\ 1\ 0\ 1\ 1\ 0\ 1\ 1\ 1\ 0\ 1\ 1_2 \\ \hline 0\ 1\ 1\ 0\ 1\ 1\ 0\ 1\ 0\ 1\ 1\ 1\ 1\ 1\ 1\ 1_2 \end{array}$$

↳ Acarreo del 16vo bit = 0; A>0; B>0

Antes de realizar la suma binaria se debe tener la precaución de sumar en decimal los números. De esta manera se puede chequear el resultado de la suma para tener la certeza de que no exceda el valor $+32767_{10}$ y por lo tanto no sobrepasar el formato de 16 bits (Esto se conoce como OVERFLOW). También el 16vo bit en uno señala el sobreflujo de la operación.

II) Suma de uno negativo y otro positivo. El resultado debe poseer el signo del que tenga mayor valor absoluto. En este caso el resultado es positivo y el 16vo bit vale cero.

$$A = 1101011001010110_2; \quad B = 110110110111011_2$$

$$\begin{array}{r} 1\ 1\ 0\ 1\ 0\ 1\ 1\ 0\ 0\ 1\ 0\ 1\ 0\ 1\ 1\ 0_2\ + \\ 0\ 1\ 1\ 0\ 1\ 1\ 0\ 1\ 1\ 0\ 1\ 1\ 1\ 0\ 1\ 1_2 \\ \hline 1\ 0\ 1\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 1_2 \end{array}$$

↳ Acarreo del 16vo bit = 0; A<0; B>0

III) Suma de uno positivo y otro negativo. El resultado debe poseer el signo del que tenga mayor valor absoluto. En este caso el resultado es negativo y el 16vo bit vale cero; del mismo modo no se debe tomar en cuenta el acarreo del 17vo bit.

$$A = 11011011010101_2; \quad B = 1001011011101001_2$$

$$\begin{array}{r} 0\ 0\ 1\ 1\ 0\ 1\ 1\ 0\ 1\ 1\ 0\ 1\ 0\ 1\ 0\ 1_2\ + \\ 1\ 0\ 0\ 1\ 0\ 1\ 1\ 0\ 1\ 1\ 1\ 0\ 1\ 0\ 0\ 1_2 \\ \hline 0\ 1\ 1\ 0\ 0\ 1\ 1\ 0\ 1\ 1\ 0\ 1\ 1\ 1\ 1\ 0_2 \end{array}$$

→ Acarreo del 16vo bit = 0; A > 0; B < 0

$$A = 1111001111110000_2; \quad B = 100111011100101_2$$

$$\begin{array}{r} 1\ 1\ 1\ 1\ 0\ 0\ 1\ 1\ 1\ 1\ 1\ 1\ 0\ 0\ 0\ 0_2\ + \\ 0\ 1\ 0\ 0\ 1\ 1\ 1\ 0\ 1\ 1\ 1\ 0\ 0\ 1\ 0\ 1_2 \\ \hline 1\ 0\ 1\ 0\ 0\ 0\ 0\ 1\ 0\ 1\ 1\ 0\ 1\ 0\ 1\ 0\ 1_2 \end{array}$$

→ Acarreo del 16vo bit = 0; A < 0; B > 0

→ Acarreo del 17vo bit = 1

Con dos números de distintos signos se dan los casos de acarreo en el 17vo bit. Si éste acarreo es cero significa que el resultado es negativo y se debe complementar para hallar su verdadero valor de la otra forma, si el acarreo es uno, entonces el signo del resultado es mayor o igual a cero y se encuentra en verdadero valor.

IV) Suma de dos números negativos. El resultado debe ser negativo, por lo tanto el bit más significativo de la suma siempre dará uno.

$$A = 1100000111110110_2; \quad B = 1101110011111011_2$$

$$\begin{array}{r} 1\ 1\ 0\ 0\ 0\ 0\ 0\ 1\ 1\ 1\ 1\ 1\ 0\ 1\ 1\ 0_2\ + \\ 1\ 1\ 0\ 1\ 1\ 1\ 0\ 0\ 1\ 1\ 1\ 1\ 1\ 0\ 1\ 1_2 \\ \hline 1\ 1\ 1\ 0\ 0\ 1\ 1\ 1\ 1\ 0\ 1\ 1\ 1\ 0\ 0\ 0\ 1_2 \end{array}$$

→ Acarreo del 16vo bit = 1; A < 0; B < 0

De esta manera, el resultado queda en forma binaria normal y es igual a valor del 17vo bit no se toma en cuenta para el resultado. En decimal $A=23751_{10}$ y $B=15186_{10}$; entonces $A-B=8565_{10} = 0010000101110101_2$

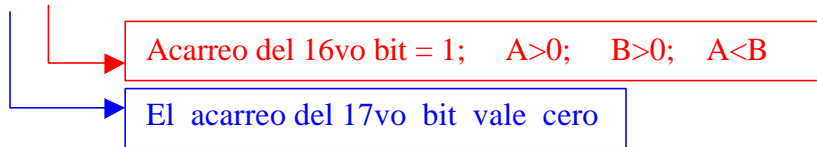
(A menor que B):

$$A = 1111001000100_2;$$

$$B = 0111100110101111_2$$

$$N_2^C(B) = 1000011001010001_2$$

$$\begin{array}{r}
 0\ 0\ 0\ 1\ 1\ 1\ 1\ 0\ 0\ 1\ 0\ 0\ 0\ 1\ 0\ 0_2\ + \\
 1\ 0\ 0\ 0\ 0\ 1\ 1\ 0\ 0\ 1\ 0\ 1\ 0\ 0\ 0\ 1_2 \\
 \hline
 0\ 1\ 0\ 1\ 0\ 0\ 1\ 0\ 0\ 1\ 0\ 0\ 1\ 0\ 1\ 0\ 1_2
 \end{array}$$



De esta manera, el resultado es negativo y queda en forma de complemento a dos, el acarreo del 17vo bit no se toma en cuenta. Sin embargo, para saber el verdadero valor, el resultado se debe complementar a dos. Este es un número binario negativo de 16 bits, lo cual tiene un valor de: $N_2^C(N_2^C(B)) = 0101101101101011_2$. En decimal la operación se efectúa: $A = 7748_{10}$ y $B = 31151_{10}$ entonces el resultado es $A-B = -23403_{10}$.

II) Resta de dos números negativos y de distinto signo. El resultado puede presentar varias formas que se determinan aplicando los mismos casos de la suma en formato de 16 bits.

Tabla 1. 3. Resumen de las operaciones suma y resta binaria con los datos A y B, utilizando el formato de 16 bits.

Operación	Acarreo 17vo bit	Acarreo 16vo bit	Resultado	Observaciones
A+B A>0; B>0	0	0	Positivo en binario normal	Chequear para no exceder el formato de 16 bits.
A+B A>0; B<0 (**)	0	1	Negativo en complemento a dos	Complementar los 16 bits para obtener el verdadero valor.
A+B A<0; B>0 (**)	1	0	Positivo en binario normal	El 17vo bit no se toma en cuenta para el resultado.
A+B A<0; B<0	1	1	Negativo en complemento a dos	Complementar los 16 bits para obtener el verdadero valor, Chequear para no exceder el formato de 16 bits y el 17vo bit no se toma en cuenta.
A-B A>0; B>0 A>=B	1	0	Positivo en binario normal	El 17vo bit no se toma en cuenta para el resultado.
A-B A>0; B>0 A<B	0	1	Negativo en complemento a dos	Complementar los 16 bits para obtener el verdadero valor.
A-B A>0; B<0	0	0	Positivo en binario normal	Chequear para no exceder el formato de 16 bits.
A-B A<0; B>0	1	1	Negativo en complemento a dos	Complementar los 16 bits para obtener el verdadero valor, Chequear para no exceder el formato de 16 bits y el 17vo bit no se toma en cuenta.
A-B A<0; B<0 (**)	0	1	Negativo en complemento a dos o positivo normal	Complementar los 16 bits para obtener el verdadero valor o dejarlo igual. Todo depende de la magnitud de A y B.
(**) Se producen resultados negativos o positivos dependiendo del mayor entre A y B.				

1.4.3 Representación numérica en coma fija y coma flotante.

Estas representaciones son utilizadas por las computadoras para procesar cálculos numéricos con formatos grandes. Consiste en una cadena de bits que guardan relación con la notación científica, y pueden representar números enteros y números reales tanto negativos como positivos. Los formatos más conocidos son la coma fija y la coma flotante, también denominados punto fijo y punto flotante respectivamente. Antes de comenzar a describir estos formatos se debe entender el funcionamiento de un caso especial de complemento a dos el cual se denomina representación con exceso o sesgada.

1.4.3.1 Representación con exceso o sesgada.

Son representaciones para números con signo que eliminan el centrado de la representación básica en complemento a dos. Por ejemplo para indicar números decimales desde un valor numérico $-P_{10}$ hasta $+P_{10}$ es necesario desplazar el equivalente binario $(-P_{10})_2$ sumando P_2 unidades positivas. Esta cantidad se conoce como exceso o sesgo. Las representaciones con exceso se utilizan, con frecuencia, para representar los exponentes de los números con coma flotante. En la tabla 1.4 se pueden observar las representaciones desde -8_{10} hasta $+8_{10}$ en complemento a dos y en código con exceso donde $P_2 = 1000_2$. En complemento a dos -8_{10} es igual a 1000_2 . Sin embargo, la representación del mismo número negativo en código desplazado con exceso 8 es de 0000_2 ; es de hacer notar que solamente ocurre un cambio en el bit más significativo (MSB: Most Significant Bit) del código con exceso. Por lo tanto, la representación de cualquier código con exceso $-P$, para indicar números negativos, se forma sumando el valor de P a cada palabra o número del código.

Tabla 1.4. Comparación de códigos en complemento a dos y exceso 8.

DECIMAL	COMPLEMENTO A DOS	EXCESO 8
+7	0111	1111
+6	0110	1110
+5	0101	1101
+4	0100	1100
+3	0011	1011
+2	0010	1010
+1	0001	1001
0	0000	1000
-1	1111	0111
-2	1110	0110
-3	1101	0101
-4	1100	0100
-5	1011	0011
-6	1010	0010
-7	1001	0001
-8	1000	0000

1.4.3.2 Representación numérica en coma fija.

Los números fraccionarios y con signo se pueden representar mediante la coma fija; ejemplo de esto se puede apreciar en la tabla 1.2 y la figura 1.3(a) donde se tiene la representación de números enteros con signo en formato de 16 bits. No obstante, existe otra representación para coma fija, la cual consiste en fijar la posición de la coma después del bit de signo; ver figura 1.3(b) respectivamente. Los restantes bits deben indicar la magnitud fraccionaria.

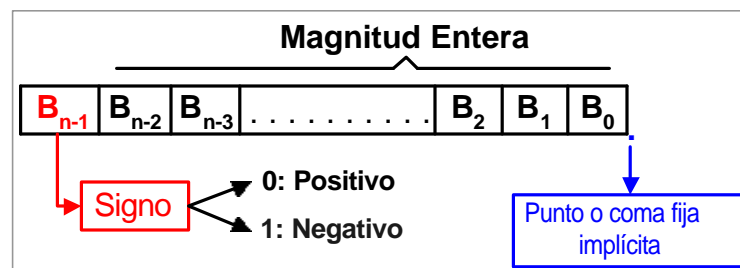


Figura 1.3 (a). Representación entera de coma fija.

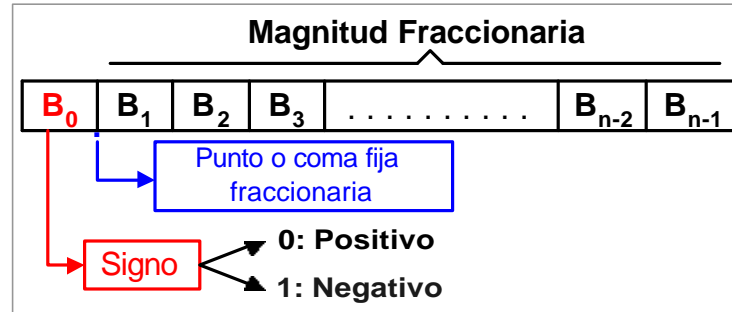


Figura 1.3 (b). Representación fraccionaria de coma fija.

1.4.3.3 Representación numérica en coma flotante.

Los números representados en coma flotante tienen la misma forma que la notación científica. La representación tiene la siguiente forma

$N = Mxb^E$ (Ec.1.6); donde M es la mantisa o significado y se representa en coma fija, este valor indica la cantidad de dígitos significativos que tiene el número N de coma flotante. El valor E es el exponente o característica, también de coma fija; está dado en formato de complemento a dos con exceso y b es la base del sistema. En forma general, de la Ec.1.1 se puede obtener la representación con signo de coma fija y está dada por: $N = \pm(a_{n-1}a_{n-2}.....a_0, a_{-1}a_{-2}.....a_{-k})_b$, ahora sustituyendo por el formato de coma fija, dada en la figura 1.3(b), se obtiene la forma de coma flotante

$$N = \pm(0, a_{n-1}a_{n-2}.....a_{-k})xb^n$$

$M = \pm(0, a_{n-1}a_{n-2}a_{n-3}.....a_{-k})$ (Ec.1.7). La fórmula general queda del siguiente modo;

$$N = (-1)^{bs} x(0, a_{n-1}a_{n-2}.....a_{-k})_b xb^{E'+2^{(e-1)}} \text{ (Ec.1.8)}$$

donde bs es el bit de signo, e es el número de bits del exponente con $E = E' + 2^{(e+1)}$; esto es equivalente a escribir E con formato de exceso en base dos de la siguiente manera; $E' = (c_{e-1}c_{e-2}.....c_0)_2$, por lo tanto, $E = (c_{e-1}c_{e-2}.....c_0)_2 + 2^{e-1}$

Existen varias formas de representar los formatos de coma flotante; sin embargo, los que más se utilizan son los siguientes:

- $N = Mxb^E$

- $N = (M \div b)xb^{E+1}$
- $N = (Mxb)xb^{E-1}$

En las figuras 1.4(a) y 1.4(b) se definen los formatos en coma flotante para datos numéricos reales cortos y largos utilizados en los computadores.

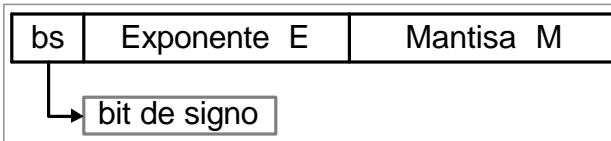


Figura 1.4(a). Declaración de datos cortos en coma flotante.

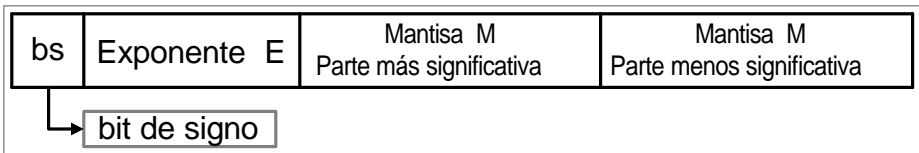


Figura 1.4(b). Declaración de datos largos en coma flotante.

La tabla 1.5 muestra un resumen de los formatos de precisión sencilla y doble (corto y largo) respectivamente; usados en los sistemas de computación.

FORMATO	TOTAL DE BITS	BITS DE LA MANTISA	BITS DEL EXPONENTE	EXCESO DEL EXPONENTE
Estándar IEEE				
754-1985				
Precisión sencilla	32	24	8	128
Doble Precisión	64	53	11	1024
IBM 360				
Precisión sencilla	32	24	7	64
Doble precisión	64	56	7	64
DEC VAX 11/780				
Formato F	32	24	8	128
Formato D	64	56	8	128
Formato G	64	53	11	1024

Tabla 1.5. Formatos comunes para números representados en coma flotante.

Ejemplo 1.21. Escribir en formato de coma flotante los números: a) $11011101,1101_2$
 b) $0,0000111010101_2$

Solución (a): Se debe llevar a la forma $N = Mxb^E$; primero hay que hallar la mantisa con la Ec.1.7 y luego el exponente **E** con exceso;

$$M=+(0,1101110111010)_2$$

$E=+8_{10}=+(1000)_2$; si el bit de signo es positivo entonces $E'=0100_2$. En este caso hay que sumarle al exponente un exceso de 16_{10} ; $E=0100_2+1000_2 = 11000_2$

La solución final queda de la siguiente forma:

bs	Exponente E	Mantisa M
0	11000	1101110111010

Solución (b): Se debe llevar a la forma $N = Mxb^E$; primero hay que hallar la mantisa con la Ec.1.7 y luego el exponente **E** con exceso;

$$M=+(0,111010101)_2$$

$E=-4_{10}=-100_2$; si el bit de signo es negativo entonces $E'=1100_2$. En este caso hay que sumarle al exponente un exceso de 8_{10} ; $E=1100_2+1000_2 = 0100_2$

La solución final queda de la siguiente forma:

bs	Exponente E	Mantisa M
0	0100	111010101

1.5 Códigos de numeración, alfanuméricos y de errores.

Los códigos en los sistemas digitales se clasifican en tres tipos: códigos numéricos, códigos alfanuméricos y códigos detectores y correctores de errores. El objetivo de los códigos es simplificar la comunicación entre los distintos circuitos digitales, normalizar el funcionamiento de los mismos y detectar posibles fallas de datos para su posterior corrección.

1.5.1 Códigos numéricos.

Los más utilizados, en circuitos digitales combinacionales son el código BCD, Exceso 3, Aiken o 2421, 5421, Biquinario, Dos de Cinco. Existen otros códigos de tipo secuencial cíclicos, dos de ellos es son código Jhonson y el código Gray. En la tabla 1.6 se describen algunos de ellos con sus respectivos equivalentes decimales.

Decimal	BCD	Exceso 3	2421	5421	Biquinario	Dos de cinco	Gray
0	0000	0011	0000	0000	0100001	00011	0000
1	0001	0100	0001	0001	0100010	00101	0001
2	0010	0101	0010	0010	0100100	01001	0011
3	0011	0110	0011	0011	0101000	10001	0010
4	0100	0111	0100	0100	0110000	00110	0110
5	0101	1000	1011	1000	1000001	01010	0111
6	0110	1001	1100	1001	1000010	10010	0101
7	0111	1010	1101	1010	1000100	01100	0100
8	1000	1011	1110	1011	1001000	10100	1100
9	1001	1100	1111	1100	1010000	11000	1101
10	0001 0000	0100 0011	0001 0000	0001 0000	0100010 0100001	00101 00011	1111
11	0001 0001	0100 0100	0001 0001	0001 0001	0100010 0100010	00101 00101	1110
12	0001 0010	0100 0101	0001 0010	0001 0010	0100010 0100100	00101 01001	1010
13	0001 0011	0100 0110	0001 0011	0001 0011	0100010 0101000	00101 10001	1011
14	0001 0100	0100 0111	0001 0100	0001 0100	0100010 0110000	00101 00110	1001
15	0001 0101	0100 1000	0001 1011	0001 1000	0100010 1000001	00101 01010	1000

Tabla 1.6. Equivalencia desde cero hasta quince de algunos códigos numéricos más utilizados.

1.5.1.1 Código BCD.

(Binario Codificado en Decimal): La conversión con el sistema decimal se realiza directamente, en grupos de cuatro bits por cada dígito decimal con ponderación 8421. Este código tiene aplicación en visualizadores (displays) hechos con diodos led o LCD, los cuales poseen previamente convertidores que transforman el grupo de cuatro bits BCD en otro especial, llamado 7 segmentos. En el capítulo V se ahondará más el tema.

Por ejemplo, para transformar el número decimal 78905_{10} en código BCD se toman los equivalentes en grupos de cuatro bits cada uno; ver tabla 1.6:

7	8	9	0	5_{10}
0111	1000	1001	0000	0101_{BCD}

Resp: $78905_{10} = 0111\ 1000\ 1001\ 0000\ 0101_{BCD}$

Para realizar la equivalencia del BCD con el sistema binario se debe tomar la precaución de realizar primero la transformación decimal y posteriormente la conversión al BCD.

Ejemplo 1.22. Transformar en BCD los siguientes números:

- a) 1011101111111_2 ; b) $5F3C_{16}$

Solución (a): $1011101111111_2 = 6015_{10} = 0110\ 0000\ 0001\ 0101_{BCD}$

Solución (b): $5F3C_{16} = 24380,6875_{10} = 0010\ 0100\ 0011\ 1000\ 0000, 0110\ 1000\ 0111\ 0101_{BCD}$

1.5.1.1.1 Suma en BCD.

La suma en BCD puede dar como resultado un número no perteneciente al código. Por ejemplo, al sumar los números BCD $1000 + 0001$ el resultado es 1001 , este número también pertenece al código; sin embargo, cuando se suman $0111 + 1000$ el resultado es 1111 , este número no pertenece al código BCD y su valor equivalente es quince unidades.

Cuando suceden estos casos es necesario sumar un factor de corrección que depende del rango donde se encuentre el resultado de la suma. La tabla 1.7 muestra los valores del factor de corrección con su respectivo rango. Para el rango binario desde diez (1010_2) hasta diecinueve (10011_2) el factor de corrección es seis 0110_2 ; este factor se duplica en forma proporcional del mismo modo que aumenta la decena en el resultado. Por lo que se debe aplicar la fórmula $Fc = n_2 \cdot (0110)_2$ donde n es igual al valor binario de la decena del resultado. El factor de corrección se debe aplicar siempre y cuando el resultado de la suma sea mayor o igual a diez. Del mismo modo, la suma debe realizarse en binario.

Factor de corrección Binario $(Fc)_2$	Valor decimal de rango	Valor decimal de la decena (n)	Factor de corrección Decimal $(Fc)_{10}$
0110	(10 ~ 19)	1	6
1100	(20 ~ 29)	2	12
10010	(30 ~ 39)	3	18
11000	(40 ~ 49)	4	24
.	.	.	.
.	.	.	.
.	.	.	.
$Fc = n_2 \times (0110)_2$	$(n0 \sim n9)$	n	$Fc = n \times 6$

Tabla 1.7. Factores de corrección para la suma BCD.

Ejemplo 1.23. Dado los números p, q, r en código BCD, sumar: **a) $p + q + r$, b) $q + r$, c) $p + q$** y obtener el resultado también en BCD.

$p = 1000\ 0110\ 0010\ 0000\ 1001_{BCD}$; $q = 0100\ 1001\ 1001\ 0011\ 0111_{BCD}$

$r = 0111\ 1001\ 1000\ 0110\ 0010\ 0011\ 1001_{BCD}$

Solución (a): Los resultados que superen el **1001** hay que sumarle el factor de corrección según la tabla 1.7 y llevar el acarreo correspondiente.

Acarreo	1	10	10	1	10	
P =	↑	↑	↑	↑	↑	1000 0110 0010 0000 1001 +
Q =						0100 1001 1001 0011 0111
R =	0111	1001	1000	0110	0010	0011 1001
	1000	1011	10110	10110	1101	1000 11001 +
Fc =	0000	0110	1100	1100	0110	0000 1100
Resultado =	1000	10001	100010	100010	10011	1000 100101

Respuesta (a): $p+q+r = 1000\ 0001\ 0010\ 0010\ 0011\ 1000\ 0101_{BCD} = 8122385_{10}$

Solución (b): Los resultados que superen el **1001** hay que sumarle el factor de corrección según la tabla 1.7 y llevar el acarreo correspondiente.

	1		1		1		1		1	
			0100	1001	1001	0011	0111	+		
0111	1001	1000	0110	0010	0011	1001				
1000	1010	1101	10000	1011	0111	10000	+			
0000	0110	0110	0110	0110	0000	0110				
1000	10000	10011	10110	10001	0111	10110				

Respuesta (b): $q+r = 1000\ 0000\ 0011\ 0110\ 0001\ 0111\ 0110_{BCD} = 8036176_{10}$

Solución (c): Los resultados que superen el **1001** hay que sumarle el factor de corrección según la tabla 1.7 y llevar el acarreo correspondiente.

	1		1		1		1	
	1000	0110	0010	0000	1001	+		
	0100	1001	1001	0011	0111			
	1101	10000	1011	0100	10000	+		
	0110	0110	0110	0000	0110			
0001	10011	10110	10001	0100	10110			

Respuesta (c): $p+q = 0001\ 0011\ 0110\ 0001\ 0100\ 0110_{BCD} = 136146_{10}$

1.5.1.2 Código Exceso 3.

Es un código igual al BCD, sin embargo se deben añadir tres unidades a este para transformarlo en exceso 3.

1.5.1.3 Código Aiken o 2421.

La ponderación de este código es diferente al BCD, para hallar su peso se debe tomar también grupos de cuatro bits, considerando los valores 2421, por dígito decimal.

Este código se conoce como autocomplementado a uno porque sus diez valores, en la tabla 1.6; se pueden formar, complementando, a partir de los primeros cinco dígitos.

1.5.1.4 Código 5421.

La ponderación de este código es diferente al BCD, para hallar su peso se debe tomar también grupos de cuatro bits, considerando los valores 5421, por dígito decimal. Este código se forma repitiendo los cinco primeros valores de la tabla 1.6, de modo tal, que cambia solo el bit más significativo de cero a uno.

1.5.1.5 Código Biquinario.

Necesita siete bits para formarse; siempre hay dos bits en nivel alto (uno) y los restantes cinco deben estar en nivel bajo (cero). El primer bit del código, en uno, se usa para indicar si el dígito se encuentra comprendido entre 5 y 9; el segundo bit del código, en uno, señala que se encuentra en el rango de 0 a 4. La desventaja de este código es la cantidad de bits que se deben utilizar para transmitir información, siete por cada dígito. Sin embargo, tiene la ventaja de poder realizar fáciles algoritmos para el chequeo de errores de transmisión; solamente se debe detectar que hayan dos bits, en nivel uno, por cada dato. Uno de estos se debe encontrar entre los primeros dos bits y el otro en los cinco restantes que forman el dígito.

1.5.1.6 Código Dos de cinco.

Este código es similar al Biquinario, pero requiere de cinco bits para el correcto funcionamiento. Dos bits deben estar en nivel alto y los otros tres en cero.

1.5.1.7 Código Gray.

Este código cíclico no posee una relación directa con la ponderación de los dígitos del sistema decimal. Se forma cambiando el bit menos significativo de manera continua y consecutiva. Solamente cambia un bit, y éste, debe ser el menos significativo; de manera que no se repita con alguna combinación anterior. También se puede formar obteniendo las primeras ocho combinaciones con tres bits y luego, desde

la 8va combinación hay que repetir simétricamente los valores, cambiando solamente el bit más significativo de cero a uno. Por ejemplo, la 8va posición es **0100** y a continuación viene la 9na **1100**; del mismo modo, la 7ma **0101** es simétrica con la 11va **1101**. El código Gray tiene aplicaciones en contactos de escobillas de motores, sistemas donde solo se necesite cambiar un bit de estado cíclicamente.

La ventaja del código Gray radica en que la probabilidad de ocurrir menos errores y problemas de transición aumenta a medida que cambian mas bits de estado simultáneamente. El cambio consecutivo del código BCD desde **0111** a **1000** puede producir transiciones intermedias que originan el **1111** antes de estabilizarse en **1000**. Sin embargo, el código Gray pasará desde **0111** a **0101** cambiando solamente un bit y por lo tanto, con menos posibilidad de cometer errores.

1.5.2 Códigos alfanuméricos.

Estos códigos son interpretados por el computador como caracteres e indistintamente pueden representar símbolos numéricos, símbolos de control y letras. Las computadoras se comunican mediante estos códigos y los más utilizados son el código ASCII y el UNICODE.

1.5.2.1 Código ASCII.

ASCII: American Standard Code Interchange Information. Cada caracter alfanumérico esta formado por una cadena de siete bits. Este código representa 128 símbolos diferentes entre dígitos, letras e instrucciones de control del computador. La tabla 1.xx muestra los símbolos con su respectivo valor hexadecimal. Por ejemplo, para codificar la palabra UNEXPO se procede de la siguiente forma:

1010101 1001110 1000101 1011000 1010000 1001111

U	N	E	X	P	O
55H	4EH	45H	58H	50H	4FH

Tabla 1.8. Código ASCII.

		B₆B₅B₄							
	BIN	000	001	010	011	100	101	110	111
B₃B₂B₁B₀	HEX	0	1	2	3	4	5	6	7
0000	0	NUL	DLE	SP	0	@	P	`	p
0001	1	SOH	DC1	!	1	A	Q	a	q
0010	2	STX	DC2	"	2	B	R	b	r
0011	3	ETX	DC3	#	3	C	S	c	s
0100	4	EOT	DC4	\$	4	D	T	d	t
0101	5	ENQ	NAK	%	5	E	U	e	u
0110	6	ACK	SYN	&	6	F	V	f	v
0111	7	BEL	ETB	'	7	G	W	g	w
1000	8	BS	CAN	(8	H	X	h	x
1001	9	HT	EM)	9	I	Y	i	y
1010	A	LF	SUB	*	:	J	Z	j	z
1011	B	VT	ESC	+	;	K	[k	{
1100	C	FF	FS	,	<	L	\	l	
1101	D	CR	GS	-	=	M]	m	}
1110	E	SO	RS	.	>	N	^	n	~
1111	F	SI	US	/	?	O	_	o	DEL

1.5.2.2 UNICODE.

Es un código universal actualizado de propósito general, sirve para representar todos los símbolos utilizados en los alfabetos internacionales. Es una nueva norma de códigos alfanuméricos de 16 bits. Los símbolos se representan con cuatro dígitos hexadecimales como se muestra en la tabla 1.9. El código ASCII es un subconjunto de éste y está representado desde 0000_{16} hasta $007F_{16}$. En la figura 1.4 se observa la distribución del código en cuatro zonas que van desde 0000_{16} hasta $FFFF_{16}$. La zona A comprende los códigos para alfabetos, sílabas, y símbolos. En la zona I están los códigos ideográficos como lo son los alfabetos Chinos y Japoneses. La zona O no es utilizada actualmente, sin embargo, está reservada para futuros ideogramas.

La zona R es de uso restringido. Se subdivide en Área de uso privado, Área de compatibilidad y Códigos especiales. FFFE y FFFF no son códigos de carácter y se excluyen específicamente del UNICODE. El Área de uso privado está a disposición de quienes necesiten caracteres especiales para sus programas de aplicación; por ejemplo, los iconos empleados en los menús podrían especificarse por medio de códigos de carácter en esta área. La zona de compatibilidad tiene caracteres correlacionados con otras áreas del espacio global de código. La transmisión serial de un carácter UNICODE se realiza con dos bytes (**byte 0 y byte 1**). Primero se envía la palabra de control FFFE o FEFF indicando cual de los dos bytes es el más significativo; Por ejemplo, al enviar los símbolos FFFE, 4100, 4E00, 4700, 4500, 4C00 indica que se debe cambiar el orden de los bytes, esto es: 0041, 004E, 0047, 0045, 004C que se codifica como 'ANGEL' en la tabla 1.9. Sin embargo, en caso de haber enviado la palabra de control FEFF indicaba que el orden de los bytes era el mismo. Lo que no correspondía con los códigos ASCII del UNICODE.

Estos ordenamientos en los bytes del UNICODE guardan relación con los formatos de datos para comunicación de computadoras Little-Endian o Big-Endian.

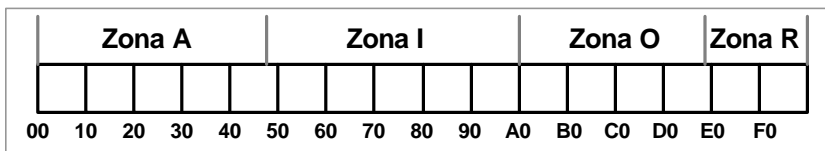


Figura 1.4. Distribución del código UNICODE.

Ejemplo 1.24. Indicar si es posible decodificar las siguientes palabras dadas en UNICODE.

- a) FFFE, 4300, A200, 6400, 6900, 6700, 6F00
- b) FEFF, 0055, 004E, 0045, 0058, 0050, 004F

Solución (a): El orden de los bytes debe ser invertido; 0043, 00A2, 0064, 0069, 0067, 006F que corresponde con la palabra '**Código**'.

Solución (b): El orden de los bytes es el correcto 0055, 004E, 0045, 0058, 0050, 004F que corresponde con la palabra '**UNEXPO**'.

Tabla 1.9. Primeros 256 Símbolos UNICODE.

HEX	000	001	002	003	004	005	006	007	008	009	00A	00B	00C	00D	00E	00F
0	CTL	CTL	SP	0	@	P	`	P	Ç	É	á	_	+	ð	Ó	-
1	CTL	CTL	!	1	A	Q	a	Q	ü	æ	í	_	-	Ð	ß	±
2	CTL	CTL	"	2	B	R	b	R	é	Æ	ó	_	-	È	Ö	_
3	CTL	CTL	#	3	C	S	c	S	â	ô	ú	!	+	É	Ó	¾
4	CTL	CTL	\$	4	D	T	d	T	ä	ö	ñ	!	-	Ë	õ	¶
5	CTL	CTL	%	5	E	U	e	U	à	ò	N	À	+	Ì	Ò	§
6	CTL	CTL	&	6	F	V	f	V	â	û	ª	À	À	Ì	µ	÷
7	CTL	CTL	'	7	G	W	g	W	ç	ù	º	À	À	Ì	Þ	¸
8	CTL	CTL	(8	H	X	h	X	ê	ÿ	¿	©	+	Ì	Þ	°
9	CTL	CTL)	9	I	Y	i	Y	ë	Û	®	!	+	+	Û	”
A	CTL	CTL	*	:	J	Z	j	Z	è	Ü	¬	!	-	+	Û	•
B	CTL	CTL	+	;	K	[k	{	ï	ø	½	+	-	_	Û	¹
C	CTL	CTL	,	<	L	\	l		î	£	¼	+	!	_	Û	³
D	CTL	CTL	-	=	M]	m	}	ì	Ø	¡	¢	-	!	Û	²
E	CTL	CTL	.	>	N	^	n	~	À	x	«	¥	+	Ì	-	_
F	CTL	CTL	/	?	O	_	o	CTL	À	f	»	+	CTL	_	'	SP

1.5.3 Códigos detectores y correctores de errores.

La transmisión y recepción de datos binarios, desde un dispositivo a otro, están propensas a errores, campos magnéticos, interferencias y ruidos eléctricos pueden ocasionar este problema. El costo agregado que ocasiona añadir circuitos detectores y correctores de error se ve compensado con el avance de la tecnología en el área de las telecomunicaciones. Los sistemas de comunicación digital son la tecnología de punta en el ámbito mundial y, específicamente, las redes de computadoras; ejemplo de esto son las redes locales, Internet, etc.

Los sistemas deben detectar y/o corregir errores de comunicación en el menor tiempo posible de manera que puedan mantener el intercambio de información digital en línea y en tiempo real. La tarea no parece sencilla; sin embargo, los diseñadores de sistemas digitales deben considerar el costo de estos circuitos adicionales, a la hora de

implementar el circuito. De hecho, es necesario agregar más bits al dato que se desea transmitir con la finalidad de chequear, en el receptor, los posibles errores durante el proceso de comunicación.

El método para realizar esto; va desde solicitar que reenvíen el dato, el bloque o hasta la información completa. También hay métodos más seguros que implementan sistemas redundantes de tres o más circuitos de comunicación idénticos que operan en paralelo y por lo tanto disminuyen considerablemente el índice de errores.

En esta sección se analizarán los métodos de detección de errores por paridad y detección y/o corrección mediante el código Hamming.

1.5.3.1 Distancia y peso de los datos binarios.

Para chequear un bit de dato, en el receptor, es necesario agregar al sistema de comunicación, por lo menos, otro bit. De esta manera, el código queda formado por dos bits; uno para dato y el otro para chequeo y control. De esta misma forma, se debe establecer un patrón de comunicación (protocolo de comunicación). Por ejemplo, establecer que el bit de control se genere de la siguiente forma: sea el más significativo y además, la suma de los dos bits sea siempre par. Esto se ilustra en la figura 1.5; aquí se puede ver los cuatro cambios posibles de los bits X y b_0 . El bit b_0 tiene dos valores posibles 0 y 1; para enviar un cero se debe agregar en el generador de paridad G_P otro cero para mantener la paridad par. Si, por el contrario, el b_0 es uno entonces hay que generar en G_P un uno para mantener el protocolo de paridad par sin errores. El circuito receptor de información detecta la paridad de los dos bits $(X b_0)$, chequea las combinaciones posibles; activando la señal de error cuando es recibida la combinación $(0 1)$ o $(1 0)$. Este ejemplo se puede extender para datos que tengan n bits de información ya que, basta un bit adicional, para generar y chequear errores de paridad. Para entender mejor esta última afirmación, se definen a continuación, los términos distancia y peso en los datos binarios.

La distancia máxima entre dos datos binarios, de igual longitud, es equivalente al número de bits que cambian de estado. Por ejemplo, la distancia entre los datos $D_1=10010111$ y $D'_1=10110001$ es tres. La distancia se puede definir también como el número de bits diferentes entre dos palabras.

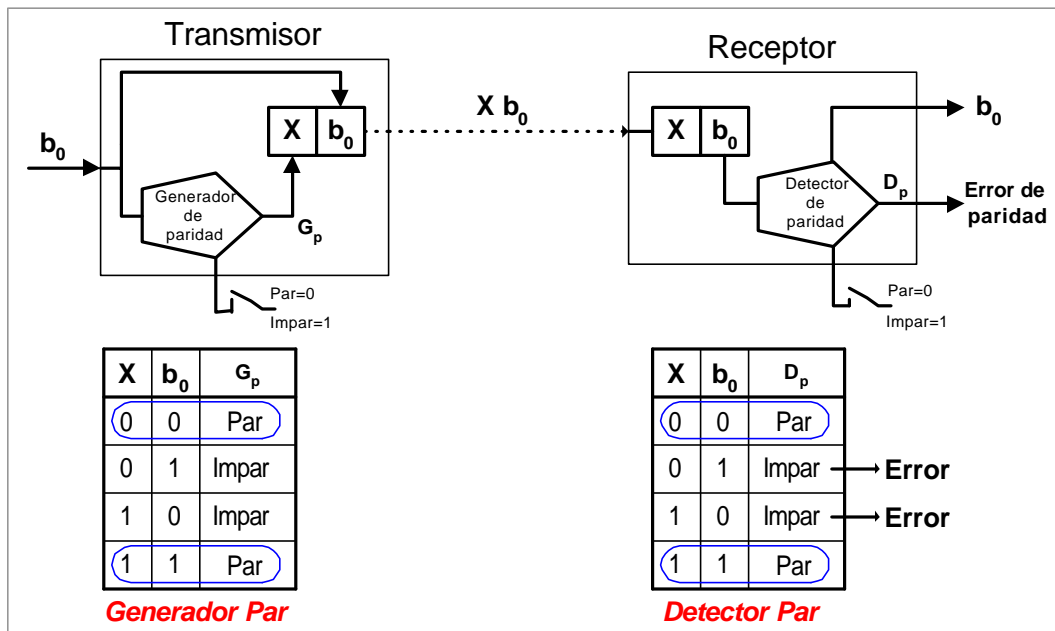


Figura 1.6. Sistema de transmisión y recepción de un bit con generación y detección de error mediante el método de paridad par.

Otro ejemplo para tomar en cuenta es el caso donde la palabra transmitida y recibida difieren en dos bits; esto es, transmitida $A=1100101$ y recibida $A'=1101100$. La distancia es dos; sin embargo, aunque la palabra cambie, la paridad se mantiene y por lo tanto no habrá señalización de error. Al comparar, este caso, con el cambio entre D_1 y D'_1 se observa que si hay señalización de error porque la paridad no se mantiene.

El número de bits en nivel uno de $(D_1 - D'_1)$ no son iguales. Por el contrario, en el caso $(A - A')$ se observa el mismo número de bits en uno. Este número de bits en nivel alto, de un dato binario, es lo que se conoce como el peso de la palabra o peso del dato binario. Por ejemplo, D_1 tiene un peso de 5 y D'_1 tiene un peso de 4; del mismo modo, A y A' pesan respectivamente 4.

1.5.3.2 Detección de error usando el método de paridad.

El sistema de chequeo de error por paridad es muy utilizado en las comunicaciones seriales de datos. El método consiste en establecer un tipo de paridad (par o impar) en el sistema de comunicación y generar en el transmisor, un bit adicional de modo que el peso del dato corresponda con la paridad (par o impar) establecida. Por lo general, este bit se agrega en la posición más significativa del dato.

Ejemplo 1.25. En los datos a, b, y c generar el bit de paridad par e impar en la posición más significativa (MSB).

a) 1010; b) 1110101; c) 00001

Solución par: El bit, hay que generarlo en el MSB de forma que el peso sea par;

a) 01010; b) 11110101; c) 100001

Solución impar: El bit, hay que generarlo en el MSB de forma que el peso sea impar;

a) 11010; b) 01110101; c) 000001

Ejemplo 1.26. Un sistema de comunicación ha recibido los siguientes caracteres ASCII: I) 01000001; II) 10111000; III) 11111110; y se desea saber si hay error. El protocolo de paridad es par. Indicar, en caso de ser correcto, el carácter enviado.

Solución (I): El peso de este dato es par (dos), por lo tanto, es correcto y corresponde al carácter ASCII 41H = 'A'.

Solución (II): El peso de este dato es par (cuatro), por lo tanto, es correcto y corresponde al carácter ASCII 38H = '8'.

Solución (III): El peso de este dato es impar (siete), por lo tanto, hay error de transmisión. En estos casos no es posible reconstruir el dato.

1.5.3.3 Detección y corrección de errores mediante el código Hamming.

El método de paridad con un solo bit es eficiente en la detección de errores cuando hay confiabilidad en el sistema de comunicación. De hecho, el peso del dato queda determinado con $m=n+1$ bits, donde n es el número de bits que contiene la información. Este método solamente puede detectar errores de dos datos que difieran en un bit; osea, tengan distancia uno y que cambie, por error del sistema, solamente un bit. Sin embargo, no los corrige y a lo sumo, puede señalar error y/o solicitar que vuelvan a enviar el byte, dato, palabra, o bloque de información que presentó el problema de comunicación.

De la misma forma, si hay cambios de distancias pares (2,4, 6,...), el método no detectará error. Sin embargo, en las distancias impares señala los errores. Ejemplo de esto se puede ver comparando, en el punto anterior, los casos ($D_1 - D'_1$) y ($A - A'$).

En 1950 R.W. Hamming introdujo un método para detectar y corregir errores de datos en los sistemas de comunicación donde las distancias pueden ser mayores a la unidad. Este código trabaja con una distancia mínima de tres y puede detectar errores con cambios de 1 o 2 bits y corregir, cambios de un solo bit.

Los bits necesarios para el código Hamming se dividen en dos grupos; m bits de información y k bits de chequeo o paridad, por lo que, el tamaño del dato a transmitir debe ser $n=m+k$ bits. Éste debe cumplir con la siguiente ecuación:

$$2^k \geq m+k+1 \text{ (Ec.1.9).}$$

La paridad del código puede ser par o impar, sin embargo, toda la información relacionada está dada en paridad par. Por lo tanto, los ejemplos se realizaran tomando como referencia codificación Hamming de paridad par con el número de bits n igual a siete. En la figura 1.7 se observa la distribución de paridades para los bits de chequeo con formato de siete bits de dato. De esta forma, al aplicar la Ec.1.9 se determina que $m=4$ y $k=3$, por lo tanto la información que se puede transmitir va desde 0000_2 hasta 1111_2 ; éstos están distribuidos, en la figura 1.7 como I_7, I_6, I_5, I_3 y deben mezclarse con los de chequeo C_4, C_2, C_1 . Estos últimos ocupan las posiciones de la potencia en base 2 indicada por los subíndices dos, uno y cero respectivamente.

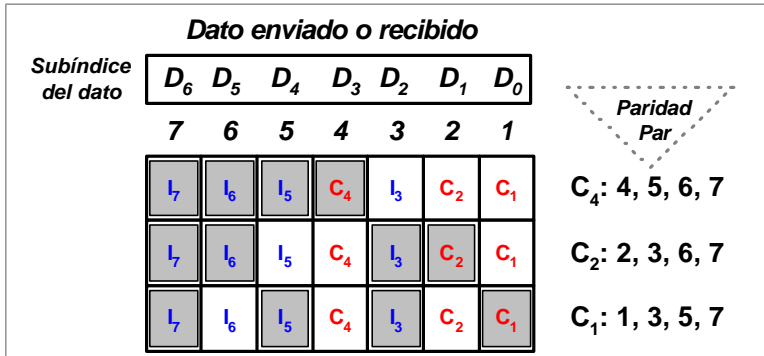


Figura 1.7. Formación del código Hamming de 7 bits.

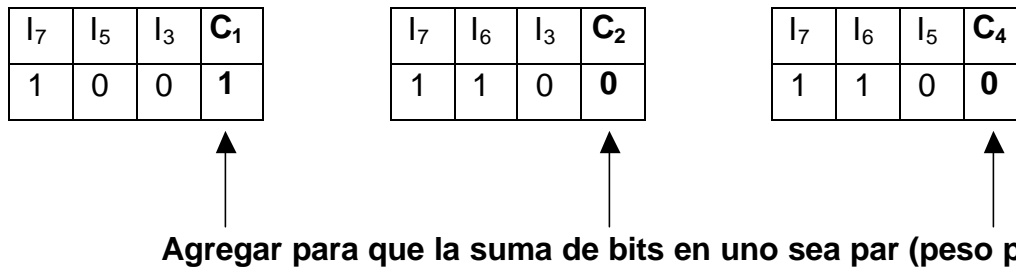
El código se forma entrelazando los bits de información ($q_3 q_2 q_1 q_0$) con los bits de control ($h_2 h_1 h_0$) de forma que los subíndices de h correspondan con la posición decimal del código formado. Los bits ($q_3 q_2 q_1 q_0$) de información se hacen corresponder, en la figura 1.7, con los bits ($I_7 I_6 I_5 I_3$) respectivamente; la finalidad es ubicarlos en la posición decimal del código. Del mismo modo, ($h_2 h_1 h_0$) es equivalente con las posiciones según en subíndice $h_2=C_2^2=C_4$; $h_1=C_2^1=C_2$; $h_0=C_2^0=C_1$. Finalmente el código de siete bits queda formado de la siguiente manera:

q_3	q_2	q_1	h_2	q_0	h_1	h_0
I ₇	I ₆	I ₅	C ₄	I ₃	C ₂	C ₁
D ₆	D ₅	D ₄	D ₃	D ₂	D ₁	D ₀

Al enviar el dato de siete bits, este es recibido como un paquete formado por ($D_6 D_5 D_4 D_3 D_2 D_1 D_0$) donde no se reconoce quien es información y/o quien es control. Sin embargo, con el método se realizan tres grupos de detección y corrección formado por cuatro bits cada uno, los cuales siempre deben tener paridad par. Estos grupos están resaltados de gris en la figura 1.7 y forman tres cuartetos agrupados de la siguiente forma: ($I_7 I_5 I_3 C_1$); ($I_7 I_6 I_3 C_2$); ($I_7 I_6 I_5 C_4$). Ellos sirven tanto para generar, detectar y corregir datos con distancia uno y dos respectivamente.

Por ejemplo, para enviar el dato de información (1100) codificado en Hamming se deben agregar tres bits de control de manera que los cuartetos tengan paridad par:

Primero hay que hacer corresponder los bits de información; (1100)=($I_7 I_6 I_5 I_3$), después se organizan los cuartetos de forma que la paridad sea par:



Los bits de control generados son: ($C_4 C_2 C_1$) = (001); en consecuencia el dato a enviar es ($D_6 D_5 D_4 D_3 D_2 D_1 D_0$) = ($I_7 I_6 I_5 C_4 I_3 C_2 C_1$) = (1100001). De la misma forma se procede a obtener la codificación de los bits en código Hamming. En la tabla 1.10 están representados los 4 bits de información y los tres bits de chequeo del código Hamming de 7 bits. También se puede observar que la mínima distancia, entre dos datos consecutivos, es tres.

Decimal	Información	Control	Dato codificado
	$I_7 I_6 I_5 I_3$	$C_4 C_2 C_1$	$I_7 I_6 I_5 C_4 I_3 C_2 C_1$
0	0000	000	0000000
1	0001	011	0000111
2	0010	101	0011001
3	0011	110	0011110
4	0100	110	0101010
5	0101	101	0101101
6	0110	011	0110011
7	0111	000	0110100
8	1000	111	1001011
9	1001	100	1001100
10	1010	010	1010010
11	1011	001	1010101
12	1100	001	1100001
13	1101	010	1100110
14	1110	100	1111000
15	1111	111	1111111

Tabla 1.10. Código Hamming de 7 bits.

También se pueden corregir errores de datos con distancia uno de la siguiente forma:

Ejemplo 1.27. Se han recibido los datos **a**, **b**, **c**, **d** codificados en Hamming de 7 bits con paridad par, y es necesario detectar y corregir los bits con errores.

- a) 1100100; b) 1110101; c) 1010101; d) 1110111

Solución (a): Para mantener la paridad par en el grupo 2,3,6,7 debe cambiarse el bit de la posición 2 (C_2). El dato corresponde a 1101.

I_7	I_6	I_5	C_4	I_3	C_2	C_1
1	1	0	0	1	0	0

I_7	I_6	I_5	C_4
1	1	0	0
I_7	I_6	I_3	C_2
1	1	1	0
I_7	I_5	I_3	C_1
1	0	1	0

Error en C_2 ; este bit debe ser **1**

Solución (b): Para mantener la paridad par en los grupos 2,3,6,7 y 4,5,6,7 se debe cambiar el bit de la posición 6 (I_6) para obtener la paridad correcta. El dato es: 1011.

I_7	I_6	I_5	C_4	I_3	C_2	C_1
1	1	1	0	1	0	1

I_7	I_6	I_5	C_4
1	1	1	0
I_7	I_6	I_3	C_2
1	1	1	0
I_7	I_5	I_3	C_1
1	1	1	1

Error en I_6 ; este bit debe ser **0**

Solución (c): En este caso, no hay error en el dato enviado.

I_7	I_6	I_5	C_4	I_3	C_2	C_1
1	0	1	0	1	0	1

I_7	I_6	I_5	C_4
1	0	1	0
I_7	I_6	I_3	C_2
1	0	1	0
I_7	I_5	I_3	C_1
1	1	1	1

Solución (d): Para mantener la paridad par en los grupos 4,5,6,7 se debe cambiar el bit de la posición 4 (C_4) para obtener la paridad correcta. El dato es: 1111.

I_7	I_6	I_5	C_4	I_3	C_2	C_1
1	1	1	0	1	1	1

I_7	I_6	I_5	C_4
1	1	1	0
I_7	I_6	I_3	C_2
1	1	1	1
I_7	I_5	I_3	C_1
1	1	1	1

Error en C_4 ; este bit debe ser **1**

Los casos **a** y **d** pueden ser aceptados como errores dobles o simple. Sin embargo, al asumir algún cambio en los bits de chequeo implica descartar errores dobles en los bits de información. Debido a esto, en el ejemplo 1.27(a) pueden ser considerado los cambios de los bits I_7 e I_5 . De esta misma forma, en el ejemplo 1.27(d), los cambios pueden ocurrir en los bits I_7 e I_3 . Los cambios dobles (distancia dos) no pueden ser corregidos con el código Hamming de 7 bits, sin embargo, para resolver esto es necesario el código Hamming de 8 bits.

Ejercicios propuestos 1.2

1.2.1 Dado los siguientes números:

- a) 10111011101_2 b) $6FAB_{16}$ c) $100100000111001010000110_{BCD}$ d) $58FF3D_{16}$
 e) 1111011010101011_2 f) $5432,76_8$ g) $11000011001110000110_{Exc3}$ h) $7964,9_{10}$

Hallar las sumas:

- I) a+b en octal II) c+e+f en hexadecimal III) c+d en binario
 IV) f+g+h en BCD V) b+e+a+f en octal VI) f+b+c en binario

1.2.2 Dado los siguientes números:

- a) $FA0B_{16}$ b) 1101101101_2 c) 43375_8
 d) $7FFF_{16}$ e) -9863_{10} f) 1111000010101000_2

Realizar las siguientes operaciones aritméticas utilizando el formato de números con signo de 16 bits:

- I) $a - c$ II) $b + a$ III) $d - b$
IV) $e + c$ V) $f - e$ VI) $b + e + d$

1.2.3 Un sistema de comunicación envía datos de 9 bits. En cada uno, se codifican dos dígitos BCD más un bit de paridad que es generado en la posición más significativa y con paridad par. Se pide detectar los errores que puedan ocurrir en los códigos BCD recibidos, e indicar si son de paridad y/o de código.

- a) 101111001 b) 110011100 c) 111110001
d) 010000100 e) 010101011 f) 100000111

1.2.4 Los siguientes caracteres UNICODE son enviados en binario con paridad impar en el MSB. Detectar, por el método de paridad, si hay errores de comunicación, y de no ser así, indicar el símbolo correspondiente.

- a) 101111110 b) 110100101 c) 101101110
d) 110101100 e) 001000001 f) 00100000

1.2.5 Dado los números:

- a) $10011000011100000100_{\text{BCD}}$ b) 789463_{10}
c) $110010001010001100111001_{\text{Exc3}}$ d) $0100011100111001100001110000_{\text{BCD}}$

Realizar las siguientes sumas en BCD.

- I) $a + c + d$ II) $c + b$ III) $a + b + c + d$

1.2.6 Detectar y corregir los errores de los siguientes datos, dados en exceso 3, y codificados en Hamming de 7 bits con paridad par.

- a) 1100001 b) 1000110 c) 0101100
d) 1111111 e) 0001110 f) 0000001

BIBLIOGRAFÍA.

- CUESTA, Luís M. PADILLA G, Antonio. REMIRO D, Fernando. (1993). Electrónica digital. Madrid: McGraw Hill. S/f. p.445.
- GAJSKI, Daniel D. (1997). Principios de diseño digital. Madrid: Prentice Hall Iberia. S/f. p.488. "Principles of digital design". Traducido por: Alberto Prieto Espinosa.
- LLORIS, Antonio. PRIETO, Alberto. (1996). Diseño lógico. Madrid: McGraw Hill. S/f. p.403.
- MANO, Morris. KIME, Charles. (1998). Fundamentos de diseño lógico y computadoras. México: Prentice Hall. Primera edición en español. P.604. "Logic and computer design fundamentals". Traducido por: Teresa Sanz Falcón.
- NEAMEN A, Donald. (1999). Análisis y diseño de circuitos electrónicos. Tomo II. México: McGraw Hill. S/f. p.1176. "Electronic circuit analysis and design". Traducido por: Felipe Castro Pérez.
- NELSON, V. NAGLE, H. CARROLL, B. IRWIN, J. (1996). Análisis y diseño de circuitos lógicos digitales. México: Prentice Hall. Primera edición. p.842. "Digital logic circuit analysis and design". Traducido por: Oscar A. Palmas V.
- TOCCI, Ronald. (1995). Sistemas digitales principios y aplicaciones. México: Prentice Hall. Quinta edición. p.823. "Digital systems principles and applications". Traducido por: Edmundo G. Urbina M.
- WARKELY, John F. (1997). Diseño digital principios y prácticas. México: Prentice Hall. S/f. p.743. "Digital design principles and practices". Traducido por: Gutiérrez R. Raymundo H.

CAPITULO 2.

2. ÁLGEBRA DE BOOLE Y COMPUERTAS.

Es un tipo de álgebra que tiene sus fundamentos en la Teoría de Conjuntos, sus variables solamente pueden tomar dos valores: cero “0” ó uno “1”. En el álgebra de Boole se define un conjunto $B = \{0,1\}$ donde cualquier variable $x \in B$ puede valer $x=0$ ó $x=1$. En la teoría de conjuntos, los valores de las variables también adquieren valores de pertenencia binaria (pertenece, o no pertenece); y sus postulados, al igual que cualquier estructura matemática, son las hipótesis de partida, aceptadas como verdaderas y sus respectivos consecuentes, demostrables a partir de su **sistema axiomático**. Los postulados y los teoremas pueden comprobarse sustituyendo las variables por los dos elementos del conjunto **B**.

Los postulados, también llamados **axiomas**, son relativos tanto al conjunto de elementos como a los operadores que se hayan definido en el sistema. Para el caso concreto del álgebra de Boole se pueden utilizar diferentes conjuntos de postulados. No obstante, el más utilizado es el propuesto por Huntington en 1904 que se detalla a continuación.

2.1 Teoremas y leyes del álgebra de Boole.

Primero se establece la relación de igualdad o equivalencia “=” para indicar que las dos variables **x** e **y**, pertenecientes al conjunto **B**, son iguales; por ejemplo, $x = y$.

I. Leyes de composición interna.

En **B** se definen dos leyes de composición interna, “+” (operador “O”, “OR”, o suma lógica) y “.” (operador “Y”, “AND”, multiplicación o producto lógico); siendo **B** cerrado para estas operaciones.

$$\forall x \in \mathbf{B}, \Rightarrow \text{a) } x + y \in \mathbf{B}$$

$$\text{b) } x \cdot y \in \mathbf{B}$$

El punto (\cdot), utilizado como símbolo para denotar el producto, no es indispensable, aunque no aparezca, se sobreentiende. Por lo tanto, la operación $x \cdot y \in \mathbf{B}$; se puede escribir de la forma: $x y \in \mathbf{B}$.

II. Elementos neutros.

Existen elementos neutros para ambas leyes de composición interna; las cuales son:

- a) Elemento neutro para la suma, $\exists 0 \in \mathbf{B} / \forall x \in \mathbf{B}, x + 0 = 0 + x = x$
- b) Elemento neutro para la multiplicación, $\exists 1 \in \mathbf{B} / \forall x \in \mathbf{B}, x \cdot 1 = 1 \cdot x = x$

III. Conmutatividad de las leyes de composición interna.

La suma y la multiplicación lógica son conmutativa; $\forall x, y \in \mathbf{B}$;

- a) $x + y = y + x$
- b) $x y = y x$

IV. Distributividad de las leyes de composición interna.

En el álgebra de Boole la suma y la multiplicación son distributivas recíprocamente.

$\forall x, y, z \in \mathbf{B}$;

- a) $x + (y z) = (x + y)(x + z)$
- b) $x(y + z) = x y + x z$

En el álgebra de los números reales, no se cumple el caso “a” de la distributividad.

V. Elemento opuesto.

Todo elemento de B tiene su opuesto (o función NOT). A este elemento se le denomina inverso, opuesto, complemento o negado. Se representa de varias formas, dos de ellas son: (\bar{x}, x') . La suma y el producto de una variable con su complemento da como resultado “1” y “0” respectivamente.

- $\forall x \in \mathbf{B}, \exists \bar{x} \in \mathbf{B}$
- a) $x + \bar{x} = 1$
 - b) $x \bar{x} = 0$

VI. Elementos del conjunto Booleano “B”.

Este postulado es muy obvio, sin embargo, se debe reglamentar; el postulado dice:

En **B** hay al menos dos elementos diferentes. $\exists x, y \in \mathbf{B} / x \neq y$. Los dos elementos distintos son “0” y “1”.

Con estos postulados se pueden demostrar las siguientes identidades del álgebra de Boole descritas en la tabla 2.1. Estas identidades también pueden ser demostradas mediante la teoría de conjuntos.

Suma Lógica	Multiplicación Lógica	Complemento
$0 + 0 = 0$	$0 \cdot 0 = 0$	$\overline{0} = 1$
$0 + 1 = 1$	$0 \cdot 1 = 0$	$\overline{1} = 0$
$1 + 0 = 1$	$1 \cdot 0 = 0$	$\overline{\overline{x}} = x$
$1 + 1 = 1$	$1 \cdot 1 = 1$	$\overline{\overline{\overline{x}}} = \overline{x}$
$x + 0 = x$	$x \cdot 0 = 0$	
$x + 1 = 1$	$x \cdot 1 = x$	
$x + x = x$	$x \cdot x = x$	
$x + \overline{x} = 1$	$x \cdot \overline{x} = 0$	

Tabla 2.1. Identidades del álgebra de Boole.

Principio de dualidad: En los postulados anteriores se observaron que las dos proposiciones (a y b) son duales y esto significa que se pueden obtener aplicando este principio: *si en una igualdad se sustituyen “0” por “1”, “+” por “.” y viceversa, en todos los lugares que aparezcan, se obtiene otra igualdad que se puede llamar “forma dual”.* Esto trae como consecuencia que cada teorema del álgebra de Boole tenga otro dual igualmente válido.

Teorema de absorción (T1):

$$\forall x, y \in \mathbf{B}; \quad a) x + x y = x$$

$$b) x(x + y) = x$$

Las dos partes del teorema son demostrables, no obstante, se demostrará la proposición “b”. Ya que incluye la demostración de la parte “a”.

Para demostrar (**P.d**):

$$\begin{aligned}
 &x(x + y) \\
 x(x + y) &= xx + xy && \text{Propiedad distributiva} \\
 &= x + xy && \text{Identidad de la multiplicación} \\
 &= x(1 + y) && \text{Identidad y propiedad distributiva (factor común)} \\
 &= x(1) && \text{Identidad de la suma} \\
 &= x && \text{Identidad de la multiplicación}
 \end{aligned}$$

Lo que se quería demostrar (**L.q.d**)

Teorema (T2):

$$\begin{aligned}
 \forall x, y \in B; & \quad a) x(\bar{x} + y) = xy \\
 & \quad b) x + \bar{x}y = x + y
 \end{aligned}$$

P.d: Se demostrará la parte **a**; la parte **b**, se deja para el lector.

$$\begin{aligned}
 &x(\bar{x} + y) \\
 x(\bar{x} + y) &= x\bar{x} + xy && \text{Propiedad distributiva} \\
 &= 0 + xy && \text{Identidad de la multiplicación} \\
 &= xy && \text{Identidad de la suma}
 \end{aligned}$$

L.q.d

Teorema (T3): El complemento de una variable existe, y es único.

$$\forall x, y \in \mathbf{B} / xy = 0 \wedge (x + y) = 1 \Rightarrow x \neq y \Rightarrow (x = \bar{y}) \vee (\bar{x} = y)$$

Para todo x, y que pertenezca a **B**, si se cumple que el producto de estas dos variables es cero, y la suma es igual a uno entonces, significa que dichas variables son diferentes; por lo tanto una es complemento de la otra. El símbolo \vee significa disyunción “o”; y el símbolo \wedge significa conjunción “y”. Las condiciones de las hipótesis en el producto y la suma conllevan a que la única forma de cumplir la proposición es cuando las dos variables son complementarias recíprocamente.

P.d:

$$\begin{array}{llll}
 \bar{\bar{x}} = x & \text{Identidad} & \bar{\bar{y}} = y & \text{Identidad} \\
 \bar{x} = \bar{x} + 0 & \text{Identidad de suma} & \bar{y} = \bar{y} + 0 & \text{Identidad de suma}
 \end{array}$$

$= \bar{x} + x y$	Por hipótesis	$= \bar{y} + x y$	Por hipótesis
$= (\bar{x} + x)(\bar{x} + y)$	Propiedad distributiva	$= (\bar{y} + y)(\bar{y} + x)$	Propiedad distributiva
$= (1)(\bar{x} + y)$	Identidad de suma	$= (1)(\bar{y} + x)$	Identidad de suma
$= (x + y)(\bar{x} + y)$	Por hipótesis	$= (x + y)(\bar{y} + x)$	Por hipótesis
$= y + (\bar{x} x)$	Propiedad distributiva	$= x + (\bar{y} y)$	Propiedad distributiva
$= y + (0)$	Identidad del producto	$= x + (0)$	Identidad del producto
$= y$	Elemento neutro	$= x$	Elemento neutro
$\bar{\bar{x}} = x$		$\bar{\bar{y}} = y$	
L.q.d		L.q.d	

Teorema (T4):

$\forall x, y \in \mathbf{B};$ a) $x y + x \bar{y} = x$
 b) $(x + y)(x + \bar{y}) = x$

P.d: (se demostrará la parte b)

$(x + y)(x + \bar{y}) = x + y \bar{y}$ Propiedad distributiva
 $= x + 0$ Identidad del producto
 $= x$ Identidad de la suma

L.q.d.

Teorema (T5):

$\forall x, y, z \in \mathbf{B};$ a) $x y + x \bar{y} z = x y + x z$
 b) $(x + y)(x + \bar{y} + z) = (x + y)(x + z)$

P.d: (se demostrará la parte a)

$x y + x \bar{y} z = x(y + \bar{y} z)$ Factor común
 $= x(y + z)$ Teorema 2
 $= x y + x z$ Propiedad distributiva

L.q.d.

Teorema (T6): Teorema de DeMorgan.

$$\forall x, y \in \mathbf{B}, \quad a) \overline{x + y} = \bar{x} \cdot \bar{y}$$

$$b) \overline{x \cdot y} = \bar{x} + \bar{y}$$

El teorema de la unicidad del complemento (T3) indica que se debe demostrar que los dos miembros de las igualdades (a) y (b) son complementarios. Por ejemplo, basta comprobar en (a) que $\bar{x} \cdot \bar{y}$ es el complemento de $x + y$. Por lo tanto, el producto de estos dos valores debe dar "0" y su suma debe dar "1". La aplicación de este teorema es muy importante en los circuitos de compuertas digitales.

$$aI) (x + y) + (\bar{x} \cdot \bar{y}) = 1$$

$$aII) (x + y) \cdot (\bar{x} \cdot \bar{y}) = 0$$

P.d: caso *aI*

$(x + y) + (\bar{x} \cdot \bar{y}) = \{(x + y) + \bar{x}\} \cdot \{(x + y) + \bar{y}\}$	Propiedad distributiva
$= \{\bar{x} + (x + y)\} \cdot \{x + (y + \bar{y})\}$	Propiedad conmutativa y asociativa
$= \{(\bar{x} + x) + y\} \cdot \{x + 1\}$	Propiedad asociativa e Identidad suma
$= (1 + y)(x + 1)$	Identidad de la suma
$= (1)(1)$	Identidad de la suma
$= 1$	Identidad de la multiplicación

P.d: caso *aII*

$(x + y) \cdot (\bar{x} \cdot \bar{y}) = \{x \cdot (\bar{x} \cdot \bar{y})\} + \{y \cdot (\bar{x} \cdot \bar{y})\}$	Propiedad distributiva
$= \{(x \cdot \bar{x}) \cdot \bar{y}\} + \{(y \cdot \bar{y}) \cdot \bar{x}\}$	Propiedad conmutativa y asociativa
$= (0 \cdot \bar{y}) + (0 \cdot \bar{x})$	Identidad de la multiplicación
$= (0) + (0)$	Identidad de la multiplicación
$= 0$	Identidad de la suma

L.q.d.

2.2 Compuertas básicas y universales.

Las compuertas básicas fueron nombradas en los postulados del álgebra de Boole; la ley de composición interna suma y multiplicación lógica (OR y AND), y el postulado **V** del elemento opuesto, que trata de la compuerta inversora NOT. Estas compuertas se denominan **básicas** porque, a través de ellas, se pueden desarrollar todos los circuitos digitales de lógica binaria. No obstante, la dificultad que se puede presentar está en los diseños de circuitos digitales grandes que necesitan combinaciones de compuertas básicas para efectuar una función lógica particular. Esta necesidad trajo como consecuencia la creación de otros tipos, llamadas **compuertas universales** que son el resultado de combinaciones de las tres compuertas básicas OR, AND y NOT. Por otra parte, mediante la conexión de compuertas universales, es posible lograr arreglos que funcionen igual a las compuertas básicas.

A continuación, la tabla 2.1 y 2.2, presentan los tipos de compuertas con su respectiva función lógica, símbolo, tabla de la verdad y circuito eléctrico equivalente.

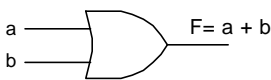
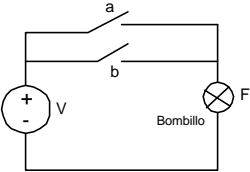
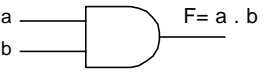
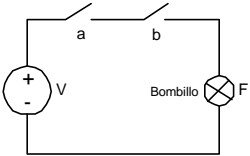
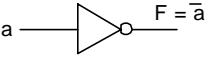
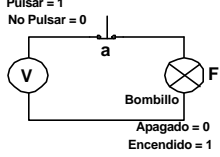
Función lógica	Símbolo	Tabla de la verdad	Circuito eléctrico equivalente															
OR $F(a,b) = a + b$		<table border="1" data-bbox="950 1220 1073 1341"> <thead> <tr> <th>a</th> <th>b</th> <th>F</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>0</td> </tr> <tr> <td>0</td> <td>1</td> <td>1</td> </tr> <tr> <td>1</td> <td>0</td> <td>1</td> </tr> <tr> <td>1</td> <td>1</td> <td>1</td> </tr> </tbody> </table>	a	b	F	0	0	0	0	1	1	1	0	1	1	1	1	
a	b	F																
0	0	0																
0	1	1																
1	0	1																
1	1	1																
AND $F(a,b) = a \cdot b$		<table border="1" data-bbox="950 1430 1073 1551"> <thead> <tr> <th>a</th> <th>b</th> <th>F</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>0</td> </tr> <tr> <td>0</td> <td>1</td> <td>0</td> </tr> <tr> <td>1</td> <td>0</td> <td>0</td> </tr> <tr> <td>1</td> <td>1</td> <td>1</td> </tr> </tbody> </table>	a	b	F	0	0	0	0	1	0	1	0	0	1	1	1	
a	b	F																
0	0	0																
0	1	0																
1	0	0																
1	1	1																
NOT $F(a) = \bar{a}$		<table border="1" data-bbox="976 1661 1052 1732"> <thead> <tr> <th>a</th> <th>F</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>1</td> </tr> <tr> <td>1</td> <td>0</td> </tr> </tbody> </table>	a	F	0	1	1	0	 <p data-bbox="1203 1675 1421 1837"> Pulsar = 1 No Pulsar = 0 Apagado = 0 Encendido = 1 </p>									
a	F																	
0	1																	
1	0																	

Tabla 2.1. Compuertas básicas y sus circuitos eléctrico y electrónico equivalentes.

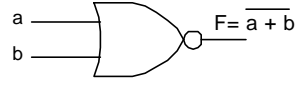
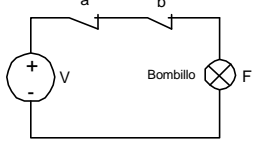
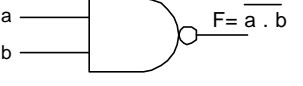
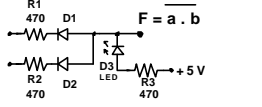
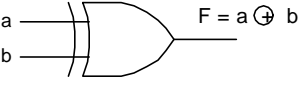
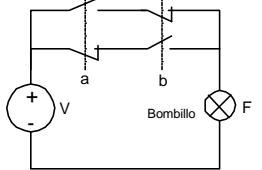
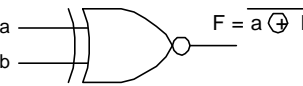
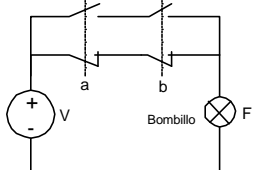
Función lógica	Símbolo	Tabla de la verdad	Circuito eléctrico equivalente															
NOR $F(a,b) = \overline{a + b}$		<table border="1" data-bbox="966 451 1088 577"> <thead> <tr> <th>a</th> <th>b</th> <th>F</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>1</td> </tr> <tr> <td>0</td> <td>1</td> <td>0</td> </tr> <tr> <td>1</td> <td>0</td> <td>0</td> </tr> <tr> <td>1</td> <td>1</td> <td>0</td> </tr> </tbody> </table>	a	b	F	0	0	1	0	1	0	1	0	0	1	1	0	
a	b	F																
0	0	1																
0	1	0																
1	0	0																
1	1	0																
NAND $F(a,b) = \overline{a \cdot b}$		<table border="1" data-bbox="966 661 1088 787"> <thead> <tr> <th>a</th> <th>b</th> <th>F</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>1</td> </tr> <tr> <td>0</td> <td>1</td> <td>1</td> </tr> <tr> <td>1</td> <td>0</td> <td>1</td> </tr> <tr> <td>1</td> <td>1</td> <td>0</td> </tr> </tbody> </table>	a	b	F	0	0	1	0	1	1	1	0	1	1	1	0	
a	b	F																
0	0	1																
0	1	1																
1	0	1																
1	1	0																
XOR $F(a,b) = \overline{a} \cdot b + a \cdot \overline{b}$ $F(a,b) = a \oplus b$		<table border="1" data-bbox="966 871 1088 997"> <thead> <tr> <th>a</th> <th>b</th> <th>F</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>0</td> </tr> <tr> <td>0</td> <td>1</td> <td>1</td> </tr> <tr> <td>1</td> <td>0</td> <td>1</td> </tr> <tr> <td>1</td> <td>1</td> <td>0</td> </tr> </tbody> </table>	a	b	F	0	0	0	0	1	1	1	0	1	1	1	0	
a	b	F																
0	0	0																
0	1	1																
1	0	1																
1	1	0																
XNOR $F(a,b) = \overline{a} \cdot \overline{b} + a \cdot b$ $F(a,b) = \overline{a \oplus b}$		<table border="1" data-bbox="966 1123 1088 1249"> <thead> <tr> <th>a</th> <th>b</th> <th>F</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>1</td> </tr> <tr> <td>0</td> <td>1</td> <td>0</td> </tr> <tr> <td>1</td> <td>0</td> <td>0</td> </tr> <tr> <td>1</td> <td>1</td> <td>1</td> </tr> </tbody> </table>	a	b	F	0	0	1	0	1	0	1	0	0	1	1	1	
a	b	F																
0	0	1																
0	1	0																
1	0	0																
1	1	1																

Tabla 2.2. Compuertas universales y sus circuitos eléctricos y electrónicos equivalentes.

2.2.1 Arreglos equivalentes entre las compuertas universales y básicas.

Las compuertas NAND y NOR son universales; esto significa que, realizando arreglos con ellas, se pueden obtener todas las configuraciones de compuertas básicas y también, configuraciones de compuertas XOR y XNOR. Esto está sustentado en el teorema de DeMorgan, los teoremas del álgebra de Boole y el principio de identidad donde la doble negación, de una función, es equivalente a la misma función. Del mismo modo, las variables de una función lógica pueden ser sustituidas por una sola variable equivalente (principio de sustitución), también puede generalizarse para n variables.

$$F(x_1, x_2, \dots, x_n) = \overline{\overline{F(x_1, x_2, \dots, x_n)}} \quad \text{Ec. 2.1}$$

Las tablas 2.3 y 2.4 presentan los circuitos equivalentes de compuertas básicas realizados con NOR y NAND respectivamente.

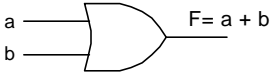
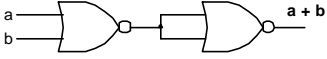
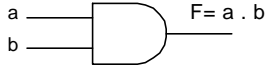
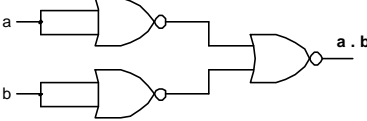
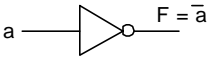
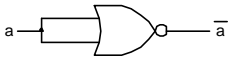
Función lógica	Símbolo	Circuito eléctrico equivalente NOR
OR $F(a,b) = a + b$		
AND $F(a,b) = a \cdot b$		
NOT $F(a) = \bar{a}$		

Tabla 2.3. Circuitos equivalentes OR, AND y NOT realizados con compuertas universales NOR.

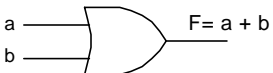
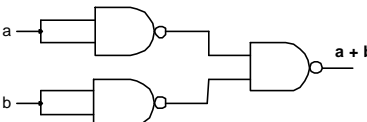
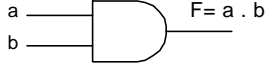
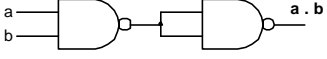
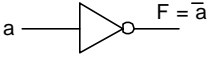
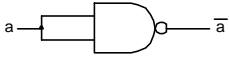
Función lógica	Símbolo	Circuito eléctrico equivalente NAND
OR $F(a,b) = a + b$		
AND $F(a,b) = a \cdot b$		
NOT $F(a) = \bar{a}$		

Tabla 2.4. Circuitos equivalentes OR, AND y NOT realizados con compuertas universales NAND.

Las demostraciones de estas equivalencias se realizan a continuación:

Función NOT (realización con compuertas **NOR**);

$$F(a) = \bar{a} \quad \text{Función original}$$

$$F(a) = \overline{a + a} \quad \text{Unión de las dos entradas NOR y conectadas a la entrada "a"}$$

$$F(a) = \bar{a} \quad \text{Identidad de la suma}$$

Función OR (realización con compuertas **NOR**);

$$F(a,b) = a + b \quad \text{Función original}$$

$$F(a,b) = \overline{\overline{a + b}} = \overline{\overline{a + b}} \quad \text{Función equivalente con la doble negación (dos NOR)}$$

Función AND (realización con compuertas **NOR**);

$$F(a,b) = a \cdot b \quad \text{Función original}$$

$$F(a,b) = \overline{\overline{a \cdot b}} \quad \text{Función equivalente con la doble negación}$$

$$F(a,b) = \overline{\overline{a + b}} \quad \text{Teorema de DeMorgan}$$

Las demostraciones de estas equivalencias se realizan a continuación:

Función NOT (realización con compuertas **NAND**);

$$F(a) = \bar{a} \quad \text{Función original}$$

$$F(a) = \overline{a \cdot a} \quad \text{Unión de las dos entradas de la NAND y conectadas en "a"}$$

$$F(a) = \bar{a} \quad \text{Identidad de la suma}$$

Función OR (realización con compuertas **NAND**);

$$F(a,b) = a + b \quad \text{Función original}$$

$$F(a,b) = \overline{\overline{a + b}} \quad \text{Función equivalente con la doble negación}$$

$$F(a,b) = \overline{\overline{a \cdot b}} \quad \text{Teorema de DeMorgan}$$

Función AND (realización con compuertas **NAND**);

$$F(a,b) = a \cdot b \quad \text{Función original}$$

$$F(a,b) = \overline{\overline{a \cdot b}} = \overline{\overline{a} \cdot \overline{b}} \quad \text{Función equivalent e con la doble negación; (dos NAND)}$$

2.3 Simplificación de funciones de conmutación.

Los teoremas y leyes del álgebra de Boole permiten realizar simplificaciones en circuitos digitales con la finalidad de minimizar el costo y tamaño del mismo. La tarea fundamental consiste en reducir el número de términos de la función lógica, y de esta forma minimizar las compuertas utilizadas en el diseño sin cambiar el **funcionamiento combinacional digital**.

Las funciones que se tratarán en este tema tienen una sola salida y son de tipo **combinacional**. Esto último significa que la salida de la función debe ser la misma para una entrada en particular, y mantener siempre la misma condición de salida, siempre y cuando se produzca esa combinación en las variables de entrada. Otra forma de decirlo es: "No pueden haber dos niveles lógicos distintos en la salida de la función; para la misma combinación de las variables de entrada".

Del mismo modo, las funciones de conmutación, que son la base del diseño de los circuitos digitales combinacionales, deben cumplir con la siguiente condición: para **n** variables de entrada deben haber, a lo sumo, **2ⁿ** combinaciones distintas de productos y/o sumas que incluyan todas las variables de entrada del circuito. A continuación se presentan algunos ejercicios para minimizar funciones lógicas aplicando las leyes y los teoremas del álgebra de Boole.

Ejercicio 2.1. Simplificar la siguiente función lógica: $F(X, Y, Z) = \overline{X} \overline{Y} \overline{Z} + \overline{Y} Z + X \overline{Y}$

$$\begin{aligned} \overline{X} \overline{Y} \overline{Z} + \overline{Y} Z + X \overline{Y} &= \overline{Y} (\overline{X} \overline{Z} + Z + X) && \text{Factor común} \\ &= \overline{Y} (\overline{X} + Z + X) && \text{T2} \\ &= \overline{Y} ((\overline{X} + X) + Z) && \text{P. asociativa y conmutativa} \\ &= \overline{Y} (1 + Z) && \text{Identidad de suma} \\ &= \overline{Y} (1) && \text{Identidad de suma} \\ &= \overline{Y} && \text{Identidad del producto} \end{aligned}$$

Ejercicio 2.2. Simplificar la siguiente función lógica: $F(a,b,c,d) = a\bar{b} + \bar{c}d + \bar{a} + d$

$$\begin{aligned} a\bar{b} + \bar{c}d + \bar{a} + d &= a\bar{b} + \bar{a} + \bar{c}d + d && \text{Propiedad conmutativa} \\ &= a\bar{b} + \bar{a} + d && \text{Teorema de absorción (T1)} \\ &= \bar{b} + \bar{a} + d && \text{T2} \end{aligned}$$

Ejercicio 2.3. Simplificar la siguiente función lógica:

$$\begin{aligned} F(a,b,c) &= a+b+c \cdot \overline{(a+\bar{b})} + (a+\bar{b}) \cdot (a+\bar{b}+c) \\ &= a+b+c \cdot \overline{(a+\bar{b})} + (a+\bar{b}) \cdot (a+\bar{b}+c) \\ &= a+b+c \cdot (\bar{a} \cdot b) + (a+\bar{b}) \cdot (a+\bar{b}+c) && \text{T. DeMorgan} \\ &= a+b+\bar{a} \cdot b \cdot c + (a+\bar{b}) \cdot (a+\bar{b}+c) && \text{P. Conmutativa} \\ &= a+b+(a+\bar{b}) && \text{T. de Absorción aplicado 2 veces} \\ &= (a+a)+(b+\bar{b}) && \text{P. Asociativa} \\ &= a+1 && \text{Identidad de la suma} \\ &= 1 && \text{Identidad de la suma} \end{aligned}$$

Ejercicio 2.4. Simplificar la siguiente función lógica:

$$\begin{aligned} F(A,B,C,D) &= \bar{A} \cdot \bar{B} \cdot \bar{C} \cdot D + \bar{A} \cdot \bar{B} \cdot C \cdot \bar{D} + \bar{A} \cdot B \cdot \bar{C} \cdot \bar{D} + \bar{A} \cdot B \cdot C \cdot D + A \cdot \bar{B} \cdot \bar{C} \cdot \bar{D} \\ &\quad + A \cdot \bar{B} \cdot C \cdot D + A \cdot B \cdot \bar{C} \cdot D + A \cdot B \cdot C \cdot \bar{D} \\ &= \bar{A} \cdot \bar{B} \cdot (\bar{C} \cdot D + C \cdot \bar{D}) + \bar{A} \cdot B \cdot (\bar{C} \cdot \bar{D} + C \cdot D) + A \cdot \bar{B} \cdot (\bar{C} \cdot \bar{D} + C \cdot D) \\ &\quad + A \cdot B \cdot (\bar{C} \cdot D + C \cdot \bar{D}) \\ &= (\bar{A} \cdot \bar{B} + A \cdot B) \cdot (\bar{C} \cdot D + C \cdot \bar{D}) + (\bar{A} \cdot B + A \cdot \bar{B}) \cdot (\bar{C} \cdot \bar{D} + C \cdot D) \end{aligned}$$

Realizando la sustitución: $X = (\bar{A} \cdot B + A \cdot \bar{B})$; $Y = (\bar{C} \cdot D + C \cdot \bar{D})$

$$\begin{aligned} &= \bar{X} \cdot Y + X \cdot \bar{Y} = X \oplus Y \\ &= (\bar{A} \cdot B + A \cdot \bar{B}) \oplus (\bar{C} \cdot D + C \cdot \bar{D}) \\ &= (A \oplus B) \oplus (C \oplus D) \end{aligned}$$

La función resultante es una agrupación de compuertas OR Exclusivas (XOR) cuya expresión básica es: $\bar{A} \cdot B + A \cdot \bar{B}$ ó $\bar{C} \cdot D + C \cdot \bar{D}$

Para facilitar la simplificación, en ocasiones, es necesario repetir uno o más términos de la función lógica con la finalidad de reducir una variable. También, hay casos donde se puede realizar la ampliación del número de variables de una función para obtener un resultado minimizado.

Ejercicio 2.5. Simplificar la siguiente función lógica:

$$F(m,n,p,q) = m \cdot n \cdot p \cdot \bar{q} + m \cdot \bar{n} \cdot p \cdot q + m \cdot n \cdot p \cdot q + \bar{m} \cdot n \cdot p \cdot q + m \cdot n \cdot \bar{p} \cdot q$$

La simplificación se puede realizar repitiendo el término $m \cdot n \cdot p \cdot q$ y sacando factor común.

$$\begin{aligned} &= m \cdot n \cdot p \cdot \bar{q} + m \cdot n \cdot p \cdot q + m \cdot \bar{n} \cdot p \cdot q + m \cdot n \cdot p \cdot q + \bar{m} \cdot n \cdot p \cdot q + m \cdot n \cdot p \cdot q \\ &\quad + m \cdot n \cdot \bar{p} \cdot q + m \cdot n \cdot p \cdot q \\ &= m \cdot n \cdot p \cdot (\bar{q} + q) + m \cdot p \cdot q \cdot (\bar{n} + n) + n \cdot p \cdot q \cdot (\bar{m} + m) + m \cdot n \cdot q \cdot (\bar{p} + p) \\ &= m \cdot n \cdot p \cdot (1) + m \cdot p \cdot q \cdot (1) + n \cdot p \cdot q \cdot (1) + m \cdot n \cdot q \cdot (1) \\ &= m \cdot n \cdot p + m \cdot p \cdot q + n \cdot p \cdot q + m \cdot n \cdot q \end{aligned}$$

Ejercicio 2.6. Demostrar el teorema de consenso (T7):

$$I) m \cdot n + \bar{m} \cdot p + n \cdot p = m \cdot n + \bar{m} \cdot p$$

$$II) (m+n) \cdot (\bar{m}+p) \cdot (n+p) = (m+n) \cdot (\bar{m}+p)$$

P.d: (parte I); La simplificación se puede realizar expandiendo el término $n \cdot p$

$m \cdot n + \bar{m} \cdot p + n \cdot p = m \cdot n + \bar{m} \cdot p + (1) \cdot n \cdot p$	Identidad Producto
$= m \cdot n + \bar{m} \cdot p + (m + \bar{m}) \cdot n \cdot p$	Identidad suma
$= m \cdot n + \bar{m} \cdot p + m \cdot n \cdot p + \bar{m} \cdot n \cdot p$	P. distributiva
$= m \cdot n + m \cdot n \cdot p + \bar{m} \cdot p + \bar{m} \cdot n \cdot p$	P. conmutativa
$= (m \cdot n + m \cdot n \cdot p) + (\bar{m} \cdot p + \bar{m} \cdot n \cdot p)$	P. asociativa
$= m \cdot n + \bar{m} \cdot p$	T. de absorción

L.q.d

2.3.1 Formas canónicas de las funciones de conmutación.

Las funciones de conmutación deben ser expresadas con todos los términos y variables que intervienen en el circuito digital combinacional. Por lo cual, los diferentes procedimientos de simplificación que se verán en este trabajo parten normalmente de expresiones o formas canónicas de las funciones de conmutación. Para llegar a esas formas de representación hay que introducir, previamente, algunas definiciones básicas.

Un **literal**, se define como la variable de una función de conmutación que puede ser complementada ($\bar{a}, \bar{x}, \bar{y}$), o no complementada (a, x, y). Las funciones son expresadas por uno o más términos; un **término producto** es una serie de literales conectados por el operador **Y**, (\cdot) o (**AND**). Un **término suma** es una serie de literales conectados por el operador **O**, ($+$) u (**OR**).

Cualquier término producto o suma, es **normal**, si en el mismo, no se repiten dos o más veces las mismas variables, complementadas o no complementadas. Además, el producto o suma debe tener todas las variables de entrada de la función lógica. Por ejemplo, la función: $F(a,b,c) = (\bar{c} + b + \bar{a}) + abc\bar{b} + (\bar{a} + b + a + \bar{c}b)\bar{b} + \bar{b}\bar{a}\bar{c}$

Solo tiene un producto normal ($\bar{b}\bar{a}\bar{c}$); y una suma normal ($\bar{c} + b + \bar{a}$).

También se define una regla de sustitución de literales de las funciones lógicas, Booleanas o de conmutación, esto significa que si $F(x_1, x_2, \dots, x_{n-1}, x_n)$ es una función de conmutación entonces el resultado de la evaluación de los literales con "0" y "1" dan como resultado una salida de la función igual a "0" o "1". Esto implica que hay dos resultados posibles que pueden ser aceptados sin demostración, y por ende, también pueden ser asignados a otra variable Booleana: $a = F(x_1, x_2, \dots, x_n)$.

Dado un conjunto de variables, x_1, x_2, \dots, x_n ; un "**minterms**" (término mínimo) llamado producto fundamental canónico, es cualquier término producto normal en el que aparecen todas las variables de la función. Por ejemplo, $F(a,b,c) = a\bar{b}\bar{c} + abc$ posee dos minterms y hacen que la función de conmutación valga "1".

Cuando se establece un orden de posición a las variables, se hace una relación equivalente entre la variable complementada con el cero lógico "0" y, la variable no complementada con el "1". Esto se aprovecha para obtener el valor decimal del número binario correspondiente a la expresión vectorial (ordenada) de cada minterm, y ese

valor decimal se puede utilizar para representar cada minterm de la función. En el ejemplo anterior: $a\bar{b}\bar{c}$ se le asigna el valor binario 100_2 lo que significa que se trata del minterm decimal (m_4). La primera variable que aparece en la función se considera la más significativa, esto es una regla relativa; sin embargo, es la que se utilizará en el texto. También existe otra forma de representación estándar, la cual se expone a continuación.

Dado un conjunto de variables, x_1, x_2, \dots, x_n ; un “**maxterms**” (término máximo) llamado suma fundamental canónica, es cualquier término suma normal en el que aparecen todas las variables de la función. Por ejemplo, $F(x, y, z) = (\bar{x} + \bar{y} + \bar{z})(x + \bar{y} + z)$ posee dos maxterms y hacen que la función de conmutación valga “0”.

Cuando se establece un orden de posición a las variables, se hace una relación equivalente entre la variable complementada con el uno lógico “1” y, la variable no complementada con el “0”. Esto se aprovecha para obtener el valor decimal del número binario correspondiente a la expresión vectorial (ordenada) de cada maxterms, y ese valor decimal se puede utilizar para representar cada maxterms de la función. En el ejemplo anterior: $(x + \bar{y} + z)$ se le asigna el valor binario 010_2 lo que significa que se trata del maxterms decimal (M_2).

La **forma canónica minterms** se obtiene sumando los términos productos (suma de productos) normales con los literales de la función de conmutación y, la **forma canónica maxterms** se obtiene realizando el producto de los términos sumas (producto de sumas) normales con los literales de la función de conmutación. Cada variable puede tomar dos valores “0” o “1”, significa que en cada forma canónica, para n variables deben haber 2^n minterms y/o maxterms. Se puede demostrar que en una función de conmutación el número de minterms es igual a $(2^n - \text{número de maxterms})$, y viceversa, el número de maxterms es igual a $(2^n - \text{número de minterms})$.

La formalización de los *maxterms* y *minterms*, con respecto a la asignación de los ceros y unos de los literales; se demuestran mediante el desarrollo de Shannon. El mismo, parte de la equivalencia entre las funciones constantes $\{f(0)$ y $f(1)\}$ con respecto a cualquier función de conmutación de una variable $f(x_1)$ y, de n variables $f(x_1, x_2, \dots, x_n)$.

Desarrollo de Shannon para demostrar la relación de equivalencia entre las variables de la forma canónica minterms con respecto a la asignación de ceros “0” y unos “1”. Cualquier función simple $F(x)$ se puede desarrollar de la forma:

$$F(x) = \bar{x} \cdot F(0) + x \cdot F(1) \quad \text{Ec. 2.2}$$

Los valores lógicos 0 y 1 que la variable x puede tener: $x = \{0, 1\} \Rightarrow \bar{x} = \{1, 0\}$

$$\text{Para } x=0; F(0) = \bar{0} \cdot F(0) + 0 \cdot F(1) = 1 \cdot F(0) + 0 = F(0)$$

$$\text{Para } x=1; F(1) = \bar{1} \cdot F(0) + 1 \cdot F(1) = 0 \cdot F(0) + 1 \cdot F(1) = 0 + F(1) = F(1)$$

Por lo cual, los dos resultados, están de acuerdo con la Ec.2.2; ahora se va a demostrar el mismo desarrollo para n variables:

$$F(x_1, x_2, \dots, x_j, \dots, x_n) = \bar{x}_j \cdot F(x_1, x_2, \dots, 0, \dots, x_n) + x_j \cdot F(x_1, x_2, \dots, 1, \dots, x_n) \quad \text{Ec. 2.3}$$

De la misma forma que el anterior se demuestra que:

$$F(x_1, x_2, \dots, 0, \dots, x_n) = 1 \cdot F(x_1, x_2, \dots, 0, \dots, x_n) + 0 \cdot F(x_1, x_2, \dots, 1, \dots, x_n) = F(x_1, x_2, \dots, 0, \dots, x_n)$$

$$F(x_1, x_2, \dots, 1, \dots, x_n) = 0 \cdot F(x_1, x_2, \dots, 0, \dots, x_n) + 1 \cdot F(x_1, x_2, \dots, 1, \dots, x_n) = F(x_1, x_2, \dots, 1, \dots, x_n)$$

Teorema 7.1: Toda función de conmutación se puede expresar como una suma única de *minterms*.

Tomando la primera variable se tiene que:

$$F(x_1, x_2, \dots, x_n) = \bar{x}_1 \cdot F(0, x_2, \dots, x_n) + x_1 \cdot F(1, x_2, \dots, x_n)$$

Ahora tomando dos, tres y n variable se tiene que:

$$F(x_1, x_2, \dots, x_n) = \bar{x}_1 \cdot [\bar{x}_2 \cdot F(0, 0, \dots, x_n) + x_2 \cdot F(0, 1, \dots, x_n)]$$

$$+ x_1 \cdot [\bar{x}_2 \cdot F(1, 0, \dots, x_n) + x_2 \cdot F(1, 1, \dots, x_n)]$$

$$F(x_1, x_2, \dots, x_n) = \bar{x}_1 \cdot \bar{x}_2 \cdot F(0, 0, \dots, x_n) + \bar{x}_1 \cdot x_2 \cdot F(0, 1, \dots, x_n) + x_1 \cdot \bar{x}_2 \cdot F(1, 0, \dots, x_n)$$

$$+ x_1 \cdot x_2 \cdot F(1, 1, \dots, x_n)$$

$$F(x_1, x_2, \dots, x_n) = \bar{x}_1 \cdot \bar{x}_2 \cdot \bar{x}_3 \cdot \dots \cdot \bar{x}_n \cdot F(0, 0, 0, \dots, 0) + \dots + x_1 \cdot x_2 \cdot x_3 \cdot \dots \cdot x_n \cdot F(1, 1, 1, \dots, 1)$$

Aquí aparecen 2^n productos normales que se suman, (*minterms*) y la relación del “0” con la variable complementada y “1” con la variable sin complementar. De forma similar, se pueden escribir y demostrar los desarrollos siguientes para la forma *maxterms*:

$$F(x) = [x + F(0)] \cdot [\bar{x} + F(1)] \quad \text{Ec. 2.4}$$

$$F(0) = [0 + F(0)] \cdot [1 + F(1)] = F(0) \cdot 1 = F(0)$$

$$F(1) = [1 + F(0)] \cdot [0 + F(1)] = 1 \cdot F(1) = F(1)$$

Para n variables se demuestra que:

$$F(x_1, x_2, \dots, x_j, \dots, x_n) = [x_j + F(x_1, x_2, \dots, 0, \dots, x_n)] \cdot [\bar{x}_j + F(x_1, x_2, \dots, 1, \dots, x_n)] \quad \text{Ec. 2.5}$$

Con los valores 0 y 1 que la variable x puede tener: $x = \begin{cases} 0 \\ 1 \end{cases} \Rightarrow \bar{x} = \begin{cases} 1 \\ 0 \end{cases}$

$$F(x_1, x_2, \dots, 0, \dots, x_n) = [0 + F(x_1, x_2, \dots, 0, \dots, x_n)] \cdot [1 + F(x_1, x_2, \dots, 1, \dots, x_n)]$$

$$F(x_1, x_2, \dots, 0, \dots, x_n) = F(x_1, x_2, \dots, 0, \dots, x_n) \cdot (1) = F(x_1, x_2, \dots, 0, \dots, x_n)$$

$$F(x_1, x_2, \dots, 1, \dots, x_n) = [1 + F(x_1, x_2, \dots, 0, \dots, x_n)] \cdot [0 + F(x_1, x_2, \dots, 1, \dots, x_n)]$$

$$F(x_1, x_2, \dots, 1, \dots, x_n) = (1) \cdot F(x_1, x_2, \dots, 1, \dots, x_n) = F(x_1, x_2, \dots, 1, \dots, x_n)$$

Teorema 7.2: Toda función de conmutación se puede expresar como un producto único de *maxterms*.

Tomando la primera variable se tiene que:

$$F(x_1, x_2, \dots, x_n) = [x_1 + F(0, x_2, \dots, x_n)] \cdot [\bar{x}_1 + F(1, x_2, \dots, x_n)]$$

Ahora tomando dos, tres y n variables se tiene que:

$$F(x_1, x_2, \dots, x_n) = \{x_1 + [x_2 + F(0, 0, \dots, x_n)] \cdot [\bar{x}_2 + F(0, 1, \dots, x_n)]\} \\ \cdot \{\bar{x}_1 + [x_2 + F(1, 0, \dots, x_n)] \cdot [\bar{x}_2 + F(1, 1, \dots, x_n)]\}$$

$$F(x_1, x_2, \dots, x_n) = [(x_1 + x_2) + F(0, 0, \dots, x_n)] \cdot [(x_1 + \bar{x}_2) + F(0, 1, \dots, x_n)] \\ \cdot [(\bar{x}_1 + x_2) + F(1, 0, \dots, x_n)] \cdot [(\bar{x}_1 + \bar{x}_2) + F(1, 1, \dots, x_n)]$$

$$F(x_1, x_2, \dots, x_n) = [(x_1 + x_2 + x_3 + \dots + x_n) + F(0, 0, 0, \dots, 0)] \cdot \dots \cdot [(\bar{x}_1 + \bar{x}_2 + \bar{x}_3 + \dots + \bar{x}_n) + F(1, 1, 1, \dots, 1)]$$

Aquí aparecen 2^n sumas normales que se multiplican, (maxterms) y la relación del “1” con la variable complementada y “0” con la variable sin complementar.

Cada **producto minterm** pertenece a la función de conmutación si la función constante $F(0, 0, \dots, 0)$ o $F(1, 1, \dots, 1)$ se hace uno “1”. De lo contrario, si estas funciones constantes valen cero, entonces el minterm se anulará.

Cada **producto maxterm** pertenece a la función de conmutación si la función constante $F(0,0,\dots,0)$ o $F(1,1,\dots,1)$ se hace cero "0". De lo contrario, si estas funciones constantes valen uno, entonces el maxterm se anulará.

Las funciones también se pueden representar en forma explícita mediante **tablas de la verdad**, que incluyen todas las combinaciones normales posibles de los literales; y las asignaciones que se establecen en las formas canónicas maxterms y minterms. Este tipo de representación es la mas completa de todas y, como se verá más adelante, sirve de base para el diseño y realización de circuitos digitales. La tabla 2.5 muestra en forma explícita la función: $F(a,b,c) = \bar{b}\bar{c} + b(a+c)$, la tabla se puede llenar mediante la evaluación, con "0" y "1", de las variables de la función lógica. También se puede realizar mediante la expansión, de la función, en sus formas canónicas. Es de hacer notar, que se puede llenar con cualquiera de los criterios (minterms y maxterms). No obstante, la forma minterms resulta más sencilla para evaluar la función y sus literales.

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

$$F(0,0,0) = \bar{0}\bar{0} + 0\cdot(0+0) = 1\cdot 1 + 0\cdot(0+0) = 1 + 0\cdot 0 = 1 + 0 = 1$$

$$F(0,0,1) = \bar{0}\bar{1} + 0\cdot(0+1) = 1\cdot 0 + 0\cdot(0+1) = 0 + 0\cdot 1 = 0 + 0 = 0$$

$$F(0,1,0) = \bar{1}\bar{0} + 1\cdot(0+0) = 0\cdot 1 + 1\cdot(0+0) = 0 + 1\cdot 0 = 0 + 0 = 0$$

$$F(0,1,1) = \bar{1}\bar{1} + 1\cdot(0+1) = 0\cdot 0 + 1\cdot(0+1) = 0 + 1\cdot 1 = 0 + 1 = 1$$

$$F(1,0,0) = \bar{0}\bar{0} + 0\cdot(1+0) = 1\cdot 1 + 0\cdot(1+0) = 1 + 0\cdot 1 = 1 + 0 = 1$$

$$F(1,0,1) = \bar{0}\bar{1} + 0\cdot(1+1) = 1\cdot 0 + 0\cdot(1+1) = 0 + 0\cdot 1 = 0 + 0 = 0$$

$$F(1,1,0) = \bar{1}\bar{0} + 1\cdot(1+0) = 0\cdot 1 + 1\cdot(1+0) = 0 + 1\cdot 1 = 0 + 1 = 1$$

$$F(1,1,1) = \bar{1}\bar{1} + 1\cdot(1+1) = 0\cdot 0 + 1\cdot(1+1) = 0 + 1\cdot 1 = 0 + 1 = 1$$

Los ceros "0" de la función corresponden con los maxterms y los unos "1" con los minterms. Del mismo modo, ambos son complementarios con respecto a las **n** variables (3 variables: a, b, c) por lo cual, existen, $2^3=8$ términos. En la tabla 2.5 los minterms son: m_0, m_3, m_4, m_6 y m_7 . Los maxterms son M_1, M_2, M_5 .

m_j	a	b	c	F	m_0	M_1	M_2	m_3	m_4	M_5	m_6	m_7
0	0	0	0	1	1	1	1	0	0	1	0	0
1	0	0	1	0	0	0	1	0	0	1	0	0
2	0	1	0	0	0	1	0	0	0	1	0	0
3	0	1	1	1	0	1	1	1	0	1	0	0
4	1	0	0	1	0	1	1	0	1	1	0	0
5	1	0	1	0	0	1	1	0	0	0	0	0
6	1	1	0	1	0	1	1	0	0	1	1	0
7	1	1	1	1	0	1	1	0	0	1	0	1

Tabla 2.5. Tabla de la verdad de la función: $F(a,b,c) = \bar{b}\bar{c} + b(a+c)$

La tabla 2.5 muestra los términos (minterms y maxterms) activos por columna. Por ejemplo, el minterm m_6 está activo, por lo que hay un solo uno "1" en la columna respectiva, todos los demás son cero "0"; por esto recibe el nombre de número de términos mínimos. A diferencia del maxterm (número de términos máximos), por ejemplo M_1 , en el cual todos son unos "1" excepto él mismo que vale cero "0".

También, mediante la expansión de funciones se pueden obtener las formas canónicas:

$$\begin{aligned}
 F(a,b,c) &= \bar{b}\bar{c} + b(a+c) \\
 &= 1 \cdot \bar{b}\bar{c} + ab + bc = (a + \bar{a}) \cdot \bar{b}\bar{c} + ab + bc = a\bar{b}\bar{c} + \bar{a}\bar{b}\bar{c} + ab + bc \\
 &= a\bar{b}\bar{c} + \bar{a}\bar{b}\bar{c} + 1 \cdot ab + 1 \cdot bc = a\bar{b}\bar{c} + \bar{a}\bar{b}\bar{c} + (c + \bar{c}) \cdot ab + (a + \bar{a}) \cdot bc \\
 &= a\bar{b}\bar{c} + \bar{a}\bar{b}\bar{c} + abc + ab\bar{c} + abc + \bar{a}bc
 \end{aligned}$$

Los minterms que se repiten, se escriben una sola vez y se obtiene:

$$F(a,b,c) = a\bar{b}\bar{c} + \bar{a}\bar{b}\bar{c} + abc + ab\bar{c} + \bar{a}bc \quad \text{Forma canónica algebraica minterms.}$$

Existen otras dos formas de representación de las formas canónicas que se obtienen evaluando ordenadamente (a, más significativa) con los criterios de los minterms; "0" para la variable complementada y "1" para la variable sin complementar.

$$F(a,b,c) = m_4 + m_0 + m_7 + m_6 + m_3$$

$$F(a,b,c) = m_0 + m_3 + m_4 + m_6 + m_7 \quad \text{Forma canónica compacta minterms.}$$

$$F(a,b,c) = \sum_m (0,3,4,6,7) \quad \text{Forma canónica lista de minterms.}$$

Los maxterms se obtienen con los términos faltantes de algunas de las tres formas (algebraica, compacta o lista), o directamente con los ceros de la función de conmutación en la tabla de la verdad. Por lo tanto, la forma canónica maxterms queda:

$$F(a,b,c) = \prod_M (1,2,5) \quad \text{Forma canónica lista de maxterms.}$$

$$F(a,b,c) = M_1 \cdot M_2 \cdot M_5 \quad \text{Forma canónica compacta maxterms.}$$

$$F(a,b,c) = (a+b+\bar{c}) \cdot (a+\bar{b}+c) \cdot (\bar{a}+b+\bar{c}) \quad \text{Forma canónica algebraica maxterms.}$$

2.4 Diseño, simulación y síntesis de circuitos digitales.

El diseño de un circuito digital de compuertas comienza cuando se expone el planteamiento de un problema; aquí, las señales de entrada y salida deben ser de tipo digital (on-off). El proyecto debe ser realizable y los parámetros para medir esto último no existen; sin embargo, la experiencia como diseñador es la que prevalece a la hora de distinguir si un proyecto o diseño digital es realizable. Además de esto, el diseñador debe ser capaz de determinar las mejores opciones en la selección de compuertas, chips y dispositivos a utilizar en el prototipo del circuito que va a diseñar.

Si el circuito es realizable entonces se deben identificar las variables de entrada y el número de salidas; estas últimas se identificarán como funciones de las variables de entrada. Los pasos a seguir después de haber formulado el problema son los siguientes:

1. Se construye la tabla de la verdad, colocando ordenadamente las variables de entrada y salidas. Se deben numerar, por filas, todos los términos (maxterms y minterms) que intervienen en el circuito digital. La cantidad de filas depende del número de variables de entrada. Por ejemplo, cinco variables de entrada darán como resultado $2^5=32$ filas.
2. Se obtiene la forma canónica de cada función de salida. Cualquiera de las dos formas canónicas es válida. No obstante, la elección de esto depende del método de simplificación que se utilice. Por ejemplo, si se aplica el método algebraico hay que optar por la menor cantidad de términos de la tabla; de lo contrario, si el método es gráfico (como se verá en temas posteriores) hay que tomar los grupos con mayor número de términos.
3. Simplificar la función de conmutación (o las funciones) hasta obtener la mínima cantidad de compuertas digitales. La finalidad es ahorrar costo y espacio en la tarjeta de circuito impreso (PCB) que se utilizará para montar los componentes del diseño. Esta etapa tiene la opción de simplificar la función mediante el uso del álgebra de Boole, método gráfico o tabular.

(Estos dos últimos, serán vistos en temas posteriores). El circuito simplificado se lleva a diagramas de compuertas, o chips, y se construye el plano del circuito electrónico digital.

4. Realizar la simulación por computadora del diseño minimizado, mediante un software para tal fin. Los programas de simulación digital se consiguen en el mercado a precios razonables que dependen de la empresa que lo fabrica, y a quién va dirigido. Por ejemplo, los programas de simulación CIRCUIT MAKER y ELECTRONIC WORK WEND pueden ser adquiridos a un precio de 300\$ para usuarios comunes (Estudiantes, técnicos y otros), posee librerías básicas de componentes. Por otra parte, los programas de simulación electrónica profesionales están alrededor de los 3000\$ y 5000\$; con librerías avanzadas, ayuda en línea, depuradores, simulación mixta (analógica-digital). De mismo modo, la finalidad de esta etapa es comprobar el funcionamiento del diseño, puede resultar indispensable cuando se trata de circuitos electrónicos grades (cinco o más chips). No obstante, la simulación se puede obviar para aplicaciones de pocas compuertas o, cuando el diseñador lo considere conveniente.
5. Una vez simulado el circuito digital, y luego de haber comprobado el buen funcionamiento de la simulación del diseño. Se procede a montar los circuitos integrados, componentes y materiales en una tarjeta para prototipo (Proto Board), con la finalidad de comprobar el verdadero funcionamiento de los dispositivos y materiales seleccionados. Aquí es necesario tener a la mano equipos de medida, equipos de laboratorio y herramientas para el montaje como lo son: Multímetro, Osciloscopio, Generadores de señal, Fuentes de poder, Soldador, Pinzas, Pinzas de corte, estaño y cables. Esto permite corregir alguna falla que se pueda presentar en el montaje del circuito.

Ejercicio 2.7. Diseñar un circuito digital que posea una salida y tres entradas. El circuito debe indicar en la salida cuando dos o más entradas tienen nivel lógico cero.

Solución: (Paso 1). Primero se construye **la tabla de la verdad** con el planteamiento del problema. Como se trata de tres variables, entonces el número de filas será igual a 8 y, la salida señala con uno lógico (F=1) la condición de dos o más entradas en bajo.

n	a	b	c	F
0	0	0	0	1
1	0	0	1	1
2	0	1	0	1
3	0	1	1	0
4	1	0	0	1
5	1	0	1	0
6	1	1	0	0
7	1	1	1	0

Tabla de la verdad del ejercicio 2.7

(Paso 2). Una vez obtenida la tabla se seleccionan los minterms o maxterms para formar la función de conmutación. Debido a que la cantidad de maxterms y minterms son iguales se puede optar por cualquiera de los dos; por lo cual, se toman los minterms:

$$F(a,b,c) = \bar{a}\bar{b}\bar{c} + \bar{a}\bar{b}c + \bar{a}b\bar{c} + a\bar{b}\bar{c} \quad \text{Forma canónica algebraica minterms.}$$

(Paso 3). Se simplifica la función algebraicamente con teoremas y leyes:

$$\begin{aligned} F(a,b,c) &= \bar{a}\bar{b}\bar{c} + \bar{a}\bar{b}c + \bar{a}b\bar{c} + a\bar{b}\bar{c} \\ &= \bar{a}\bar{b}\bar{c} + \bar{a}\bar{b}c + \bar{a}b\bar{c} + \bar{a}b\bar{c} + \bar{a}\bar{b}\bar{c} + a\bar{b}\bar{c} \\ &= \bar{a}\bar{b}(\bar{c} + c) + \bar{a}c(\bar{b} + b) + \bar{b}\bar{c}(\bar{a} + a) \\ &= \bar{a}\bar{b}(1) + \bar{a}c(1) + \bar{b}\bar{c}(1) \end{aligned}$$

$$F(a,b,c) = \bar{a}\bar{b} + \bar{a}c + \bar{b}\bar{c} \quad \text{Función de conmutación simplificada.}$$

Las compuertas necesarias son: NOT, AND y OR. Los complementos se realizan con la NOT, los productos con la AND y la suma final se hace con compuertas OR. También

se puede realizar el diseño con compuertas universales NAND y NOR aplicando la doble negación a la función y luego el teorema de DeMorgan.

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

$$= \overline{\overline{a} \cdot \overline{b} \cdot \overline{a} \cdot \overline{c} \cdot \overline{b} \cdot \overline{c}}$$

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

Función de conmutación implementada con NAND.

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

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

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

Función de conmutación implementada con NOR.

El paso siguiente es realizar el esquema eléctrico o plano del circuito obtenido en la simplificación de la función. La figura 2.1 muestra el esquema electrónico del circuito digital con compuertas básicas y universales.

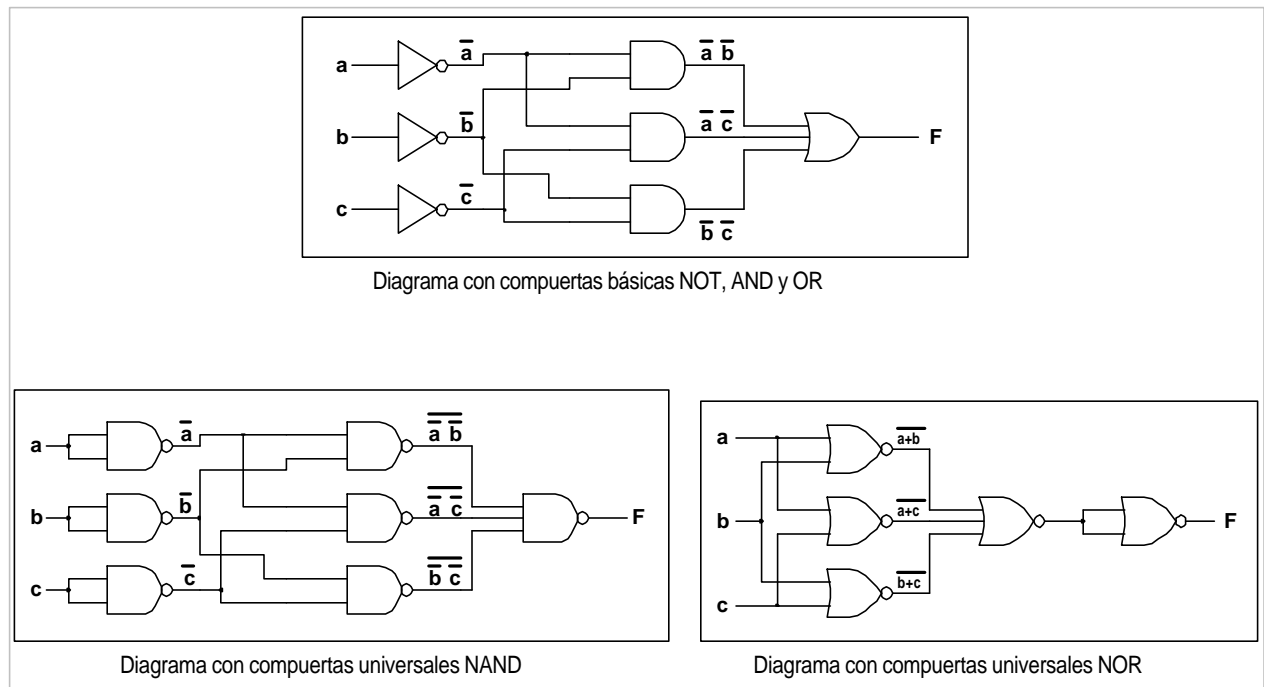


Figura 2.1. Distintos esquemas de compuertas de la función: $F(a,b,c) = \overline{\overline{a} \cdot \overline{b} \cdot \overline{a} \cdot \overline{c} \cdot \overline{b} \cdot \overline{c}}$

A continuación, se puede seguir con la simulación del plano del circuito digital. La salida de la función se debe señalar con un diodo led o un "logic display". La figura 2.2 muestra la simulación del circuito realizada con el software de simulación "Circuit Maker"; cada variable de entrada posee un "logic Switch" para conmutar los niveles lógicos del circuito ($0 = 0V$ y $1 = 5V$). El encendido de L1 indica que dos o más entradas son cero lógico, de lo contrario, el Logic Display se apaga.

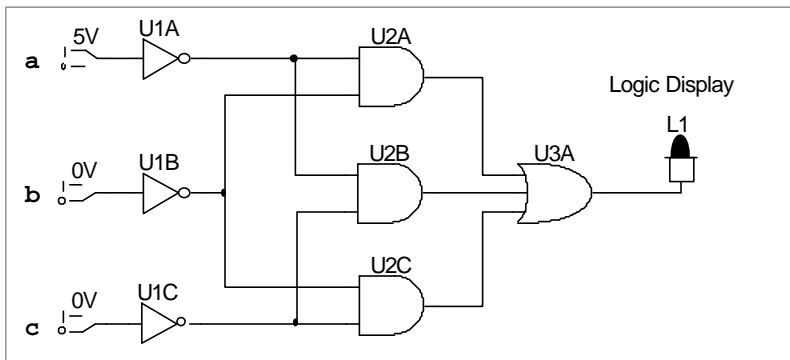


Figura 2.2. Simulación del esquema en Circuit Maker.

Por último, se debe realizar el montaje de componentes y conectar los pines de los circuitos integrados utilizados en el diseño del circuito digital. Esta etapa tiene dos alternativas: se puede realizar directamente en un circuito impreso PCB, o en Protoboard. En circuitos grandes es recomendable, el Protoboard, ya que éste permite la remoción de algún componente y por ende, la reparación de una falla eventual que se pueda presentar en el montaje. Con el PCB también se puede dar el caso; sin embargo, hay que desoldar el componente que presente fallas.

Ejercicio 2.8. Diseñe un circuito digital con compuertas que funcione de la siguiente forma: Cuando "C" es igual a cero se debe detectar y señalar, con un diodo led de color verde, todos los números divisibles por cuatro; el led rojo no debe encender. Si "C" cambia a un nivel de cinco voltios, entonces se debe detectar y señalar, con un led de color rojo, todos los números divisibles por tres y además, con el led verde, los números divisibles por cinco. El conjunto de números va desde cero hasta quince, inclusive.

El cero se puede admitir como divisible para los dos casos. Ver figura 2.3.

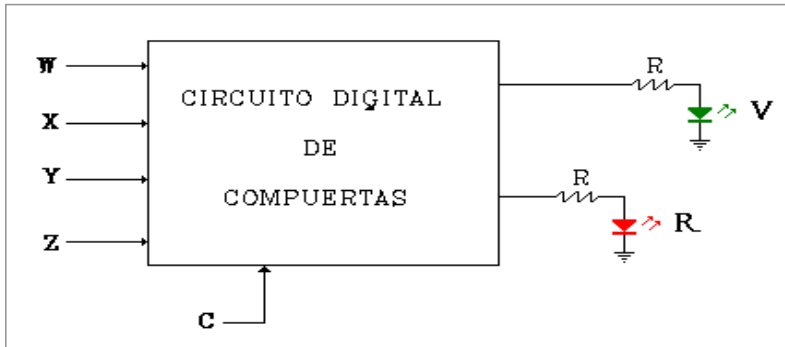


Figura 2.3. Diagrama en bloque del ejercicio 2.8.

Solución. se construye la tabla de la verdad y se extraen las formas canónicas:

n	c	w	x	y	z	Fv	Fr
0	0	0	0	0	0	1	0
1	0	0	0	0	1	0	0
2	0	0	0	1	0	0	0
3	0	0	0	1	1	0	0
4	0	0	1	0	0	1	0
5	0	0	1	0	1	0	0
6	0	0	1	1	0	0	0
7	0	0	1	1	1	0	0
8	0	1	0	0	0	1	0
9	0	1	0	0	1	0	0
10	0	1	0	1	0	0	0
11	0	1	0	1	1	0	0
12	0	1	1	0	0	1	0
13	0	1	1	0	1	0	0
14	0	1	1	1	0	0	0
15	0	1	1	1	1	0	0

Tabla de la verdad del ejercicio 2.8.

n	c	w	x	y	z	Fv	Fr
16	1	0	0	0	0	1	1
17	1	0	0	0	1	0	0
18	1	0	0	1	0	0	0
19	1	0	0	1	1	0	1
20	1	0	1	0	0	0	0
21	1	0	1	0	1	1	0
22	1	0	1	1	0	0	1
23	1	0	1	1	1	0	0
24	1	1	0	0	0	0	0
25	1	1	0	0	1	0	1
26	1	1	0	1	0	1	0
27	1	1	0	1	1	0	0
28	1	1	1	0	0	0	1
29	1	1	1	0	1	0	0
30	1	1	1	1	0	0	0
31	1	1	1	1	1	1	1

$$F_v(c, w, x, y, z) = \sum_m (0, 4, 8, 12, 16, 21, 26, 31)$$

$$F_r(c, w, x, y, z) = \sum_m (16, 19, 22, 25, 28, 31)$$

$$F_v(c, w, x, y, z) = \bar{c} \bar{w} \bar{x} \bar{y} \bar{z} + \bar{c} \bar{w} x \bar{y} \bar{z} + \bar{c} w \bar{x} \bar{y} \bar{z} + \bar{c} w x \bar{y} \bar{z} + c \bar{w} \bar{x} \bar{y} \bar{z} + c \bar{w} x \bar{y} \bar{z} + c w \bar{x} \bar{y} \bar{z} + c w x \bar{y} \bar{z}$$

$$F_v(c, w, x, y, z) = \bar{c} \bar{w} \bar{y} \bar{z} + \bar{c} w \bar{y} \bar{z} + c[\bar{w} \bar{y}(\bar{x} \bar{z} + x y) + w y(\bar{x} \bar{z} + x y)]$$

$$F_v(c, w, x, y, z) = \bar{c} \bar{y} \bar{z} + c[\bar{w} \bar{y}(\bar{x} \oplus z) + w y(\bar{x} \oplus z)]$$

$$F_v(c, w, x, y, z) = \bar{c} \bar{y} \bar{z} + c[(\bar{w} \bar{y} + w y)(\bar{x} \oplus z)]$$

$$F_v(c, w, x, y, z) = \bar{c} \bar{y} \bar{z} + c(\bar{w} \oplus y)(\bar{x} \oplus z)$$

$$F_r(c, w, x, y, z) = c \bar{w} \bar{x} \bar{y} \bar{z} + c \bar{w} \bar{x} y \bar{z} + c \bar{w} x y \bar{z} + c w \bar{x} \bar{y} \bar{z} + c w x \bar{y} \bar{z} + c w x y \bar{z}$$

$$F_r(c, w, x, y, z) = c \bar{w} \bar{x}(\bar{y} \bar{z} + y z) + c \bar{w} x y \bar{z} + c w \bar{x} \bar{y} \bar{z} + c w x(\bar{y} \bar{z} + y z)$$

$$F_r(c, w, x, y, z) = c[(\bar{w} \bar{x} + w x)(\bar{y} \bar{z} + y z)] + c \bar{w} x y \bar{z} + c w \bar{x} \bar{y} \bar{z}$$

$$F_r(c, w, x, y, z) = c(\bar{w} \oplus x)(\bar{y} \oplus z) + c \bar{w} x y \bar{z} + c w \bar{x} \bar{y} \bar{z}$$

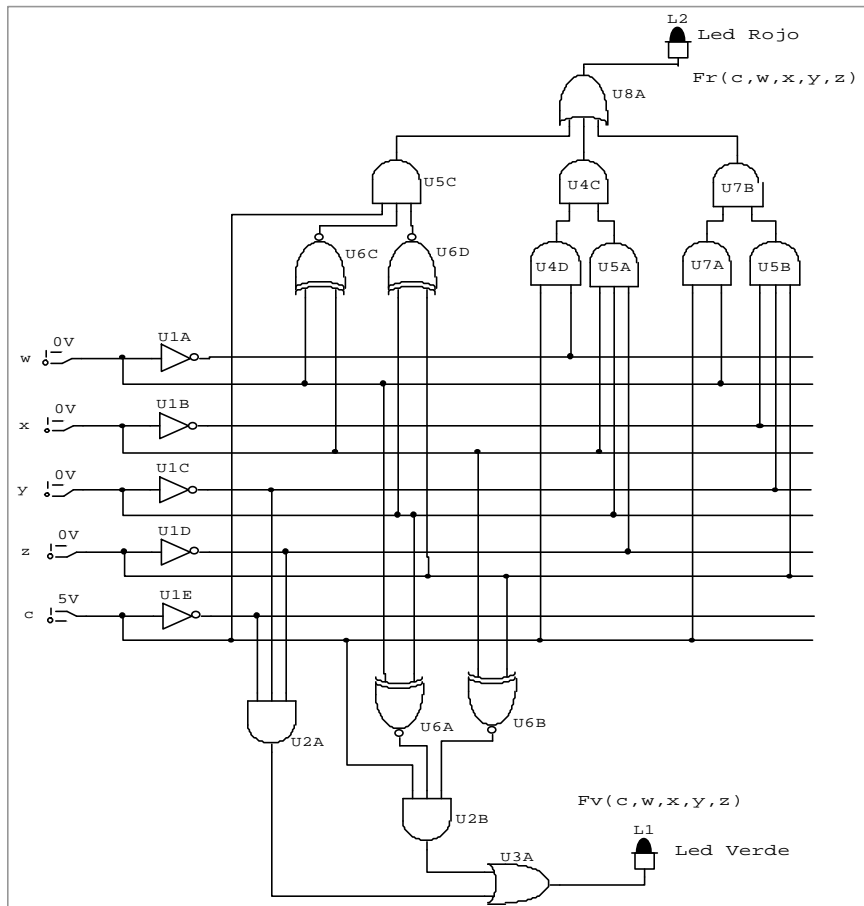


Figura 2.4. Esquema y simulación de compuertas del ejercicio 2.8.

2.5 Aplicaciones de los circuitos digitales combinacionales de compuertas.

En el tema anterior (2.4) se describió el procedimiento que se debe seguir para realizar implantar o sintetizar circuitos digitales de compuertas. Los ejercicios 2.8 y 2.9 son ejemplos de aplicaciones de circuitos digitales combinacionales realizados con compuertas digitales que se obtienen a partir de la construcción de la tabla de la verdad del problema; luego, las formas canónicas, la simplificación de las funciones, la simulación y por último, la implementación del circuito. Cualquier diseño digital debería ser realizado por este procedimiento. No obstante, en aplicaciones sencillas, por ejemplo en un diseño “**ad hoc**”, es posible que se omitan algunos de estos pasos.

El rango de aplicaciones de los circuitos digitales combinacionales de compuertas es muy amplio. Se pueden implementar circuitos de comparación; circuitos aritméticos (sumadores, restadores, multiplicadores, divisores); convertidores de códigos; selectores de datos (multiplexores); concentradores de datos (demultiplexores), aplicaciones específicas de algunos problemas cotidianos e implementaciones mixtas de ellos.

Ejercicio 2.9. Diseñar e implementar un circuito digital que mediante una señal de control “C” pueda seleccionar el tipo de conversión de código: con C =1, GRAY a BINARIO y con C =0, BINARIO a GRAY. El código es de tres bits.

Solución: Se construye la tabla de la verdad realizando la equivalencia de los códigos.

n	C	X	Y	Z	F ₁	F ₂	F ₃
0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	1
2	0	0	1	0	0	1	1
3	0	0	1	1	0	1	0
4	0	1	0	0	1	1	0
5	0	1	0	1	1	1	1
6	0	1	1	0	1	0	1
7	0	1	1	1	1	0	0

n	C	X	Y	Z	F ₁	F ₂	F ₃
8	1	0	0	0	0	0	0
9	1	0	0	1	0	0	1
10	1	0	1	0	0	1	1
11	1	0	1	1	0	1	0
12	1	1	0	0	1	1	1
13	1	1	0	1	1	1	0
14	1	1	1	0	1	0	0
15	1	1	1	1	1	0	1

Tabla de la verdad del ejercicio 2.9.

La figura 2.5 muestra el diagrama en bloque del circuito digital combinacional, convertidor de código (Binario – Gray; Gray – Binario) de tres bits.

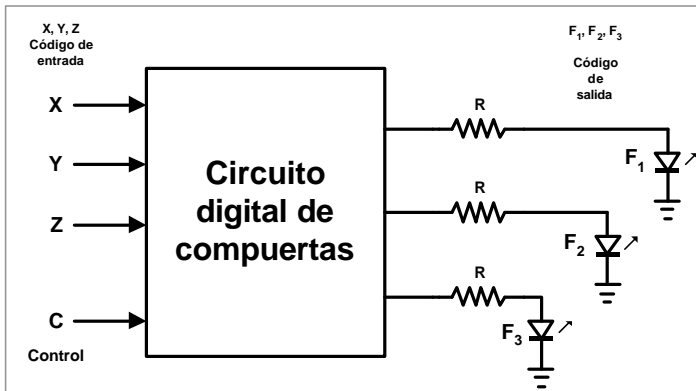


Figura 2.5. Diagrama en bloque del ejercicio 2.9.

Las funciones resultantes en forma de lista de minterms son:

$$F_1(C, X, Y, Z) = \sum_m (4, 5, 6, 7, 12, 13, 14, 15)$$

$$F_2(C, X, Y, Z) = \sum_m (2, 3, 4, 5, 10, 11, 12, 13, 14)$$

$$F_3(C, X, Y, Z) = \sum_m (1, 2, 5, 6, 9, 10, 12, 15)$$

Simplificación algebraica de cada una de las funciones de conmutación:

$$F_1(C, X, Y, Z) = \bar{C}X\bar{Y}\bar{Z} + \bar{C}X\bar{Y}Z + \bar{C}XY\bar{Z} + \bar{C}XYZ + CX\bar{Y}\bar{Z} + CX\bar{Y}Z \\ + CXY\bar{Z} + CXYZ$$

$$F_1(C, X, Y, Z) = \bar{C}X\bar{Y}(\bar{Z}+Z) + \bar{C}XY(\bar{Z}+Z) + CX\bar{Y}(\bar{Z}+Z) + CXY(\bar{Z}+Z)$$

$$F_1(C, X, Y, Z) = \bar{C}X\bar{Y} + \bar{C}XY + CX\bar{Y} + CXY$$

$$F_1(C, X, Y, Z) = \bar{C}X(\bar{Y}+Y) + CX(\bar{Y}+Y)$$

$$F_1(C, X, Y, Z) = \bar{C}X + CX$$

$$F_1(C, X, Y, Z) = X(\bar{C}+C)$$

$$F_1(C, X, Y, Z) = X$$

$$F_2(C, X, Y, Z) = \overline{C}\overline{X}Y\overline{Z} + \overline{C}\overline{X}YZ + \overline{C}X\overline{Y}\overline{Z} + \overline{C}X\overline{Y}Z + C\overline{X}Y\overline{Z} + C\overline{X}YZ \\ + CX\overline{Y}\overline{Z} + CX\overline{Y}Z$$

$$F_2(C, X, Y, Z) = \overline{C}\overline{X}Y(\overline{Z}+Z) + \overline{C}X\overline{Y}(\overline{Z}+Z) + C\overline{X}Y(\overline{Z}+Z) + CX\overline{Y}(\overline{Z}+Z)$$

$$F_2(C, X, Y, Z) = \overline{C}\overline{X}Y + \overline{C}X\overline{Y} + C\overline{X}Y + CX\overline{Y}$$

$$F_2(C, X, Y, Z) = \overline{C}(\overline{X}Y + X\overline{Y}) + C(\overline{X}Y + X\overline{Y})$$

$$F_2(C, X, Y, Z) = (\overline{C} + C)(\overline{X}Y + X\overline{Y})$$

$$F_2(C, X, Y, Z) = (\overline{X}Y + X\overline{Y})$$

$$F_2(C, X, Y, Z) = X \oplus Y$$

$$F_3(C, X, Y, Z) = \overline{C}\overline{X}\overline{Y}Z + \overline{C}\overline{X}YZ + \overline{C}X\overline{Y}Z + \overline{C}XY\overline{Z} + C\overline{X}\overline{Y}Z + C\overline{X}YZ \\ + CX\overline{Y}\overline{Z} + CXYZ$$

$$F_3(C, X, Y, Z) = \overline{C}\overline{X}(\overline{Y}Z + Y\overline{Z}) + \overline{C}X(\overline{Y}Z + Y\overline{Z}) + C\overline{X}(\overline{Y}Z + Y\overline{Z}) + CX(\overline{Y}\overline{Z} + YZ)$$

$$F_3(C, X, Y, Z) = (\overline{C}\overline{X} + \overline{C}X + C\overline{X})(\overline{Y}Z + Y\overline{Z}) + CX(\overline{Y}\overline{Z} + YZ)$$

$$F_3(C, X, Y, Z) = [\overline{C}(\overline{X}+X) + C\overline{X}](\overline{Y}Z + Y\overline{Z}) + CX(\overline{Y}\overline{Z} + YZ)$$

$$F_3(C, X, Y, Z) = [\overline{C} + C\overline{X}](\overline{Y}Z + Y\overline{Z}) + CX(\overline{Y}\overline{Z} + YZ)$$

$$F_3(C, X, Y, Z) = (\overline{C} + \overline{X})(\overline{Y}Z + Y\overline{Z}) + CX(\overline{Y}\overline{Z} + YZ)$$

$$F_3(C, X, Y, Z) = (\overline{C} + \overline{X})(Y \oplus Z) + CX(\overline{Y \oplus Z})$$

$$F_3(C, X, Y, Z) = \overline{C}\overline{X}(Y \oplus Z) + CX(\overline{Y \oplus Z})$$

$$F_3(C, X, Y, Z) = (C\overline{X}) \oplus (Y \oplus Z)$$

Luego se realiza la simulación del circuito y se implementa en Protoboard. El plano de compuertas del circuito digital, convertidor de código, se muestra en la figura 2.6:

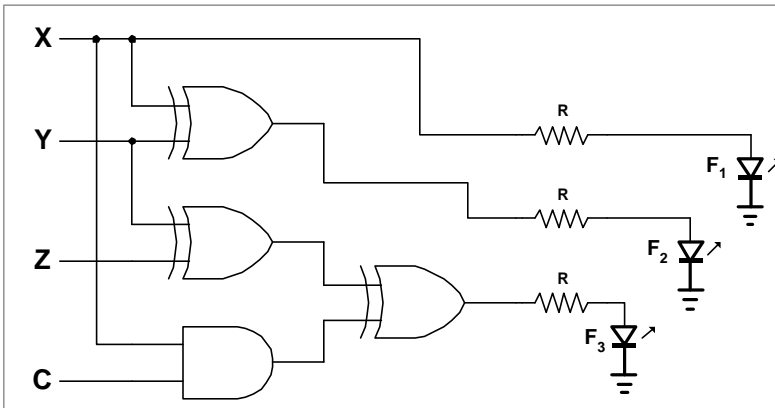


Figura 2.6. Esquema de compuertas del ejercicio 2.9.

A continuación se describen algunos ejercicios propuestos para diseñar e implementar las aplicaciones de los circuitos digitales combinacionales de compuertas. Estos pueden ser resueltos utilizando la metodología descrita en el tema 2.4.

Ejercicio 2.10. Luis, Ana, Pedro y sus padres van para el cine bajo ciertas condiciones que son limitadas por la Madre y el Padre. Ellos van al cine si la Madre y el Padre le dan permiso; de lo contrario, si los dos no le dan permiso, entonces no pueden ir. Ahora, si los dos están en desacuerdo; la salida al cine debe ser resuelta por mayoría absoluta entre todos los integrantes de la familia. Diseñar un circuito digital que señalice con un diodo led el momento cuando puedan ir al cine.

Ejercicio 2.11. Diseñar un circuito digital, con compuertas, que permita restar dos datos de dos bits cada uno. La salida del circuito debe indicar el resultado de la operación con su respectivo signo.

Ejercicio 2.12. Diseñar un circuito digital, con compuertas, que sume dos datos de un bit cada uno. El circuito debe poseer una entrada adicional que permita detectar cuando hay acarreo de entrada en la operación. La salida del circuito debe indicar y señalar en el resultado de la suma y el acarreo de salida.

Ejercicio 2.13. Diseñe un señalizador de juego con compuertas digitales comerciales. El circuito debe indicar con un led de color verde si el jugador gana, rojo en caso de que pierda, amarillo si repite la jugada. El juego consiste en un acumulado que se incrementa o disminuye, según sea el valor dado por un contador aleatorio que debe pulsarse para realizar la jugada. En la tabla se indican los porcentajes ganados o perdidos. Sin embargo, no deben ser representados en la salida del circuito digital.

Contador Aleatorio	Resultado
○ ● ○ ○	Pierde 75%
○ ● ○ ●	Pierde 70%
○ ○ ● ●	Pierde 65%
○ ● ● ○	Pierde 60%
● ● ○ ○	Gana 30%
● ● ● ○	Gana 50%
● ○ ● ○	Gana 60%
● ○ ● ●	Gana 70%
● ● ○ ●	Gana 90%
● ● ● ●	Repite Jugada
○ ○ ○ ●	Repite Jugada
○ ○ ● ○	Repite Jugada
● ○ ○ ○	Repite Jugada

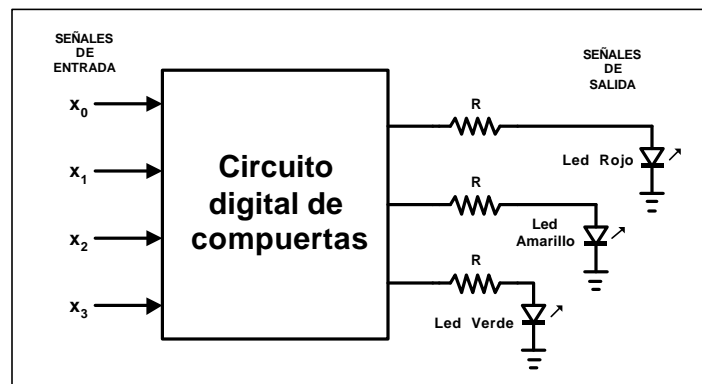


Tabla de señales y diagrama en bloque del ejercicio 2.13.

Las combinaciones que no están contempladas en la tabla, las salidas de las funciones, deben ser colocadas a cero.

Ejercicio 2.14. Se desea diseñar e instalar un sistema que pueda detectar y señalar el momento cuando cinco líneas telefónicas, sean utilizadas por el personal de una empresa. Las líneas L_1 y L_2 son utilizadas por el presidente de la empresa, y por ende, no deben generar señal de alarma; sin embargo, debe encender un indicador cuando las dos están ocupadas simultáneamente. L_3 , L_4 y L_5 generan alarma cuando dos o más están ocupadas al mismo tiempo; por otra parte, el indicador de ocupado debe encender cuando alguna de las tres líneas está ocupada. Las líneas telefónicas tienen un dispositivo acoplado que genera 0 Volt, en ocupado y 5 Volt cuando no está en uso. Diseñe el circuito digital de compuertas que pueda indicar la señal de alarma y la señal de línea telefónica ocupada.

PRÁCTICA DE LABORATORIO #1

TÍTULO: Diseño y montaje, en PROTOBOARD, de circuitos digitales combinacionales utilizando compuertas básicas y universales.

INTRODUCCIÓN: Esta práctica contempla el diseño y montaje de circuitos digitales combinacionales, en PROTOBOARD, utilizando para éste propósito compuertas básicas y universales. El primer montaje posee una sola salida y se realiza con compuertas universales, el segundo montaje es un circuito combinacional de múltiples salidas y se utilizan compuertas básicas y/o universales.

OBJETIVO: Se persigue que el estudiante, una vez finalizada la práctica, pueda realizar montajes de tipo lógico combinacional diseñados para resolver problemas típicos presentados en la vida real que puedan ser resueltos utilizando compuertas digitales.

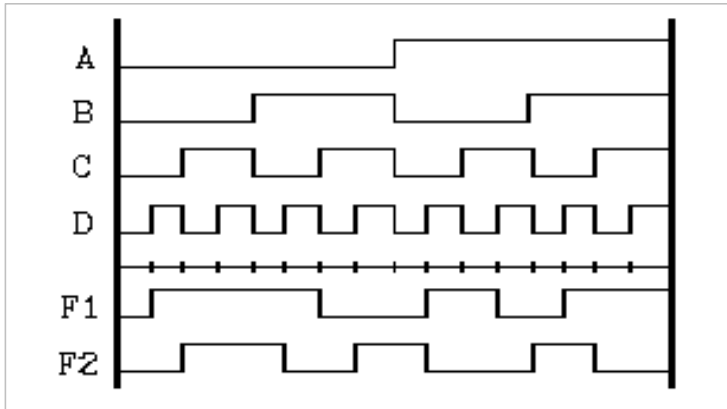
PRE-LABORATORIO: Investigar los siguientes tópicos.

- Teoremas y postulados del álgebra de Boole. Fundamentos de Leds.
- Descripción y manejo del **PROTOBOARD**.
- Manejo de paquetes para simulación digital (por ejemplo **CIRCUIT MAKER, EWB**)
- Compuertas AND, OR, NOT, NOR, NAND, XOR.
- Investigar el funcionamiento de los circuitos integrados de compuertas: 7400, 7402, 7404, 7408, 7410, 7420, 7430, 7432, 7486.
- Simplificar e implementar con compuertas la siguiente función de conmutación:

$$F(m, n, o, p) = \overline{\overline{\overline{\overline{m.n.o.p}} + \overline{\overline{\overline{m.n.o.p}} + \overline{\overline{\overline{m.n.o.p}} + \overline{\overline{\overline{m.n.o.p}}}}}}}$$

- Implementar y simular con compuertas básicas.
- Implementar y simular con compuertas NAND de dos entradas.
- Implementar y simular con compuertas NOR de dos entradas.
- Simplificar y simular las siguientes funciones combinacionales:
F=S_m(0, 4, 8, 12); G=P_m(1, 2, 4, 5, 7, 10, 13, 14, 15); H=S_m(3, 4, 6, 8, 10, 15, 21, 24, 26, 27, 31)

- Simplificar y simular las siguientes funciones combinacionales que deben ser generadas del siguiente diagrama de tiempo:

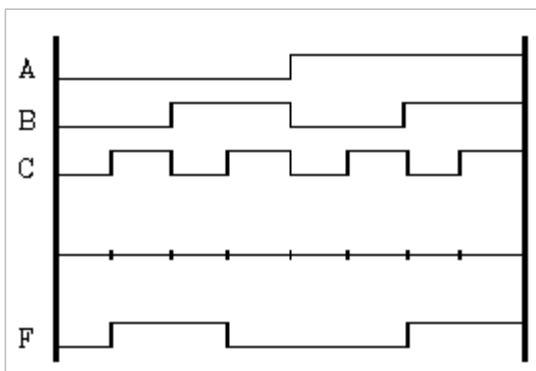


MATERIALES Y EQUIPOS NECESARIOS:

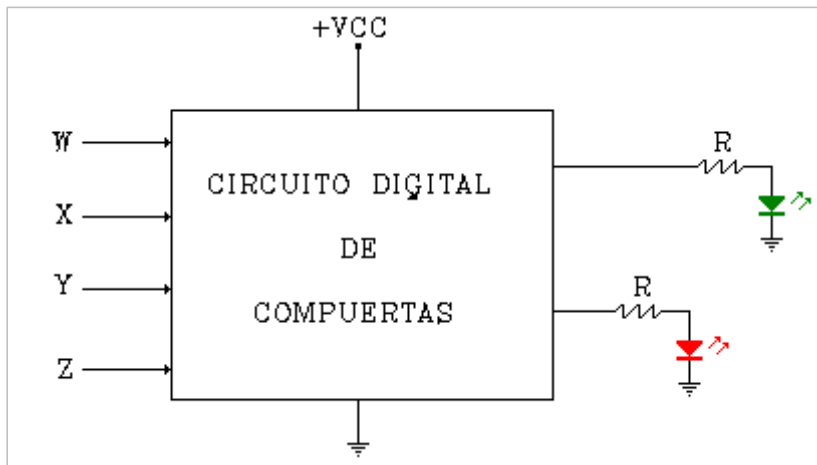
- Circuitos integrados según las necesidades de cada montaje.
- PROTOBOARD de tres regletas.
- Diodos leds de acuerdo con las necesidades de cada montaje.
- Fuente de 5 Vdc / 1 amp.
- Cable telefónico de un solo pelo, Pinzas, Piquetas.
- Multímetro digital.
- Software de simulación CIRCUIT MAKER, EWB.

DESARROLLO:

1. Diseñar y montar en PROTOBOARD, un circuito digital que pueda reproducir en forma fiel el siguiente diagrama de tiempo; hacerlo con compuertas universales.



2. Un circuito, que recibe cuatro señales digitales, debe detectar e indicar con un diodo led rojo, la coincidencia de dos señales con nivel alto y también debe detectar e indicar, con un led verde, el momento en que una sola señal suba a un nivel alto. Diseñar el circuito e implementarlo en PROTOBOARD, con compuertas básicas y/o universales, de la forma más simple posible.



MONTAJES ALTERNATIVOS:

1. En un sistema de seguridad hay tres niveles de jerarquía, el bloqueo es controlado por pares. Cada uno de los niveles genera una señal de 5 voltios cuando desea bloquearse, para que esto suceda, basta que el nivel 1 y el nivel 2 estén en alto. Si esto no sucede, entonces el bloqueo podrán hacerlo los niveles 1 y 3, o los niveles 2 y 3 respectivamente. El sistema no permite el bloqueo cuando los tres niveles se encuentran en alto. Diseñar e implementar con compuertas NOR, además, se debe señalar con led rojo el bloqueo y led verde el desbloqueo.
2. El código binario posee la desventaja de cambiar más de un bit; al pasar de un estado inferior a otro superior o viceversa. Sin embargo, el código GRAY elimina esta desventaja cambiando solamente un bit. Esto se debe, a que es más probable cometer errores; por ejemplo, de transmisión, cuando cambian muchos bits simultáneamente que cuando cambia solo uno. Se pide, Implementar con compuertas básicas y universales un convertidor de: código binario a código GRAY; y de GRAY a

binario, de tres bits, más una entrada de control. Señalizar entradas y salidas con diodos leds.

POST-LABORATORIO.

1. ¿ De qué forma puede, el teorema de Morgan, ayudar a simplificar circuitos digitales?.
2. Nombre algunas ventajas de usar un paquete de simulación digital.
3. ¿ Qué tipo de compuertas utilizó?. Demuéstrelo mediante simplificación algebraica.
4. Demuestre las dos formas canónicas del álgebra de conmutación mediante el teorema de SHANNON.
5. Normas para el uso del PROTOBOARD.
6. Explique el funcionamiento de cada uno de los montajes realizados.
7. Diseñe un restador completo, de dos bits, e impleméntelo con compuertas.

BIBLIOGRAFÍA.

- CUESTA, Luís M. PADILLA G, Antonio. REMIRO D, Fernando. (1993). Electrónica digital. Madrid: McGraw Hill. S/f. p.445.
- GAJSKI, Daniel D. (1997). Principios de diseño digital. Madrid: Prentice Hall Iberia. S/f. p.488. "Principles of digital design". Traducido por: Alberto Prieto Espinosa.
- LLORIS, Antonio. PRIETO, Alberto. (1996). Diseño lógico. Madrid: McGraw Hill. S/f. p.403.
- MANDADO, Enrique. (1987). Sistemas electrónicos digitales. Barcelona (España): Marcombo Boixareu Editores. Sexta edición. p.705.
- MANO, Morris. KIME, Charles. (1998). Fundamentos de diseño lógico y computadoras. México: Prentice Hall. Primera edición en español. P.604. "Logic and computer design fundamentals". Traducido por: Teresa Sanz Falcón.
- NEAMEN A, Donald. (1999). Análisis y diseño de circuitos electrónicos. Tomo II. México: McGraw Hill. S/f. p.1176. "Electronic circuit analysis and design". Traducido por: Felipe Castro Pérez.
- NELSON, V. NAGLE, H. CARROLL, B. IRWIN, J. (1996). Análisis y diseño de circuitos lógicos digitales. México: Prentice Hall. Primera edición. p.842. "Digital logic circuit analysis and design". Traducido por: Oscar A. Palmas V.
- TOCCI, Ronald. (1995). Sistemas digitales principios y aplicaciones. México: Prentice Hall. Quinta edición. p.823. "Digital systems principles and applications". Traducido por: Edmundo G. Urbina M.
- WARKELY, John F. (1997). Diseño digital principios y prácticas. México: Prentice Hall. S/f. p.743. "Digital design principles and practices". Traducido por: Gutiérrez R. Raymundo H.

MANUALES.

- NATIONAL SEMICONDUCTOR. (1981). Manual TTL y CMOS.
- MOTOROLA Inc. (1992). Fast and LS TTL. (DL121/D REV. 5).

CAPÍTULO 3.

3. METODOS DE SIMPLIFICACIÓN DE FUNCIONES LÓGICAS.

Hasta el momento se han realizado simplificaciones de circuitos digitales combinacionales en forma algebraica utilizando los teoremas y leyes del álgebra de Boole. Este método es eficiente si la cantidad de variables de la función de conmutación es menor o igual a cuatro variables. No obstante, cuando se superan las cuatro variables, la simplificación algebraica es engorrosa, y por ende, difícil de determinar una solución óptima para el diseño del circuito digital.

Las realizaciones de circuitos con compuertas deben mantener tres condiciones fundamentales: 1) Utilizar el mínimo de compuertas con la finalidad de reducir costos; 2) El circuito combinacional minimizado debe funcionar igual que el circuito sin reducir. Osea, deben estar activos los mismos minterms y maxterms de la función. 3) Tomar en cuenta, para la simplificación, que cada circuito integrado comercial "C.I." posee internamente varias compuertas; por lo cual, hay que tratar de utilizar todas las compuertas que tiene el chip.

Existen dos métodos que ayudan a simplificar las funciones de conmutación; el primero de ellos es utilizando los **MAPAS DE KARNAUGH** (Mapas K). Este método es gráfico y las simplificaciones se pueden realizar visualmente agrupando en cuadrículas los minterms o maxterms, con la finalidad de minimizar la función lógica. El otro método es tabular y consiste en ir reduciendo la función por medio de tablas que se relacionan con el índice del producto de las variables de la función; el método se denomina **TABLAS DE QUINE-McCLUSKEY** (Tablas Q-M).

3.1 Minimización de funciones mediante MAPAS DE KARNAUGH.

Este es un método gráfico de simplificación para funciones de conmutación, se fundamenta en la teoría de conjuntos; donde, los conjuntos, representan los espacios de cada minterms o maxterms de la función lógica. Las letras con que se denotan los

conjuntos son equivalentes a las variables de la función; por ejemplo, para dos variables se presentan dos conjuntos, para tres variables tres conjuntos y así sucesivamente. A continuación se muestra, en la figura 3.1, la descripción de los Mapas de Karnaugh “Mapas K” para dos variables.

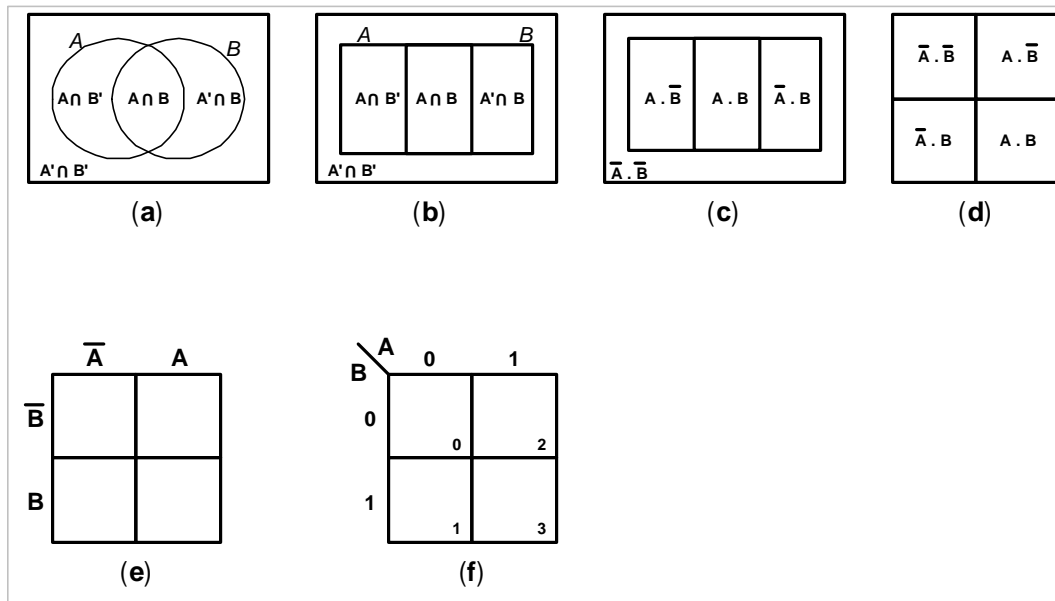


Figura 3.1. Construcción del Mapa de Karnaugh de dos variables partiendo de los conjuntos A y B.

Las secciones a, b y c de la figura 3.1 indican la forma como se creó el mapa que lleva su nombre. Los cuatro espacios que se forman con los dos conjuntos A y B tienen áreas iguales y poseen fronteras comunes; de esta forma, la figura 3.1(a) indica que el complemento de la unión de los conjuntos A y B, que abarca toda el área externa del universo U: $(A \cup B)' = A' \cap B'$ es adyacente a los espacios $A' \cap B$ y $A \cap B'$; por otra parte, $A \cap B$ también, es adyacente a los espacios $A' \cap B$ y $A \cap B'$.

Los espacios sin frontera común (no adyacentes) son $A \cap B$ con $A' \cap B'$, del mismo modo, lo son $A' \cap B$ con $A \cap B'$. Los Mapas de Karnaugh se construyen de tal forma que los espacios adyacentes queden de manera vertical y horizontal; y los no adyacentes se colocan diagonalmente, como se muestra en la figura 3.1(d). Aquí, los operadores intersección y complemento de la teoría de conjuntos se sustituye por sus equivalentes en el álgebra de Boole.

Los espacios de los Mapas son llamados celdas o cuadrículas y, cada una de ellas, forma un producto minterms, o una suma maxterms. Para n variables en la función de conmutación habrán 2^n celdas en el Mapa de Karnaugh; cada una formando un producto minterms, o una suma maxterms de la función lógica. La figura 3.1(e) y (f) muestran la representación de los MAPAS DE KARNAUGH, para dos variables, utilizada en la mayoría de los textos de circuitos digitales.

Las celdas del mapa se pueden numerar en forma decimal siguiendo una relación binaria directa de los literales con respecto a los minterms de la función de conmutación. De esta manera, se procede a colocar, en cada celda, su correspondiente número decimal.

Mapa de tres variables: La intersección de tres conjuntos dan como resultado el mapa de Karnaugh de tres variables; el cual posee $2^3=8$ celdas. La figura 3.2 muestra el diagrama del mapa K. La variable **A** es más significativa y **C** es la menos significativa.

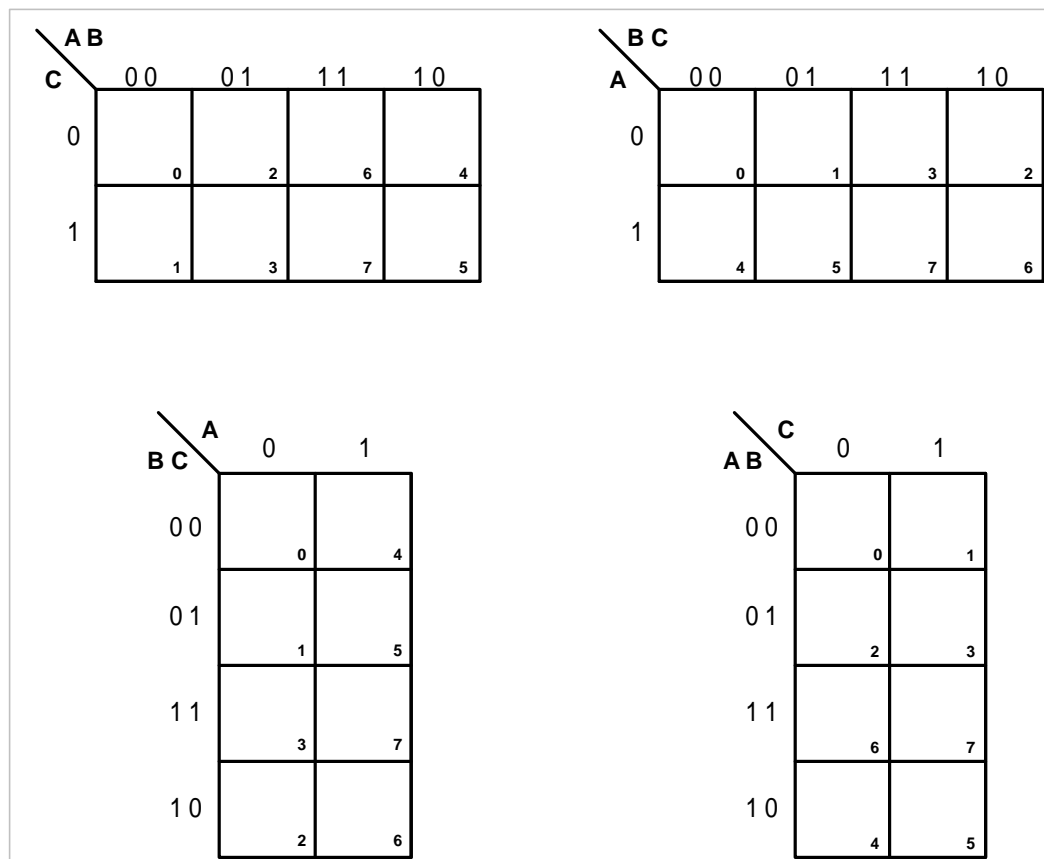


Figura 3.2. Mapas de Karnaugh de tres variables.

Mapa de cuatro variables: La intersección de cuatro conjuntos dan como resultado el mapa de Karnaugh de cuatro variables; el cual posee $2^4=16$ celdas. La figura 3.3 muestra el diagrama del mapa K de 4 variables A, B, C y D; donde la variable **A** es más significativa y **D** es la menos significativa.

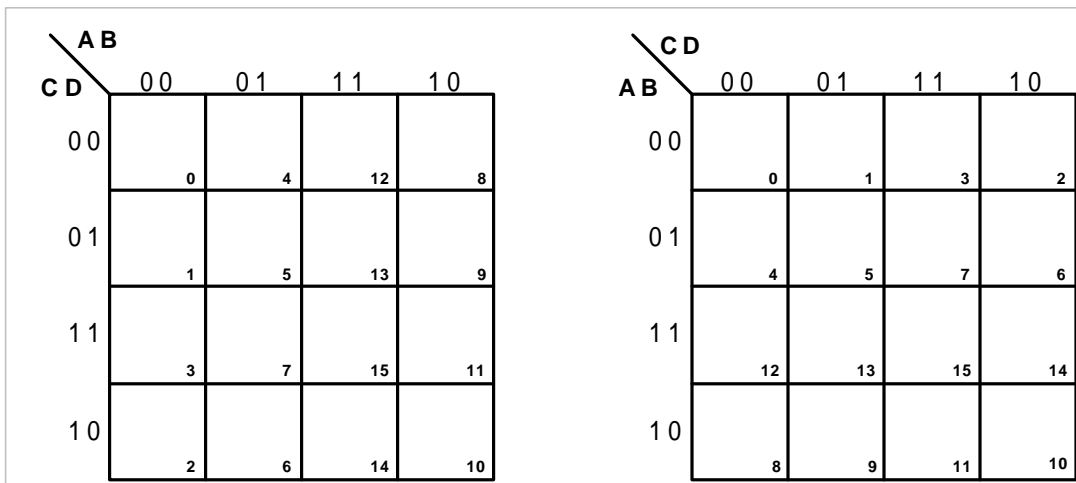


Figura 3.3. Mapas de Karnaugh de cuatro variables.

Mapa de cinco variables: La intersección de cinco conjuntos dan como resultado el mapa de Karnaugh de cinco variables; el cual posee $2^5=32$ celdas. La figura 3.4 muestra el diagrama del mapa K de 5 variables A, B, C, D y E; donde la variable **A** es más significativa y **E** es la menos significativa. Este tipo de mapa posee un eje central que lo divide en dos de cuatro variables; más adelante se verá que las simplificaciones de términos sumas o productos de los mapas de 5 y 6 variables se pueden realizar con respecto a este eje de simetría.

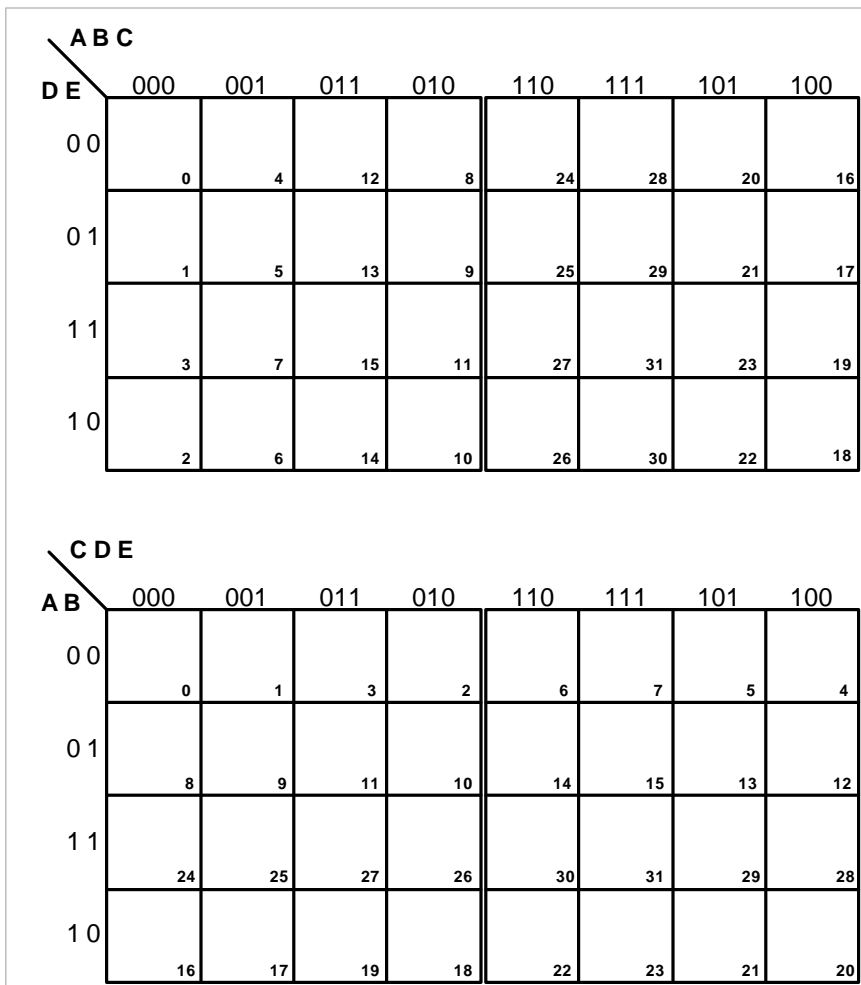


Figura 3.4. Mapas de Karnaugh de cinco variables.

Mapa K de seis variables: La intersección de seis conjuntos dan como resultado el mapa de Karnaugh de seis variables; el cual posee $2^6=64$ celdas. La figura 3.5 muestra el diagrama del mapa K de 6 variables A, B, C, D, E y F; donde la variable **A** es más significativa y **F** es la menos significativa.

DEF \ ABC									
		000	001	011	010	110	111	101	100
000	0	8	24	16	48	56	40	32	
001	1	9	25	17	49	57	41	33	
011	3	11	27	19	51	59	43	35	
010	2	10	26	18	50	58	42	34	
110	6	14	30	22	54	62	46	38	
111	7	15	31	23	55	63	47	39	
101	5	13	29	21	53	61	45	37	
100	4	12	28	20	52	60	44	36	

Figura 3.5. Mapas de Karnaugh de seis variables.

La simplificación de términos, en la función de conmutación, se realiza agrupando los minterms en las celdas del mapa. Cada minterm activado equivale a colocar un "1" en la respectiva celda. De la misma forma, se procede con los maxterms que están activados en la función de conmutación; para cada celda activada, se coloca "0" dentro de la misma. La minimización de la función se realiza tomando alguna de las dos opciones: agrupando minterms (agrupando unos) ó agrupando maxterms (agrupando ceros). Ver figura 3.6. En cualquiera de las dos opciones, la simplificación, deberá ser equivalente.

La figura 3.6(a) y 3.6(b) muestran la forma como se deben agrupar los términos en una función de conmutación de tres y cuatro variables respectivamente. Los grupos que se forman horizontal y verticalmente deben ser múltiplos de la base binaria; por lo tanto, la cantidad de unos ó ceros agrupados pueden ser: 1; 2; 4; 8; 16; 32; 64; etc.

La distribución horizontal y vertical de los grupos, en las celdas del mapa de Karnaugh obedece a que, en los espacios adyacentes, es donde se producen los cambios de estado de las variables y es por esto, que los valores binarios asignados a las celdas contiguas solo presentan cambios de un solo bit. Del mismo modo, los términos agrupados tendrán solo una variable complementada y no complementada; por lo tanto, esta variable se simplifica en la función de conmutación. La figura 3.6(a) muestra la simplificación que se produce en una agrupación de dos maxterms de celdas contiguas y, la figura 3.6(b), describe los grupos de 4 minterms que se forman con las adyacencias de los bordes del mapa; aquí se puede ver el grupo de cuatro minterms formado por las esquinas adyacentes del mapa.

La cantidad de variables que son simplificadas depende del grupo que se forme; por ejemplo, en la figura 3.6(b) son simplificadas dos variables; (A, C) de las cuatro esquinas y (B, C) en el grupo de los bordes del mapa. De la misma forma, en la figura 3.6(a) se simplifica la variable (A) en el grupo formado por los dos maxterms 1 y 5, por otra parte, en la celda del maxterms 6 no se realiza simplificación de variables. En el mapa de Karnaugh las variables de la función que son simplificadas dependen de la cantidad de términos que son agrupados; a medida que aumenta la cantidad de términos, la minimización de la función es más optima.

La reducción de variables está determinada por la expresión: $Dg = 2^r$ donde Dg el número de términos agrupados y r la cantidad de variables que son simplificadas. Por ejemplo, si el grupo formado tiene 8 términos entonces se simplifican 3 variables: $8 = 2^3$.

Los mapas de Karnaugh de cinco variables presentan adyacencias con respecto a un eje de simetría que divide al mapa de cinco variables en dos de cuatro. De forma similar, el mapa K de seis variables presenta adyacencia con respecto a dos ejes de simetría que dividen el mapa en cuatro partes. La distribución de bits en las variables del mapa (en los bordes de las filas y columnas contiguas), se realiza de forma que los cambios se produzcan en un solo bit.

La figura 3.7 muestra las adyacencias con respecto al eje de simetría del mapa de Karnaugh de cinco variables. Las puntas de flechas indican las zonas de adyacencia en el mapa, por lo cual, los minterms o maxterms que sean simétricos pueden formar

grupos de dos, cuatro, ocho, etc. La figura 3.8 también muestra las zonas adyacentes en el mapa K de seis variables. Las filas o columnas apuntadas con las flechas indican cambio de un solo bit; por lo tanto, se puede realizar simplificación de variables en esa zona. Observe que cambia un solo bit en las adyacencias que son con respecto al eje de simetría; tanto en el mapa de cinco, como en el de seis variables.

En todos los casos vistos anteriormente, los mapas de Karnaugh presentan adyacencia donde hayan cambios de un solo bit, y los minterms o maxterms que estén involucrados en las celdas del mapa, de forma horizontal y vertical, traerán como consecuencia una simplificación de una o más variables en la función de conmutación.

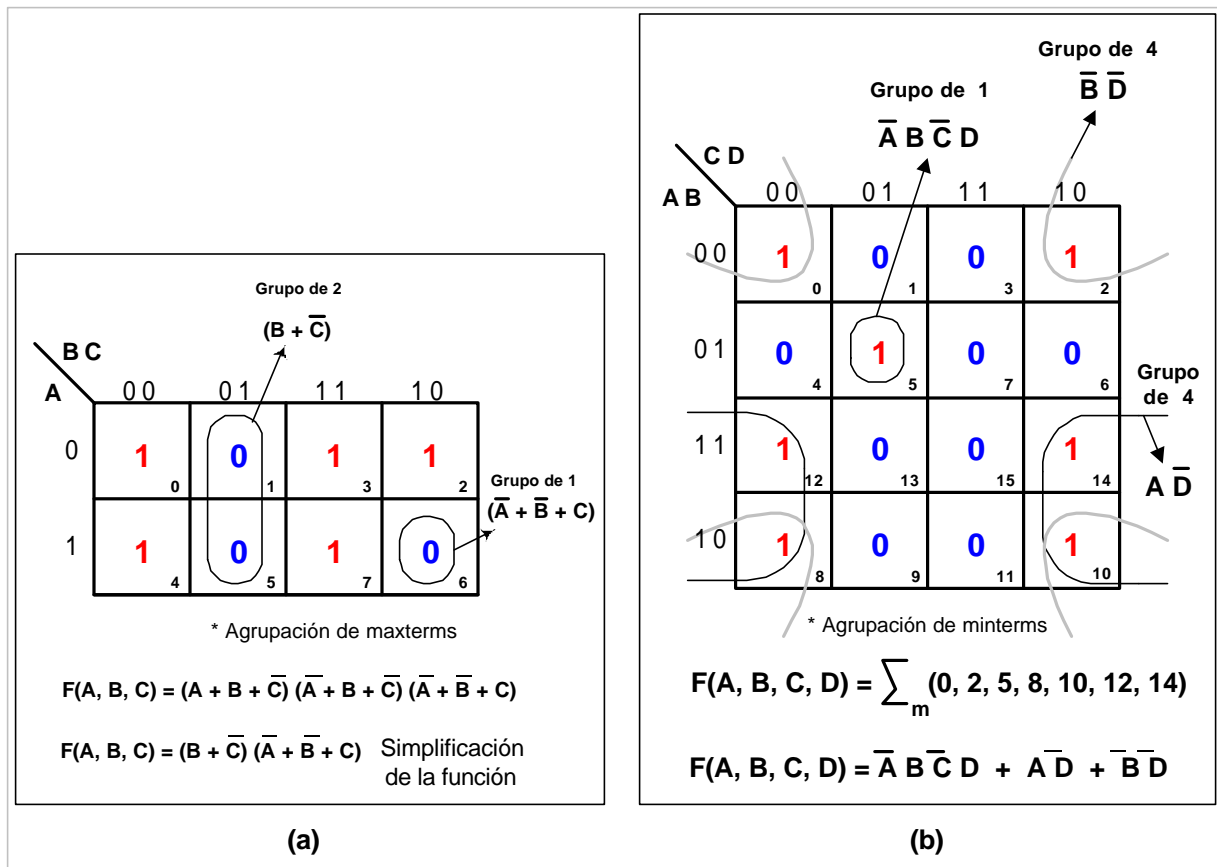


Figura 3.6. (a) Agrupamiento de uno y dos maxterms (celdas). (b) Agrupamiento de uno y cuatro Minterms (celdas).

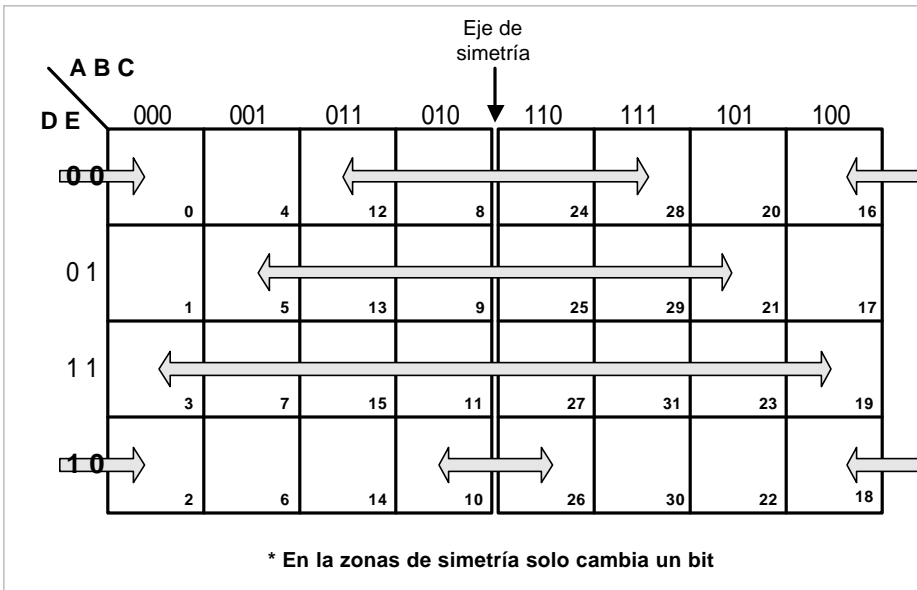


Figura 3.7. Celdas adyacentes con respecto al eje de simetría en un mapa de cinco variables.

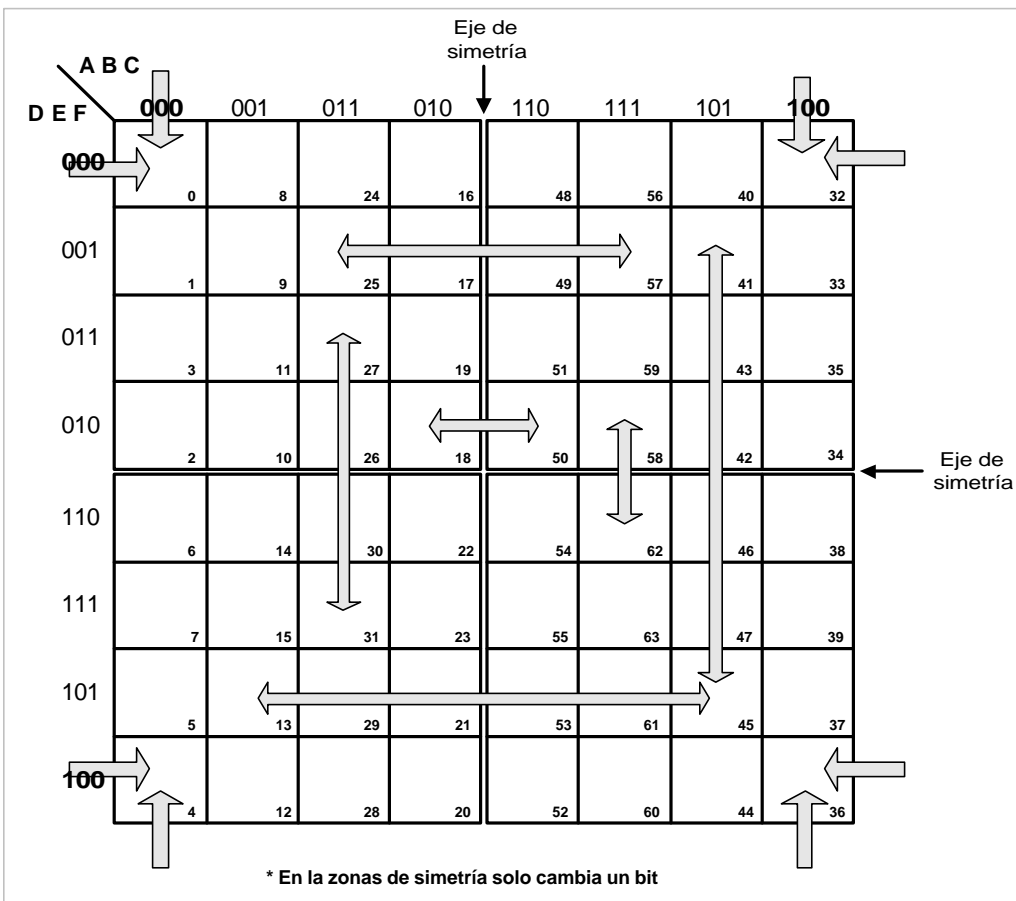


Figura 3.8. Celdas adyacentes con respecto al eje de simetría en un mapa de seis variables.

3.1.1 Simplificación de funciones aplicando mapas de Karnaugh.

El método gráfico del mapa K posee varias ventajas con respecto al método algebraico de minimización de funciones. Permite visualizar el grupo de celdas de mayor tamaño y obtener en un solo paso, el resultado de la simplificación.

A continuación se presentan varios ejercicios resueltos para comprobar lo eficaz del método:

Ejercicio 3.1. Dada la siguiente tabla de la verdad, minimice la función de conmutación utilizando mapas K y realice la implementación del mismo.

#	m	n	p	q	F
0	0	0	0	0	0
1	0	0	0	1	1
2	0	0	1	0	0
3	0	0	1	1	1
4	0	1	0	0	0
5	0	1	0	1	1
6	0	1	1	0	0
7	0	1	1	1	1
8	1	0	0	0	0
9	1	0	0	1	0
10	1	0	1	0	0
11	1	0	1	1	0
12	1	1	0	0	1
13	1	1	0	1	1
14	1	1	1	0	1
15	1	1	1	1	0

Obteniendo los minterms y maxterms de la tabla:

$$F(m,n,p,q) = \sum_m (1,3,5,7,12,13,14)$$

$$F(m,n,p,q) = \prod_M (0,2,4,6,8,9,10,11,15)$$

La solución se puede obtener agrupando minterms; ver figura 3.9(a), o por agrupación de maxterms; ver figura 3.9(b). En cualquiera de los casos los niveles lógicos de salida del circuito de compuertas debe ser el mismo; ver figura 3.10. La función simplificada queda de la siguiente manera:

$$F(m,n,p,q) = \bar{m}q + mn\bar{p} + mn\bar{q} \quad \text{Suma de productos.}$$

$$F(m,n,p,q) = (m+q)(\bar{m}+n)(\bar{m}+\bar{p}+\bar{q}) \quad \text{Producto de sumas.}$$

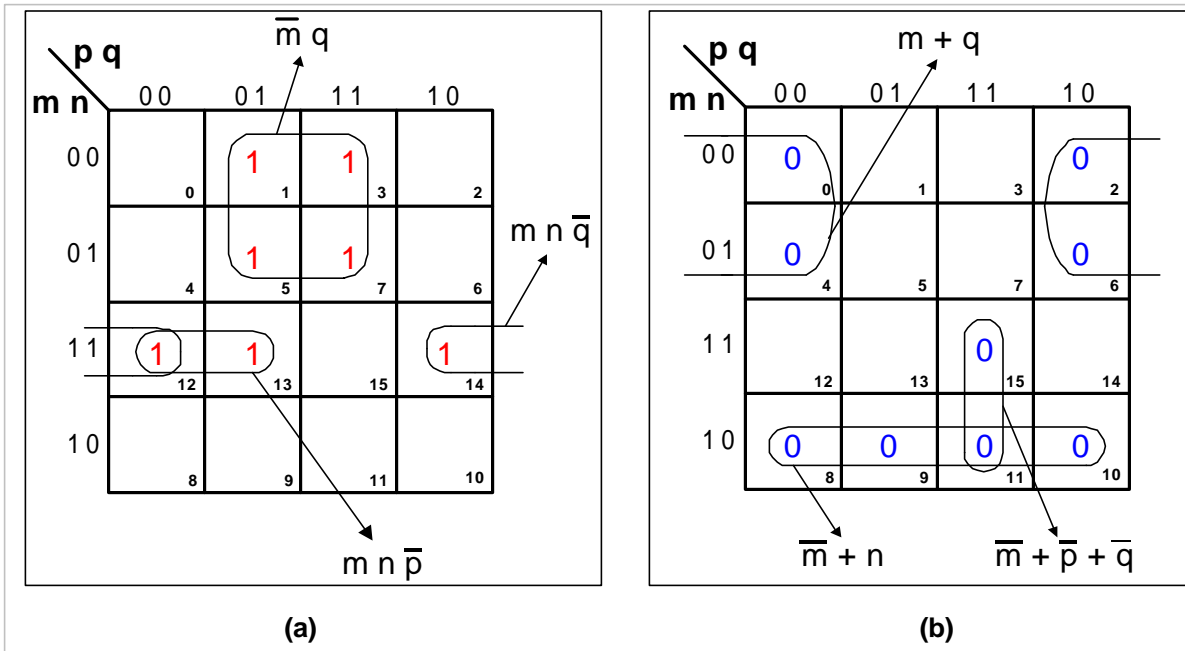


Figura 3.9. Simplificación de la función por minterms y maxterms mediante mapas K.

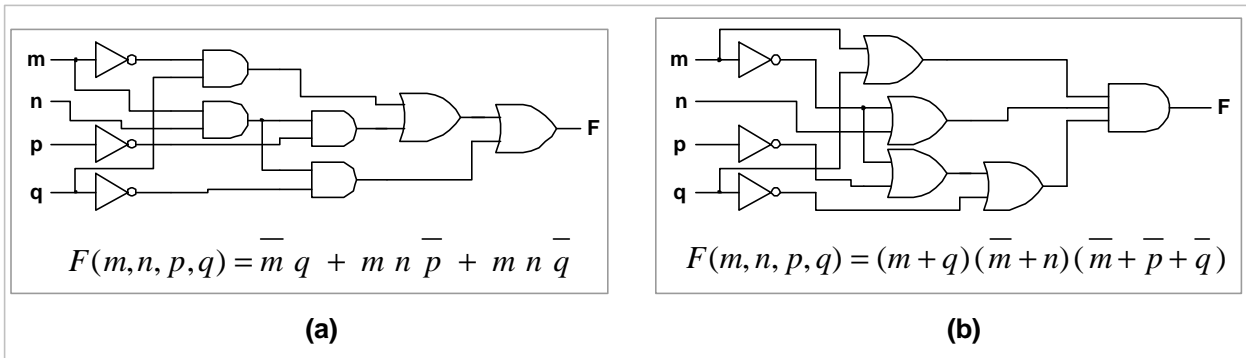


Figura 3.10. Circuito de compuertas con suma de productos (a), y con producto de sumas (b).

Ejercicio 3.2. Dada la siguiente función:

$$F(a,b,c,d,e) = \sum_m (0,1,3,4,7,9,10,11,12,16,17,18,19,23,25,28,31)$$

Minimízela con mapas de Karnaugh e implementarla con compuertas NAND.

Solución: La figura 3.11. muestra la simplificación mediante mapa de Karnaugh y la función resultante. El circuito de compuertas NAND es mostrado en la figura 3.12.

Los minterms que forman grupos de 4 son: (1, 9, 17, 25); (3, 7, 19, 23); (16, 17, 18, 19); (0, 1, 16, 17).

Los minterms que forman grupos de 2 son: (4, 12); (10, 11); (23, 31); (12, 28).

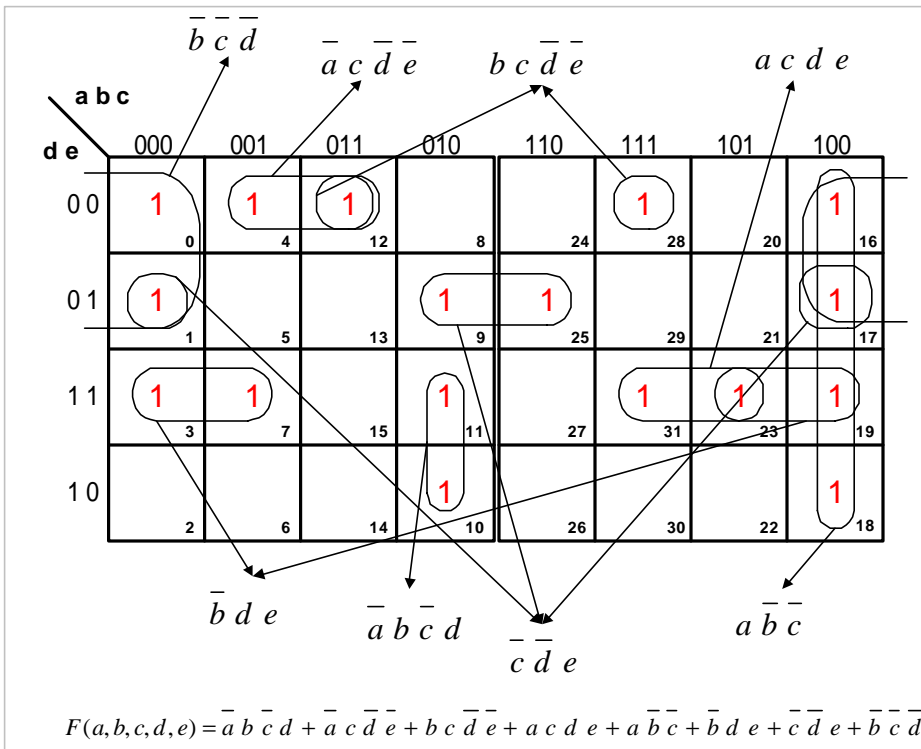


Figura 3.11. Simplificación de la función de conmutación del ejercicio 3.2.

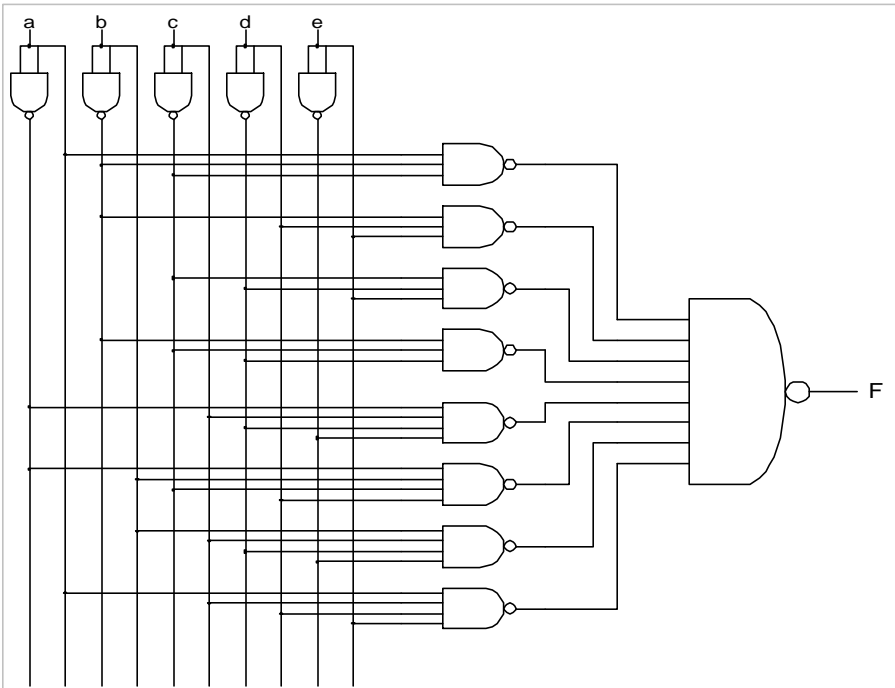


Figura 3.12. Circuito de compuertas NAND del ejercicio 3.2.

Ejercicio 3.3. Dada la siguiente función:

$$F(m,n,o,p,q) = \prod_M (1,2,5,7,8,12,13,15,21,23,24,25,26,28,29,31)$$

Simplifíquela utilizando mapas de Karnaugh.

Solución: La figura 3.13. muestra la simplificación mediante mapa de Karnaugh y la función resultante. La reducción de la función se realiza agrupando los maxterms.

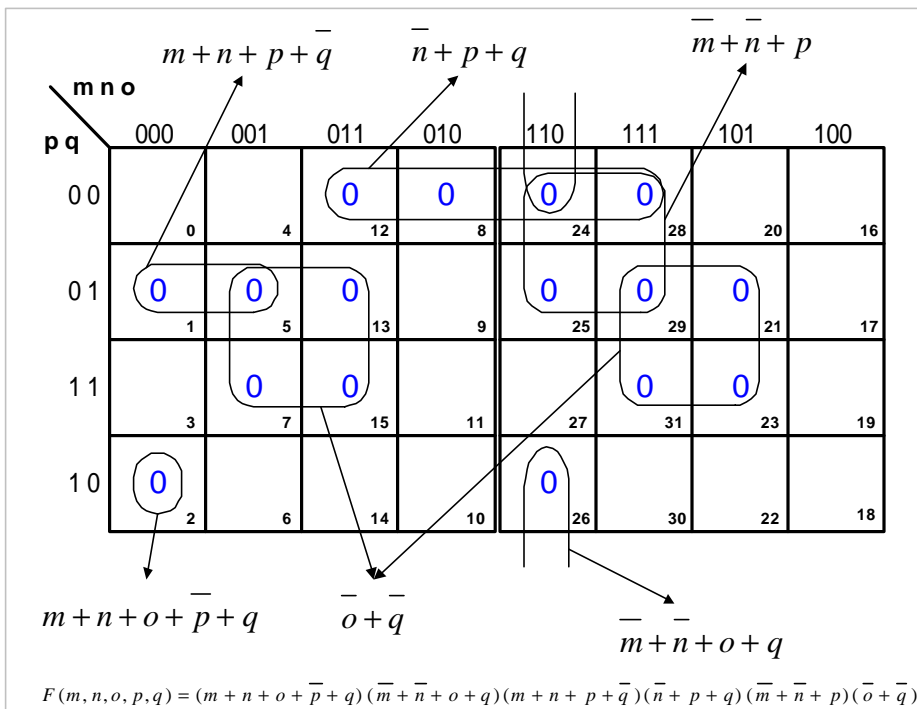


Figura 3.13. Minimización de la función del ejercicio 3.3 mediante mapa de Karnaugh.

Los maxterms que forman grupo de 8 son: (5, 7, 13, 15, 21, 23, 29, 31).

Los maxterms que forman grupos de 4 son: (8, 12, 24,28); (24, 25, 28, 29).

Los maxterms que forman grupos de 2 son: (1, 5); (24, 26).

Los maxterms que forman grupo de 1 son: (2).

3.1.2 Términos y entradas indiferentes.

Existen valores en las variables de entrada de un circuito digital que nunca se presentan y por tanto, no afectan la salida del circuito. Por ejemplo, en la figura 3.14 se observa un convertidor de código BCD a Exceso 3; este circuito necesita que se conecten en la entrada los diez primeros valores binarios desde cero (0000) hasta nueve (1001). No obstante, las cuatro variables de entrada (w, v, y, z) pueden llegar hasta quince (1111). La combinación de valores comprendidos entre (1010) y (1111) no deben estar presentes; por lo cual, la salida del circuito digital puede tomar un nivel lógico indiferente, puede valer uno "1", o puede valer cero "0". Estas combinaciones de los literales de la función que hacen que la salida pueda tomar cualquier valor lógico binario se denominan términos indiferentes.

La salida de términos indiferentes se marca con "X", "d" ó "-" en la tabla de la verdad, y pueden ser utilizados en el mapa de Karnaugh para ayudar a simplificar la función de conmutación. A continuación se presenta el diseño del convertidor de código:

n	Código BCD				Código Exceso 3				
	w	v	y	z	F ₃	F ₂	F ₁	F ₀	
0	0	0	0	0	0	0	1	1	Código Válido
1	0	0	0	1	0	1	0	0	
2	0	0	1	0	0	1	0	1	
3	0	0	1	1	0	1	1	0	
4	0	1	0	0	0	1	1	1	
5	0	1	0	1	1	0	0	0	
6	0	1	1	0	1	0	0	1	
7	0	1	1	1	1	0	1	0	
8	1	0	0	0	1	0	1	1	
9	1	0	0	1	1	1	0	0	
10	1	0	1	0	X	X	X	X	Términos Indiferentes
11	1	0	1	1	X	X	X	X	
12	1	1	0	0	X	X	X	X	
13	1	1	0	1	X	X	X	X	
14	1	1	1	0	X	X	X	X	
15	1	1	1	1	X	X	X	X	

Tabla de la verdad del convertidor BCD – Exc 3.

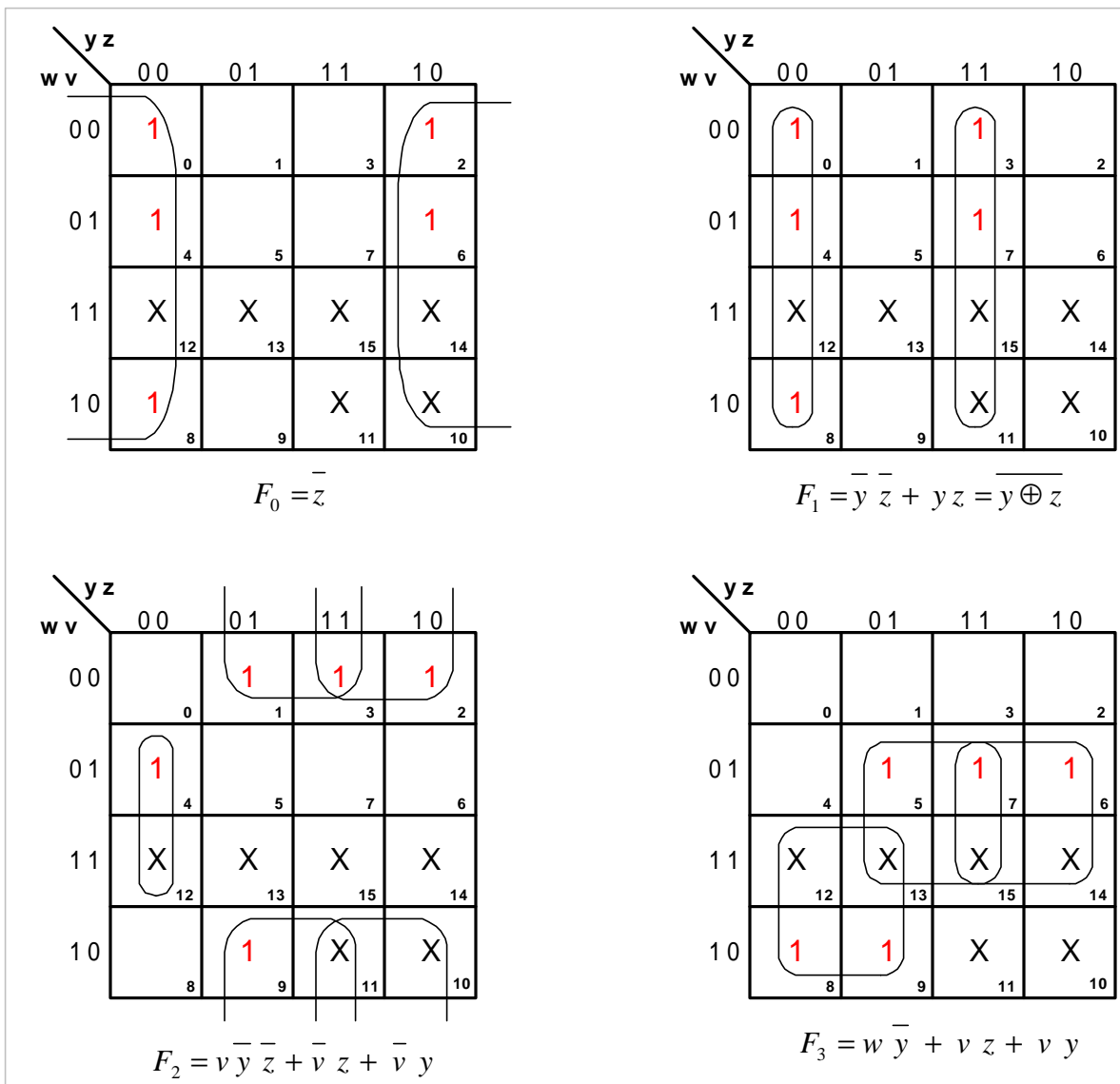


Figura 3.14. Simplificación con mapas K, agrupando como unos los términos indiferentes.

Los términos indiferentes son tomados como minterms con el objeto de obtener una mejor minimización de circuito digital. Se utilizan menos compuertas, pero, hay que tener cuidado de no colocar en la entrada del circuito la combinación de algún término indiferente. Porque la respuesta del circuito presentará una salida no deseada. Por lo tanto; para no cometer, errores se debe estar seguro que ninguno de los términos indiferentes estará presente en la entrada del circuito. La figura 3.15 muestra el circuito digital de compuertas que ha sido reducido con los términos indiferentes.

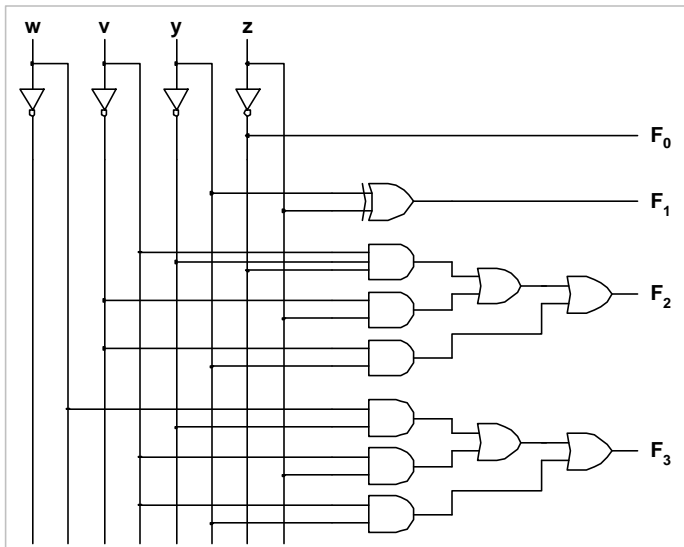


Figura 3.15. Circuito de compuertas del convertidor BCD – Exc 3.

También existen las entradas indiferentes que pueden trabajar de forma tal; que, para una o más variables, la salida del circuito mantenga un mismo nivel lógico. Un ejemplo de ello sería la compuerta NAND de dos o más entradas. Esta compuerta mantiene su salida en uno lógico cuando alguna de sus entradas se coloca en nivel cero. Por lo cual, esas entradas de la NAND pueden cambiar su nivel sin alterar el valor de uno lógico en la salida de la compuerta. Estas entradas que no cambian el estado de la salida cuando otro dispositivo, de mayor prioridad, toma el control del circuito se denominan entradas indiferentes y se representan con “X”, “d” o “-”.

3.2 Implicantes primos.

Los implicantes son una terminología empleada por diseñadores de circuitos digitales para indicar agrupamientos que se forman con los literales de una función de conmutación. En la forma algebraica los implicantes expresan el producto o la suma de literales ya sea en forma simplificada o forma normal. Esto significa que un implicante sirve para cubrir un minterm o maxterm en la función lógica. En el mapa K los implicantes son el resultado de los grupos que se puedan formar mediante las adyacencias. La figura 3.16 muestra todos los 19 implicantes que se pueden formar con los agrupamientos de los minterms, los grupos de uno también son válidos para formar implicantes.

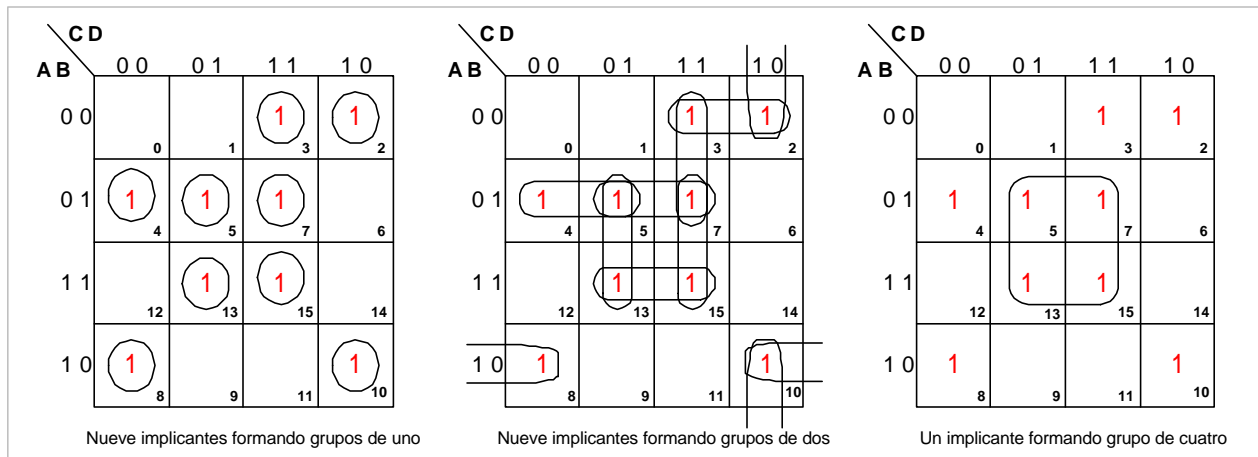


Figura 3.16. Grupos de 19 implicantes que se forman en la función de conmutación.

Los cuatro implicantes que abarcan dos minterms (5, 7); (7, 15); (13, 15) y (5, 13) forma otro implicante de cuatro minterms (5, 7, 13, 15). Este último es un superconjunto de los implicantes pares anteriores y más aún, de los implicantes de un solo minterm.

Los implicantes que forman el máximo agrupamiento de minterms (o maxterms) posible se conocen como **implicantes primos** de la función de conmutación y son los que determinan la simplificación del circuito digital. La figura 3.17 muestra todos los implicantes primos de la función lógica que se pueden obtener en el mapa de Karnaugh. Existen implicantes primos que abarcan los mismos minterms; por ejemplo, los dos implicantes primos (2, 10) y (3, 7) cubren los minterms del otro implicante primo (2, 3); por lo cual, este último se denomina **implicante primo redundante**.

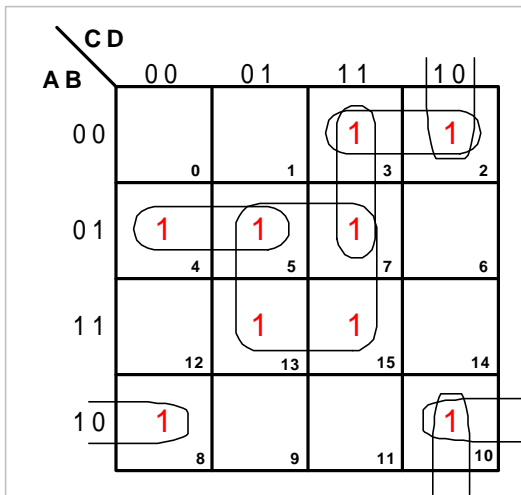


Figura 3.17. Todos los implicantes primos de la función.

Los grupos que cubren algún minterm que no es abarcado por otro implicante se denominan **implicante primo esencial**. Los términos resultantes de la simplificación de la función contiene todos los implicantes primos esenciales más los implicantes primos que no sean redundantes. Los implicantes primos esenciales del mapa K de la figura 3.17 son: $AD + \bar{A}B\bar{C} + A\bar{B}\bar{D}$ por otra parte, los implicantes primos no esenciales de la función son: $\bar{A}CD + \bar{B}C\bar{D} + \bar{A}\bar{B}C$

La minimización de las funciones de conmutación pueden presentar varias alternativas mínimas de solución; llamadas **cubierta mínima** de la función. Del mapa K anterior se obtienen, la **cubierta completa** y las cubiertas mínimas:

$F = AD + \bar{A}B\bar{C} + A\bar{B}\bar{D} + \bar{A}CD + \bar{B}C\bar{D} + \bar{A}\bar{B}C$ Cubierta completa con todos los implicantes primos.

$F = AD + \bar{A}B\bar{C} + A\bar{B}\bar{D} + \bar{A}CD + \bar{B}C\bar{D}$ Simplificación con cubierta no mínima.

$F = AD + \bar{A}B\bar{C} + A\bar{B}\bar{D} + \bar{A}\bar{B}C$ Simplificación con cubierta mínima.

La figura 3.18 describe la simplificación de los resultados de la cubierta mínima y cubierta no mínima. Lo importante en un diseño digital es hallar la cubierta mínima con el fin de ahorrar costo y espacio en la implementación del circuito digital de compuertas.

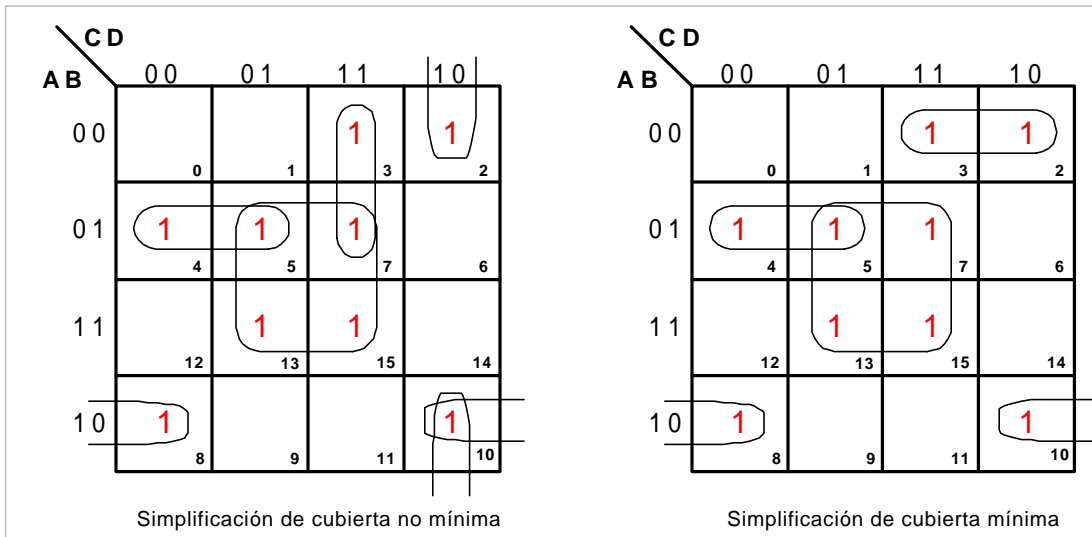


Figura 3.18. Cubiertas no mínima y cubierta mínima de la función lógica.

A continuación se presentan un ejercicio propuesto con tres simplificaciones de funciones de conmutación.

Ejercicio 3.4. Hallar la cubierta completa y la cubierta mínima de las funciones siguientes:

1. $F = \sum_m (0, 4, 6, 7, 8, 9, 11, 13, 15, 17, 20, 22, 24, 25, 27, 28, 31)$
2. $G = \prod_M (1, 2, 3, 4, 6, 8, 11, 12, 13)$
3. $H = \sum_m (0, 2, 5, 9, 12, 14, 16, 17, 15, 17, 22, 25, 26, 27) + d(1, 6, 7, 24, 29)$

Los términos encerrados dentro del paréntesis presidido por la letra “d”, esto es: $d(1, 6, 7, 24, 29)$ son términos indiferentes.

3.3 Minimización de funciones mediante el método de Quine-McClueskey.

Es un método tabular donde se van simplificando las variables de la función a medida que se generan tablas con índices. La ventaja de este método, con respecto al método de Karnaugh, es que se puede simplificar funciones de conmutación con un número de variables superior a seis. No obstante, manualmente, es muy laborioso cuando existen muchas variables y términos de por medio. El método de Quine-McClueskey (Q-M) es efectivo para ser aplicado como algoritmo en algún lenguaje de programación.

La simplificación se realiza en minterms; primero se organizan los minterms por índice. Este índice es la suma de los bits que están con nivel lógico uno "1" en el término. Por ejemplo, los minterms $\bar{a}\bar{b}\bar{c}\bar{d}\bar{e}\bar{f}$, $a\bar{b}\bar{c}\bar{d}\bar{e}\bar{f}$, $\bar{a}b\bar{c}\bar{d}\bar{e}\bar{f}$ tienen índice 0, 3 y 5 respectivamente.

Los índices se organizan en una tabla; primero índice cero, luego todos los de índice uno, después todos los de índice dos y así sucesivamente hasta llegar al de mayor índice.

Luego, se realizan los cruces de minterms, cuyos índices tengan una diferencia numérica que deberá ser múltiplo de la base binaria. Se deben restar los de menor índice con los de mayor índice inmediato en la tabla; por lo cual, la diferencia siempre será negativa.

En los valores negativos, múltiplos de dos, debe cambiar solamente un bit y la variable que corresponde a esa posición queda simplificada. Por lo tanto, esa posición debe ser sustituida por un símbolo que indique un valor indiferente; como puede ser un guión "-", una equis "x" o la letra "d".

Con los términos simplificados se va construyendo otra tabla secundaria, se obtienen nuevos índices y se repite el procedimiento. Sin embargo, a partir de la segunda tabla los términos que se pueden reducir, además de ser negativos y múltiplos de potencia dos, son los que presentan coordenadas indiferentes ubicadas en la misma posición. A medida que se van construyendo tablas se deben marcar y/o numerar los términos que no pueden cruzarse con otros. Esta tabulación termina cuando no haya más cruces por realizar.

Los términos de las tablas que fueron reducidos y los que no se cruzaron con otros son organizados en una matriz tipo grilla (**matriz de implicantes**) donde deben ser colocados todos los minterms involucrados en las expresiones reducidas y no reducidas. Para luego ir descartando los términos que son redundantes en la matriz y obtener una expresión minimizada de la función de conmutación. En esta etapa puede ser necesario realizar reducciones de las matrices e ir obteniendo otras secundarias (**matriz de implicantes reducida**) hasta que la selección sea la más óptima posible. A continuación se presentan dos ejercicios para ilustrar mejor el método.

Ejercicio 3.5. Simplificar por el método de Q-M la siguiente función de conmutación dada como lista de maxterms: $G = \prod_M (2,3,5,6,8,10,14,15,17,18,19,22,23,24,26,30)$

Solución: Primero se obtiene la forma minterms.

$$G = \sum_m (0,1,4,7,11,12,13,16,20,21,25,27,28,29,31)$$

Luego ordenamos los minterms en una tabla de índices con su equivalente binario

Se forma la primera tabla ordenando los productos minterms según el índice correspondiente. Aquí se realizan las comparaciones o cruces de los minterms de índice inferior con los de índice superior inmediato. Si la diferencia es negativa y múltiplo de base dos, entonces debe existir cambio de estado en una sola variable. Por ejemplo, el minterm cero (00000) de índice 0 combina con el minterm uno (00001) de índice 1; por lo tanto, la segunda tabla debe construirse comenzando también con índice cero y la combinación del cruce anterior (0-1 = 0000_), el guión señala que la variable de esa posición ha sido simplificada. Sucesivamente se van realizando las combinaciones y formando la segunda tabla; estas combinaciones que van formando la segunda tabla son las adyacencias contiguas de la función de conmutación, y se conocen como **adyacencias de primer orden**. No obstante, el cruce del minterm 9 de índice dos no se puede cruzar con el minterm 7 porque la diferencia es positiva; pero, si se combina con 11 para formar el nuevo término de índice 2: (9-11 = 010_1).

Los términos que se combinan hay que marcarlos; también se marcan, en secuencia u orden alfabético, los productos que no se combinen. Como por ejemplo, el T_1 (minterm 7) de la primera tabla y el término T_3 (1-9 = 0_001) de la segunda tabla no

se combinan. Estos términos que no se combinan son los **implicantes primos** de la simplificación de la función y deben aparecer en la solución final del problema.

Índice	Minterms (Dec - Bin)	Señal
0	0 = 00000	∇
1	1 = 00001	∇
	4 = 00100	∇
	16 = 10000	∇
2	9 = 01001	∇
	12 = 01100	∇
	20 = 10100	∇
3	7 = 00111	T₁
	11 = 01011	∇
	13 = 01101	∇
	21 = 10101	∇
	25 = 11001	∇
28 = 11100	∇	
4	27 = 11011	∇
	29 = 11101	∇
5	31 = 11111	∇

Tabla primaria.

Índice	Término	Señal
0	0-1 = 0000_	T₂
	0-4 = 00_00	∇
	0-16 = _0000	∇
1	1-9 = 0_001	T₃
	4-12 = 0_100	∇
	4-20 = _0100	∇
2	16-20 = 10_00	∇
	9-11 = 010_1	∇
	9-13 = 01_01	∇
	9-25 = _1001	∇
	12-13 = 0110_	∇
	12-28 = _1100	∇
	20-21 = 1010_	∇
	20-28 = 1_100	∇
3	11-27 = _1011	∇
	13-29 = _1101	∇
	21-29 = 1_101	∇
	25-27 = 110_1	∇
	25-29 = 11_01	∇
	28-29 = 1110_	∇
4	27-31 = 11_11	∇
	29-31 = 111_1	∇

Segunda tabla.

En la tercera tabla, se colocan los nuevos índices de las combinaciones de la segunda; se colocan las marcas y se procede a señalar los términos implicantes de la función.

Los términos que se producen en la tercera tabla provienen de la combinación de la segunda. Tomando en cuenta que deben coincidir, en los dos índices consecutivos, las coordenadas del guión y; además, cumplir con la siguiente condición: La diferencia de los minterms colocado por pares entre paréntesis debe ser negativa, múltiplo de base dos e igual valor numérico. Este valor indica una posición de coordenada indeterminada (guión). Por otra parte, la diferencia de los minterms de un paréntesis con respecto al otro señala la posición de la otra coordenada indeterminada.

Por ejemplo, el grupo [(25-27)-(29-31)] señalan que debe haber un guión en la posición binaria 2; esto es (25-27 = -2); (29-31 = -2) y; de la misma forma, el otro guión estará en [25-29 = -4], [27-31 = -4]. En la tercera tabla, se escribe el valor binario del primer minterm del primer paréntesis y se colocan los respectivos guiones.

En la tercera tabla el término de índice 3 esta formado por: [(25-27)-(29-31) = 11__1].

Índice	Minterms (Dec - Bin)	Señal
0	(0-4)-(16-20) = _0_00	T ₄
1	(4-12)-(20-28) = __100	T ₅
2	(9-11)-(25-27) = _10_1	T ₆
	(9-13)-(25-29) = _1_01	T ₇
	(12-13)-(28-29) = _110_	T ₈
	(20-21)-(28-29) = 1_10_	T ₉
3	(25-27)-(29-31) = 11__1	T ₁₀

Tercera tabla

Los términos de la tercera tabla no se cruzan por lo tanto, finaliza la simplificación de la función de conmutación. El resultado de la función es la suma de todos los términos que fueron marcados en la construcción de las tablas. Los guiones indican que las variables de esas posiciones fueron simplificadas.

Por ejemplo, al tomar las variables a, b, c, d y e.

$$G(a,b,c,d,e) = T_1 + T_2 + T_3 + T_4 + T_5 + T_6 + T_7 + T_8 + T_9 + T_{10}$$

$$G(a,b,c,d,e) = \bar{a}\bar{b}cde + \bar{a}\bar{b}\bar{c}\bar{d} + \bar{a}c\bar{d}\bar{e} + \bar{b}\bar{d}\bar{e} + c\bar{d}\bar{e} + b\bar{c}\bar{e} + b\bar{d}\bar{e} + bc\bar{d} + ac\bar{d} + ab\bar{e}$$

En esta función, escrita como suma de productos, aparecen todos los implicantes relacionados con la minimización. No obstante, la función puede ser reducida a su mínima expresión, llamada cubierta mínima; existe la posibilidad de obtener varias soluciones válidas para el problema; la forma de realizar ésto, es utilizando una **matriz de implicantes** para eliminar los términos que sean reducibles en la función de conmutación. En la figura 3.19 se muestra la simplificación, utilizando la matriz de implicantes.

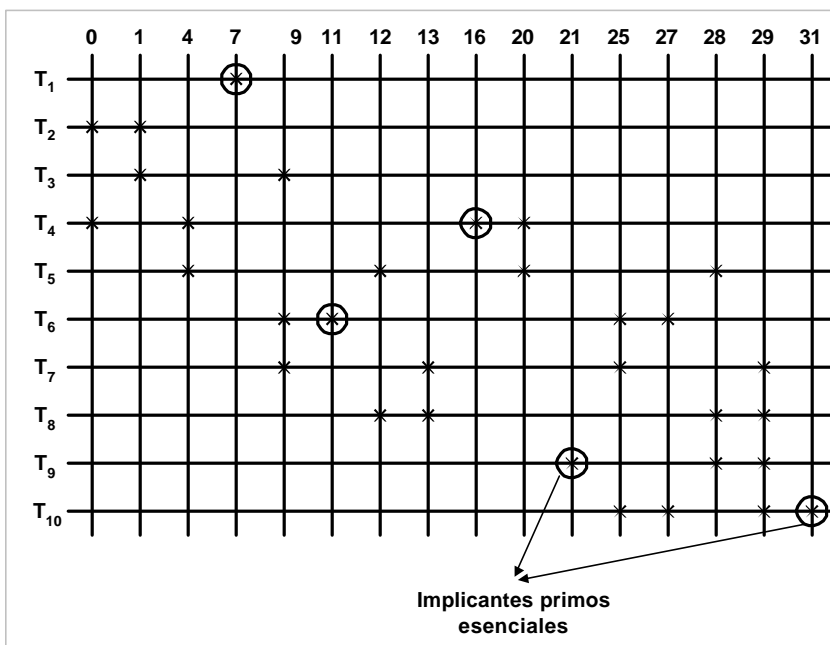


Figura 3.19. Matriz de implicantes del ejercicio 3.5.

Las filas corresponden a los implicantes primos, obtenidos con la reducción de las tablas de índices y, en las columnas van todos los minterms de la función de conmutación. En la intersección se coloca una marca (en este caso una x) si el implicante de la fila incluye al minterm de la columna. Analizando la matriz, se puede observar los implicantes que cubren un solo minterm; por ejemplo, T₁, T₄, T₆, T₉ y T₁₀.

La expresión resultante mínima debe incluir estos implicantes primos esenciales que sirven; a su vez, para reducir la matriz de implicantes primos. Los minterms involucrados, conjuntamente con los implicantes primos esenciales, son eliminados de la **matriz de implicantes primos reducida**.

Son eliminadas las siguientes filas y columnas:

Fila: (implicantes primos esenciales)	Columna: (minterms)
T ₁	7
T ₄	0, 4, 16, 20
T ₆	9, 11, 25, 27
T ₉	20, 21, 28, 29
T ₁₀	25, 27, 29, 31

Implicantes y minterms eliminados de la matriz.

	1	12	13
T ₂	*		
T ₃	*		
T ₅		*	
T ₇			*
T ₈		*	*

Matriz de implicantes primos reducida.

El implicante primo T₈ abarca a los minterms 12 y 13; por lo que, los implicantes T₅ y T₇ quedan descartados, T₈ debe ser seleccionado para la solución final. No obstante, se puede elegir entre T₂ o T₃. La minimización tiene dos soluciones válidas; una incluye T₂ y la otra incluye T₃.

$$G = T_1 + T_2 + T_4 + T_6 + T_8 + T_9 + T_{10}$$

$$G = T_1 + T_3 + T_4 + T_6 + T_8 + T_9 + T_{10}$$

$$G(a,b,c,d,e) = \bar{a}\bar{b}cde + \bar{a}\bar{b}\bar{c}\bar{d} + \bar{b}\bar{d}\bar{e} + \bar{b}\bar{c}e + b\bar{c}\bar{d} + a\bar{c}\bar{d} + abe \quad \text{Solución 1.}$$

$$G(a,b,c,d,e) = \bar{a}\bar{b}cde + \bar{a}\bar{c}\bar{d}e + \bar{b}\bar{d}\bar{e} + \bar{b}\bar{c}e + b\bar{c}\bar{d} + a\bar{c}\bar{d} + abe \quad \text{Solución 2.}$$

El método es engorroso manualmente pero, es muy útil, cuando se implementa en una computadora como algoritmo de algún lenguaje de programación.

3.4 Funciones multiterminales.

Son funciones digitales combinacionales que poseen varias salidas que trabajan en forma conjunta, donde las variables de entrada son comunes a dichas funciones. Ejemplo de función multiterminal es el circuito de la figura 3.21; el cual muestra un restador de dos bits diseñado con compuertas digitales. La simplificación mediante mapas K, mostrado en la figura 3.20, proporcionan la minimización de las tres funciones; la salida **S** del circuito indica el signo negativo del resultado y las otras dos salidas **R₁** y **R₀**, el resultado de la operación. Cuando la diferencia es negativa la salida **S**, se coloca en uno lógico y la magnitud del resultado quedan en **R₁** y **R₀**.

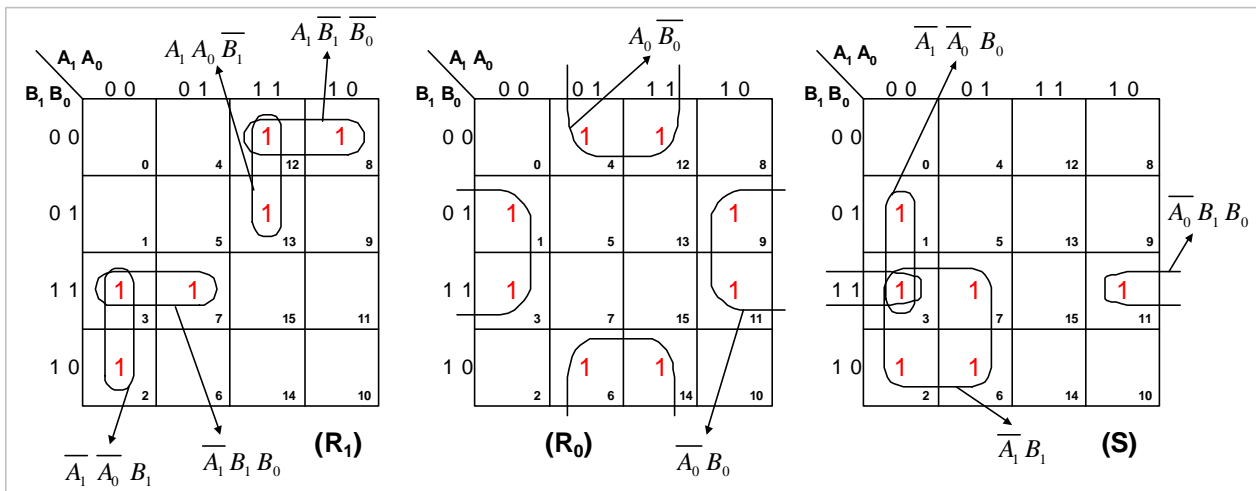


Figura 3.20. Diseño del circuito restador multiterminal de tres salidas (S, R1 y R0).

Las tres funciones simplificadas quedan expresadas de la siguiente forma:

$$S = \overline{A_1} B_1 + \overline{A_1} \overline{A_0} B_0 + \overline{A_0} B_1 B_0$$

$$R_1 = A_1 \overline{B_1} \overline{B_0} + \overline{A_1} \overline{A_0} B_1 + \overline{A_1} B_1 B_0 + A_1 A_0 \overline{B_1}$$

$$R_0 = A_0 \overline{B_0} + \overline{A_0} B_0 = A_0 \oplus B_0$$

La figura 3.21 muestra el circuito de compuertas digitales correspondiente a este diseño, las compuertas AND de tres entradas son 74LS11 y las de dos entradas son 74LS08. Las compuertas OR son 74LS32 y los inversores pertenecen al chip 74LS04. El circuito posee cuatro líneas de entradas, dos bits por cada dato, y tres líneas de salida. Por lo tanto, este circuito corresponde con una función multiterminal.

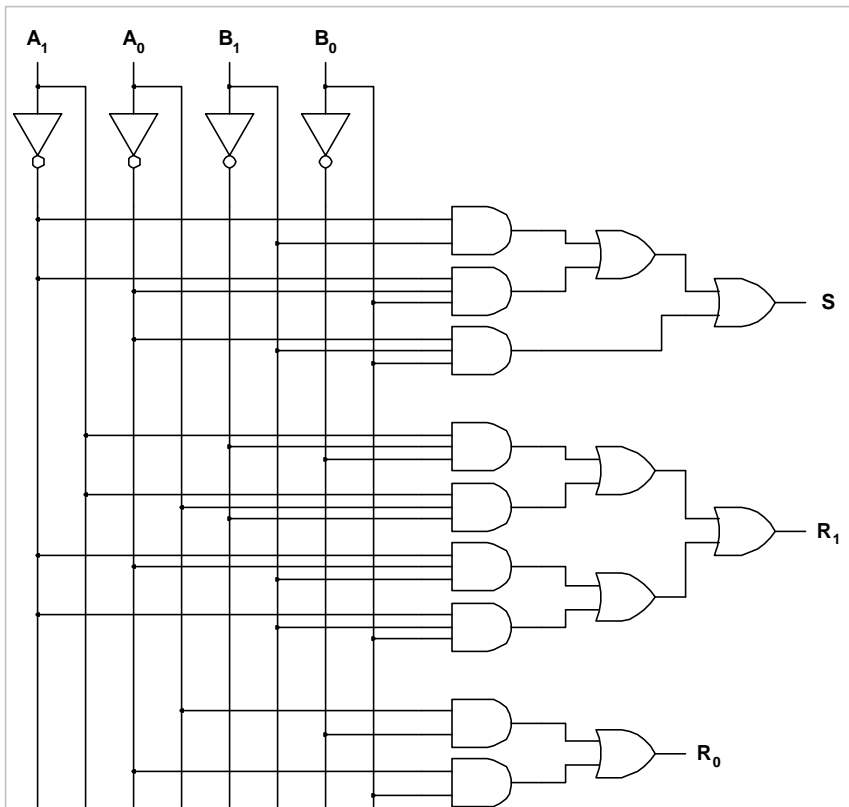


Figura 3.21. Circuito de compuertas, restador de dos bits con indicador de signo.

3.5 Simplificación simultanea de funciones.

Por lo general, los sistemas digitales no son tan sencillos como para que tengan una sola salida, de manera que es frecuente tener que sintetizar simultáneamente varias funciones con las mismas variables de entrada. En estos casos es conveniente simplificar de forma que haya coincidencia en la salida de algunas compuertas para que compartan señales comunes. De esta forma, los productos o las sumas puedan ser las mismas en las distintas funciones que interviene en el circuito digital de compuertas.

Se debe tener precaución en la simplificación simultanea de funciones, ya que puede traer como consecuencia la utilización de un mayor número de compuertas en el circuito. En el ejercicio 3.6 se describen dos formas de simplificar varias funciones utilizando mapas de Karnaugh; la primera minimización, mostrada en la figura 3.22 no es optima porque los implicantes no fueron debidamente agrupados. Por otra parte, el agrupamiento realizado en la figura 3.23 es el más adecuado para la síntesis del circuito digital.

Ejercicio 3.6. Simplificar simultáneamente las dos funciones descritas a continuación, utilizando mapas de Karnaugh.

$$F_1(w, x, y, z) = \sum_m (1, 3, 5, 7, 10, 11, 14, 15)$$

$$F_2(w, x, y, z) = \sum_m (1, 5, 10, 12, 13, 14, 15)$$

Solución: La figura 3.22 muestra un mal agrupamiento de los implicantes primos; aquí la minimización de las funciones utilizan grupos no comunes de minterms y por lo tanto el circuito resultante será más grande.

$$F_1(w, x, y, z) = \bar{w}z + wy$$

$$F_2(w, x, y, z) = \bar{w}\bar{y}z + wy\bar{z} + wx$$

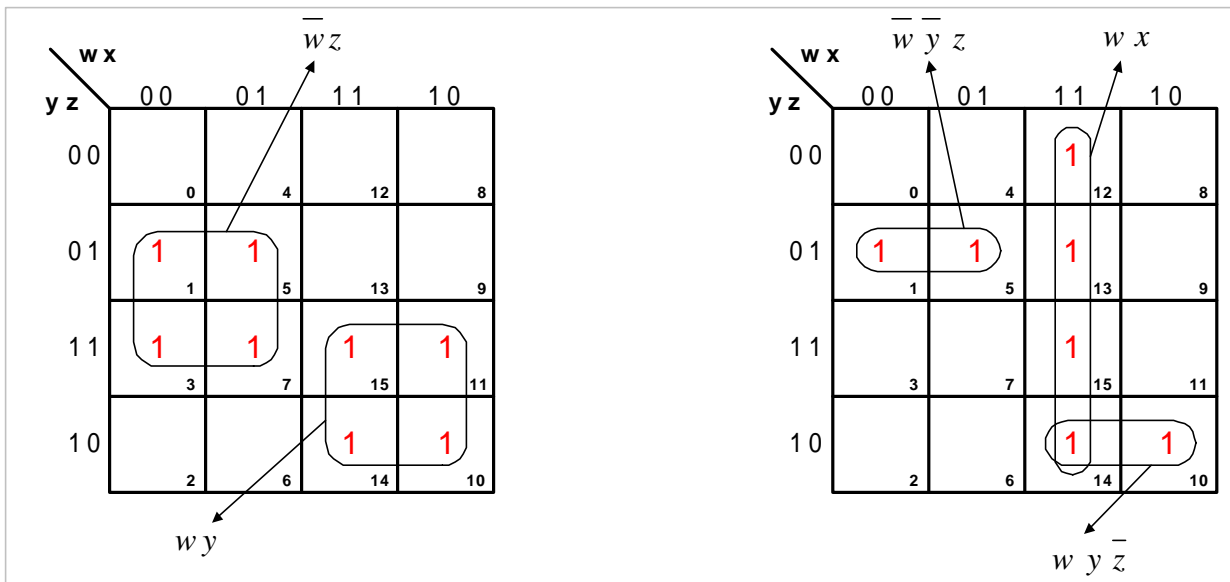


Figura 3.22. Simplificación con implicantes no comunes de F_1 y F_2 con mapas K. Ejercicio 3.6.

La síntesis del circuito digital, ver figura 3.24, queda reducido a 11 compuertas; dos AND de tres entradas, tres AND de dos entradas, tres OR de dos entradas y tres compuertas NOT. Por otra parte, la figura 3.23 muestra la simplificación simultanea de las dos funciones F_1 y F_2 mediante mapas K; las agrupaciones se forman para que los implicantes coincidan y las señales de las compuertas $(\bar{w}\bar{y}z)$, $(wy\bar{z})$ sean comunes en el circuito. La figura 3.24 describe ésta diferencia que se traduce en la reducción de una compuerta AND de dos entradas.

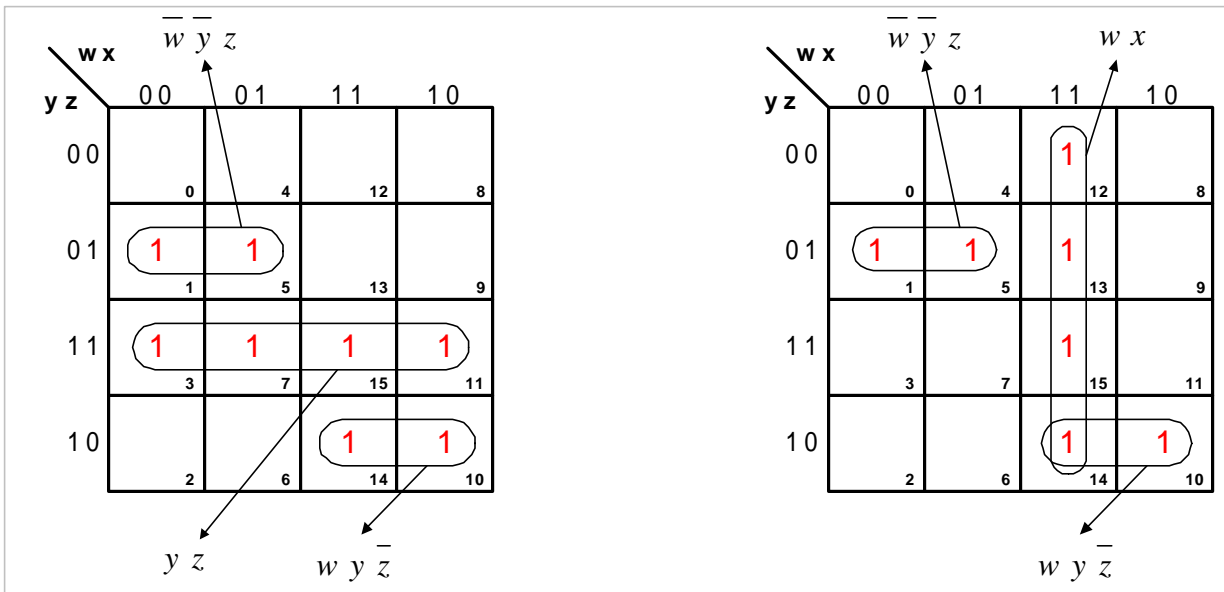


Figura 3.23. Simplificación con implicantes comunes de F_1 y F_2 con mapas K. Ejercicio 3.6.

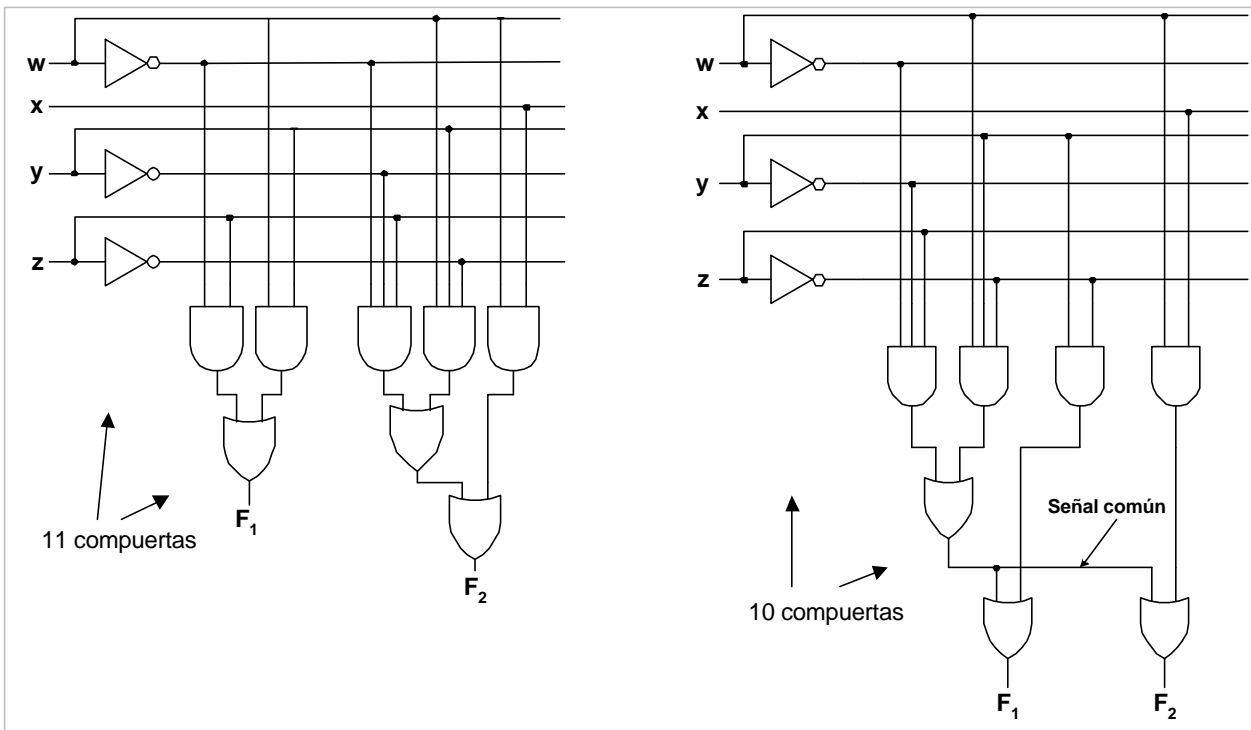


Figura 3.24. Comparación de los circuitos de compuertas F_1 y F_2 del ejercicio 3.6.

3.5 Aplicaciones.

Los métodos de simplificación coadyuvan al diseño de los circuitos digitales combinacionales de compuertas; por lo cual, son más eficaces que el método simplificación algebraica. Por lo tanto, existe una mayor optimización en la implementación del circuito digital.

Las aplicaciones de circuitos digitales combinacionales comprenden los circuitos de salida simple y los circuitos de salidas multiterminal. A continuación se presentan varios ejercicios propuestos para ser resueltos por los métodos de minimización explicados anteriormente.

Ejercicio 3.7. Simplificar con mapas K, e implementar con compuertas básicas la siguiente función dada en la tabla de la verdad.

n	p	q	r	m	n	o	F
0	0	0	0	0	0	0	0
1	0	0	0	0	0	1	0
2	0	0	0	0	1	0	0
3	0	0	0	0	1	1	0
4	0	0	0	1	0	0	0
5	0	0	0	1	0	1	0
6	0	0	0	1	1	0	0
7	0	0	0	1	1	1	1
8	0	0	1	0	0	0	X
9	0	0	1	0	0	1	X
10	0	0	1	0	1	0	1
11	0	0	1	0	1	1	0
12	0	0	1	1	0	0	X
13	0	0	1	1	0	1	0
14	0	0	1	1	1	0	1
15	0	0	1	1	1	1	0

n	p	q	r	m	n	o	F
16	0	1	0	0	0	0	0
17	0	1	0	0	0	1	1
18	0	1	0	0	1	0	0
19	0	1	0	0	1	1	1
20	0	1	0	1	0	0	0
21	0	1	0	1	0	1	0
22	0	1	0	1	1	0	0
23	0	1	0	1	1	1	1
24	0	1	1	0	0	0	1
25	0	1	1	0	0	1	1
26	0	1	1	0	1	0	1
27	0	1	1	0	1	1	0
28	0	1	1	1	0	0	1
29	0	1	1	1	0	1	0
30	0	1	1	1	1	0	1
31	0	1	1	1	1	1	0

n	p	q	r	m	n	o	F
32	1	0	0	0	0	0	0
33	1	0	0	0	0	1	X
34	1	0	0	0	1	0	0
35	1	0	0	0	1	1	1
36	1	0	0	1	0	0	0
37	1	0	0	1	0	1	0
38	1	0	0	1	1	0	0
39	1	0	0	1	1	1	1
40	1	0	1	0	0	0	X
41	1	0	1	0	0	1	X
42	1	0	1	0	1	0	1
43	1	0	1	0	1	1	0
44	1	0	1	1	0	0	X
45	1	0	1	1	0	1	0
46	1	0	1	1	1	0	1
47	1	0	1	1	1	1	0

n	p	q	r	m	n	o	F
48	1	1	0	0	0	0	0
49	1	1	0	0	0	1	1
50	1	1	0	0	1	0	0
51	1	1	0	0	1	1	1
52	1	1	0	1	0	0	0
53	1	1	0	1	0	1	0
54	1	1	0	1	1	0	0
55	1	1	0	1	1	1	1
56	1	1	1	0	0	0	1
57	1	1	1	0	0	1	1
58	1	1	1	0	1	0	1
59	1	1	1	0	1	1	0
60	1	1	1	1	0	0	1
61	1	1	1	1	0	1	0
62	1	1	1	1	1	0	1
63	1	1	1	1	1	1	0

Tabla de la verdad para el ejercicio 3.7.

Ejercicio 3.8. Hallar todos los implicantes primos e implicantes primos esenciales de las siguientes funciones de conmutación.

$$G(A, B, C, D, E) = \prod_M (0, 1, 2, 6, 8, 10, 12, 14, 21, 24, 25, 26, 29, 31)$$

$$H(V, W, X, Y, Z) = \sum_m (2, 4, 6, 9, 11, 13, 15, 18, 20, 25, 27) + d(22, 29, 31)$$

$$J(m, n, o, p) = \prod_M (0, 3, 5, 6, 7, 9, 10, 11, 12, 13, 14)$$

Ejercicio 3.9. Simplificar simultáneamente con mapas K e implementar con compuertas NOR las siguientes funciones.

$$F_1(a,b,c,d,e) = \prod_M(3, 6, 7, 11, 12) \cdot d(0, 13, 15)$$

$$F_2(a,b,c,d,e) = \sum_m(0, 1, 4, 5, 10) + d(2, 8, 14, 15)$$

$$F_3(a,b,c,d,e) = \prod_M(2, 7, 9, 11, 12, 13, 15) \cdot d(3, 4, 5, 10)$$

Ejercicio 3.10. Dado el diagrama de tiempo de la figura 3.25. Diseñar e implementar con compuertas básicas y universales la función de salida F.

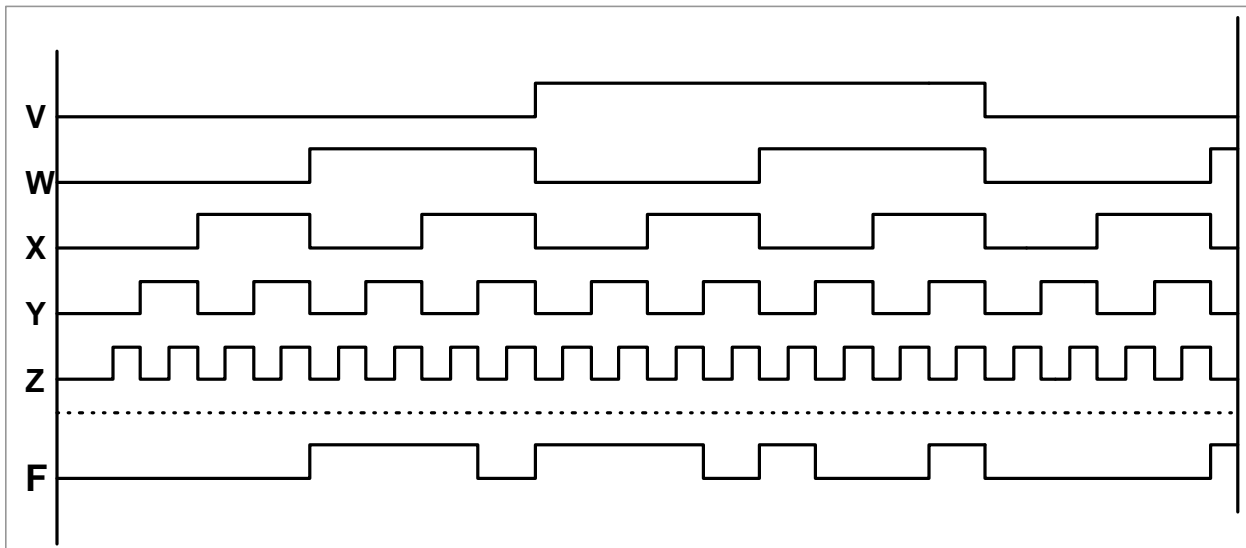


Figura 3.25. Diagrama de tiempo de la función F; para el ejercicio 3.10.

Ejercicio 3.11. Obtener e implementar una expresión algebraica mínima de una función que se coloca en nivel lógico cero cuando el número de variables en estado bajo es menor que las de nivel lógico uno. Las variables de entrada son cinco y no se pueden presentar casos donde más de tres variables se encuentren en cero. Hacer el diseño con compuertas NAND y NOR.

Ejercicio 3.12. Una máquina de contar dinero (monedas) debe ser diseñada y funcionará de la siguiente forma: Las monedas aceptadas son 10, 7, 5, 3, 1; trabaja entregando la mayor cantidad de monedas posibles; no repite monedas de un mismo tipo en cada cuenta y a lo sumo puede chequear los cinco tipos distintos de monedas. Diseñe un circuito digital que indique en la salida el tipo de moneda entregada. Realizar el diseño con mapas K y el método tabular de Quine-McCluskey.

Ejercicio 3.13. Implementar con compuertas NAND y NOR de dos niveles las siguientes funciones de conmutación:

$$f(a,b,c,d) = \prod_M (1,3,4,5,6,7,9,11,12,13)$$

$$g(a,b,c) = \sum_m (0,2,3,5,7)$$

$$h(a,b,c,d,e) = \sum_m (0,1,2,3,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,26,28,30)$$

Ejercicio 3.14. Minimizar por el método de Quine-McCluskey las siguientes funciones:

I) $F(v, w, x, y, z) = \prod_M (0, 2, 3, 6, 11, 12, 14, 15, 16, 20, 27, 28, 30, 31)$

II) La función que se obtiene en la tabla del ejercicio 3.7.

Ejercicio 3.15. Diseñar e implementar con compuertas un sumador de dos bits por dato; con acarreo de entrada. El circuito también debe tener acarreo de salida.

Ejercicio 3.16. Obtener e implementar una expresión algebraica mínima de las funciones de conmutación distribuidas en las celdas de los siguientes mapas de Karnaugh. Además de las formas con compuertas básicas, también se deben implementar con compuertas universales NAND y NOR.

		jk			
		00	01	11	10
mn	00	1			1
	01	1			
	11			1	X
	10	X		1	1
		0	4	12	8
		1	5	13	9
		3	7	15	11
		2	6	14	10

(a)

		ABC							
		000	001	011	010	110	111	101	100
DE	00	1	X					1	1
	01			X	1	1	1		
	11			1	X	X	1		
	10	X	1		1	1		X	1
		0	4	12	8	24	28	20	16
		1	5	13	9	25	29	21	17
		3	7	15	11	27	31	23	19
		2	6	14	10	26	30	22	18

(b)

		jk			
		00	01	11	10
mn	00				
	01	0	0	X	0
	11	X	0	0	X
	10			0	0
		0	4	12	8
		1	5	13	9
		3	7	15	11
		2	6	14	10

(c)

		vw x							
		000	001	011	010	110	111	101	100
yz	00	1			X	1		1	1
	01				1	1			
	11		1	X	X	X	1	1	
	10	X	1	1	1	1	X		1
		0	4	12	8	24	28	20	16
		1	5	13	9	25	29	21	17
		3	7	15	11	27	31	23	19
		2	6	14	10	26	30	22	18

(d)

CAPÍTULO 4.

4 CARACTERÍSTICAS INTERNAS DE LAS FAMILIAS LÓGICAS.

Los circuitos integrados digitales están caracterizados por la tecnología de fabricación utilizada. La base de esta integración es el silicio que junto a otros materiales, usados como aditivos, ionizan y dan características eléctricas transitorias y permanente de corriente, tensión, retardo de tiempo, etc.

Los componentes básicos de la integración son: transistores bipolares, FET, resistencias y diodos; éstos originan comportamientos de tipo analógico en el circuito integrado digital. Los niveles lógicos 0 y 1 están supeditados a rangos de corriente y voltaje que van a depender de las cargas que se conecten en esas líneas digitales y, específicamente, de la tecnología de fabricación e integración utilizada en la construcción del chip.

Las familias lógicas más utilizadas en el diseño de circuitos digitales son: TTL, CMOS y ECL. Las diferencias entre ellas determinan el tipo de aplicación en la implementación del diseño lógico digital y el rendimiento del mismo.

Existen actualmente otras subfamilias de circuitos integrados que trabajan con voltajes bajos y altas frecuencias como lo son las series LVC y LVT que trabajan con tensiones entre 2.5 y 5.0 Voltios. En este capítulo no se estudiarán estos dispositivos; no obstante, se sugiere consultar los manuales de fabricantes como Texas Instruments (2.5V-5V Standard Logic IC "SN74LV-A series 2000) o la dirección electrónica: www.ti.com/sc/logic/lva.

4.1 Parámetros eléctricos de un circuito integrado digital.

Los parámetros de las compuertas lógicas están determinados por el fabricante del circuito integrado y alguno de estos parámetros comprenden valores y rangos de corriente, voltaje, retardo de tiempo, disipación de potencia, margen de ruido, fan-out. Todos ellos determinan las condiciones de operación del circuito: consumo de corriente que suministra la fuente, temperatura de trabajo, tiempo de propagación de las señales en los acoplamientos de compuertas, ruido externo, etc. Los tipos de tecnologías (familias lógicas TTL, CMOS, ECL, etc.) diferencian estas condiciones de operación, y es aquí donde el diseñador debe tomar las precauciones necesarias a la hora de implementar un circuito digital.

4.1.1 Niveles lógicos.

Los niveles alto y bajo (H y L) de las entradas y salidas digitales tienen rangos fijos dentro de una misma familia lógica. Sin embargo, existen pequeñas variaciones entre las subfamilias de los circuitos y compuertas digitales comúnmente denominada **Series** de la familia lógica. En la figura 4.1 se muestran los rangos de voltaje entrada/salida (Input/Output) de los circuitos digitales; los valores de éstos están dados en los manuales de características técnicas del fabricante y se definen de la siguiente forma:

V_{iH} (mín): Voltaje de entrada mínimo reconocido como un nivel lógico alto (1 ó H). Las tensiones por debajo de éste valor no garantiza una tensión, V_{iH} válida.

V_{iL} (máx): Voltaje de entrada máximo reconocido como un nivel lógico bajo (0 ó L). Las tensiones por encima de éste valor no garantiza una tensión, V_{iL} válida.

V_{oH} (mín): Voltaje de salida mínimo reconocido como un nivel lógico alto (1 ó H). Las tensiones por debajo de éste valor no garantiza una tensión, V_{oH} válida.

V_{oL} (máx): Voltaje de salida máximo reconocido como un nivel lógico bajo (0 ó L). Las tensiones por encima de éste valor no garantiza una tensión, V_{oL} válida.

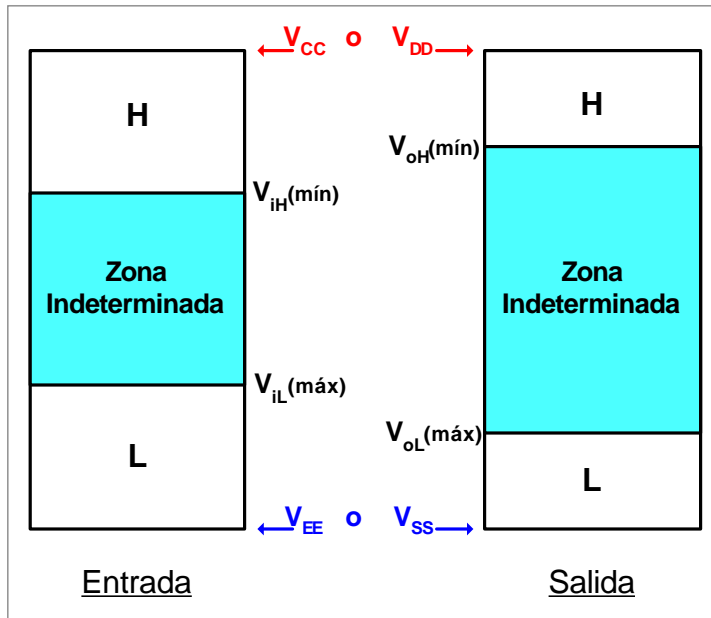


Figura 4.1. Rangos definidos para los niveles lógicos de voltaje.

Los valores correspondientes de (V_{CC} , V_{DD}) y (V_{EE} , V_{SS}) se establecen con la fuente de poder, dependiendo de la familia utilizada. Del mismo modo, las líneas de los circuitos integrados digitales drenan y conducen corrientes que dependen de la familia utilizada, y de los niveles lógicos.

$I_{iH}(\text{máx})$: Corriente de entrada máxima cuando la línea o compuerta digital está en nivel lógico alto.

$I_{iL}(\text{máx})$: Corriente de entrada máxima cuando la línea o compuerta digital está en nivel lógico bajo.

$I_{oH}(\text{máx})$: Corriente de salida máxima cuando la línea o compuerta digital está en nivel lógico alto.

$I_{oL}(\text{máx})$: Corriente de salida máxima cuando la línea o compuerta digital está en nivel lógico bajo.

Estos parámetros, dados por los fabricantes de circuitos integrados, deben ser respetados, ya que de ello dependerá el buen funcionamiento del circuito digital implementado.

De hecho, los fabricantes garantizan compatibilidad cuando se acoplan o conectan circuitos integrados de una misma Subfamilia o Serie. Por ejemplo, con $V_{CC}=+5V$ y $V_{EE}=0V$ no deben aparecer tensiones por encima del V_{CC} ni voltajes negativos por debajo del V_{EE} ; estas variaciones en la fuente de poder o en los niveles de entrada y salida ocasionan daños irreparables en los circuitos integrados.

Los voltajes de entrada/salida que se muestran en la figura 4.1 comprenden los valores que se deben aplicar en cualquier circuito digital:

$$\text{Nivel alto, uno o H: } V_{iH}(\text{mín}) \leq V_{iH} < [V_{CC} \text{ o } V_{DD}]; \quad V_{oH}(\text{mín}) \leq V_{oH} < [V_{CC} \text{ o } V_{DD}]$$

$$\text{Nivel bajo, cero o L: } [V_{EE} \text{ o } V_{SS}] \leq V_{iL} \leq V_{iL}(\text{máx}); \quad [V_{EE} \text{ o } V_{SS}] \leq V_{oL} \leq V_{oL}(\text{máx})$$

4.1.2 Conexión de salida (fan - out).

El acoplamiento directo de compuertas tiene limitaciones que determinan la cantidad de entradas que se pueden conectar a una salida. Esto es debido a que la corriente suministrada y absorbida en los distintos niveles de tensión de las compuertas. En la figura 4.2 se observa el acoplamiento de varias entradas de compuertas inversoras a una salida de compuerta NAND. Los cambios en las entradas de la NAND hacen que la salida pase de un nivel lógico a otro.

Este es un acoplamiento estático de compuertas, ya que solamente se toman en cuenta las corrientes y tensiones DC de las mismas. De esta forma será necesario saber solamente la cantidad de compuertas que se pueden conectar a la salida de un chip perteneciente a una familia o serie específica.

El **fan - out** de una familia es el número máximo de líneas de entrada que se le pueden conectar a la salida de un circuito o compuerta. En la figura 4.2 se obtiene el valor del fan-out para un nivel lógico alto y bajo en la salida **S**:

$$m \leq \frac{I_{oL}(\text{máx})}{I_{iL}(\text{máx})} \quad \text{Ec.4.1}$$

Por lo menos, el valor $S=V_{OL}(\text{máx})$ debe estar comprendido en un rango de tensión para nivel lógico bajo. De esta misma manera, un nivel lógico alto en $S=V_{OH}(\text{mín})$,

Da como resultado:

$$n \leq \frac{I_{oH}(\text{máx})}{I_{iH}(\text{máx})} \quad \text{Ec.4.2}$$

Las familias lógicas y sus respectivas series tratan de mantener la compatibilidad entre n y m de forma que sean similares o iguales.

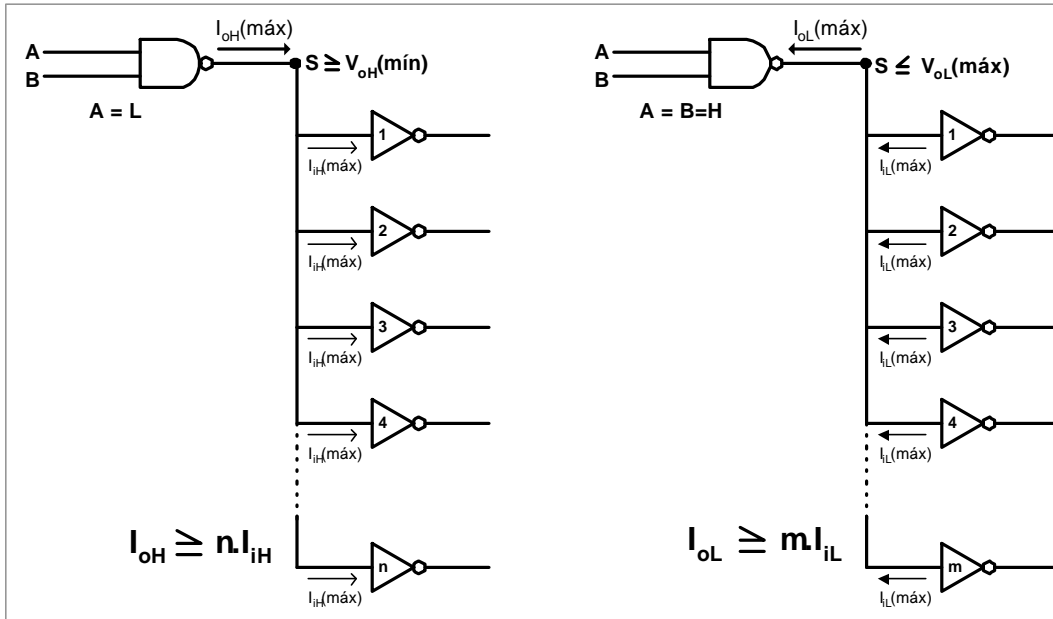


Figura 4.2. Conectividad o fan-out de las compuertas digitales.

4.1.3 Márgenes de ruido.

Los componentes y circuitos electrónicos son susceptibles a ruidos que pueden ser producidos por: variaciones de temperatura, ruido ambiental, inducción de transformadores, motores, relays, conmutadores eléctricos, etc. Los fabricantes de circuitos integrados prevén estas posibilidades de generación de ruido y por consiguiente incluyen en el diseño, una diferencia entre la entrada y la salida de las compuertas; con la finalidad de mantener la conectividad y los niveles lógicos H y L de entrada / salida de las mismas. Esta diferencia se conoce como margen de ruido y está indicada en la figura 4.3.

Margen de ruido estático en nivel alto (V_{NSH}): Es la máxima variación permitida en el nivel alto de salida, dentro de la cual queda garantizado el reconocimiento como nivel alto en la entrada del otro circuito o compuerta de la misma familia.

$$V_{NSH} = V_{oH}(\text{mín}) - V_{iH}(\text{mín}) \quad \text{Ec. 4.3}$$

Margen de ruido estático en nivel bajo (V_{NSL}): Es la máxima variación permitida en el nivel alto de salida, dentro de la cual queda garantizado el reconocimiento como nivel bajo en la entrada del otro circuito o compuerta de la misma familia.

$$V_{NSL} = V_{iL}(\text{máx}) - V_{oL}(\text{máx}) \quad \text{Ec. 4.4}$$

La tecnología utilizada por el fabricante busca siempre aumentar el margen de ruido para poder obtener más inmunidad al ruido.

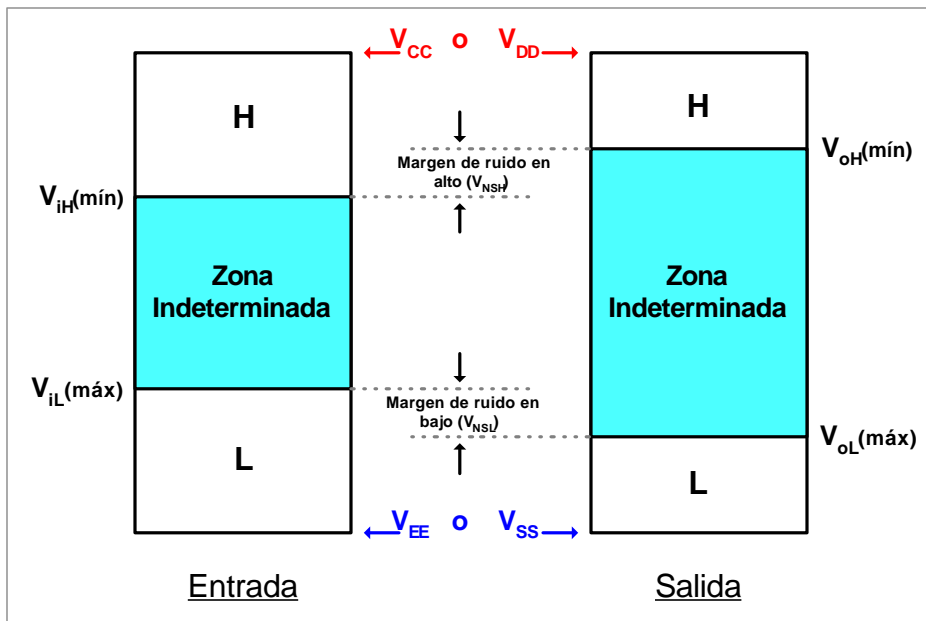


Figura 4.3. Márgenes de ruido estáticos.

4.1.4 Disipación de potencia y consumo de corriente.

Las fuentes de alimentación son las encargadas de suministrar corriente a los circuitos integrados que conforman, internamente, a las compuertas lógicas digitales; a esta corriente se le denomina I_{CC} . La potencia disipada o consumida es muy pequeña y está por el orden de los miliwatts (mW); el término utilizado para el consumo de corriente, cuando todas las compuertas se encuentran en nivel bajo, es I_{CCL} y para el nivel alto es I_{CCH} . No obstante, el consumo de corriente continua (DC) en todas las compuertas se promedia asumiendo que ellas, se encuentran el mismo tiempo en nivel alto que en nivel bajo, y por lo tanto, la corriente suministrada por la fuente debe ser:

$$I_{CC} = \frac{I_{CCL} + I_{CCH}}{2} \quad \text{Ec. 4.5}$$

En consecuencia, la disipación o consumo de potencia estática está expresada por:

$$P_D = I_{CC} \times V_{CC} \quad \text{ó} \quad P_D = I_{DD} \times V_{DD} \quad \text{Ec. 4.6}$$

Por lo general, los circuitos digitales son utilizados para conmutar de un estado a otro; en el momento que son acoplados generan transiciones, producen cambios en el consumo de corriente y en la disipación de potencia. Esta forma de consumo de energía se conoce como **disipación de potencia dinámica** " P_{DD} " y es igual a la energía almacenada en el condensador que origina la carga acoplada a la compuerta " C_L ", multiplicada por el cuadrado del voltaje; siendo proporcional al número de transiciones por segundo (frecuencia " f ").

$$P_{DD} = C_L \times V_{CC}^2 \times f \quad \text{Ec. 4.7}$$

A medida que aumenta la frecuencia también incrementa el promedio de consumo de corriente y por lo tanto más calentamiento habrá en el circuito. Al conectar compuertas aumentamos la capacitancia parásita acoplada y, como se verá más adelante, la potencia dinámica reducirá el fan-out de las compuertas.

4.2 Lógica TTL.

La lógica transistor transistor (TTL) es un tipo de tecnología bipolar que utiliza transistores para generar las distintas funciones lógicas. Esta formada por las variantes denominadas series de la familia TTL, mostradas en la tabla 4.1.

SERIE	Nomenclatura	Rango de temperatura
Estándar comercial	74xxx	[0 °C ~ 70 °C]
Estándar militar	54xxx	[-55 °C ~ +125 °C]
Bajo consumo	74Lxxx	[0 °C ~ 70 °C]
Bajo consumo militar	54Lxxx	[-55 °C ~ +125 °C]
Técnica Schottky	74Sxxx	[0 °C ~ 70 °C]
Técnica Schottky militar	54Sxxx	[-55 °C ~ +125 °C]
Bajo consumo Schottky	74LSxxx	[0 °C ~ 70 °C]
Bajo consumo Schottky militar	54LSxxx	[-55 °C ~ +125 °C]
Rápida (FAST)	74Fxxx	[0 °C ~ 70 °C]
Rápida (FAST) militar	54Fxxx	[-55 °C ~ +125 °C]
Avanzada Schottky	74ASxxx	[0 °C ~ 70 °C]
Avanzada Schottky militar	54ASxxx	[-55 °C ~ +125 °C]
Avanzada de bajo consumo Schottky	74ALSxxx	[0 °C ~ 70 °C]
Avanzada de bajo consumo militar	54ALSxxx	[-55 °C ~ +125 °C]
Alta velocidad	74Hxxx	[0 °C ~ 70 °C]
Alta velocidad militar	54Hxxx	[-55 °C ~ +125 °C]

Tabla 4.1. Series de la familia TTL.

La serie militar 54 trabaja en un rango de temperatura bastante amplio [-55 °C ~ +125 °C], es utilizada en la industria militar y equipos médicos. La serie 74 indica un rango de temperatura menor [0 °C ~ 70 °C], es la más utilizada comercialmente y tiene menor costo. En la figura 4.4 se muestra la forma de numerar los chips TTL.

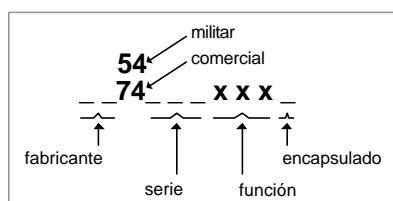


Figura 4.4. Nomenclatura de los chips TTL.

FABRICANTES	CÓDIGOS	SERIE TTL	SUB-FAMILIA	TIPO DE FUNCIÓN	DIP CERÁMICO 14 - 16 - 24	DIP PLÁSTICO 14 - 16 - 24	ENC. PLANO
Texas Instrumenst	SN	74xxx 54xx	L	C.I. SSI	- J -	- N -	-
Fairchild	F	9Nxx 93xx 96xx 74xxx	H	00 01 02	- D -	- P -	F
Motorola	MC	74xx	S	04	- L -	- P -	F
Nacional Semiconductor	DM	8000, 74xxx	AS	05	- J -	- N -	W
Ferranti	ZN	74xxx	LS	07	- J -	- E -	F
Sinetics/Philips	N	74xxx	ALS	08	- F -	A B N	W
SGS/Ates	T	74xxx		10	- - -	- B1 -	-
Philips	FJ	H101 J101 K101 L101 Q101 R101 Y101.		20	- - -	- - -	-
Siemens	FL			30	- - -	- - -	-
ITT	MIC	74xxxx		40	- J -	- N -	-
AEG/Telefunken	TL			C.I. MSI	- - -	- N -	-
Sescosem	SFC	400		42	- K -	- E -	P
Stewart Telefunken	SW	74xxx		75	- - -	- P -	-
Toshiba	PD	3400		85	- DC -	- DP -	FC
ProElectron	GFB	74xxx		91	- - -	- N -	-
Nec	###PB	74xxx		93	- D -	- D -	-

Tabla 4.2. Especificaciones de algunos fabricantes.

Estructura de la Fecha: El código de la fecha es otro código que trae el circuito integrado junto al que lo describe, indica lugar y fecha de la manufacturación. Con una o más letras especifica el país, en la parte numérica, las dos primeras cifras indican el año y las dos últimas se refieren a la semana de fabricación.

Lugar país donde fue manufacturado	Año de fabricación	Semana de fabricación
------------------------------------	--------------------	-----------------------

Por ejemplo, el chip [SN74LS00J 9532] indica que se trata de una compuerta NAND de dos entradas, serie de bajo consumo Schottky con rango de temperatura desde 0 °C hasta 70 °C y fue fabricado por Texas Instruments el año 1995 semana 32.

4.2.1 Series Estándar, S, F, H, AS, ALS y LS.

En la figura 4.5(a) y 4.5(b) se pueden observar las compuertas NAND de las series estándar y de bajo consumo Schottky "LS".

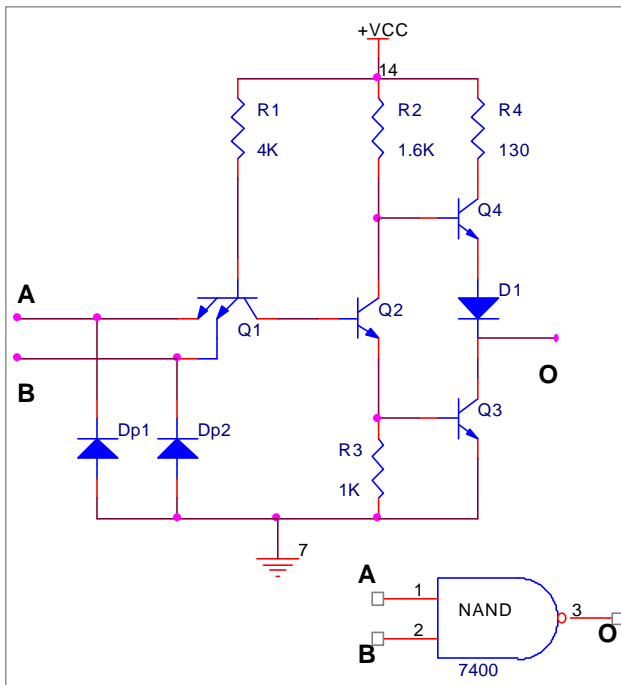


Figura 4.5(a). NAND TTL estándar.

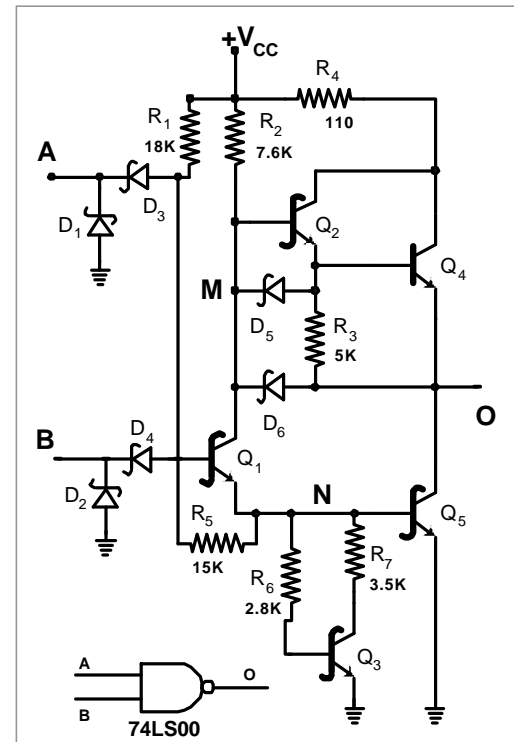


Figura 4.5(b). NAND Schottky de bajo Consumo "LS".

La primera serie creada, figura 4.5a, fue la estándar, sus aplicaciones se ven limitadas por el considerable consumo de corriente de los transistores BJT Q_3 y Q_4 ; y la posible sobresaturación de los transistores. Esto también aumenta la corriente y la temperatura del circuito integrado, produciendo ruido y retardos de tiempo en la señal digital.

El transistor de múltiple emisor Q_1 conduce cuando una, o las dos entradas, **A** y **B** tienen un nivel de tensión menor que la suma de: $(V_{be Q_3} + V_{be Q_2}) \leq 1.2V$; de este modo queda polarizada inversamente la unión base-colector, por lo cual, el transistor Q_2 no conduce y por ende Q_3 . Por otra parte, el transistor Q_4 queda polarizado directamente en la unión base-emisor y la salida **O** queda con un nivel de tensión aproximadamente igual que $+V_{CC}$.

Cuando ambas entradas superan la tensión: $(V_{be} Q_3 + V_{be} Q_2 + V_{be} Q_1) \geq 1.8V$ el transistor Q_1 queda polarizado inversamente en la unión base-emisor, pero, la unión base-colector se polariza directamente; esto hace que Q_2 y Q_3 se activen y coloquen una tensión de nivel bajo en la salida "O". De esta misma forma, trabaja la serie **LS**; sin embargo, las entradas poseen diodos Schottky de baja tensión de polarización ($< 0.3V$) para evitar la saturación y aumentar el margen de ruido de las compuertas.

Los transistores de éste tipo no permiten que la unión base-colector se sature, logrando que los portadores mayoritarios de las uniones N-P-N se desplacen, o se coloquen en reposo, con mayor rapidez cuando ocurran las conmutaciones on-off del transistor Schottky. En la figura 4.5(b) se puede observar, la ventaja de utilizar éstos diodos y transistores; los valores de las resistencias son superiores a los de la serie estándar y por lo tanto es menor: el consumo de corriente, la disipación de potencia, más inmunidad al ruido y menor retardo de tiempo.

También existen otras series que entregan un producto consumo-velocidad más eficiente como lo son: la serie "High Speed" (H); "Avanzada Schottky" (AS, ALS); "FAST" (F). En las figuras 4.7(a, b, c, d) se muestran algunos tipos de compuertas y sus respectivas series.

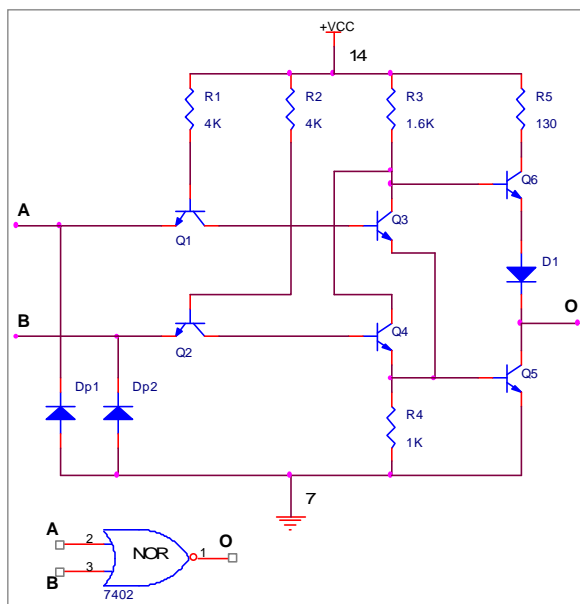


Figura 4.6(a). NOR estándar.

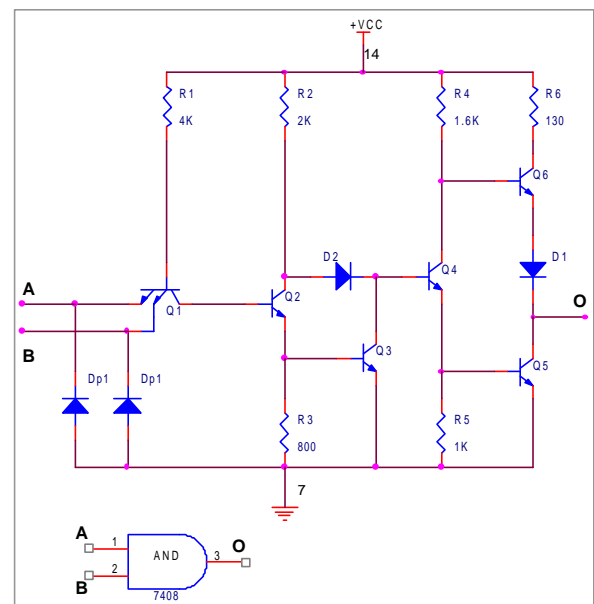


Figura 4.6(b). AND estándar.

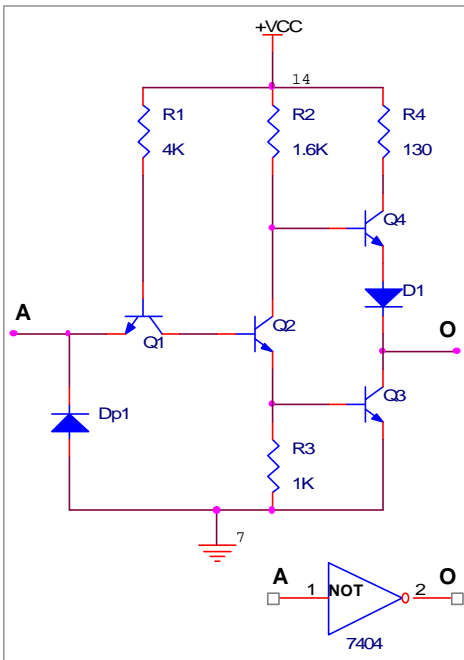


Figura 4.7(a). NOT estándar.

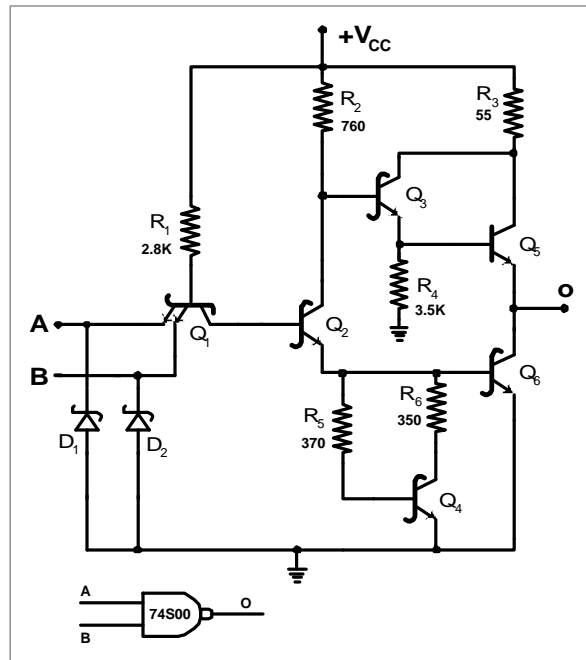


Figura 4.7(b). NAND Schottky "S".

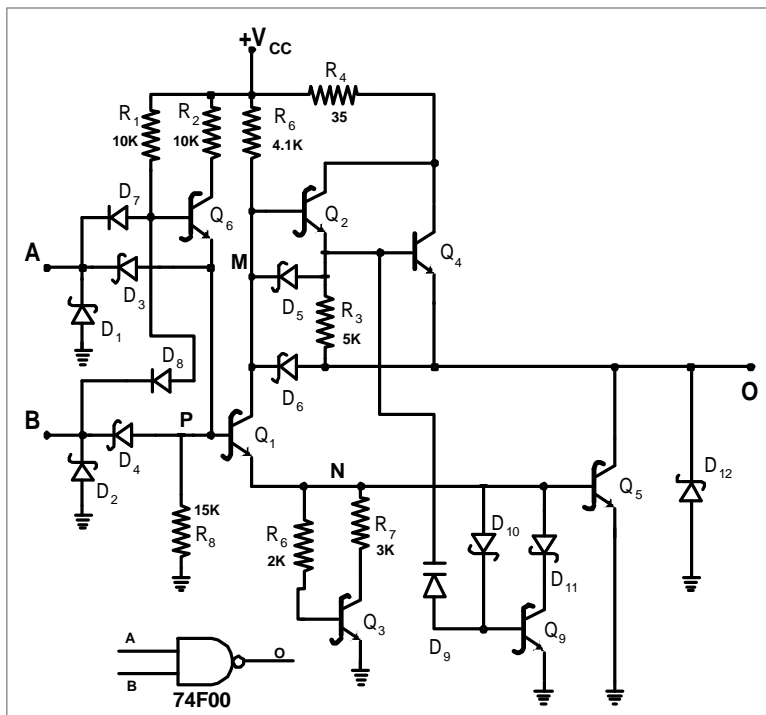


Figura 4.7(c). NAND FAST Schottky "F".

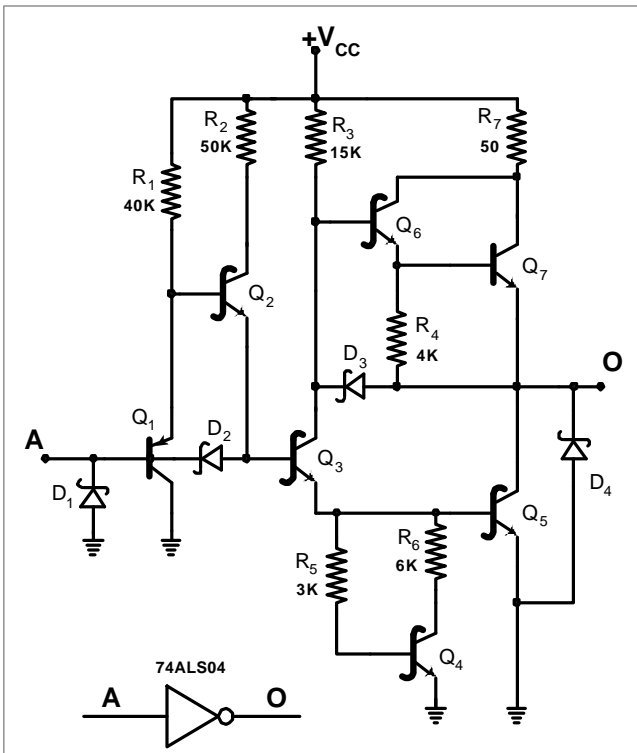


Figura 4.7(d). NOT Avanzada Schottky de baja potencia "ALS".

4.2.2 Conectividad, Margen de ruido, consumo de corriente, retardo de tiempo de las series TTL.

La familia TTL posee compatibilidad de corriente, tensión y retardo de tiempo entre las series que la componen. No obstante, es recomendable utilizar circuitos integrados pertenecientes a una misma serie para que sus características técnicas sean exactamente iguales y por ende, se disminuyan los errores de propagación de señales en los acoplamientos de los dispositivos. Por ejemplo, al acoplar en paralelo dos compuertas de distinta serie, la compuerta más rápida colocara primero la señal en la salida, ocasionando que el circuito alimentado por ésta responda a mayor frecuencia. En la tabla 4.3 se muestran las características más comunes de las compuertas pertenecientes a la familia TTL.

Caract. / Serie	Fast (74F)	(74LS)	(74AS)	(74L)	(74H)	(74ALS)	74
Tiempo de propagación (t_p)	4 ns	9 ns	1.6 ns	33 ns	6 ns	5 ns	10 ns
Consumo de potencia por compuerta (P)	15 mW	4 mW	20 mW	1 mW	22 mW	1.3 mW	10 mW
V_{iL} (máx)	0.8 V	0.8 V	0.8 V	0.7 V	0.8 V	0.8 V	0.8 V
V_{oL} (máx)	0.5 V	0.5 V	0.5 V	0.4 V	0.4 V	0.5 V	0.4 V
V_{iH} (mín)	2.0 V	2.0 V	2.0 V	2.0 V	2.0 V	2.0 V	2.0 V
V_{oH} (mín)	2.7 V	2.7 V	2.7 V	2.4 V	2.4 V	2.7 V	2.4 V
I_{iL} (máx)	-600 μ A	-400 μ A	-2 mA	-180 μ A	-2 mA	-200 μ A	-1.6 mA
I_{oL} (máx)	20 mA	8 mA	20 mA	3.6 mA	20 mA	8 mA	16 mA
I_{iH} (máx)	20 μ A	20 μ A	200 μ A	10 μ A	50 μ A	20 μ A	40 μ A
I_{oH} (máx)	-1000 μ A	-400 μ A	-2 mA	-200 μ A	-500 μ A	-400 μ A	-400 μ A
Fan-out	33	20	10	20	10	40	10
I_{oH} (máx) colector abierto	-----	100 μ A	-----	50 μ A	250 μ A	-----	250 μ A
Corriente de salida en corto circuito (I_{os})	-150 mA	-100 mA	-150 mA	-15 mA	-100 mA	-100 mA	-55 mA

Tabla 4.3. Algunas características técnicas promediadas de la familia TTL.

Conectividad (Fan-out): La tabla 4.3 indica el número de entradas de compuertas que se pueden conectar a una línea de salida, las que tienen mayor conectividad son las series: $FAST \leq 33$; $LS \leq 20$; $L \leq 20$ y $ALS \leq 40$. Sin embargo, esta cantidad debe ser reducida para asegurar que la corriente de salida no supere el 80% de I_{oL} e I_{oH} (máx) y de esta manera garantizar el buen funcionamiento del circuito integrado.

Margen de ruido: En la sección 4.1.3, figura 4.3, se definen los parámetros de márgenes de ruido V_{NSL} y V_{NSH} ; este último, determina la diferencia entre las tensiones mínimas del nivel lógico alto V_{oH} (mín) y por tanto, muy fundamental para poder determinar la inmunidad al ruido. El V_{NSH} de las series FAST, LS, AS, y ALS es igual a 0.7 V lo que implica una mayor inmunidad al ruido que las series L, H y estándar donde el margen de ruido V_{NSH} es de 0.4 V.

Consumo de corriente: Las series que manejan mayor corriente son las **FAST**, **AS** y **H**. Están diseñadas para este propósito, la corriente que soportan en nivel bajo I_{oL} es menor o igual que 20 mA; la diferencia con respecto a las series de tecnología **L**, **LS** y **ALS** es, efectivamente, la baja corriente I_{oL} (3.6mA, 8mA, 8mA) que circula a través de ellas. En este aspecto los chips de mayor consumo de corriente son más rápidos, pero con el inconveniente de generar mayor calor en el circuito integrado y ruido de picos de corriente en la fuente de alimentación. La ventaja de la serie **FAST** es que puede soportar cargas mayores a las otras series TTL y de este modo, mejorar el *fan-out*.

Retardo de tiempo (t_p): La característica de retardo de tiempo esta intrínsecamente ligado a los materiales semiconductores con que fabrican los circuitos integrados. Una capa delgada de material **N** o **P** hace que los portadores minoritarios necesiten menor tiempo para conmutar de un estado de encendido hacia la condición de apagado. Los tiempos de retardo que ocasionan *los períodos de almacenamiento y transición* de la unión **NP** o **PN** determinan la respuesta transitoria de las compuertas TTL. Esto se conoce como tiempo de propagación (t_p) o retardo de tiempo y es una característica muy importante que el diseñador debe tomar en cuenta a la hora de realizar el diseño digital.

En la representación, aproximada, de la onda cuadrada con niveles TTL de la figura 4.8 se puede observar, la respuesta **S** de un inversor 74LS04. Si en la entrada **E** se inyecta un pulso de esta onda; la señal de salida se invierte y se propaga en el tiempo. El instante t_1 y t_3 , son tomados respectivamente del 10% y 90% de la rampa de subida; esta diferencia de tiempo $t_3-t_1=tr$ es conocida como tiempo de subida (tr : time rising), flanco de subida o transición positiva (**TSP**). De igual forma, la diferencia $t_7-t_5=tf$ se conoce como tiempo de bajada (tf : time falling), flanco de bajada o transición negativa (**TSN**).

El tiempo de propagación de la señal de entrada, a la mitad de la rampa (50%), con respecto a la salida; cuando ésta cambia del nivel alto al nivel bajo, se conoce como t_{pHL} . La figura 4.8 indica el t_{pHL} como la diferencia de tiempo t_4-t_2 . Del mismo modo t_8-t_6 es t_{pLH} y ocurre cuando la señal de salida pasa de un nivel bajo a un nivel alto. El tiempo de propagación se obtiene sacando el promedio de estos dos valores:

$$t_p = \frac{t_{p_{HL}} + t_{p_{LH}}}{2} \quad \text{Ec. 4.8}$$

La frecuencia máxima de trabajo o frecuencia de corte de la compuerta es el inverso de t_p :

$$f(\text{máx}) = \frac{1}{t_p} \quad \text{Ec. 4.9}$$

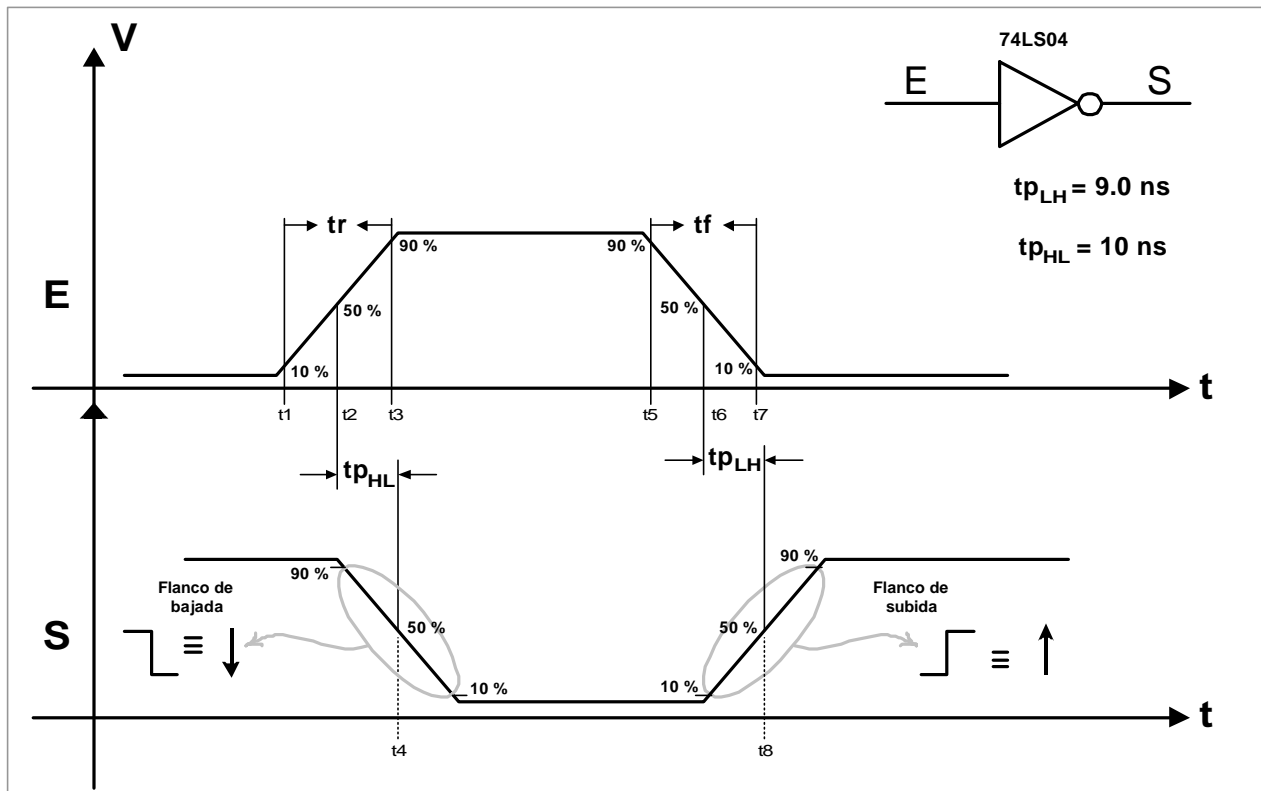


Figura 4.8. Propagación de tiempo de la compuerta inversora 74LS04.

Los tiempos de propagación de las series: AS, F, H y ALS están por debajo de los **10 ns** lo que permite colocarlas como las más rápidas de la familia TTL. La frecuencia máxima o de corte de la serie AS, según Ec.4.9, es igual a: $f_{\text{máx}} = 1/1.6\text{ns} = 650\text{MHz}$; le siguen, en rendimiento de velocidad, las series: **F**(250 MHz), **ALS**(200 MHz), **H**(167 MHz), **LS**(111 MHz), **Estándar**(100 MHz). No obstante, la serie de bajo consumo **L** posee el mayor tiempo de propagación y por lo tanto la más lenta de la familia TTL; con una frecuencia de corte por debajo de $1/33\text{ns}$.

En la figura 4.8 se observa la representación que utilizan algunos manuales técnicos de compuertas para indicar una transición positiva (t_r , flanco de subida) con una flecha ascendente y transición negativa (t_f , flanco de bajada) con una flecha descendente.

Una característica importante de los circuitos integrados de compuertas digitales es el factor formado por producto del tiempo de propagación y el consumo promedio de potencia. Este factor debe ser lo más pequeño posible; los fabricantes de circuitos integrados, a través de las tecnologías, buscan constantemente la forma de disminuirlo.

$$Factor_{s,p} = tp \times P \quad \text{Ec. 4.10.}$$

De la tabla 4.3 se obtiene el producto para estas series:

- High Speed **H** (132 ns.mW).
- Estándar (100 ns.mW).
- Fast **F** (60 ns.mW).
- Low power Schottky **LS** (36 ns.mW).
- Low power: bajo consumo **L** (33 ns.mW).
- Avanzada Schottky **AS** (32 ns.mW).
- Avanzada de bajo consumo Schottky **ALS** (6.5 ns.mW).

La serie que tiene mejor factor es la **ALS** (6.5) y el factor más pobre es el de la serie **H** (132). Al mejorar la velocidad de respuesta de un circuito integrado se debe sacrificar, por otra parte, el consumo de potencia y viceversa. Las nuevas tecnologías de fabricación buscan la forma de aumentar la velocidad de los dispositivos y al mismo tiempo disminuir el consumo de potencia con el fin de mejorar el **Factor**_{s,p}. Una de las alternativas que se han aplicado es la de disminuir la tensión de alimentación de los circuitos, con la finalidad de poder utilizar capas más delgadas de silicio y reducir el tiempo de almacenamiento de los portadores minoritarios en las uniones de los semiconductores. De esta forma, se pueden ver en el mercado circuitos integrados digitales y analógicos con tensiones de alimentación menores a 5 voltios.

4.2.3 Compuertas TTL de colector abierto.

Las compuertas de colector abierto no poseen internamente el transistor Q_4 , ver figura 4.5(a), que conduce cuando la salida del dispositivo es alta. La salida de la compuerta queda trabajando solamente con el transistor Q_3 (ver figura 4.9), éste conduce a través de la resistencia externa R_e , que debe ser calculada, para que la salida de la compuerta pueda conmutar los niveles lógicos. Las ventajas de las compuertas de colector abierto son las siguientes:

- Puede soportar cargas con voltaje superior al +VCC de 5 voltios; la tensión de la resistencia externa llega, en algunos chips, hasta 30 voltios.
- Permite manejar más corriente que una compuerta TTL Estándar Totem Pole, 40 mA en algunas compuertas.
- Se pueden conectar varios diodos Leds directamente a la salida de la compuerta, bombillos de baja corriente y pequeños relés.

Sin embargo, una desventaja de las compuertas de colector abierto es que la conexión de la resistencia externa puede ocasionar retardos de propagación en la respuesta de alta frecuencia del dispositivo. La figura 4.9 muestra la parte interna de una compuerta inversora de colector abierto 7406; algunas de las características más importantes del circuito integrado se describen en la tabla 4.4.

Símbolo	Unidades
V_{CC}	5 V
$V_{iH}(\text{mín})$	2.0 V
$V_{iL}(\text{máx})$	0.8 V
V_{OH}	(mín = 2.4 V); (máx = 30 V)
$V_{OL}(\text{máx})$	0.7 V
$I_{iH}(\text{máx})$	40 μ A
$I_{iL}(\text{máx})$	-1.6 mA
$I_{oH}(\text{máx})$	250 μ A
$I_{oL}(\text{máx})$	40 mA
t_p	12.5 ns
P_D	26 mW por compuerta.

Tabla 4.4. Características del chip 7406.

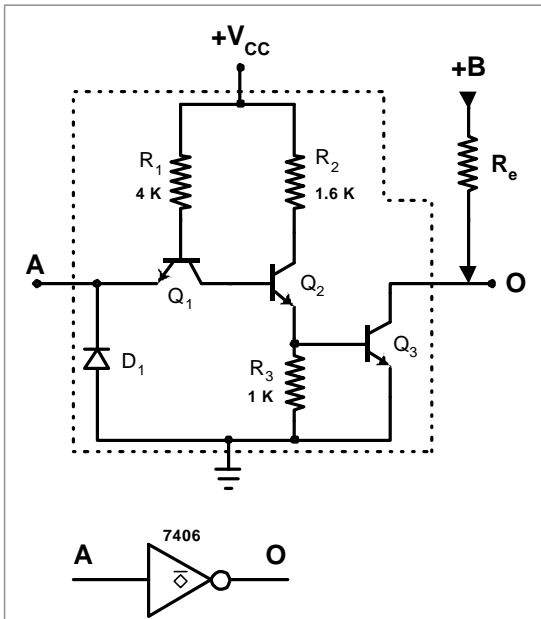


Figura 4.9. Esquema interno de la compuerta 7406.

El colector del transistor bipolar Q_3 debe ser polarizado por medio de la resistencia externa R_e . La tensión máxima que soporta Q_3 es de 30 Voltios con una corriente máxima I_{OL} de 40 mA. Estos parámetros también se aplican en los casos donde R_e es una carga resistiva – inductiva. El valor de R_e está comprendido entre un rango que va desde un valor de resistencia externa mínima; $R_e(\text{mín})$, calculado con V_{OL} e I_{OL} , hasta el valor de resistencia externa máxima; $R_e(\text{máx})$, que se obtiene con V_{OH} e I_{OH} .

$$V_{Re} = +B - V_{OL}(\text{máx})$$

$$I_{Re} = I_{OL}(\text{máx})$$

$$R_e(\text{mín}) = \frac{+B - V_{OL}(\text{máx})}{I_{OL}(\text{máx})} \quad \text{Ec. 4.11}$$

Donde $+B$ es la tensión de alimentación, ésta puede ser diferente a los 5 Volt necesarios en los circuitos integrados de la familia TTL estándar.

$$V_{Re} = +B - V_{OH}(\text{mín})$$

$$I_{Re} = I_{OH}(\text{máx})$$

$$R_e(\text{máx}) = \frac{+B - V_{OH}(\text{mín})}{I_{OH}(\text{máx})} \quad \text{Ec. 4.12}$$

Si la tensión $+B$ es de 12 Voltios, la resistencia externa debe tener un rango entre:

$$R_e(\text{mín}) = \frac{12V - 0.7V}{40mA} = 282.5\Omega$$

$$R_e(\text{máx}) = \frac{12V - 2.4V}{40mA} = 240K\Omega$$

$$282.5\Omega \leq R_e \leq 240K\Omega$$

Este análisis se hace sin considerar cargas acopladas a la salida de la compuerta, y en condiciones de corriente continua. La figura 4.10 presenta otro caso, donde se deben encender dos diodos leds conectados en paralelo y en la figura 4.11, se conectan cuatro diodos en serie.

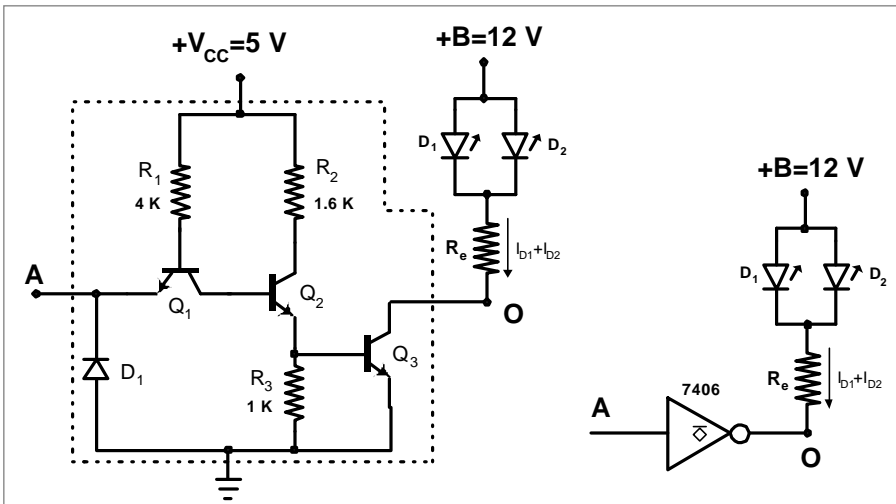


Figura 4.10. Leds de carga paralela a una compuerta de colector abierto 7406.

La tensión del diodo led es 1.8 Voltios ($V_D=1.8V$) y la corriente de diodo es 15 mA ($I_D=15mA$). Para $V_O = V_{OL}(\text{máx})$ se tiene un valor de resistencia externa mínima:

$$I_{Re} = I_{D1} + I_{D2} = 15mA + 15mA = 30mA$$

$$V_{Re} = +B - [V_D + V_{OL}(\text{máx})] = 12V - [1.8V + 0.7V] = 9.5V$$

$$R_e = \frac{V_{Re}}{I_{Re}} = \frac{9.5V}{30mA} = 317\Omega \cong 320\Omega$$

En la condición de nivel lógico alto de salida, se debe asumir un V_{OH} igual a la tensión de la fuente +B. Con ésto se garantiza que los leds no prenden debido a que la corriente es despreciable y, por el transistor interno Q_3 solo circulará la corriente de fuga

que establece un parámetro de cuarenta microAmpers; $I_{OH} = 40 \mu A$. Por lo tanto, la resistencia externa que enciende los leds conectados en paralelo es de 320 ohmios.

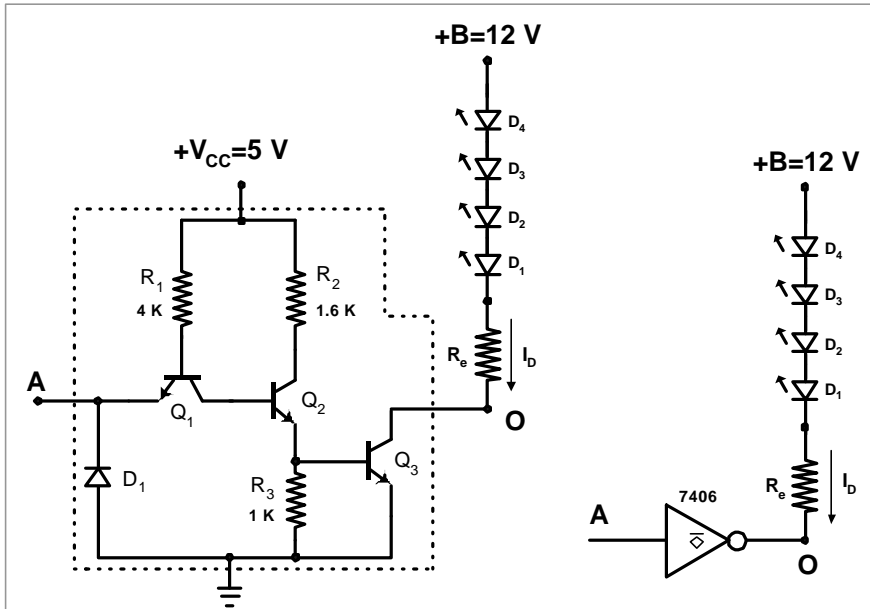


Figura 4.11. Leds de carga serie a una compuerta de colector abierto 7406.

En la figura 4.11 la resistencia externa de colector abierto se obtiene con el valor de $V_O = V_{OL}(\text{máx})$; a través de R_e circula la corriente I_D y la tensión en los extremos de ésta, es la diferencia entre el $+B$ y la suma de $[V_{D1} + V_{D2} + V_{D3} + V_{D4} + V_{OL}(\text{máx})]$.

$$I_{Re} = I_D = 15 \text{ mA}$$

$$V_{Re} = +B - [V_{D1} + V_{D2} + V_{D3} + V_{D4} + V_{OL}(\text{máx})] = 12V - [1.8V + 1.8V + 1.8V + 1.8V + 0.7V]$$

$$V_{Re} = 12V - 7.9V = 4.1V$$

$$R_e = \frac{V_{Re}}{I_{Re}} = \frac{4.1V}{15 \text{ mA}} = 273.3 \Omega$$

$$R_e = 274 \Omega$$

Ejercicio 4.1. Calcular el valor de R_e , R_z , R_B y R_D del circuito de la figura 4.12, para que pueda funcionar correctamente en los dos niveles lógicos. Las señales H_1 y Sw_1 determinan el valor de la salida en **M**; el circuito debe encender los Leds y el Buzzer cuando la tensión del punto **M** sea alta. Indicar cual debe ser la tensión Zener V_z .

El circuito integrado NAND es el 7426 de colector abierto con las siguientes características:

Símbolo	Unidades
V_{CC}	5 V
$V_{iH}(\text{mín})$	2.0 V
$V_{iL}(\text{máx})$	0.8 V
$V_{oH}(\text{máx})$	15 V
$V_{oL}(\text{máx})$	0.7 V
$I_{iH}(\text{máx})$	40 μA
$I_{iL}(\text{máx})$	-1.6 mA
$I_{oH}(\text{máx})$	250 μA
$I_{oL}(\text{máx})$	40 mA

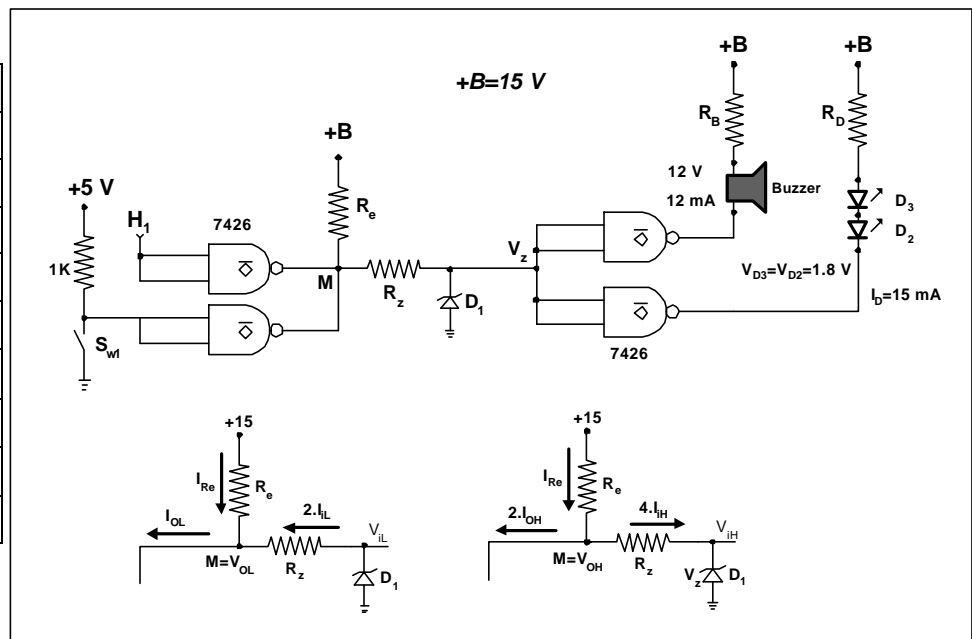


Figura 4.12. Circuito del ejercicio 4.1; Compuertas de colector abierto.

Solución:

Considerando la salida en el punto M con nivel bajo $V_{OL}(\text{máx})$:

$$V_{Re} = +B - V_{OL}(\text{máx}) = 15V - 0.4V = 14.6V$$

$$I_{Re} = I_{OL}(\text{máx}) - 2 \cdot I_{iL}(\text{máx}) = 16 \text{ mA} - 3.2 \text{ mA} = 12.8 \text{ mA}$$

$$R_e(\text{mín}) = \frac{V_{Re}}{I_{Re}} = \frac{14.6 \text{ V}}{12.8 \text{ mA}} = 1.14 \text{ K}\Omega$$

$$V_{Rz} = V_{OL}(\text{máx}) - V_{iL}(\text{máx}) = 0.4V - 0.8V = -0.4V$$

$$I_{Rz} = 2 \cdot I_{iL}(\text{máx}) = 3.2 \text{ mA}$$

$$R_z = \frac{V_{Rz}}{I_{Rz}} = \frac{0.4V}{3.2 \text{ mA}} = 125 \Omega$$

Considerando la salida en el punto M con nivel alto $V_{OH}(\text{máx})$, la tensión zener debe ser menor o igual a 5 Voltios para proteger las entradas de las compuertas ($V_z=5\text{ V}$) y la corriente inversa del mismo debe ser despreciada:

$$I_{Re} = 2 \cdot I_{OH}(\text{máx}) + 4 \cdot I_{iH}(\text{máx}) = 2 \cdot 1000 \text{ mA} + 4 \cdot 40 \text{ mA} = 2160 \text{ mA}$$

$$V_{Re} = +B - [V_{Rz} + V_z] = 15V - [V_{Rz} + 5V]$$

$$V_{Re} = 10V - V_z = 10V - 4 \cdot I_{iH}(\text{máx}) \cdot R_z$$

$$I_{Re} \cdot R_e = 10V - 160 \text{ mA} \cdot R_z$$

$$2160 \text{ mA} \cdot R_e(\text{mín}) = 10V - 160 \text{ mA} \cdot R_z$$

$$R_z = \frac{10V - 2160 \text{ mA} \cdot 1140 \Omega}{160 \text{ mA}} = \frac{10V - 2.5V}{160 \text{ mA}} = 46.9 \text{ K}\Omega$$

$$R_z(\text{máx}) = 47 \text{ K}\Omega$$

Para calcular la resistencia externa R_e máxima, se debe garantizar que la tensión en el punto M no sea inferior al $V_{OH}(\text{mín})=2.4\text{ V}$:

$$V_{Re} = +B - V_{OH}(\text{mín}) = 15V - 2.4V = 12.6V$$

$$R_e(\text{máx}) = \frac{V_{Re}}{I_{Re}} = \frac{12.6V}{2160 \text{ mA}} = 5.8 \text{ K}\Omega$$

Los valores resistivos de R_e y R_z comprenden el siguiente rango:

$$1140 \Omega \leq R_e \leq 5.8 \text{ K}\Omega$$

$$125 \Omega \leq R_z \leq 47 \text{ K}\Omega$$

En las resistencias R_e y R_z se pueden considerar valores cercanos al mínimo que garantizan los niveles de corriente del circuito integrado, y en particular, R_z debe ser mínima para que mantenga el $V_{iL}(\text{máx})$ en nivel bajo conjuntamente con la corriente $I_{iL}(\text{máx})$. Los valores recomendados son:

$$R_e = 1.2 \text{ K}\Omega \text{ y } R_z = 150 \Omega$$

Los valores de R_D y R_B se calculan solamente para la condición de nivel bajo,

$$R_B = \frac{+B - [V_{Buz} + V_{OL}(\text{máx})]}{I_{Buz}} = \frac{15V - [12V + 0.4V]}{12 \text{ mA}} = \frac{2.6V}{12 \text{ mA}} = 217 \Omega \cong 220 \Omega$$

$$R_D = \frac{+B - [V_{D2} + V_{D3} + V_{OL}(\text{máx})]}{I_D} = \frac{15V - [1.8V + 1.8V + 0.4V]}{15 \text{ mA}} = \frac{11V}{15 \text{ mA}} = 733 \Omega \cong 750 \Omega$$

4.2.4 Buffer y dispositivos de tres estados.

Los circuitos integrados, en algunos casos, deben compartir un mismo punto o línea de conducción común (Bus común), y sus salidas pueden tener distintos niveles lógicos en ese punto común de conexión. De antemano, esta conexión no se puede hacer debido al cortocircuito que se produciría cuando una salida se encuentre en alto y la otra en bajo. La figura 4.13 muestra los transistores internos Q_3 y Q_4 de dos compuertas TTL estándar G_1 y G_2 de salida **Totem Pole** conectadas a una misma línea o Bus común. Si las dos salidas son llevadas a un mismo nivel lógico alto, o bajo los transistores internos no presentarán ningún inconveniente en el funcionamiento. Sin embargo, cuando las salidas de las compuertas tienen niveles lógicos diferentes; por ejemplo, G_1 en alto y G_2 en bajo, se presenta un cortocircuito para la compuerta G_1 y se produce una sobrecorriente en R_{11} , D_{11} y Q_{41} que es igual a la $I_{OL}(\text{máx})$ de G_2 , la línea común de las dos compuertas es forzada a cero, superando el parámetro $I_{OH}(\text{máx})$ de G_1 y por lo tanto, la compuerta G_1 corre el riesgo de dañarse.

La solución a éste problema la tienen las compuertas y buffer de tres estados que poseen un tercer estado de alta impedancia (Hi-z) que desconecta internamente las salidas de los dispositivos que no están utilizando el Bus común.

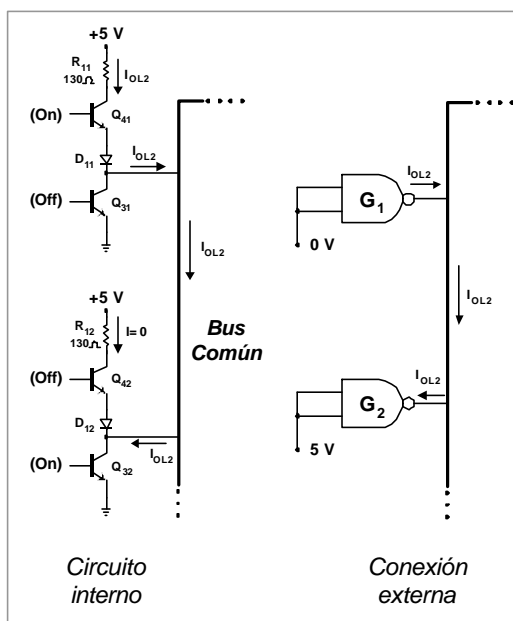


Figura 4.13. Cortocircuito que se produce al unir dos salidas Totem Pole.

De esta forma, los dispositivos desconectados deben estar en estado de alta impedancia y él que necesita colocar cero o uno en el Bus, debe permanecer conectado por un tiempo determinado. La figura 4.14 muestra una compuerta inversora de tres estados; cuando la señal de la línea de entrada **E** (Enable) es baja, la entrada **A** es indiferente y Q_1 queda polarizado directamente, debido a esto, el transistor Q_2 y Q_3 se ponen en corte. Con **E** en bajo el diodo D_2 se polariza directamente y la tensión en el punto **W** es igual a 0.7 Voltios. Por otra parte, Q_4 y D_3 necesitan una tensión superior a 1.4 Voltios para polarizarlos directamente, por lo tanto, quedan en la región de corte y la salida **O** pasa a estado de alta impedancia.

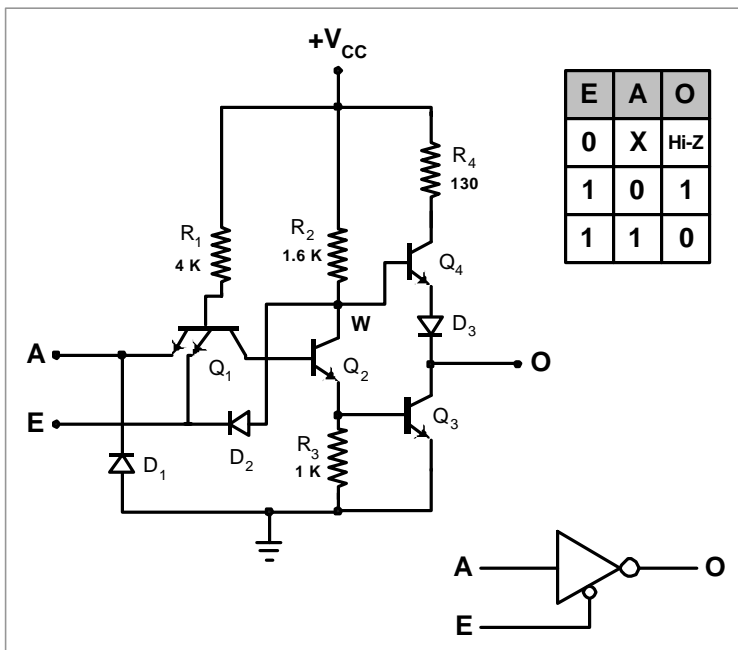


Figura 4.14. Características internas de un inversor de tres estados.

4.2.4.1 Buffers de tres estados 74125 y 74126.

Los buffers son acopladores que permiten adaptar niveles de corriente, voltaje o impedancia entre circuitos y dispositivos. Estos pueden ser inversores o no inversores. La familia TTL posee los chips de tres estados no inversores 74125 y 74126 que tienen cuatro buffers internamente cada uno y línea de habilitación independiente en bajo y

alto respectivamente. La figura 4.15 muestra la configuración de estos circuitos integrados y sus respectivas tablas de funcionamiento.

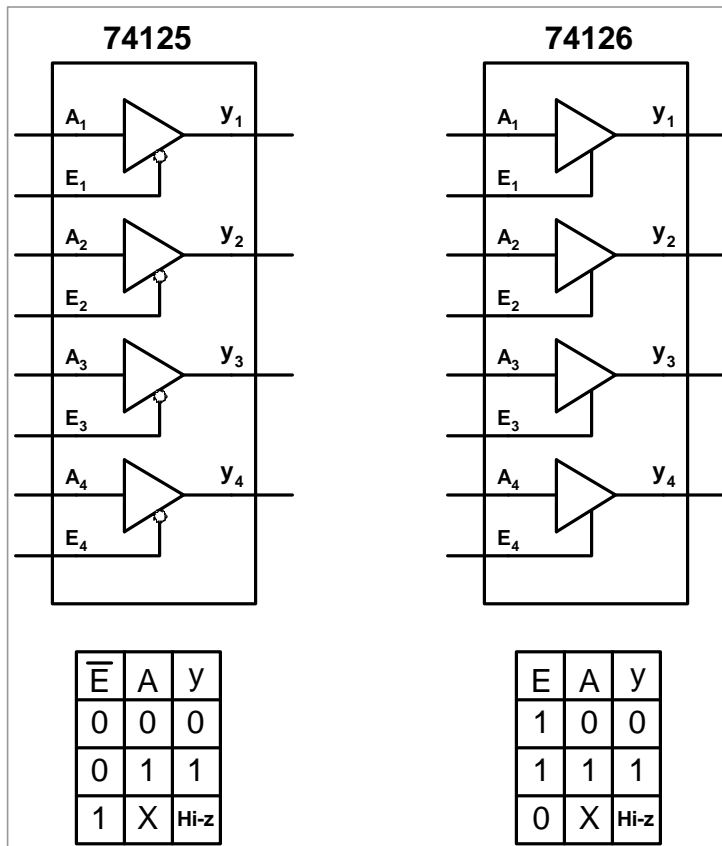


Figura 4.15. Buffers tri-state 74125 y 74126.

La aplicación mostrada en la figura 4.16 indica la forma de acoplar las líneas de datos del circuito integrado de memoria RAM (Memoria de Acceso Aleatorio) a un Bus externo común de cuatro líneas bidireccional; utilizando dos buffer de tres estados 74126. El chip de memoria realiza una operación de lectura cuando R/\bar{W} es uno lógico; el buffer IC₃ se conecta al Bus externo y la compuerta inversora 7404 hace que IC₂ quede en tercer estado y por lo tanto, desconectado del mismo Bus. El dato binario de IC₁ es colocado en ese Bus externo común. Por otra parte, la operación de escritura (Grabar) en el IC₁ se realiza cuando R/\bar{W} es cero lógico. Lo que está en el Bus externo común pasa hacia el Bus de datos del chip de memoria RAM (D₃, D₂, D₁, D₀) y se guarda en él. Este proceso de guardar o grabar el dato en la memoria RAM se conoce como

escritura. Al mismo tiempo el buffer IC_2 queda deshabilitado, y en tercer estado, por la acción de la compuerta inversora 7404.

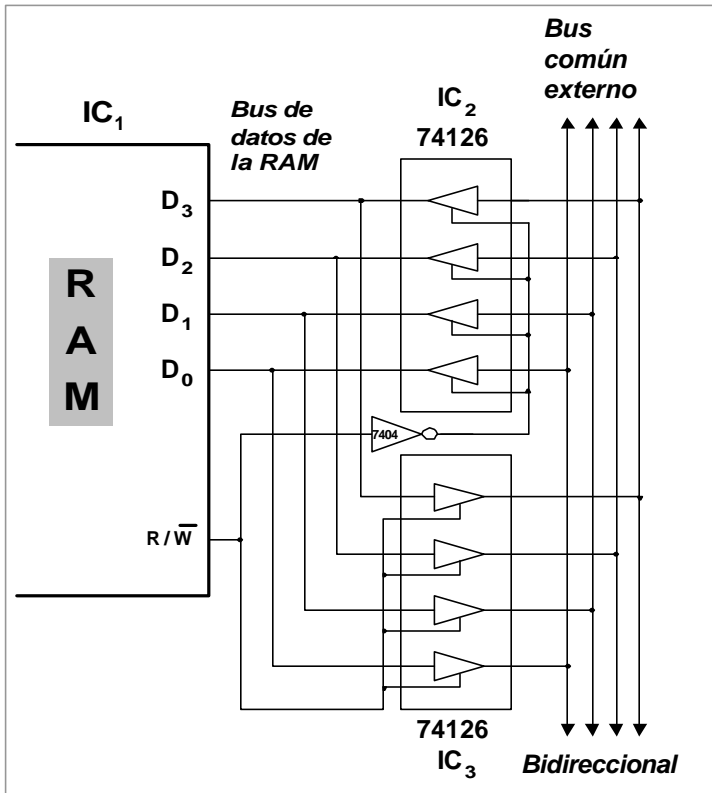


Figura 4.16. Conexión de datos en memoria RAM con buffer tres estados 74126.

Los circuitos integrados de nueva generación que comparten información común “Buses” traen incorporadas líneas de tres estados que se desconectan de éste cuando es deshabilitado el chip. Por ejemplo, si se conectan varios circuitos integrados en una tarjeta electrónica que compartan un mismo Bus de datos; las entradas de habilitación **Cs** (Chip Select) de los circuitos integrados deben ser administradas por un bloque de control que ponga en tercer estado a los chips que hacen operaciones de lectura y escritura a manera de evitar conflictos en el Bus. La línea de entrada **Cs**, por lo general, es activa en bajo, y cuando es llevada a uno lógico, coloca al circuito integrado en alta impedancia e internamente lo desconecta del resto de los dispositivos del circuito digital.

4.2.5 Circuitos digitales Smith Trigger.

Las señales digitales están sometidas a efectos transitorios en las líneas y conexiones de los circuitos impresos. Las ondas se deforman a medida que aumenta la distancia del cableado, o medio de comunicación, entre el transmisor y el receptor de la señal binaria. La figura 4.17 representa los cambios que ha tenido una onda cuadrada (0 ~ 5) V que fue enviada al receptor para que la procesara. Sin embargo, la señal que llega al receptor está deformada, pero con posibilidad de recuperarla, por lo que se debe “limpiar” y “acondicionar” de la mejor forma posible para restablecer la señal transmitida originalmente. De esto se encarga un buffer o compuerta de disparo Schmitt (Schmitt trigger) la cual posee un rango de inmunidad al ruido entre dos niveles lógicos denotados como V_{T^-} y V_{T^+} (V_T : Tensión Umbral). La diferencia entre las dos tensiones umbrales se llama histéresis y tiene un valor de 0.8 V para la familia TTL estándar. La señal con ruido es pasada por un buffer seguidor que experimenta cambios en la salida solo en los umbrales V_{T^+} y V_{T^-} .

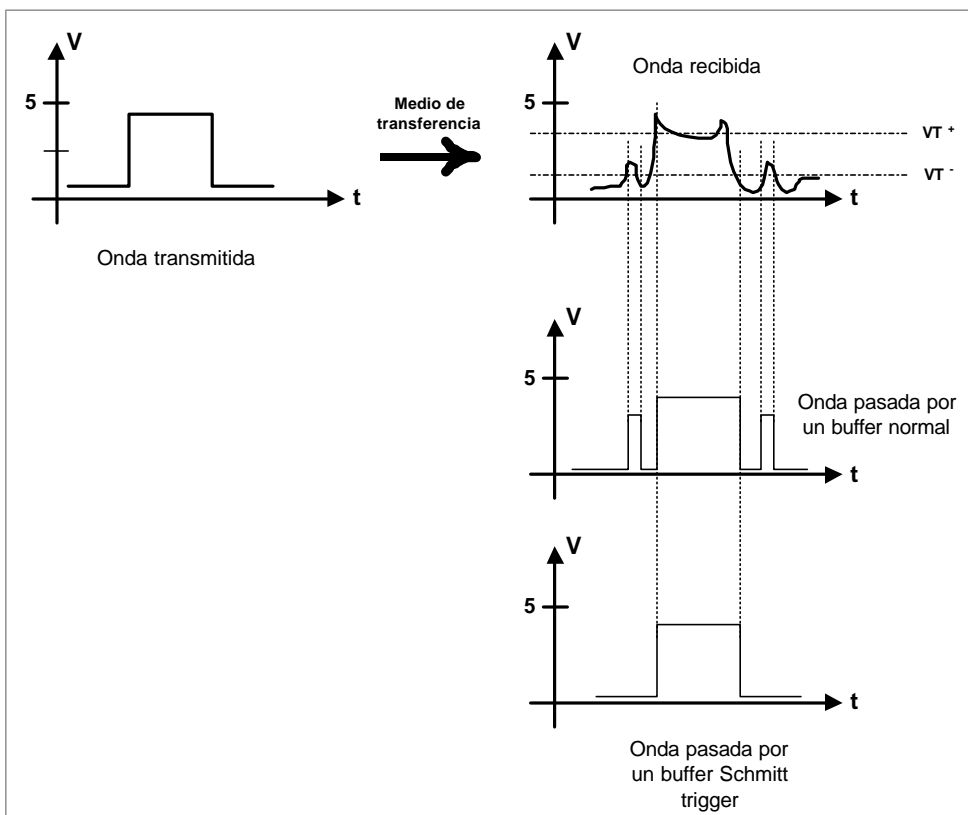


Figura 4.17. Recuperación de la forma de onda con buffer Schmitt trigger.

No obstante, cuando pasa por un buffer normal pueden ocurrir cambios no deseados en la forma de onda.

La figura 4.18 representa las curvas de transferencia del circuito integrado TTL 7414; éste posee, internamente, seis compuertas inversoras con entrada Schmitt trigger. Las características de las tensiones umbrales se describen a continuación:

Símbolo	Descripción	Valores (Volt)
V_{T^+}	Voltaje umbral en flanco de subida	1.7
V_{T^-}	Voltaje umbral en flanco de bajada	0.9
$V_h = V_{T^+} - V_{T^-}$	Voltaje de histéresis	0.8

Las otras características internas del chip 7414 son equivalentes a las compuertas normales de la familia TTL estándar.

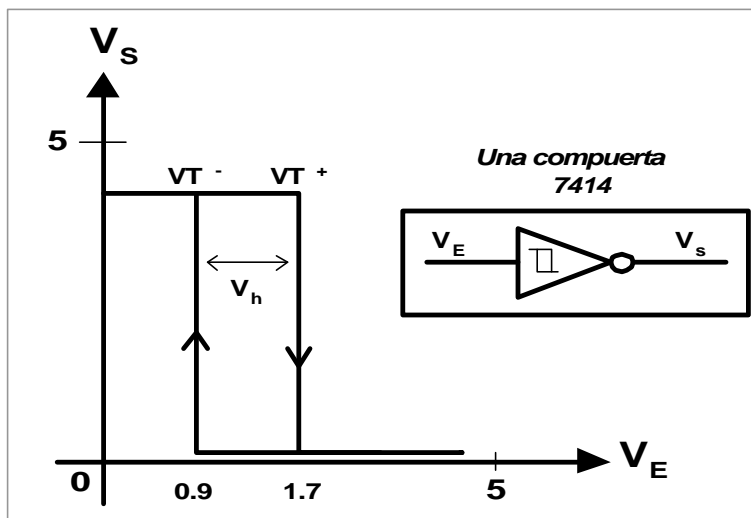


Figura 4.18. Curva de transferencia de la compuerta Schmitt trigger 7414.

4.2.5.1 Aplicaciones de las compuertas Schmitt trigger.

Estos dispositivos son ideales para aplicaciones donde la señal de entrada cambia con lentitud o para limpiar señales con ruido. Se utilizan para acondicionar señales provenientes de sensores magnéticos, tacómetros, señales de fotoresistencias, resistencias térmicas, "One-Shot" para retardo de pulsadores de teclas, Osciladores de

Onda cuadrada, buffer de tres estados para transferencia de información, etc. La figura 4.19 muestra dos aplicaciones típicas utilizando el inversor 7414, la primera es un generador de onda cuadrada a partir de una senoidal y la segunda es un circuito para iluminación nocturna con fotoresistencia y termoresistencia.

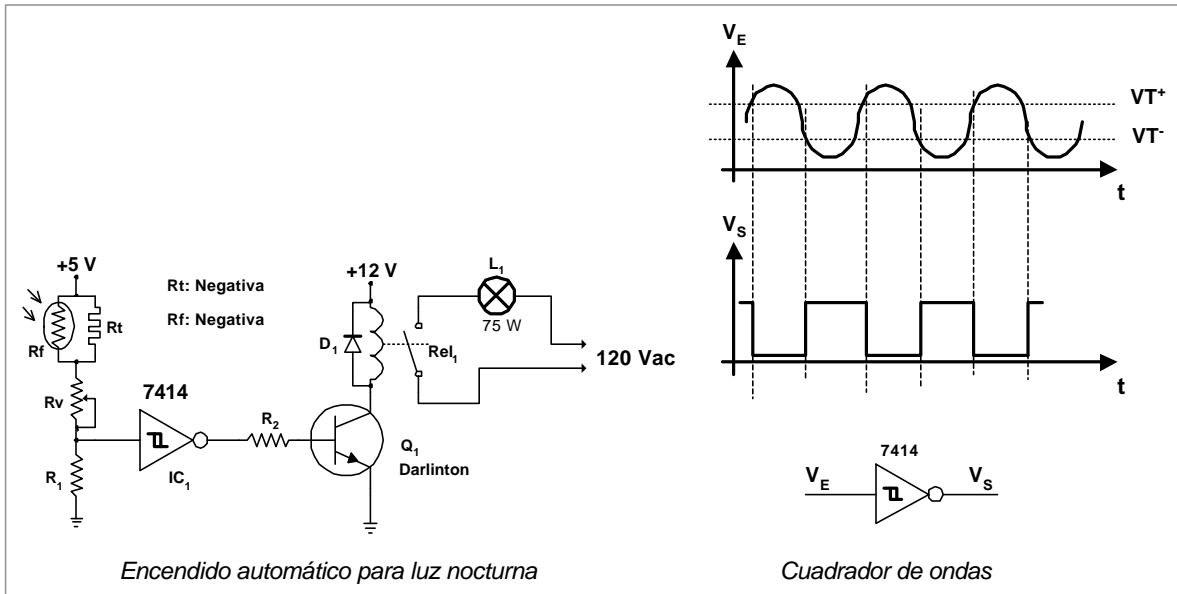


Figura 4.19. Aplicaciones utilizando las compuertas inversoras Schmitt trigger 7414.

4.3 Lógica CMOS.

Los circuitos integrados CMOS están constituidos por MOSFET de canal N y MOSFET de canal P. Presentan gran impedancia de entrada y su capacidad de integración los coloca en el renglón de la tecnología de mediana y alta escala de integración. En la figura 4.20(a) se observa el corte transversal de un MOSFET canal N de enriquecimiento; la circulación de corriente i_{DS} se establece cuando la tensión V_{GSN} supera la tensión umbral V_{ThN} . Por debajo de esta tensión el MOSFET queda en corte y, la completa conducción se establece cuando:

$$V_{ThN} \leq V_{GSN} \leq V_{DD} \quad \text{Ec. 4.13}$$

Las figuras 4.20(a) y (b) muestran dos símbolos utilizados por los transistores MOSFET de enriquecimiento y de agotamiento.

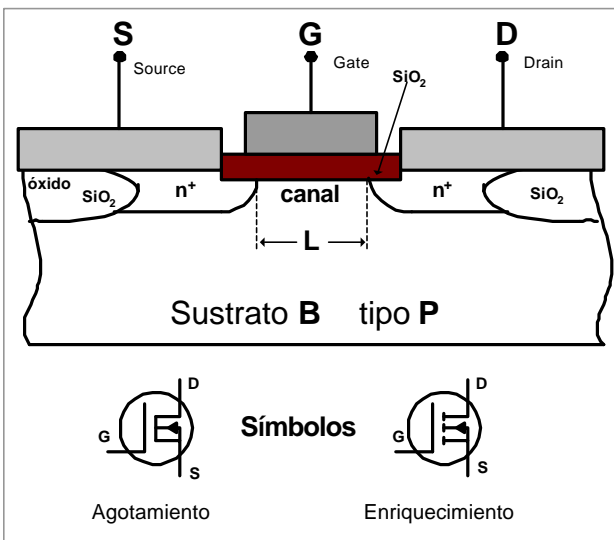


Figura 4.20(a). MOSFET canal N.

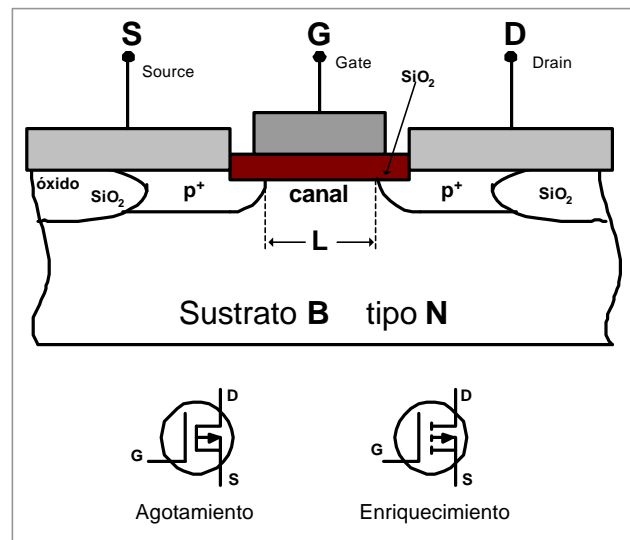


Figura 4.20(b). MOSFET canal P.

El significado de CMOS (Complementary Metal Oxide Semiconductor) implica que deben ser utilizados, dos tipos de canal "NMOS" y "PMOS", para fabricar los circuitos integrados lógicos. Los electrodos (**S**: source: fuente); (**G**: gate: puerta); (**D**: drain: fuente), sirven para polarizar el dispositivo. En la figura 4.21 se muestra el circuito interno de este tipo de arreglo complementario, donde Q_1 y Q_2 son los MOSFET canal P y canal N respectivamente. Las puertas (gates) de los dispositivos MOSFET se

conectan entre sí para formar la entrada (**Vi**) y, del mismo modo, los dos drenajes (Drain) para formar la salida (**Vo**).

Estos transistores están formados por tecnologías de enriquecimiento, las tensiones umbrales de Q_1 y Q_2 son V_{ThP} y V_{ThN} y los parámetros de conducción K_p y K_n . También las gráficas de la figura 4.21 (b) y (c) indican el funcionamiento de los MOSFET.

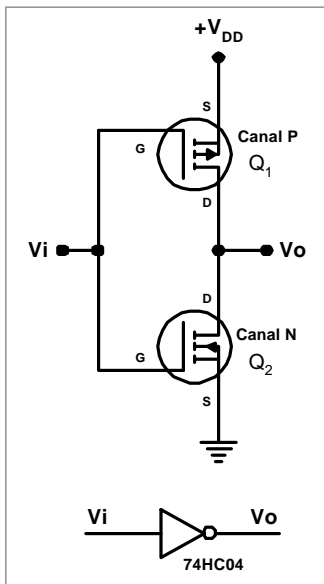


Figura 4.20(a). Inversor CMOS.

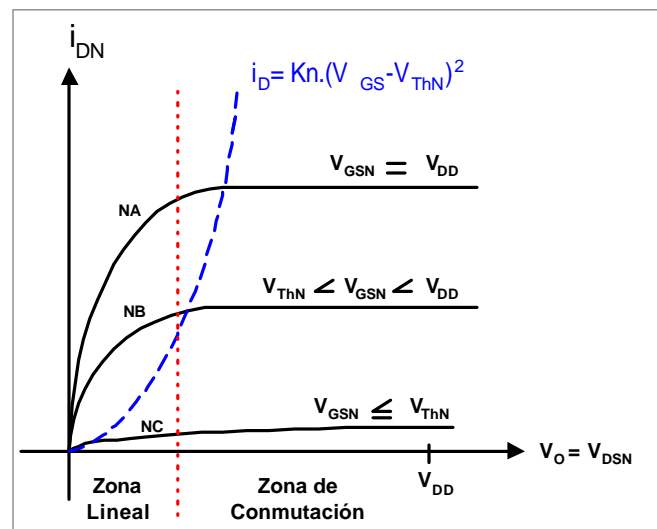


Figura 4.20(b). Gráfica del transistor NMOS.

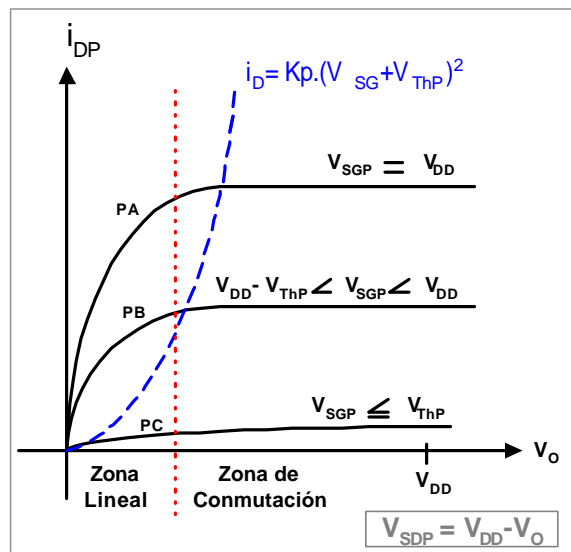


Figura 4.20(c). Gráfica del transistor PMOS.

La tensión de entrada V_i de la compuerta NOT es la misma para Q_1 y Q_2 . Esta es equivalente a las tensiones:

$$V_i = V_{GSN} \quad \text{Para el transistor de canal N, } Q_2.$$

$$V_i = V_{DD} - V_{ThP} \quad \text{Para el transistor de canal P, } Q_1.$$

De esta forma la tensión de entrada queda en el intervalo

$$V_{DD} - V_{ThP} \leq V_i \leq V_{DD} \quad \text{Ec. 4.14}$$

Al igualar las corrientes de drenaje ($i_D = i_{DN} = i_{DP}$) en la unión de salida con respecto a la unión de entrada (V_i) del inversor para los dos transistores, se obtiene:

$$\begin{aligned} K_n.(V_{GSN} - V_{ThN})^2 &= K_p.(V_{SGP} + V_{ThP})^2 \\ K_n.(V_i - V_{ThN})^2 &= K_p.(V_{DD} - V_i + V_{ThP})^2 \end{aligned} \quad \text{Ec. 4.15}$$

$$V_i = \frac{V_{DD} + V_{ThP} + V_{ThN} \cdot \sqrt{\frac{K_n}{K_p}}}{1 + \sqrt{\frac{K_n}{K_p}}} \quad \text{Ec. 4.16}$$

Para $V_i = 0$ el NMOS se pone en corte y actúa como un circuito abierto para Q_2 , ver figura 4.21(a); en ese momento $i_D = i_{DN} = i_{DP} = 0$. Al mismo tiempo el V_{SGP} del MOSFET de canal P (Q_1) queda polarizado de acuerdo con la curva **PA** de la figura 4.21(c). En este caso $V_{SDP} = 0 = V_{DD} - V_O$, por lo tanto $V_O = V_{DD}$; esta condición existe siempre que el transistor NMOS (Q_2) esté en corte, o V_i sea menor o igual que la tensión umbral V_{ThN} .

Para $V_i = V_{DD}$, el transistor MOSFET de canal P queda en corte, $i_{DP} = 0$, e $i_{DN} = 0$. El V_{GSN} de Q_1 es V_{DD} y el mismo está en conducción según la curva **NA** de la gráfica de la figura 4.21(b). El voltaje de salida $V_O = 0$ siempre que Q_2 esté en la región de corte, o $V_{SGP} = V_{DD} - V_i \leq V_{ThP}$. El intervalo de tensión de entrada en el inversor CMOS viene dado por: $V_{DD} - V_{ThP} \leq V_i \leq V_{DD}$.

En la curva de la figura 4.22, se observa la zona de transición entre el nivel bajo (L) y el nivel alto (H). El V_{ON} es la tensión de salida cuando el transistor NMOSFET (Q_2) está conduciendo y el V_{OP} es la tensión de salida cuando el transistor PMOSFET (Q_1) conduce. La zona de transición se conoce como zona indeterminada.

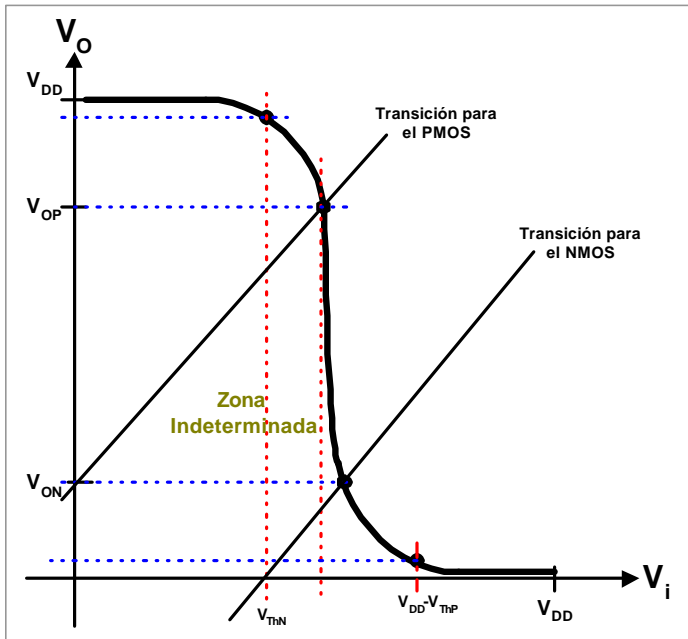


Figura 4.22. Gráfica de transferencia de voltaje del inversor CMOS.

4.3.1 Compuerta Lógica NOR.

La figura 4.23 muestra una compuerta NOR de dos entradas de la familia CMOS. El número de esta compuerta puede ser: 74HC02, 74C02 o 74HCT02. Cuando las entradas A y B tienen un nivel lógico bajo entonces los NMOS Q_3 y Q_4 se ponen en corte. Al mismo tiempo, los transistores PMOS Q_1 y Q_2 tienen tensión cero en sus puntos D (Drain) y S (Source); drenaje y fuente respectivamente. Por lo que, la tensión V_o pasa a tener un nivel lógico alto equivalente a la tensión V_{DD} . Por otra parte, si alguna de las entradas (o las dos) se colocan en uno lógico Q_1 y/o Q_2 se ponen en corte. Al mismo tiempo, Q_3 y/o Q_4 con su V_{DS} igual a cero hacen que la tensión V_o pase a nivel bajo ($V_o=0$).

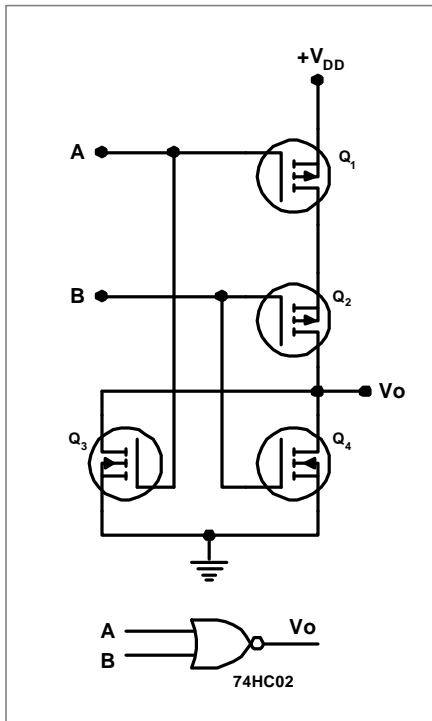


Figura 4.23. Compuerta NOR 74HC02.

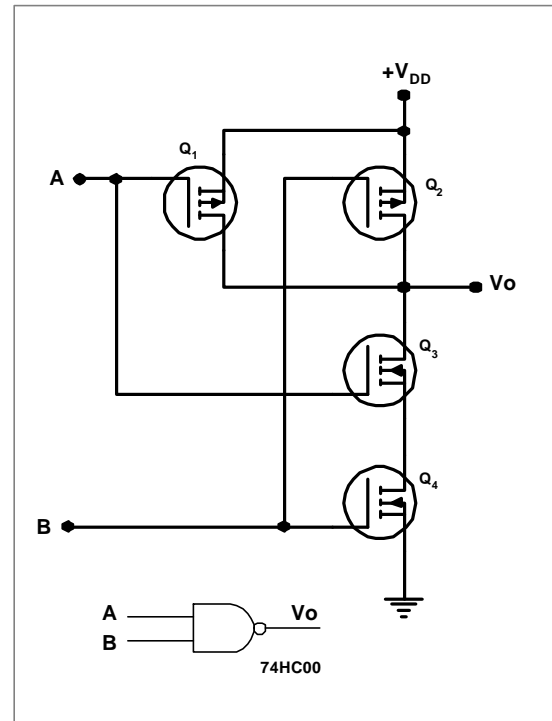


Figura 4.24. Compuerta NAND 74HC00.

4.3.2 Compuerta Lógica NAND.

La figura 4.24 muestra una compuerta digital NAND de dos entradas de la familia CMOS. El número de esta compuerta puede ser: 74HC00, 74C00 o 74HCT00. Cualquiera de las dos entradas A o B que se coloque a un nivel lógico bajo hace que Q_1 y/o Q_2 tengan una tensión V_{DS} igual a cero; del mismo modo, Q_3 o Q_4 se colocarán en corte y en consecuencia, la tensión V_o será igual a V_{DD} . Por otra parte, cuando A y B tienen un nivel lógico alto Q_1 y Q_2 se ponen en corte y al mismo tiempo, las V_{DS} de Q_3 y Q_4 pasan a valer cero por lo que V_o también se coloca en un nivel lógico bajo.

4.3.3 Compuertas Lógicas OR y AND.

Para obtener estas compuertas digitales de la familia CMOS deben integrarse en la base del material semiconductor compuertas tipo NAND y NOR conjuntamente con inversores CMOS, conectados internamente a la salida de dichas compuertas. Las figuras 4.25 y 4.26 muestran las configuraciones internas de las compuertas OR y AND respectivamente.

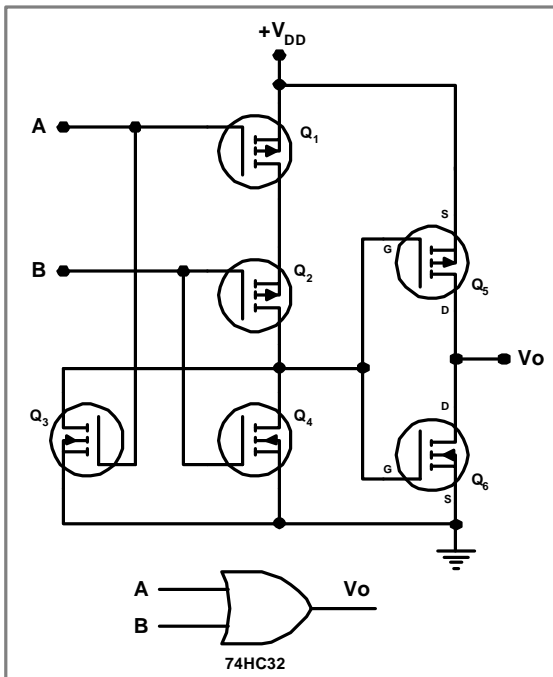


Figura 4.25. Compuerta OR 74HC32

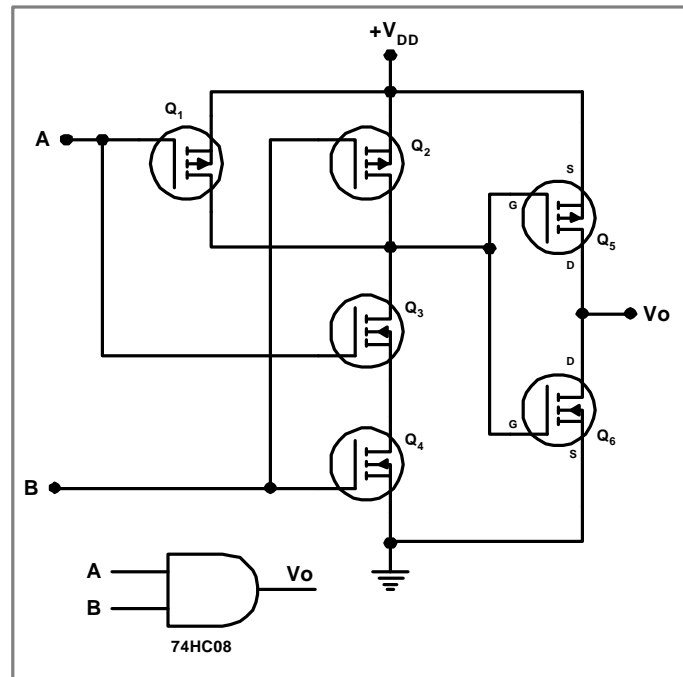


Figura 4.26. Compuerta AND 74HC08.

Las figuras 4.25 y 4.26 indican como están constituidas internamente las compuertas básicas OR y AND de la familia CMOS. La combinación de ellas dos, conjuntamente con los inversores NOT dan como resultado las compuertas de tipo exclusivo: OR-Exclusiva y NOR-Exclusiva.

Las características internas de ésta familia son un poco diferentes a la familia TTL, ya que la impedancia de entrada/salida de los circuitos CMOS es bastante alta y el consumo de corriente es muy bajo. Esto trae como consecuencia tiempos de propagación muy largos en la respuesta de los mismos.

4.3.4 Características de las compuertas CMOS.

Los circuitos integrados CMOS han evolucionado en el proceso de fabricación. El avance fundamental ha sido la reducción del tamaño del área de fabricación del material semiconductor; han reducido el área de la compuerta estándar (4XXX) casi a la mitad por lo que el canal de conducción se ha reducido también. Las compuertas CMOS estándar se realizan en una capa de material base (silicio) de 120 micrones y los chips de alta velocidad CMOS (HCXXXX) son fabricados sobre una capa de 65 micrones. Esto hace aumentar la integración de la serie HC; reduce el solapamiento de capas que se hacía anteriormente en la serie estándar para aumentar la cantidad de puertas; disminuye la capacitancia intrínseca y por ende disminuyen los tiempos de respuestas de estos dispositivos. También se han integrados diodos de protección en los pines de entrada del chip con la finalidad de dar protección contra los choques electrostáticos.

Los resultados de estos cambios se muestran en la tabla 4.5 donde los dispositivos **HC** son comparados con las series **estándar CMOS**, **LSTTL** y **ALS**. Existe también una sub-serie con la nomenclatura **HCT** de la gran familia CMOS que es compatible pin a pin con los circuitos integrados de la familia TTL. Esto significa que poseen internamente elementos que igualan las impedancias de entrada y salida para que puedan ser compatibles en voltajes y corrientes con los chips TTL.

4.3.4.1 Disipación de potencia de las compuertas CMOS.

El inversor CMOS y los dispositivos lógicos en general se utilizan para excitar a otros circuitos, la impedancia de estos dispositivos se puede modelar como una capacitancia. Por lo que, durante la conmutación de los niveles lógicos, esta carga capacitiva se debe cargar y descargar. La figura 4.27 muestra los periodos de carga y descarga del condensador de carga C_L en el inversor CMOS. En las figuras 4.27(b) y 4.27(c) se observan la carga y descarga respectivamente. Aquí se asume, como condición inicial, que el condensador está descargado totalmente.

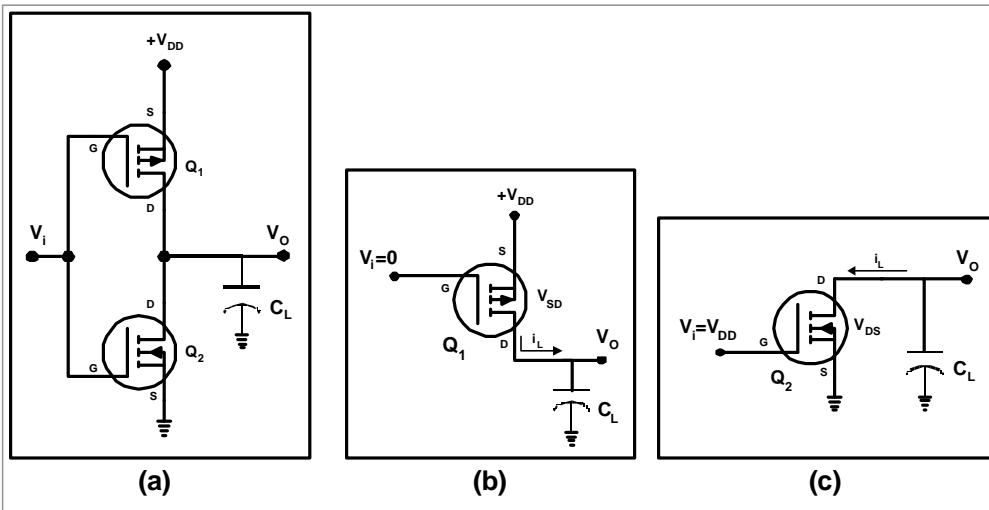


Figura 4.27. Carga y descarga del condensador C_L .

La sección (b) indica el momento en que se carga C_L a través del transistor PMOSFET Q_1 . La disipación de potencia en el transistor está determinada por:

$$P_{Q_1} = i_L \cdot V_{SD} = i_L \cdot (V_{DD} - V_O) \quad \text{Ec. 4.17}$$

La corriente a través del condensador C_L es:

$$i_L = C_L \cdot \frac{dv_O}{dt} \quad \text{Ec. 4.18}$$

La energía que se disipa en Q_1 cuando la salida conmuta de nivel bajo a alto es:

$$E_{Q_1} = \int_0^\infty P_{Q_1} \cdot dt = \int_0^\infty C_L (V_{DD} - V_O) \frac{dv_O}{dt} dt = C_L \cdot V_{DD} \int_0^{V_{DD}} dv_O - C_L \int_0^{V_{DD}} v_O dv_O$$

$$E_{Q_1} = C_L \cdot V_{DD}^2 - C_L \cdot \frac{V_{DD}^2}{2} = C_L \cdot \frac{V_{DD}^2}{2}$$

El condensador C_L almacena la energía E_{Q_1} ; luego, toda esta energía es consumida o disipada por Q_2 cuando el inversor cambia de nivel alto a bajo, ver figura 4.27(c). De modo que la energía en Q_2 es:

$$E_{Q_2} = C_L \cdot \frac{V_{DD}^2}{2}$$

Ahora, la energía total queda de la forma:

$$E_T = E_{Q_1} + E_{Q_2} = \frac{C_L \cdot V_{DD}^2}{2} + \frac{C_L \cdot V_{DD}^2}{2} = C_L \cdot V_{DD}^2$$

$$E_T = C_L \cdot V_{DD}^2 \quad \text{Ec. 4.19}$$

La potencia disipada se traduce en la energía consumida en unidades de tiempo; por lo que, el circuito debe conmutar a determinada frecuencia f . Por lo tanto, la potencia que disipa el inversor CMOS es:

$$P_D = E_T \cdot f = C_L \cdot V_{DD}^2 \cdot f \quad \text{Ec. 4.20}$$

Lo que indica que la disipación de potencia de un circuito integrado CMOS es proporcional a la frecuencia de conmutación; al cuadrado de la tensión de alimentación y a la capacitancia de carga. Las fabricantes de circuitos integrados realizan permanentemente investigaciones para reducir la disipación de potencia y de igual forma buscan aumentar la respuesta de frecuencia de estos dispositivos.

El consumo de potencia de los dispositivos CMOS (series Estándar y HC), ver tabla 4.5, depende de varios factores internos y externos. No obstante, aquí se van ha tomar en cuenta los cuatro factores más importantes como lo son:

- *Voltaje de la fuente de alimentación (V_{CC} o V_{DD})*. Como se observa en la tabla 4.5 las características de los circuitos CMOS Estándar y HC varían de rango en función del valor de tensión de la fuente. En la serie Estándar el rango va desde 3.0 hasta 18 voltios y para la serie HC el rango va desde 2.0 hasta 6.0 voltios.
- *Frecuencia de operación (f)*. Los dispositivos CMOS consumen energía solo en las transiciones de los niveles lógicos. Por esto al aumentar la frecuencia en las señales de entrada también se incrementa el consumo del dispositivo. La frecuencia debe estar dada en **MHz**.
- *Capacitancia interna (C_{PD})*. Es la capacitancia intrínseca de la fabricación del dispositivo. Por lo general, está dada en pico faradios **pf**.
- *Capacitancia de la carga (C_L)*. Carga total capacitiva presente en el pin de salida. Se debe sumar todas las capacidades que se encuentres en la línea y se maneja en **pf**.

Características generales y parámetros de las compuertas.						
Características	Símbolo	TTL		CMOS		Unidades
		LS	ALS	4xxxx (STD)	HC	
Rango de Voltaje de alimentación	V_{CC}, V_{DD}	5 +/-5%	5 +/-5%	3.0 hasta 18	2.0 hasta 6.0	V
Rango de temperatura	T_A	0 ~ +70	0 ~ +70	-40 ~ +85	-55 ~ +125	°C
Parámetros en el voltaje de entrada	$V_{IH}(\min)$	2.0	2.0	3.5	3.5	V
	$V_{IL}(\max)$	0.8	0.8	1.5	1.0	V
Parámetros en el voltaje de Salida	$V_{OH}(\min)$	2.7	2.7	$V_{DD} - 0.05$	$V_{CC} - 0.1$	V
	$V_{OL}(\max)$	0.5	0.5	0.05	0.1	V
Corriente de Entrada	$I_{IH}(\max)$	20	20	0.3	1.0	μA
	$I_{IL}(\max)$	-400	-200	-0.3	-1.0	μA
Corriente de Salida	$I_{OH}(\max)$	-0.4	-0.4	-2.1 @ 2.5V	-4.0 @ $V_{CC} - 0.8V$	mA
	$I_{OL}(\max)$	8.0	8.0	0.44 @ 0.4V	4.0 @ 2.5V	mA
Margen de ruido en DC	V_{NSL}	0.3	0.3	1.45 @ 5V	0.90 @ 5V	V
	V_{NSH}	0.7	0.7	1.45 @ 5V	1.35 @ 5V	V
Fan out DC	-----	20	20	50 [1 LSTTL]	50 [10 LSTTL]	-----
Características de potencia y velocidad						
Consumo estático de corriente por compuerta	I_G	0.4	0.2	0.0001	0.0005	mA
Potencia por compuerta estática	P_G	2.0	1.0	0.0006	0.001	mW
Tiempo de propagación	t_p	9.0	7.0	125	8.0	ns
Producto velocidad potencia	-----	18	7.0	0.075	0.01	pJ
Frecuencia máxima (FF; Reg)	F_{max}	33	35	4.0	40	MHz
Frecuencia máxima (contadores)	F_{max}	40	45	5.0	40	MHz

Tabla 4.5. Características técnicas de las familias LSTTL, ALS, Estándar CMOS y HC.

Los requerimientos de consumo de energía del dispositivo en régimen dinámico se calcula con la Ec.4.20 en la que se ha agregado la capacitancia intrínseca de la fabricación del circuito (C_{PD}):

$$P_D = (C_L + C_{PD}) \cdot V_{CC}^2 \cdot f \quad \text{Ec. 4.21}$$

Sin embargo, la disipación total debe tomar en cuenta la componente estática DC ($V_{CC} \cdot I_{CC}$) más la parte dinámica $[(C_L + C_{PD}) \cdot V_{CC}^2 \cdot f]$ de tal manera que la ecuación completa sin carga acoplada queda de la siguiente forma:

$$P_D = C_{PD} \cdot V_{CC}^2 \cdot f + V_{CC} \cdot I_{CC} \quad \text{Ec. 4.22}$$

Ejercicio 4.1. Calcular la disipación total de potencia del chip 74HC00 cuando cada una de sus compuertas (G_1, G_2, G_3, G_4) son sometidas a las siguientes condiciones:

G_1 . Trabaja a una frecuencia de 1 KHz.

G_2 . Trabaja a una frecuencia de 1 MHz.

G_3 . Trabaja con tensión fija DC.

G_4 . Trabaja con tensión fija DC.

El fabricante indica que los valores de consumo de corriente estática del encapsulado completo a temperatura ambiente son $2 \mu A$ y una capacitancia intrínseca de 22 pf por compuerta. La alimentación externa del chip es 5 Voltios con una carga capacitiva para cada compuerta de 50 pf.

Solución: La solución se obtiene aplicando las ecuaciones 4.21 y 4.22.

$I_{CC} = 2 \text{mA}$ En condiciones estáticas a temperatura ambiente.

$C_{PD} = 22 \text{ pf}$; $C_L = 50 \text{ pf}$ y $V_{CC} = 5 \text{ Voltios}$

$$P_D = (C_{PD} + C_L) \cdot V_{CC}^2 f + V_{CC} I_{CC}$$

$$P_{D_1} = (22 \text{ pf} + 50 \text{ pf}) \cdot (5V)^2 (1 \text{ KHz}) = 1.8 \text{ mW}$$

$$P_{D_2} = (22 \text{ pf} + 50 \text{ pf}) \cdot (5V)^2 (1 \text{ MHz}) = 1800 \text{ mW}$$

$$P_{D_3} = (22 \text{ pf} + 50 \text{ pf}) \cdot (5V)^2 (0 \text{ Hz}) = 0 \text{ mW}$$

$$P_{D_4} = (22 \text{ pf} + 50 \text{ pf}) \cdot (5V)^2 (0 \text{ Hz}) = 0 \text{ mW}$$

$$P_D(\text{total}) = V_{CC} I_{CC} + P_{D_1} + P_{D_2} + P_{D_3} + P_{D_4}$$

$$P_D(\text{total}) = (5V \cdot 2 \text{ mA}) + 1.8 \text{ mW} + 1800 \text{ mW} + 0 + 0$$

$$P_D(\text{total}) = (10 \text{ mW}) + 1.8 \text{ mW} + 1800 \text{ mW}$$

$$P_D(\text{total}) = 1812 \text{ mW}$$

4.3.4.2 Margen de ruido de las compuertas CMOS.

En los circuitos integrados CMOS el margen de ruido aumenta a medida que se incrementa la tensión de alimentación (V_{CC} o V_{DD}); esto es una ventaja para el diseño con dispositivos de esta familia de chips. Sin embargo, el aumento de tensión incrementa la disipación de potencia y como consecuencia, reduce la respuesta de frecuencia del chip. El diseñador debe sopesar los requerimientos de disipación, voltaje, frecuencia y consumo de corriente del circuito digital a la hora de realizar el prototipo.

En este particular los simuladores básicos digitales no ofrecen mucha ayuda debido a que están hechos con modelos matemáticos lógicos que no toman en cuenta estos márgenes de ruido de señales y variaciones eléctricas. No obstante, los simuladores profesionales mixtos (Analógicos – Digitales) como el SPICE si pueden ser configurados para tomar en cuenta las variaciones, ruidos y tolerancias eléctricas a las que deba ser sometido el diseño antes de realizar el prototipo.

La tabla 4.6 muestra los márgenes de ruido que posee la compuerta 74HC08. Aquí se observa que el $V_{NSH} = V_{OH(mín)} - V_{IH(mín)}$ y $V_{NSL} = V_{IL(máx)} - V_{OL(máx)}$, con una alimentación de 2 Voltios es 0.4 Voltios y, con una tensión de alimentación de 6 Voltios el V_{NSH} y V_{NSL} es igual a 1.7 Voltios.

4.3.4.3 Impedancia de salida CMOS.

La impedancia de salida depende del estado que posea el circuito integrado CMOS. Está definida como R_O y se presentan dos casos: R_{OH} cuando el nivel lógico de la salida es alto y R_{OL} cuando el nivel lógico de la salida es bajo.

$$V_{OH}(\text{mín}) \leq V_{CC} - I_{OH}(\text{máx}) \cdot R_{OH}$$

$$R_{OH} \leq \frac{V_{CC} - V_{OH}(\text{mín})}{I_{OH}(\text{máx})} \quad \text{Ec. 4.23}$$

$$V_{OL}(\text{máx}) \geq I_{OL}(\text{máx}) \cdot R_{OL}$$

$$R_{OL} \leq \frac{V_{OL}(\text{máx})}{I_{OL}(\text{máx})} \quad \text{Ec. 4.24}$$

4.3.4.4 Tiempo de propagación de los dispositivos CMOS.

El problema de la tecnología CMOS son los tiempos de retardo en las respuestas de las señales digitales. La tabla 4.5 muestra el retardo de la serie Estándar CMOS ($t_p=125$ ns) y la serie HC ($t_p=8.0$ ns) ésta última iguala y hasta mejora los tiempos de propagación de la serie LSTTL. En la tabla 4.7 se describen los tiempos de propagación t_{pHL} y t_{pLH} de algunas compuertas de la serie HC.

Parámetro	Símbolo	Condiciones	V _{CC} o V _{DD} Volt	Temperatura °C		Unidades
				≤ 85	≤ 125	
Voltaje mínimo de entrada en alto	V _{IH} (min)	V _{out} = 0.1V ó V _{CC} -0.1V I _{out} ≤ 20 μA	2.0	1.50	1.50	V
			3.0	2.10	2.10	V
			4.5	3.15	3.15	V
			6.0	4.20	4.20	V
Voltaje máximo de entrada en bajo	V _{IL} (max)	V _{out} = 0.1V ó V _{CC} -0.1V I _{out} ≤ 20 μA	2.0	0.50	0.50	V
			3.0	0.90	0.90	V
			4.5	1.35	1.35	V
			6.0	1.80	1.80	V
Voltaje mínimo de salida en alto	V _{OH} (min)	V _{in} = V _{IH} ó V _{IL} I _{out} ≤ 20 μA	2.0	1.90	1.90	V
			4.5	4.40	4.40	V
			6.0	5.90	5.90	V
		I _{out} ≤ 2.4 mA	3.0	2.34	2.20	V
		I _{out} ≤ 4.0 mA	4.5	3.84	3.70	V
Voltaje máximo de salida en bajo	V _{OL} (max)	V _{in} = V _{IH} ó V _{IL} I _{out} ≤ 20 μA	2.0	0.1	0.1	V
			4.5	0.1	0.1	V
			6.0	0.1	0.1	V
		I _{out} ≤ 2.4 mA	3.0	0.33	0.40	V
		I _{out} ≤ 4.0 mA	4.5	0.33	0.40	V
		I _{out} ≤ 5.2 mA	6.0	0.33	0.40	V
Máxima corriente de entrada	I _{in}	V _{in} = V _{CC} ó GND	6.0	± 1.0	± 1.0	μA
Máxima corriente de suministro	I _{CC}	V _{in} = V _{CC} ó GND Estático: I _{out} = 0 mA	6.0	10	40	μA

Tabla 4.6. Características de tensiones del chip 74HC08 de la Motorola High-Speed CMOS, DL129-Rev 6.

Símbolo	Tipo de compuerta	V _{DD} ó V _{CC} Volt	Temperatura de trabajo			Unidades
			-55 a 25 °C	≤ 85 °C	≤ 125 °C	
t _{pLH} y t _{pHL}	74HC00	2.0	75	95	110	ns
		3.0	30	40	55	ns
		4.5	15	19	22	ns
		6.0	13	16	19	ns
C _{in}	Capacitancia máxima por cada pin de entrada		10	10	10	pf
C _{PD}	Capacitancia para la disipación de potencia (intrínseca)		-----	22	-----	pf
t _{pLH} y t _{pHL}	74HC86	2.0	100	125	150	ns
		3.0	80	90	110	ns
		4.5	20	25	31	ns
		6.0	17	21	26	ns
C _{in}	Capacitancia máxima por cada pin de entrada		10	10	10	pf
C _{PD}	Capacitancia para la disipación de potencia (intrínseca)		-----	33	-----	pf
t _{pLH} y t _{pHL}	74HC32	2.0	75	95	110	ns
		3.0	30	40	55	ns
		4.5	15	19	22	ns
		6.0	13	16	19	ns
C _{in}	Capacitancia máxima por cada pin de entrada		10	10	10	pf
C _{PD}	Capacitancia para la disipación de potencia (intrínseca)		-----	20	-----	pf

Tabla 4.7. Tiempos de propagación y capacitancia de los chips 74HC00, 74HC86 y 74HC86.

4.3.4.5 Conectividad de las compuertas CMOS (fan out).

El factor de carga estático de los chips CMOS es bastante alto, debido a que la corriente promedio de entrada y salida de una compuerta de la serie HC es 1μA y 5 mA respectivamente. Esto significa que se deberían acoplar 5000 compuertas a una salida CMOS. Sin embargo, la capacitancia de estos dispositivos disminuye significativamente su rendimiento y en consecuencia también reduce el fan out. Se debe considerar un factor que involucre el efecto de la capacitancia acoplada conjuntamente con los tiempos de transición y la frecuencia de trabajo de las señales aplicadas. Este se conoce como factor dinámico de carga de los chips CMOS o fan-out y se utiliza para saber cuantas entradas de compuertas o pines del chip se pueden conectar a la salida de otra de una misma familia u otra del tipo equivalente.

Ejercicio 4.2. Hallar el fan out dinámico del circuito integrado 74HC00 cuando una salida de compuerta se acopla con N entradas HC; figura 4.28. A continuación, se mencionan las características y condiciones necesarias para resolver este problema:

Símbolo	Unidades
V_{CC}	6 Voltios
f_{in}	5 MHz
$V_{OL}(máx)$	0.33 Voltios
$V_{OH}(mín)$	5.34 Voltios
$V_{iH}(mín)$	4.20 Voltios
$I_{OH}(máx)$	-5.2 mA
$I_{OL}(máx)$	5.2 mA
$I_{in}(máx)$	$\pm 1 \mu A$
$t_r = t_f$	6 ns
t_p	8 ns

Valores y parámetros del ejercicio.

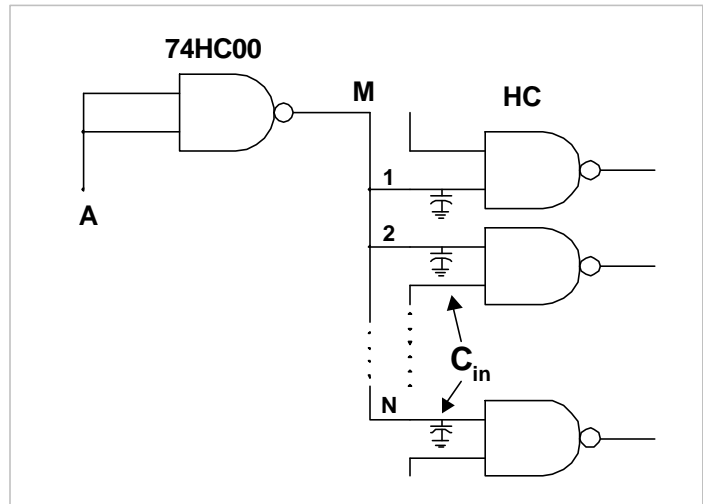


Figura 4.28. Esquema del ejercicio 4.2e.

La carga de la capacitancia total C_T ($C_T = N \cdot C_{in}$) se realiza cuando la señal de entrada **A** pasa de alto a bajo. La constante de tiempo en la carga de C_T viene dada por:

$$t = R_{OH} \cdot N \cdot C_{in} \quad \text{EC. 4.25}$$

donde N es la cantidad de entradas CMOS que se pueden conectar a la salida de la compuerta 74HC00. Las impedancias de salida de la compuerta se obtienen de la siguiente forma:

$$R_{OL} \leq \frac{V_{OL}(máx)}{I_{OL}(máx)} = \frac{0.33V}{5.2mA} = 63.5 \Omega$$

$$R_{OH} \leq \frac{V_{CC} - V_{OH}(mín)}{I_{OH}(máx)} = \frac{6V - 4.20V}{5.2mA} = 346 \Omega$$

Se debe tomar un T mayor que el $t_p + t_r = 14 \text{ ns}$ y esto se puede acotar con la frecuencia de entrada $f_{in} = 5 \text{ MHz}$; significa que el rango debe estar comprendido entre:

$$14 \text{ ns} < t < \frac{1}{5 \text{ MHz}} = 200 \text{ ns}$$

Sin embargo, se toma un margen de seguridad menor o igual

al 50% de este valor.

De este modo el acoplamiento queda limitado a $\tau=100$ ns. El valor aproximado de N es:

$$t \geq R_{OH} \cdot N \cdot C_{in}$$

$$N \leq \frac{t}{R_{OH} \cdot C_{in}} = \frac{100ns}{346\Omega \cdot 10 pf} = \frac{100 \cdot 10^{-9} \cdot 10^{-12} s}{346 \frac{V}{A} \cdot 10 \frac{A \cdot s}{V}} = \frac{100000}{3460} = 28.9$$

$N \leq 28$ Entradas de compuertas CMOS HC.

Con la impedancia de salida en nivel bajo (R_{OL}) se obtiene un factor de 157 entradas HC. Por lo cual, se toma el mínimo de los dos que es 28. Si es necesario un cálculo más exacto se procede con la fórmula de carga para condensadores tomando en cuenta la $V_{iH}(\text{mín})$ en las entradas de las compuertas HC.

$V_C = V_0(1 - e^{-t/RC})$ Fórmula para la carga de un condensador.

$$V_{iH}(\text{mín}) = V_{CC}(1 - e^{-t_r/R_{OH} \cdot N \cdot C_{in}})$$

$$N = \frac{t_r}{-R_{OH} \cdot C_{in} \cdot \ln(1 - \frac{V_{iH}(\text{mín})}{V_{CC}})} = \frac{100ns}{-346\Omega \cdot 10 pf \cdot \ln(1 - \frac{4.20V}{6V})} = 24.1$$

El factor de carga fan out es de 24 entradas para una salida de compuerta 74HC00.

4.3.4.6 Compuertas con drenador abierto.

Son compuertas donde se ha eliminado el transistor PMOS de la salida complementaria interna; por lo que la polarización del drenador del NMOS debe ser realizada con una resistencia externa (R_e). La figura 4.29 muestra el acoplamiento de la resistencia externa en las **N** compuertas 74HC03 unidas todas en las líneas de salida (AND alambradas) y conectadas a ellas, con otras **M** entradas de compuertas CMOS HC. Los dos estados lógicos que se tienen en **Q** determinan las condiciones de tensión y corriente que permiten hallar el valor de la resistencia externa. Sin embargo, debido a que las entradas HC tienen una capacitancia significativa ($C_{in}=10$ pf) el diseñador debe considerar el retardo de tiempo causado por la componente RC ; ($t=R_e \cdot N \cdot C_{in}$).

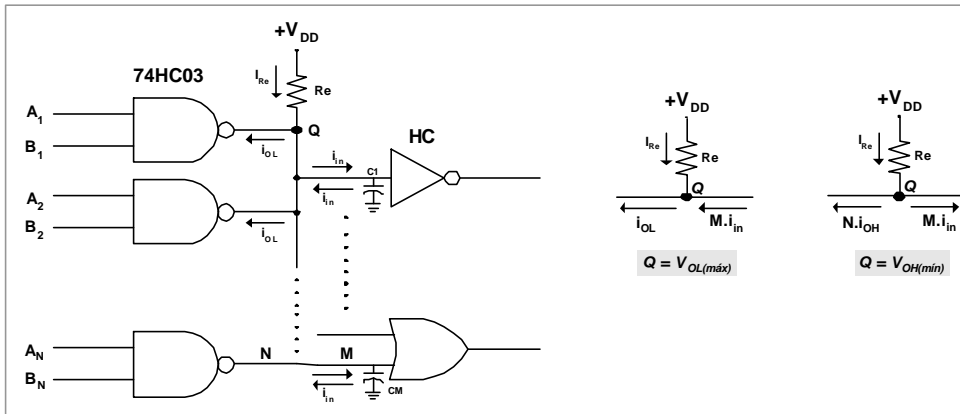


Figura 4.29. Compuertas con drenador abierto.

$$\frac{V_{DD} - V_{OL}(\text{máx})}{I_{OL}(\text{máx}) - M \cdot I_{in}} \leq R_e \leq \frac{V_{DD} - V_{iH}(\text{mín})}{N \cdot I_{OH}(\text{máx}) + M \cdot I_{in}} \quad \text{Ec. 4.26}$$

Ejercicio 4.3. Calcular la resistencia externa que se debe conectar en cuatro compuertas 74HC03 de drenador abierto cableadas en forma de AND alambradas. Estas, a su vez, están conectadas a tres entradas de circuitos integrados HC. Las características y condiciones para corriente continua son las siguientes:

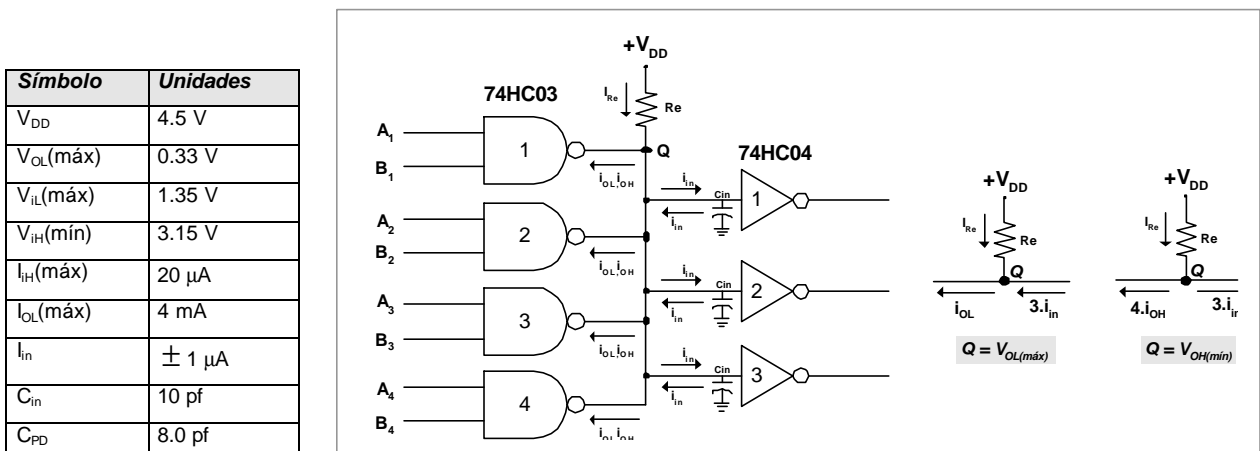


Figura 4.30. Conexión de compuertas del ejercicio 4.3.

Solución: Las condiciones se toman para corriente continua (DC). En la figura 4.29 se describe el sentido de las corrientes en nivel bajo y nivel alto. Aplicando la Ec. 4.26 se obtiene la solución del ejercicio:

$$R_e(\text{mín}) = \frac{V_{DD} - V_{OL}(\text{máx})}{I_{OL}(\text{máx}) - M \cdot I_{in}} = \frac{4.5V - 0.33V}{4mA - 3 \cdot 1mA} = 1043\Omega$$

$$R_e(\text{máx}) = \frac{V_{DD} - V_{iH}(\text{mín})}{N \cdot I_{OH}(\text{máx}) + M \cdot I_{in}} = \frac{4.5V - 3.15V}{4 \cdot 20mA + 3 \cdot 1mA} = 50K\Omega$$

Se debe colocar una resistencia comprendida entre 1043Ω y 50KΩ. No obstante, la resistencia debe ser seleccionada de forma que funcione con un retardo adecuado para la aplicación del circuito. Un valor cercano a la resistencia mínima es necesario cuando la frecuencia de funcionamiento es relativamente grande. Por otra parte, un valor de resistencia máxima reducirá el consumo de corriente de la fuente, pero tendrá el inconveniente de producir retardos y respuestas lentas en las señales.

4.3.4.7 Conmutador o compuertas de transmisión CMOS.

En los circuitos electrónicos es necesario conmutar señales analógicas controlándolas digitalmente. Para lograr esto es necesario utilizar las compuertas de transmisión CMOS, las cuales están constituidas por un arreglo de un transistor NMOS acoplado en paralelo con otro del tipo PMOS. La figura 4.31 muestra la configuración de los dispositivos que forman el switch bilateral. La combinación en paralelo de los dos, conjuntamente con las señales C y \bar{C} permiten que la señal de entrada V_i sea transmitida a la salida sin sufrir alteración. Ambos transistores deben ser simétricos y bilaterales; por lo tanto, los substratos NMOS y PMOS se conectan al potencial más negativo y al más positivo, normalmente se conectan a GND y V_{DD} respectivamente.

En la figura 4.32(a); si $C=V_{DD}$ y $\bar{C}=0$, $V_i=V_{DD}$ y V_o es inicialmente cero, entonces para el dispositivo NMOS el terminal **A** actúa como drenaje y el terminal **B** actúa como fuente, mientras tanto, para el dispositivo PMOS, el terminal **E** actúa como drenaje y el terminal **F** como fuente. La corriente entra al terminal drenaje del dispositivo NMOS y al terminal fuente del dispositivo PMOS, como se muestra en la figura 4.32(a), para cargar al condensador C_L . El voltaje compuerta fuente del NMOS es:

$$V_{GSN} = C - V_o = V_{DD} - V_o; \text{ y en el dispositivo PMOS es:}$$

$$V_{SGN} = V_i - \bar{C} = V_{DD} - 0 = V_{DD}$$

El transistor NMOS queda en corte cuando $V_{SGN}=0$, osea que $V_O=V_{DD}$. No obstante, como $V_{SGP}=V_{DD}$, el dispositivo PMOS sigue conduciendo la corriente por lo que i_{DP} llegará a cero solo cuando el voltaje fuente drenador del PMOS (V_{SDP}) sea cero. De ésta forma, C_L continuará cargándose hasta que iguale el nivel de V_i ; esto es, $V_O=V_i=V_{DD}$. Ahora si las condiciones iniciales son:

$$C=V_{DD}, \bar{C}=0, V_i=0 \text{ y } V_O=V_{DD}.$$

Para el dispositivo NMOS, el terminal **A** actúa como fuente y el **B** como drenaje, mientras que en el transistor PMOS, el terminal **E** actúa como fuente y **F** como drenaje. La corriente entra por los terminales **E** y **F**, y el condensador C_L comienza a descargarse; como se muestra en la figura 4.31(b). El voltaje compuerta fuente del dispositivo NMOS es:

$$V_{GSN} = C - V_i = V_{DD} - 0 = V_{DD}$$

Por otra parte, el voltaje fuente compuerta del PMOS es:

$$V_{SGP} = V_O - \bar{C} = V_O - 0 = V_O$$

El transistor PMOS se pone en corte cuando $V_{SGP}=V_O$; la corriente i_{DP} llega a cero. Sin embargo, como $V_{GSN}=V_{DD}$, el transistor NMOS sigue conduciendo y C_L se descarga completamente hasta llegar a cero. El conmutador electrónico se abre cuando los valores de compuertas son $C=0$ y $\bar{C}=V_{DD}$; debido a que el transistor PMOS y el NMOS quedan en corte.

4.3.4.7.1 Switch analógico - digital cuádruple 74HC4066.

Es un circuito integrado CMOS que posee cuatro conmutadores electrónicos, (Sw1, Sw2, Sw3, Sw4) cada uno con línea de control independiente (C_1, C_2, C_3, C_4) que permiten conmutar señales analógicas y/o digitales bilateralmente en sus terminales de entrada salida ($io_1, io_2, io_3, io_4; oi_1, oi_2, oi_3, oi_4$). La figura 4.33 muestra el diagrama del chip 74HC4066 y un conmutador analógico digital de cuatro canales (Multiplexor ó Demultiplexor). Mediante el contador binario y el decodificador se selecciona el canal que pasa hacia el punto común. Por otra parte, el circuito también puede funcionar en forma inversa; las señales que entran al punto común son enviadas al canal que se

encuentre seleccionado por el decodificador 74HC139 para ese momento. Son diversas las aplicaciones que se pueden realizar con el chip 74HC4066 y los circuitos integrados combinatoriales. No obstante, en el capítulo cinco es donde se analiza el funcionamiento de éstos tipos de circuitos combinatoriales y sus aplicaciones.

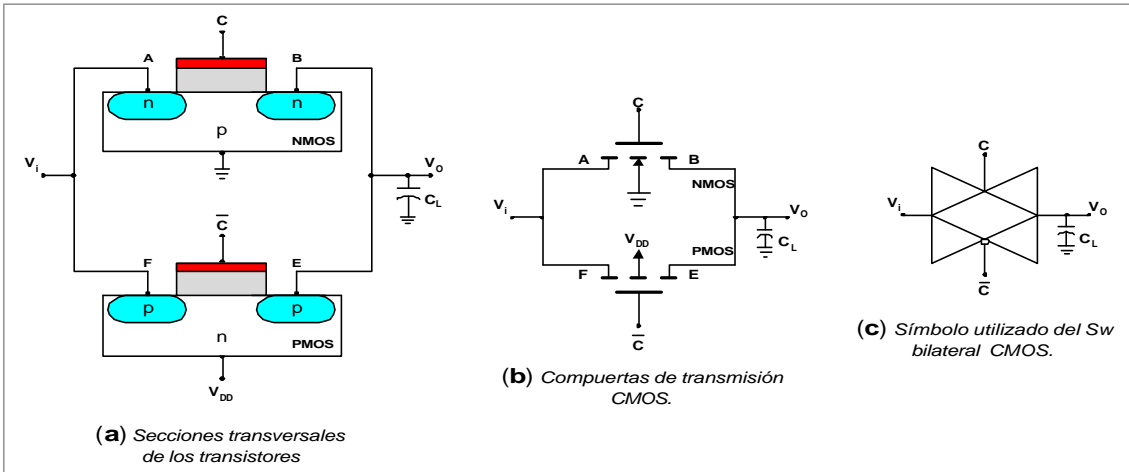


Figura 4.31. Compuerta de transmisión CMOS y símbolo utilizado.

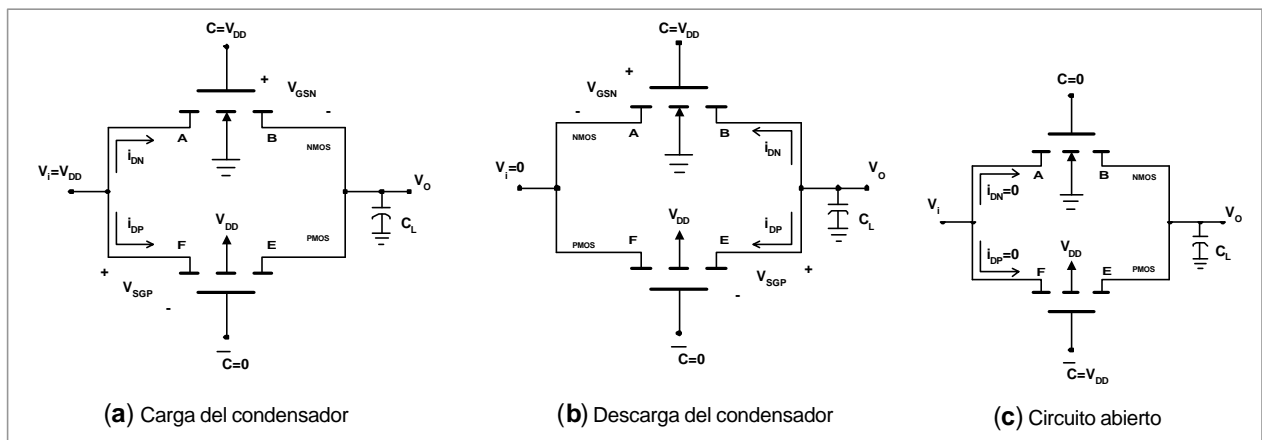


Figura 4.32. Funcionamiento de la compuerta de transmisión CMOS.

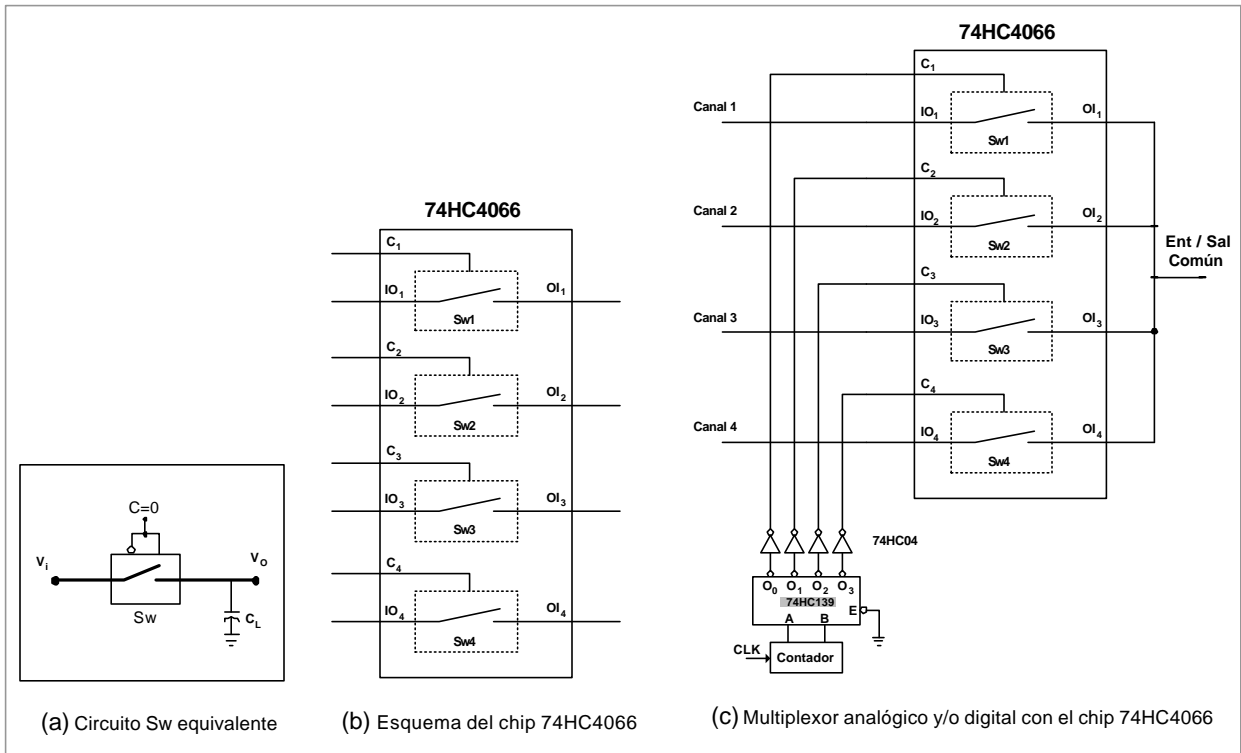


Figura 4.33. Sw equivalente, Circuito integrado CMOS y Multiplexor analógico – digital.

4.4 Lógica ECL

Los circuitos ECL (**Emitter-Coupled Logic**; «lógica de emisores acoplados»), se caracterizan por ser la familia de circuitos convencionales de Silicio con la que se obtiene una mayor velocidad, ya que los tiempos de propagación son del orden de 1ns; solo se utilizan en aplicaciones que requieren muy altas velocidades ya que, considerando otras características, son muy superiores a otros tipos de familias. Su consumo de potencia es muy elevado y son difíciles de miniaturizar, consiguiéndose densidades de integración muy pobre. Los niveles y características de entrada salida no los hacen compatibles con las familias TTL ni con las CMOS.

La lógica TTL, frente a la MOS, ofrece la ventaja de la velocidad; pero ésta es frenada por el hecho de que los transistores entran en saturación, y el paso de saturación a conducción implica consumir un tiempo para eliminar el exceso de electrones de la región de base acumulados por la saturación. En parte se puede reducir este problema utilizando transistores Schottky, pero la lógica ECL ofrece otra alternativa mejor. Cuando el objetivo de diseño es conseguir las mayores velocidades posibles entonces se debe tomar en consideración la lógica ECL. En la Figura 4.34 se muestra el esquema de un amplificador diferencial, circuito en el que se fundamenta la familia ECL. En este circuito vamos a suponer que la ganancia en corriente de los transistores es $\beta=10$ y que los niveles de entrada y salida son los siguientes:

$$V_{iH} = 4.4 \text{ V} \quad V_{iL} = 3.6 \text{ V} \quad V_{oH} = 5.0 \text{ V} \quad V_{oL} = 4.2 \text{ V}$$

En los circuitos ECL se obtiene siempre, simultáneamente, una salida (z_1 , en el circuito de la figura 4.34) y su complementaria z_2 . El transistor Q_2 actúa como transistor de referencia, siendo alimentada su base a una tensión constante $V_{ref} = 4.0\text{V}$. Cuando V_x posee un nivel alto; es decir, $V_x = 4.4 \text{ V}$, Q_1 está en conducción y;
 $V_E = V_i - V_{BE1(on)} = 4.4 \text{ V} - 0.6 \text{ V} = 3.8 \text{ V}$. En estas condiciones Q_2 no conduce, ya que $V_{BE2} = V_B(Q_2) - V_E = 4.0 \text{ V} - 3.8 \text{ V} = 0.2 \text{ V} < 0.6 \text{ V}$.

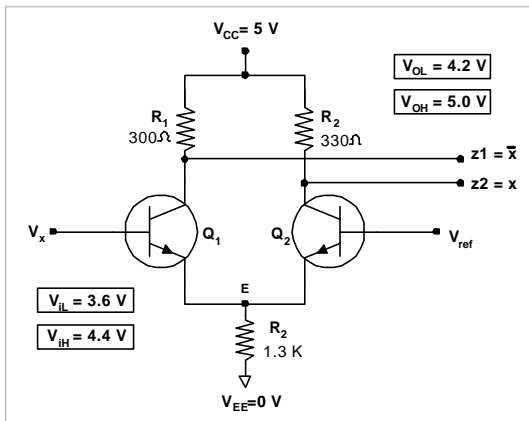


Figura 4.34. Amplificador diferencial utilizado como etapa de entrada en los circuitos ECL.

Se pueden calcular fácilmente las tensiones de salida. Para ello, se debe tener en cuenta que Q_2 está cortando la corriente de emisor de Q_1 , y que ésta aproximadamente es:

$$I_{E1} \approx \frac{V_E}{R_3} = \frac{3.8V}{1.3K\Omega} = 2.92 \text{ mA}$$

Pero, como $I_E = I_C + I_B$, la corriente de colector de Q_1 será:

$$I_{C1} = I_{E1} - I_{B1} = I_{E1} - \frac{I_{C1}}{\hat{\alpha}} = \frac{I_{E1} \cdot \hat{\alpha}}{\hat{\alpha} - 1} = 2.65 \text{ mA}$$

y, en consecuencia la tensión de salida V_{z1} , será:

$$V_{z1} \approx V_{CC} - R_1 \cdot I_{C1} = 5.0V - (300\Omega \cdot 2.65 \text{ mA}) = 4.2V$$

Como Q_2 está en la región de corte, no hay caída en R_2 , y por consiguiente:

$$V_{z2} = V_{CC} - R_2 \cdot I_{C2} \cong V_{CC} = 5.0V$$

Cuando la entrada está en bajo nivel ($V_i = 3.6V$) Q_1 pasará al corte, V_E irá descendiendo de valor (ya que I_{E1} disminuye a medida que Q_1 va pasando al corte), y en el momento que llegue a $3.4V$ Q_2 conducirá, ya que en ese instante;

$V_{BE}(Q_2) = 4.0V - 3.4V = 0.6V$; y ahora se obtienen como salidas:

$$V_{z1} = 5.0V \quad \text{y} \quad V_{z2} = 4.2V$$

El circuito anterior se comporta, en su salida z_1 , como un inversor. Para realizar otras funciones lógicas podemos añadir transistores en paralelo con Q_1 . Así el circuito de la Figura 4.35 realiza la operación lógica **NOR** en la salida **z1** y **OR** en la salida **z2**.

En efecto, basta con que una de las entradas (x o y) esté en el nivel alto, para que el transistor correspondiente (Q_x o Q_y) entre en conducción obteniéndose así la función NOR y OR. El circuito de la Figura 4.35 es el esquema de la compuerta OR/NOR del chip referencia **10102**. Respecto al circuito de la Figura 4.34 tiene añadidos dos etapas adicionales:

1) Alimentación interna que proporciona V_{ref} (tensión en la base de Q_2); se realiza con el transistor Q_4 , que está siempre en conducción y al tener en su base una tensión fija, el voltaje de emisor es también constante del diseño está hecho de tal forma que:

$$V_{ref} = -1.29V$$

2) Los niveles de entrada y salida en el circuito de la Figura 4.xx son distintos, por lo que no podríamos interconectar dos compuertas de la misma familia. Si los niveles de salida se reducen hasta 0.6V no se tendría ese problema. Esta reducción se puede hacer sencillamente ubicando un diodo en cada una de las salidas ($z1$ y $z2$). Esto se logra con los transistores Q_5 y Q_6 de la Figura 4.35, que además hacen que la impedancia de salida sea muy baja, lo que permite interconectar en las salidas muchas compuertas de la misma familia. Las salidas deben conectarse a resistencias y permiten al diseñador elegir valores adecuados para reducir problemas inherentes a la alta velocidad de funcionamiento. Debido a la alta velocidad de funcionamiento las conexiones entre circuitos se comportan como líneas de transmisión, necesitándose cables coaxiales con resistencias terminales si las distancias son mayores que algunos centímetros. El tiempo de propagación de la compuerta ECL (10102) es de 2 ns, y el consumo de potencia de 25 mW.

A continuación se presenta la tabla 4.7 donde se describen las características promediadas de propagación de tiempo y consumo de potencia de las distintas tecnologías y familias de circuitos integrados digitales.

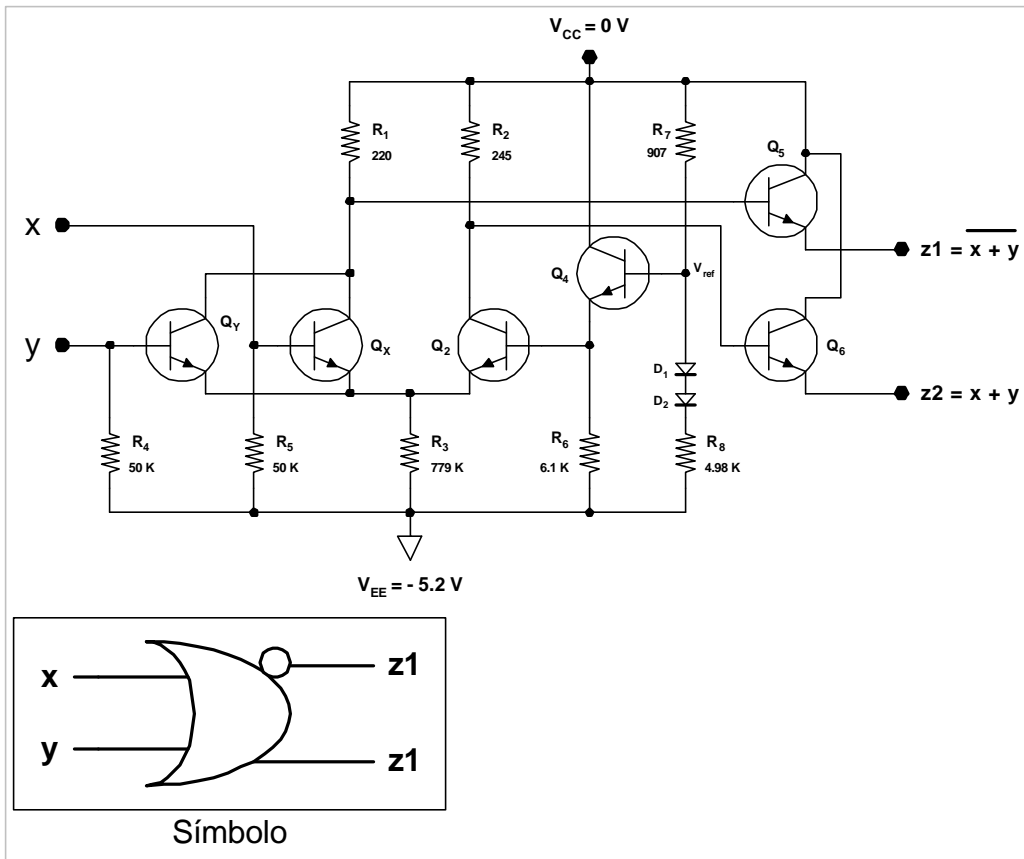


Figura 4.35. Estructura interna y símbolo de la compuerta ECL 10102.

Tecnología	Familia	Serie	Tiempo de Propagación (ns)	Potencia Disipada (mW)	Velocidad x Potencia (pJ)
Bipolar	TTL	LS	9	2	18
		AS	1.7	8	13.6
		ALS	4	1.2	4.8
		F	2.7	4	12
	ECL	10K	2	25	50
		100K	0.8	40	32
MOS	CMOS	4xxxB	100	1	100
		HC	18	0.6	10.8
		HCT	18	0.6	10.8
		AC	5.3	0.8	3.9
		ACT	4.8	0.8	3.6

Tabla 4.7. Características de tiempo y potencia de las familias TTL, ECL y CMOS.

A continuación se describe un ejercicio de compuertas de colector abierto, el cual se muestra en la figura 4.36.

Ejercicio 4.2. Cuatro compuertas AND de colector abierto 74LS09 se conectan en forma alambrada con su resistencia externa (R_{ex}), a un BUS cuya longitud es de 30 centímetros, con una capacitancia uniformemente distribuida de 100 pF/m. En el bus, se conectan 5 entradas de compuertas 74F04 desplegadas a lo largo del mismo, como se muestra en la figura 4.36. A partir de los datos del manual colocados en la tabla, y manteniendo el margen de ruido de 0.7V, calcule el rango de valores de R_{ex} para que el circuito garantice un tiempo de subida, de la onda cuadrada, que esté por debajo de 150 ns. La fuente de alimentación es de 5V y la capacitancia que ofrece cada compuerta 74F04 al BUS, es de 15 pF.

74LS09	74F04
$V_{OH}(\text{mín}) = 2.7 \text{ V}$	$V_{IH}(\text{mín}) = 2.0 \text{ V}$
$V_{OL}(\text{máx}) = 0.5 \text{ V}$	$V_{iL}(\text{máx}) = 0.8 \text{ V}$
$I_{OH}(\text{máx}) = 100 \mu\text{A}$	$I_{iH}(\text{máx}) = 20 \mu\text{A}$
$I_{OL}(\text{máx}) = 8 \text{ mA}$	$I_{iL}(\text{máx}) = -0.6 \text{ mA}$

Solución: Primero se obtiene el rango de la resistencia externa calculando la resistencia mínima y la resistencia máxima sin cargas capacitivas. Luego, se calcula también la resistencia máxima tomando en cuenta el tiempo de subida.

Para V_{OL}

La figura 4.36(b) indica los sentidos de las corrientes en nivel bajo, aquí basta que una de las compuertas de colector abierto 74LS09 se coloque en nivel bajo para que el resto también lo haga. La corriente de fuga de las compuertas que quedan en alto se desprecia.

En el nodo se debe cumplir lo siguiente:

$$I_{OL}(\text{máx}) \geq 5 \cdot I_{iL}(\text{máx}) + I_{ex}$$

$$I_{ex} \leq I_{OL}(\text{máx}) - 5 \cdot I_{iL}(\text{máx}) = 8 \text{ mA} - 5 \cdot (0.6 \text{ mA}) = 5 \text{ mA}$$

$$\frac{V_{CC} - V_{OL}}{R_{ex}} \leq 5 \text{ mA}$$

$$R_{ex} \geq \frac{V_{CC} - V_{OL}(máx)}{I_{ex}} = \frac{5V - 0.5V}{5mA} = 900\Omega$$

Para V_{OH}

La figura 4.36(c) indica los sentidos de las corrientes en nivel alto, todas las compuertas de colector abierto están en alto y consumen, en el peor de los casos, la corriente $I_{iH}(máx)$. Aquí se debe garantizar el margen de ruido:

$$V_{NSH} = V_{OH}(máx) - V_{iH}(máx) = 0.7V.$$

En el nodo se debe cumplir lo siguiente:

$$I_{ex} = 4 \cdot I_{OH}(máx) + 5 \cdot I_{iH}(máx) = 400mA + 100mA = 500mA$$

$$R_{ex} \leq \frac{V_{CC} - V_{OH}(mín)}{I_{ex}} = \frac{5V - 2.7V}{500mA} = 4.6K\Omega$$

Para V_{iH} tomando en consideración el flanco de subida menor a 150 ns.

Se debe suponer que los transistores internos de colector abierto de las compuertas 74LS09 conmutan de nivel bajo a nivel alto sin retardo de tiempo. Cada circuito integrado 7404 posee una capacitancia de 15 pf que se suma a la del BUS; la capacitancia parásita total es igual:

$$C_T = 5 \cdot C_{in} + L \cdot C_{BUS} = 75 pf + (0.30m \cdot 100 pf / m) = 105 pf$$

La ecuación que determina el tiempo de carga de un condensador a través de una resistencia desde un valor inicial V_1 hasta otro valor final de tensión V_2 viene dada por la expresión:

$$t_{carga} = R \cdot C \cdot \ln\left(\frac{V_{CC} - V_1}{V_{CC} - V_2}\right)$$

La tensión V_1 es la condición de nivel bajo $V_{OL}(máx)$ y V_2 debe garantizar la transición del nivel alto en $V_{iH}(mín)$, este tiempo de subida " t_r " es acotado en 150 ns.

$$t_r = R_{ex} \cdot C_T \cdot \ln\left(\frac{V_{CC} - V_{OL}(máx)}{V_{CC} - V_{iH}(mín)}\right) < 150 ns$$

$$R_{ex} < \frac{150 \text{ ns}}{105 \text{ pf} \cdot \ln(1.5)} = \frac{150 \text{ ns}}{42.6 \text{ pf}} = 3.52 \text{ K}\Omega$$

El rango de la resistencia externa debe estar comprendido entre los valores:
 $900 \Omega < R_{ex} < 3.5 \text{ K}\Omega$

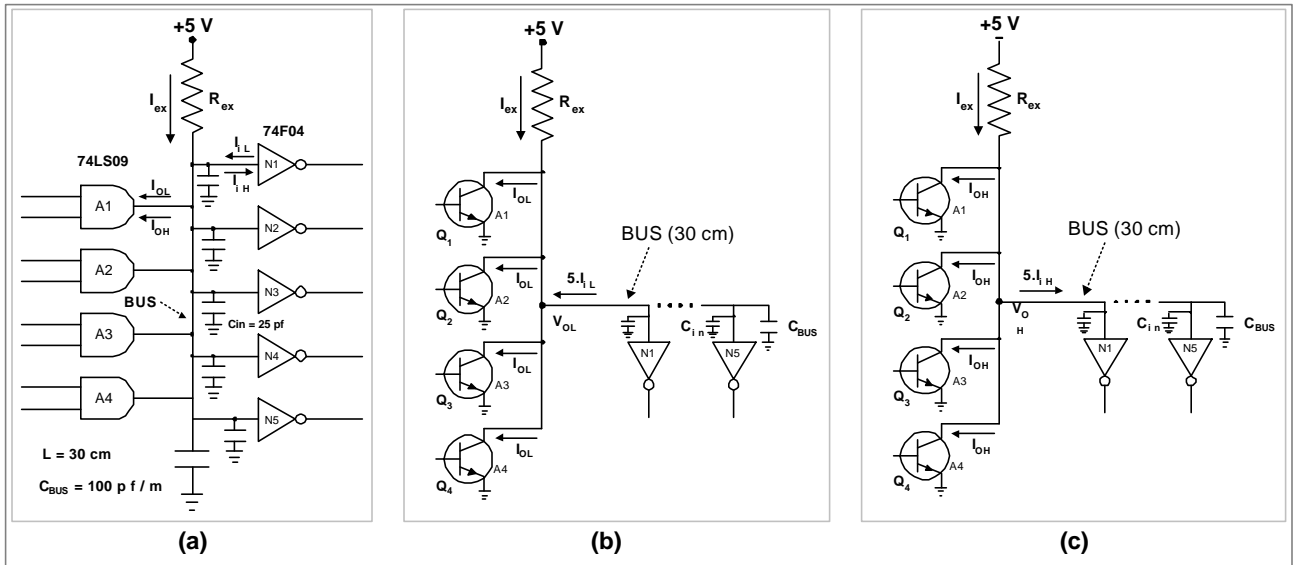


Figura 4.36. (a) Conexión de las compuertas del ejercicio 4.2, (b) Sentido de las corrientes para V_{OL} y (c) Sentido de las corrientes para V_{OH} .

PRÁCTICA DE LABORATORIO #2

TÍTULO: Características eléctricas internas de tensión y corriente de las compuertas lógicas.

INTRODUCCIÓN: En esta práctica se toman muestras de voltaje y corriente a la entrada y salida de las compuertas digitales, con la finalidad de comprobar los parámetros dados por los fabricantes de compuertas digitales y observar el comportamiento de las señales en los respectivos montajes.

OBJETIVO: Al terminar esta práctica el estudiante estará en capacidad de determinar las características de corriente y voltaje de las compuertas TTL y CMOS. Podrá seleccionar el chip adecuado para una aplicación específica según los requerimientos del circuito digital; esto es, saber realizar conexión de compuertas, administrar el consumo de corriente y el manejo de los niveles tensión de las mismas.

PRE-LABORATORIO: Investigar los siguientes tópicos.

- Circuito interno de las compuertas AND, OR, NOT, NOR, NAND, XOR.
- Investigar las características técnicas de las familias y series de los circuitos integrados;

TTL Series: Estándar, 74LS, 74ALS, 74F. **CMOS Series:** Estándar, 74HC, 74C, 74HCT.

- Fundamentos del TRANSISTOR BJT y MOSFET usado como switch.
- FAN out, Margen de Ruido.

El significado de: **Voh, Vol, Vih, Vil, Ioh, Iol, Iih, Iil, Iss, Isc.**

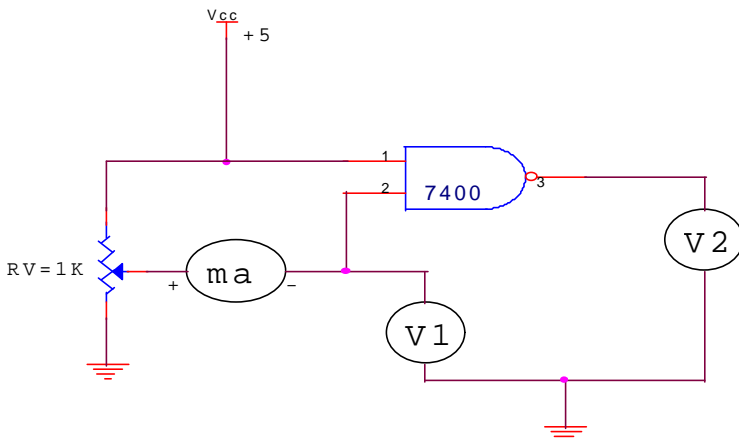
MATERIALES Y EQUIPOS NECESARIOS:

- 7400, 74LS00, 74HC00.
- Potenciómetros de 1k, 10k, 50k, 100k.
- Resistencias Variadas (10 ohmios en adelante).

- Fuente de 5 voltios / 1 Amp. , 2 Multímetros con micro amperímetro.
- Generador de Funciones. Software de simulación.

DESARROLLO:

1. Simular, en forma analógica, las compuertas OR, AND, NOT hechas con transistores. Obtener la onda de salida para todas las combinaciones de entradas de cada compuerta.
2. Montar el circuito, tomar varias medidas de corriente y tensión con el fin de graficar las curvas de transferencia V_2/V_1 , I_i/V_1 , I_i/V_2 . Tomar, en la entrada de la compuerta, la mayor cantidad de medidas en el rango de (0.8 ~ 2.0) Voltios. Cambiar la compuerta estándar por **LS** y **HC**; si es necesario, cambie el potenciómetro.

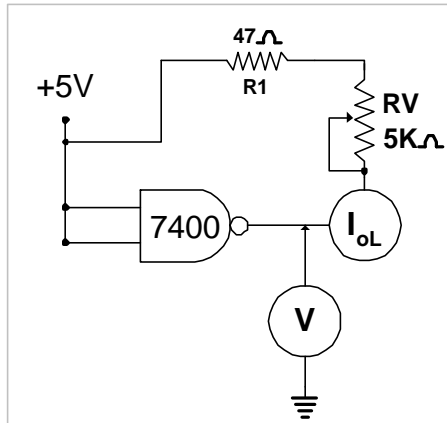


I_i																				
V_1																				
V_2																				

En caso de tomar medidas con la compuerta 74HC00, se deben hacer para tensiones de alimentación de 3.0 V y 5.5 V respectivamente. Además, hay que tener precaución de no tocar, con los dedos, los pines del chip.

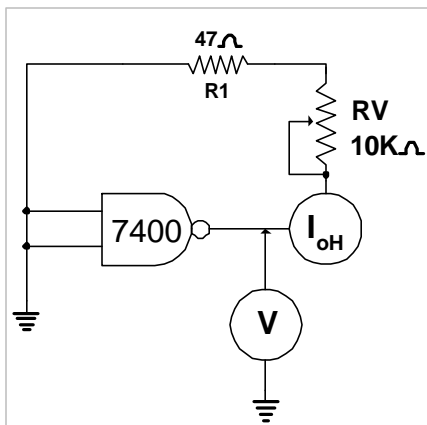
La corriente de entrada de la 74HC00 es muy baja y debe tomarse con un microamperímetro.

3. Tomar varias medidas de corriente hasta obtener la máxima que pueda entregar la compuerta en la salida.



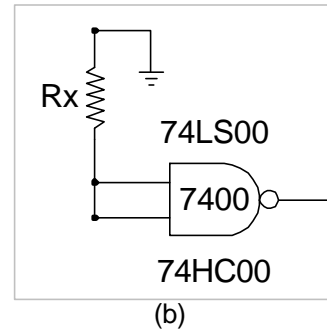
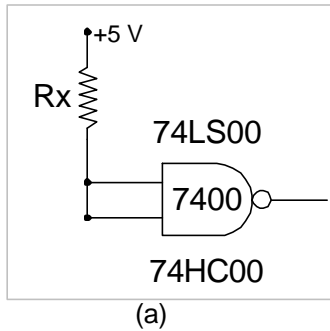
I_{OL}														
V														

4. Tomar varias medidas de corriente hasta obtener la máxima que pueda entregar la compuerta en la salida.



I_{OH}														
V														

5. Calcular y realizar el montaje de la resistencia R_x . Obtenga los 3 rangos de la resistencia con los dos niveles lógicos de entrada. Utilice, también, una sola entrada.



POST-LABORATORIO.

1. ¿ Qué se debe comprobar en el montaje tres y cuatro?. Explique que nombre recibe.
2. ¿Cuál es la mínima corriente que debe suministrar la fuente de alimentación en un circuito formado por dos chips 74F00 y tres 74LS32?. Explique y demuestre.
3. Realizar las gráficas ($V_E \times V_S$, $I_E \times V_S$, $V_E \times I_E$) y conclusiones del primer montaje.
4. Calcule el fan-out de las series estándar, LS y HC.
5. ¿Cuál es la resistencia máxima y mínima (Rango de R_x) del circuito 5?. Demostrarlo.
6. Determine la impedancia estática con respecto a tierra, de entrada y salida, Z_E y Z_S , de las compuertas LS y HC.

BIBLIOGRAFÍA.

- NEAMEN A, Donald. (1999). Análisis y diseño de circuitos electrónicos. Tomo II. México: McGraw Hill. S/f. p.1176. "Electronic circuit analysis and design". Traducido por: Felipe Castro Pérez.
- TOCCI, Ronald. (1995). Sistemas digitales principios y aplicaciones. México: Prentice Hall. Quinta edición. p.823. "Digital systems principles and applications". Traducido por: Edmundo G. Urbina M.
- WARKELY, John F. (1997). Diseño digital principios y prácticas. México: Prentice Hall. S/f. p.743. "Digital design principles and practices". Traducido por: Gutiérrez R. Raymundo H.

MANUALES.

- NATIONAL SEMICONDUCTOR. (1981). Manual TTL y CMOS.
- MOTOROLA Inc. (1992). Fast and LS TTL. (DL121/D REV. 5).

PRÁCTICA DE LABORATORIO #3

TÍTULO: Características eléctricas internas transitorias y tiempos de retardo de las compuertas lógicas.

INTRODUCCIÓN: En esta práctica se realizan medidas de señales de entrada y salida de las compuertas digitales, con la finalidad de comprobar los parámetros de retardos, tiempos de propagación y análisis transitorio de las distintas tecnologías de compuertas digitales.

OBJETIVO: Al terminar esta práctica el estudiante estará en capacidad de determinar las características de retardo de tiempo y deformación de señales en las compuertas TTL y CMOS. Podrá seleccionar el chip adecuado para una aplicación específica según los requerimientos del circuito digital; esto es, saber realizar conexión de compuertas, determinar la frecuencia máxima de trabajo y los tiempos de propagación de las compuertas digitales.

PRE-LABORATORIO: Investigar los siguientes tópicos.

- Frecuencia de corte, tiempo de propagación, t_{pHL} , t_{pLH} .
- Investigar las características del producto Velocidad x Potencia de las distintas familias de circuitos digitales: F, LS, HC, HCT, AC, ACT, 10K, 100K.
- FAN OUT dinámico de los dispositivos CMOS, Margen de Ruido.
- Margen de ruido en CMOS, Capacitancia e inductancia parásita en los circuitos digitales, Capacitancia de carga.
- Picos de corriente, tercer estado.
- Compuertas de colector abierto y Smith Trigeer.

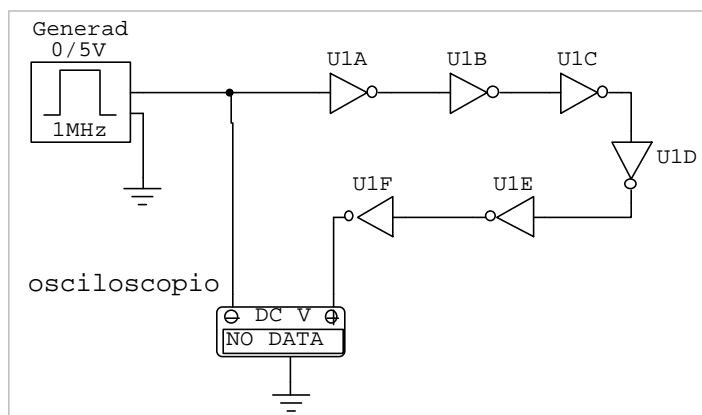
El significado de: **tp**, **tphl**, **tplh**, **tr**, **tf**, **loz**, **Vth⁺**, **VTh⁻**.

MATERIALES Y EQUIPOS NECESARIOS:

- 74LS04, 74HC04, 74LS06, 74LS00.
- Potenciómetros de 1k, 10k, 50k, 100k.
- Resistencias varias de acuerdo a los cálculos realizados.
- Fuente de 5 voltios / 1 Amp, 1 Osciloscopio de 60 MHz doble trazo con dos puntas.
- Generador de Funciones mayor o igual a 2 MHz. Software de simulación.

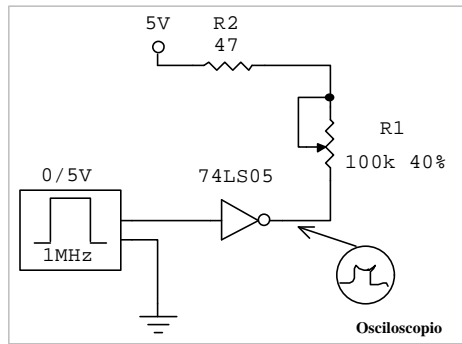
DESARROLLO:

1. Realizar, con el chip 74x04, el siguiente montaje con la finalidad de obtener el **tp** de una compuerta inversora. Mida el defase entre la señal de entrada y la señal de salida del montaje 1; las compuertas están conectadas en serie (74LS04 y 74HC04) intercambie las compuertas y tome lectura en el Osciloscopio. Tome nota con el generador en 100 KHz, 500 KHz, 1 MHz y 2 MHz. Las medidas deber ser realizadas aproximadamente en la zona del 50% de los flancos de la señal. Desarrolle algún método para hallar el **tp** de una sola compuerta.



Montaje 1.

2. Montar el circuito, utilizando una compuerta de colector abierto (74LS05 o 74LS06). Variar el potenciómetro entre la $R_{\text{máx}}$ y R_{min} , también varíe la frecuencia del generador entre 1 KHz y 2 MHz. Observe el comportamiento, forma de la señal y haga su análisis.

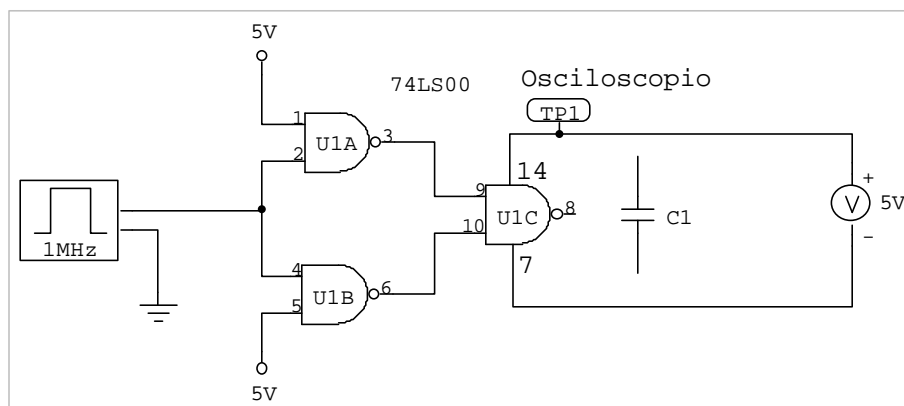


Montaje 2.

3. Medir, con el osciloscopio, los picos de corriente que se forman en el pin de alimentación del circuito integrado. Colocar el generador a una frecuencia mayor o igual que 100 KHz. Colocar un condensador (C1) con capacidad entre $0.01\mu\text{F}$ y $0.1\mu\text{F}$ en los terminales de alimentación:

- I) Colocar en los terminales de la fuente.
- II) Colocar en los terminales del circuito integrado.
- III) Colocar en los terminales del Protoboard.

Cambiar el chip 74LS00 por 74HC00, observe los cambios con las distintas series y haga su análisis.



Montaje 3.

POST-LABORATORIO.

1. Realice las conclusiones de los resultados obtenidos en el montaje 1.
2. ¿ Qué se debe comprobar en el montaje tres?. Explique que nombre recibe el condensador y ¿por qué?.
3. ¿ Realice un modelo transitorio aproximado del circuito del montaje tres?.
4. Saque las conclusiones del montaje 2; indique cual debería ser el valor optimo de resistencia externa.
5. Haga una tabla con las principales características de las series LS y HC.

BIBLIOGRAFIA.

- MANDADO, Enrique. (1987). Sistemas electrónicos digitales. Barcelona (España): Marcombo Boixareu Editores. Sexta edición. p.705.
- MANO, Morris. KIME, Charles. (1998). Fundamentos de diseño lógico y computadoras. México: Prentice Hall. Primera edición en español. P.604. "Logic and computer design fundamentals". Traducido por: Teresa Sanz Falcón.
- NEAMEN A, Donald. (1999). Análisis y diseño de circuitos electrónicos. Tomo II. México: McGraw Hill. S/f. p.1176. "Electronic circuit analysis and design". Traducido por: Felipe Castro Pérez.
- NELSON, V. NAGLE, H. CARROLL, B. IRWIN, J. (1996). Análisis y diseño de circuitos lógicos digitales. México: Prentice Hall. Primera edición. p.842. "Digital logic circuit analysis and design". Traducido por: Oscar A. Palmas V.
- TOCCI, Ronald. (1995). Sistemas digitales principios y aplicaciones. México: Prentice Hall. Quinta edición. p.823. "Digital systems principles and applications". Traducido por: Edmundo G. Urbina M.
- WARKELY, John F. (1997). Diseño digital principios y prácticas. México: Prentice Hall. S/f. p.743. "Digital design principles and practices". Traducido por: Gutiérrez R. Raymundo H.

MANUALES.

- NATIONAL SEMICONDUCTOR. (1981). Manual TTL y CMOS.
- MOTOROLA Inc. (1992). Fast and LS TTL. (DL121/D REV. 5).

CAPITULO V.

5 CIRCUITOS DIGITALES COMBINACIONALES MSI.

Los circuitos digitales MSI (mediana escala de integración) son bloques completos que ejecutan una función específica. Están hechos internamente con muchas compuertas básicas y universales con un rango aproximado de 12 a 99 compuertas discretas, obteniendo así, el beneficio de ahorro de costo y espacio a la hora de hacer un diseño digital. Dentro de estos módulos se pueden mencionar: Decodificadores, Codificadores, Multiplexores, Sumadores, Comparadores, Generadores de Paridad.

5.1. Decodificadores.

Son circuitos integrados digitales combinacionales que poseen n líneas de entrada y, a lo sumo, 2^n líneas de salida, además de poseer una o más líneas de entrada para la habilitación del bloque; las cuales puede desactivar todas las líneas de salida. La característica fundamental de este circuito es que solamente activa una línea de salida, por cada combinación binaria en las líneas de entrada. También pueden ser especificados atendiendo a la relación: **1 de m**; donde m es la cantidad de salidas.

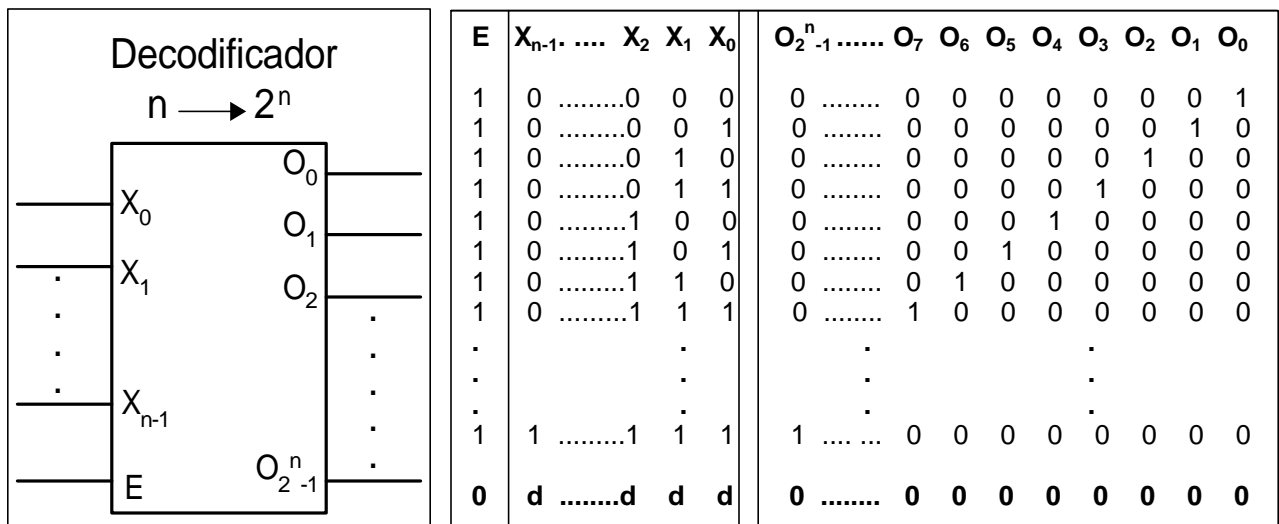


Figura 5.1. Decodificador genérico de n a 2^n y su respectiva tabla de funcionamiento.

En la **figura 5.1** se puede observar un decodificador de n a 2^n líneas con su respectiva tabla de funcionamiento. Cuando la línea de habilitación **E** (Enable) se encuentra en nivel lógico uno se activará solamente una salida ($O_{2^{n-1}}, \dots, O_7, O_6, O_5, O_4, O_3, O_2, O_1, O_0$) y esta corresponderá a la combinación en binario que tengan las líneas de entrada ($X_{n-1}, \dots, X_2, X_1, X_0$). Si la habilitación **E** pasa a un nivel bajo, todas las salidas se pondrán a cero lógico (se desactivan) sin importar el valor de las entradas; esto lo indican los términos indiferentes "d" de la tabla.

Los decodificadores también pueden ser diseñados con compuertas. Sin embargo, estos son construidos, por ejemplo, en caso tal de no poder adquirir el circuito integrado en un solo chip. La **figura 5.2** se muestra el diseño de un decodificador con compuertas básicas: posee tres entradas, ocho salidas y una línea de habilitación **E**. Las funciones que generan este circuito son extraídas de la tabla de la verdad:

$$\begin{aligned}
 O_0 &= E \bar{X}_2 \bar{X}_1 \bar{X}_0 & O_1 &= E \bar{X}_2 \bar{X}_1 X_0 & O_2 &= E \bar{X}_2 X_1 \bar{X}_0 & O_3 &= E \bar{X}_2 X_1 X_0 \\
 O_4 &= E X_2 \bar{X}_1 \bar{X}_0 & O_5 &= E X_2 \bar{X}_1 X_0 & O_6 &= E X_2 X_1 \bar{X}_0 & O_7 &= E X_2 X_1 X_0
 \end{aligned}$$

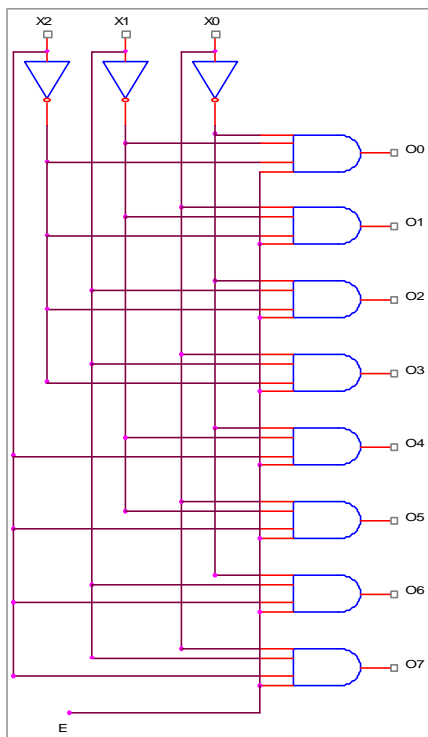


Tabla de la verdad

E	X ₂	X ₁	X ₀	O ₇	O ₆	O ₅	O ₄	O ₃	O ₂	O ₁	O ₀
1	0	0	0	0	0	0	0	0	0	0	1
1	0	0	1	0	0	0	0	0	0	1	0
1	0	1	0	0	0	0	0	0	1	0	0
1	0	1	1	0	0	0	0	1	0	0	0
1	1	0	0	0	0	0	1	0	0	0	0
1	1	0	1	0	0	1	0	0	0	0	0
1	1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	1	0	0	0	0	0	0	0
0	d	d	d	0	0	0	0	0	0	0	0

Figura 5.2. Decodificador de compuertas y la tabla de funcionamiento.

5.1.1 Salidas y entradas activas en nivel bajo.

Los decodificadores, con líneas de salidas activas en alto, tienen un nivel lógico alto en la salida activa y las restantes, que están desactivadas, poseen un nivel lógico bajo; esto se puede apreciar en la figura 5.2. Sin embargo, existe la forma contraria, que consiste en activar las salidas con los niveles bajos y desactivarlas con el nivel alto. También se pueden presentar estos mismos casos para las líneas de entradas y líneas de habilitación. La figura 5.3 es un decodificador con sus cuatro líneas de salida activas en bajo y una sola entrada de habilitación también activa en nivel bajo.

$$O_0 = E + X_1 + X_0 = \overline{\overline{E} \cdot \overline{X_1} \cdot \overline{X_0}}$$

$$O_1 = E + X_1 + \overline{X_0} = \overline{\overline{E} \cdot \overline{X_1} \cdot X_0}$$

$$O_2 = E + \overline{X_1} + X_0 = \overline{\overline{E} \cdot X_1 \cdot \overline{X_0}}$$

$$O_3 = E + \overline{X_1} + \overline{X_0} = \overline{\overline{E} \cdot X_1 \cdot X_0}$$

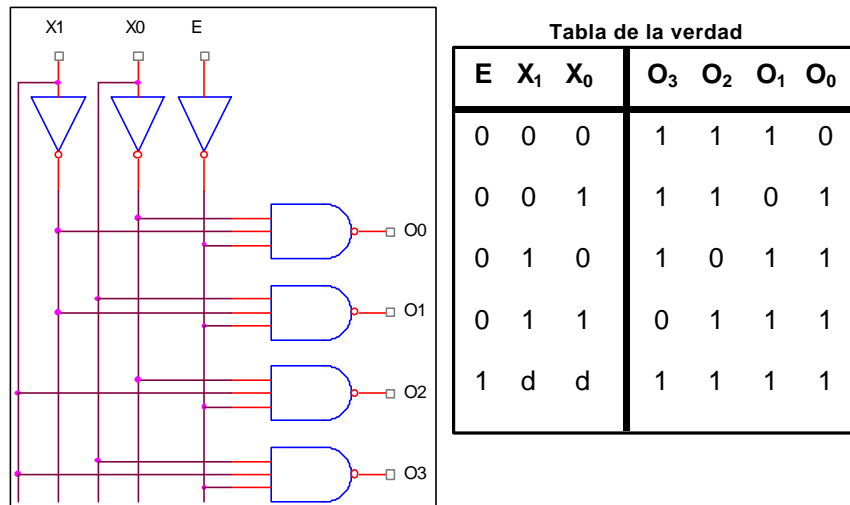


Figura 5.3. Decodificador con salidas y habilitación “E” activas en bajo.

Ejercicio 5.1: Diseñar un decodificador de 3 entradas y 8 salidas con enable activo en nivel bajo.

5.1.2 Decodificadores integrados MSI.

Los decodificadores se consiguen en el mercado en pastillas de circuitos integrados con tecnología TTL y CMOS. A continuación se nombran algunos:

N° Decodificador	Líneas de entrada.	Líneas de salida	Habilitaciones
74139	(Doble) c/u 2 entradas activas en alto.	(Doble) c/u 4 salidas activas en bajo.	(Doble) c/u 2 líneas activas en bajo
74138	3 entradas activas en alto.	8 salidas activas en bajo.	Dos activas en bajo y una activa en alto.
74154	4 entradas activas en alto.	16 salidas activas en bajo.	Dos activas en bajo

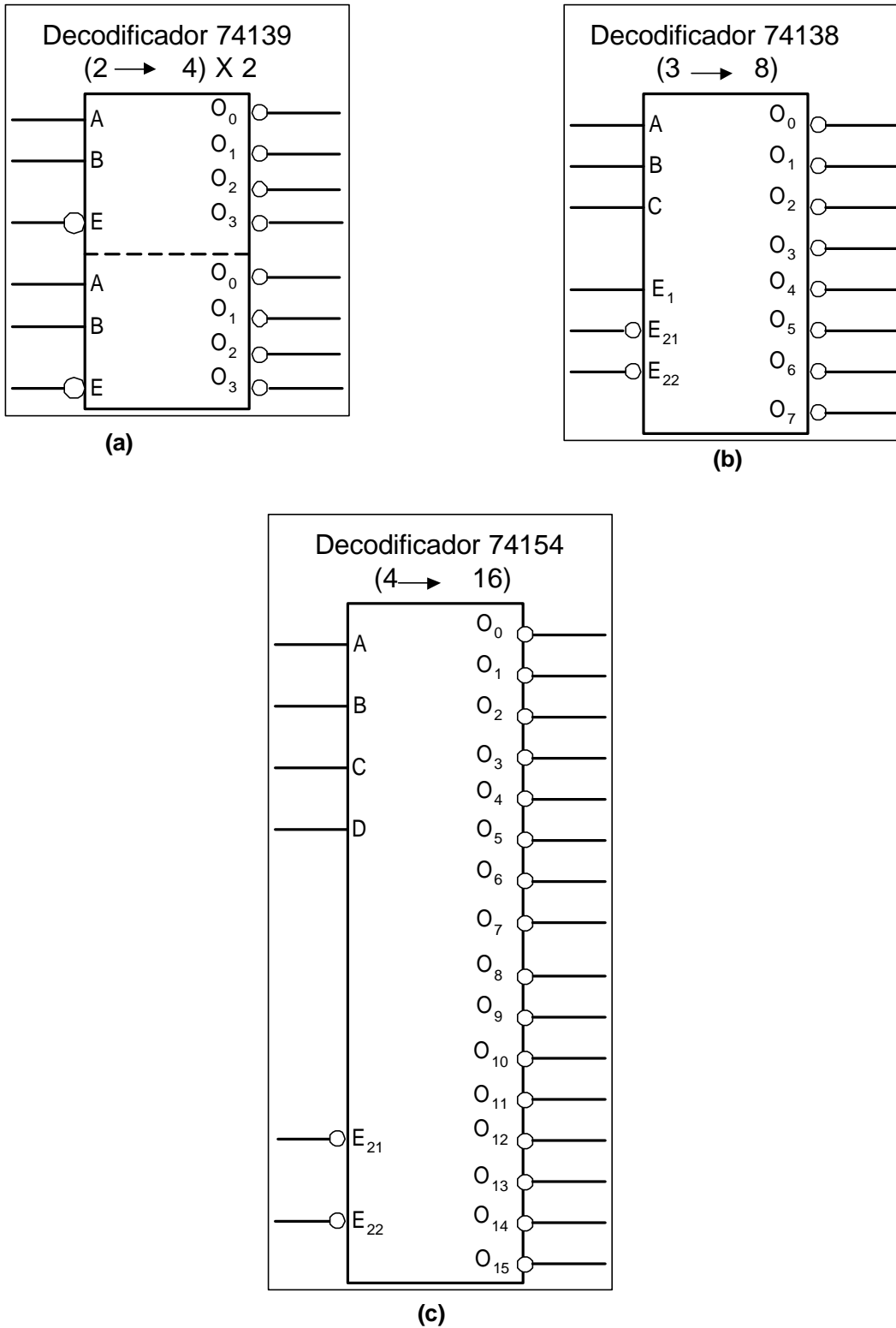


Figura 5.4. Decodificadores MSI estándar (a) 74139, (b) 74138, (c) 74154.

Tabla de la verdad del 74139.

E	B	A	O ₃	O ₂	O ₁	O ₀
0	0	0	1	1	1	0
0	0	1	1	1	0	1
0	1	0	1	0	1	1
0	1	1	0	1	1	1
1	d	d	1	1	1	1

(a)

Tabla de la verdad del 74138.

E ₂₁	E ₂₂	E ₁	C	B	A	O ₇	O ₆	O ₅	O ₄	O ₃	O ₂	O ₁	O ₀
0	0	1	0	0	0	1	1	1	1	1	1	1	0
0	0	1	0	0	1	1	1	1	1	1	1	0	1
0	0	1	0	1	0	1	1	1	1	1	0	1	1
0	0	1	0	1	1	1	1	1	1	0	1	1	1
0	0	1	1	0	0	1	1	1	0	1	1	1	1
0	0	1	1	0	1	1	1	0	1	1	1	1	1
0	0	1	1	1	0	1	0	1	1	1	1	1	1
0	0	1	1	1	1	0	1	1	1	1	1	1	1
1	d	d	d	d	d	1	1	1	1	1	1	1	1
d	1	d	d	d	d	1	1	1	1	1	1	1	1
d	d	0	d	d	d	1	1	1	1	1	1	1	1

(b)

Tabla de la verdad del 74154.

E ₂₁	E ₂₂	D	C	B	A	O ₁₅	O ₁₄	O ₁₃	O ₁₂	O ₁₁	O ₁₀	O ₉	O ₈	O ₇	O ₆	O ₅	O ₄	O ₃	O ₂	O ₁	O ₀
0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1
0	0	0	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1
0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1
0	0	0	1	0	0	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1
0	0	0	1	0	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1
0	0	0	1	1	0	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1
0	0	0	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1
0	0	1	0	0	0	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1
0	0	1	0	0	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1
0	0	1	0	1	0	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1
0	0	1	0	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1
0	0	1	1	0	0	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1
0	0	1	1	1	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0	0	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	d	d	d	d	d	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
d	1	d	d	d	d	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

(c)

Figura 5.5. Tablas de los decodificadores MSI estándar (a) 74139, (b) 74138, (c) 74154.

En la **figura 5.4 y 5.5** se puede observar el comportamiento de las entradas, salidas y líneas de habilitación de los decodificadores 74139, 74138 y 74154; que son ampliamente utilizados para aplicaciones digitales.

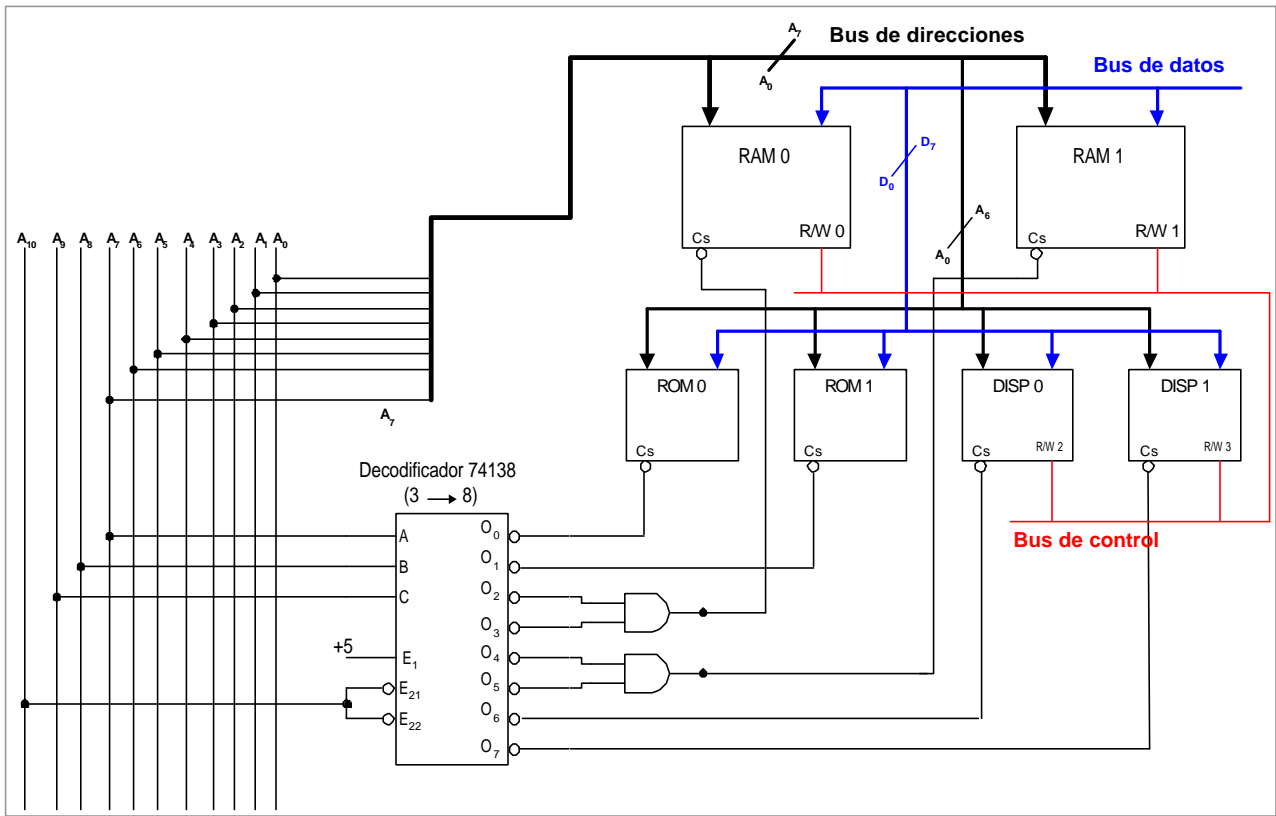
5.1.3 Aplicaciones de los decodificadores.

Las aplicaciones de los circuitos decodificadores son diversas, entre las cuales se pueden citar: los decodificadores de direcciones, decodificador de dispositivos de Entrada/Salida en un sistema de desarrollo o computadora, convertidores de código, generador de funciones de conmutación, etc.

5.1.3.1 Decodificador de direcciones.

El hardware de un computador está constituido por tres **buses** principales: el **bus de datos**, el **bus de direcciones** y el **bus de control**. Cada uno de ellos está formado por varias líneas de señal binaria que determinan la capacidad de memoria del sistema. En los inicios de la computación el bus de datos, del cual depende el tamaño del número y/o palabra a procesar, era de cuatro líneas (cuatro bits), en la actualidad este puede llegar a tener hasta 128 bits. El bus de direcciones determina la capacidad en localidades de memoria de un computador llegando a tener varios GIGABYTES (GB) de localidades. El bus de control sincroniza en el tiempo las operaciones de **Lectura/Escritura** del sistema, por ejemplo, con el bus de direcciones se selecciona una localidad de memoria para escribir el contenido de esta en un chip o dispositivo específico, se deben activar en sincronismo, las señales de selección de los circuitos integrados correspondientes a la operación, e inmediatamente colocar el dato en el bus. Todos estos buses están determinados y controlados por un circuito integrado llamado microprocesador.

El papel que juega aquí el decodificador es de vital importancia porque permite seleccionar el chip o dispositivo de Lectura/Escritura para transferencia de información. En la **figura 5.6** se observa un diagrama en bloques donde el **74138** administra la selección de dos **chips de memoria RAM**, dos dispositivos de entrada/salida y dos **chips de memoria ROM**. La señal **R/W** es el control de lectura y escritura en la RAM y los dispositivos.



A_{10}	A_9	A_8	A_7	A_6	A_0	A_{10}	A_0
				Rango de memoria (bytes)		Rango de memoria (bytes)	
				DECIMAL		HEXADECIMAL	
0	0	0	0	ROM 0		000H	----- 03FH
0	0	0	1	ROM 1		040H	----- 07FH
0	0	1	0	RAM 0		080H	----- 0BFH
0	0	1	1	RAM 0		0C0H	----- 0FFH
0	1	0	0	RAM 1		100H	----- 13FH
0	1	0	1	RAM 1		140H	----- 17FH
0	1	1	0	DISP 0		180H	----- 1BFH
0	1	1	1	DISP 1		1C0H	----- 1FFH
1	x	x	x	Deshabilitar		200H

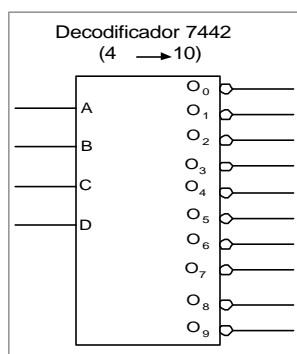
Figura 5.6. Circuito integrado 74138 como decodificador de direcciones y dispositivos.

Los chips de memoria tienen la siguiente característica: Los IC ROM son de 128 bytes, Los IC RAM de 256 bytes y cada dispositivo de E/S tiene 128 bytes. El sistema posee un bloque continuo de memoria de 1024 bytes y debe desactivarse a partir de esta dirección. El bus de direcciones tiene un tamaño de 11 líneas ($A_{10} \dots A_0$) y el bus de datos es de ocho bits ($D_7 \dots D_0$); las líneas de control son manejadas por el microprocesador del sistema.

Ejercicio 5.2. Diseñar un bloque continuo de memoria de 16 kilobytes (Kb) comenzando con 8 Kb de memoria ROM a partir de la dirección cero. Cada chip de memoria ROM es de 2Kb y los de RAM tienen una capacidad de 4 Kb. Cada circuito integrado tiene un **Cs** activo en bajo. Seleccionar el decodificador más adecuado.

5.1.3.2 Circuitos MSI convertidores de código.

Los decodificadores con n líneas de entrada y 2^n líneas de salida son convertidores de binario a códigos: Octal (8 salidas), Hexadecimal (16 salidas), etc. En estos circuitos solamente hay una salida activa en cada combinación binaria de entrada. Sin embargo, cuando el código de salida no es múltiplo de 2^n se necesita un número menor de salidas por cada combinación binaria en la entrada. Esto significa que si m es el número de salidas y n las entradas se debe cumplir que; $n < m \leq 2^n$. Por ejemplo, en la **figura 5.7** se puede observar el convertidor BCD a DECIMAL 7442 que posee 10 líneas de salida activas en bajo y 4 líneas de entrada BCD.



A	B	C	D	O ₀	O ₁	O ₂	O ₃	O ₄	O ₅	O ₆	O ₇	O ₈	O ₉
0	0	0	0	0	1	1	1	1	1	1	1	1	1
0	0	0	1	1	0	1	1	1	1	1	1	1	1
0	0	1	0	1	1	0	1	1	1	1	1	1	1
0	0	1	1	1	1	1	0	1	1	1	1	1	1
0	1	0	0	1	1	1	1	0	1	1	1	1	1
0	1	0	1	1	1	1	1	1	0	1	1	1	1
0	1	1	0	1	1	1	1	1	1	0	1	1	1
0	1	1	1	1	1	1	1	1	1	1	0	1	1
1	0	0	0	1	1	1	1	1	1	1	1	0	1
1	0	0	1	1	1	1	1	1	1	1	1	1	0
1	0	1	x	1	1	1	1	1	1	1	1	1	1
1	1	x	x	1	1	1	1	1	1	1	1	1	1

Figura 5.7. Decodificador 7442 (BCD - DECIMAL) donde se cumple $n < m \leq 2^n$.

Existen otros convertidores de código que pueden controlar el encendido de indicadores o visualizadores (Display_s) llamados 7 segmentos, están fabricados con 7 diodos Leds, 7 lámparas Nixie o Cristal líquido (LCD). La característica de éstos es la de tener más de una salida activa, por cada combinación de entrada. En la **figura 5.8a** se puede apreciar un Display 7 segmentos conjuntamente con el manejador 7448, en este caso, el display es un arreglo de 7 diodos con el cátodo común. Por lo tanto, para encender cada led es necesario que el convertidor tenga las salidas activas en nivel alto. Sin embargo, existen displays 7 segmentos que tienen el ánodo común (**figura 5.8b**). Este tipo de visualizador debe ser manejado con circuitos integrados que tengan las salidas activas en bajo como lo son: 7446, 7447. En la **figura 5.9** se describen las características de algunos convertidores de código.

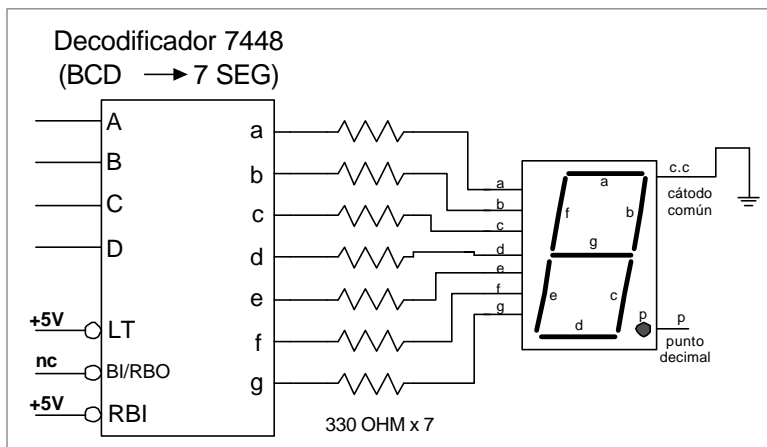


Figura 5.8a. Decodificador BCD - 7 segmentos 7448 y visualizador cátodo común.

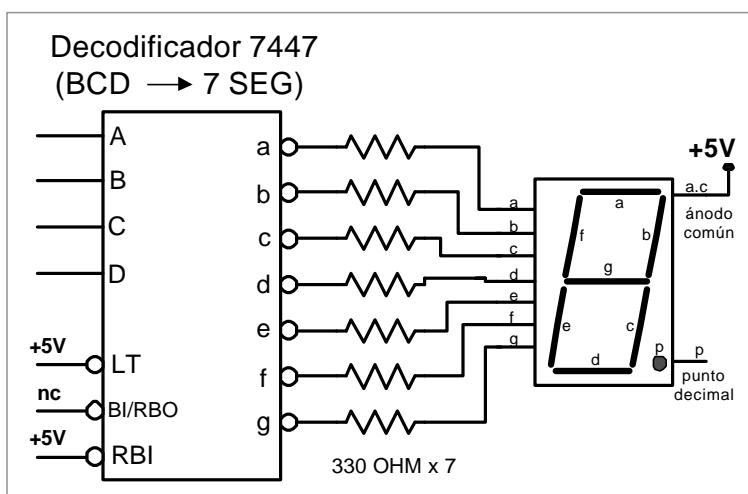


Figura 5.8b. Decodificador BCD - 7 segmentos 7447 y visualizador ánodo común.

LT	RBI	BI/RBO	D	C	B	A	a	b	c	d	e	f	g	BI/RBO	Visualizador
1	1	Nc	0	0	0	0	1	1	1	1	1	1	0	Nc	0
1	1	Nc	0	0	0	1	0	1	1	0	0	0	0	Nc	1
1	1	Nc	0	0	1	0	1	1	0	1	1	0	1	Nc	2
1	1	Nc	0	0	1	1	1	1	1	0	0	0	1	Nc	3
1	1	Nc	0	1	0	0	0	1	1	0	0	1	1	Nc	4
1	1	Nc	0	1	0	1	1	0	1	1	0	1	1	Nc	5
1	1	Nc	0	1	1	0	0	0	1	1	1	1	1	Nc	6
1	1	Nc	0	1	1	1	1	1	1	0	0	0	0	Nc	7
1	1	Nc	1	0	0	0	1	1	1	1	1	1	1	Nc	8
1	1	Nc	1	0	0	1	1	1	1	0	0	1	1	Nc	9
1	1	Nc	1	0	1	0	0	0	0	1	1	0	1	Nc	A
1	1	Nc	1	0	1	1	0	0	1	1	0	0	1	Nc	B
1	1	Nc	1	1	0	0	0	1	0	0	0	1	1	Nc	C
1	1	Nc	1	1	0	1	1	0	0	1	0	1	1	Nc	D
1	1	Nc	1	1	1	0	0	0	0	1	1	1	1	Nc	E
1	1	Nc	1	1	1	1	0	0	0	0	0	0	0	Nc	No prende
0	x	x	x	x	x	x	1	1	1	1	1	1	1	Nc	8
1	0		EN ESTA CONDICIÓN CUANDO LA ENTRADA BCD ES CERO (0 0 0 0) ENTONCES TODAS LAS SALIDAS SE DESACTIVAN.				EL DISPLAY SE APAGA SOLO CUANDO APARECE UN CERO EN LA ENTRADA DEL CONVERTIDOR. SIN EMBARGO, TODOS LOS DEMÁS DIGITOS SE VISUALIZAN. POR SUPUESTO, DESCARTANDO EL QUINCE QUE NUNCA SE VE.							ESTA SALIDA PASA DE UNO A CERO. SI HAY (0000) EN LA ENTRADA.	RBO SE MANTIENE EN UNO SI LA ENTRADA BCD ES DIFERENTE DE (0 0 0 0).
1	x	0	x x x x				0 0 0 0 0 0 0							AQUI BI/RBO ACTÚA COMO ENTRADA "Blanking Input"	SE BORRA EL DISPLAY SIN IMPORTAR EL DATO DE ENTRADA.

Tabla 5.1. Descripción de funcionamiento del 7448 y 7449 con salidas/ activas en alto.

La **tabla 5.1** muestra todas las combinaciones que tiene el circuito integrado decodificador 7448 y 7449. Se pueden observar tres líneas de control (LT, RBI, BI/RBO) activas en nivel bajo, cuatro líneas de entrada (D,C,B,A) activas en alto y las salidas (a, b, c, d, e, f, g) también activas en alto, que sirven para alimentar un display de siete segmentos. Las líneas de control funcionan de la siguiente forma:

LT (Lamp Test): Cuando esta línea de control se pone a cero, todas las salidas se activan y no reconoce ningún dato de entrada; el número que se visualiza es el ocho. Esta línea sirve para realizar pruebas de los segmentos y/o las salidas del convertidor.

RBI (Riple Blanking Input): Esta línea de control funciona con un nivel bajo y desactiva todas las salidas cuando hay cero en la entrada BCD, de este modo, se apaga el display solo con el cero. De esta misma forma, la línea de entrada/salida BI/RBO trabaja como salida y se pone en nivel bajo solamente cuando hay cero en la entrada del decodificador. Si RBI es alto se observaran todos los dígitos, con excepción del quince que nunca visualiza símbolo alguno.

BI/RBO (Blanking Input / Riple Blanking Output): Tiene una función como entrada y otra como salida. Al activarse como entrada se apaga todo el display sin importar el dato que se encuentre en la entrada del convertidor. La función de salida se describió anteriormente. Si la línea RBI ha sido activada entonces el pin BI/RBO pasará a un nivel bajo solo cuando hay cero en la entrada del convertidor. De lo contrario, siempre se mantendrá en nivel alto.

Los decodificadores con salidas activas en bajo 7446 y 7447 se rigen también por la tabla 5.1 pero, se debe invertir la condición para las salidas de los mismos. La familia CMOS también posee decodificadores de este propósito como lo son el 4543B y 4511B.

Ejercicio 5.3: Una aplicación ampliamente utilizada es apagar los dígitos de la izquierda, en una cantidad entera, cuando estos son ceros. Para ello se debe activar la función de los pines **RBI** y/o **BI/RBO**. En la **figura 5.10** se pueden observar las conexiones de los tres displays 7 segmentos cableados para que realicen esta aplicación.

Solución: En el convertidor, del display más significativo, se debe conectar el RBI a tierra para que no se visualice el cero; y su salida BI/RBO cablearla con el RBI del siguiente convertidor y así sucesivamente hasta llegar al display menos significativo, el cual debe señalar todos los diez dígitos. Por lo tanto hay que dejar este último RBI en nivel alto, ver **figura 5.10**.

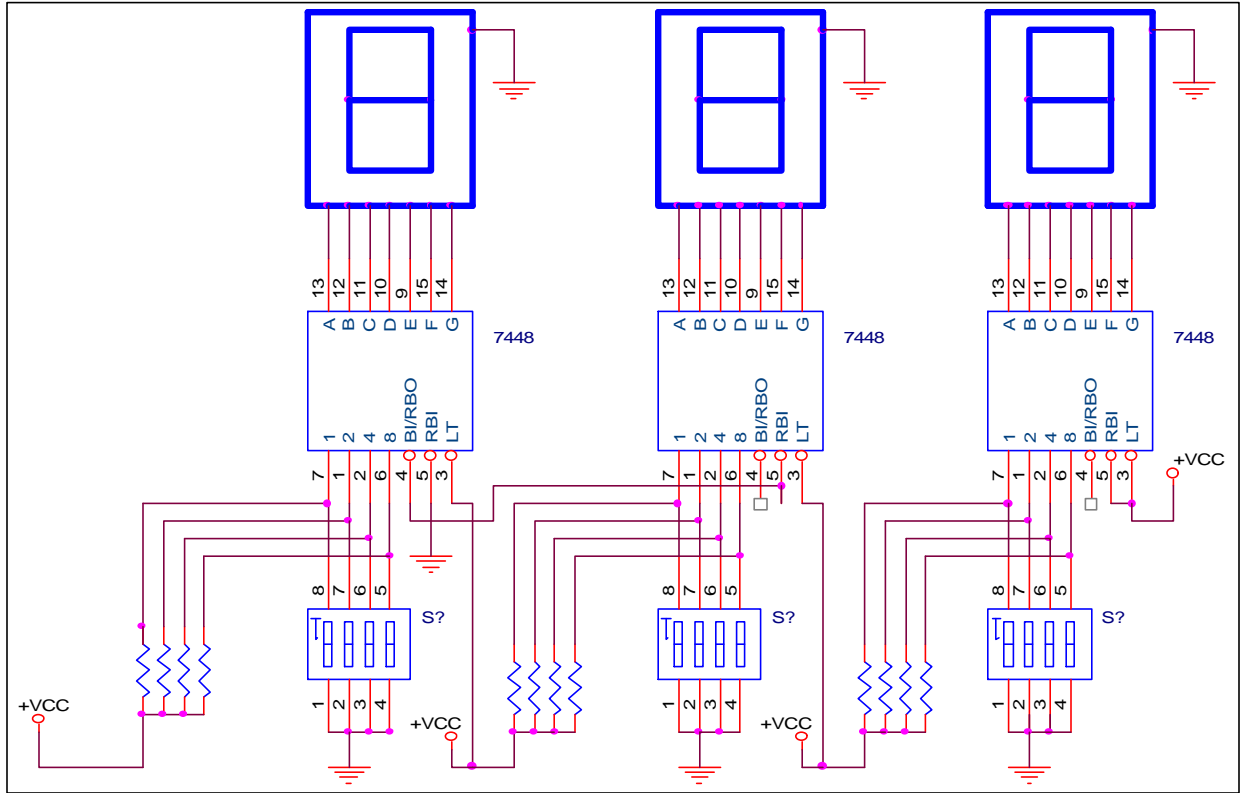


Figura 5.10. Visualizador de tres dígitos, con apagado de ceros a la izquierda.

5.1.3.3. Decodificador como generador de funciones de conmutación.

Tomando en cuenta que los decodificadores tienen activa solo una de las 2^n salidas y los minterms o Maxterms coinciden con esto cuando la función tiene n variables de entrada. Entonces se puede generar, con los decodificadores convencionales, funciones lógicas que correspondan con las salidas en minterms o Maxterms agregando compuertas lógicas de la siguiente forma:

Dada la función de conmutación; $F(X, Y, Z) = \sum_m (0, 2, 5, 7)$.

- I. Para decodificadores con salidas activas en alto se tiene que $F = m_0 + m_2 + m_5 + m_7$ en forma compacta, lo que indica la conexión de una compuerta OR a la salida del decodificador. Figura 5.11a.

- II. Para decodificadores con salidas activas en alto se tiene que $F = \overline{M_1 + M_3 + M_4 + M_6}$ en forma compacta, lo que indica la conexión de una compuerta NOR a la salida del decodificador. Figura 5.11b.
- III. Para decodificadores con salidas activas en bajo se tiene que $F = \overline{m_0 + m_2 + m_5 + m_7}$ en forma compacta, lo que indica la conexión de una compuerta NAND a la salida del decodificador. Figura 5.11c.
- IV. Para decodificadores con salidas activas en bajo se tiene que $F = M_1 * M_3 * M_4 * M_6$ en forma compacta, lo que indica la conexión de una compuerta AND a la salida del decodificador. Figura 5.11d.

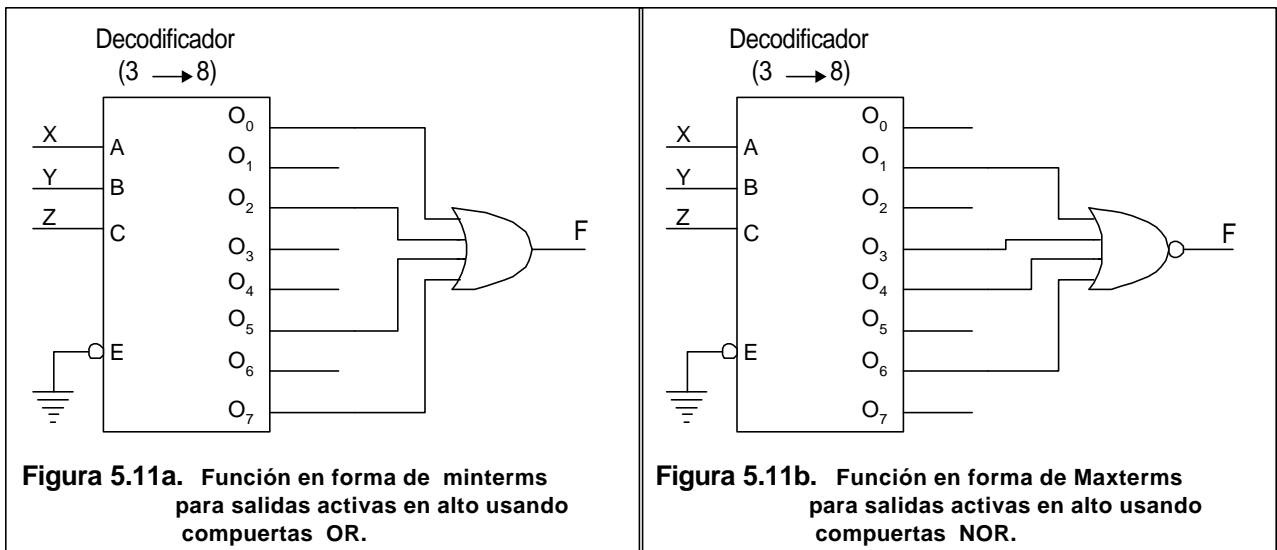


Figura 5.11

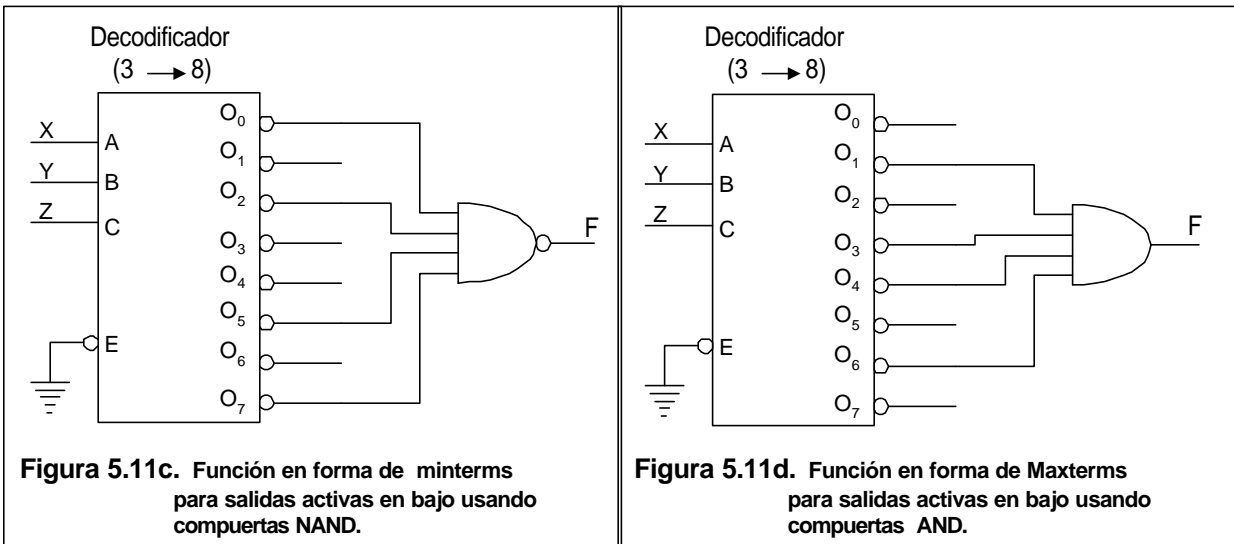


Figura 5.11

Ejercicio 5.4. Diseñar un convertidor de código binario a código gray de tres bits, utilizando un decodificador 74139 y sus respectivas compuertas.

Solución: Primero se debe construir la tabla de la verdad para generar las tres funciones lógicas de la conversión binario - gray. Luego, como es necesario un decodificador de tres entradas, que representen las tres variables del código entrante, se debe hacer expansión con los dos decodificadores que posee internamente el integrado 74139. Cada uno de ellos tiene: dos entradas activas en alto, un enable activo en bajo y cuatro salidas activas en bajo. Se debe hacer la expansión con este chip para transformarlo en otro de tres variables de entrada y ocho salidas. Por último, se deben utilizar compuertas NAND debido a que las salidas son activas en bajo.

B ₂	B ₁	B ₀	G ₂	G ₁	G ₀
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	1
0	1	1	0	1	0
1	0	0	1	1	0
1	0	1	1	1	1
1	1	0	1	0	1
1	1	1	1	0	0

$$G_2 = B_2$$

$$G_0(B_2, B_1, B_0) = \sum_m(1,2,5,6)$$

$$G_1(B_2, B_1, B_0) = \sum_m(2,3,4,5)$$

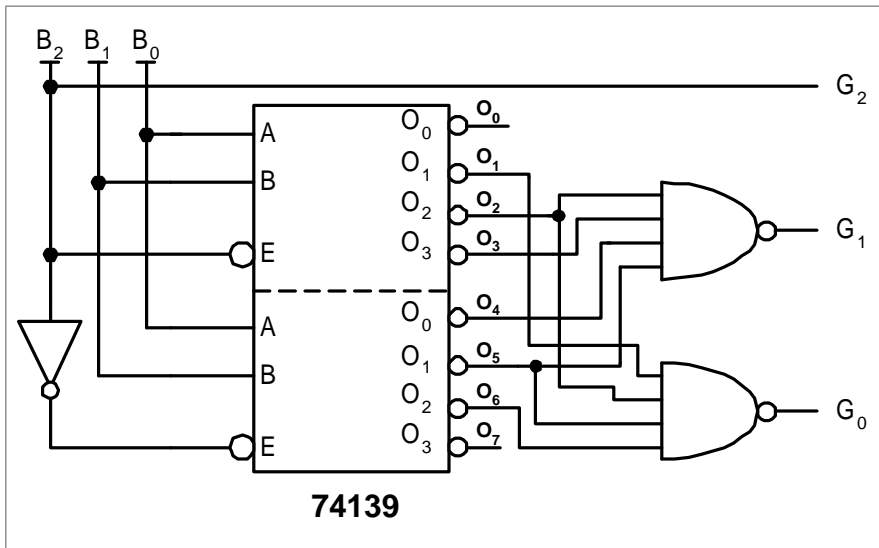


Figura 5.12. Circuito para la solución del ejercicio 5.4.

En la figura 5.12 se puede observar que B_0 y G_0 son iguales debido a que sus correspondientes columnas, en la tabla de la verdad, también lo son. Las salidas que corresponden con los minterms se conectan directamente a las entradas de las NAND.

Ejercicio 5.5. Diseñar un sumador completo de un bit utilizando el decodificador 74138 y compuertas.

Ejercicio 5.6. Realizar un bloque decodificador que tenga 26 salidas activas en bajo, las entradas activas en alto y un enable activo en bajo; utilizando para ello, solamente, chips 74139.

Ejercicio 5.7. Realizar una expansión de 6 entradas a 64 salidas con el decodificador 74154.

PRÁCTICA DE LABORATORIO #4

TITULO: Circuitos combinacionales Decodificadores y Convertidores de código.

OBJETIVO: El estudiante al terminar esta práctica estará en capacidad de poder analizar y diseñar circuitos combinacionales Decodificadores y Convertidores de código de mediana escala de integración (MSI).

INTRODUCCIÓN: Los decodificadores y convertidores de código tienen diversas aplicaciones en los circuitos digitales combinacionales, pueden generar funciones de conmutación, sirven para manejar displays, son utilizados como decodificador de direcciones de memoria, etc. Las bases teóricas para realizar esta práctica están contenidos en los temas 5.1, 5.2, 5.3 y 5.4 de la presente bibliografía; no obstante, se puede utilizar otra bibliografía recomendada en esta guía; también es necesario utilizar un manual TTL y un programa de simulación electrónica digital, de cualquier fabricante, para complementar el laboratorio. Los dos montajes planteados contribuirán a obtener las destrezas necesarias para avanzar en circuitos digitales combinacionales MSI. Sin embargo, al final de esta guía se proponen otros montajes adicionales para que el profesor del curso pueda sustituir o modificar la práctica a conveniencia de todos los participantes.

PRELABORATORIO: Investigar los siguientes tópicos.

- Displays 7 segmentos, LCD, Tubos de gas neón, Matrix de puntos.
- Decodificadores, Convertidores de código discretos MSI.
- Estudiar las características de los Decodificadores 7448, 7447, 74138, 74139.
- Implementación de funciones con Decodificadores.
- Aplicaciones de con decodificadores y convertidores de código.

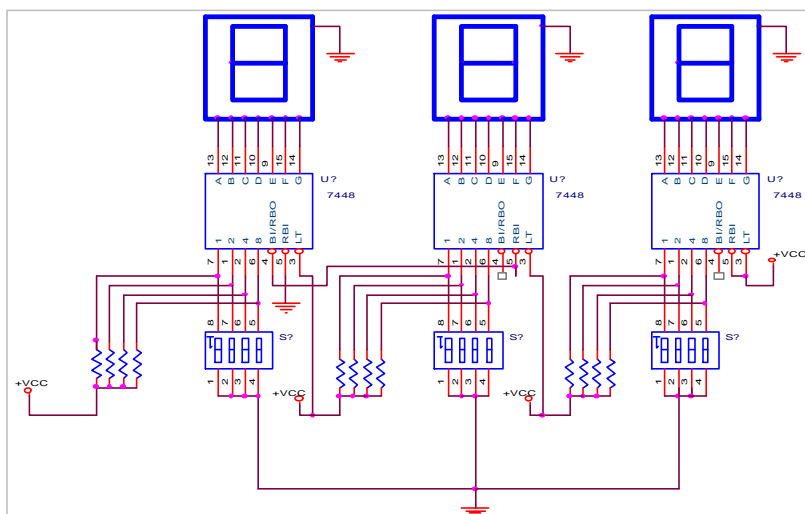
MATERIALES Y EQUIPOS NECESARIOS:

- Tres display 7 segmentos de acuerdo al convertidor utilizado y tres diodos leds.
- Decodificadores 74138, 74139 o equivalentes.
- Convertidores de código (TTL ó CMOS) de acuerdo al tipo de display que se vaya ha usar.
- Protoboard, cable telefónico, pinza, piqueta.
- Chips de compuertas de acuerdo al diseño.
- Multímetro digital y fuente de 5 Volt / 2 Amp.

DESARROLLO:

1. Realizar el montaje de un circuito con tres dígitos 7 segmentos que muestre los valores comprendidos entre 0 y 999. En el circuito no se deben visualizar los ceros que están a la izquierda (ceros no significativos). **Nota:** El diseño es libre Ud. debe tratar de obtener la mejor minimización del circuito digital.

El circuito mostrado puede servir de modelo en éste montaje.



2. Implementar con decodificadores **74138** y/o **74139** las siguientes funciones:

- $F(A,B,C,D) = \mathbf{S}_m(3,6,9,13) + d(0,1,5)$
- $F(A,B,C,D) = \mathbf{P}_M(2,3,6,10,12,14,15) * d(1,4,7,13)$
- $F(A,B,C,D) = \mathbf{S}_m(1,6,9,14)$

POST-LABORATORIO.

- Describa como funcionan los **pinos RBI, BI/RBO y LT del 7447 o 7448**.
- Haga un análisis de todas las posibles combinaciones que se pueden realizar para colocar compuertas en la salida de un decodificador, cuando éste último es utilizado como generador de funciones lógicas.
- Investigue una aplicación donde los tres dígitos puedan manejarse con **punto decimal flotante**, de forma tal que pueda ser usado como "**autoescala**".
- El **montaje número uno** también se puede realizar con un solo convertidor **7448 o 7447**. Explique como se implementa y haga el plano completo.

MONTAJES ALTERNATIVOS:

1. Diseñar con visualizadores 7 segmentos un display de dos dígitos que muestre en código hexadecimal los valores binarios de la entrada.
2. Implementar un sumador completo de un bit con decodificadores 74138 y/o 74139 que pueda indicar con diodos leds la suma y el acarreo de salida.
3. Implementar un circuito digital, con dos salidas, que señale por una de ellas cuando un dato de entrada binario de cuatro bits sea divisible por cuatro y en la otra, los números divisibles por tres. Diseñar el circuito con decodificadores.

4. Diseñar, con decodificadores 74138, 74139 o 74154 un restador de dos bits con signo. Este último se puede visualizar con un diodo led y el resultado con display 7 segmentos.
5. Diseñar e implementar un decodificador de 24 líneas de salidas, y una entrada de habilitación activa en bajo.

BIBLIOGRAFÍA.

- CUESTA, Luís M. PADILLA G, Antonio. REMIRO D, Fernando. (1993). Electrónica digital. Madrid: McGraw Hill. S/f. p.445.
- GAJSKI, Daniel D. (1997). Principios de diseño digital. Madrid: Prentice Hall Iberia. S/f. p.488. "Principles of digital design". Traducido por: Alberto Prieto Espinosa.
- LLORIS, Antonio. PRIETO, Alberto. (1996). Diseño lógico. Madrid: McGraw Hill. S/f. p.403.
- MANDADO, Enrique. (1987). Sistemas electrónicos digitales. Barcelona (España): Marcombo Boixareu Editores. Sexta edición. p.705.
- MANO, Morris. KIME, Charles. (1998). Fundamentos de diseño lógico y computadoras. México: Prentice Hall. Primera edición en español. P.604. "Logic and computer design fundamentals". Traducido por: Teresa Sanz Falcón.
- NELSON, V. NAGLE, H. CARROLL, B. IRWIN, J. (1996). Análisis y diseño de circuitos lógicos digitales. México: Prentice Hall. Primera edición. p.842. "Digital logic circuit analysis and design". Traducido por: Oscar A. Palmas V.
- TOCCI, Ronald. (1995). Sistemas digitales principios y aplicaciones. México: Prentice Hall. Quinta edición. p.823. "Digital systems principles and applications". Traducido por: Edmundo G. Urbina M.
- WARKELY, John F. (1997). Diseño digital principios y prácticas. México: Prentice Hall. S/f. p.743. "Digital design principles and practices". Traducido por: Gutiérrez R. Raymundo H.

MANUALES.

- NATIONAL SEMICONDUCTOR. (1981). Manual TTL y CMOS.
- MOTOROLA Inc. (1992). Fast and LS TTL. (DL121/D REV. 5).

5.2 Codificadores.

Son circuitos integrados digitales combinacionales que poseen 2^n líneas de entrada y n líneas de salida; realizan la operación contraria a los decodificadores. Las líneas de entrada y salida pueden ser también activas en los dos niveles: alto o bajo. El circuito codificador responde de forma tal que coloca un código binario en la salida cuando una de sus entradas se encuentra activa. En la figura 5.13 se puede observar un bloque codificador genérico con 2^n entradas y n salidas. La figura 5.14 muestra un circuito codificador y su respectiva tabla de la verdad, diseñado con compuertas, el codificador posee 4 entradas y 2 salidas activas en alto. Las ecuaciones son:

$$O_0 = \bar{X}_3 \bar{X}_2 X_1 \bar{X}_0 + X_3 \bar{X}_2 \bar{X}_1 \bar{X}_0$$

$$O_1 = \bar{X}_3 X_2 \bar{X}_1 \bar{X}_0 + X_3 \bar{X}_2 \bar{X}_1 \bar{X}_0$$

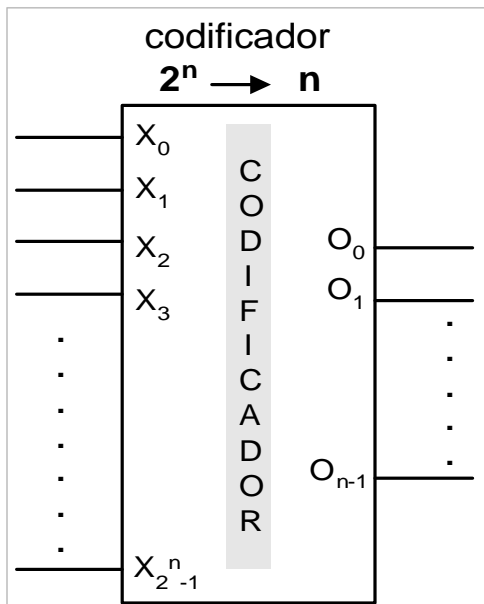


Figura 5.13. Codificador genérico con 2^n Entradas y n salidas binarias.

Tabla 5.1

X_3	X_2	X_1	X_0	O_1	O_0
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1

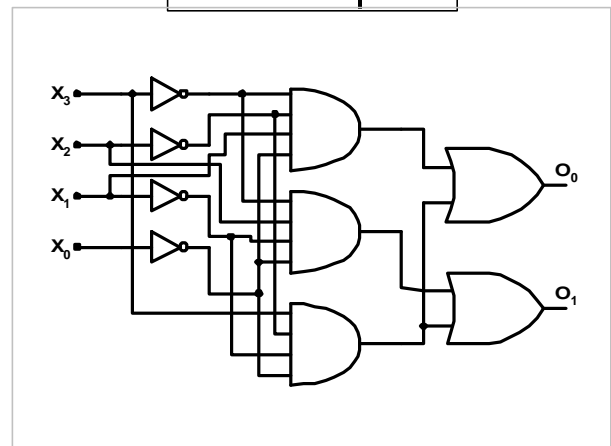


Figura 5.14. Codificador de compuertas con Cuatro entradas y dos salidas.

El circuito de la figura 5.14 tiene la desventaja de no admitir más de una entrada activa porque el código de salida será de condiciones inesperadas. Por ejemplo, si X_3 y X_2 están en alto al mismo tiempo y $X_1 = X_0 = 0$, entonces, se genera la salida $O_1 O_0 = 0 0$ lo cual no era de esperarse. Por los motivos antes expuestos, este tipo de codificador no posee aplicaciones prácticas y en su lugar se utiliza el codificador con prioridad.

5.2.1 Codificadores de prioridad.

Para evitar el inconveniente presentado en los codificadores citados anteriormente y asegurar una salida binaria que responda correctamente, sin ambigüedades, a la señal de entrada, se debe diseñar un codificador de prioridad. Este circuito debe generar el código de salida correspondiente a la línea activa de entrada más significativa; de esta manera, al activarse simultáneamente más de una línea de entrada, éste colocará en la salida el código correspondiente a la más significativa. Las figuras 5.15 y 5.16 muestran el diseño de un codificador de prioridad con cuatro entradas. En la tabla 5.2 se pueden apreciar los valores irrelevantes (**d**) en las entradas ($X_3 X_2 X_1 X_0$) menos significativas, la habilitación de grupo en la entrada (**EI**), las líneas de salida ($O_1 O_0$) y el señalizador de grupo (**SG**), que indica si hay entrada activa.

Tabla 5.2. Codificador de prioridad

EI	X_3	X_2	X_1	X_0	O_1	O_0	SG
0	d	d	d	d	0	0	0
1	0	0	0	0	0	0	0
1	0	0	0	1	0	0	1
1	0	0	1	d	0	1	1
1	0	1	d	d	1	0	1
1	1	d	d	d	1	1	1

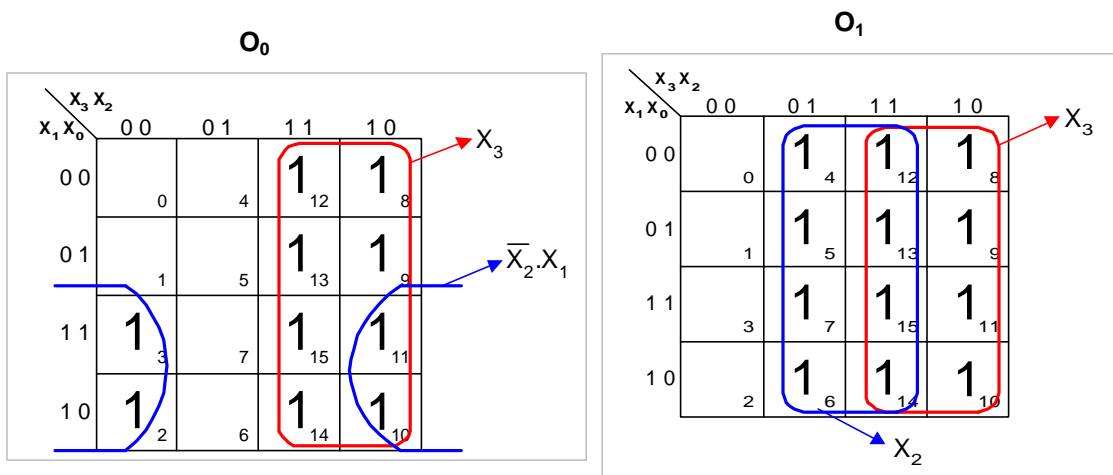


Figura 5.15. Mapas K para la simplificación de las funciones O_0 y O_1 .

La entrada **EI=1** es común para todos los códigos; al cambiar a cero se desactivan todas las salidas, por lo tanto, se puede implementar con AND para cada salida. De la **tabla 5.x** y los mapas K se obtiene las funciones: $O_0 = EI \cdot (\overline{X_2} \cdot X_1 + X_3)$, $O_1 = EI \cdot (X_2 + X_3)$ y $SG = EI \cdot (\overline{X_3 + X_2 + X_1 + X_0})$ las cuales representan el circuito de compuertas para un codificador de prioridad.

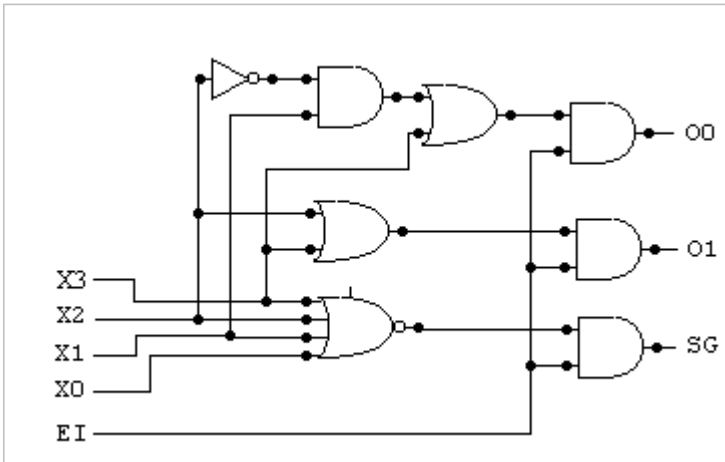


Figura 5.16. Codificador de cuatro entradas con prioridad, habilitación de entrada (EI) e indicador de activación de código (SG).

La entrada **EI** en cero desactiva todas las salidas y la condición en las entradas es irrelevante. La señal de salida **SG** detecta cuando hay alguna entrada activa en el circuito, la compuerta **NOR** de cuatro entradas es la encargada de esto. Si alguna de sus entradas se coloca en uno, entonces la salida cambia a cero desactivando la compuerta **AND** que hace de llave para la salida **SG**. En este circuito la condición de que dos o más entradas estén activas al mismo tiempo no tiene importancia porque el código de salida corresponderá a la entrada más significativa. Por ejemplo, la condición anormal del codificador anterior es resuelta aquí: Si **EI** está activa y la combinación de entrada es: $X_3 \ X_2 \ X_1 \ X_0 = 1 \ 1 \ 0 \ 0$, entonces la salida es $O_1=1$ y $O_0=1$.

5.2.1.1 Codificadores de prioridad MSI.

Los circuitos integrados codificadores más conocidos son los chips **74147** y **74148** de la familia TTL, los cuales son descritos en la tabla 5.3 y sus respectivos diagramas en las figuras 5.17 y 5.18.

N° del Codificador	FUNCIÓN	ENTRADAS	SALIDAS	CONTROL
74147	Convierte código Decimal a BCD.	9 entradas activas En nivel bajo.	4 líneas de Salidas activas En bajo.	No tiene línea De control para Las E/S.
74148	Convierte código Octal a Binario.	8 entradas activas En nivel bajo.	3 líneas de Salidas activas En bajo.	Tiene 3 líneas De control para Las E/S.

Tabla 5.3. Descripción de la función del 74147 y 74148.

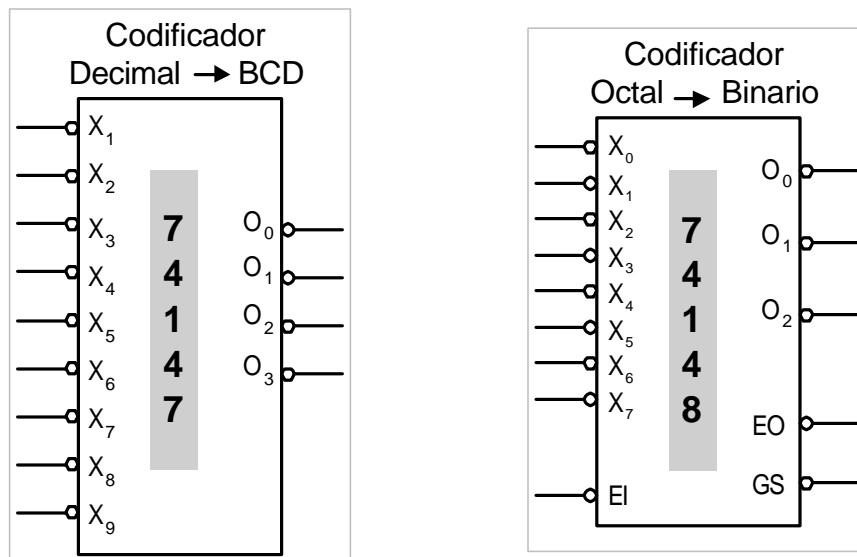


Figura 5.17. Codificadores de prioridad 74147 y 74148.

Tabla 5.4: funcionamiento del 74147.

X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	O_0	O_1	O_2	O_3
1	1	1	1	1	1	1	1	1	1	1	1	1
0	1	1	1	1	1	1	1	1	1	1	1	0
d	0	1	1	1	1	1	1	1	1	1	0	1
d	d	0	1	1	1	1	1	1	1	1	0	0
d	d	d	0	1	1	1	1	1	1	0	1	1
d	d	d	d	0	1	1	1	1	1	0	1	0
d	d	d	d	d	0	1	1	1	1	1	0	1
d	d	d	d	d	d	0	1	1	1	1	0	0
d	d	d	d	d	d	d	0	1	1	1	1	0
d	d	d	d	d	d	d	d	0	1	1	1	1
d	d	d	d	d	d	d	d	d	0	1	1	0

Tabla 5.5: funcionamiento del 74148.

EI	X_0	X_1	X_2	X_3	X_4	X_5	X_6	X_7	O_2	O_1	O_0	GS	EO
1	d	d	d	d	d	d	d	d	1	1	1	1	1
0	1	1	1	1	1	1	1	1	1	1	1	1	0
0	0	1	1	1	1	1	1	1	1	1	1	0	1
0	d	0	1	1	1	1	1	1	1	1	0	0	1
0	d	d	0	1	1	1	1	1	1	0	1	0	1
0	d	d	d	0	1	1	1	1	1	0	0	0	1
0	d	d	d	d	0	1	1	1	1	0	1	1	0
0	d	d	d	d	d	0	1	1	1	0	1	0	1
0	d	d	d	d	d	d	0	1	1	0	0	1	0
0	d	d	d	d	d	d	d	0	1	0	0	1	0
0	d	d	d	d	d	d	d	d	0	0	0	0	1

Figura 5.18. Tablas de la verdad de los codificadores de prioridad 74147 y 74148.

Cuando la entrada de habilitación **EI** del chip 74148 está en nivel alto, todas las líneas de entradas (X_0, \dots, X_7) son indiferentes, las salidas se desactivan, la línea de salida Enable Output **EO** se coloca en alto y el Señalizador de Grupos **GS** también se desactiva. Esta condición es equivalente a la deshabilitación del circuito integrado; sin embargo, no se debe confundir con la condición de salida para el cero (tercera fila de la tabla del 74148) ni con la condición cuando todas las entradas están desactivadas (segunda fila de la tabla). Estas tres condiciones están diferenciadas por los valores de las líneas de salida **EO** y **GS**. Estas últimas son complementarias, el Enable Output es cero solo cuando no hay entrada activa; también, el **GS** es cero cuando hay alguna entrada activa en el codificador.

En el ejemplo de la figura 5.19 se muestra un diagrama que corresponde a una aplicación de un teclado lineal hexadecimal realizado con expansión de dos codificadores 74148. Este circuito detecta cuando ha sido pulsada una o más teclas y la convierte en su correspondiente código binario de cuatro bits. Al presionar simultáneamente más de una tecla, entonces aparece en la salida (O_0, O_1, O_2, O_3), la combinación binaria de la tecla más significativa del código hexadecimal entrante. La línea de salida (**T_P**) indica, con un uno, el momento cuando se presiona alguna tecla.

Ejercicio 5.8. Construir un convertidor de código decimal a BCD de cuatro bits.

Solución: En la figura 5.20 se detalla el circuito convertidor Decimal - BCD. El código de salida está complementado a uno, por lo cual, es necesario colocar inversores para

obtener el verdadero valor binario. El cero, en binario invertido, se obtiene cuando todos los DIP-SW están abiertos.

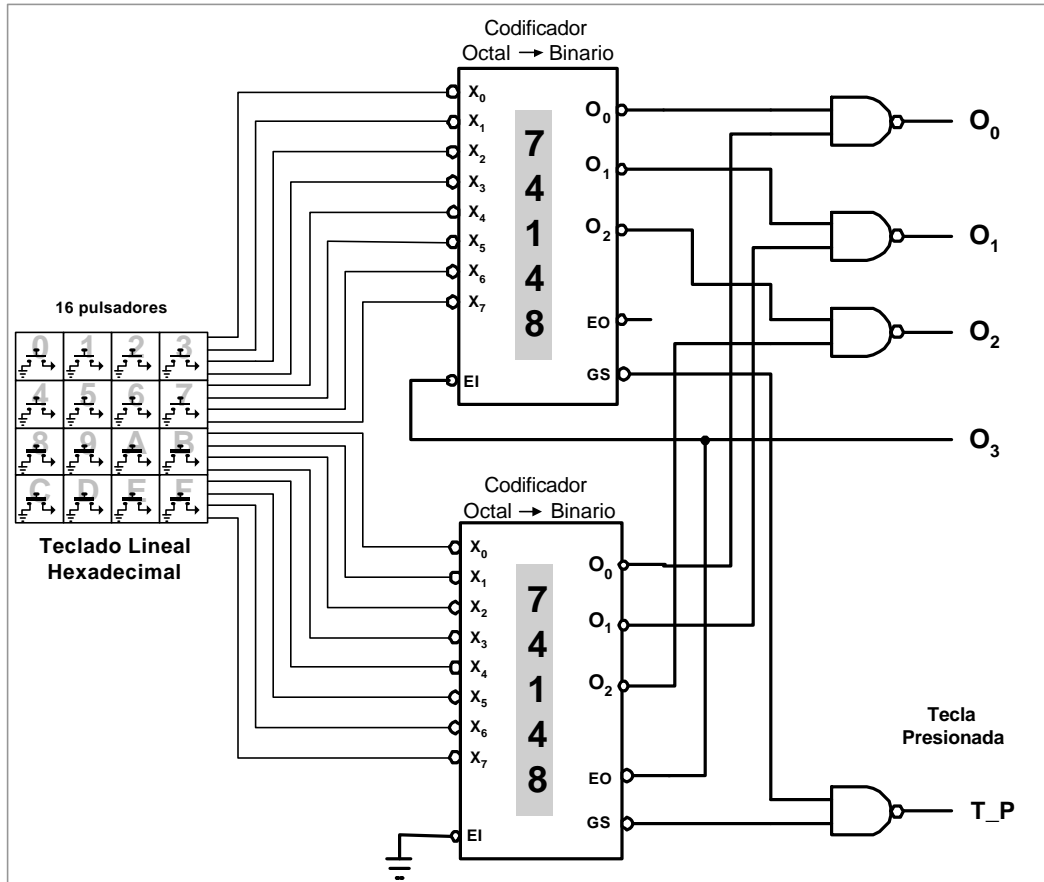


Figura 5.19. Teclado lineal hexadecimal implementado con codificadores 74148.

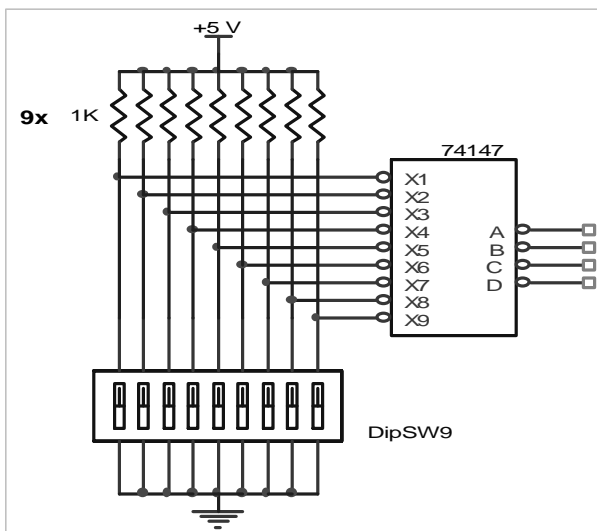


Figura 5.20. Circuito que convierte código Decimal a BCD de 4 bits utilizando 74147.

5.3 Multiplexores.

Es un circuito combinacional que selecciona una línea de entrada de datos y la coloca en la salida, Posee 2^n líneas de entrada de datos y n líneas de selección. Cada línea de entrada es conmutada hacia la salida por intermedio de las líneas de selección, formando éstas últimas una combinación binaria que determinarán cual línea de entrada (I_r), equivalente en decimal, le corresponderá colocarse en la salida (F) del multiplexor (**MUX**).

La fórmula que identifica a un MUX es: $F = I_r \cdot m_r$, donde r es igual al valor decimal de $(S_{n-1} \cdot S_{n-2} \cdot \dots \cdot S_2 \cdot S_1 \cdot S_0)_2$ y m_r el símbolo correspondiente. Los multiplexores se conocen también como selectores de datos y en la figura 5.21, se puede apreciar el multiplexor genérico, descrito anteriormente. También, se puede ver en la figura 5.22, un MUX de cuatro entradas, dos líneas de selección y un enable activo en bajo realizado con compuertas digitales. Las compuertas AND de cuatro entradas poseen una línea de habilitación común, activada por la salida de la compuerta NOT (c_1); ésta hace que la línea sea activa en bajo. Luego, las líneas S_1 y S_0 seleccionan y activan una de las cuatro AND; la compuerta, AND, seleccionada dejará pasar hacia las compuertas OR el valor de su respectiva entrada I_r . Por lo cual, F tomará este valor lógico de la entrada ($F=I_r$). La función que describe el comportamiento es:

$$F = \bar{E} [I_3(S_1 \cdot S_0) + I_2(S_1 \cdot \bar{S}_0) + I_1(\bar{S}_1 \cdot S_0) + I_0(\bar{S}_1 \cdot \bar{S}_0)]$$

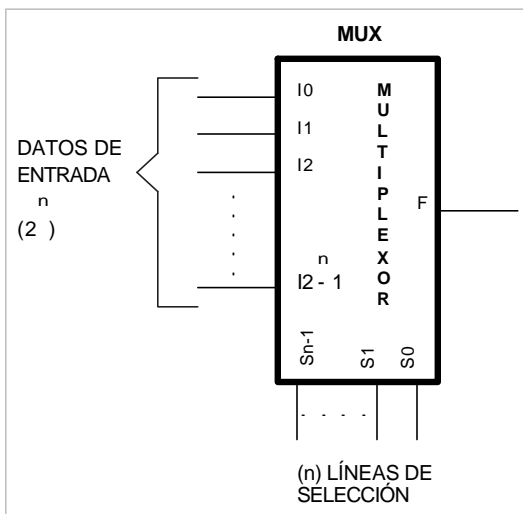


Figura 5.21. Multiplexor genérico.

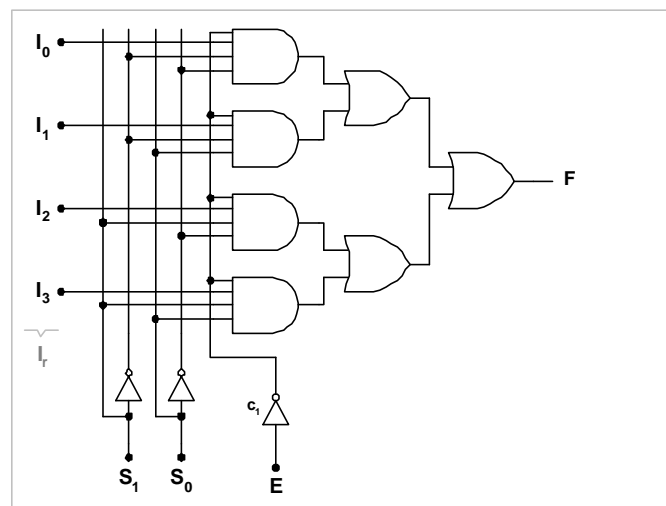


Figura 5.22. Multiplexor de 4 entradas discreto.

5.3.1 Aplicaciones de los multiplexores.

Los multiplexores pueden ser utilizados como selectores de datos, convertidores de datos paralelo - serial y también sirven como generadores de funciones lógicas.

5.3.1.1 Multiplexor como selector de datos.

El circuito de la figura 5.23 puede seleccionar una de entre cuatro palabras binarias, cada una de ellas, con un tamaño de 4 bits. Esta aplicación puede servir para mostrar uno de cuatro procesos realizados con esas palabras y dichos procesos pueden ser operaciones lógicas o aritméticas.

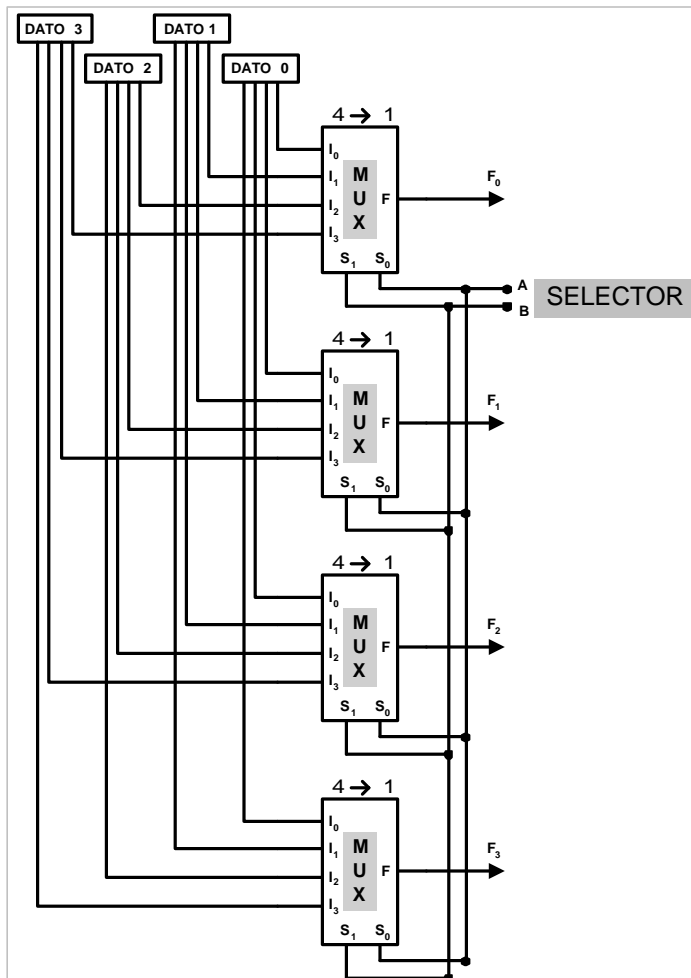


Figura 5.23. Selector de datos implementado con multiplexores genéricos de 4 entradas.

Las dos líneas del selector (A y B) determinan cual palabra de cuatro bits (Dato 0, Dato 1, Dato 2 o Dato 3) va hacia las salidas (F₀, F₁, F₂ y F₃).

A	B	F ₀	F ₁	F ₂	F ₃
0	0	Dato 0			
0	1	Dato 1			
1	0	Dato 2			
1	1	Dato 3			

5.3.1.2 Convertidores paralelo - serial con multiplexores.

Los datos que entran simultáneamente a un **MUX** salen por un solo canal y este es, precisamente, la salida del circuito integrado. Por lo cual, todo esto, se traduce en una línea serial de datos. Por otra parte, las líneas de selección deben ser conmutadas para colocar cada entrada, una por una, en la salida del multiplexor. Esta conmutación en las líneas de selección debe ser realizada por un contador binario de frecuencia fija. La forma de onda cuadrada, en la salida serial **F**, puede ser programada por la combinación paralela que hay a la entrada del multiplexor. En la figura 5.24 se muestra un circuito de este tipo que convierte ocho líneas de entrada paralela en 256 formas posibles de ondas cuadradas que salen por la línea de salida **F** del multiplexor.

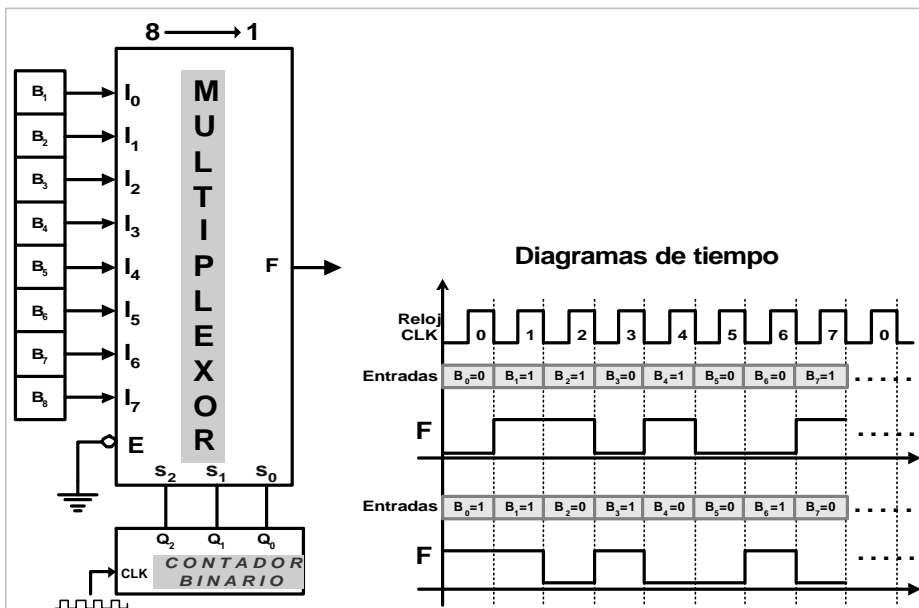


Figura 5.24. Convertidor de datos paralelo serial.

El contador binario cíclico que se coloca en las líneas de selección, debe ir desde cero hasta siete; la figura 5.24 posee dos ejemplos de formas de ondas cuadradas que se obtienen al realizar los cambios correspondientes en las líneas de entrada B_i .

5.3.1.3 Circuitos integrados multiplexores MSI.

Los multiplexores vienen encapsulados en chips con distintas configuraciones de líneas de entradas, y líneas de selección. Las familias TTL y CMOS poseen varios tipos de multiplexores que van desde 2 hasta 16 líneas de entrada; a continuación se indican las características de los circuitos integrados más utilizados, ver tabla 5.6 y figura 5.25.

Número	TTL y CMOS (Función)	N° de entradas	Líneas de Selección	Líneas de Habilitación	Salidas
74LS157 74HC157 74157	4 Multiplexores	2 C/U	1 Línea común	1 Línea común	1 por cada MUX.
74LS158 74158	4 Multiplexores	2 C/U	1 Línea común	1 Línea común	1 por cada MUX. Activas en bajo
74LS153 74HC153 74153	2 Multiplexores	4 C/U	2 líneas comunes	2 líneas independientes	1 activa en alto
74LS151 74HC151 74151	1 Multiplexor	8	3	1	1 activa en alto 1 activa en bajo
74150	1 Multiplexor	16	4	1	1 activa en bajo
74LS251 74HC251 74251	1 Multiplexor	8	3	1 común; coloca las salidas el alta Impedancia	1 activa en alto 1 activa en bajo
74LS253 74HC253 74253	2 Multiplexores	4 C/U	2 líneas comunes	2 independientes; coloca las salidas el alta Impedancia	1 activa en alto
74LS257 74HC257 74257	4 Multiplexores	2 C/U	1 Línea común	1 común; coloca las salidas el alta Impedancia	1 por cada MUX.

Tabla 5.6. Multiplexores MSI más utilizados de las familias TTL y CMOS.

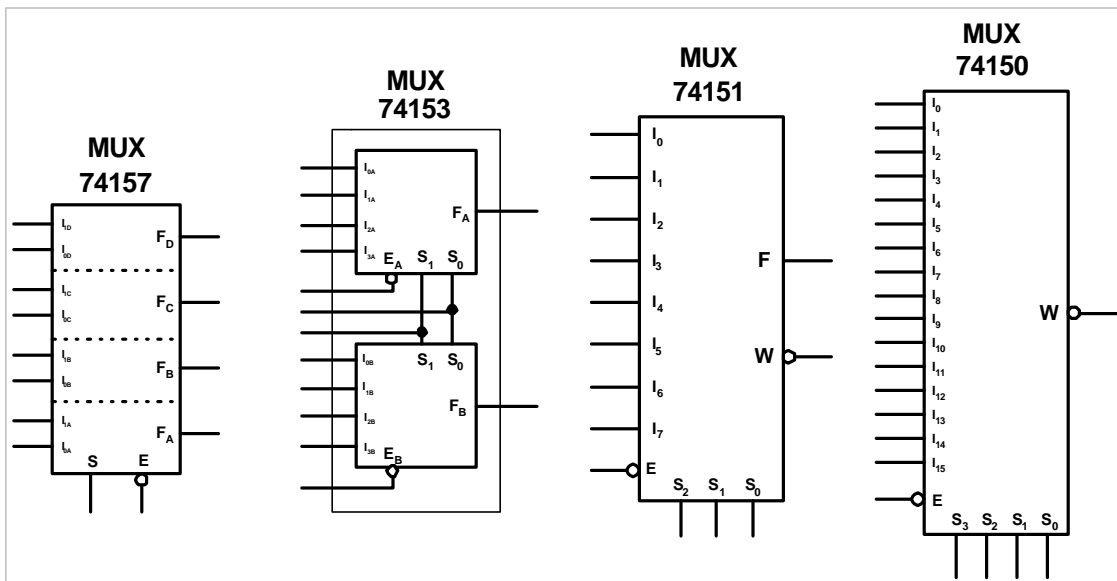


Figura 5.25. Circuitos integrados multiplexores más utilizados.

5.3.2 El Multiplexor como generador de funciones lógicas.

Las funciones lógicas de conmutación se pueden generar con multiplexores; esto se obtiene relacionando cada variable de la función, con las entradas y líneas de selección del circuito integrado específico. Cada minterms ó Maxterms se pueden obtener con la combinación binaria de las líneas de selección, son 2^n entradas (minterms) y n líneas de selección (variables) para implementar las funciones. Sin embargo, la cantidad de variables puede ser mayor que el número de líneas de selección del multiplexor.

Atendiendo al número de variables y líneas de selección, necesarias para generar funciones con circuitos integrados multiplexores, se toman en cuenta tres casos:

1. La cantidad de variables es igual al número de líneas de selección.
2. Una variable excede al número de líneas de selección.
3. Dos variables exceden al número de líneas de selección.

5.3.2.1 El número de variables y líneas de selección son iguales.

Para generar la función de conmutación, se toman las entradas del multiplexor y se colocan a cero lógico (GND) ó uno lógico (+VCC) en correspondencia con los Maxterms y minterms.

Ejercicio 5.9. Implantar la siguiente función de conmutación con el MUX 74151.

$$F(A, B, C) = \bar{A}C + AB$$

Solución: El circuito integrado posee tres líneas de selección al igual que el número de variables de la función. Primero, se expande la función y se expresa en lista de minterms:

$$F(A, B, C) = \bar{A}C + AB = \bar{A}BC + \bar{A}\bar{B}C + AB\bar{C} + ABC = \sum_m(1, 3, 6, 7)$$

Luego, la lista de minterms debe ser generada colocando +5V en cada entrada respectiva (I_1, I_3, I_6, I_7) y las entradas que corresponden con los Maxterms (I_0, I_2, I_4, I_5) se conectan a tierra (GND). Las líneas de selección se deben conectar con las variables de la función. Ver figura 5.26.

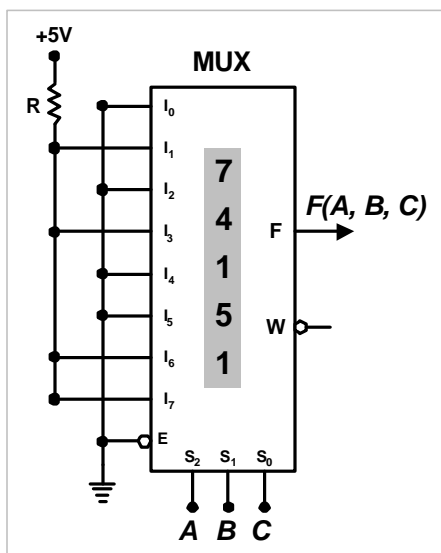


Figura 5.26. Solución del ejercicio 5.9.

Ejercicio 5.10. Generar la función $g(w, x, y, z) = \sum_m(0, 1, 2, 3, 4, 8, 9, 13, 14)$ con el chip 74150.

Solución: El 74150 tiene cuatro líneas de selección y 16 entradas; las cuales corresponden con la misma cantidad de variables, minterms y/o Maxterms de la función lógica. Ver figura 5.27.

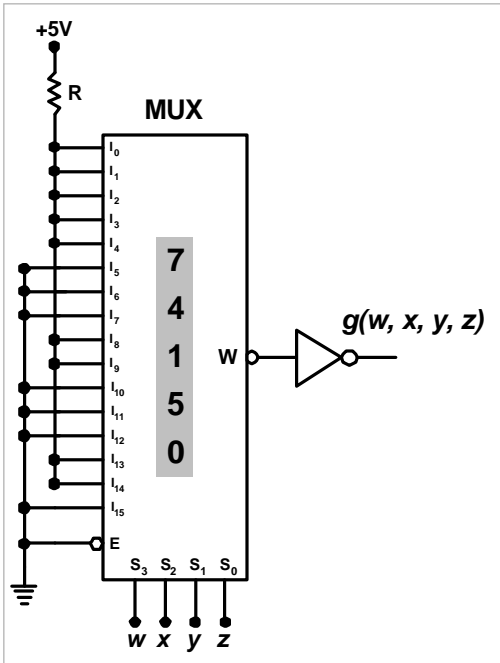


Figura 5.27. Solución del ejercicio 5.10.

5.3.2.2 La función excede en una variable al número de líneas de selección.

En este caso se introduce una de las variables de la función por la entrada de datos. Esta variable puede ser cualquiera; sin embargo, se recomienda que sea la más significativa de la función lógica. El procedimiento a seguir para implantar la función se realiza con el ejercicio 5.10 y el chip a utilizar es el multiplexor de tres líneas de selección 74151.

- I. Se determina cual de las variables será introducida por la entrada de datos; en el ejercicio dado, la variable que se introducirá por la entrada de datos es “w”.
- II. Realizar un mapa de KARNAUGH con las variables de la función; el mapa debe tener dos filas ó columnas y debe corresponder con la variable que entra por la línea de datos. Se colocan, en las celdas, los minterms y Maxterms respectivos. Ver figura 5.28.

		x	y	z								
	w				000	001	011	010	110	111	101	100
	0	1	1	1	1	0	0	0	0	1		
					0	1	3	2	6	7	5	4
	1	1	1	0	0	1	0	1	0	1	0	
					8	9	11	10	14	15	13	12

Figura 5.28. Mapa K de dos filas.

III.A cada una de las entradas del multiplexor se le asignan las combinaciones formadas por las variables restantes de la función (las menos significativas). En el ejercicio dado deben ser las letras “x y z”.

$$I_0 = \bar{x}.\bar{y}.\bar{z}$$

$$I_1 = \bar{x}.\bar{y}.z$$

$$I_2 = \bar{x}.y.\bar{z}$$

$$I_3 = \bar{x}.y.z$$

$$I_4 = x.\bar{y}.\bar{z}$$

$$I_5 = x.\bar{y}.z$$

$$I_6 = x.y.\bar{z}$$

$$I_7 = x.y.z$$

IV. La variable que entra por las líneas de datos del multiplexor puede presentar cuatro alternativas para su conexión:

1. $I_i = 0$; Si las dos celdas correspondientes a la variable más significativa son ceros la entrada debe conectarse a tierra o GND.
2. $I_i = 1$; Si las dos celdas correspondientes a la variable más significativa son unos; entonces la entrada debe conectarse al +Vcc.
3. $I_i = w$; Esto sucede cuando cada valor de celda del mapa K posee el mismo nivel lógico que los estados asignados a la variable que entra por datos. En la figura 5.28 el valor de la celda 5 y 13 es cero lógico y uno lógico respectivamente. Estos son los mismos estados asignados a la

variable w (cero y uno lógico); por lo tanto esa entrada será igual a la variable asignada por la entrada de datos.

4. $I_i = \bar{w}$; Esto sucede cuando cada valor de celda del mapa K posee el nivel lógico contrario a los estados asignados en la variable que entra por datos. En la figura 5.28 el valor de la celda 3 y 11 es uno lógico y cero lógico respectivamente. Estos estados asignados a la variable w (cero y uno lógico) son contrarios; por lo tanto esa entrada será igual al complemento de la variable asignada por la entrada de datos. El circuito resultante se muestra en la figura 5.29.

Las entradas del multiplexor 74151 quedan de la siguiente forma:

$$I_7 = GND$$

$$I_0 = I_1 = +5V$$

$$I_2 = I_3 = I_4 = \bar{w}$$

$$I_5 = I_6 = w$$

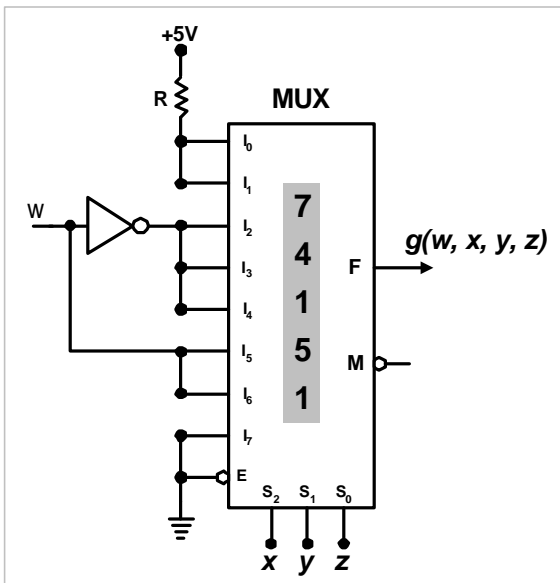


Figura 5.29. Variable "w" asignada a la entrada de datos del 74151.

5.3.2.3 La función excede en dos variables al número de líneas de selección.

En este caso se introducen dos de las variables, de la función, por la entrada de datos del multiplexor. Estas variables pueden ser cualquiera; sin embargo, se recomienda que sean las más significativas de la función lógica. El procedimiento a seguir para implantar la función se realiza con el ejercicio 5.10 y el chip a utilizar es el multiplexor de dos líneas de selección 74153.

El procedimiento que se sigue es idéntico al caso de una variable. No obstante, el Mapa de KARNAUGHT debe tener cuatro filas ó columnas. Ver figura 5.30.

$$I_0 = \overline{w} + \overline{x}; \quad I_1 = w + \overline{x}; \quad I_2 = \overline{x}; \quad I_3 = \overline{w \oplus x}$$

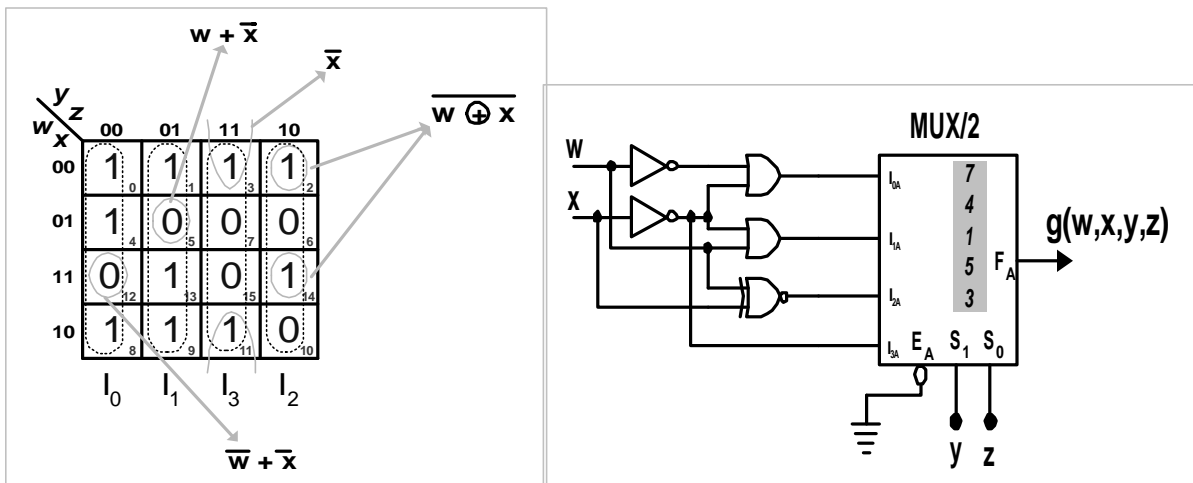


Figura 5.30. Circuito de compuertas para introducir dos variables por las líneas de datos.

Ejercicio 5.11. Generar la función:

$$F(A, B, C, D, E) = \sum_m (0,1, 4,7,9,14,16,18,21,22,29,31) + d(5,12,13) \text{ con un solo multiplexor}$$

74151 y compuertas. Introducir dos variables por la entrada de datos.

Solución: En esta función de conmutación se aprovechan los términos indiferentes para reducir el circuito de compuertas. Los minterms indiferentes 5 y 13 se colocan en uno lógico para llevar la entrada cinco I_5 a $+V_{cc}$; Por otra parte, con el minterm indiferente 12 se pueden igualar las entradas I_4 e I_1 . La figura 5.31 presenta la solución a este ejercicio.

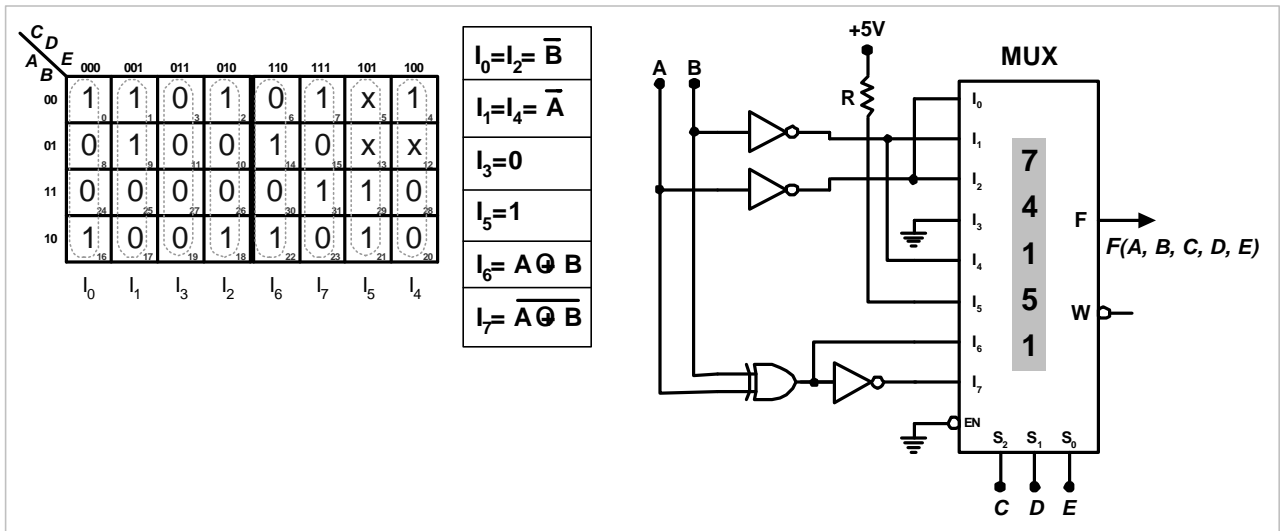


Figura 5.31. Solución del ejercicio 5.11 con el chip 74151 y compuertas.

Ejercicio 5.12. Diseñe un multiplexor de 32 entradas utilizando solamente circuitos integrados 74153 y compuertas. Utilizar la menor cantidad de compuertas posibles.

Ejercicio 5.13. Generar las siguientes funciones con multiplexores 74150.

$$F(w, x, y, z) = \sum_m (0,1,6,7,8,10,12,14,15)$$

$$F(a, b, c, d) = \prod (3,5,7,8,9,13) \cdot d(0,5)$$

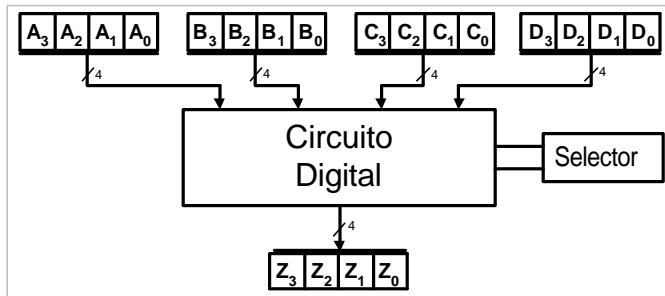
$$F(m, n, o, p) = \sum_m (1,3,5,7,9,11,13,14,15) + d(2,8)$$

Ejercicio 5.14. Realizar las implementaciones con el chip 74151; utilizando las mismas funciones del ejercicio 5.13 y colocando una variable por la entrada de datos.

Ejercicio 5.15. Diseñar un restador completo de un bit utilizando multiplexores 74153.

Ejercicio 5.16. Obtener las dos funciones en forma de Maxterms y minterms del convertidor de datos paralelo – serial de la figura 5.24.

Ejercicio 5.17. Dadas cuatro palabras **A**, **B**, **C** y **D** de cuatro bits cada una, seleccionar una sola y colocarla en la salida **Z**. A continuación se muestra el diagrama.



Ejercicio 5.18. Generar la función dada en la tabla con el multiplexor 74151. colocar dos variables por la entrada de datos.

<i>n</i>	A	B	C	D	E	F
0	0	0	0	0	0	1
1	0	0	0	0	1	1
2	0	0	0	1	0	1
3	0	0	0	1	1	0
4	0	0	1	0	0	0
5	0	0	1	0	1	X
6	0	0	1	1	0	1
7	0	0	1	1	1	1
8	0	1	0	0	0	X
9	0	1	0	0	1	0
10	0	1	0	1	0	1
11	0	1	0	1	1	0
12	0	1	1	0	0	0
13	0	1	1	0	1	X
14	0	1	1	1	0	0
15	0	1	1	1	1	0

<i>n</i>	A	B	C	D	E	F
16	1	0	0	0	0	0
17	1	0	0	0	1	0
18	1	0	0	1	0	0
19	1	0	0	1	1	1
20	1	0	1	0	0	0
21	1	0	1	0	1	X
22	1	0	1	1	0	0
23	1	0	1	1	1	0
24	1	1	0	0	0	1
25	1	1	0	0	1	1
26	1	1	0	1	0	1
27	1	1	0	1	1	1
28	1	1	1	0	0	0
29	1	1	1	0	1	X
30	1	1	1	1	0	0
31	1	1	1	1	1	1

Tabla de la verdad para el ejercicio 5.18

PRÁCTICA DE LABORATORIO #5

TITULO: Circuitos combinacionales Codificadores.

OBJETIVO: El estudiante al terminar esta práctica estará en capacidad de poder analizar y diseñar circuitos combinacionales codificadores de mediana escala de integración (MSI).

INTRODUCCIÓN: Entre los usos que tienen los codificadores de prioridad, en los circuitos digitales combinacionales, se encuentran los controladores de interrupciones y codificadores de teclados octales, decimales y hexadecimales. Esta práctica de laboratorio está elaborada para realizar un codificador de teclado decimal y un controlador de interrupciones de cuatro entradas. El primero debe ser realizado con el chip 74147 y el segundo puede realizar con compuertas o cualquier circuito integrado codificador. Se recomienda para esta práctica repasar la unidad 5.2, utilizar un manual TTL y consultar la bibliografía al final de esta guía.

PRELABORATORIO: Investigar los siguientes tópicos.

- Codificadores simples y codificadores de prioridad MSI.
- Generar funciones de conmutación con compuertas (codificadores).
- Características de los circuitos integrados 74148, 74147.
- Teclados de contactos matriciales, contactos independientes y otros.
- Manejo de teclado e interrupciones con chips codificadores.

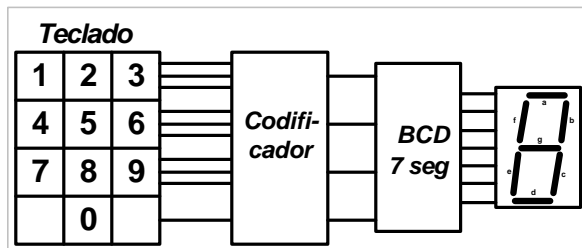
MATERIALES Y EQUIPOS NECESARIOS:

- Circuitos integrados 7447, 7448, 74147 y/o algún otro chip codificador de acuerdo con las necesidades del diseño (74148, etc.).
- Teclado de 10 contactos “independientes” (no matricial).
- Tres diodos leds y un display 7 segmentos.

- Protoboard, cable telefónico, pinza, piqueta.
- Chips de compuertas de acuerdo al diseño.
- Multímetro digital y fuente de 5 Volt / 2 Amp.

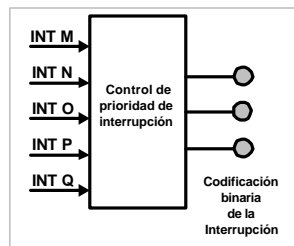
DESARROLLO:

1. Realizar el diseño de un teclado decimal que señale en display 7 segmentos el valor de la tecla presionada desde cero hasta nueve. Este último debe permanecer apagado mientras no se presione ninguna tecla. El teclado debe funcionar de forma que al presionar dos o más teclas el circuito muestre el mayor valor. A continuación se muestra el diagrama en bloques del circuito que puede servir de modelo en este montaje.



Circuito en bloques del manejador de teclado decimal.

2. Implementar un control de interrupciones por prioridad con compuertas o chips codificadores; el circuito debe tener cinco entradas (**M**, **N**, **O**, **P**, **Q**) donde, correlativamente, **M** es la entrada de mayor prioridad y **Q** la menor. La salida del circuito debe señalar con diodos leds el valor binario de la orden de interrupción.



Esquema para el montaje del control de interrupciones.

POST-LABORATORIO.

- Diseñar teclados de tipo octal y hexadecimal.
- Explicar con diagramas y tablas de la verdad el funcionamiento para apagar el display.
- Realizar expansión con el codificador 74148 de un circuito con 32 líneas de entradas.
- Demostrar como se implemento la visualización de la tecla "0".
- Realizar los diagramas y las simulaciones del segundo montaje con:
a) compuertas; b) con circuitos integrados codificadores 74147 y 74148.

MONTAJES ALTERNATIVOS:

1. Diseñar y realizar montajes de teclados octales y hexadecimales.
2. Expansiones con circuitos integrados codificadores 74147 y 74148.
3. Diseño e implementación de circuitos digitales con peticiones de interrupciones por prioridad.
4. Realizar un circuito digital que indique el valor de la jugada de dos jugadores de dados. Los dados pueden ser equivalentes a los chips 74147 y cada jugada puede ser implementada con DIP-SW de seis interruptores.

BIBLIOGRAFÍA.

- CUESTA, Luís M. PADILLA G, Antonio. REMIRO D, Fernando. (1993). Electrónica digital. Madrid: McGraw Hill. S/f. p.445.
- MANDADO, Enrique. (1987). Sistemas electrónicos digitales. Barcelona (España): Marcombo Boixareu Editores. Sexta edición. p.705.
- NELSON, V. NAGLE, H. CARROLL, B. IRWIN, J. (1996). Análisis y diseño de circuitos lógicos digitales. México: Prentice Hall. Primera edición. p.842. "Digital logic circuit analysis and design". Traducido por: Oscar A. Palmas V.
- TOCCI, Ronald. (1995). Sistemas digitales principios y aplicaciones. México: Prentice Hall. Quinta edición. p.823. "Digital systems principles and applications". Traducido por: Edmundo G. Urbina M.
- WARKELY, John F. (1997). Diseño digital principios y prácticas. México: Prentice Hall. S/f. p.743. "Digital design principles and practices". Traducido por: Gutiérrez R. Raymundo H.

MANUALES.

- NATIONAL SEMICONDUCTOR. (1981). Manual TTL y CMOS.
- MOTOROLA Inc. (1992). Fast and LS TTL. (DL121/D REV. 5).

PRÁCTICA DE LABORATORIO #6

TÍTULO: Circuitos combinacionales Multiplexores y Demultiplexores.

OBJETIVO: El estudiante al terminar esta práctica estará en capacidad de poder analizar y diseñar circuitos combinacionales multiplexores (MUX) y demultiplexores (DEMUX) de mediana escala de integración (MSI).

INTRODUCCIÓN: Los circuitos integrados multiplexores tienen aplicaciones tales como convertidor de datos paralelo a serial, generador de funciones y selector de datos. Por otra parte, los chips demultiplexores funcionan de forma contraria; ellos reciben la información por una sola línea de entrada y las distribuye a la salida en paralelo. No obstante, necesitan también otras líneas de entrada que sirvan para el control de esta distribución de datos. Por lo general, el decodificador es el circuito utilizado para tal fin con lo cual se considera equivalente a los demultiplexores. Esta práctica consta de dos montajes: un primer montaje llamado Multiplexor Demultiplexor (MUXDEMUX) que permite la conversión de datos paralelo a serial para transmitirlos en el MUX, luego, en el DEMUX recibirlos en serial y llevarlos a paralelo; el segundo montaje es un convertidor de código, implementado con multiplexores, que debe transformar el código cuando cambia una señal de control. El fundamento teórico para este laboratorio está en la bibliografía recomendada al final de la guía y en los temas 5.1, 5.3 y 5.4 del capítulo cinco. El montaje número uno utiliza un bloque contador de tres bits que es descrito en el apéndice de esta bibliografía.

PRELABORATORIO: Investigar los siguientes tópicos.

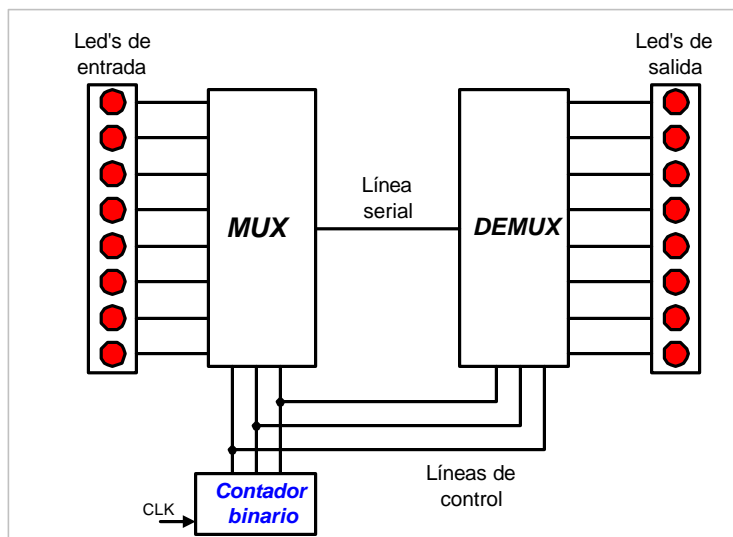
- Circuitos Multiplexores y Demultiplexores (decodificadores) MSI.
- Implementación de funciones lógicas con Multiplexores.
- Aplicaciones generales de los Multiplexores.

MATERIALES Y EQUIPOS NECESARIOS:

- Un decodificador (DEMUX) 74138, un multiplexor (MUX) 74151 y dos chips 74153.
- 16 diodos leds y DIP_SW de 8 y 4 interruptores.
- Compuertas digitales de acuerdo a los diseños realizados.
- Módulo contador binario de tres bits.
- Protoboard, cable telefónico, pinza, piqueta.
- Multímetro digital y fuente de 5 Volt / 2 Amp.

DESARROLLO:

1. Implementar en Protoboard un circuito multiplexor-demultiplexor (**MUXDEMUX**) de ocho bits utilizando multiplexores y decodificadores; visualizar la entrada paralela del multiplexor con diodos leds y la salida paralela DEMUX también. Al encender cualquier led en la entrada también debe encender él (los) correspondientes a la salida. Se debe colocar a la entrada CLK del contador un generador de onda cuadrada (generador de funciones) o un circuito “oscilador astable TTL”.



Esquema en bloques del montaje número uno.

2. Diseñar e implementar un circuito convertidor de código de tres bits con dos chips multiplexores 74153. El circuito debe tener una señal de control "R" que, en uno lógico, el circuito cambie de binario al código descrito en la tabla y de esta última a binario si la señal de control es cero lógico. Señalizar la salida del circuito con diodos leds.

R	X ₂	X ₁	X ₀	f ₂	f ₁	f ₀			R	X ₂	X ₁	X ₀	f ₂	f ₁	f ₀
1	0	0	0	0	0	0			0	0	0	0	0	0	0
1	0	0	1	1	0	0			0	0	0	1	0	1	0
1	0	1	0	0	0	1			0	0	1	0	1	0	0
1	0	1	1	1	0	1			0	0	1	1	1	1	0
1	1	0	0	0	1	0			0	1	0	0	0	0	1
1	1	0	1	1	1	0			0	1	0	1	0	1	1
1	1	1	0	0	1	1			0	1	1	0	1	0	1
1	1	1	1	1	1	1			0	1	1	1	1	1	1

POST-LABORATORIO.

- En el montaje número uno se debe explicar en que momento dejan de parpadear los leds y ¿por qué?.
- Analizar el funcionamiento paso a paso en el selector del multiplexor e indicar una posible solución para eliminar los tres cables de control que van hacia el decodificador.
- Hacer el diagrama y la simulación de los dos montajes utilizando otros chips multiplexores y demultiplexores.
- Diseñar un selector de datos de cinco palabras; cada una de ellas tiene un tamaño de cuatro bits.

- Explicar las ventajas y desventajas de los multiplexores sobre los decodificadores cuando se utilizan como generador de funciones lógicas.

MONTAJES ALTERNATIVOS:

1. Realizar el montaje de un circuito Multiplexor donde se visualicen cuatro dígitos en displays 7 segmentos utilizando un solo circuito integrado convertidor de código. El diseño debe mostrar valores en unidades (U), decenas (D), centenas (C) y unidades de mil (UM) desde "0" hasta "9999". Los valores numéricos deben ser introducidos al circuito mediante DIP_SW; además de ello, los ceros a la izquierda no deben mostrarse en los displays.

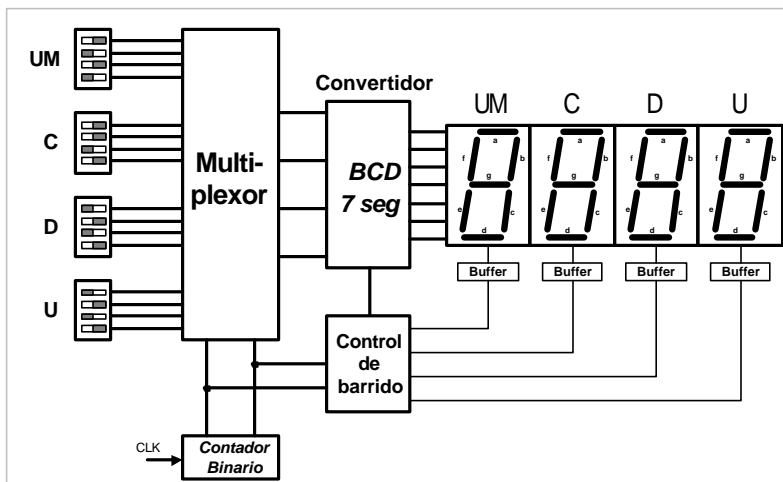


Diagrama en bloques del montaje alternativo número uno.

2. Implementar con un multiplexor 74151 la función dada a continuación:

$$F(A, B, C, D, E) = \prod_M (3,5,6,7,12,13,14,17,19,20,21,23,27,29,30,31) \cdot d(0,8,9,11,15)$$

3. Implementar un circuito multiplicador de dos bits ($A_1A_0 \times B_1B_0$) utilizando dos chips 74153 o cuatro chips 74151. La salida debe ser mostrada en un display 7 segmentos.

4. Diseñar e implementar con el 74157 un circuito que seleccione de tres datos entrantes, de cuatro bits cada uno, cual de ellos debe ser colocado en la salida. Esta última debe ser visualizada con diodos leds.

BIBLIOGRAFÍA.

- CUESTA, Luís M. PADILLA G, Antonio. REMIRO D, Fernando. (1993). Electrónica digital. Madrid: McGraw Hill. S/f. p.445.
- GAJSKI, Daniel D. (1997). Principios de diseño digital. Madrid: Prentice Hall Iberia. S/f. p.488. "Principles of digital design". Traducido por: Alberto Prieto Espinosa.
- LLORIS, Antonio. PRIETO, Alberto. (1996). Diseño lógico. Madrid: McGraw Hill. S/f. p.403.
- MANDADO, Enrique. (1987). Sistemas electrónicos digitales. Barcelona (España): Marcombo Boixareu Editores. Sexta edición. p.705.
- MANO, Morris. KIME, Charles. (1998). Fundamentos de diseño lógico y computadoras. México: Prentice Hall. Primera edición en español. P.604. "Logic and computer design fundamentals". Traducido por: Teresa Sanz Falcón.
- NELSON, V. NAGLE, H. CARROLL, B. IRWIN, J. (1996). Análisis y diseño de circuitos lógicos digitales. México: Prentice Hall. Primera edición. p.842. "Digital logic circuit analysis and design". Traducido por: Oscar A. Palmas V.
- TOCCI, Ronald. (1995). Sistemas digitales principios y aplicaciones. México: Prentice Hall. Quinta edición. p.823. "Digital systems principles and applications". Traducido por: Edmundo G. Urbina M.
- WARKELY, John F. (1997). Diseño digital principios y prácticas. México: Prentice Hall. S/f. p.743. "Digital design principles and practices". Traducido por: Gutiérrez R. Raymundo H.

MANUALES.

- NATIONAL SEMICONDUCTOR. (1981). Manual TTL y CMOS.
- MOTOROLA Inc. (1992). Fast and LS TTL. (DL121/D REV. 5).

5.4 Circuitos digitales sumadores.

El sumador digital es un circuito combinacional que realiza la operación aritmética de sumar dos o más datos. La operación suma es la base de las unidades de computo en un sistema de procesamiento digital debido a que las operaciones de resta, multiplicación y división pueden crearse a partir de ésta. Por ejemplo, la resta de dos números binarios se puede expresar como la suma del minuendo más el complemento a dos del sustrayendo; por otra parte el producto y la división de dos números se obtienen realizando operaciones recursivas de sumas y restas respectivamente.

En la figura 5.32 se muestra un bloque sumador genérico de un bit, donde los datos a sumar son de un bit cada uno. El circuito debe tener una salida que corresponde con el resultado aritmético y otra que señala el acarreo de la operación. Debido a que no posee acarreo de entrada, el circuito se conoce como **semisumador**; y esto hace que no pueda ser acoplado en cascada directamente con otros bloques del mismo tipo. Sin embargo, el acoplamiento de los bloques semisumadores puede obtenerse a través de circuitos de compuertas. La solución a este problema se resuelve en la figura 5.33, donde se agrega un bit de acarreo en la entrada del circuito de forma que pueda ser utilizado para realizar expansiones de sumadores digitales con varios bloques de un solo bit acoplados en serie o en cascada.

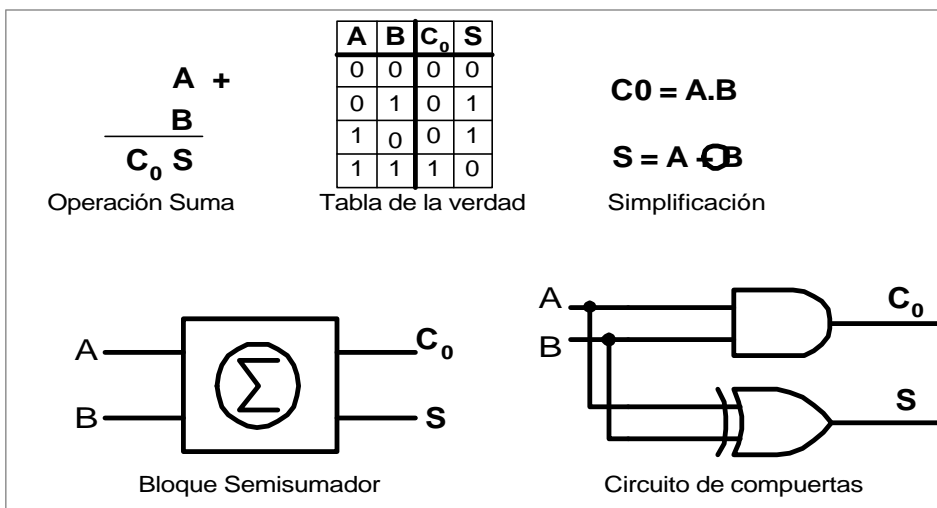


Figura 5.32. Circuito semisumador de un bit con compuertas digitales

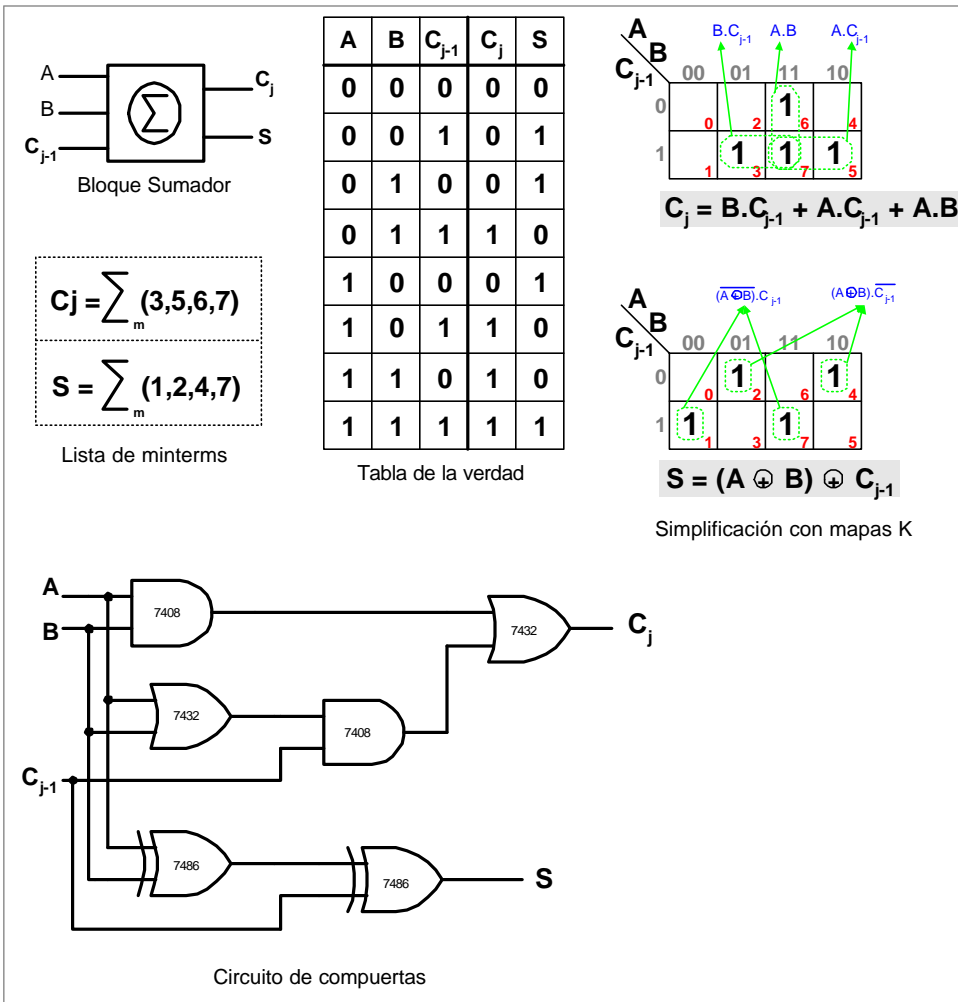


Figura 5.33. Sumador completo de un bit realizado con compuertas.

5.4.1 Sumador completo de un bit.

El circuito de la figura 5.33 es un sumador completo de un bit; este circuito puede acoplarse directamente en cascada para obtener sumadores de varios bits. El inconveniente del acoplamiento es el retardo de tiempo que se origina en cada bloque y que trae consigo una propagación total del circuito equivalente al producto del retardo de un bloque por la cantidad que van a ser conectados en serie. La figura 5.34 muestra un sumador serie de tres bits realizado con bloques de un bit. Al asumir retardos uniformes, el tiempo de propagación total será: $t_t = 3 \cdot t$; a medida que aumenta la cantidad de bloques, el retardo se hace mayor disminuyendo así la velocidad de respuesta del sumador.

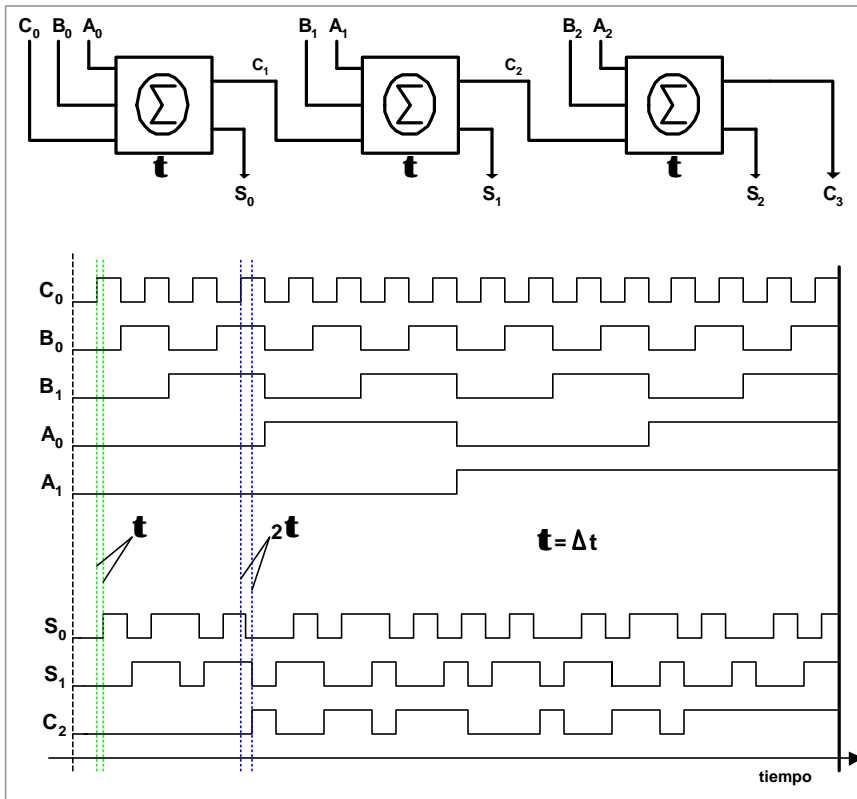


Figura 5.34. Sumador de tres bits con diagrama de tiempo para las salidas S0, S1 y C2.

El **sumador serie** de tres bits posee dos datos de entrada de tres bits cada uno A_2, A_1, A_0 y B_2, B_1, B_0 más el acarreo de entrada C_0 que es el bit menos significativo. La salida del sumador debe tener cuatro bits: los bits de resultado S_2, S_1, S_0 y el bit de acarreo de salida C_3 el cual es más significativo. En la figura 5.34 se observa el sumador de tres bits formado mediante el acoplamiento en serie de tres bloques individuales. La desventaja de este circuito se puede apreciar en el diagrama de tiempo, que por razones de espacio, se realiza para dos bits con acarreo de entrada; allí se observa que el retardo $Dt = t$ aparece en la salida S_0 duplicándose para la salida S_1 y C_2 respectivamente. En estos bloques se asume que los tiempos de propagación de las dos salidas S_i y C_{i+1} son iguales, cuestión ésta que no ocurre en la realidad; no obstante, las diferencia de retardo que existen entre S_i y C_{i+1} son muy pequeñas y pueden ser despreciadas. Los sumadores serie no son recomendados para sistemas donde se realicen operaciones aritméticas de alta velocidad.

5.4.2 Sumador paralelo.

El problema presentado por el sumador serie se resuelve utilizando sumadores paralelos donde los acarrees de salida deben ser conectados a un arreglo de compuertas digitales. El retardo del acarreo en cascada se puede reducir utilizando la técnica del acarreo anticipado (*CLA: carry look ahead generator*). Esta técnica consiste en generar bloques pequeños semisumadores con salidas K_i y M_i a los cuales se le agregan compuertas OR exclusivas por cada salida S_i obtenida del acarreo anticipado. El circuito de compuertas de la figura 5.35 se obtiene modificando la simplificación del mapa de Karnaugh de la figura 5.33 para la salida C_1 .

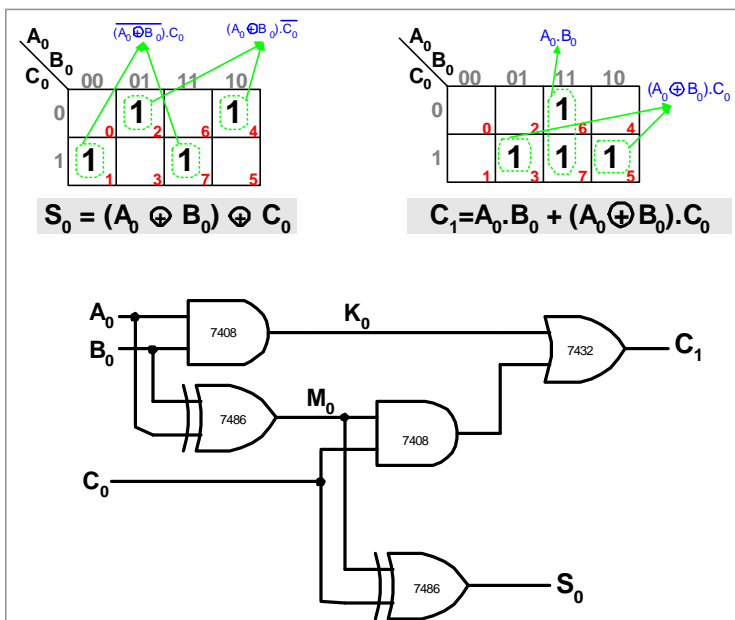


Figura 5.35. Circuito sumador paralelo de acarreo anticipado de un solo bit.

Los acarrees de un circuito sumador paralelo de cuatro bits se obtienen partiendo del bloque sencillo de la figura 5.35; a partir de éste se puede desarrollar la siguiente ecuación recursiva:

$$C_1 = K_0 + M_0 \cdot C_0$$

$$C_1 = A_0 \cdot B_0 + (A_0 \oplus B_0) \cdot C_0 \quad \text{Ec. 5.1}$$

$$C_2 = K_1 + M_1 \cdot C_1$$

$$C_2 = K_1 + M_1 \cdot (K_0 + M_0 \cdot C_0) = K_1 + M_1 K_0 + M_1 M_0 \cdot C_0$$

$$C_2 = A_1B_1 + (A_1 \oplus B_1) \cdot A_0B_0 + (A_1 \oplus B_1)(A_0 \oplus B_0) \cdot C_0$$

$$C_3 = K_2 + M_2C_2$$

$$C_3 = K_2 + M_2(K_1 + M_1K_0 + M_1M_0 \cdot C_0) = K_2 + M_2K_1 + M_2M_1K_0 + M_2M_1M_0C_0$$

$$C_3 = A_2B_2 + (A_2 \oplus B_2) \cdot A_1B_1 + (A_2 \oplus B_2)(A_1 \oplus B_1) \cdot A_0B_0 + (A_2 \oplus B_2)(A_1 \oplus B_1)(A_0 \oplus B_0) \cdot C_0$$

$$C_4 = K_3 + M_3C_3$$

$$C_4 = K_3 + M_3(K_2 + M_2K_1 + M_2M_1K_0 + M_2M_1M_0C_0)$$

$$C_4 = K_3 + M_3K_2 + M_3M_2K_1 + M_3M_2M_1K_0 + M_3M_2M_1M_0C_0$$

$$C_4 = A_3B_3 + (A_3 \oplus B_3)A_2B_2 + (A_3 \oplus B_3)(A_2 \oplus B_2)(A_1 \oplus B_1)A_0B_0 + (A_3 \oplus B_3)(A_2 \oplus B_2)(A_1 \oplus B_1)(A_0 \oplus B_0)C_0$$

⋮
 ⋮
 ⋮

$$C_n = K_{n-1} + M_{n-1}C_{n-1} \quad \text{Ec. 5.2}$$

$$C_n = K_{n-1} + M_{n-1}K_{n-2} + \dots + M_{n-1}M_{n-2} \dots M_2K_1 + M_{n-1}M_{n-2} \dots M_2M_1K_0 + M_{n-1}M_{n-2} \dots M_1M_0C_0$$

En La figura 5.35 se pueden observar el circuito básico para implementar la ecuación 5.2 formado por dos semisumadores acoplados. El primero genera la salida **K₀ M₀** con entradas **A₀ B₀** y el segundo genera la salida **S₀** con entradas **M₀ C₀**. Ambos semisumadores generan un retardo igual a la suma de los tiempos de propagación de las compuertas exclusivas. La figura 5.36 muestra un sumador paralelo de cuatro bits diseñado con la técnica de acarreo anticipado “CLA”; es de hacer notar que el tiempo total de propagación de este circuito sumador es menor o igual a cuatro niveles de acoplamiento de compuertas digitales. Asumiendo 10 ns como tiempo de retardo por cada nivel, la propagación total será de 40 ns.

Al comparar el circuito paralelo de la figura 5.36 con el circuito serie de la figura 5.37 se puede notar que casi duplica el retardo total (70 ns) para el sumador acoplado en cascada, y por ende más lento que el sumador paralelo. Esta diferencia de tiempo es más pronunciada a medida que aumenta el número de bits del sumador; sin embargo, un circuito paralelo con más de cuatro bits implicaría un circuito de compuertas CLA demasiado grande para ser manejado por componentes discretos. Esta desventaja no la posee el circuito serie, este último solamente se debe acoplar con bloques en cascada de acuerdo a la cantidad de bits de salida que se requieran.

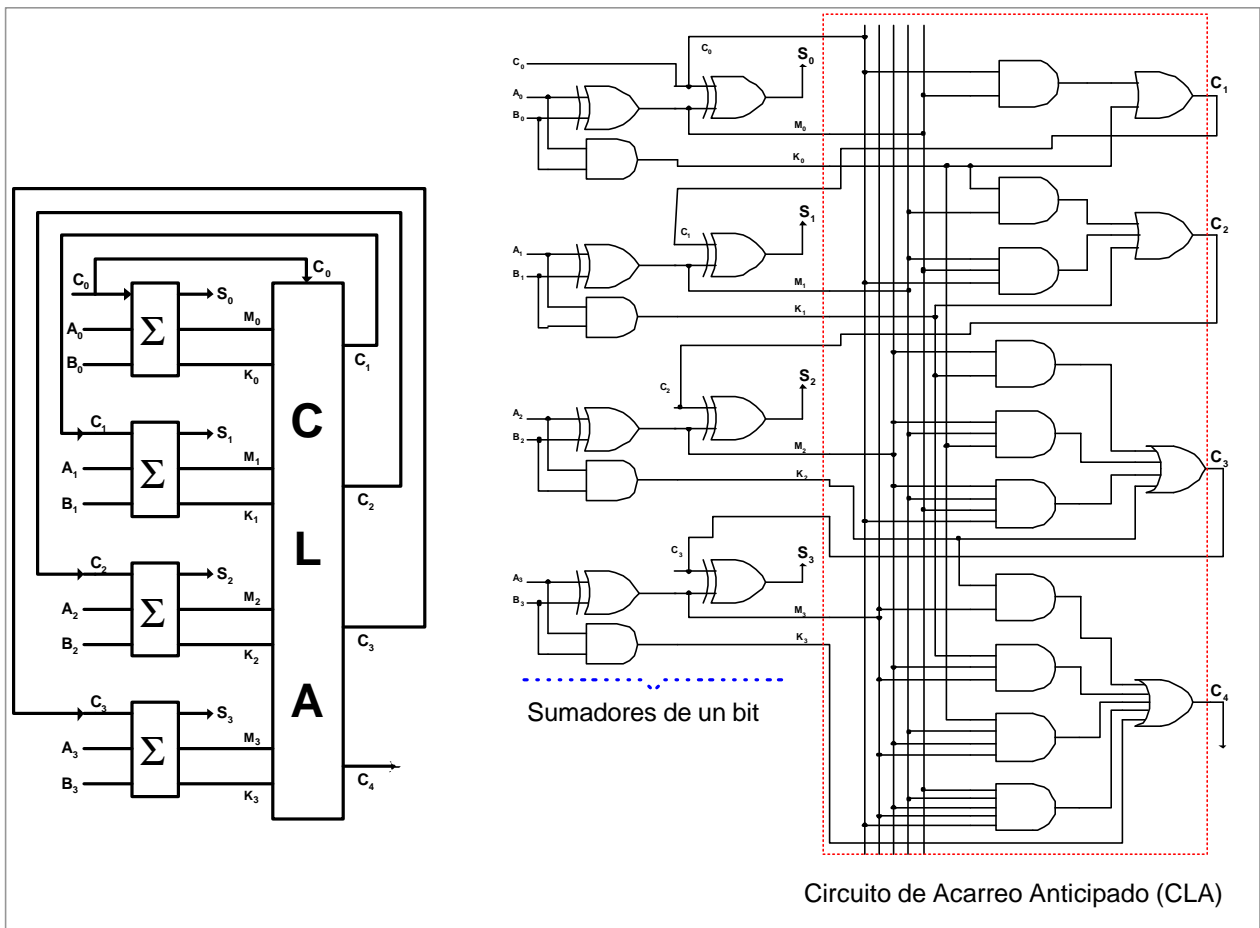


Figura 5.36. Diagrama en bloques y circuito sumador de cuatro bits paralelo con CLA.

El circuito CLA de la figura 5.36 tiene dos niveles de propagación de tiempo y los sumadores de un bit también poseen dos niveles; por lo que el acoplamiento de ellos dos, tendrán un retardo total de cuatro niveles. Por otra parte, si cada nivel de retardo es aproximadamente 10 ns (para compuertas TTL Estándar), entonces el tiempo de propagación total será de 40 ns. Este retardo de tiempo se mantiene igual para una mayor integración de compuertas por ejemplo, cinco, seis, siete u ocho líneas de entrada por cada dato; sin embargo, el circuito sumador serie de la figura 5.37 tiene ocho niveles de retardo de compuertas ($4 \times 2 = 8$) en la propagación de los acarreos de entrada y salida, C_j y C_{j+4} respectivamente. El retardo total es de 70 ns si no se toma en cuenta la propagación de la última compuerta del acarreo C_4 . Si es necesario aumentar la cantidad de bits en los dos sumandos, la consecuencia será un retardo de 20 ns por cada bloque que se agregue.

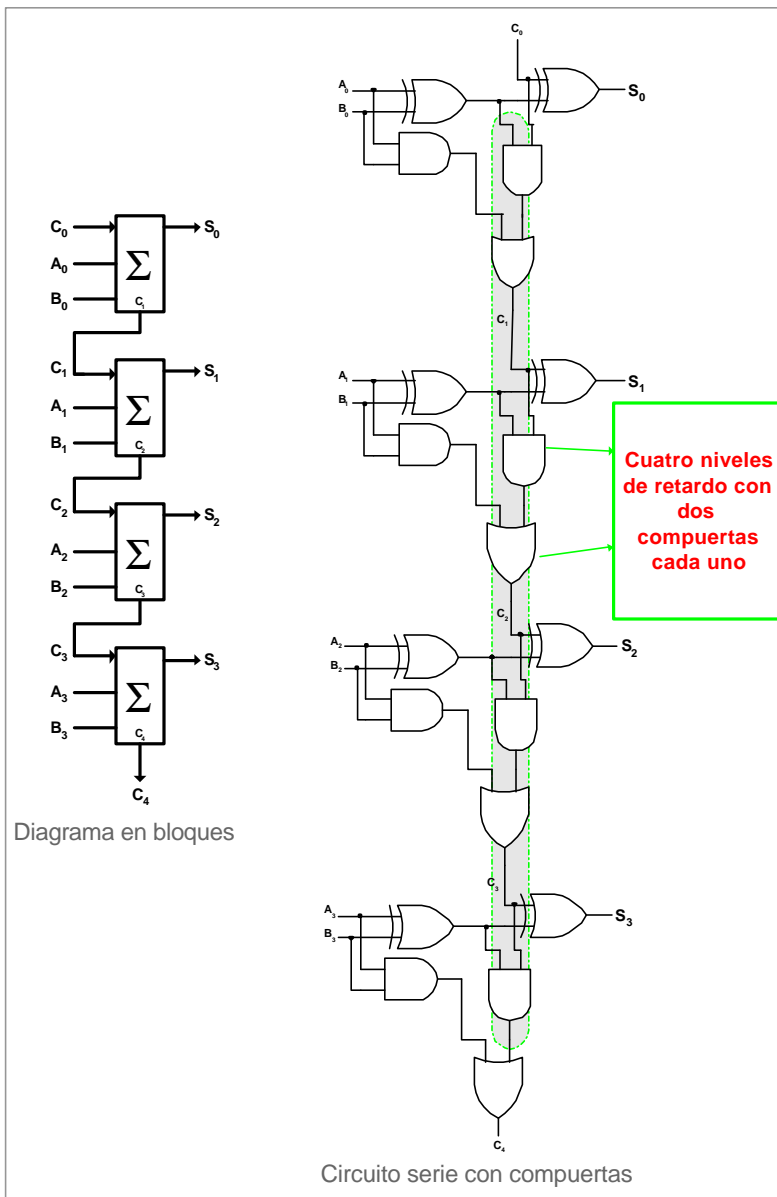


Figura 5.37. Circuito sumador serie de cuatro bits.

Existen también sumadores paralelos con acarreo anticipado (CLA) con técnicas de paralelismo doble en los CLA que son utilizados cuando es necesario diseñar un sumador con capacidad de 8, 16, 32 y 64 bits. El libro de *Principios de diseño digital de Daniel D. Gajski* tiene los fundamentos teóricos para realizar este tipo de acoplamiento.

5.4.2.1 Circuito sumador paralelo MSI 7483.

Existen circuitos integrados sumadores de la familia TTL que pueden sumar dos datos binarios de cuatro bits cada uno en forma paralela; a su vez, cada chip puede ser acoplado a través de los acarrees de entrada y salida para realizar expansión de los bits del circuito sumador. Los circuitos 7483 y 74283 son equivalentes y realizan ésta operación aritmética de cuatro bits. Los sumandos son: $A_3A_2A_1A_0$, $B_3B_2B_1B_0$ y el acarreo de entrada C_0 que es el bit menos significativo; la salida del chip se obtiene en $S_3S_2S_1S_0$ además del bit más significativo de la suma llamado acarreo de salida C_4 . La figura 5.38 muestra el diagrama del circuito integrado sumador paralelo de cuatro bits 7483; el resultado máximo de la suma se obtiene cuando todos los bits de las entradas valen uno: $A_3A_2A_1A_0 = 1111$; $B_3B_2B_1B_0 = 1111$ y $C_0 = 1$. Este resultado es 31 en binario: $C_4 = 1$; $S_3S_2S_1S_0 = 1111$.



Figura 5.38. Sumador paralelo de cuatro bits 7483.

5.4.2.2 Circuito de acarreo anticipado MSI 74182.

La familia TTL también posee un chip con lógica de acarreo anticipado (CLA) para dos sumandos de cuatro bits cada uno; este circuito es el 74182. Las entradas G_i y P_i son equivalentes a K_i y M_i respectivamente explicados en el tema anterior, y los acarrees: C_n , C_{n+x} , C_{n+y} , C_{n+z} también corresponden con C_0 , C_1 , C_2 , C_3 .

La figura 5.39 muestra el diagrama del circuito integrado CLA 74182, éste posee salidas **G** y **P** que sirven para realizar expansiones de 8, 16, 32 y más bits mediante el acoplamiento de varios chips de este tipo.

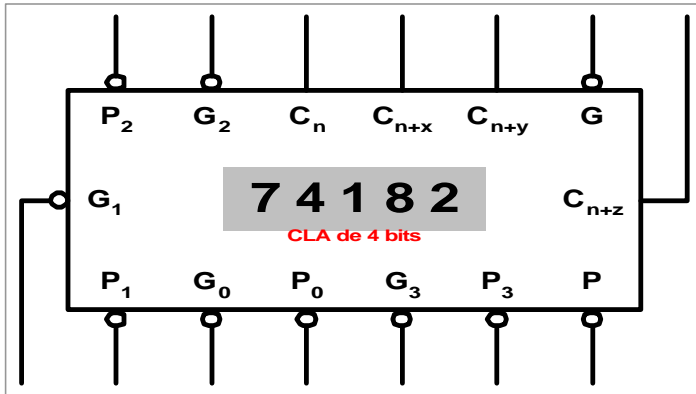


Figura 5.39. Circuito de acarreo anticipado de cuatro bits 74182.

5.4.3 Aplicaciones de los circuitos sumadores 7483 y 74182.

Los circuitos integrados MSI 7483 y 74182 sirven para sumar datos binarios de cuatro y más bits; también, agregando algunos dispositivos y compuertas digitales en el circuito, se pueden obtener restadores, comparadores o convertidores de código numérico. Con dos o más chips 7483 se hacen expansiones superiores a cuatro bits en el tamaño de los datos a ser procesados, formando circuitos acoplados en cascada. Las expansiones realizadas con el 74182 se implementan utilizando la técnica de acarreo anticipado obteniendo menor tiempo de respuesta en el procesamiento de los datos.

A continuación se muestran algunas de éstas aplicaciones con los integrados descritos anteriormente.

5.4.3.1 Convertidor de código con sumadores MSI.

Una aplicación del 7483 es la conversión del código BCD natural en Exceso 3; se implementa colocando los dos bit menos significativos de **A** en uno lógico. Esto equivale a sumar tres en el dato BCD que entra por **B**. La figura 5.40 muestra este circuito con entradas BCD igual a $N_3N_2N_1N_0$.

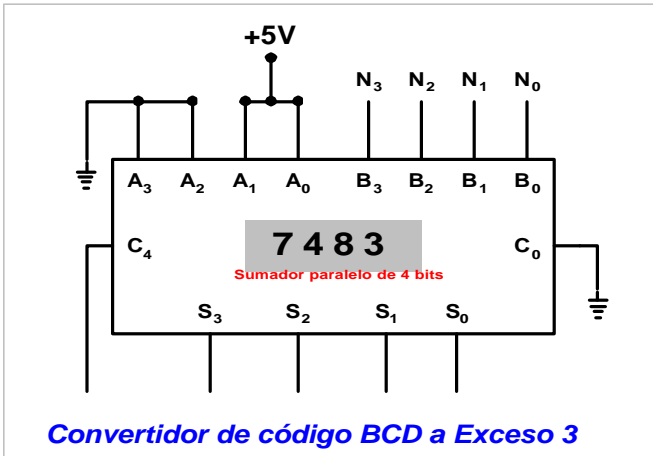


Figura 5.40. Convertidor de código BCD –Exceso 3 con el 7483.

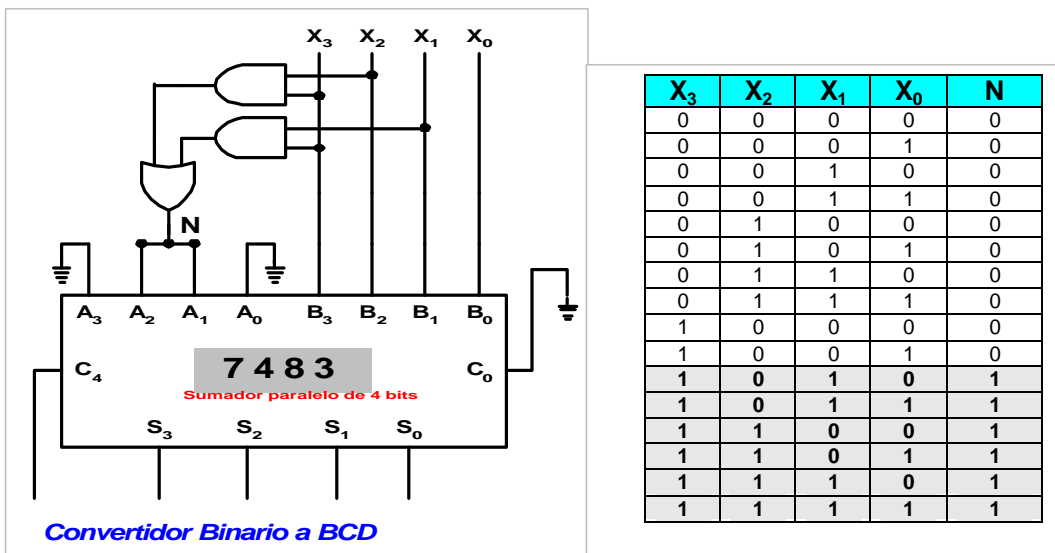


Figura 5.41. Convertidor de código binario de cuatro bits a BCD.

La salida del circuito de la figura 5.41 debe ser tomada desde C_4 como bit más significativo del código BCD y $S_3S_2S_1S_0$ como los cuatro bits del grupo menos significativo BCD. De este modo, la señal **N** es alta solo cuando X_3 y X_2 son uno lógico ó cuando respectivamente X_3 y X_1 , también lo son; las combinaciones diferentes a éstas indican que la entrada está en el rango de 0 a 9 por lo que **N** es baja y por lo tanto no se suma el factor de corrección seis al dato de entrada.

5.4.3.2 Circuito sumador y restador con el 7483.

La figura 5.42 muestra un circuito restador de cuatro bits implementado mediante la técnica de la suma del complemento a dos, visto en el Capítulo I. Mediante esta operación se obtiene, a la salida del 7483, el resultado de la resta. El fundamento de la implementación se basa en la fórmula: $S = X - Y = X + \bar{Y} + 1$. La expresión $\bar{Y} + 1$ es el complemento a dos de **Y**; que se logra con las compuertas inversoras, más el uno lógico en la línea de acarreo de entrada C_0 .

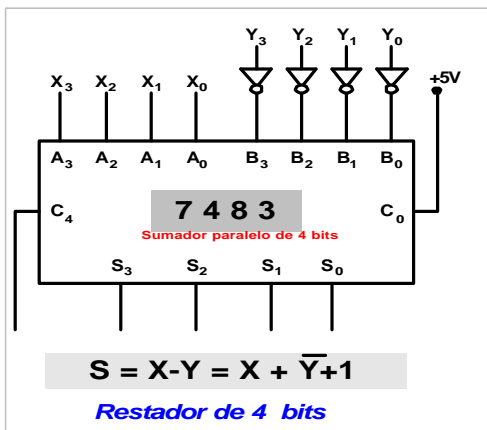


Figura 5.42. Restador paralelo de cuatro bits.

El acarreo de salida C_4 y el dato de cuatro bits del resultado $S_3S_2S_1S_0$ pueden presentar las siguientes condiciones mostradas en la tabla 5.7:

C_4	$S_3S_2S_1S_0$	Entradas: $X_3X_2X_1X_0$ y $Y_3Y_2Y_1Y_0$
1	Resultado mayor o igual a cero en binario normal	$X \geq Y$
0	Resultado negativo en complemento a dos	$X < Y$

Tabla 5.7. Condiciones de entrada /salida del circuito restador.

La figura 5.43 muestra un circuito que complementa a dos el dato que entra por $Z_3Z_2Z_1Z_0$. Las cuatro compuertas OR-exclusivas invierten el dato $Z_3Z_2Z_1Z_0$ cuando la señal C tiene un nivel lógico alto; al mismo tiempo C_0 recibe también un nivel alto lo que determina que el resultado sea complemento a dos del dato de entrada. La otra entrada del 7483 está cableada a cero por lo que solamente en el circuito de salida queda el resultado del cambio de signo. La fórmula aplicada es la siguiente: $A - Z = 0 - Z = -Z = \bar{Z} + 1$. En la figura 5.44 se muestra un Sumador-Restador binario de cuatro bits con corrector de resultado negativo en complemento a dos.

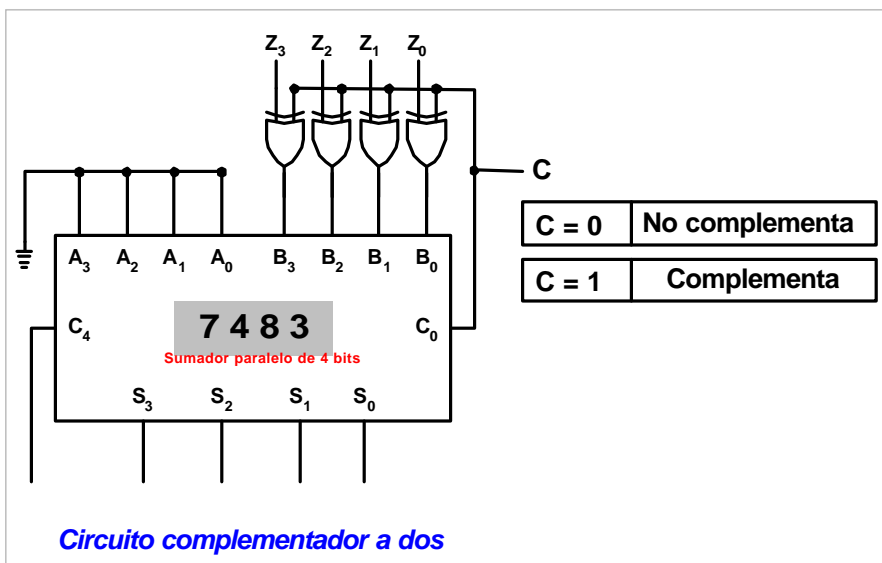


Figura 5.43. Circuito para obtener el complemento a dos de Z.

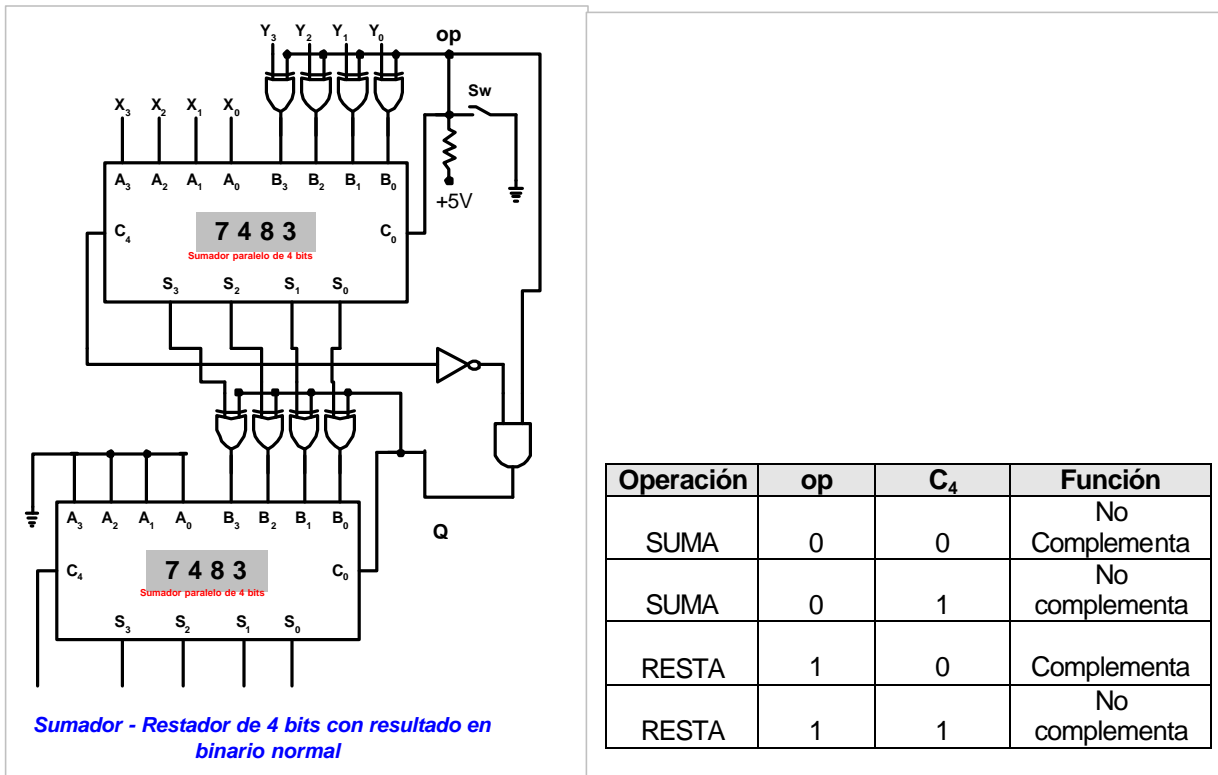


Figura 5.44. Sumador restador de cuatro bits con corrección de complemento a dos.

El **Sw**, se abre para que **op** tenga un nivel alto; cuando **C₄**, del primer sumador, tiene un nivel bajo, **Q** se coloca en alto indicando que $X < Y$, por lo que el resultado, de la resta del primer 7483 será negativo y estará complementado a dos. El segundo chip 7483 se encarga de complementar de nuevo el dato con lo cual es transformado en binario normal.

5.4.3.3 Expansión de sumadores con el 7483 y 74182.

Las operaciones aritméticas de cuatro bits no satisfacen las necesidades de los sistemas de desarrollo, sistemas de computo, etc. Se necesitan resultados más amplios en número de bits 8, 16, 32 y hasta 64 son requeridos por los sistemas y computadoras actuales. De esta forma, se hace necesario la expansión de bits en los circuitos realizados con chips sumadores; sin sacrificar la velocidad de transferencia de información entre los distintos dispositivos y circuitos integrados.

La figura 5.45 muestra un circuito sumador de 12 bits con tres 7483 acoplados en cascada; donde el acarreo de salida C_4 de un chip se une con el acarreo de entrada C_0 del siguiente. La salida posee 12 bits ($Z_{11} \dots Z_0$) más el acarreo C_{12} . Las entradas del sumador son: ($M_{11} \dots M_0$) y ($N_{11} \dots N_0$) respectivamente.

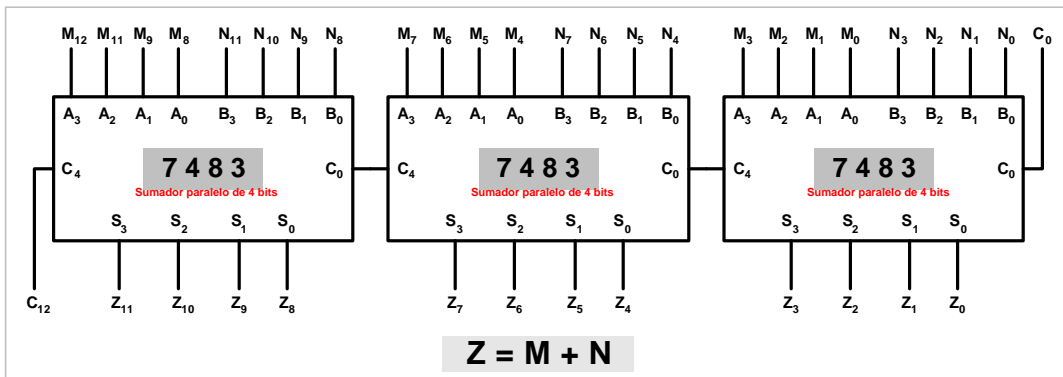


Figura 5.45. Sumador de 12 bits con tres 7483 acoplados en cascada.

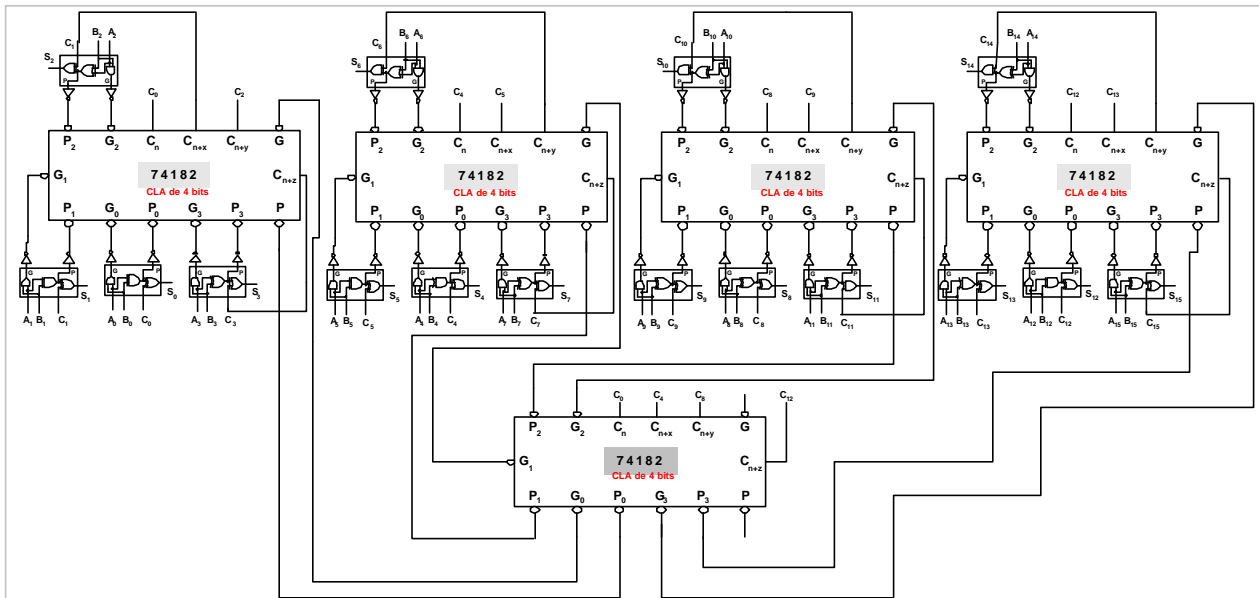


Figura 5.46. Sumador de 16 bits con dos niveles CLA utilizando cinco 74182 y compuertas.

La figura 5.46 muestra una expansión de 16 bits, con la técnica de acarreo anticipado utilizando para ello cinco chips 74182, compuertas AND y OR-exclusivas.

El circuito tiene dos niveles de lógica en el CLA, sin embargo, no posee acarreo de salida C16; éste último puede obtenerse colocando otro nivel CLA con un circuito integrado 74182. La ventaja de este circuito es la velocidad con que se ejecutan las operaciones aritméticas.

Ejercicio 5.18. Diseñe con el 7483 un sumador de dos datos BCD de cuatro bits cada uno; el resultado debe estar expresado en BCD natural.

Ejercicio 5.19. Diseñe con el 7483 un comparador de dos datos binario A y B de cuatro bits cada uno; la salida debe tener tres indicaciones $A > B$, $A = B$ y $A < B$.

Ejercicio 5.20. Diseñe con el 7483 un sumador restador de dos datos de ocho bits cada dato; la salida debe tener el resultado en binario normal e indicar el signo menos con el encendido de un led.

Ejercicio 5.21. Realizar el esquema de un circuito sumador CLA utilizando el 74182. El circuito debe sumar datos de 32 bits.

Ejercicio 5.22. Diseñe con el 7483 un sumador que muestre en displays 7 segmentos el resultado de la operación en decimal.

Ejercicio 5.23. Implementar un convertidor de código que convierta datos de cuatro bits en AIKEN a binario normal.

Ejercicio 5.24. Diseñe un restador de seis bits; el resultado debe estar en binario normal y con señalización de signo negativo.

PRÁCTICA DE LABORATORIO #7

TITULO: Circuitos combinacionales Sumadores.

OBJETIVO: El estudiante al terminar esta práctica estará en capacidad de poder analizar y diseñar las aplicaciones de los circuitos combinacionales aritméticos tales como sumadores y restadores.

INTRODUCCIÓN: Los circuitos integrados sumadores tienen aplicaciones en los circuitos aritméticos sumadores y restadores, circuitos convertidores de código y comparadores de datos binarios. El chip sumador 7483 o 74283 es utilizado en esta práctica para realizar dos montajes: el primero es un sumador restador paralelo de cuatro bits y el segundo es un circuito que transforma un número binario de cuatro bits en BCD. Se recomienda repasar los temas 1.5 y 5.4 de este material, consultar la bibliografía citada al final de la guía y estudiar las características de los chips sumadores en un manual TTL.

PRELABORATORIO: Investigar los siguientes tópicos.

- Funcionamiento de los chips 7482, 7483, 74182 y 74283.
- Circuitos sumadores de acarreo anticipado (CLA).
- Aplicaciones generales de los sumadores.
- Sumadores serie y paralelo.

MATERIALES Y EQUIPOS NECESARIOS:

- Dos chips 7483 o dos 74283 y otros chips más dependiendo del diseño.
- Dos chips 7447 o 7448, dos displays siete segmentos y varios diodos leds.
- Compuertas digitales de acuerdo a los diseños realizados.
- Protoboard, cable telefónico, pinza, piqueta.
- Multímetro digital y fuente de 5 Volt / 2 Amp.

DESARROLLO:

1. Diseñar e implementar un sumador - restador con el chip 7483 y compuertas digitales que visualice con diodos leds, en la salida, el resultado de la operación. El circuito debe tener un **Sw** que conmute la operación aritmética del siguiente modo: (Suma --> Sw=0) y (Resta --> Sw=1).
2. Realizar un circuito que transforme un valor binario de cinco bits de entrada en un código normal BCD. El valor equivalente debe ser mostrado en displays siete segmentos y debe ser visualizado hasta el número "19". Utilizar para esto un chip 7483 o 74283 más las compuertas necesarias.

POST-LABORATORIO.

- Explicar con tablas y diagramas, el funcionamiento de cada uno de los montajes.
- Realizar cuadro comparativo ventajas y desventajas de los sumadores con acoplamiento en cascada y acoplamiento paralelo.
- Hacer expansiones de circuitos sumadores en cascada (serie) y paralelo.
- Diseñar algunos convertidores de código utilizando el chip 7483.

MONTAJES ALTERNATIVOS:

1. Realizar el montaje de un restador de ocho bits utilizando dos chips 7483.
2. Implementar un sumador – restador de cuatro bits donde se puedan visualizar en displays 7 segmentos los verdaderos valores de las operaciones aritméticas con signo. Este último puede ser implementado con un led.

3. Implementar con el 7483 un circuito que pueda complementar a dos un dato entrante de ocho bits.
4. Diseñar e implementar con 7483 o 74283 un circuito digital que permita convertir un código entrante BCD de cinco bits en código binario normal.
5. Implementar un sumador / restador de 4 bits con indicador de signo, que pueda detectar y señalar cuando exista "overflow" en la operación.

BIBLIOGRAFÍA.

- CUESTA, Luís M. PADILLA G, Antonio. REMIRO D, Fernando. (1993). Electrónica digital. Madrid: McGraw Hill. S/f. p.445.
- GAJSKI, Daniel D. (1997). Principios de diseño digital. Madrid: Prentice Hall Iberia. S/f. p.488. "Principles of digital design". Traducido por: Alberto Prieto Espinosa.
- LLORIS, Antonio. PRIETO, Alberto. (1996). Diseño lógico. Madrid: McGraw Hill. S/f. p.403.
- MANDADO, Enrique. (1987). Sistemas electrónicos digitales. Barcelona (España): Marcombo Boixareu Editores. Sexta edición. p.705.
- MANO, Morris. KIME, Charles. (1998). Fundamentos de diseño lógico y computadoras. México: Prentice Hall. Primera edición en español. P.604. "Logic and computer design fundamentals". Traducido por: Teresa Sanz Falcón.
- NELSON, V. NAGLE, H. CARROLL, B. IRWIN, J. (1996). Análisis y diseño de circuitos lógicos digitales. México: Prentice Hall. Primera edición. p.842. "Digital logic circuit analysis and design". Traducido por: Oscar A. Palmas V.
- TOCCI, Ronald. (1995). Sistemas digitales principios y aplicaciones. México: Prentice Hall. Quinta edición. p.823. "Digital systems principles and applications". Traducido por: Edmundo G. Urbina M.
- WARKELY, John F. (1997). Diseño digital principios y prácticas. México: Prentice Hall. S/f. p.743. "Digital design principles and practices". Traducido por: Gutiérrez R. Raymundo H.

MANUALES.

- NATIONAL SEMICONDUCTOR. (1981). Manual TTL y CMOS.
- MOTOROLA Inc. (1992). Fast and LS TTL. (DL121/D REV. 5).

5.5 Circuitos digitales comparadores.

El comparador digital es un circuito combinacional que compara dos datos **A** y **B** de **n** bits cada uno y genera tres resultados en la salida: $F_{A>B}$, $F_{A=B}$ y $F_{A<B}$. La figura 5.47 y 5.48 muestran un comparador de dos bits realizado con compuertas digitales, la tabla de la verdad y la simplificación mediante mapas de Karnaugh. El bloque comparador debe entregar combinaciones distintas para indicar el resultado de la comparación; por ejemplo, las condiciones del comparador de dos bits en la salida son:

$F_{A>B}$	$F_{A=B}$	$F_{A<B}$	Resultado
0	0	1	A<B
0	1	0	A=B
1	0	0	A>B
0	0	0	Todas en Hi-Z
1	1	1	Todas en Hi-Z

Tabla 5.8. Resultado de las salidas del comparador genérico.

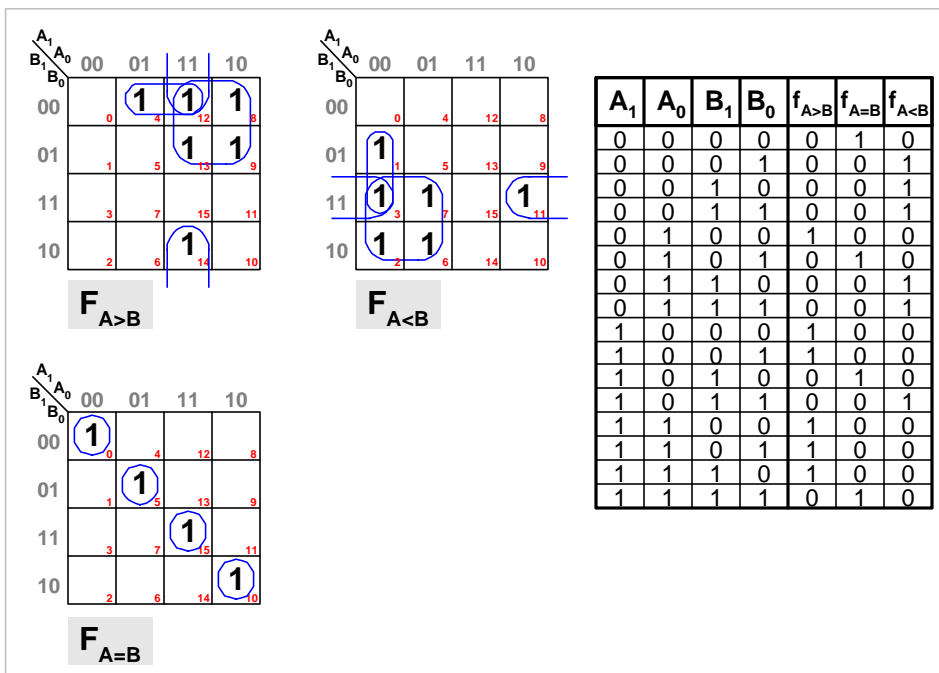


Figura 5.47. Tabla de la verdad y mapas K para diseñar el circuito comparador.

Con la tabla de la verdad se generan las siguientes funciones:

$$f_{A>B}(A_1, A_0, B_1, B_0) = \sum_m (1, 2, 3, 6, 7, 11)$$

$$f_{A=B}(A_1, A_0, B_1, B_0) = \sum_m (0, 5, 10, 15)$$

$$f_{A<B}(A_1, A_0, B_1, B_0) = \sum_m (4, 8, 9, 12, 13, 14)$$

Realizando las simplificaciones respectivas, con los grupos formados, en los tres mapas de Karnaugh las funciones quedan reducidas así:

$$f_{A>B} = A_1 A_0 \overline{B_0} + A_0 \overline{B_1} \overline{B_0} + A_1 \overline{B_1}$$

$$f_{A=B} = \overline{A_1} \overline{A_0} \overline{B_1} \overline{B_0} + A_1 \overline{A_0} B_1 \overline{B_0} + \overline{A_1} A_0 \overline{B_1} B_0 + A_1 A_0 B_1 B_0$$

$$f_{A=B} = (\overline{A_1} \overline{B_1} + A_1 B_1) \overline{A_0} \overline{B_0} + (\overline{A_1} \overline{B_1} + A_1 B_1) A_0 B_0$$

$$f_{A=B} = (\overline{A_1} \oplus \overline{B_1}) \overline{A_0} \overline{B_0} + (\overline{A_1} \oplus \overline{B_1}) A_0 B_0$$

$$f_{A=B} = (\overline{A_1} \oplus \overline{B_1}) (\overline{A_0} \oplus \overline{B_0})$$

$$f_{A=B} = (\overline{A_1} \oplus \overline{B_1}) + (A_0 \oplus B_0)$$

$$f_{A<B} = \overline{A_1} \overline{A_0} B_0 + \overline{A_0} B_1 B_0 + \overline{A_1} B_1$$

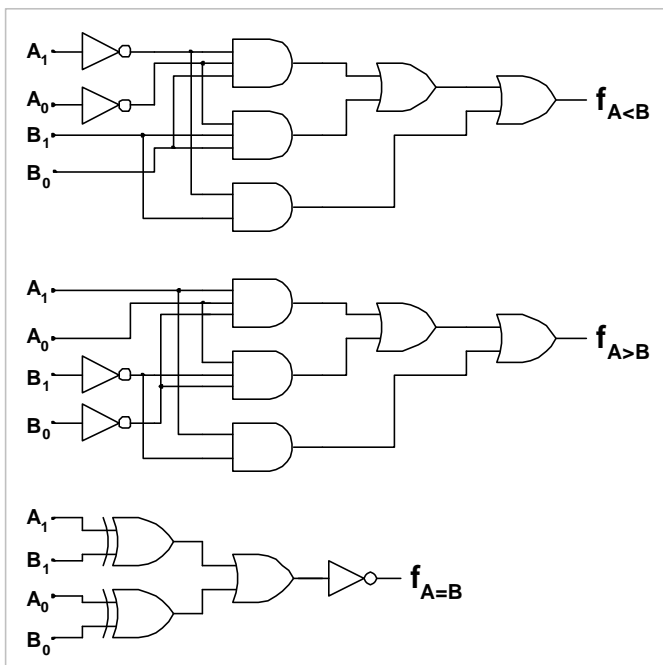


Figura 5.48. Comparador de dos bits realizado con compuertas.

5.5.1 Circuito integrado comparador 7485.

Es un chip que compara dos datos de cuatro bits cada uno y genera a la salida una combinación binaria de tres líneas. Los datos de entrada $A_3A_2A_1A_0$ y $B_3B_2B_1B_0$ se comparan y activan alguna de las líneas de salida $f_{A>B}$, $f_{A=B}$ o $f_{A<B}$; además de esto posee tres líneas de entrada $I_{A>B}$, $I_{A=B}$ e $I_{A<B}$ que sirven para realizar expansiones, utilizando dos o más chips 7485. La figura 5.49 muestra el diagrama del circuito integrado comparador 7485 y la tabla resumida de funcionamiento del mismo.

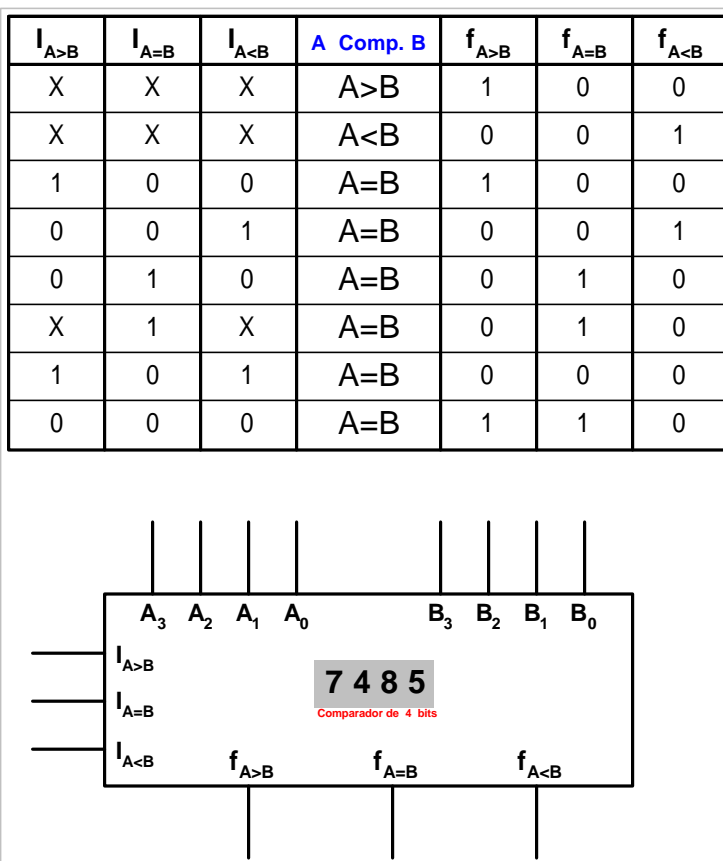


Figura 5.49. Tabla de funcionamiento y esquema del chip 7485.

Las entradas $I_{A>B}$, $I_{A=B}$ e $I_{A<B}$ son menos significativas que las entradas A y B del chip; por lo cual se deben realizar expansiones en cascada tomando los bits acoplados en estas líneas como menos significativos. La figura 5.50 muestra una expansión en cascada realizada con este circuito integrado para formar un comparador de ocho bits con entradas que van desde X_0 hasta X_7 y Y_0 hasta Y_7 .

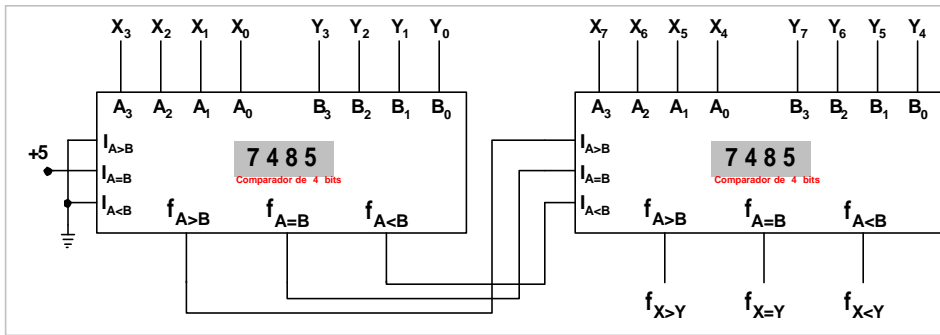


Figura 5.50. Comparador de 8 bits realizado con dos 7485 en cascada.

5.5.2 Aplicaciones de los circuitos comparadores.

Los sistemas de computación y procesamiento de datos toman decisiones de bifurcación, saltos y bucles cuando comparan dos o más datos. Las unidades de control también utilizan los circuitos comparadores como acción fundamental en las instrucciones de microprogramas de hardware. Las aplicaciones de este circuito son muy diversas y pueden ir desde una simple comparación de dos datos de cuatro bits hasta circuitos complejos de controladores digitales discretos. La figura 5.51 muestra una expansión realizada con tres 7485 para implementar un comparador paralelo de doce bits.

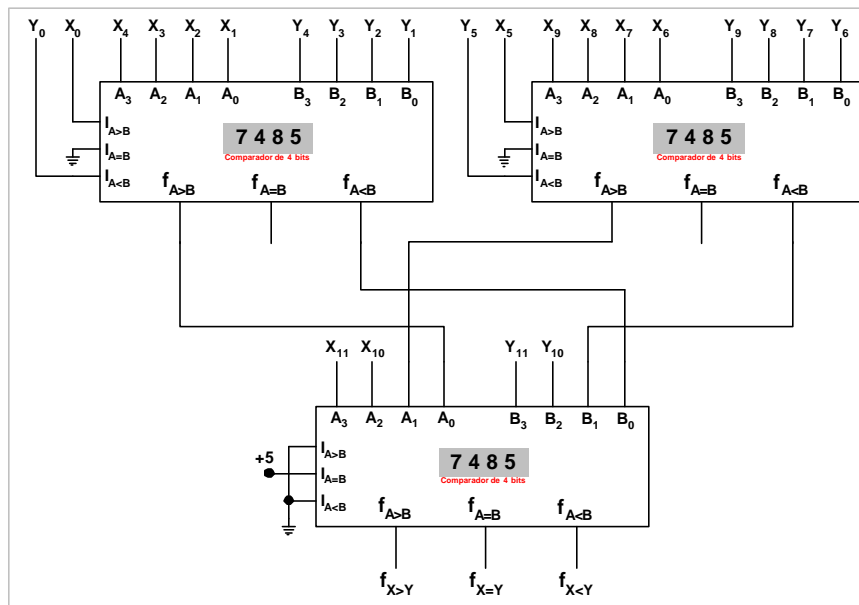


Figura 5.51. Comparador paralelo de doce bits.

Un circuito que se puede implementar con el 7485 es un comparador que simula el juego aleatorio con 16 valores numéricos posibles por participante, el diseño debe mostrar el valor de quién gana o pierde. La figura 5.52 muestra la solución de este problema.

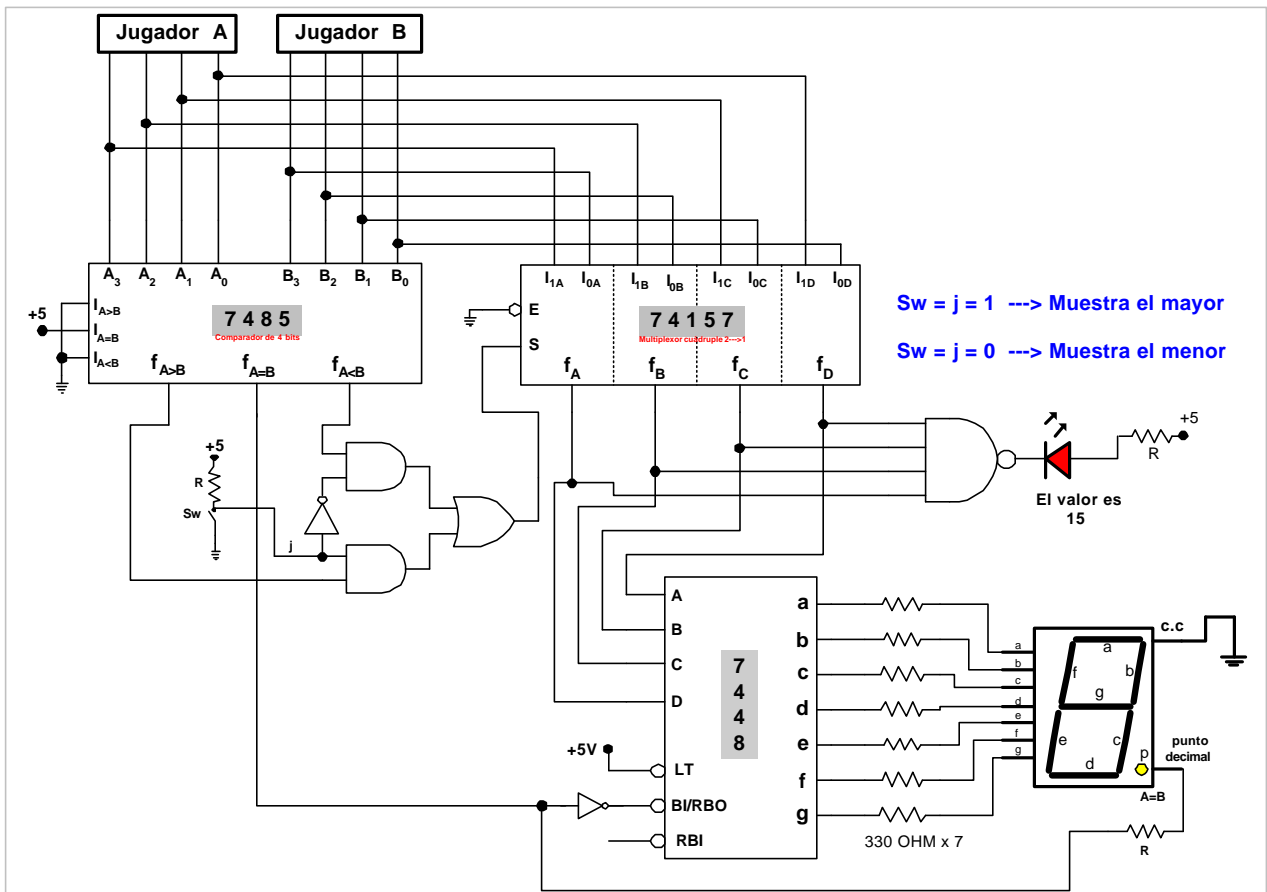


Figura 5.52. Circuito que muestra el ganador en una jugada con 16 valores por jugador.

El circuito multiplexor 74157 selecciona cual de los dos jugadores tiene el valor mayor ó menor; esto depende de la posición de **Sw**. Si **j=0** entonces el valor que se muestra en el display es el menor de los dos; por el contrario, si **j=1** se verá en él siete segmentos el resultado mayor. Los valores que pueden colocar los jugadores van desde cero hasta quince; sin embargo, para visualizar esto el led rojo se enciende. Por otra parte, el led del punto decimal enciende cuando las jugadas son iguales. Cada jugada puede ser simulada por dos contadores binarios independientes con start / stop cada uno.

La figura 5.53 muestra un comparador de cuatro bits implementado con el sumador 7483, configurado como restador, y la mitad del decodificador 74139. Las compuertas OR colocadas a la salida del sumador permiten detectar la igualdad entre **M** y **N** colocando en nivel alto la entrada **B** del 74139; ésto permite diferenciar la condición de mayor o igual cuando **C₄** vale uno lógico.

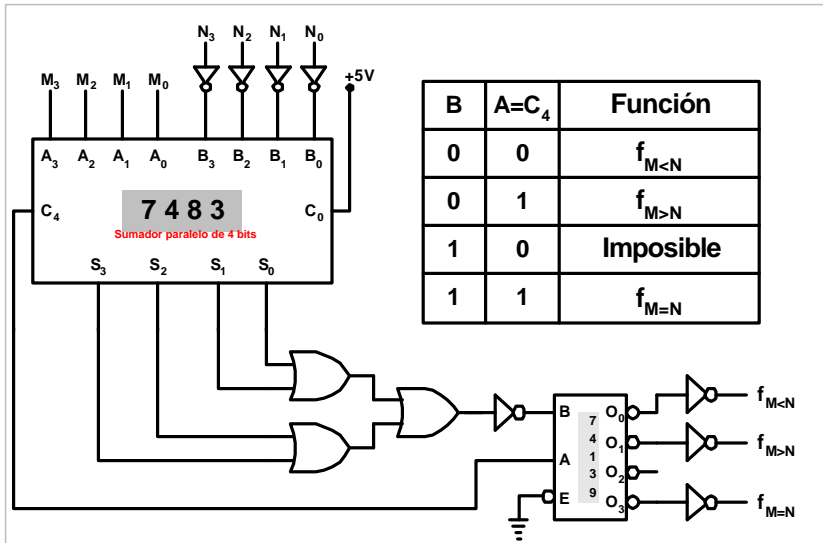


Figura 5.53. Comparador de 4 bits realizado con 7483, 74139 y compuertas.

Ejercicio 5.25. Implementar un comparador de dos bits por dato X_1X_0 e Y_1Y_0 con tres entradas ($I_{X>Y}$, $I_{X=Y}$, $I_{X<Y}$) para expansión.

Ejercicio 5.26. Diseñar con el circuito integrado 7485 un sistema digital que compare tres datos de cuatro bits cada uno.

Ejercicio 5.27. Diseñar un circuito que muestre el resultado de dos jugadores cuando lanzan los dados aleatoriamente.

Ejercicio 5.28. Implementar comparadores serie y paralelo de dos datos con la siguiente cantidad de bits por dato: cinco, seis, diez y veinticuatro.

PRÁCTICA DE LABORATORIO #8

TITULO: Circuitos combinacionales Comparadores de magnitud.

OBJETIVO: El estudiante al terminar esta práctica estará en capacidad de poder analizar y diseñar las aplicaciones de los circuitos digitales combinacionales comparadores de magnitud.

INTRODUCCIÓN: Los circuitos integrados comparadores tienen aplicaciones en los circuitos de control y bifurcación de datos. Estos circuitos son efectivos en la toma de decisiones; donde es muy importante realizar primero una comparación, y luego la acción a seguir. El chip 7485 permite comparar dos datos de cuatro bits e indicar en alguna de sus tres salidas el estatus de la operación obteniéndose, una indicación del resultado mayor, igual o menor. La práctica consta de dos montajes: el primer montaje es un comparador de tres datos de cuatro bits cada dato y el segundo montaje es un comparador de cinco bits a partir de un solo chip 7485. La bibliografía necesaria para realizar este laboratorio se encuentra al final de la guía y en el capítulo 5 de la misma.

PRELABORATORIO: Investigar los siguientes tópicos.

- Funcionamiento del chip 7485 y equivalente.
- Diseño de comparadores con compuertas.
- Expansiones con chips comparadores.
- Uso del chip 7483 como comparador y Aplicaciones generales de los comparadores.

MATERIALES Y EQUIPOS NECESARIOS:

- Tres chips 7485 o 7483 como opcional, ocho diodos leds.
- Compuertas y/o chips combinacionales de acuerdo a los diseños realizados.
- Protoboard, cable telefónico, pinza, piqueta.

- Multímetro digital y fuente de 5 Volt / 2 Amp.

DESARROLLO:

1. Implementar un circuito que compare tres datos (A, B, C) de cuatro bits cada uno. El circuito debe indicar con diodos leds el dato mayor y por otra parte, señalar el momento cuando los tres valores son iguales ($A=B=C$).

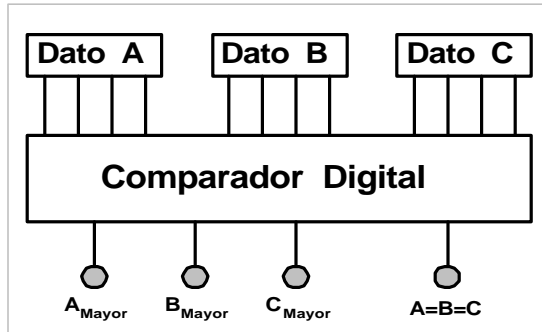


Diagrama en bloques del comparador de tres datos.

2. Realizar un circuito que compare dos datos de cinco bits cada uno. El diseño se debe realizar con un solo chip 7485.

POST-LABORATORIO.

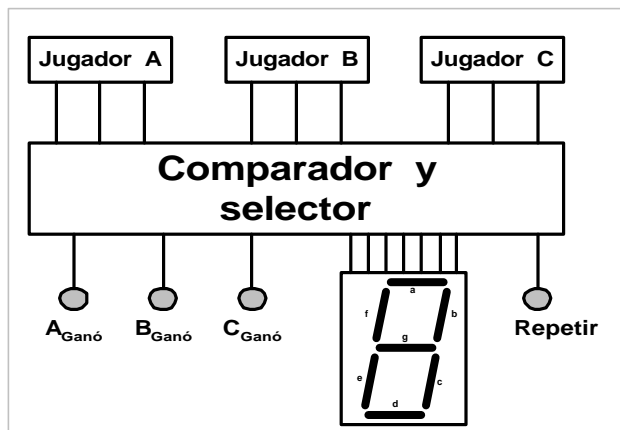
- Explicar con tablas y planos cada uno de los montajes realizados.
- Realizar comparadores con compuertas básicas y exclusivas.
- Hacer expansiones de circuitos comparadores en cascada (serie) y paralelo.
- Diseñar un comparador con el chip 7483 y compuertas.

MONTAJES ALTERNATIVOS:

1. Implementar con circuitos integrados 7483, chips combinacionales y compuertas el montaje de un comparador de dos datos de ocho bits cada uno. El circuito debe tener tres salidas ($F_{A>B}$, $F_{A=B}$, $F_{A<B}$) sin entradas de expansión.

2. Implementar un juego de dados de tres jugadores donde se pueda visualizar al ganador y el valor numérico de la jugada en displays. El valor mayor gana, de modo que, el circuito debe señalar quién ganó e indicar con que cantidad numérica lo hizo. No obstante, en caso de que el valor mayor sea doble empate se debe repetir la jugada e igual para el caso de que haya triple empate. Las entradas de los tres jugadores pueden ser colocadas con DIP-SW o contadores binarios independientes.

Esquema en bloques del juego de dados.



3. Realizar el montaje de un comparador de 9 bits con dos chips 7485.
4. Diseñar e implementar un comparador de dos datos de tres bits cada dato utilizando decodificadores y compuertas digitales.

BIBLIOGRAFÍA.

- CUESTA, Luís M. PADILLA G, Antonio. REMIRO D, Fernando. (1993). Electrónica digital. Madrid: McGraw Hill. S/f. p.445.
- MANDADO, Enrique. (1987). Sistemas electrónicos digitales. Barcelona (España): Marcombo Boixareu Editores. Sexta edición. p.705.
- NELSON, V. NAGLE, H. CARROLL, B. IRWIN, J. (1996). Análisis y diseño de circuitos lógicos digitales. México: Prentice Hall. Primera edición. p.842. "Digital logic circuit analysis and design". Traducido por: Oscar A. Palmas V.
- WARKELY, John F. (1997). Diseño digital principios y prácticas. México: Prentice Hall. S/f. p.743. "Digital design principles and practices". Traducido por: Gutiérrez R. Raymundo H.

MANUALES.

- NATIONAL SEMICONDUCTOR. (1981). Manual TTL y CMOS.
- MOTOROLA Inc. (1992). Fast and LS TTL. (DL121/D REV. 5).

5.6 Circuitos generadores y detectores de paridad.

En los sistemas de transmisión y recepción de datos digitales es necesario comprobar errores en la información enviada. Esto se realiza chequeando los bits que forman el dato de manera tal que el receptor pueda detectar y/o corregir si hubo cambio en uno o más bits. Los métodos para lograrlo son diversos; no obstante, en este tema se van a considerar dos: el método de generación - chequeo de paridad par e impar de un bit y el método de corrección-detección Hamming de uno y dos bits respectivamente. En el capítulo I se explica el fundamento teórico necesario para este tema por lo que se comenzará directamente con el diseño de los circuitos y descripción de chips.

5.6.1 Método de generación y chequeo de paridad de un bit.

Este método consiste en generar un bit con paridad par o impar en el dato transmitido y luego, en el receptor, chequear que la suma de los bits, que forman el dato, esté de acuerdo con la paridad prefijada por el transmisor y receptor. La figura 5.54 muestra un circuito de generación y chequeo de tres bits realizado con compuertas según la siguiente tabla de la verdad:

X_2	X_1	X_0	F_p	F_i
0	0	0	0	1
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	0

Tabla 5.9. Tabla de la verdad de un generador de paridad de tres bits.

$$F_p(X_2, X_1, X_0) = \sum_m (1, 2, 4, 7)$$

$$F_i(X_2, X_1, X_0) = \sum_m (0, 3, 5, 6)$$

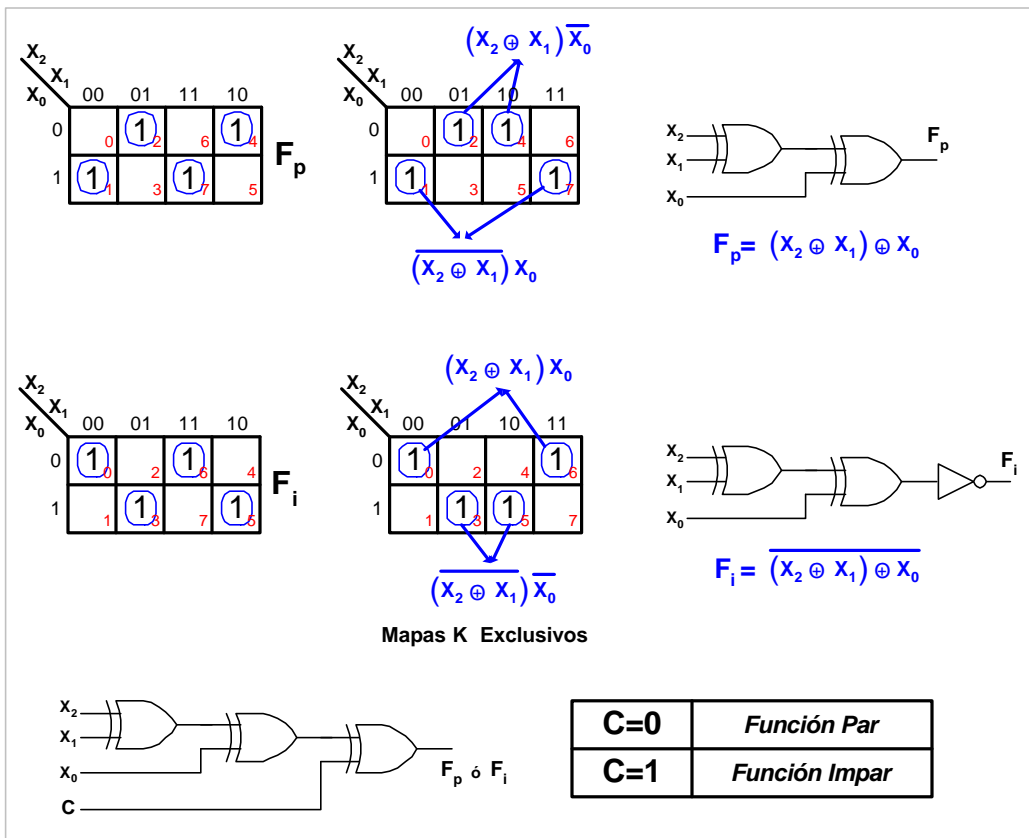


Figura 5.54. Generador– chequeador de paridad de tres bits con compuertas.

Las compuertas OR-Exclusivas resuelven el problema de los mapas K cíclico y por lo tanto se obtiene, un circuito digital reducido; el circuito se expande en número de bits acoplando más compuertas exclusivas. La señal C , en cero lógico, permite generar paridad par y, paridad impar con C igual a uno lógico. Este mismo circuito es utilizado como receptor para chequear la paridad de un dato; es necesario acoplar dos circuitos de éstos para tener un sistema completo de generación y chequeo de paridad par o impar. La figura 5.55 muestra un sistema generador y chequeador de paridad de tres bits realizado con compuertas digitales, el tipo de paridad par e impar puede ser seleccionada cerrando o abriendo Sw_1 y Sw_2 , tanto en el generador como en el detector. En el circuito detector se ha agregado una compuerta OR-Exclusiva para seleccionar el tipo de paridad; además de esto la figura también presenta el diagrama en bloques del sistema de generación y detección de paridad de tres bits.

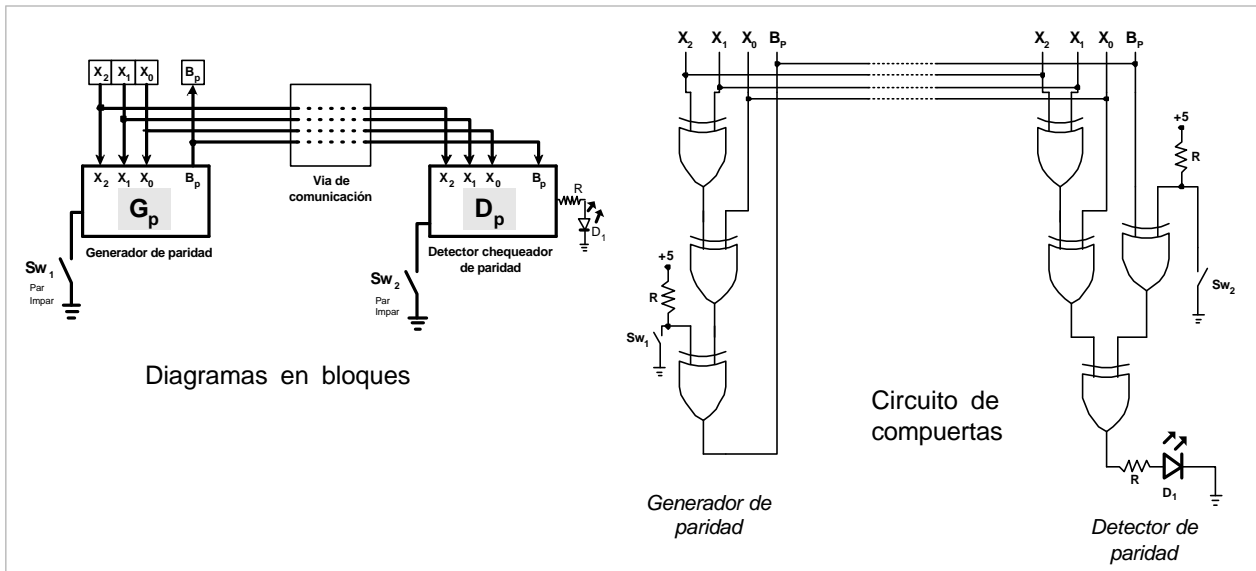


Figura 5.55. Diagrama en bloques y compuertas del generador y detector de paridad de tres bits.

5.6.2 Generador y detector de paridad 74180 y 74280.

Es un circuito integrado generador y detector de paridad par e impar con ocho bits de entrada (A, B, C, D, E, F, G, H); dos entradas que sirven para configurar el tipo de paridad (I_{even} , I_{odd}) y dos líneas de salida (Σ_{even} , Σ_{odd}). Sirve para transmitir un byte de información más el bit de paridad; donde, el valor par (even) e impar (odd) está determinada por la tabla de funcionamiento del chip, dado en la figura 5.56.

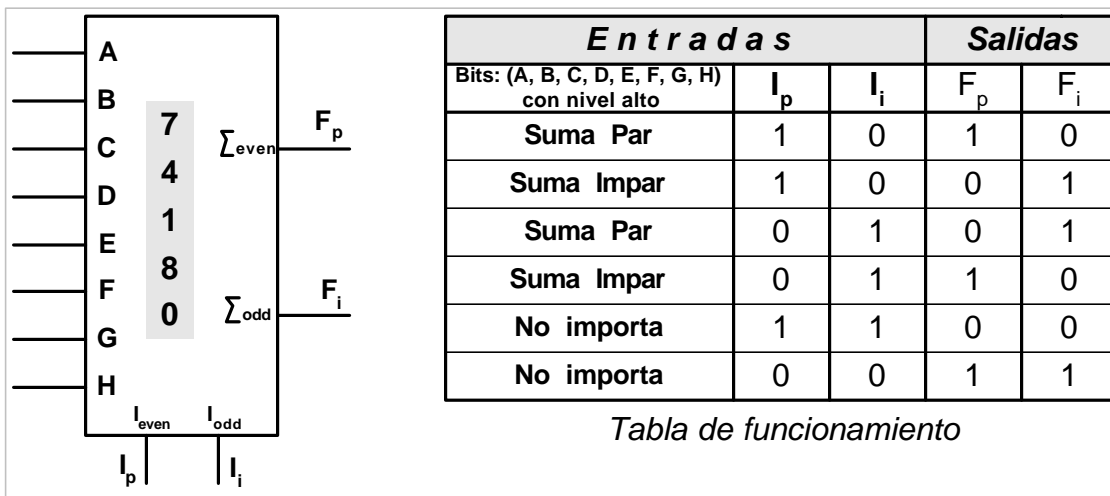


Figura 5.56. Descripción del chip generador y detector de paridad de 9 bits 74180.

El circuito integrado 74280 funciona en forma equivalente al 74180; sin embargo, posee nueve entradas (A, B, C, D, E, F, G, H, I) donde una de ellas (por lo general la entrada I) es llevada a tierra cuando se utiliza como generador de paridad. Por otra parte, Cuando el circuito es utilizado como detector de paridad deben entrar en él las nueve líneas que vienen del transmisor y por ende, conectar la salida par (F_p) o impar (F_i) a la entrada I del receptor o circuito detector. Estos dos circuitos integrados generan uno lógico en la línea de salida par ($F_p = \sum_{\text{even}}$) cuando la suma de los bits con nivel uno en las entradas (A, B, C, D, E, F, G, H, I) es par, lo que trae como consecuencia una sumatoria total impar en el dato que se forma con los ocho bits de entrada más el bit de paridad generado en la salida par F_p . Lo mismo sucede cuando se configura para generar paridad impar; la suma total de los nueve bits da un resultado par. La figura 5.57 muestra el diagrama y la tabla de funcionamiento de este circuito integrado.

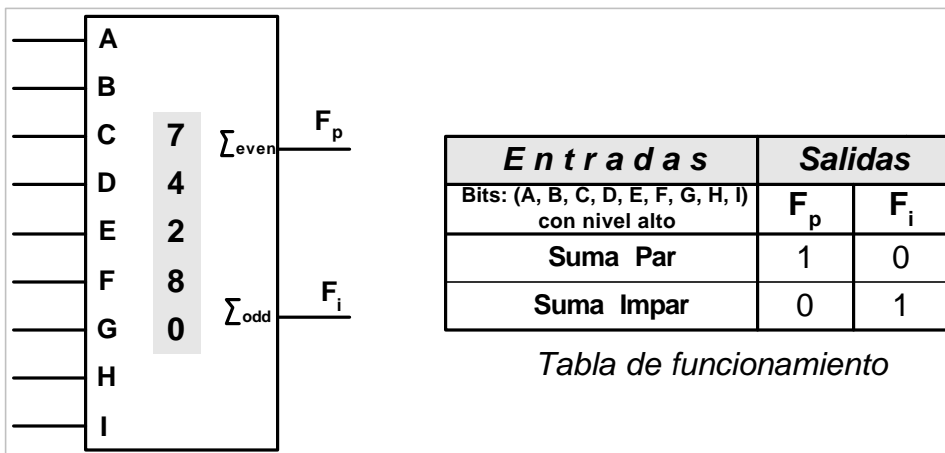


Figura 5.57. Generador y detector de paridad de 9 bits 74280.

5.6.2.1 Aplicaciones de los circuitos integrados 74180 y 74280.

Estos chips tienen aplicaciones específicas en la generación y detección de errores de paridad con distancia uno; también pueden ser acoplados en cascada para aumentar el tamaño de la palabra. Tienen aplicaciones en los Sistemas de transferencia de información a través de buses de computadoras, Control de transmisiones de datos digitales de un lugar remoto, y muchas otras aplicaciones. El chip 74280 fue diseñado para reemplazar exactamente a su antecesor 74180; cae perfectamente en la misma

base con la diferencia del pin tres que no debe ser conectado en el chasis (Nc: no conexión). A continuación se muestran algunas aplicaciones y expansiones realizadas con estos circuitos integrados.

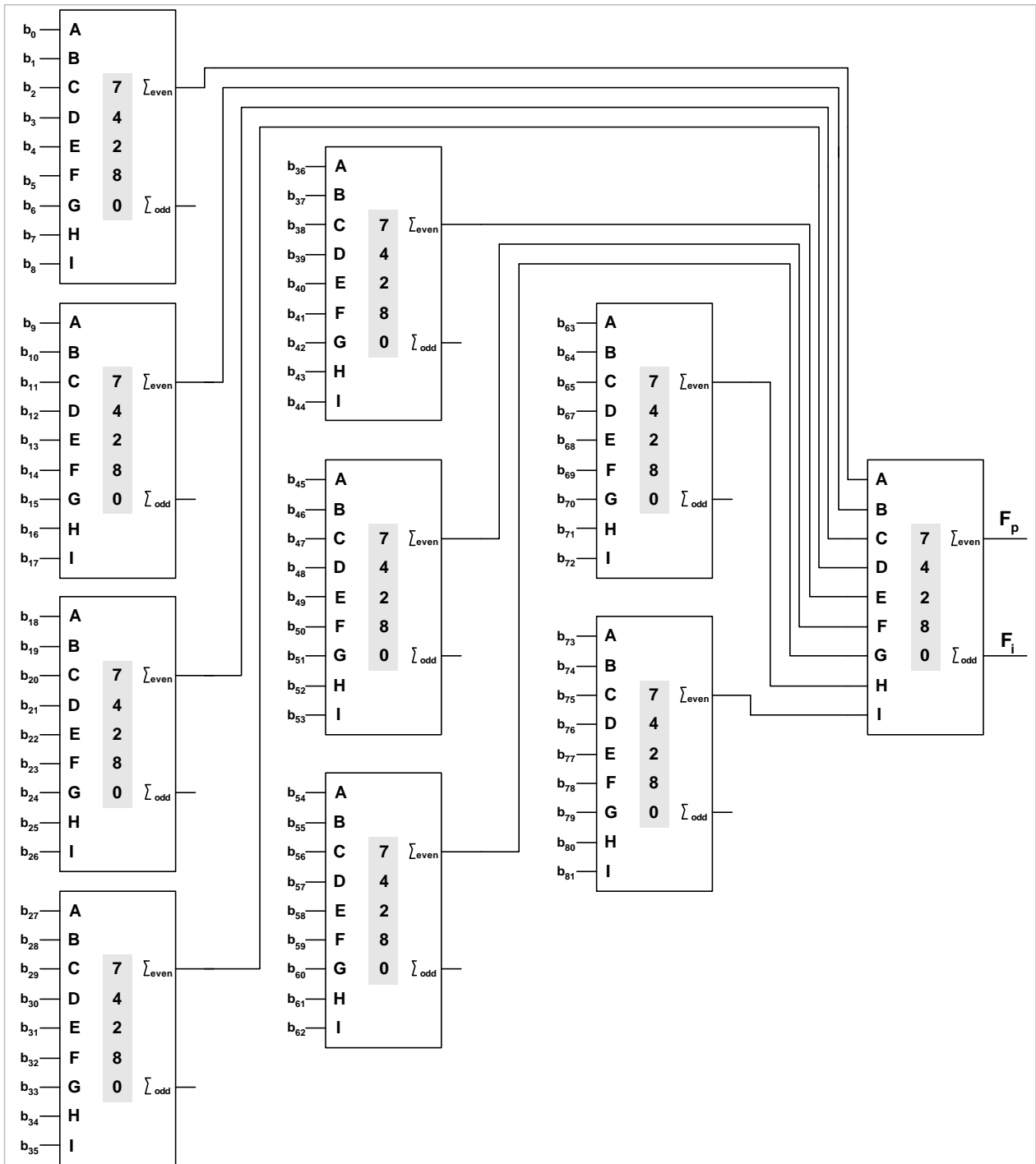


Figura 5.58. Generador o detector de paridad de 81 bits realizado con el chip 74280.

El circuito de la figura 5.58 es un generador o detector de paridad realizado con diez chips 74280; esto da como resultado una expansión paralela del dato transmitido o recibido hasta 81 bits y, adicionalmente se debe agregar el bit de paridad para que el circuito completo llegue hasta 82 bits. La figura 5.59 muestra otra expansión en cascada hecha con tres circuitos integrados 74280 y su equivalente realizado con el chip 74180.

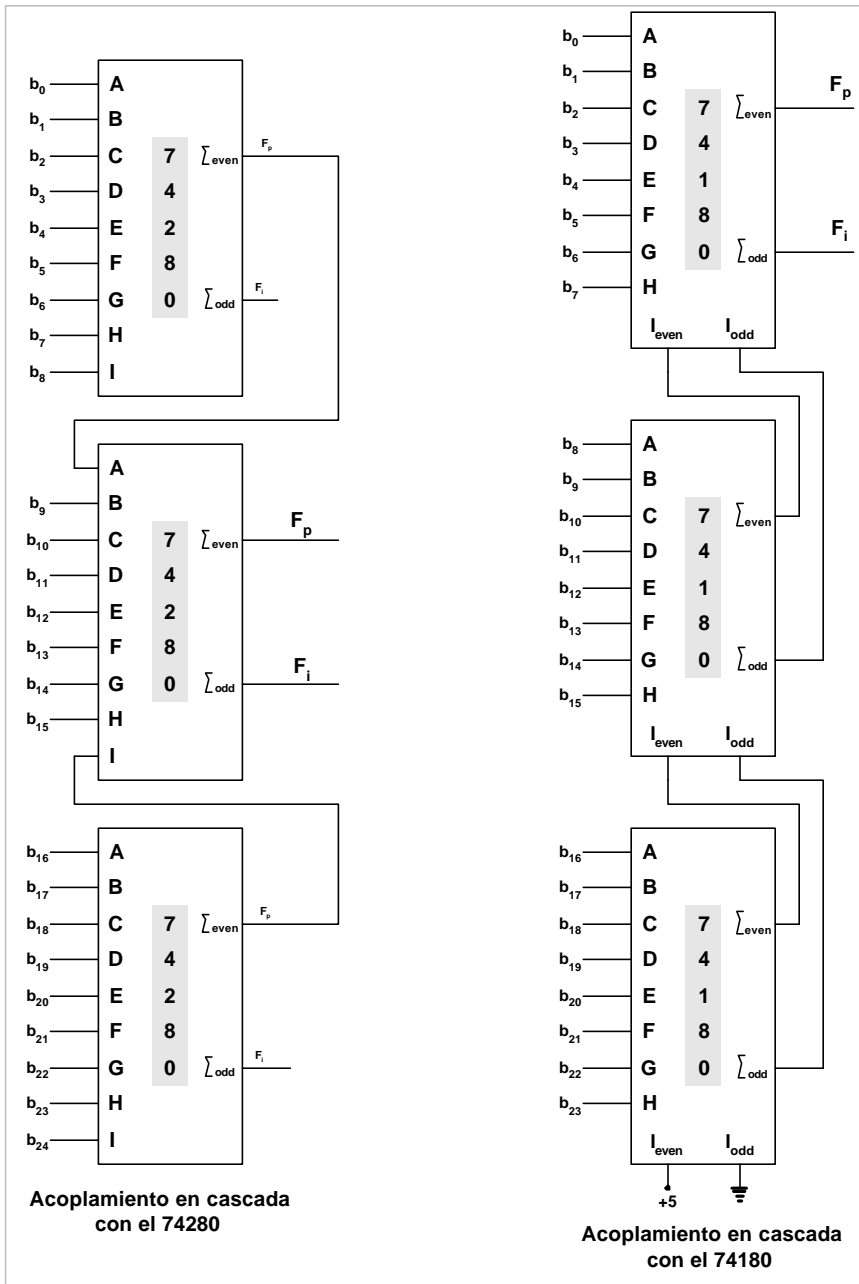


Figura 5.59. Generador detector de paridad de 24 bits con los chips 74280 y 74180.

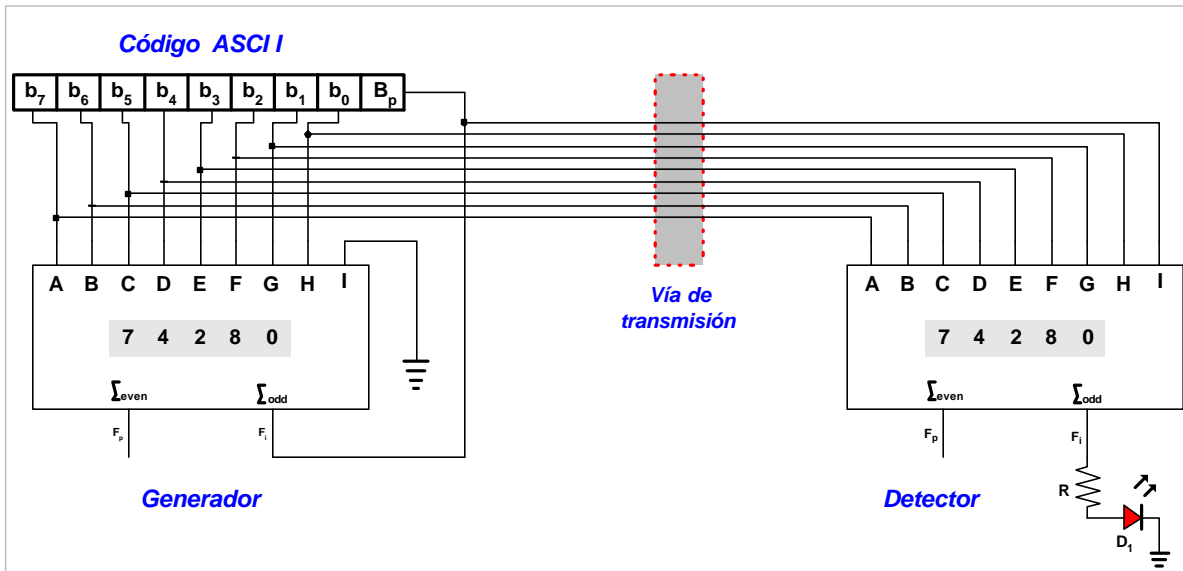


Figura 5.60. Circuito que chequea errores de transmisión de un bit en el código ASCII.

Si el número de bits en nivel lógico uno del sistema de generación y chequeo de la figura 5.60 es impar el led D_1 enciende indicando que hubo error de un bit en la transmisión del código ASCII. No obstante, el cambio de dos bits en las líneas de transferencia (Vía de comunicación) no afecta al detector de paridad el cual no indicará error de paridad. Este circuito también puede ser configurado para que indique error si el número de bits en uno es par. De la misma forma se puede hacer un diseño equivalente utilizando los circuitos integrados 74180.

5.6.3 Circuitos detectores y correctores Hamming.

Para diseñar un circuito detector y corrector Hamming de siete bits se necesitan tres circuitos generadores de paridad; cada uno de ellos con tres bits de dato más un bit de paridad. Por otra parte, el receptor debe tener tres detectores de paridad de cuatro bits cada uno, los cuales generan la posición del error en la palabra código y por ende, al cambiar un bit en la transmisión, se podrá hacer la transformación de este bit por su valor original. La figura 5.61 muestra un circuito que permite realizar la generación y corrección de error en código Hamming de siete bits con paridad par.

Para transmitir cuatro bits de información ($D_3D_2D_1D_0$) se necesitan tres bits para la generación y detección de paridad ($C_2C_1C_0$); de esta forma, la información completa del código queda codificada en siete bits ($I_7I_6I_5I_4I_3I_2I_1$). Si por algún evento no deseado cambia un bit del código Hamming; por ejemplo, perturbaciones en la vía de comunicación. El circuito detector conjuntamente con el decodificador 74138 ubican la posición del bit con error y mediante las compuertas NOR-Exclusivas cambian el nivel lógico bits y por lo tanto corrigen el valor de ese bit. El código detector de error Hamming coloca en el decodificador la posición del error ($e_2e_1e_0$) de acuerdo a las señales de paridad detectadas por los tres bloques D_p ; cuando éstas señales digitales están en cero ($e_2e_1e_0=000$) indican la única forma de no tener error en el sistema de comunicación. En la tabla 5.10 se describen las siete combinaciones de errores, desde I_1 hasta I_7 , con su correspondiente posición y los tres cuartetos para generar el código Hamming de siete bits.

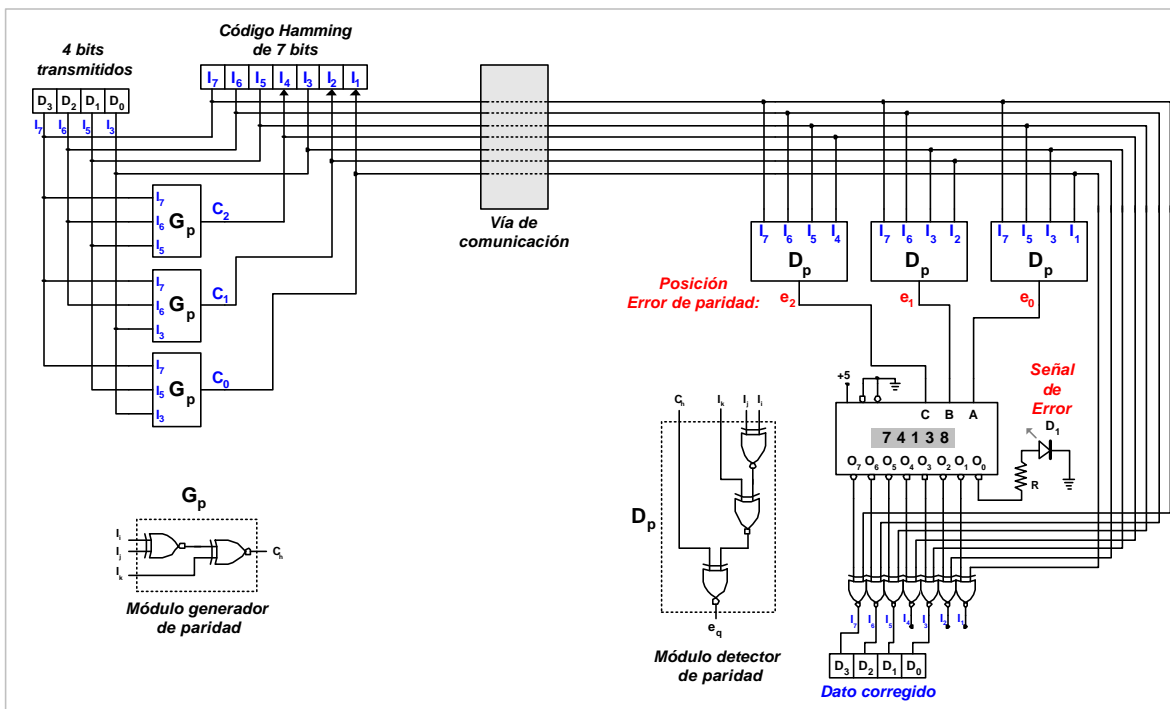


Figura 5.61. Circuito detector y corrector Hamming de siete bits.

Posición del error	C ₂	C ₁	C ₀		Formación del
	I ₄	I ₂	I ₁		Código Hamming
Sin error	0	0	0		
I ₁	0	0	1		(C ₀ =I ₁), I ₃ , I ₅ , I ₇
I ₂	0	1	0		(C ₁ =I ₂), I ₃ , I ₆ , I ₇
I ₃	0	1	1		
I ₄	1	0	0		(C ₂ =I ₄), I ₅ , I ₆ , I ₇
I ₅	1	0	1		
I ₆	1	1	0		
I ₇	1	1	1		

Tabla 5.10. Posiciones del error de paridad Hamming 7 bits y generación del código.

Ejercicio 5.29. Diseñe dos generadores de paridad de nueve bits: uno par y el otro impar; haga el diagrama con un solo chip 74180.

Ejercicio 5.30. Diseñe un circuito sencillo que permita detectar errores de transmisión con el código bi-quinario de siete bits.

Ejercicio 5.31. Modificar el circuito de la figura 5.61 para que muestre en displays el valor numérico de la posición del bit con error.

Ejercicio 5.32. Diseñe un generador de paridad impar de 32 bits con el circuito integrado 74280.

Ejercicio 5.33. Diseñe el mismo generador de paridad paralelo de 64 bits con el 74180 y con el 74280.

PRÁCTICA DE LABORATORIO #9

TITULO: Circuitos generadores, detectores de paridad y correctores de error.

OBJETIVO: El estudiante al terminar esta práctica estará en capacidad de poder analizar y diseñar circuitos de generación, detección y corrección de errores en la transmisión y recepción de información binaria mediante el método de paridad.

INTRODUCCIÓN: La información binaria está expuesta a variaciones y perturbaciones de diversos tipos en los hilos o medios conductores que la transportan; muchas veces uno o más bits cambian de nivel y en consecuencia el receptor obtiene una información errada del dato que fue transmitido. Los circuitos generadores – detectores simples de un bit de paridad se encargan de detectar errores en la información chequeando la suma par e impar de los bits del dato; indicando error si las paridades tanto del generador como el receptor no son iguales. Sin embargo, un sistema de éste tipo solo puede detectar errores de cambio en un solo bit (suma impar) y no es capaz de indicar errores cuando el cambio de bits es par. Por lo que es necesario recurrir al método de detección y corrección de paridad en código Hamming. La información necesaria para elaborar esta práctica está contenida en los capítulos uno y cinco de este material, y en la bibliografía recomendada al final de la presente guía. La práctica consiste en un primer montaje, con el chip generador y chequeador de paridad 74280 y otro montaje realizando un detector y corrector Hamming de siete bits.

PRELABORATORIO: Investigar los siguientes tópicos.

- Funcionamiento de los chips 74280, 74180 y equivalentes.
- Diseño de generadores y detectores de paridad con compuertas.
- Expansión con los chips generadores y chequeadores de paridad 74280 y 74180.
- Generadores, detectores y correctores de paridad en código Hamming.

MATERIALES Y EQUIPOS NECESARIOS:

- Dos chips 74280, diodos leds, DIP-SW's y, de ser necesario, chips combinacionales.
- Compuertas exclusivas y básicas de acuerdo a los diseños realizados.
- Protoboard, cable telefónico, pinza, piqueta.
- Multímetro digital y fuente de 5 Volt / 2 Amp.

DESARROLLO:

1. Implementar un circuito generador y detector de paridad que permita chequear errores cuando se transmiten desde un punto a otro un caracter en código ASCII. El sistema debe tener un circuito que permita generar manualmente los errores de transmisión.

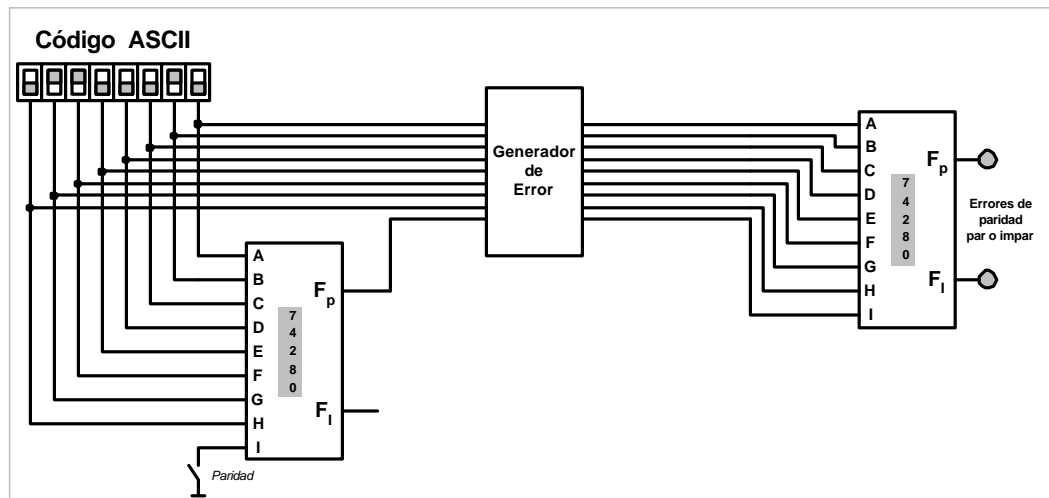


Diagrama del generador y detector de error de paridad.

2. Realizar un circuito que permita detectar y corregir, en el receptor, errores de transmisión. El dato a transmitir es de cuatro bits; el circuito debe detectar errores de cambio en dos bits y corregir cuando cambie un solo bit. Se recomienda un diseño donde se pueda aplicar el método de detección y corrección Hamming de siete bits.

POST-LABORATORIO.

- Explicar con tablas y planos cada uno de los montajes realizados.
- Realizar el diseño y simulación del sistema de detección y corrección Hamming de siete bits. Explique la formación del código.
- Hacer expansiones de generadores y detectores de paridad en cascada y paralelo utilizando los chips 74180 y 74280.
- Diseñar generadores y chequeadores de paridad con compuertas digitales.

MONTAJES ALTERNATIVOS:

1. Utilizando un diseño del método de detección y corrección del código Hamming de ocho bits realizar un circuito que permita detectar y corregir, en el receptor, errores de transmisión. El dato a transmitir es de cuatro bits; el circuito debe detectar errores de cambio en dos bits y corregir cuando cambie un solo bit.
2. Implementar un detector de errores de paridad par o impar de un bit donde se puedan recibir palabras con un tamaño de dos bytes, más el bit de paridad.
3. Realizar con compuertas digitales el montaje de un generador y chequeador de paridad de medio byte. El circuito debe indicar con un led el momento cuando haya errores de transmisión de un bit.
4. Diseñar e implementar un sistema de generación y detección de paridad que permita enviar datos de un byte en forma serial, transferirlos al receptor en forma serial. Una vez allí, convertirlos en paralelo y por último detectar los posibles errores que se puedan presentar durante la transmisión.

BIBLIOGRAFÍA.

- CUESTA, Luís M. PADILLA G, Antonio. REMIRO D, Fernando. (1993). Electrónica digital. Madrid: McGraw Hill. S/f. p.445.
- GAJSKI, Daniel D. (1997). Principios de diseño digital. Madrid: Prentice Hall Iberia. S/f. p.488. "Principles of digital design". Traducido por: Alberto Prieto Espinosa.
- LLORIS, Antonio. PRIETO, Alberto. (1996). Diseño lógico. Madrid: McGraw Hill. S/f. p.403.
- MANDADO, Enrique. (1987). Sistemas electrónicos digitales. Barcelona (España): Marcombo Boixareu Editores. Sexta edición. p.705.
- MANO, Morris. KIME, Charles. (1998). Fundamentos de diseño lógico y computadoras. México: Prentice Hall. Primera edición en español. P.604. "Logic and computer design fundamentals". Traducido por: Teresa Sanz Falcón.
- NELSON, V. NAGLE, H. CARROLL, B. IRWIN, J. (1996). Análisis y diseño de circuitos lógicos digitales. México: Prentice Hall. Primera edición. p.842. "Digital logic circuit analysis and design". Traducido por: Oscar A. Palmas V.
- TOCCI, Ronald. (1995). Sistemas digitales principios y aplicaciones. México: Prentice Hall. Quinta edición. p.823. "Digital systems principles and applications". Traducido por: Edmundo G. Urbina M.
- WARKELY, John F. (1997). Diseño digital principios y prácticas. México: Prentice Hall. S/f. p.743. "Digital design principles and practices". Traducido por: Gutiérrez R. Raymundo H.

MANUALES.

- NATIONAL SEMICONDUCTOR. (1981). Manual TTL y CMOS.
- MOTOROLA Inc. (1992). Fast and LS TTL. (DL121/D REV. 5).

CAPÍTULO 6.

6. CIRCUITOS DIGITALES COMBINACIONALES VLSI.

Desde la aparición del transistor, los fabricantes de dispositivos semiconductores han ido ampliando la integración de componentes en los circuitos electrónicos “Chips”; y es a partir de los años sesenta, cuando comienzan a hacer su aparición los circuitos integrados analógicos y digitales. La cantidad de transistores y compuertas necesarios en la integración de un circuito está descrita en la tabla 6.1.

Año	Grado de integración	Nº aproximado de compuertas	Nº de transistores	Funciones de los circuitos
1960	SSI: Pequeña escala de integración	1 a 10	4 a 100	Compuertas
1966	MSI: Mediana escala de integración	10 a 100	100 a 1000	Combinacional, FF, etc.
1969	LSI: Larga escala de integración	100 a 1000	1000 a 10000	Memorias TTL
1975	VLSI: Muy alta escala de integración	Mayor a 1000	Mayor a 10000	Eprom, RAM, μ P, etc.

Tabla 6.1. Cronología de las escalas de integración de circuitos integrados “Chips”.

La tecnología VLSI, ha creado circuitos integrados que pueden ser programados a conveniencia del diseñador del circuito lógico digital; la síntesis del dispositivo se realiza con ayuda de un equipo programador universal de circuitos integrados, una computadora y algún lenguaje de programación específico para tal fin. La síntesis de éstos puede ser realizada por: “quemado de fusibles”, antifusible “fusión de puntos”, máscara de programación, programación eléctrica fija y programación eléctrica volátil. Estos tipos de circuitos se conocen como dispositivos lógicos programables (PLD) y, su implementación reduce el costo de fabricación de circuitos electrónicos digitales; así como los tamaños de las tarjetas de circuito impreso (PCB: Printed Circuit Board) y por ende, el consumo de energía.

Dentro de los circuitos integrados que comprende esta tecnología se encuentran los de memoria, los circuitos integrados PAL (Programmable Array Logic); los circuitos integrados PLA (Programmable Logic Array), GAL (Gate Array Logic), etc.

Los dispositivos se clasifican más rigurosamente como se describe a continuación: en la figura 6.1 se tiene la clasificación de los circuitos integrados según sea la alternativa de selección e integración del diseñador, y en la figura 6.2, la clasificación de los circuitos lógicos programables.

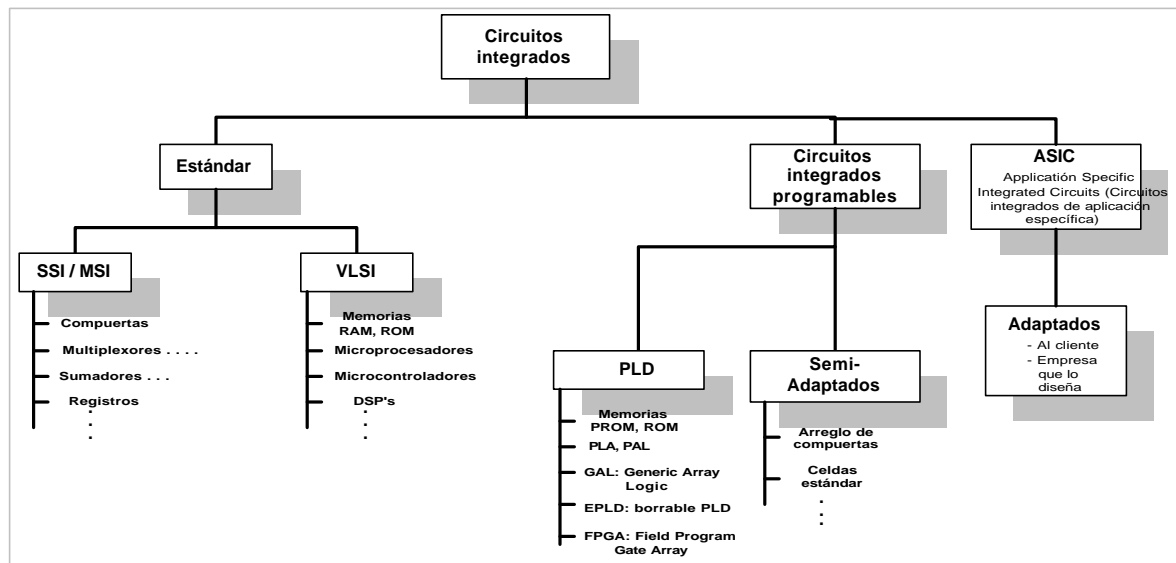


Figura 6.1. Clasificación de acuerdo al tipo de integración seleccionada por el diseñador. Tomado del libro “Diseño lógico” A. Lloris; A. Prieto

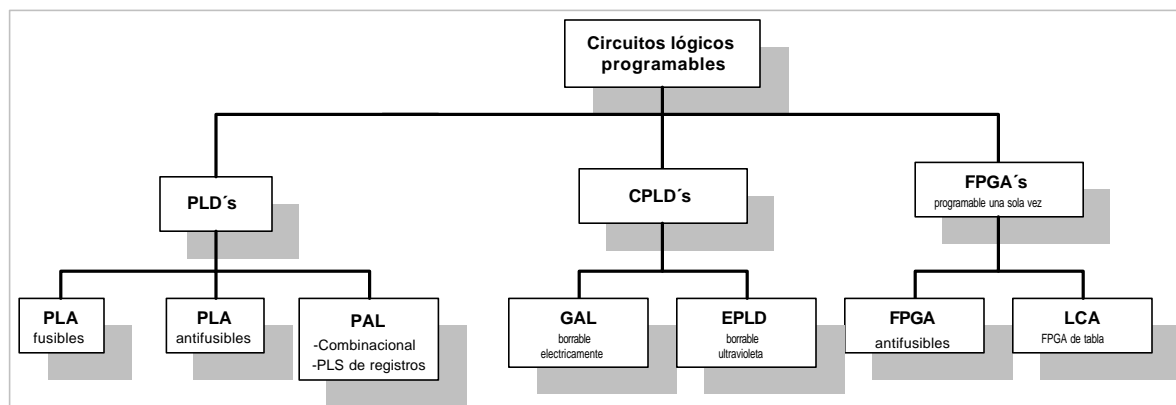


Figura 6.2. Clasificación de los circuitos lógicos programables. Tomado del libro “Diseño lógico” A. Lloris; A. Prieto

El ingeniero que diseña sistemas digitales necesita comprender el proceso de integración de estos circuitos integrados y así poder tener una idea clara de las nuevas tecnologías que se están utilizando, y de esta manera, adaptarse a los cambios de la electrónica, específicamente, la electrónica digital. Este capítulo describe la parte fundamental de los dispositivos lógicos programables, así como, las clasificaciones que reciben estos circuitos digitales.

6.1 Circuitos integrados de memoria ROM.

Un circuito integrado de memoria ROM (memoria de solo lectura) está organizada en 2^n palabras de m bits, tiene n entradas de dirección, m líneas de salida de datos, y $m \cdot 2^n$ celdas de memoria; además puede tener r entradas, adicionales, de (datos y/o programación) y p entradas de control. Esquemáticamente se representa tal como en la figura 6.3. Cada celda de memoria contiene un bit de información, que a través de las salidas de datos puede ser leído cuando se desee. Si la información contenida en cada celda se puede modificar, entonces se llama de lectura y escritura. Por el contrario, si la información contenida en la celda no se puede modificar durante la operación normal de la memoria, o sea, es una información permanente, se dice que la memoria es de solo lectura (ROM).

La figura 6.4 muestra la clasificación de circuitos ROM programables; de acuerdo al tipo de grabación y borrado. Por ejemplo, las memorias EPROM's (Erase Programmable Read Only Memory) son grabadas (programadas) una sola vez por un equipo electrónico específico para esa tarea. No obstante, el borrado de la misma tiene que ser hecho con otro equipo, emisor de rayos ultravioleta. El chip de memoria trae una ventana transparente plastificada por donde entran los rayos, luego de ser sometida durante un tiempo (20 minutos aproximadamente) se borran los valores binarios que habían sido programados anteriormente. Cuando la memoria EPROM está totalmente borrada (en blanco), todas las $m \cdot 2^n$ celdas quedan con un nivel lógico alto de salida.

La memoria RAM (“Random Access Memory”: memoria de acceso aleatorio) funciona de forma equivalente a un dispositivo combinacional, donde las n líneas de direcciones son las variables de entrada, r y p son entradas de control (habilitación, etc.) y m son las salidas del circuito digital combinacional. Tanto la memoria ROM, como la RAM pueden ser considerados circuitos digitales combinacionales, ésto debido a que produce m funciones de conmutación con n variables comunes para cada función, y por tanto, 2^n *minterms* y/o *maxterms*. No obstante, los chips de memoria RAM poseen un pin adicional de Lectura / Escritura que hace la diferencia con respecto a la memoria tipo ROM. Ver figura 6.3.

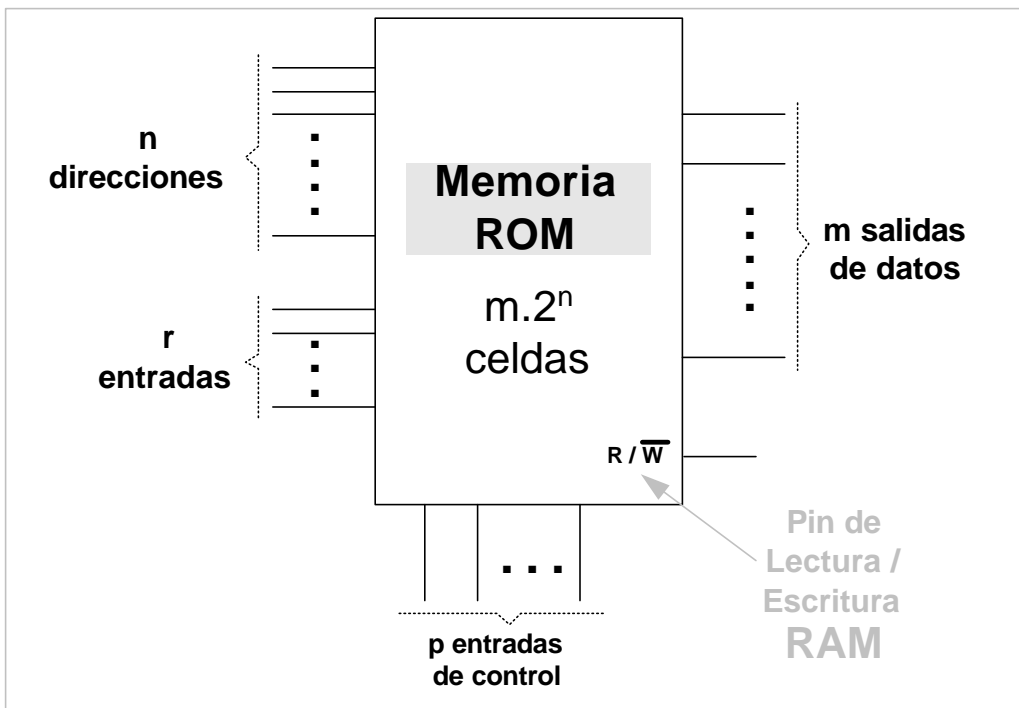


Figura 6.3. Esquema en bloque de una memoria ROM, o RAM.

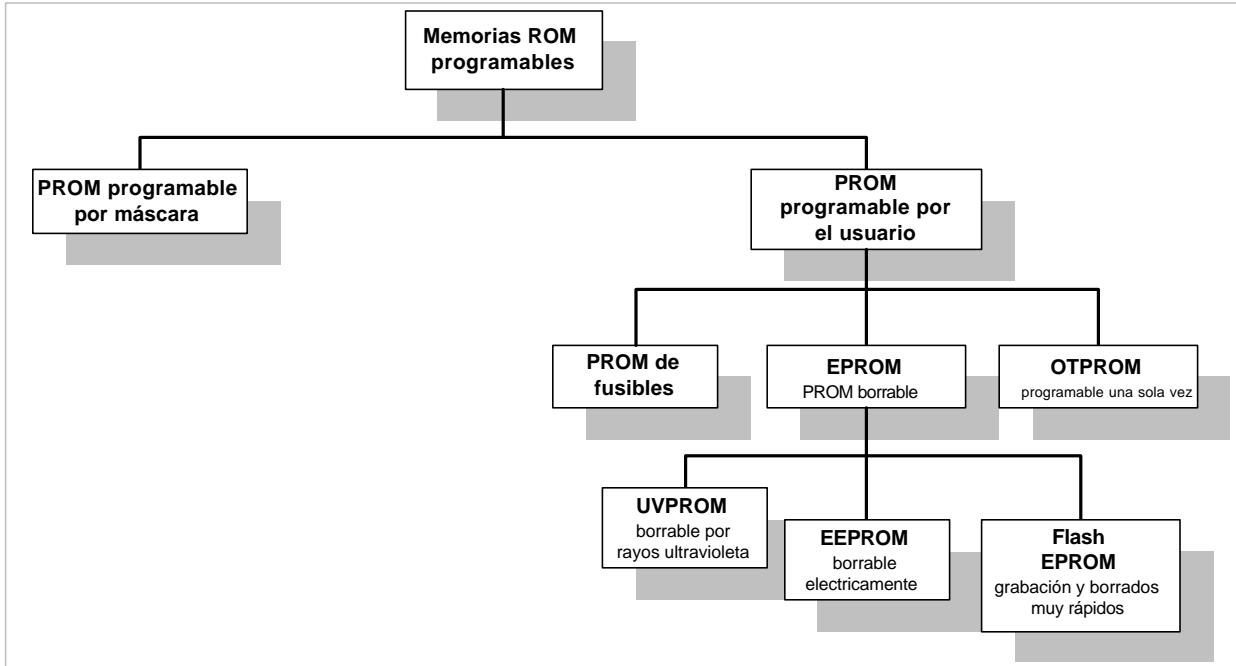


Figura 6.4. Clasificación de las memorias ROM de acuerdo al tipo de grabado y borrado.

La programación, y funcionamiento interno, de una ROM hecha con transistores se describe en la figura 6.5, allí se ve que el dispositivo está formado por transistores que pueden ser polarizados en la unión base emisor por medio de la “quema” de un fusible conectado a la base del mismo. Cada transistor está configurado como “seguidor emisor”, por lo cual, las líneas de datos (D_0, D_1, \dots) tendrán la misma tensión de salida del decodificador. Las líneas de direcciones (A_0, A_1, \dots) activan alguna de las salidas del decodificador (O_0, O_1, O_2, \dots); y ésta a su vez, determina cual fila de transistores estará activa, colocando el dato (D_0, D_1, D_2, \dots) en la salida del circuito integrado de memoria.

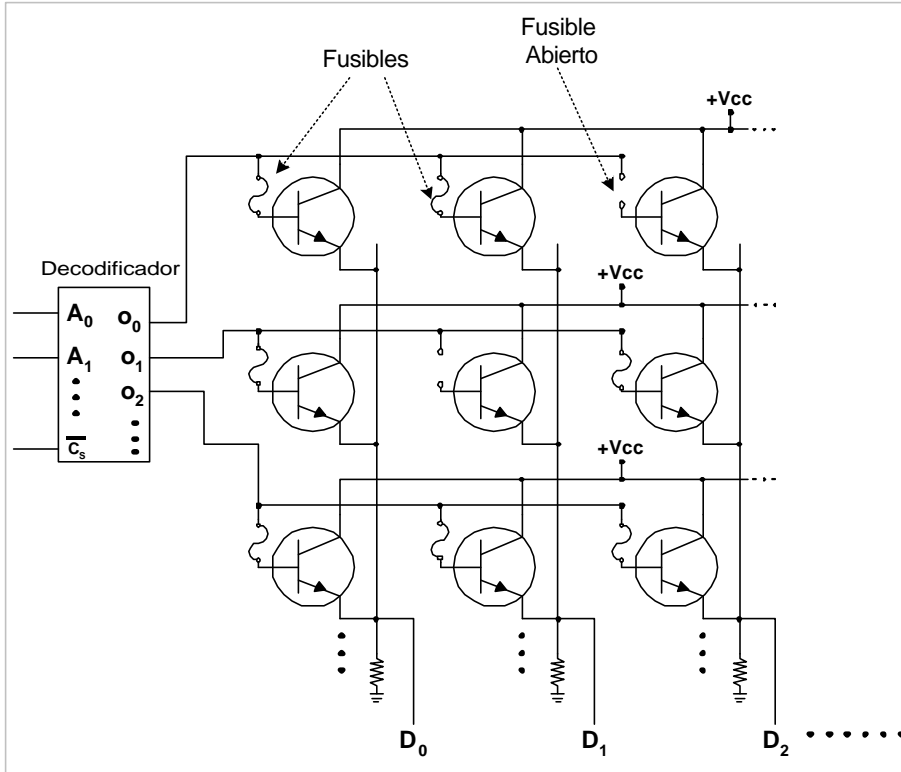


Figura 6.5. Circuito interno de memoria ROM con transistores.
Programación mediante quema de fusibles.

6.1.1 Circuitos integrados de memoria EPROM.

Son memorias que pueden ser re-programadas (Grabadas y borradas) varias veces con dispositivos universales de programación de memorias, sin embargo, el borrado debe realizarse, sometiendo con luz ultravioleta, la ventanilla del circuito integrado con un equipo de luz ultravioleta por un tiempo que va de 15 a 20 minutos.

Las EPROM presentan una nomenclatura que comienza con el número **27** seguido de la(s) letra(s) que indica la tecnología de fabricación, y luego, el valor de la capacidad en *Kilobits*. No obstante, como la capacidad debe presentarse casi siempre en bytes, se debe dividir por 8. La capacidad determinará las líneas de direcciones que posee el circuito integrado de memoria EPROM. A continuación se presentan algunos de estos circuitos:

Nº del C.I. EPROM	Tecnología	Capacidad en bits	Capacidad en bytes	
2732	Bipolar	32 Kbits	32/8 = 4 Kbytes	4.096 bytes
27N64	NMOS	64 Kbits	64/8 = 8 Kbytes	8.192 bytes
27C256	CMOS	256 Kbits	256/8 = 32 Kbytes	32. 768 bytes

El circuito integrado 27C256 tiene una capacidad de 32.768 bytes, este valor determina las líneas de direcciones del integrado $2^{13} = 32.768$; lo que implica 13 líneas ($A_0, A_1, A_2, A_3, \dots, A_{12}$). El integrado 27C256 puede almacenar 32.768 datos de ocho bit cada uno, y su diagrama de pines, está descrito en el anexo de este texto.

6.2 Dispositivos Lógicos Programables (PLD).

Son circuitos integrados que, como su nombre lo indica, son programables por diversos métodos y equipos. Estos circuitos necesitan equipos adicionales para realizar síntesis de diseños lógicos; es necesario utilizar lenguajes de programación de chips, un Computador asociado a éste y el equipo donde se grabará (programará) el circuito integrado. Adicionalmente, puede ser necesario el uso de un equipo borrador de chips.

Lenguajes de programación como ABEL (compilador de PLD) son utilizados para tal fin. Además de esto, es necesaria la utilización de programas editores y capturadores de esquemas electrónicos como lo son, por ejemplo, ORCAD y PROTEL. Por otra parte, también están, para los FPGA, los lenguajes descriptores de Hardware como pueden ser VHDL y Verilog. La simulación del diseño se realiza con programas de aplicación tales como ORCAD, SPICE, VDHL y otros de aplicación profesional. Los circuitos que van ha ser sintetizados y grabados deben ser sometidos a todas estas pruebas, más aún, si el chip es programable¹ una sola vez.

1. Para profundizar más en el tema se recomienda utilizar la bibliografía: Warkely, J.F. "Diseño digital, principios y prácticas" 1994; y el texto de la ingeniero Zulay Franco, "Circuitos electrónicos digitales utilizando dispositivos lógicos programables.

Los circuitos integrados ROM, PLA, PAL y GAL están incluidos dentro de los dispositivos lógicos programables. En la próxima sección se describirá el funcionamiento de estos tres últimos dispositivos.

6.3 Arreglos lógicos programables combinacionales PAL y PLA.

En una PAL (Programmable Array Logic) hay n entradas y una matriz de compuertas AND; con un número menor que 2^n . Siendo, el plano programable; las conexiones entre las entradas y la matriz de compuertas AND. La figura 6.5 muestra los diagramas internos equivalentes de los circuitos ROM, PAL y PLA. La salida de la PAL está formada por una matriz de compuertas OR que se puede fijar o fusionar con la matriz AND. El número de entradas de las PAL está por el orden de 20. Por ejemplo, la **PAL16L8** es un chip que posee hasta 16 entradas y hasta 8 salidas.

Las PLA (Programmable Array Logic) también tiene n entradas, una matriz con N compuertas AND, donde N es mucho menor que 2^n , y una matriz de compuertas OR en la salida. Este circuito a diferencia del PAL, puede programarse tanto la matriz formada por las líneas de entradas con las AND, como las salidas de las compuertas AND con las entradas de las compuertas OR. En las PLA actuales pueden haber de 10 a 20 entradas, de 30 a 60 compuertas AND y de 10 a 20 compuertas OR. En las PLA cada salida incluye un número de productos programable, aquí se pueden compartir los productos entre las distintas salidas. Ver figura 6.7.

Las memorias ROM, los circuitos PAL y las PLA se pueden utilizar para sintetizar simultáneamente varias funciones que compartan las mismas variables de entradas. Pero dada la versatilidad de las conexiones de estos circuitos, entre las entradas y las compuertas AND, pueden programarse también varias funciones de diferentes variables. Las ROM, PAL y PLA pueden ser programados en fábrica (mandadas a diseñar) o también pueden ser programadas por el usuario con el equipo de programación respectivo.

Los circuitos lógicos programables con conexiones de fusibles, se fabrican de forma que contienen todas las interconexiones posibles, y se programan eliminando los puntos de conexión necesarios para sintetizar las funciones requeridas. Por ejemplo, para programar un producto de tres variables en un arreglo AND, donde se pretende sintetizar la función: $F = \bar{a} \cdot b \cdot \bar{c}$ hay que dejar intactas, sin quemar los fusible, a estas líneas; y colocar una tensión elevada (10 a 30 V), con la finalidad de provocar la vaporización del fusible en las conexiones correspondientes a los literales: a, \bar{b}, c . La figura 6.6 muestra el esquema del resultado de esta programación.

Las uniones o fusiones, Contrario a quema de fusible, se representan con un punto en el plano de la matriz de programación del dispositivo, y se realizan en dispositivos que son programables una sola vez. Sin embargo, los fusibles que son quemados debe representarse, con un punto, solo los que quedan intactos. Estas uniones forman la suma de productos que se establecen con las compuertas NOT, AND y OR de los circuitos lógicos programables.

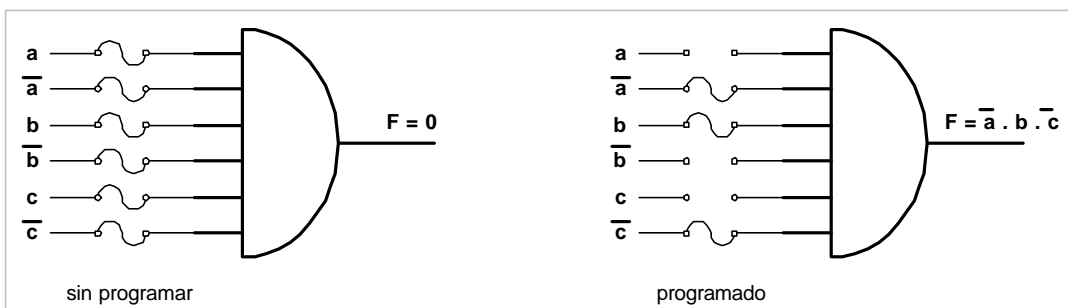


Figura 6.6. Síntesis de la función: $F = \bar{a} \cdot b \cdot \bar{c}$ en un arreglo AND.

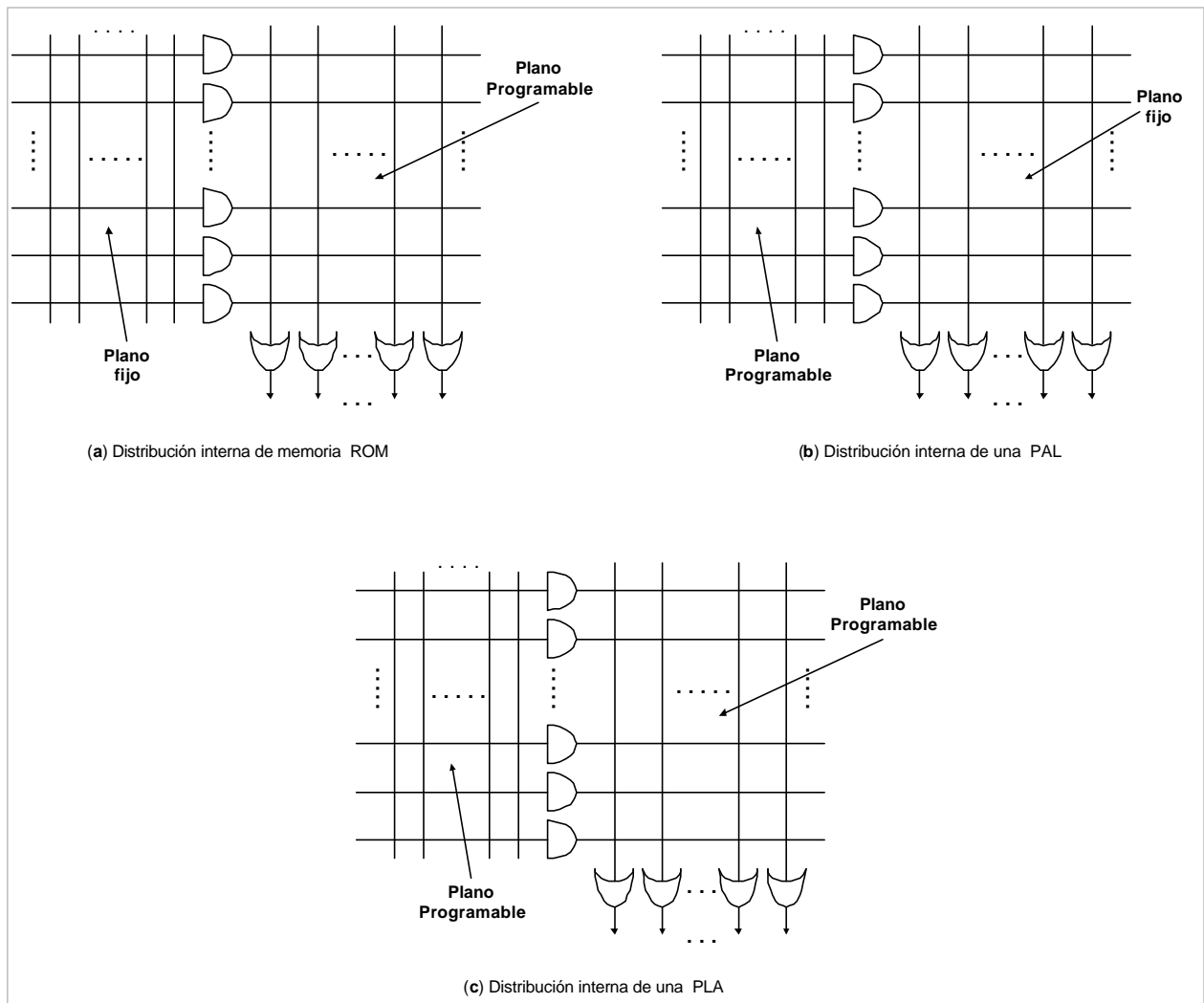


Figura 6.7. Esquema interno de los dispositivos lógicos programables ROM, PAL y PLA.
Tomado del libro “Diseño Lógico” de A. Lloris y A. Prieto.

Ejercicio 6.1. Obtener las cinco funciones de salida **f**, **g**, **h**, **j** y **k** que sintetiza el PLA de la figura 6.8. Las variables de entrada son **w**, **x**, **y**, **z**. Obtenga también la función en PAL y ROM.

Solución: Se utiliza la representación de puntos para indicar las filas y columnas que se deben interceptar en el plano programable de la matriz AND. Del mismo modo, sucede en el plano programable de la matriz OR. (Las soluciones en ROM y PAL se dejan para el lector).

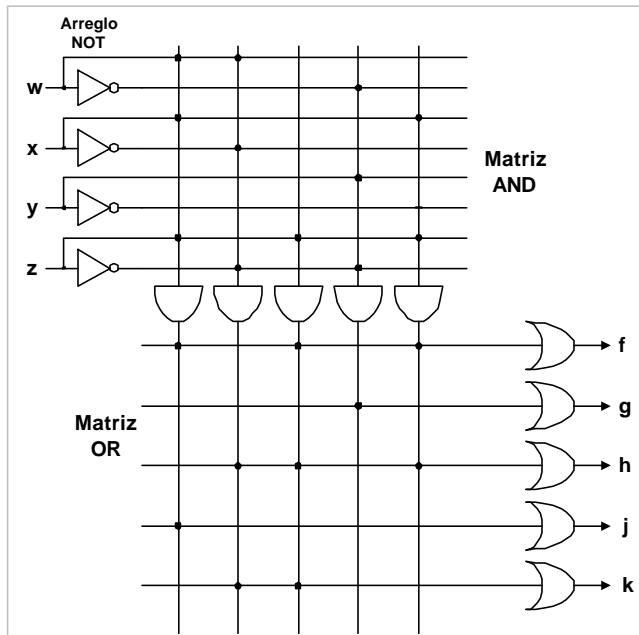


Figura 6.8. PLA genérico para el ejercicio 6.1.

Ejercicio 6.2. Realizar la síntesis “simbólica” con puntos de conexión en una PLA que permita generar las siguientes funciones de conmutación:

$$f = \sum_m (1, 2, 3, 8, 11, 12, 13, 17, 23, 26, 30) + d(0, 14, 22, 27)$$

$$g = \sum_m (0, 2, 3, 4, 8, 10, 11, 23, 26, 27, 28) + d(9, 19, 25, 31)$$

$$h = \sum_m (0, 2, 5, 9, 12, 16, 17, 25, 26, 27)$$

$$j = \prod_M (1, 2, 3, 8, 9, 11, 13, 14, 15, 22, 24, 27, 30, 31)$$

$$k = \prod_M (3, 4, 5, 11, 16, 17, 21, 22, 24, 26) \cdot d(0, 2, 7, 31)$$

$$q = \sum_m (1, 3, 8, 9, 10, 12, 14, 17, 19, 21, 22, 24, 25, 27)$$

Los PAL y PLA son circuitos integrados combinacionales con diversas configuraciones de pines de entrada/salida. Por ejemplo, la PAL16L8 puede tener hasta

16 entradas y hasta ocho salidas, estas últimas, pueden ser configuradas como pines de entrada o salida, el dispositivo genera funciones con un máximo de siete productos AND por cada salida y tiene otra AND dedicada solamente a controlar el tercer estado de la salida OR.

El lector que desee profundizar más sobre el tema se recomienda que lea el capítulo 7 del libro "Diseño Digital Principios y Prácticas" (1997) de John Warkely.

6.4 Arreglo de compuertas lógicas (GAL).

GAL (Gate Array Logic): Es un tipo de dispositivo lógico que puede ser reprogramado muchas veces con tensiones menores o igual a 5 voltios. Esta formada por una matriz conectada a la entrada de varias compuertas AND, y las salidas de éstas, conectadas a una matriz fija de compuertas OR. La configuración NOT, AND y OR de las GAL permiten implementar funciones de conmutación tipo suma de productos, donde la cantidad de variables de entradas están limitadas. La figura 6.9 muestra la estructura interna de una GAL genérica con una sola salida y dos variables de entrada; las uniones, en la matriz AND, se realizan con dispositivos CMOS programables y borrables eléctricamente (EECMOS = E²CMOS).

Los dispositivos de E²CMOS son activados (on) para unir las entradas de las compuertas AND con las variables. Por ejemplo, la figura 6.9 muestra la configuración del GAL para que genere la función: $F = \overline{A} \cdot \overline{B} + A \cdot B + C$

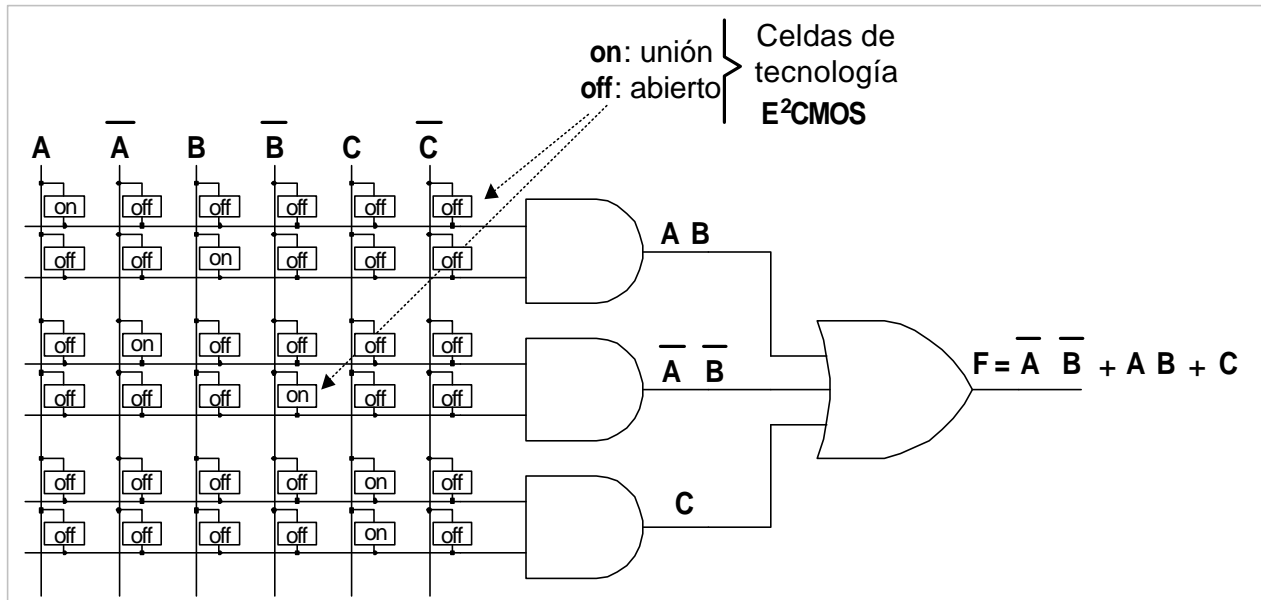


Figura 6.9. Función: $F = A \cdot B + A \cdot \bar{B} + C$, generada con una GAL.

6.4.1 Circuitos integrados GAL.

Estos circuitos programables tienen una nomenclatura que siempre comienza con las letras “GAL” estas letras preceden un número de dos dígitos que indica la cantidad de entradas del dispositivo. Luego, la letra “V” que se coloca a continuación, señala que el circuito integrado es de configuración de salida variable. Después, vienen uno o más dígitos que indican la cantidad de entradas/salidas configurables. Por ejemplo, la **GAL22V10** puede tener hasta 22 entradas y hasta diez salidas; el valor 22 determina el número máximo de entradas y/o salidas que debe poseer el dispositivo, sin embargo, doce líneas son exclusivamente entradas y las otras diez pueden ser programadas como entrada/salida. A continuación se presenta una aplicación en GAL que sirve para generar un decodificador equivalente al 74138.

Los pasos que se deben realizar son los siguientes¹.

I) Diagrama en bloque: Es el que representa en forma gráfica los principales módulos que van a constituir el sistema, así como las entradas y salidas y las interconexiones entre módulos.

II) Codificación en ABEL (Advanced Boolean Equation Language): Es el archivo texto que describe el diseño, esta descripción se pueden hacer de tres formas: mediante ecuaciones, tabla de verdad y diagrama de estado. Con este archivo se puede simular para verificar el diseño y compilar para obtener un archivo que configura el PLD.

III) Simulación: Es donde se verifica el correcto funcionamiento del diseño utilizando los vectores de prueba del archivo ABEL.

IV) Reporte: En esta sección se coloca el archivo que se genera cuando se compila el archivo texto, él nos indica características tales como fan-in, fan-out, y las ecuaciones lógica que rigen el diseño.

1. Tomado del texto "Circuitos electrónicos digitales utilizando dispositivos lógicos programables" 2001, Ing. Zulay Franco,

Diagrama en Bloque

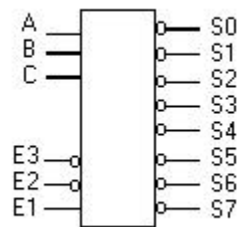


Tabla de Verdad

E3	E2	E1	A	B	C	S7	S6	S5	S4	S3	S2	S1	S0
x	x	0	x	X	x	1	1	1	1	1	1	1	1
x	1	1	x	X	x	1	1	1	1	1	1	1	1
1	x	1	x	X	x	1	1	1	1	1	1	1	1
0	0	1	0	0	0	1	1	1	1	1	1	1	0
0	0	1	0	0	1	1	1	1	1	1	1	0	1
0	0	1	0	1	0	1	1	1	1	1	0	1	1
0	0	1	0	1	1	1	1	1	1	0	1	1	1

0	0	1	1	0	0	1	1	1	0	1	1	1	1
0	0	1	1	0	1	1	1	0	1	1	1	1	1
0	0	1	1	1	0	1	0	1	1	1	1	1	1
0	0	1	1	1	1	0	1	1	1	1	1	1	1

Descripción en ABEL.

Las señales de entrada (A,B,C,E1,E2,E3) se asignan a pines de entradas .Las señales de salida (S0,S1,S2,S3, S4,S5,S6,S7) se asignan a los pines de salidas.

```

MODULE Decodificador
TITLE 'Decodificador'

"entradas
A,B,C,E1,E2,E3 pin 2,3,4,5,6,7;

"salidas
S0,S1,S2,S3,S4,S5,S6,S7 pin 23,22,21,20,19,18,17,16;

"Definiciones
Dato=[C,B,A];
Hab=[E3,E2,E1];
Salida=[S7..S0];

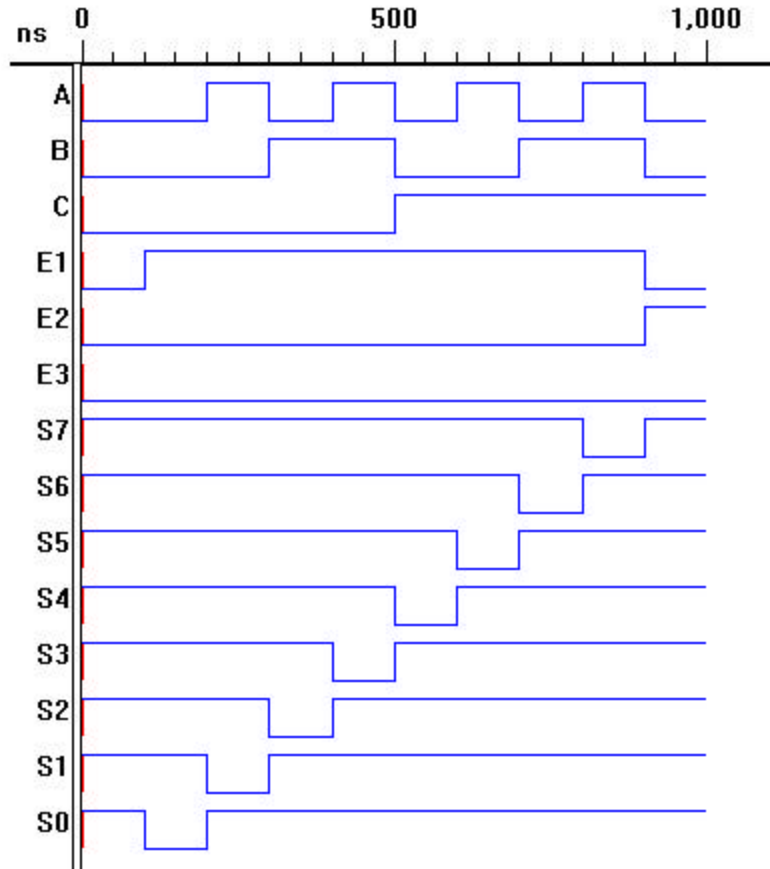
"Ecuaciones

Equations
when Hab==[0,0,1] THEN {when Dato==0 then Salida=^hfe;
                        when Dato==1 then Salida=^hfd;
                        when Dato==2 then Salida=^hfb;
                        when Dato==3 then Salida=^hf7;
                        when Dato==4 then Salida=^hef;
                        when Dato==5 then Salida=^hdf;
                        when Dato==6 then Salida=^hbf;
                        when Dato==7 then Salida=^h7f}
else Salida=^hFF;

test_vectors

( [C, B, A, [Hab]] -> [Salida])
[0, 0, 0, [0]] -> [^hff];
[0, 0, 0, [1]] -> [^hfe];
[0, 0, 1, [1]] -> [^hfd];
[0, 1, 0, [1]] -> [^hfb];
[0, 1, 1, [1]] -> [^hf7];
[1, 0, 0, [1]] -> [^hef];
[1, 0, 1, [1]] -> [^hdf];
[1, 1, 0, [1]] -> [^hbf];
[1, 1, 1, [1]] -> [^h7f];
[1, 0, 0, [2]] -> [^hff];
END
    
```

Simulación



Reporte

ispDesignEXPERT 8.2

Design decodificador created Sun Aug 05 21:59:10 2001

Title: Decodificador

P-Terms Fan-in Fan-out Type Name (attributes)

P-Terms	Fan-in	Fan-out	Type	Name (attributes)
6	6	1	Pin	S0
6	6	1	Pin	S1
6	6	1	Pin	S2
6	6	1	Pin	S3
6	6	1	Pin	S4
6	6	1	Pin	S5
6	6	1	Pin	S6
6	6	1	Pin	S7

=====
 48 P-Term Total: 48
 Total Pins: 14
 Total Nodes: 0
 Average P-Term/Output: 6

Equations:

- S0 = (A # B # C # E3 # E2 # !E1);
- S1 = (!A # B # C # E3 # E2 # !E1);
- S2 = (!B # A # C # E3 # E2 # !E1);
- S3 = (!B # !A # C # E3 # E2 # !E1);
- S4 = (!C # A # B # E3 # E2 # !E1);
- S5 = (!C # !A # B # E3 # E2 # !E1);
- S6 = (!C # !B # A # E3 # E2 # !E1);
- S7 = (!C # !B # !A # E3 # E2 # !E1);

Reverse-Polarity Equations:

PRÁCTICA DE LABORATORIO #10

TÍTULO: Diseño de circuito digital combinacional utilizando chips VLSI.

OBJETIVO: El estudiante al terminar esta práctica estará en capacidad de poder analizar y diseñar circuitos combinacionales de tecnología VLSI con memorias ROM.

INTRODUCCIÓN: Los chips de memoria EPROM permiten generar funciones con gran capacidad de integración de compuertas lógicas. El circuito digital se reduce considerablemente y por tanto, ocupa menos espacio físico. Los chips de la serie 27XXX ofrecen ocho salidas combinacionales y n líneas de entrada (direcciones) que dependen de la capacidad del dispositivo. El circuito a implementar es un generador de caracteres ASCII, que debe ser realizado con una matriz (8X5) de diodos led y debe ser capaz de visualizar el valor ASCII cuando a la entrada se coloque su equivalente binario. Las ocho líneas (D7,D0) son las salidas que deben excitar las 8 filas de diodos de la matriz. Por otra parte, cada columna debe ser conectada a un amplificador de corriente con un decodificador para realizar el barrido de las columnas. El contador binario utilizado en esta práctica debe ser implementado como un bloque genérico de módulo 8 y suministrado como un dispositivo o material.

PRELABORATORIO: Investigar los siguientes tópicos.

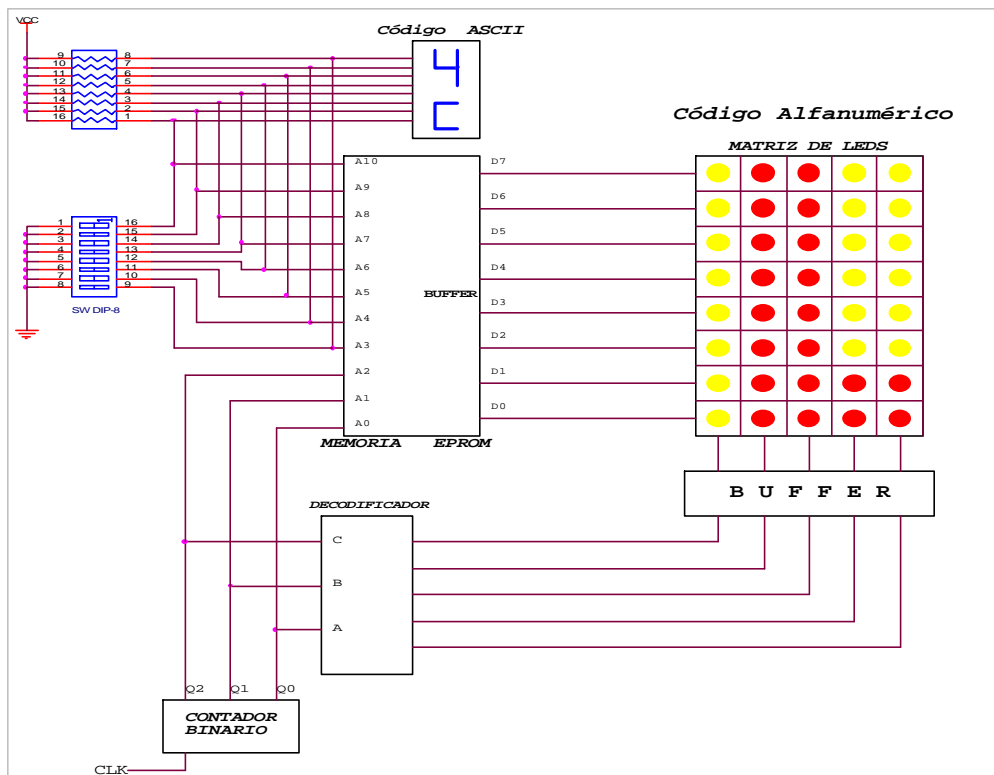
- Fundamentos de memorias y PLD's.
- Funcionamiento de los chips EPROM 2764, 27128, 27256 y equivalentes.
- Generar funciones con memorias ROM, PAL y PLA.
- Expansión de memorias en capacidad y en dato.
- Circuitos de barrido de matriz de leds.
- Manejo de programador universal de memorias y circuitos integrados.

MATERIALES Y EQUIPOS NECESARIOS:

- Un chip de memoria EPROM que Ud. considere conveniente.
- Matriz de diodos (40 diodos leds), dos displays 7 seg. Dos 7448 o 7447.
- Decodificadores y Buffers de corriente.
- Multímetro digital, fuente de 5 Volt / 2 Amp, Generador de señales y osciloscopio.
- Equipo de Programación Universal de circuitos integrados.
- Contador binario módulo 8.

DESARROLLO:

1. Implementar el circuito generador de caracteres mostrado en la gráfica.



El circuito debe mostrar en display 7 segmentos el valor hexadecimal que corresponde con el caracter alfanumérico del código ASCII visualizado en la matriz de diodos Leds. Observe el valor de la frecuencia cuando la matriz deja de “parpadear”.

POST-LABORATORIO.

- Explicar los planos de la implementación del circuito.
- ¿Por qué hay parpadeo cuando la frecuencia del generador es baja?
- Hacer expansiones de Bloques de memoria utilizando EPROM.
- ¿Que se debe agregar al circuito para ver una secuencia cualquiera de caracteres? .

BIBLIOGRAFÍA.

- ARTIGAS, J; BARRAGÁN, L; ORRITE, C. (1999). Aplicaciones y problemas de electrónica digital. España: Textos docentes Zaragoza. p.341.
- CUESTA, Luís M. PADILLA G, Antonio. REMIRO D, Fernando. (1993). Electrónica digital. Madrid: McGraw Hill. S/f. p.445.
- GAJSKI, Daniel D. (1997). Principios de diseño digital. Madrid: Prentice Hall Iberia. S/f. p.488. “Principles of digital design”. Traducido por: Alberto Prieto Espinosa.
- LLORIS, Antonio. PRIETO, Alberto. (1996). Diseño lógico. Madrid: McGraw Hill. S/f. p.403.
- MANDADO, Enrique. (1987). Sistemas electrónicos digitales. Barcelona (España): Marcombo Boixareu Editores. Sexta edición. p.705.
- MANO, Morris. KIME, Charles. (1998). Fundamentos de diseño lógico y computadoras. México: Prentice Hall. Primera edición en español. P.604. “Logic and computer design fundamentals”. Traducido por: Teresa Sanz Falcón.
- NEAMEN A, Donald. (1999). Análisis y diseño de circuitos electrónicos. Tomo II. México: McGraw Hill. S/f. p.1176. “Electronic circuit analysis and design”. Traducido por: Felipe Castro Pérez.
- NELSON, V. NAGLE, H. CARROLL, B. IRWIN, J. (1996). Análisis y diseño de circuitos lógicos digitales. México: Prentice Hall. Primera edición. p.842. “Digital logic circuit analysis and design”. Traducido por: Oscar A. Palmas V.

- TOCCI, Ronald. (1995). Sistemas digitales principios y aplicaciones. México: Prentice Hall. Quinta edición. p.823. "Digital systems principles and applications". Traducido por: Edmundo G. Urbina M.
- WARKEY, John F. (1997). Diseño digital principios y prácticas. México: Prentice Hall. S/f. p.743. "Digital design principles and practices". Traducido por: Gutiérrez R. Raymundo H.

MANUALES.

- NATIONAL SEMICONDUCTOR. (1981). Manual TTL y CMOS.
- NATIONAL SEMICONDUCTOR. (1992). Manual DATA MEMORY.
- MOTOROLA Inc. (1992). Fast and LS TTL. (DL121/D REV. 5).

OTRAS FUENTES.

- FRANCO, Zulay. (2001). Circuitos Electrónicos Digitales utilizando Dispositivos Lógicos Programables. Trabajo de Ascenso. UNEXPO. Puerto Ordaz.
- FRANCO, Zulay. (1997). Prácticas para laboratorio de técnicas digitales. Trabajo de Ascenso. UNEXPO. Puerto Ordaz.
- MARQUEZ, Alejandro. (1996). Electrónica Digital. Trabajo de Ascenso. UNEXPO. Puerto Ordaz.

Algunos sitios de consulta de INTERNET.

- <http://www.ti.com/>
- <http://www.national.com/>
- <http://www.cypress.com/>

ANEXO

DIAGRAMAS DE CIRCUITOS INTEGRADOS DIGITALES