Universidad Autónoma de Nuevo León
Facultad de Ingeniería Mecánica y Eléctrica
Coordinación de la División de Ingeniería Electrónica

Fundamentos de Diseño Digital
Sistemas Combinacionales

M. C. Cesar Augusto Leal Chapa

M. C. Juan Ángel Garza Garza

M. C. José Ángel Castillo Castro

M. C. Julián E. Hernández Venegas

Impreso en México.
ISBN: 978-607-433-589-7
Febrero 2011
Constancia de Número
Número Internacional Normalizado del Libro
Agencia Mexicana del ISBN
AGENCIA Mexicana ISBN, www.indautor.sep.gob.mx
No Radicación 84337

"Educación de calidad, un compromiso social"

Excelencia y Humanismo con Visión
FIME - UANL

Prólogo

Este libro es resultado del rescate el texto de Fundamentos de Diseño Digital que data de los años 70 que fue elaborado en la FIME UANL con los recursos disponibles en esa época tales como; máquina de escribir, correctores, fotocopiadora, duplicadora de tinta, dibujos hechos a mano o con ayuda de plantillas.

En esta edición se integra la colaboración de los profesores de la Academia de Electrónica Digital: Juan Ángel Garza Garza, José Ángel Castillo Castro y Julián Eduardo Hernández Venegas, quienes aportan su gran experiencia tanto académica como en el ejercicio profesional haciendo de esta obra un trabajo colegiado, logrando un documento moderno en forma electrónica del texto original.

Al inicio de los años 70 a la fecha se han perfeccionado las técnicas de enseñanza así como los recursos didácticos, que promueven en gran medida el desarrollo de la capacidad de los alumnos de aprender por cuenta propia.

Desde sus inicios este texto que apoya al curso que antes se titulaba Electrónica Lógica y hoy Electrónica Digital, cumplía y lo sigue haciendo con el propósito pasar de un aprendizaje meramente teórico a tener como principio la aplicación del conocimiento.

La mayoría de los temas tratados en la primera versión, aun siguen vigentes, que son conocimientos y conceptos básicos, los que se han mantenido aun con devenir tecnológico en el que se han presentado cambios radicales en los recursos y procedimientos.

Este libro es un recurso importante que contribuye a desarrollar la primera etapa de la competencia de Diseño de Sistemas Electrónicos Digitales, basados en la aplicación de los fundamentos teóricos y prácticos del álgebra booleana, aplicando la metodología de diseño para los sistemas combinacionales, de modo que se construyen prototipos con dispositivos de función fija para verificar su correcto funcionamiento**.**

Por ultimo deseo mencionar que la motivación que los autores de este libro han recibido, ha sido en gran medida una estrategia promovida por el director de ésta facultad el Ingeniero Esteban Báez Villarreal, brindándonos un gran apoyo, gestión y recursos para cumplir con los indicadores externos de calidad académica que nos mantienen vigentes y competitivos en ámbito universitario nacional e internacional, motivo por el cual le deseamos manifestarle nuestro agradecimiento.

M.C. Cesar Augusto Leal Chapa

ÍNDICE

Síntesis

En el capítulo I se revisa la terminología y los conceptos asociados a los Sistemas Digitales por medio de la comparación de sus características con respecto a los sistemas Analógicos.

En el capitulo 2 se aborda el tema de sistemas numéricos de notación posicional con la finalidad de conocer su naturaleza la cual es compartida por los sistemas Decimal, Binario, Octal y Hexadecimal que son los más ampliamente usados en la actualidad en los sistemas digitales como una herramienta para representar cantidad.

En el capitulo 3 es el tema central de este texto, en cual se plantea los principios del Álgebra Booleana como una herramienta para la representación de ideas de solución de problemas de ingeniería a través de símbolos, tablas etc. y otros recursos que posteriormente servirán para construir circuitos electrónicos digitales en forma óptima.

En el capitulo 4 se muestran los más importantes códigos numéricos y alfanuméricos representados en forma de unos y ceros, se analizan también sus características y se enuncian algunas de sus aplicaciones más comunes.

En el capitulo 5 se tratan las técnicas y métodos para la minimización de funciones booleanas con el propósito de reducir costos y la complejidad en la implementación de los circuitos digitales.

En el capitulo 6 se propone una metodología para el diseño óptimo de los Sistemas Combinacionales, así como algunos casos en donde se solucionan los sistemas que no están completamente especificados, todo esto que da como resultado la posibilidad de construir bloques, del tipo aritmético y lógico como el sumador o comparador o también bloques que dan solución a problemas industriales típicos.

1 LOS SISTEMAS DIGITALES SE ORIGINARON EN UN MUNDO ANALÓGICO.

1.0 Conceptos Básicos.

Para establecer una idea clara respecto a la definición de sistemas digitales y analógicos dirijamos nuestra atención hacia el mundo físico en que se originan.

Al referirnos a parámetros físicos como, temperatura, velocidad, aceleración, etc. nos topamos frecuentemente con la necesidad de medirlos, procesar la información medida e incluso controlar tal parámetro.

La medición, manipulación y control de las variables físicas se había efectuado tradicionalmente por medio de dispositivos que tienen un comportamiento análogo a la variable.

Por este motivo a los parámetros antes mencionados y a sus instrumentos de medición y control se les da el nombre de Analógicos. De hecho nuestro medio es un mundo cuyas variables físicas son en su mayoría analógicas.

Así por ejemplo, en un termómetro, la columna de mercurio que se encuentra dentro de él, aumenta o disminuye dependiendo del aumento o disminución de la temperatura del medio que lo rodea figura. 1.1.

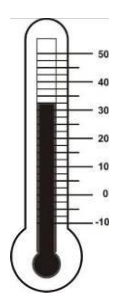


Figura. 1.1Termómetro de mercurio

Algo semejante sucede con un dinamómetro, con un manómetro o con un galvanómetro, Figura. 1.2. En cada uno de los casos, la fuerza, presión o corriente eléctrica puede medirse mediante la deflexión de una aguja indicadora sobre la superficie graduada en las unidades correspondientes a cada parámetro.

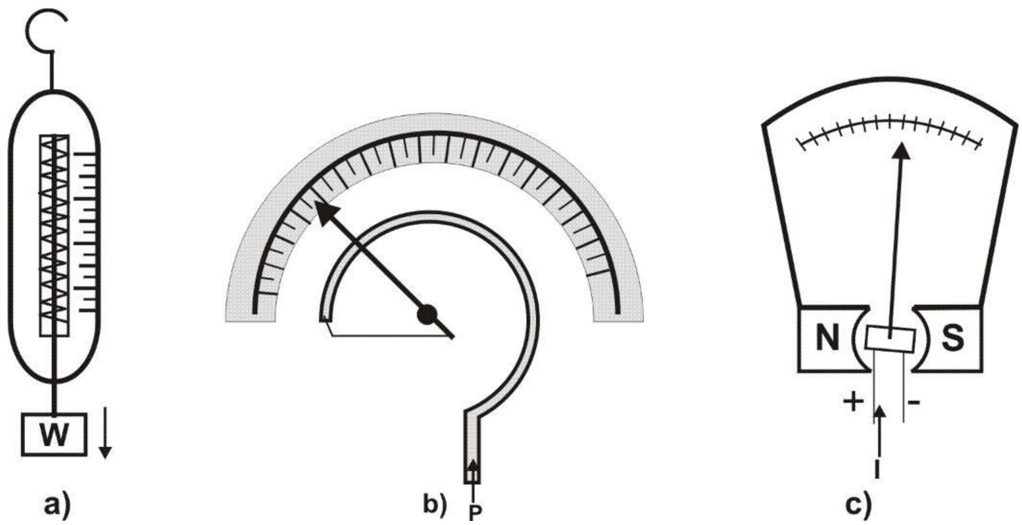


Figura. 1.2 Dispositivos de medición. a) Dinamómetro, b) Manómetro c) Galvanómetro.

1.1 Conceptos de Resolución y Exactitud.

Establezcamos la definición de dos conceptos, importantes, el primero de ellos es la Resolución de un sistema de medición, este término se refiere a la mínima separación de dos valores numéricos sucesivos que pueden resultar del proceso de medición. A esta mínima separación se le llama Unidad de Resolución y limita la exactitud del sistema. Cuando un valor cae entre dos valores numéricos sucesivos de resolución mínima, se le tendrá que dar un valor numérico mayor o menor a su valor real. Por ejemplo si dos personas encuentran una moneda de 5 centavos y se la quieren repartir, a uno de ellos le tocaran 3 centavos y al otro 2 centavos puesto que la Unidad Mínima de Resolución en nuestro sistema monetario es el centavo. En este caso no es posible una división Exacta y el error en ambas cantidades es una media de la unidad de resolución. El término Exactitud está relacionado con la calidad del proceso de medición.

El incremento de la exactitud usualmente requiere el perfeccionamiento de la técnica o dispositivo de medición. Por ejemplo, de una regla no obtendremos el mismo grado de exactitud que al usar un micrómetro.

En el ejemplo del termómetro que mencionábamos en el punto anterior pueden apreciarse claramente los conceptos de exactitud y resolución. Figura. 1.3.

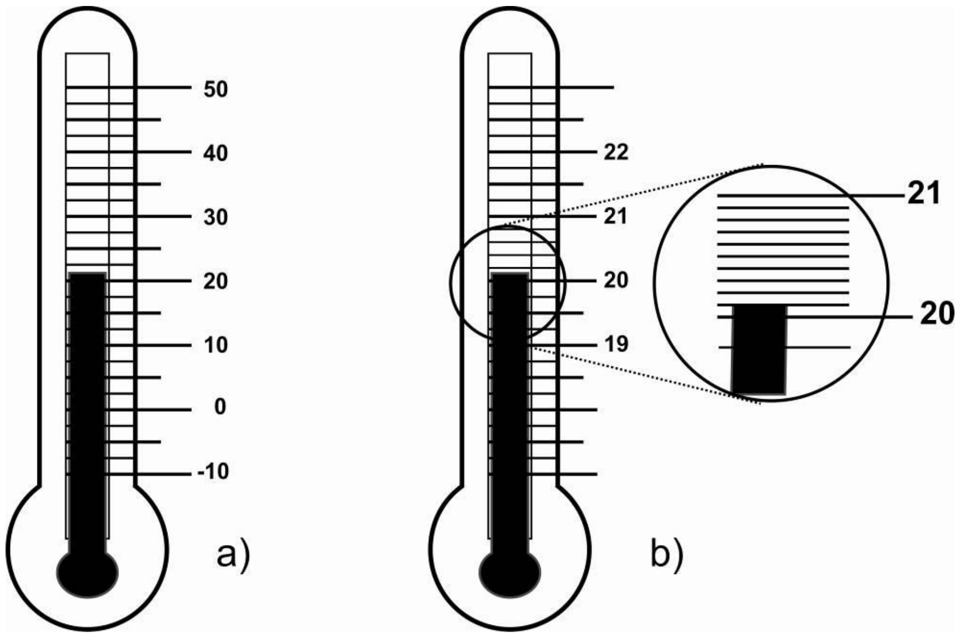


Figura. 1.3 Al disminuir el intervalo entre dos valores numéricos sucesivos en una medición se aumenta la exactitud.

A un observador que se le pregunte la temperatura en el dibujo de la Figura. 1.3a, seguramente responderá 20°C. Al amplificar la sección del termómetro entre 20° y 21° e imaginariamente aumentar la escala (Figura.1.3b) podemos apreciar que un valor más aproximado a la temperatura real será 20.1°C.

La exactitud de una medición puede incrementarse reduciendo el intervalo entre dos valores numéricos sucesivos. Este incremento de resolución lógicamente aumenta el valor numérico de la medición, para el ejemplo de 20 a 20.1 o sea de 2 a 3 dígitos.

1.2 ¿Qué es un Sistema Digital?

En la manipulación de un parámetro medido, en su proceso e incluso en la conversación cotidiana es difícil emplear el valor numérico exacto de una variable, y en lugar de él se usa un valor numérico aproximado que es representativo de su valor real. La temperatura en el ejemplo del termómetro leída por un observador, era de 20°C mientras que en realidad es un valor entre 20° y 21°C.

En la adquisición de un dato y en el proceso de medición, intervienen los conceptos de exactitud, resolución y el tiempo en el cual se determina el valor numérico de la variable medida. Comúnmente a este proceso de adquisición se le conoce como "digitalización" de una variable. Este término indica el hecho de que una variable original se reemplaza por un valor numérico cuyos dígitos representan la magnitud de la variable en un tiempo dado. Por ejemplo una vez convertida la altura de la columna de mercurio de un termómetro a un valor digital, la cantidad puede procesarse, almacenarse, controlarse, etc.

Entonces un sistema digital se puede definir como un sistema que procesa información en forma digital (numérica) en vez de procesar a la misma variable en forma analógica.

1.3 Sistemas continuos y no continuos.

Para definir estos sistemas comparemos el funcionamiento de un termómetro de mercurio y uno digital. En el primero, cualquier cambio en la temperatura corresponderá a un cambio en la altura de la columna de mercurio. El termómetro digital convertirá periódicamente la temperatura a un valor numérico y lo mostrará en una pantalla. Un cambio en la temperatura no se indicará hasta que sea lo suficientemente grande para cambiar al dígito próximo mayor o menor. Si no sucede esto el valor indicado permanecerá igual.

Por este motivo a un sistema analógico se le asocia con el término "continuo" y a un sistema digital con el término "no continuo".

1.4 Representación de información y cantidad.

En la Figura 1.4 se muestran dos formas para detectar e indicar la velocidad de un motor. El primero es un sistema analógico y el segundo es un sistema digital. En el sistema analógico aparece conectado a la flecha del motor un tacómetro generador, que produce un voltaje proporcional a la velocidad del motor. Este voltaje pasa a un voltímetro, en cuya carátula la graduación está marcada en R.P.M. (Revoluciones por Minuto). En este caso el dato Velocidad, está representado por un voltaje continuo que puede tener un rango de 0 a 10 voltios, manifestado en forma también continua por la aguja del voltímetro Figura.1.4a.

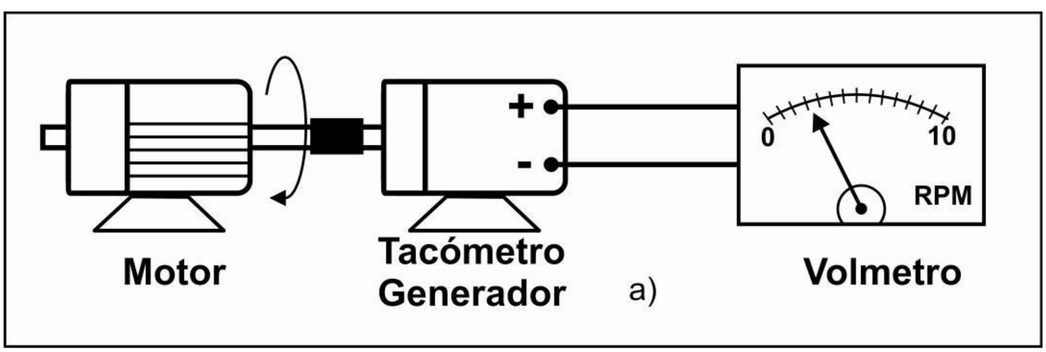
En el sistema digital la flecha del motor tiene una marca reflejante que es detectada por medio de una fotocelda. Cada pulso generado por la fotocelda al pasar la marca equivale a una revolución. Un contador digital cuenta la cantidad de pulsos que por unidad de tiempo en este caso minutos, será igual a las R.P.M. Figura. 1.4b.

En este sistema el dato Velocidad no está representado por un voltaje continuo, sino por pulsos, es decir un voltaje discreto, un nivel alto y un nivel bajo que corresponden a los voltajes típicos de 0 volts y 5 volts de corriente directa.

En ambos casos la Información se representa por medio de un voltaje.

La cantidad de voltaje en el sistema analógico es proporcional a la velocidad. En el sistema digital la velocidad es proporcional a la cantidad de pulsos.

La representación de Cantidad puede efectuarse por medio de voltajes, ya sea en forma analógica o en forma digital.



a)

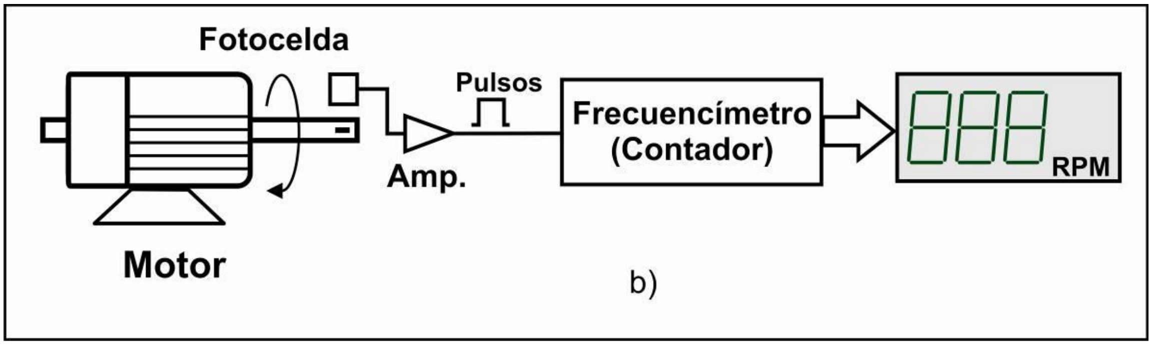


Figura. 1.4 Detección y lectura de velocidad en la flecha de un motor.

a) Sistema Analógico, Tacómetro Voltímetro, b) Sistema Digital, Fotocelda Contador de Pulsos por Unidad de Tiempo.

En la Figura.1.5 se muestra un circuito formado por una fuente, un potenciómetro lineal con una escala de 0 a 9 y un foco.

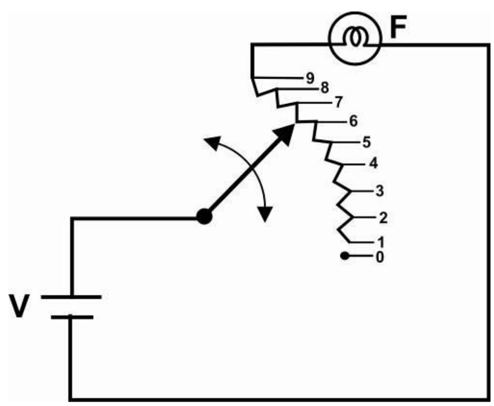


Figura.1.5 Circuito para ilustrar la representación de cantidad en forma analógica.

De acuerdo a la posición en que se encuentre el potenciómetro, existirá una intensidad luminosa proporcional al valor de la resistencia, desde "0" (circuito abierto) hasta la máxima posible (circuito cerrado).

Imaginemos que un observador trate de distinguir entre los 10 niveles, con toda seguridad será difícil apreciar el nivel 4 del 5 o el 5 del 6, sin embargo es simple detectar el foco completamente apagado (posición "0") o completamente encendido (posición "9").

Para un observador humano es difícil detectar niveles analógicos. Lo es también para un circuito electrónico, en el cual se elevará considerablemente el costo y bajará su confiabilidad. Por este motivo los circuitos digitales electrónicos trabajan solamente con dos niveles de voltaje.

Un nivel bajo llamado "0" cero lógico y un nivel alto llamado "**1**" uno lógico como se muestra en la Figura. 1.6

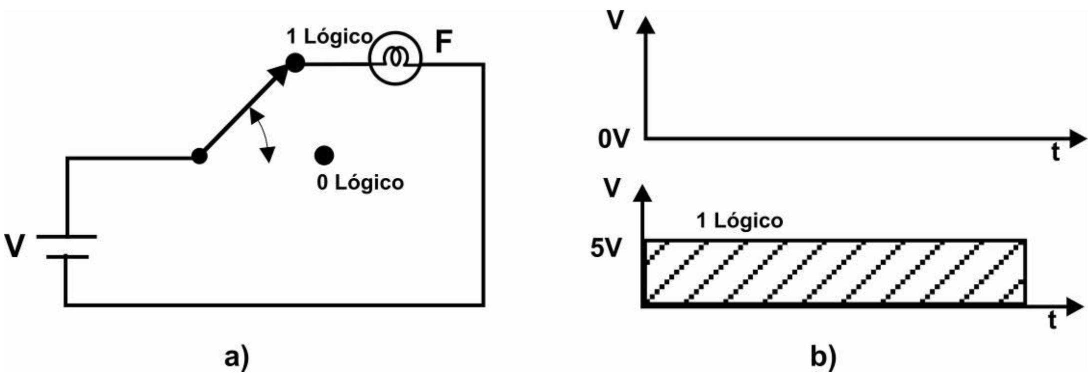


Figura.1.6a circuito simplificado, el potenciómetro se cambió por un interruptor

Figura. 1.6b niveles de voltaje para un 0 y un 1logicos

PROBLEMAS PROPUESTOS.

1.- ¿Qué diferencia existe entre el funcionamiento de un sistema digital y un sistema analógico?

2.- ¿En qué consiste la conversión analógica digital? ¿Y porque es necesaria?

3.- En la Figura 1.3b) aparecen un termómetro y una sección amplificada del mismo termómetro.

a) ¿Cuál es la unidad de resolución en ambos casos?

b) ¿Cuál graduación puede ofrecer una lectura más precisa?

4.- ¿Cuál es el concepto de continuidad (o de variable continua)?

5.- ¿Como se representa la información en un sistema digital y en un sistema analógico?

6.- ¿Como se representa la cantidad en un sistema digital?

2 SISTEMAS NUMÉRICOS

2.0 Introducción

Desde la más remota antigüedad el hombre tuvo la necesidad de contar, fue entonces cuando los números tomaron una gran importancia, aquellos símbolos que representaban cantidades evolucionaron de tal forma que estructuraron sistemas numéricos, como es el caso de los números romanos, los griegos y los egipcios.

Como seguramente hemos tenido alguna experiencia con el sistema numérico romano lo tomaremos para ilustrar el tipo de notación numérica que empleaba, en la Figura 2.1 aparecen algunos de sus símbolos.

I	1	**C**- 100
V	5	**D** - 500
X	10	**M** - 1000
L	50	

Figura 2.1 Símbolos del sistema numérico romano y su equivalente en decimal.

Existían ciertas reglas, por ejemplo, cuando un **I** (uno) aparecía antes de un **V** (cinco), "IV", el símbolo menor era restado al mayor, así el número "**IV**" = (5-1) = 4.

Por el contrario cuando el signo menor aparece delante del mayor se suman, el número "**VI**" = (5+1)= 6. Nótese que en ambos números Los símbolos **I** y **V** conservan su valor independientemente de la posición en el número, un **V** (cinco) nunca podrá ser un 50 o un 500.

2.1 Sistemas numéricos de Notación Posicional

Con una antigüedad aproximada de 2000 años y originario de la India nuestro actual sistema numérico, el "decimal" fue introducido a Europa por los Árabes, de allí el nombre de números arábigos. A cada uno de sus símbolos del 0 al 9 se les conoce como "dígito" raíz Latina que significa dedo. Supuestamente, se usan 10 dígitos porque el hombre, posee 10 dedos, que empleaba como herramientas para contar.

El sistema decimal tiene dos características importantes. Una es el concepto del "cero" que indica ausencia de cantidad o valor y la otra es la notación posicional, para explicarla usaremos el siguiente ejemplo.

Imaginemos un conteo en decimal que inicia por supuesto en cero, al llegar a 9 alcanzaremos el dígito de mayor valor, si incrementamos nuestro conteo generaremos un acarreo, como se indica en la figura. 2.2.

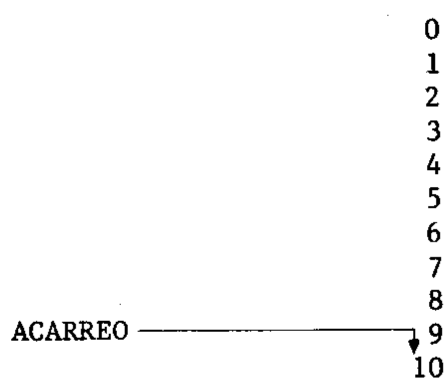


Figura 2.2 Generación del acarreo para un conteo en decimal

Este acarreo forma el número 10 diez, se dice que el "0" cero ocupa la posición de las unidades mientras que el "1" uno ocupa la posición de las decenas. Este proceso continuara cada vez que se alcanza el 9 en la posición de las unidades. Cuando aparece un 9 en la posición de las decenas se genera un acarreo a la posición de las centenas y así sucesivamente.

Nótese que un 1 en la posición de las decenas tiene un valor o "peso"10 veces mayor que un uno de la posición de las unidades. Lo mismo sucede con un I de la posición de las centenas, es 10 veces mayor que un 1 de la posición de las decenas.

Definiremos entonces "peso" de un dígito, como el valor que toma (ese dígito) según la posición que tenga en el número.

De aquí que el nombre "Sistema numérico de notación posicional" se aplica a los sistemas numéricos donde los dígitos que forman un número tienen diferentes pesos de acuerdo a su posición (en el número).

La base del sistema numérico decimal es 10. La base es igual al número de símbolos que posee un sistema numérico. El dígito mayor siempre es una unidad menor que la base. Cada posición multiplica el valor del dígito por la base elevada a esa posición. Además un acarreo de una posición a la próxima mayor, incrementa su peso por base veces. Esto es valido para un sistema de notación posicional de cualquier base.

En la figura. 2.3 se muestran los sistemas numéricos de notación posicional más comunes.

R-BASE	SIST. NUMERICO	R DIGITOS EMPLEADOS
2	BINARIO	0, 1
8	OCTAL	0, 1, 2, 3, 4, 5, 6, 7.
10	DECIMAL	0, 1, 2, 3, 4, 5, 6, 7, 8, 9.
16	HEXADECIMAL	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

Figura 2.3 Sistemas numéricos más comunes

Los sistemas de notación posicional han sido tan ampliamente aceptados que raramente los analizamos. Tomemos un número decimal, por ejemplo el 258. La posición de los dígitos en el número se indica en la figura 2.4, nótese que la posición inicia en 0.

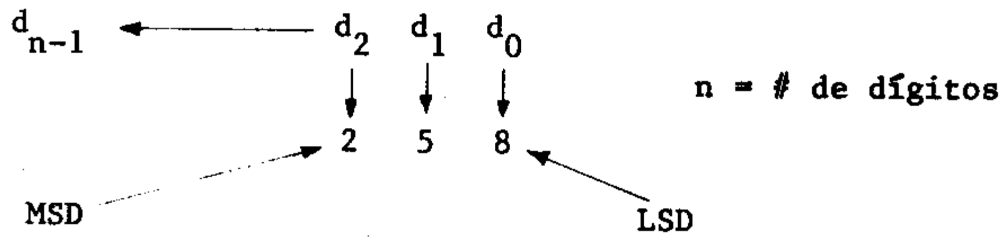


Figura 2.4 Posición de los dígitos del número 258

El número esta formado por tres dígitos 2, 5 y 8. El dígito de menor peso es el 8 y se le conoce como **LSD** siglas en inglés de Least Significant Digit, el dígito de mayor peso es el 2 y se le conoce como **MSD** Most Significant Digit.

El 8 ocupa la posición de las unidades y pesa 8 X 1= 8 unidades. El 5 ocupa la posición de las decenas y pesa 5 X10= 50 unidades. El 2 ocupa la posición de las centenas y pesa 2 X 100 = 200 unidades.

(2X100)+ (5X 10)+ (8 X 1)= 258

$2(10)^2 + 5(10)^1 + 8(10)^0 = 258$

Entonces un número decimal de N dígitos puede tomarse como una sumatoria de sus coeficientes multiplicados por la base elevada a la posición en que se encuentran.

$$N_{10} = a_{n-1}\ (10)^{n-1} + a_{n-2}\ (10)^{n-2} + \dots + a_1\ (10)^1 + a_0\ (10)^0$$

$$N_{10} = \sum_{i=0}^{n-1} a_i\ (10)^i$$

Donde: **a = coeficiente**

n = cantidad de coeficiente

N=número

A esta ecuación se le conoce como "Expresión Sumatoria". La notación posicional de un número es una expresión sumatoria abreviada donde se omiten los signos de suma y los pesos de cada posición.

$$N_{10} = a_{n-1} \quad a_{n-2} \ \cdot \ \cdot \ \cdot \ \cdot \ \cdot \ \cdot \ \cdot \quad a_1 \ a_0$$

$$N_{10} = 258$$

La expresión sumatoria puede generalizarse para cualquier sistema numérico.

$$N_r = \sum_{i=0}^{n-1} a_i \ (r)^i$$

Al considerar números fraccionarios tenemos.

$$N_r = a_{n-1} \ (r)^{n-1} + \ . \ . \ . \ a_0 \ (r)^0 + a_{-1} \ (r)^{-1} + \ . \ . \ . \ a_{-m} \ (r)^{-m}$$

$$N_r = \sum_{i=-m}^{n-1} ai \ (r)^{\ i}$$

Donde:

r = base del sistema

m = número de dígitos fraccionarios

Ejemplo 2.0

Exprese el número 258.25 de acuerdo a la expresión sumatoria.

$$\begin{aligned} N_{10} &= 258.25 \\ &= 2(10)^2 + 5 \ (10)^1 + 8 \ (10)^0 + 2 \ (10)^{-1} + 5 \ (10)^{-2} \\ &= 200 + 50 + 8 + .2 + .05 \\ N_{10} &= 258.25 \end{aligned}$$

Cuando se trabaje con sistemas numéricos de diferentes bases debe indicarse por media de un subíndice la base en que se encuentra un número.

Ejemplo 2.1

a) 258 $_{(10)}$

b) 1010 $_{(2)}$

c) 357 $_{(8)}$

d) A32 $_{(16)}$

2.2 Sistema numérico Binario

La base del sistema numérico binario es 2, por lo tanto se usan solamente dos símbolos "0" y "1" para la representación de cualquier número o cantidad.

Un número mayor que "1" puede representarse empleando el mismo método que en decimal (un número mayor que 9 genera un acarreo que indica una decena). Entonces la representación binaria de 2_{10} es 10_2, (uno cero en base 2).

Decimal	Binario	Octal	Hexadecimal
$N_{(10)}$	**$N_{(2)}$**	**$N_{(8)}$**	**$N_{(16)}$**
0	0	0	0
1	1	1	1
2	10	2	2
3	11	3	3
4	100	4	4
5	101	5	5
6	110	6	6
7	111	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

Figura 2.5 Números del 0 al 20 en base 10 con su equivalente en binario, octal y hexadecimal.

A los dígitos binarios se les da el hombre de BIT, que es una contracción de los términos en inglés **Binary-Digit**. Al dígito de mayor peso en un número se le llama **MSB** del inglés (MOST SIGNIFICANT BIT) y al dígito de menor peso se le llama **LSB** (LEAST SIGNIFICANT BIT).

2.2.1 Conversión de Binario a Decimal

La conversión de binario a decimal se efectúa por medio de la expresión sumatoria.

$$N_{10} = \sum_{i=0}^{n-1} a_i (2)^i$$

Ejemplo 2.2

Convierta a base 10 el número binario 111001_2

$$111001_2 \longrightarrow N_{10}$$

$$N_{10} = 1(2)^5 + 1\ (2)^4 + 1\ (2)^3 + 0\ (2)^2 + 0\ (2)^1 + 1\ (2)^0$$

$$= 32 + 16 + 8 + 0 + 0 + 1$$

$$N_{10} = 57_{10}$$

Ejemplo 2.3

Convierta a base 10 el número binario 1101.11

$$1101.11_2 \longrightarrow N_{10}$$

$$N_{10} = 1\ (2)^3 + 1\ (2)^2 + 0\ (2)^1 + 1\ (2)^0 + 1\ (2)^{-1} + 1\ (2)^{-2}$$

$$= 8 + 4 + 0 + 1 + 0.5 + 0.25$$

$$N_{10} = 13.75_{10}$$

2.2.2 Conversión de Decimal a Binario

El uso de la expresión sumatoria para convertir un número en base 10 a base 2 no es muy útil puesto que es difícil pensar en otro sistema numérico que no sea decimal.

Ejemplo 2.4 Convierta el número $23_{(10)}$ a Binario.

$$23_{10} \longrightarrow N_2$$

$$N_2 = 2\ (10)^1 + 3\ (10)^0$$

$$= 20 + 3$$

$$N_2 \neq 23$$

Aparentemente cometimos un error, sin embargo el problema fue que es necesario pensar en binario. Nótese que el 2, 3 y 10 están escritos en decimal y no en binario.

Intentemos de nuevo.

$$N_2 = \sum_{i=0}^{n-1} ai\ (1010)^i$$

$$= 10\ (1010)^1 + 11\ (1010)^0$$

$$= 10100 + 11$$

$$N_2 = 10111_2$$

Para comprobar usemos la expresión sumatoria:

$$10111_2 \longrightarrow N_{10}$$

$$N_{10} = 1\ (2)^4 + 0\ (2)^3 + 1\ (2)^2 + 1\ (2)^1 + 1\ (2)^0$$

$$= 16 + 0 + 4 + 2 + 1$$

$$N_{10} = 23_{10}$$

Existen dos métodos más cómodos para la conversión de decimal a binario. Se lee conoce como **Método de Extracción de Potencias y Método de los Residuos.**

El método de extracción de potencias consiste en restar la máxima potencia de 2 que pueda contener el # decimal, repitiendo esta operación con el resultado hasta agotar el # 10. El método es útil solo para números pequeños. 23

Ejemplo 2.5

$$22_{10} \longrightarrow N_2$$

$$\begin{array}{rl} 22 & \\ -\;16 & 2^4 \\ \hline 6 & \\ -\;4 & 2^2 \\ \hline 2 & \\ -\;2 & 2^1 \\ \hline 0 & \end{array}$$

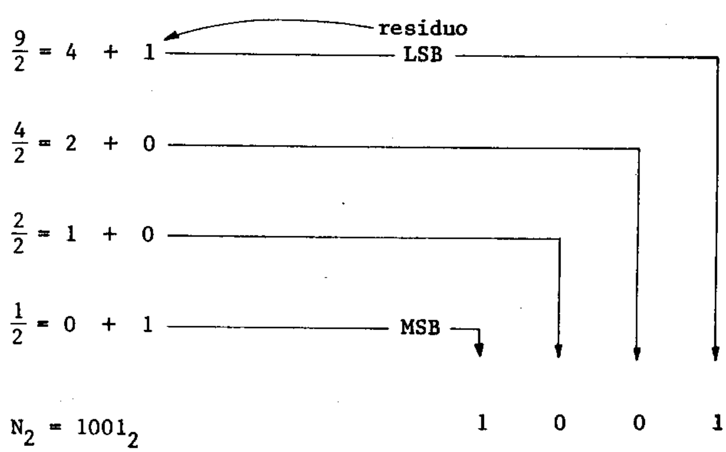
		d4	d3	d2	d1	d0
N_2	=	1	0	1	1	0_2

Los coeficientes del número binario son un "1" en la posición de la potencia restada y "0" para la posición no restada.

El Método de los Residuos consiste en dividir repetidamente el número decimal entre la base del sistema al que deseamos transformar, e ir registrando sucesivamente los residuos. Estos residuos leídos en orden inverso nos dan el equivalente del número.

Ejemplo 2.6

$$9_{10} \longrightarrow N_2$$



$$\frac{9}{2} = 4 + 1$$

$$\frac{4}{2} = 2 + 0$$

$$\frac{2}{2} = 1 + 0$$

$$\frac{1}{2} = 0 + 1$$

$$N_2 = 1001_2$$

Ejemplo 2.7

$$24_{10} \longrightarrow N_2$$

$$\frac{24}{2} = 12 + 0 \text{ —— LSB}$$

$$\frac{12}{2} = 6 + 0$$

$$\frac{6}{2} = 3 + 0$$

$$\frac{3}{2} = 1 + 1$$

$$\frac{1}{2} = 0 + 1 \text{ —— MSB}$$

$$1 \quad 1 \quad 0 \quad 0 \quad 0_2$$

$$N_2 = 11000_2$$

2.3 Sistema numérico Octal

En este sistema la base es 8, por lo tanto existen solo 8 símbolos que van del 0 al 7. Un número mayor que 7 no puede escribirse (8) puesto que este símbolo no existe en el sistema, la expresión correcta es $10_{(8)}$ que se lee “uno cero” base 8. Nótese que un acarreo de un "1" tiene un peso de 8 unidades.

2.3.1 Conversión de Octal a Decimal

Método Expresión Sumatoria

Ejemplo 2.8

$$147_8 \longrightarrow N_{10}$$

$$N_{10} = 1\ (8)^2 + 4\ (8)^1 + 7\ (8)^0$$

$$= 64 + 32 + 7$$

$$N_{10} = 103_{10}$$

2.3.2 Conversión de Decimal a Octal

Método de los Residuos

Ejemplo 2.9

$$103_{10} \longrightarrow N_8$$

$$\frac{103}{8} = 12 + 7 \quad \text{LSD}$$

$$\frac{12}{8} = 1 + 4$$

$$\frac{1}{8} = 0 + 1 \quad \text{MSD}$$

$$1 \quad 4 \quad 7$$

$$N_8 = 147_8$$

2.4 Sistema numérico Hexadecimal

El sistema numérico hexadecimal es un sistema numérico importante usado en computadoras. Su base es 16 y sus símbolos van del **0** al **9** y de **A** a **F**. Como se muestra en la figura 2.5. Un acarreo de un "1" tiene un peso de 16 unidades. Por lo tanto un $10_{(16)}$ (uno, cero en base 16) no equivale a diez en decimal.

2.4.1 Conversión de Hexadecimal a Decimal

Método: Expresión Sumatoria

Ejemplo 2.10

$$1A9_{16} \longrightarrow N_{10}$$

$$N_{10} = 1(16)^2 + 10(16)^1 + 9(16)^0$$

$$= 256 + 160 + 9$$

$$N_{10} = 425_{10}$$

Nótese que la tetra "A" se cambia por su equivalente numérico.

2.4.2 Conversión de Decimal a Hexadecimal

Método de los Residuos

Ejemplo 2.11

$$1324_{10} \longrightarrow N_{16}$$

$$\frac{1324}{16} = 82 + 12 \text{ ——— LSB}$$

$$\frac{82}{16} = 5 + 2$$

$$\frac{5}{16} = 0 + 5 \text{ ——— MSB}$$

$$5 \quad 2 \quad 12$$

$$N_{16} = 52C_{16}$$

Ejemplo 2.12

$$432_{10} \longrightarrow N_{16}$$

$$\frac{432}{16} = 27 + 0 \text{ ——— LSB}$$

$$\frac{27}{16} = 1 + 11$$

$$\frac{1}{16} = 0 + 1 \text{ ——— MSB}$$

$$1 \quad 11 \quad 0$$

$$N_{16} = 1\,B\,0_{16}$$

2.5 Conversión Binario ↔ Octal

El sistema octal puede ser un método conveniente para reducir la longitud de un número binario, esto es muy útil cuando se tienen listados en binario por ejemplo, el contenido de la memoria de una computadora digital. En la figura 2.6 aparecen los 8 símbolos en octal con su correspondiente en binario. Nótese que para expresar un dígito octal, solo son necesarios 3 bit's, esta relación surge de que la base octal e igual a 2^3.

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

Figura 2.6 Para cada dígito octal corresponden 3 bit's.

La conversión de binario a octal se obtiene dividiendo el número binario en grupos de 3 bits a partir del punto decimal, tanto para la parte entera como la parte fraccionaria.

Ejemplo 2.13 Convierta

$$11001110.1011_2 \longrightarrow N_8$$

$$\begin{array}{cccccc} 011 & 001 & 110 & . & 101 & 100 \\ \downarrow & \downarrow & \downarrow & & \downarrow & \downarrow \\ 3 & 1 & 6 & . & 5 & 4 \end{array}$$

$$N_8 = 316.54_8$$

Usando el mismo método podemos convertir un número de base 8 a base 2.

Ejemplo 2.14 Convierta:

$$345_8 \longrightarrow N_2$$

$$\begin{array}{ccc} 3 & 4 & 5 \\ \downarrow & \downarrow & \downarrow \\ 11 & 100 & 101 \end{array}$$

$$N_2 = 11100101_2$$

2.6 Conversión Binario ⇔ Hexadecimal

La representación de un número binario en hexadecimal es una mejor alternativa a la representación en octal. La relación parte de que la base hexadecimal 16 es igual a 2^4. En la figura 2.7 se muestran los dígitos hexadecimales y su correspondiente en binario.

HEXADECIMAL	BINARIO
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
A	1010
B	1011
C	1100
D	1101
E	1110
F	1111

Figura 2.7 Para cada dígito Hexadecimal corresponden 4 Bit's.

La conversión de binario a hexadecimal se obtiene empleando el mismo método que en octal, solo que aquí se toman 4 bit's por cada dígito base 16.

Ejemplo 2.15 Convierta: $1011110111_{(2)} \rightarrow N_{(16)}$

$$\begin{array}{ccc} 10 & 1111 & 0111 \\ \downarrow & \downarrow & \downarrow \\ 2 & F & 7 \end{array}$$

$$N_{16} = 2F7_{16}$$

Ejemplo 2.16 Convierta: $A51_{(16)} \rightarrow N_{(2)}$

$$\begin{array}{ccc} A & 5 & 1 \\ \downarrow & \downarrow & \downarrow \\ 1010 & 0101 & 0001 \end{array}$$

$$N_2 = 101001010001_2$$

2.7 Conversión Octal ⇔ Hexadecimal

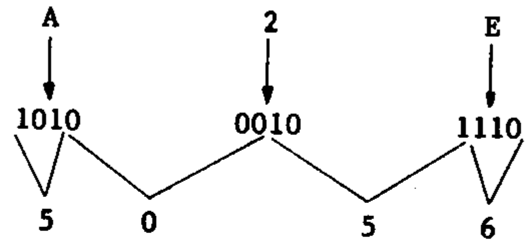
Un número en base 8 puede convertirse a un número base 16 y viceversa pasando por binario.

Ejemplo 2.17 Convierta: $472_{(8)} \rightarrow N_{(16)}$.

4	7	2	OCTAL
↓	↓	↓	
100	111	010	BINARIO
1	0011	1010	
↓	↓	↓	
1	3	A	HEXADECIMAL

$$472_8 = 13A_{16}$$

Ejemplo 2.18 Convierta: $A2E_{(16)} \rightarrow N_{(8)}$



$A2E_{16} = 5056_8$

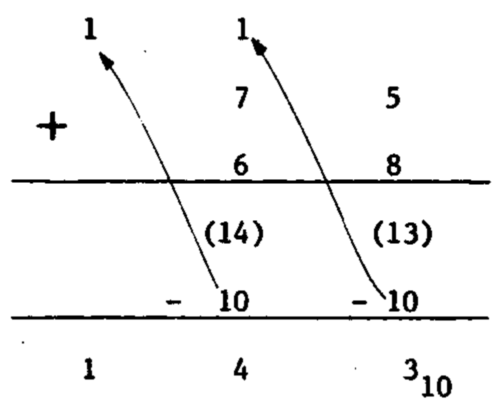
2.8 Aritmética Binaria, Octal y Hexadecimal

El método pare efectuar operaciones aritméticas es básicamente el mismo para todos los sistemas numéricos de notación posicional. Revisemos entonces el procedimiento de la suma "base 10".cuyo método seguramente lo efectuamos en forma mecánica.

Ejemplo:

$$\begin{array}{r} \mathbf{20} \\ + \underline{\mathbf{46}} \\ \mathbf{66} \end{array}$$

La suma de 2 números se efectúa columna por columna. Podemos observar en el ejemplo anterior que la suma en ambas columnas no fue mayor que 9. Veamos el siguiente ejemplo:



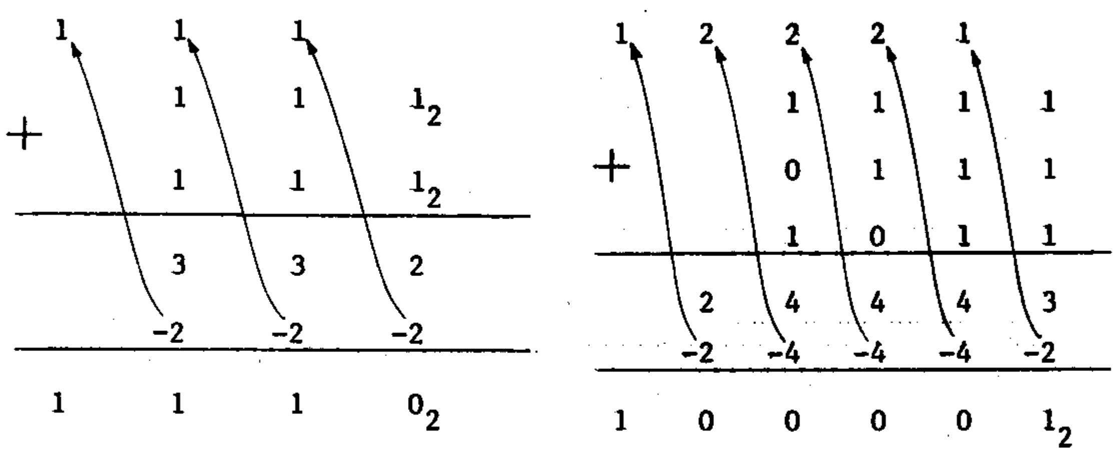
1 4

En este caso, 5 + 8 de la primer columna suman 13, este número es mayor que 9, lo cual nos indica que debe generarse un acarreo de 10 unidades a la siguiente columna. El dígito restante es una diferencia entre 10 y 13 -3. Lo mismo sucede en la posición de las decenas con el 14, que genera un acarreo hacia la posición de las centenas.

2.8.1 Suma Binaria

El método es el mismo que en decimal

Ejemplo 2.19



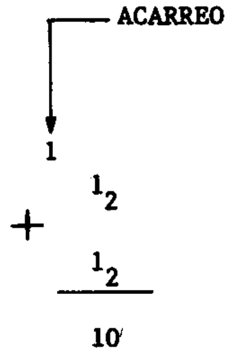
a) b)

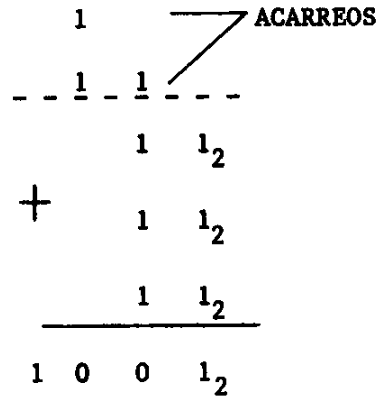
Nótese que en el ejemplo 2.19 b) se genera un acarreo igual a 2, debido a que la suma de la columna excedió 2 veces la base.

Existen otros dos métodos para sumar en base 2, en ambos casos es necesario pensar en binario.

METODO 1.- Se base en el hecho de que 1_2 + 1_2 es igual a cero y se acarrea 1_2, si existen varios 1's en la columna, cada par de unos sumados genera un acarreo.

Ejemplo 2.20

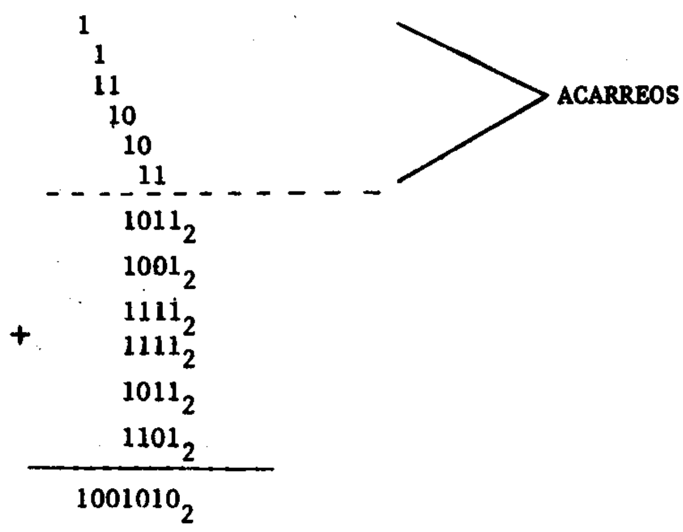




METODO 2.-

En el siguiente método existe menos probabilidad de error, consiste en sumar todos los unos de la columna, dar el resultado en binario, escribir el dígito de menor peso en su columna correspondiente y acarreara las siguientes columnas los dígitos restantes.

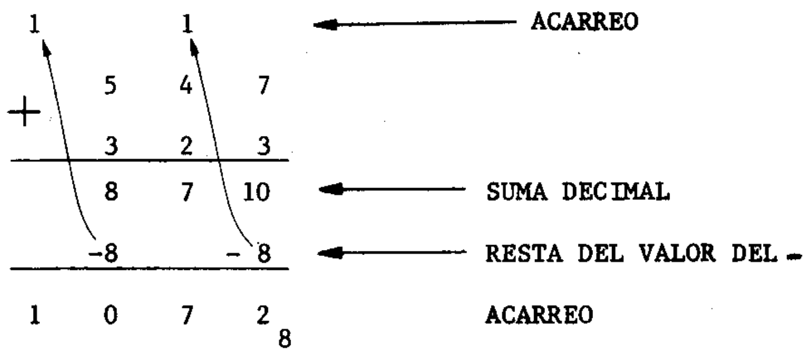
Ejemplo 2.21



La suma de la primera columna es igual a seis 110_2 se deja el 0 de menor peso y se acarrea el 11, y así sucesivamente.

2.8.2 Suma Octal

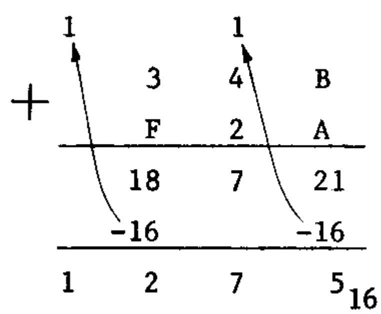
Ejemplo 2.22 Sume los números 547_8 y 323_8.



En la posición de las unidades o LSD se encuentran un 7 + 3 = 10 este número es mayor que 7, al restarle la base, se genera un acarreo a la siguiente columna con un peso de 8 unidades. En la posición de las unidades queda la diferencia entre 10 y 8 = 2.

2.8.3 Suma Hexadecimal

Ejemplo 2.23 Sume los números $34B_{16}$ y $F2A_{16}$



En la columna de menor peso, la literal se cambia por su valor numérico, B =11, A=10. La suma es mayor que 16 es necesario entonces restarle la base. El acarreo generado tendrá un peso de 16 unidades.

Ejemplo 2.24

$$\begin{array}{cccc} & 3 & C & 4 \\ + & 9 & 1 & 7 \\ \hline & (12) & (13) & (11) \\ \hline & C & D & B_{16} \end{array}$$

En este ejemplo la suma de cada columna es menor que 16 y mayor que 9, por lo tanto es necesario cambiar los números resultantes por su letra equivalente.

2.8.4. Resta

El procedimiento de la resta en base 10 es el mismo para los sistemas de notación posicional de diferentes bases.

Ejemplo 2.25

Efectué la siguiente resta decimal 45_{10} - 26_{10}

$$\begin{array}{cccc} & 3 & & \text{PRESTAMO} \\ - & \cancel{4} & (10) + 5 & \\ & 2 & 6 & \\ \hline & 1 & 9 & \end{array}$$

La resta al igual que la suma, se inicia con el dígito de menor peso, 5 menos 6 no se puede restar entonces pedimos un préstamo al dígito de la siguiente columna (decenas) el 4. La mínima cantidad que nos puede prestar es 1 (una decena), y se le llama "préstamo”.

Al sumar el préstamo al 5 tenemos 15 menos 6 es igual a 9. En la siguiente posición 3-2 = 1.

2.8.5. Resta Binaria

El proceso de la resta en base 2 es similar a la resta decimal. En este sistema el préstamo de una columna anterior tiene un peso de 2, como se puede observar en el ejemplo;

Ejemplo 2.26

$$
\begin{array}{cccccc}
 & & 0 & 2 & & \\
 & 1 & \cancel{1} & 0 & 1 & \\
- & & & 1 & 0 & \\
\hline
 & 1 & 0 & 1 & 1_2 &
\end{array}
$$

Frecuentemente no es posible obtener un préstamo de una columna anterior, es necesario en este caso acudir a la próxima columna.

Ejemplo   2.27

$$
\begin{array}{cccc}
 & & 1 & \\
 & 0 & \cancel{2} & 2 \\
 & \cancel{1} & \cancel{0} & 0_2 \\
- & & & 1_2 \\
\hline
 & 0 & 1 & 1_2
\end{array}
$$

2.8.6 Dos Complemento

En una computadora digital, la resta usualmente se desarrolla por medio de sumas. Por consiguiente, no es necesario que la unidad aritmética de la computadora cuente con un circuito que reste, la ventaja de esto es la que la unidad aritmética se reduce. Reducción de los circuitos de la unidad aritmética. El método más usado para efectuar la resta por medio de sumas es el método del 2 complemento y consiste en sumar el minuendo más el dos complemento del sustraendo. El dos complemento de un número es igual al uno complemento más 1. El uno complemento se encuentra cambiando todos los "unos" por "ceros " y viceversa.

Ejemplo 2.28 Reste usando el método del 2 complemento 1101_2 menos 0110_2

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

Usando el método del 2 Complemento

a) Obtener el uno complemento de 0110_2

0 11 0 $\rightarrow$ 1 0 0 1 (uno complemento)

b) El dos complemento se obtiene sumando 1

$$\begin{array}{rl} 1\ 0\ 0\ 1 & \\ +\ \ \ \ \ \ \ 1 & \\ \hline 1\ 0\ 1\ 0 & \leftarrow \text{(dos complemento)} \end{array}$$

c) Sumar el minuendo al dos complemento

$$\begin{array}{rl} 1\ 1\ 0\ 1 & \\ +\ 1\ 0\ 1\ 0 & \\ \hline 1\ 0\ 1\ 1\ 1 & \leftarrow \text{resultado} \\ \uparrow\ \ \ \ \ \ \ \ \ & \\ \text{se desprecia}\ \ \ \ \ \ \ & \end{array}$$

f

El acarreo que resulta del bit de mayor peso se desprecia.

2.8.7 Resta Octal

En la resta octal un préstamo de una columna anterior tiene un peso de 8.

Ejemplo 2.29

$$\begin{array}{lcr} 3 & & \\ \cancel{4}\ (8) & + & 2_8 \\ -\ 1 & & 7_8 \\ \hline 2 & & 3_8 \end{array}$$

En la primer columna no es posible la reste 2 menos 7, entonces pedimos un “préstamo" a la columna anterior y ahora tenemos (8 + 2)- 7 3.

Ejemplo 2.30

$$
\begin{array}{cccc}
 & & 7 & \\
 & 3 & \cancel{8} & \\
 & \cancel{4} & 0(8) & +\ 3_8 \\
- & 1 & 7 & 4 \\
\hline
 & 2 & 0 & 7_8
\end{array}
$$

En el ejemplo anterior el préstamo se origina en la posición "2".

2.8.8 Resta Hexadecimal

Para la resta hexadecimal un préstamo tiene peso de 16

Ejemplo 2.31

$$
\begin{array}{ccc}
 & A & \\
 & \cancel{B}\ (16) & +\ 2 \\
- & 1 & 4 \\
\hline
 & 9 & (14) \\
\hline
 & 9 & E_{16}
\end{array}
$$

Cuando un resultado no excede la base substituirse por su letra equivalente.

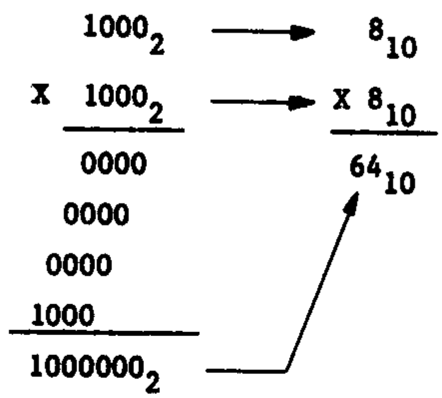
Ejemplo 2.32

$$
\begin{array}{ccccc}
 & & 15 & 15 & \\
 & 0 & \cancel{16} & \cancel{16} & 16 \\
 & \cancel{1} & \cancel{0} & \cancel{0} & \cancel{0}_{16} \\
- & & & & 1_{16} \\
\hline
 & & (15) & (15) & (15) \\
\hline
 & & F & F & F_{16}
\end{array}
$$

2.8.9 Multiplicación y División

En los puntos anteriores a este, se puede observar que el mecanismo de la suma y de la resta entre los sistemas numéricos de notación posicional es el mismo de igual forma el procedimiento para las operaciones de multiplicación y división que se usa en el sistema numérico decimal, funciona en binario, octal y hexadecimal.

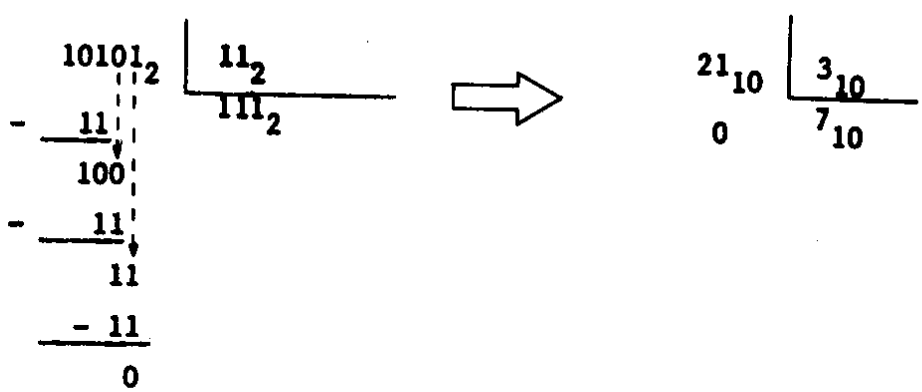
Ejemplo 2.33 Efectué la siguiente multiplicación binaria



Nótese, en el ejemplo 2.33 que el mecanismo de la multiplicación binaria es semejante a la multiplicación decimal, con la variante de que cualquier dígito del multiplicador que se multiplique con un dígito del multiplicando da como resultado solamente un cero o un uno.

Por otro lado la división binaria consiste en restar al dividendo, el divisor tantas veces come sea posible, come se muestra en el ejemplo 2.34.

Ejemplo 2.34 Efectúe la siguiente división binaria



La multiplicación o división X 2 o /2 de un número binario puede realizarse sin desarrollar operaciones aritméticas, solamente es necesario efectuar un corrimiento a la derecha para multiplicar por 2 y a la izquierda para dividir entre 2.

Ejemplo 2.35 Multiplicación Binaria X 2

$$
\begin{array}{lllllll}
1 \times 2 = & \overleftarrow{1}_2 & = & 10_2 & = & 2_{10} \\
2 \times 2 = & \overleftarrow{10}_2 & = & 100_2 & = & 4_{10} \\
4 \times 2 = & \overleftarrow{100}_2 & = & 1000_2 & = & 8_{10} \\
8 \times 2 = & \overleftarrow{1000}_2 & = & 10000_2 & = & 16_{10}
\end{array}
$$

Ejemplo 2.36 División Binaria 2

$$
\begin{array}{lllllll}
8 \div 2 = & \overrightarrow{1000}_2 & = & 100_2 & = & 4_{10} \\
4 \div 2 = & \overrightarrow{100}_2 & = & 10_2 & = & 2_{10} \\
2 \div 2 = & \overrightarrow{10}_2 & = & 1_2 & = & 1_{10}
\end{array}
$$

La multiplicación en Octal y Hexadecimal debe tomar especial cuidado, puesto que al multiplicar un dígito con otro el resultado puede ser mayor que la base, obviamente este resultado esta expresado en decimal.

Por consiguiente es necesario efectuar los ajustes pertinentes con el fin de corregir el producto.

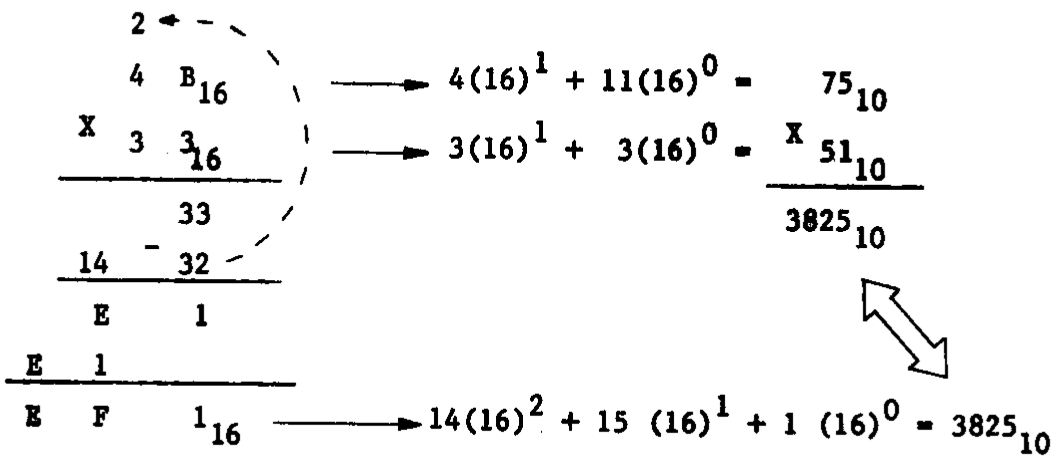
Ejemplo 2.37 Efectúe la siguiente multiplicación Octal

$$
\begin{array}{rrr}
 & 1 & \\
 & 2 & 5_8 \\
\times & 2 & 2_8 \\
\hline
 & 1 & 0 \\
 - & & 8 \\
\hline
 & 5 & 2 \\
5 & 2 & \\
\hline
5 & 7 & 2_8
\end{array}
\qquad
\begin{array}{l}
\longrightarrow 2(8)^1 + 5(8)^0 = \\
\longrightarrow 2(8)^1 + 2(8)^0 =
\end{array}
\qquad
\begin{array}{rr}
 & 21_{10} \\
\times & 18_{10} \\
\hline
 & 378_{10}
\end{array}
$$

$$\Updownarrow$$

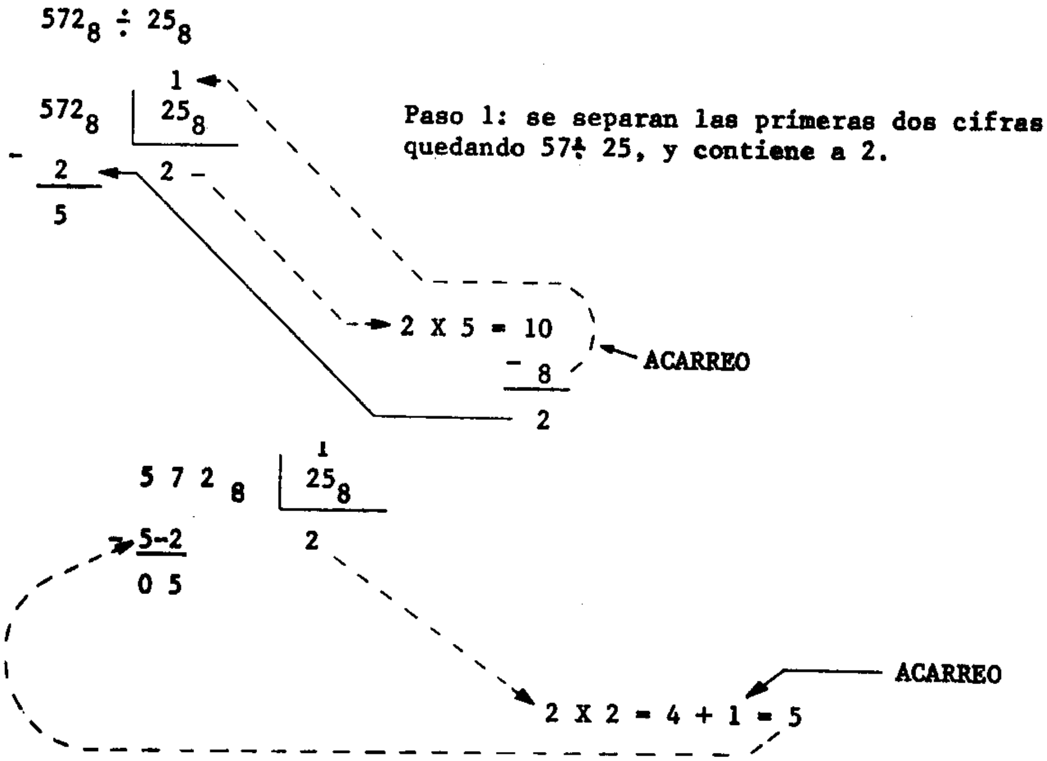
$$5\,7\,2_8 \longrightarrow 5(8)^2 + 7(8)^1 + 2(8)^0 = 378_{10}$$

Ejemplo 2.38 Efectúe la siguiente multiplicación hexadecimal

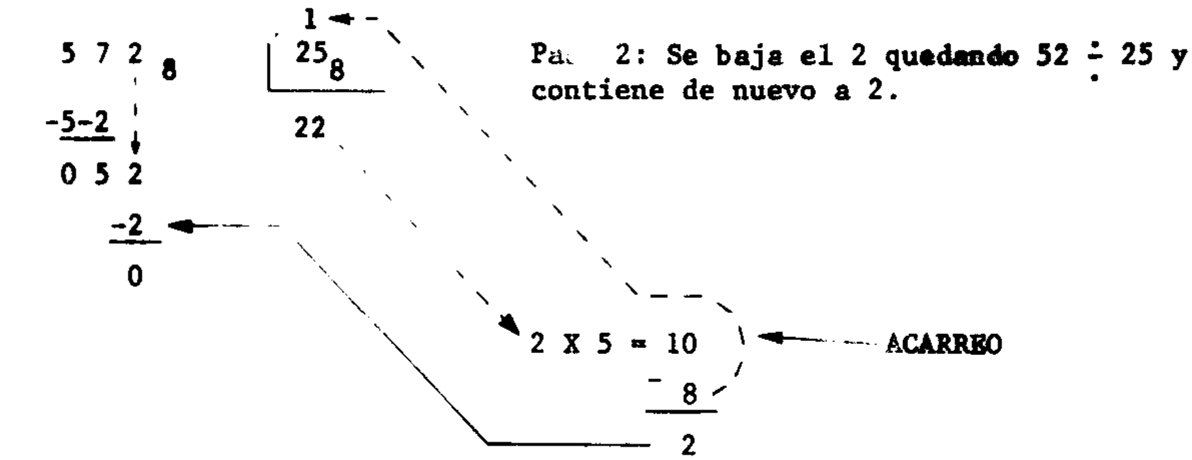


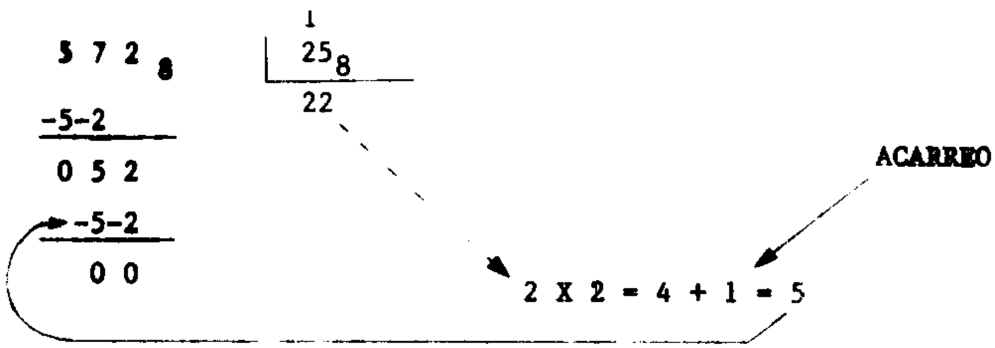
La División en cualquier sistema numérico de notación posicional, puede llevarse a cabo por medio de las operaciones básicas de suma, resta y multiplicación, usando el mismo método que en decimal. Cabe mencionar que en la división octal y hexadecimal es necesario tener precaución con el manejo de resultado mayores que la base, los cuales deberán ajustarse a cantidades validas dentro del sistema en que se esta trabajando.

Ejemplo 2.39 Desarrolle la siguiente división octal



Paso 1 se separan las primeras dos cifras quedando 57+ 25, y contiene a 2.

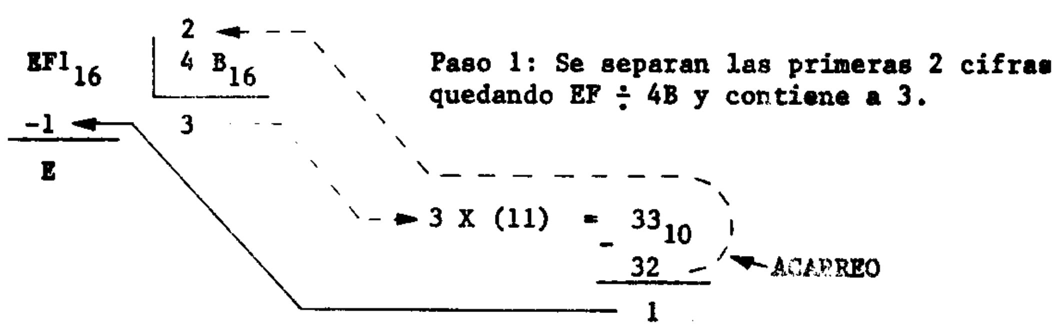


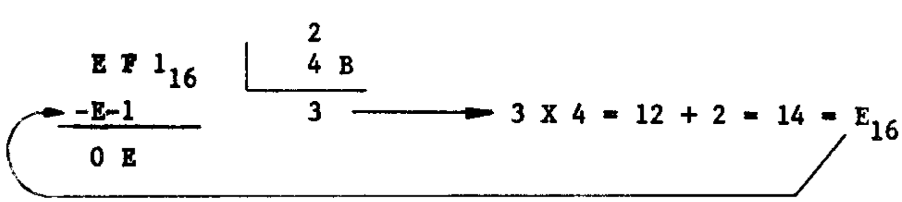


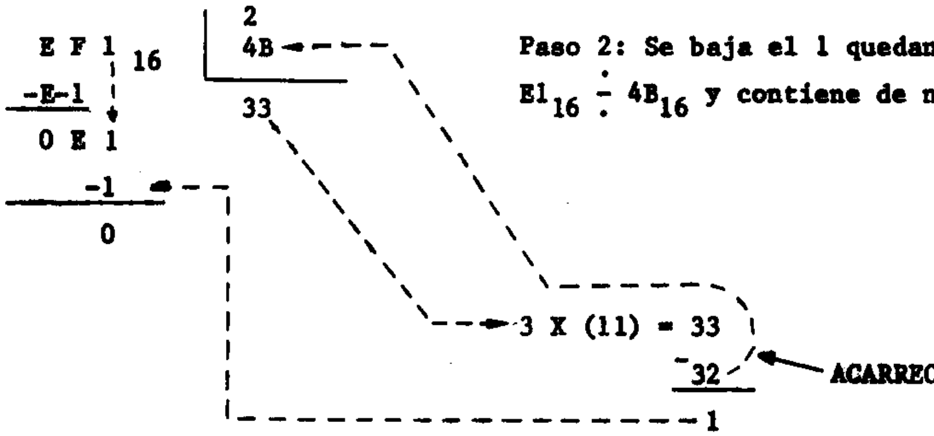
$\therefore\ 572_8 \div 25_8 = 22_8$

Ejemplo 2.40 Desarrolle la siguiente división hexadecimal

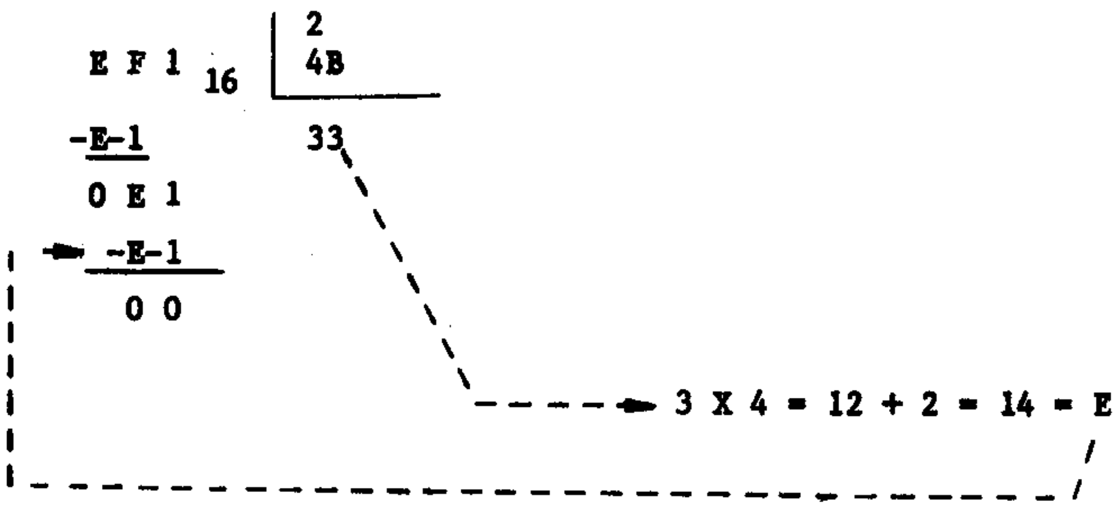
$EF1_{16} - 4B_{16}$







Paso 2: Se baja el 1 quedando $E1_{16} \div 4B_{16}$ y contiene de nuevo a 3.



$\therefore EF1_{16} \div 4B_{16} = 33_{16}$

PROBLEMAS PROPUESTOS

1.- ¿Que es Dígito?

2.- Explique que es acarreo

3.- ¿A qué se le llama "PESO" de un dígito?

4.- ¿Que se entiende por base de un Sistema Numérico?

5.- ¿Cómo es el Dígito Mayor de un Sistema de Notación Posicional con respecto a la base? Y escriba el dígito mayor de los siguientes sistemas:

SISTEMA	DIGITO MAYOR
Binario	________
Octal	________
Decimal	________
Hexadecimal	________

6.- En los siguientes ejemplos escriba cual es el dígito de mayor peso y cual es el dígito de menor peso.

EJEMPLO	MSD	LSD
842	______	______
523	______	______
176	______	______

7.- ¿Qué es un Bit?

8.- Efectué las siguientes conversiones:

$496_{10} \rightarrow N_2$	$278_{10} \rightarrow N_2$	$58_{10} \rightarrow N_8$
$525_{10} \rightarrow N_8$	$1324_{10} \rightarrow N_{16}$	$4525_{10} \rightarrow N_{16}$
$101011_{10} \rightarrow N_{10}$	$316_8 \rightarrow N_{10}$	$524_8 \rightarrow N_{10}$
$12A_{16} \rightarrow N_{10}$	$34DC_{16} \rightarrow N_{10}$	$1001110_2 \rightarrow N_8$
$110101011110_2 \rightarrow N_{16}$	$110101101_2 \rightarrow N_{16}$	$532_8 \rightarrow N_{16}$
$325_8 \rightarrow N_{16}$	$5DF_{16} \rightarrow N_8$	$A85_{16} \rightarrow N_2$
$432_8 \rightarrow N_2$	$1547_8 \rightarrow N_2$	

9.- Efectúe las siguientes operaciones: (En la resta binaria además del método tradicional efectuarlas por el método de dos complemento)

$111_2 + 011_2$

$676_8 + 420_8$

$9A6_{16} + 697_{16}$

$10110_2 + 01110_2 + 10101_2$

$01076_8 + 00350_8 + 07764_8$

$0849B_{16} + 012C5_{16} + 00D34_{16}$

$6523_8 + 7770_8 + 0546_8 + 1010_8$

$F56F_{16} + 975B_{16} + 1100_{16} + 0777_{16}$

$15236_8 + 07045_8 + 00456_8 + 00017_8$

$7FFF8_{16} + BCDE5_{16} + 0AC69_{16} + 0AAAB_{16}$

$110111_2 + 111110_2 + 110001_2 + 101110_2$

$111111_2 + 011111_2 + 000100_2 + 010010_2 + 001001_2$

$100_2 - 011_2$

$200_8 - 067_8$

$AAA_{16} - 42C_{16}$

$4006_8 - 1657_8$

$C001_{16} - 1FFF_{16}$

$11101_2 - 10011_2$

$63124_8 - 05462_8$

$6F3FE_{16} - 22DCE_{16}$

$702010_8 - 030567_8$

$F01201_{16} - ABCDEF_{16}$

$1100001_2 - 0111100_2$

$11110000000_2 - 01010101011_2$

$427_8 \times 64_8$

$CBA_{16} \times 92_{16}$

$1011_2 \times 10_2$

$123A_{16} \times 3C_{16}$

$2534_8 \times 756_8$

$10111_2 \times 101_2$

37626_8 x 405_8

46247_8 x 670_8

1010111_2 x 110_2

$4A9B8C_{16}$ x $8AD_{16}$

$8F46ED_{16}$ x BOF_{16}

1101010_2 x 101_2

$7007_8 \div 25_8$

$46707_8 \div 3_8$

$1FE58_{16} \div 1C_{16}$

$FEF10A_{16} \div A_{16}$

$3EAF67_{16} \div 2F_{16}$

$1FD376_{16} \div A2_{16}$

$2021157_8 \div 63_8$

$1010111_2 \div 11_2$

$1111111_2 \div 100_2$

$26051207_8 \div 27_8$
$11100111_2 \div 110_2$
$10101011_2 \div 1011_2$

3 ÁLGEBRA BOOLEANA

3.0 Introducción

Una vez que los circuitos implementados por medio de relevadores electromagnéticos adquirieron popularidad, fue necesario su estudio y sistematización. Eran redes formadas por interruptores y contactos de relevadores que por medio de combinaciones de circuitos abiertos y cerrados que desarrollaban funciones específicas.

Fue entonces cuando se encontró que una de las ramas de la teoría electromagnética llamada Álgebra Booleana desarrollada por el matemático ingles George Boole podía adaptarse a los circuitos de interrupción.

A diferencia del Álgebra normal, las variables booleanas toman únicamente dos valores comúnmente denominados "falso" y "verdadero", que pueden relacionarse a los dos únicos estados de los circuitos de interrupción, circuito "abierto " y "cerrado". Los símbolos 0 y 1 se usan para expresar los dos posibles valores de las variables booleanas.

Si, A =1 usualmente significa que A es verdadera

Si, A = 0 significa que A es falsa.

Regresando a los interruptores, si A = 1 significa que el interruptor asociado con A está cerrado y si A = 0 significa que el interruptor está abierto.

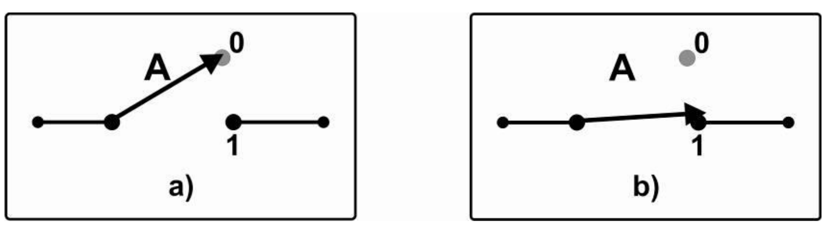


Figura 3.1 Interruptor asociado con la variable "A". a) Interruptor Abierto, A = FALSA, A =0., b) Interruptor Cerrado, A = VERDADERA, A = 1.

3.1 Operadores Lógicos

Las variables booleanas pueden manipularse por medio de operadores similares a los del álgebra normal comúnmente llamados "operadores lógicos".

3.1.1 Operador lógico "AND"

Está definido para dos o más argumentos booleanos, y puede ser relacionado con el término "CONDICIÓN", la representación más común para AND es.

$$F(AB) = A \bullet B = AB = AU\ B = A\&B$$

“F” es una función de las variables booleanas A y B. Los primeros dos símbolos son los más empleados y no indican A por B sino "A and B".

El operador AND es verdadero si y solo si todas sus variables son verdaderas. En otras palabras, es "CONDICIÓN" de que A y B sean ambas verdaderas para que F (AB) sea verdadera.

Una variable booleana puede tomar únicamente los valores de "0" o "1" LOGICOS. Entonces para una función de m variables booleanas existen 2^m posibles combinaciones de estos valores. De aquí que una forma sencilla de expresar el comportamiento de un operador lógico sea por medio de una TABLA DE VERDAD, que consiste de un listado de todas las posibles combinaciones de las variables de entrada a un operador y el valor de la operación o salida para cada combinación.

AB	F(AB) = AB
00	0
01	0
10	0
11	1

ABC	F(ABC) = A.B.C.
000	0
001	0
010	0
011	0
100	0
101	0
110	0
111	1

Figura. 3.2 Tablas de verdad para un operador AND de dos y tres variables booleanas. F(AB) es verdadera únicamente cuando todas las variables de entrada son verdaderas.

El operador AND puede relacionarse con dos o más interruptores conectados en serie con una lámpara. Esta encenderá solamente cuando ambos interruptores estén cerrados.

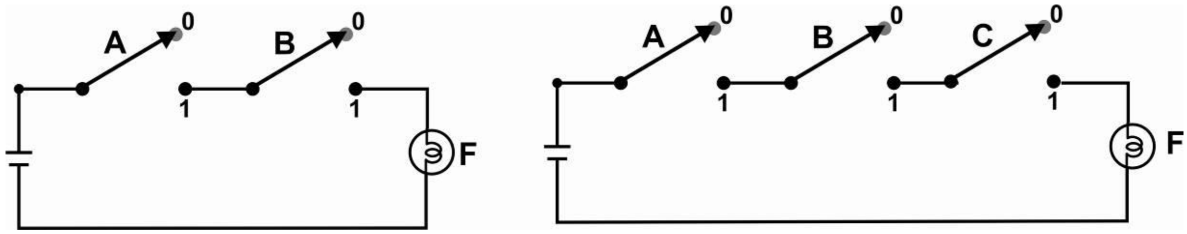


Figura 3.3 Equivalente eléctrico para un AND de 2 y 3 entradas, o variables Booleanas.

Al símbolo de un operador lógico usualmente se le llama "**COMPUERTA**", este término proviene de los antiguos sistemas de interrupción, se decía que el contacto de un relevador, era similar a una compuerta que al abrirse o cerrarse permite el paso de señales eléctricas.

Figura 3.4 Símbolos para una compuerta AND. a) Compuerta AND de 2 entradas. b) Compuerta AND de 3 entradas.

3.1.2 0perador lógico "OR"

Está definido para dos o más argumentos booleanos y puede ser relacionado con el término "ALTERNATIVA". La representación más común para el operador OR es: F(AB) = A + B= AU B = AV B

El primer símbolo es el más empleado, el signo (+) no significa más sino OR.

El operador "OR" es verdadero con solo y que una de sus variables sea verdadera. En otras palabras existe la ALTERNATIVA de que alguna de las variables sea verdadera para que el operador sea verdadero.

AB	F(AB) = A + B
00	0
01	1
10	1
11	1

ABC	F(ABC) = A + B + C
000	0
001	1
010	1
011	1
100	1
101	1
110	1
111	1

Figura 3.5 Tablas de Verdad para un operador OR, de 2 y 3 variables.

F(AB) es verdadero si A o B son verdaderas.

El operador OR puede relacionarse con dos o más interruptores conectados en paralelo con una lámpara. Esta encenderá con solo que uno de los interruptores este cerrado.

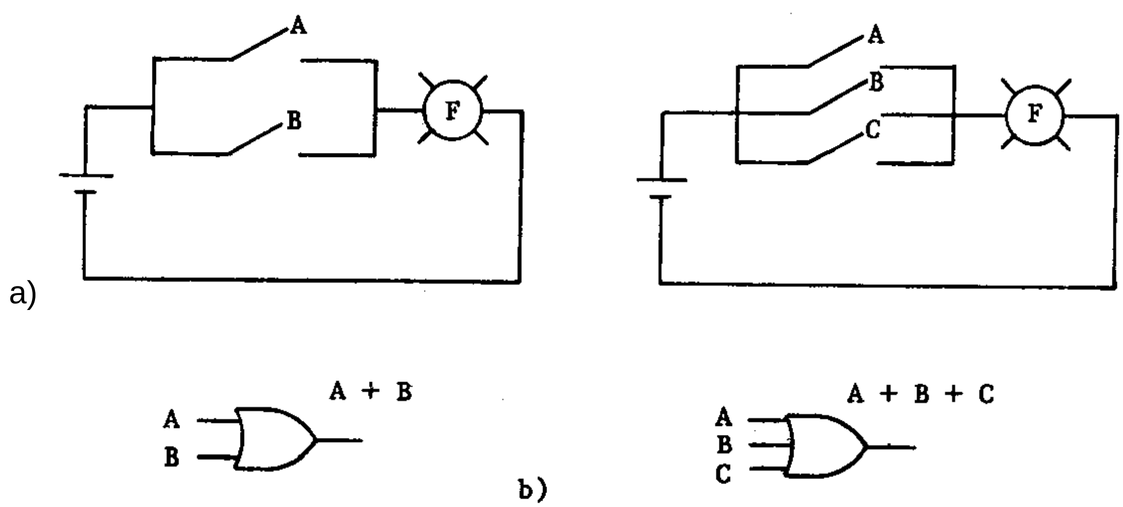


Figura 3.6 a) Equivalente eléctrico para un OR de 2 Y 3 entradas 0 variables booleanas. b) Símbolo para un OR de 2 y 3 entradas

3.1.3 Operador lógico "NOT"

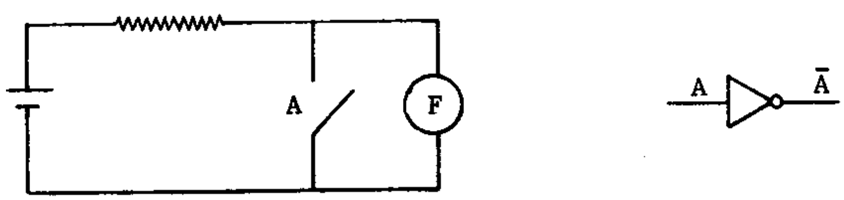
Está definido para un solo argumento booleano y su función consiste en cambiar el valor de una variable booleana por su complemento. También se le conoce como inversor o complementador. La representación más común para el operador NOT es:

$F(A) \quad = A \quad \bar{A} \quad = A^*$

La tabla de verdad para un operador NOT es la siguiente:

A	$F(A) = \bar{A}$
0	1
1	0

El operador NOT puede relacionarse con un interruptor conectado en paralelo a una lámpara que se muestra en la figura 3.7a) la lámpara encendería cuando el interruptor este abierto.



a) b)

Figura 3.7a) Equivalente eléctrico para una compuerta NOT, b) Símbolo

3.1.4. Operador lógico EX-OR (Exclusive-OR)

Está definido para dos o más argumentos booleanos. La representación del operador EX-OR es: $F(AB)=A\oplus B$

El operador EX-OR es verdadero para un número impar de variables verdaderas.

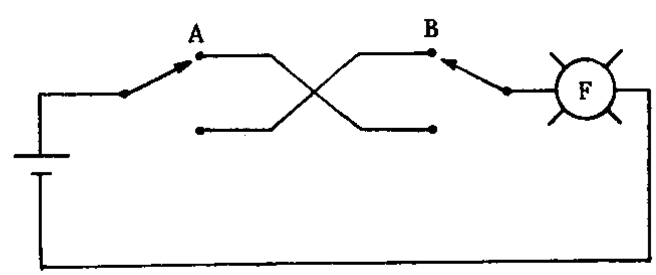
AB	$F(AB) = A \oplus B$
00	0
01	1
10	1
11	0

ABC	$F(ABC) = A \oplus B \oplus C$
000	0
001	1
010	1
011	0
100	1
101	0
110	0
111	1

Figura 3.8 Tablas de Verdad para un operador EX-OR de dos y tres variables respectivamente

La compuerta EX-OR puede relacionarse con dos interruptores de un polo, dos tiros conectados como se muestra en la figura 3.9 a)

a)



b)

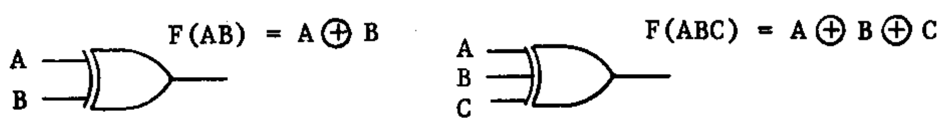


Figura 3.9 F a) Equivalente eléctrico para una compuerta EX-OR.

b) Símbolos para una compuerta EX-OR de dos y tres entradas respectivamente.

3.1.5 Operador lógico "NAND"

Está definido para una o más argumentos booleanos. El operador NAND, es la función complemento del AND, su representación es la siguiente:

$$F(AB) = \overline{A.B} = A \uparrow B \; .$$

El operador NAND es falso si y solo si sus argumentos son verdaderos.

AB	A.B	$\overline{A.B}$
00	0	1
01	0	1
10	0	1
11	1	0

Figura 3.10 Tabla de verdad para un NAND

El operador NAND puede relacionarse con un par de interruptores conectados en paralelo a una lámpara, como se muestra en la figura. 3.11.

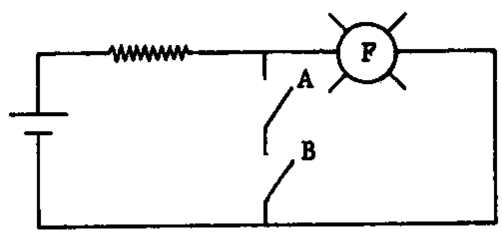


Figura 3.11 Equivalente eléctrico para un NAND

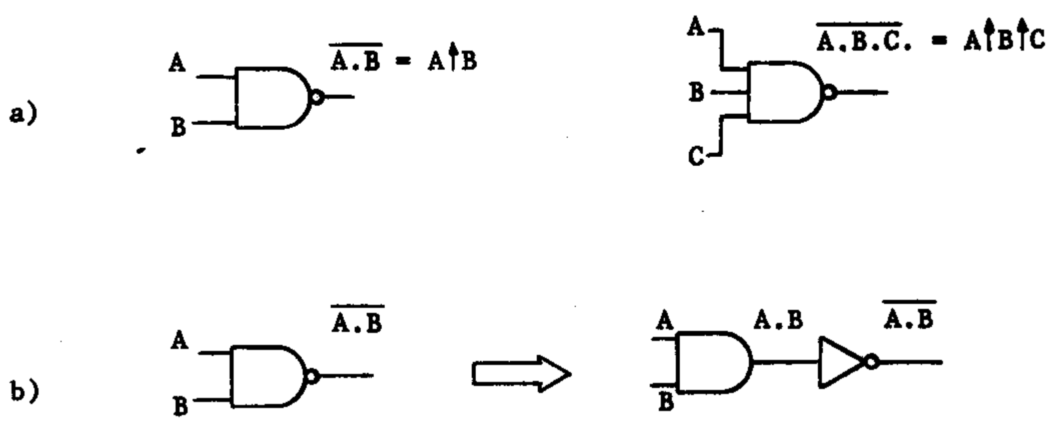


Figura 3.12 a) Símbolos para una compuerta NAND de 2 y 3 entradas, b) Un NAND es igual a un AND negado.

3.1.6 Operador lógico "NOR"

Está definido para uno o más argumentos booleanos. El operador NOR es la función complemento del OR, su representación es la siguiente:

$$F(AB) = \overline{A + B} = A \downarrow B$$

El operador NOR es verdadero si y solo si todo sus argumentos son falsos

AB	$A + B$	$\overline{A + B}$
00	0	1
01	1	0
10	1	0
11	1	0

Figura 3.13 Tabla de verdad para un NOR

El operador NOR puede relacionarse a un par, de interruptores conectados en paralelo a una lámpara, Figura3.14.

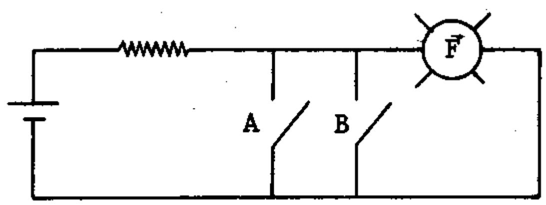


Figura 3.14 Equivalente eléctrico para un NOR

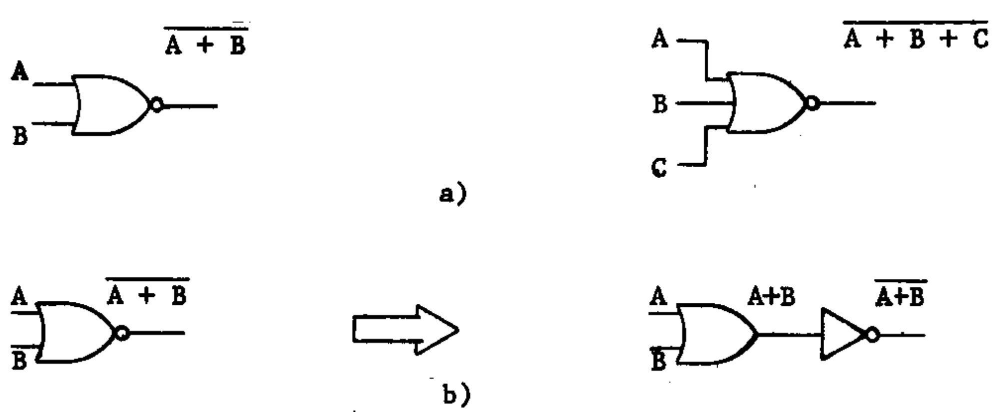


Figura 3.15 a) Símbolo para un NOR de 2 y 3 entradas, b) El NOR es igual a un OR negado.

3.1.7 Operador lógico Coincidence.

El operador lógico Concidence es la función complemento del EX-OR, también se le conoce como EX-NOR. Su representación es la siguiente:

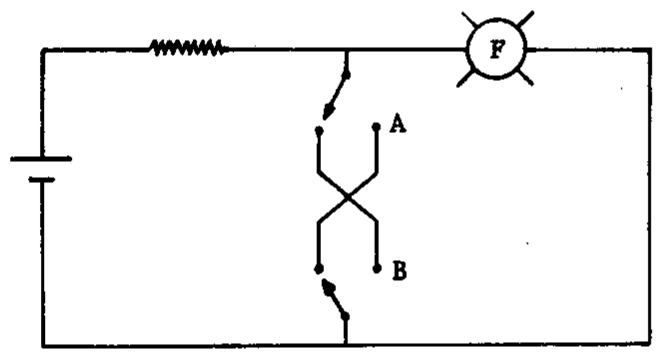
$$F(AB)= A\Theta B$$

El Coincidence es falso para un número impar de variables verdaderas.

AB	A Θ B
00	1
01	0
10	0
11	1

Figura 3.16 Tabla de Verdad para un Coincidence

a)



b)

Figura 3.17 a) Equivalente eléctrico para una compuerta Concidence. b) Símbolo para una compuerta Coincidence.

3.2 Expresiones Booleanas

La aplicación de los operadores básicos a una o más variables o constantes forma lo que se conoce como Expresiones Booleanas. Las expresiones booleanas más simples consisten en una sola variable o constante, por ejemplo, A, B, 1, etc. La formación de expresiones más complicadas se llevan a cabo combinando expresiones simples por medio de AND’S. OR'S y NOT'S, por ejemplo:

a) $A + BC$

b) $\overline{A(\bar{B}+C)}$)

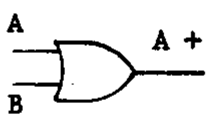
Los paréntesis se usan para indicar el orden en que se deben ejecutar

Las operaciones booleanas. Cuando no existen paréntesis, en el inciso b) debe ejecutarse primero la complementación, después el AND y por ultimo el OR.

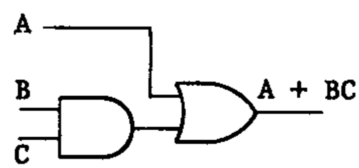
Cada expresión corresponde a un circuito de compuertas lógicas, como se muestra en el ejemplo 3.1.

Ejemplo 3.1

$A + B$



$A + BC$

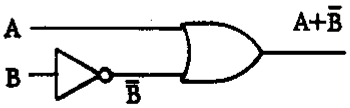


La evaluación de una expresión se hace sustituyendo los valores de 0 y 1 para cada variable. Una tabla de verdad es un método útil para este propósito, puesto que muestra todas las posibles combinaciones de los valores de las variables y su salida.

Ejemplo 3.2

a) $A + \bar{B}$

b)



c)

AB	$\bar{B}$	$A + \bar{B}$
00	1	1
01	0	0
10	1	1
11	0	1

a) Función o expresión booleana, b) Circuito, c) Tabla de verdad,

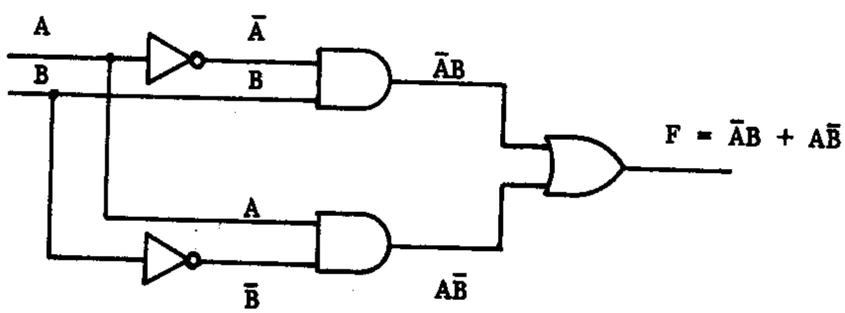
a) $F = A\bar{B} + \bar{A}B$

b) $F = A \oplus \overline{(\bar{B}+C)}$

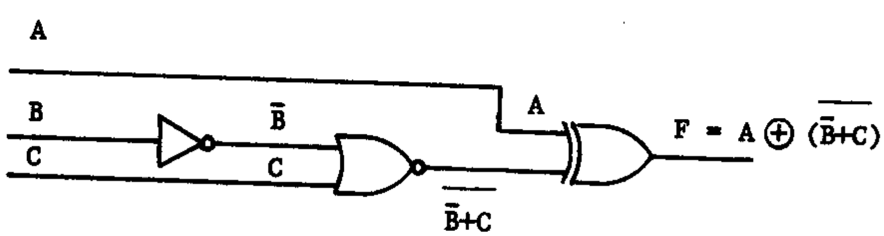
c) $F = \overline{X + B} \cdot G$

Encuentre el circuito para las siguientes funciones booleanas

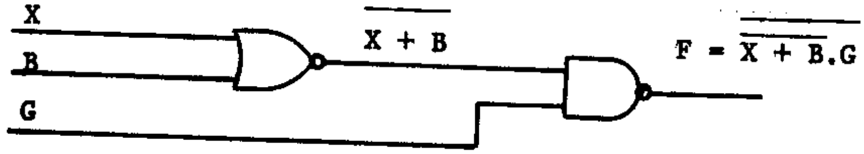
a)



b)

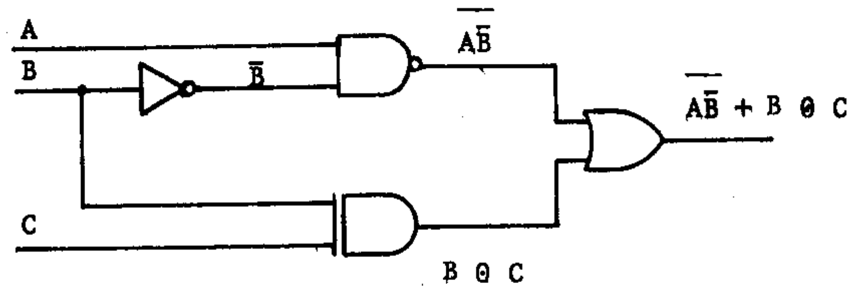


c)



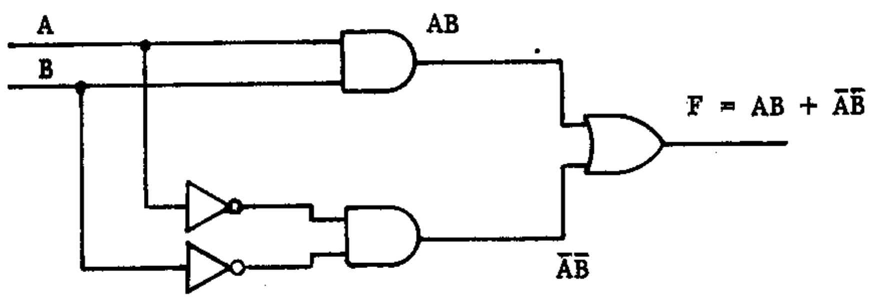
Determine las ecuaciones de los siguientes circuitos:

a)



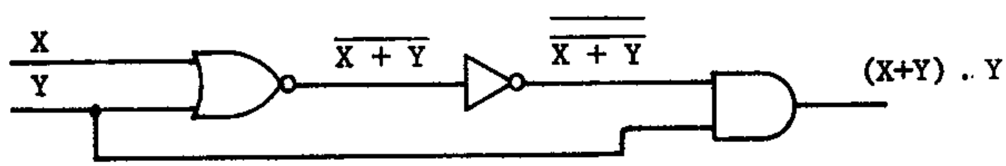
$$F(ABC) = \overline{A\bar{B}} + B \;\theta\; C$$

c)



$$F(ABC) = AB + \bar{A}\bar{B}$$

d)



$$F(XY) = (X+Y) \cdot Y$$

3.3 Propiedades fundamentales del Álgebra Booleana

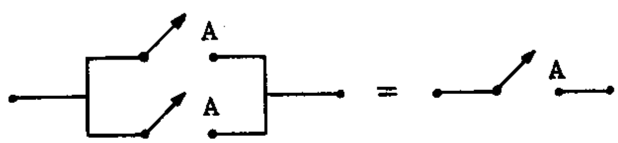
Las siguientes proposiciones son las elementales en el álgebra booleana, algunas de ellas no son correctas para el álgebra normal.

$a \cdot a = a$	$a + a = a$	
$a \cdot \bar{a} = 0$	$a + \bar{a} = 1$	$\bar{\bar{a}} = a$
$a \cdot 0 = 0$	$a + 0 = a$	
$a \cdot 1 = a$	$a + 1 = 1$	

La comprobación de estas proposiciones se ve obvia por simple inspección sin embargo pueden verificarse usando tablas de verdad o por medio de sus equivalentes eléctricos.

Ejemplo: Pruebe que a + a - a

a	a + a = a
0	0 + 0 = 0
1	1 + 1 = 1



3.3.1 Leyes fundamentales

Ley asociativa:

$$(a + b) + c = a + (b + c)$$
$$(ab)c = a\ (bc)$$

Ley conmutativa:

$$a + b = b + a$$
$$ab = ba$$

Ley distributiva:

$$a\ (b+c) = ab + ac$$
$$a+bc = (a+b)\cdot(a+c)$$

Aparentemente la última ecuación es incorrecta con respecto al, álgebra normal. Puede probarse mediante una tabla de verdad o empleando los postulados anteriormente descritos.

$$
\begin{aligned}
a + bc &= (a+b)\ (a+c) \\
&= aa + ac + ba + bc \\
&= a \ + ac + ab + bc \\
&= a \ (1 + c + b) + bc \\
&= a\ (1) + bc \\
&= a + bc
\end{aligned}
$$

3.4 Teorema de D'MORGAN

Para obtener el complemento o inverso de una expresión booleana se aplica el teorema de "D'MORGAN" en su forma más general establece que para complementar una función booleana expresada por medio de AND, OR y NOT, es necesario:

1.- Reemplazar todos los operadores AND por OR.

2.- Reemplazar todos los operadores OR por AND.

3.- Reemplazar todas las variables por su complemento.

Aplicando el teorema de D'MORGAN para dos argumentos tenemos:

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

ab	$\overline{a+b}$	$\bar{a}.\bar{b}$
00	$\overline{0+0} = 1$	$\bar{0}.\bar{0} = 1$
01	$\overline{0+1} = 0$	$\bar{0}.\bar{1} = 0$
10	$\overline{1+0} = 0$	$\bar{1}.\bar{0} = 0$
11	$\overline{1+1} = 0$	$\bar{1}.\bar{1} = 0$

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

ab	$\overline{a.b}$	$\bar{a}+\bar{b}$
00	$\overline{0.0} = 1$	$\bar{0}+\bar{0} = 1$
01	$\overline{0.1} = 1$	$\bar{0}+\bar{1} = 1$
10	$\overline{1.0} = 1$	$\bar{1}+\bar{0} = 1$
11	$\overline{1.1} = 0$	$\bar{1}+\bar{1} = 0$

Ejemplo 3.3 Complemente la siguiente función:

$$F = AC + B\bar{D}$$

$$\bar{F} = \overline{AC + B\bar{D}}$$

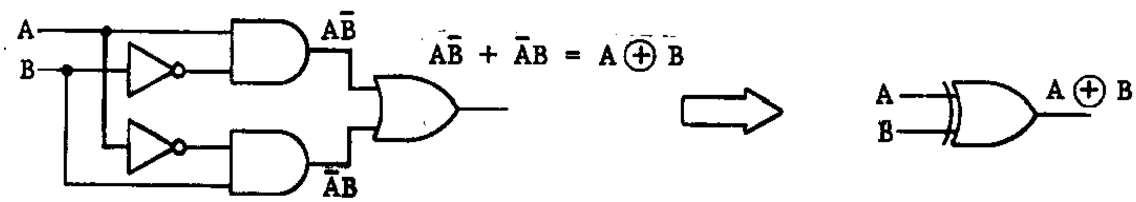
$$\bar{F} = (\bar{A} + \bar{C}) \cdot (\bar{B} + D)$$

3.5 La forma "A O N,"AND, OR, NOT

Todas las funciones booleanas pueden expresarse por medio de los operadores lógicos AND, OR y NOT, tal es el caso de los operadores EX-OR y Coincidence.

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

La expresión anterior es la forma AON para el EX-OR. El circuito de la expresión anterior es el siguiente:



Para comprobar la expresión anterior usaremos una tabla de verdad.

AB	$A\bar{B}$	$\bar{A}B$	$A\bar{B} + \bar{A}B$
00	0	0	0
01	0	1	1
10	1	0	1
11	0	0	0

variables de entrada

salida del circuito

AB	$A \oplus B$
00	0
01	1
10	1
11	0

La expresión para el COINCIDENCE se obtiene complementando, $A \oplus B$

$$\overline{A \oplus B} = \overline{A\bar{B} + \bar{A}B}$$
$$= (\bar{A} + B) \cdot (A + \bar{B})$$
$$= \bar{A}A + \bar{A}\bar{B} + BA + B\bar{B}$$
$$= 0 + \bar{A}\bar{B} + AB + 0$$
$$\overline{A \oplus B} = \bar{A}\bar{B} + AB$$
$$\therefore \quad = A \odot B = \bar{A}\bar{B} + AB$$

Forma A.O.N. para un coincidence

3.6 Expresión de Funciones Booleanas a partir de NAND y NOR

La expresión de funciones booleanas a partir de NAND'S y NOR es una alternativa a la forma AON, es decir, con un solo tipo de dispositivo lógico podemos implementar cualquier circuito.

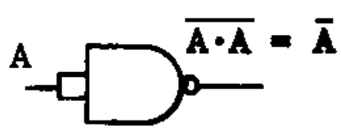
Esta propiedad es de gran utilidad en la práctica, puesto que no hay necesidad de disponer de una gran cantidad de compuertas AND, OR y NOT.

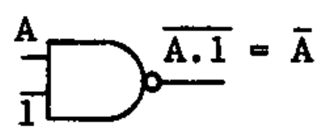
NOT a partir de NAND

El operador NAND puede hacer la función de un NOT de dos formas. La primera es efectuando la operación NAND con la misma variable.

Y la segunda es combinando la variable con un "1" lógico.

$$\overline{A \cdot 1} = \bar{A}$$



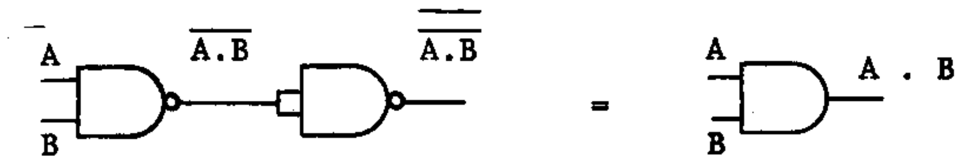


Conexiones para obtener un NOT a partir de un NAND

AND a partir de NAND

Para obtener un AND es necesario negar la salida del NAND.

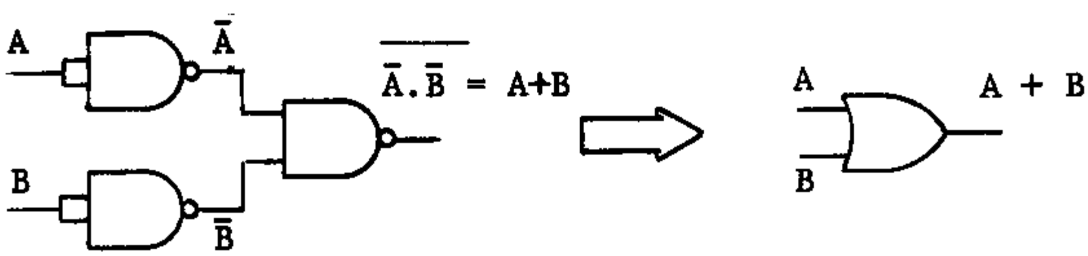
$$A.B = \overline{\overline{A.B}}$$
$$= \overline{A \uparrow B}$$



OR a partir de NAND

Para obtener un OR a partir de NAND'S as necesario cambiar el (+) OR por un (.) punto negado o NAND,

$$A + B = \overline{\overline{A + B}}$$
$$= \overline{\bar{A} \,.\, \bar{B}} = \bar{A} \uparrow \bar{B}$$



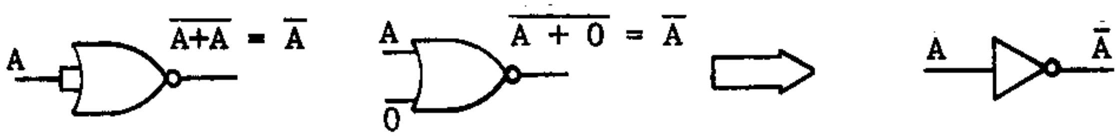
NOT a partir de NOR

Igual que el NAND el NOR puede actuar como NOT de dos formas. La primera es efectuando la operación NOR con la misma variable.

$$\overline{A + A} = \bar{A}$$

Y la segunda es combinando la variable con un "0" lógico.

$$\overline{A + 0} = \bar{A}$$

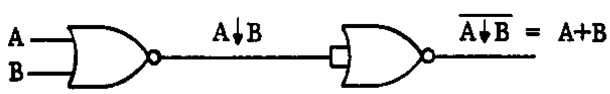


Conexiones para obtener un NOT a partir de un NOR

OR a partir de un NOR

Para obtener un OR es necesario negar la salida del NOR

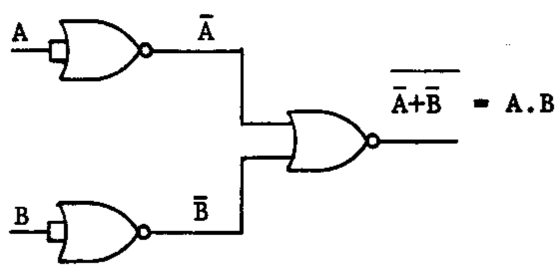
$$A + B = \overline{\overline{A + B}}$$
$$= \overline{A \downarrow B}$$



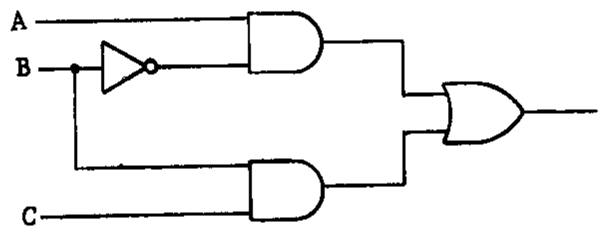
AND a partir de un NOR

Para obtener un AND a partir de un NOR es necesario cambiar el $(\cdot)$.AND por un $(\overline{+})$ OR negado.

$$A \cdot B = \overline{\overline{A \cdot B}}$$
$$= \overline{\bar{A} + \bar{B}}$$
$$= \bar{A} \downarrow \bar{B}$$



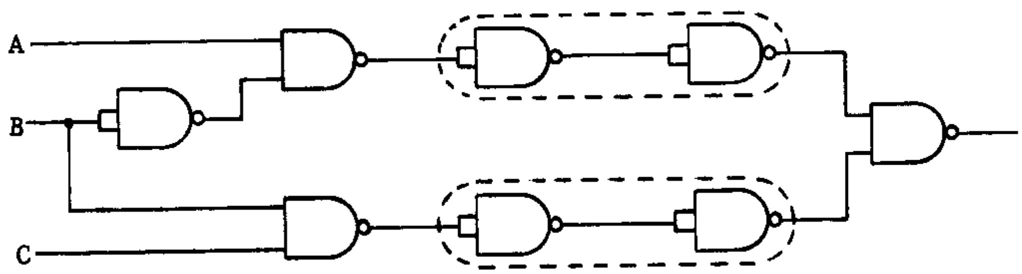
Ejemplo 3.4 Transformar el siguiente circuito implementado con compuertas AND, OR y NOT a uno que solo contenga compuertas NAND.



PROCEDIMIENTO:

1.- Reemplazar cada elemento par su equivalente en Nand's

2.- Dos negaciones seguidas deben eliminarse



Al reemplazar los elementos del circuito anterior por su equivalente en NAND'S, es necesario eliminar dos pares de NAND'S consecutivos.

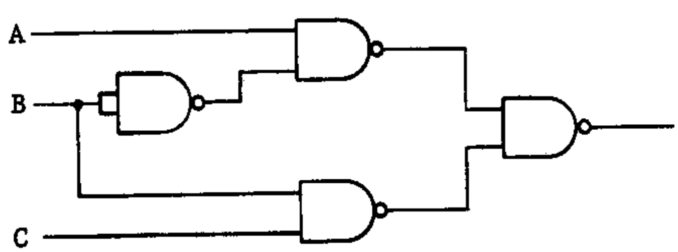


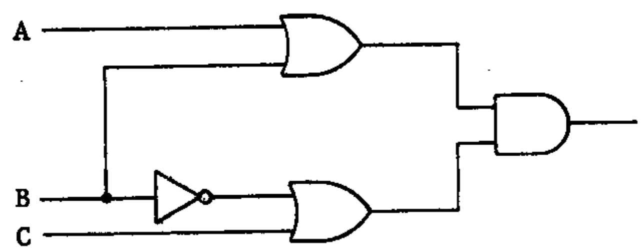
Figura 3.18 Circuito resultante

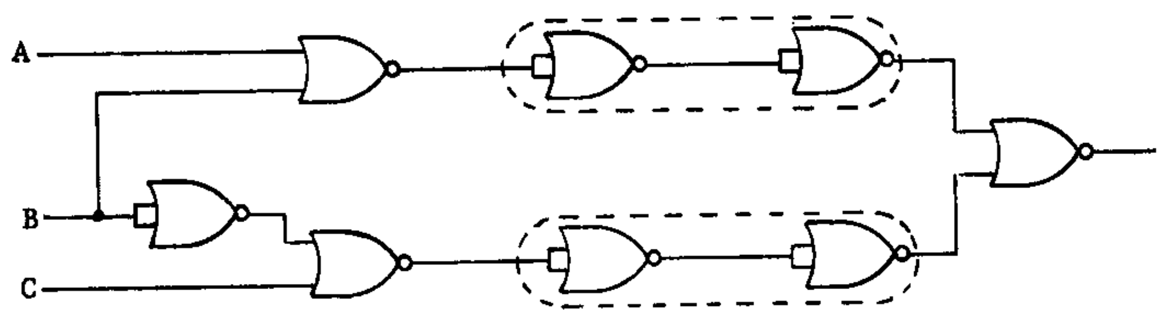
Las compuertas electrónicas se fabrican en paquetes llamados Circuitos Integrados que generalmente tienen solo compuertas del mismo tipo, por ejemplo un circuito con compuertas AND contiene solamente compuertas

AND. Para implementar el circuito equivalente anterior es necesario un solo CIRCUITO INTEGRADO, mientras que el circuito original necesita tres.

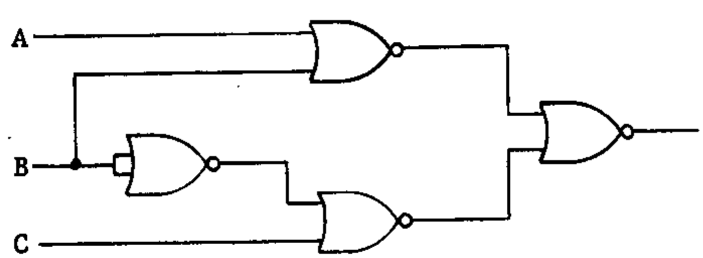
Ejemplo 3.5

Con el procedimiento del ejemplo anterior transforme el siguiente circuito a uno que solo contenga compuertas NOR.





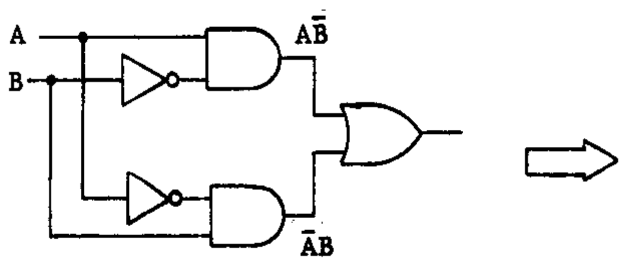
Sustitución a compuertas NOR



Circuito Equivalente

3.7 Origen de las Funciones Booleanas, Minitérminos

Como se discutió en el punto 3.2 para cada expresión booleana corresponde un circuito implementado por compuertas lógicas. Esa expresión es comúnmente llamada Función Booleana y representa el comportamiento de un circuito determinado. En el punto 3.5 podemos observar un ejemplo bastante ilustrativo. La ecuación booleana se obtiene haciendo pasar las variables a través de cada compuerta. La salida es una función de las variables de entrada, en este caso es una función de A y B, F (AB). En esta forma se puede obtener una función (ecuación) a partir de un circuito.



$$F(AB) = \bar{A}B + A\bar{B}$$
$$= A \oplus B$$

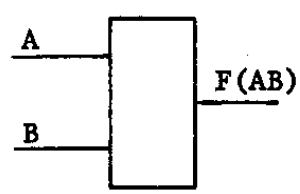
El proceso inverso, obtener un circuito a partir de una función booleana también discutido en el punto 3.2 no tiene el menor problema. Veamos ahora como obtener la función de un bloque cuyo circuito no conocemos.

Imaginemos para el caso un bloque con dos entradas y una sola salida.

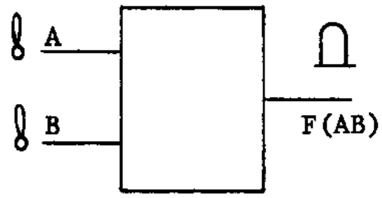
Procedimiento a seguir:

1.- Las entradas obviamente son variables booleanas

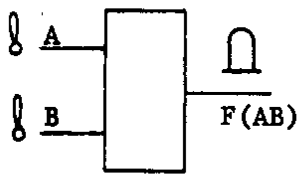
Asignémosle pues un nombre, por ejemplo A y B. La salida tendrá que llamarse F(AB), F de AB.



2.- Por medio de un par de interruptores cuyas salidas sean niveles lógicos, substituiremos las variables por unos y ceros lógicos, la función de salida será monitoreada por una lámpara, si F(AB)= 1(verdadera) la lámpara encenderá, si F(AB)=0, (falsa), la lámpara no encenderá



3. Como la función de salida es una ecuación que representa el comportamiento del bloque, obtengamos entonces su comportamiento substituyendo los valores de A y B por 1'S y 0´S, indiquemos en una tabla el valor de la salida para cada combinación.



AB	F(AB) = ?
00	0
01	0
10	0
11	1

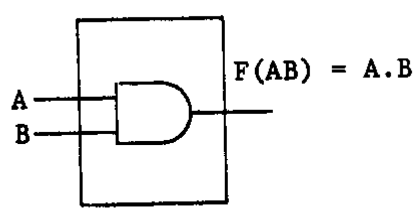
4. Supongamos que los valores que aparecen en la tabla anterior son los correspondientes al bloque. F(AB) es VERDADERA solo una vez, cuando A y B son verdaderas y es falsa F(AB), en las restantes tres combinaciones.

Por lo tanto para que F (AB) sea verdadera es CONDICIÓN de que A y B sean "ambas" verdaderas, de aquí que:

F(AB) = A . B

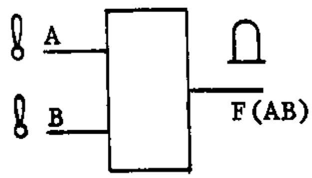
F(AB) es igual a A AND B

5. Podemos concluir que el circuito que, se encuentra en el bloque que analizamos tiene el comportamiento de una compuerta AND.



Ejemplo 3.6

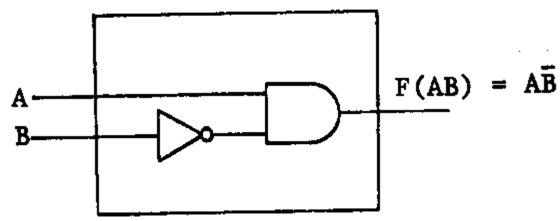
Tomemos otro bloque cuya tabla de verdad sea:



AB	F(AB) = ?
00	0
01	0
10	1
11	0

En este caso para que F(AB) sea verdadera es CONDICIÓN de que A sea verdadera y B falsa.

El circuito dentro del bloque es el siguiente:



Podemos observar que A. B para el primer ejemplo y que para, $A.\bar{B}$ el segundo, son verdaderas solo una vez, es de ir existe un solo uno para todas las combinaciones.

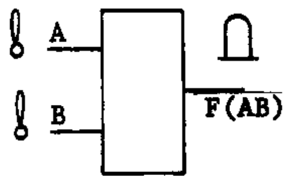
AB	F(AB) = A.B
00	0
01	0
10	0
11	1

En esta tabla el término AB tiene un número mínimo de unos en su salida, por este motivo se le conoce como Minitérmino.

El Minitérmino es un término producto que contiene todas las variables de la función ya sea en su forma normal o complementada.

Ejemplo 3.7

Tomemos un tercer bloque, cuya tabla de verdad sea la siguiente:



AB	F(AB) = ?
00	0
01	1
10	1
11	0

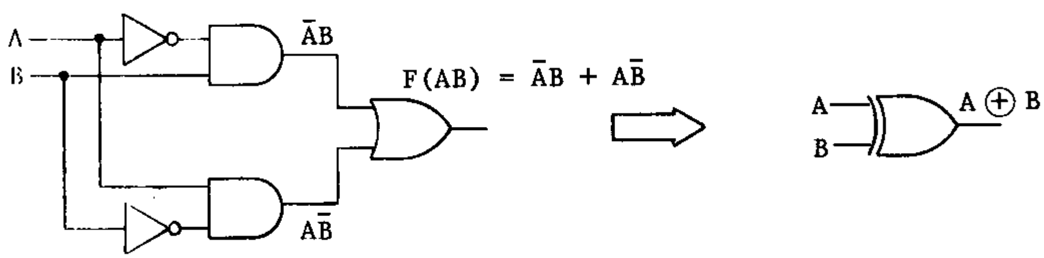
En este caso F(AB) es verdadera en dos ocasiones, es decir existe la ALTERNATIVA de que F(AB) sea dos veces verdadera, una cuando se presente la CONDICIÓN de A y B, y la otra cuando se presente la condición de A y B*

De aquí que:

$$F(AB) = \bar{A}.B + A.\bar{B}$$

F(AB) es igual a $\underline{\bar{A}\text{ AND }B}$, OR, $\underline{A\text{ AND }\bar{B}}$

El circuito dentro del bloque es un EX-OR



En este ejemplo existen dos minitéminos, el $\bar{A}B$ y el $A\bar{B}$ y la función

De F(AB) es igual a la suma(OR) de esto dos minitéminos.

$$F = \Sigma \text{ de los minitéminos}$$

Un sistema de n variable de entrada tendrá 2^n diferentes minitéminos.

Los minitérminos pueden expresarse por medio de una "m" minúscula con un subíndice decimal correspondiente al número binario que represento el minitérmino.

Ejemplo 3.8 Enuncie los minitéminos para un sistema de 2 variables.

AB	MINITERMINOS
00	$\bar{A}\bar{B} = m_o$
01	$\bar{A}B = m_1$
10	$A\bar{B} = m_2$
11	$AB = m_3$

Puesto que una función es igual a la sumatoria de sus minitéminos tenemos

AB	F(AB)
00	1
01	1
10	1
11	0

$$F(AB) = \bar{A}\bar{B} + \bar{A}B + A\bar{B}$$

$$F(AB) = m_o + m_1 + m_2$$

$$F(AB) = \Sigma\ m_0,\ m_1,\ m_2$$

$$F(AB) = \Sigma\ 0,\ 1,\ 2 \quad \text{FORMA CANONICA}$$

A esta forma minimizada se le conoce como SUMATORIA DE PRODUCTOS (SOP) o forma canónica.

Ejemplo 3.9

Obtenga la forma canónica de la expresión booleana para el sistema cuya tabla de verdad se muestra a continuación.

ABC	F(ABC)
000	1
001	1
010	0
011	0
100	0
101	0
110	0
111	1

$$F(ABC) = \bar{A}\bar{B}\bar{C} + \bar{A}\bar{B}C + ABC$$

$$F(ABC) = m_0 + m_1 + m_7$$

$$F(ABC) = \Sigma\ 0,\ 1,\ 7$$

Ejemplo 3.10 Obtenga la tabla de verdad de la siguiente función booleana expresada en su forma canónica.

$$F(ABC) = \Sigma\ 1,\ 4,\ 5,\ 7$$

ABC	F(ABC) = Σ 1, 4, 5, 7
000	0
001	1
010	0
011	0
100	1
101	1
110	0
111	1

3.8 F negada como alternativa, Maxitérminos

Para obtener una expresión booleana a partir de una tabla de verdad se hace use de la F AFIRMADA, o VERDADERA sin embargo $\bar{F}$ o (FALSA), puede ser una alternativa muy útil, sobre todo cuando se tienen pocos "0" ceros en la función.

Ejemplo 3.11

ABC	F(ABC)
000	0
001	0
010	1
011	1
100	1
101	1
110	1
111	1

$$\bar{F}\ (ABC) = \bar{A}\bar{B}\bar{C} + \bar{A}\bar{B}C$$

$$\bar{F}\ (ABC) = \Sigma\ 0,\ 1$$

Pero realmente no nos interesa F (negada) sino F (afirmada).Aplicando el teorema de D'Morgan tenemos:

$$F\ (ABC) = \overline{\bar{A}\bar{B}\bar{C} + \bar{A}\bar{B}C}$$

$$F\ (ABC) = (A + B + C)\ .\ (A + B + \bar{C})$$

A esta forma se le conoce como Producto de Sumatorias (POS) y es una

Alternativa al (SOP) sumatoria de productos. A los términos de la forma (A+B+C) se les llama maxitérminos y al igual que los minitérminos deben contener todas las variables de la función ya sea en su forma normal o complementada.

El nombre de maxitérminos surge de la tabulación de un solo maxitérminos.

ABC	(A+B+C)
000	0
001	1
010	1
011	1
100	1
101	1
110	1
111	1

La salida contiene únicamente un "0" CERO, o sea un número máximo de unos. La expresión del ejemplo 3.11 para maxitérminos queda:

$$F(ABC) = (A+B+C) \cdot (A+B+C)$$

$$F(ABC) = M_7 \cdot M_6$$

$$F(ABC) = \Pi\ 6,\ 7$$

A esta forma se le conoce también como forma canónica conjuntiva.

Algunos autores no coinciden con nombrar a los maxitérminos en esta forma, el término (A+B+C) lo toman como (000) Mo en lugar de M_7 (111).

3.9 Las ocho Formas Estándar

En los puntos 3.7 y 3.8 se vio como una expresión booleana que representa el comportamiento de un bloque, puede expresarse por medio de la sumatoria de sus minitérminos. También llamada forma AND/OR (debido a que las variables pasan primero a través de compuertas AND y después a una compuerta OR), o también por medio del producto de sus maxitérminos llamado forma OR/AND.

En el punto 3.6 se discutió el método para transformar un circuito a compuertas NAND, forma NAND/NAND y a compuertas NOR o forma NOR/NOR.

Con el propósito de disponer de una mayor versatilidad a la hora de implementar un determinado circuito por medio de compuertas, podemos combinar los 4 operadores, AND, OR, NAND y NOR, con lo cual logramos 16 posibles combinaciones, sin embargo solo se usan 8.

1.- AND/OR

2.- NAND/NAND

3.- OR/NAND

4.- NOR/OR

5.- AND/NOR

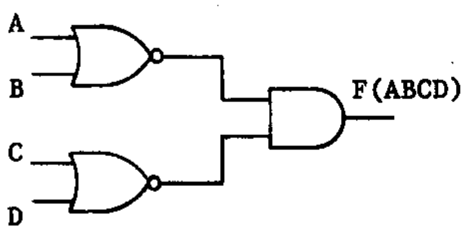
6.- NAND/AND

7.- OR/AND

8.- NOR/NOR

Esto es debido a que las ocho restantes no configuran una función de acuerdo a la sumatoria de productos o al producto de sumatorias.

Ejemplo 3.12



$$F(ABCD) = \overline{A+B} \cdot \overline{C+D}$$
$$= \bar{A}.\bar{B}.\bar{C}.\bar{D}$$

Forma NOR/AND

Considerando que se dispone de las variables y sus complementos, podemos obtener las ocho formas a partir de F y $\bar{\bar{F}}$. En dos grupos.

Ejemplo 3.13 Desarrolle las ocho formas estándar para la función definida por la siguiente tabla de verdad.

AB	F(AB)
00	1
01	0
10	0
11	1

$$F(AB) = \bar{A}\bar{B} + AB$$

$$\bar{F}(AB) = \bar{A}B + A\bar{B}$$

1. A PARTIR DE F, GRUPO AND/OR

$F(AB) = \bar{A}\bar{B} + AB$ FORMA (AND/OR)

$= \overline{\overline{\bar{A}\bar{B} + AB}}$

$= \overline{\overline{\bar{A}\bar{B}} \cdot \overline{AB}} = (\bar{A} \uparrow \bar{B}) \uparrow (A \uparrow B)$ FORMA (NAND/NAND)

$= \overline{(A+B) \cdot (\bar{A}+\bar{B})} = (A+B) \uparrow (\bar{A}+\bar{B})$ FORMA (OR/NAND)

$= \overline{(A+B)} + \overline{(\bar{A}+\bar{B})} = (A \downarrow B) + (\bar{A} \downarrow \bar{B})$ FORMA (NOR/OR)

2. A PARTIR DE $\bar{F}$, GRUPO OR/AND

$\bar{F}(AB) = A\bar{B} + \bar{A}B$

$F(AB) = \overline{\bar{A}B + A\bar{B}} = (\bar{A}B) \downarrow (A\bar{B})$ FORMA (AND/NOR)

$= \overline{(\bar{A}B)} . \overline{(A\bar{B})} = (\bar{A} \uparrow B) . (A \uparrow \bar{B})$ FORMA (NAND/AND)

$= (A+\bar{B}) \cdot (\bar{A}+B)$ FORMA (OR/AND)

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

$= \overline{\overline{(A+\bar{B})} + \overline{(\bar{A}+B)}} = (A \downarrow \bar{B}) \downarrow (\bar{A} \downarrow B)$ FORMA (NOR/NOR)

Para obtener las tres formas restantes, estando en el grupo AND/OR, o

El grupo OR/AND, Basta con aplicar sucesivamente el teorema de D'MORGAN, como se puede observar en los dos casos anteriores.

En el ejemplo siguiente se muestra la forma de cambiar de un grupo a otro.

Ejemplo 3.14

Convertir, $F = (A + B)(C + D)$ de la forma OR/AND a la forma AND/OR.

$F = (A+B)(C+D)$ DESARROLLANDO EL PRODUCTO

$F = AC + AD + BC + BD$ FORMA (AND/OR)

Ejemplo 3.15

Convertir $F = A\bar{B} + \bar{A}B$ de la forma AND/OR a la forma OR/AND.

$F = A\bar{B} + \bar{A}B$

$= (A\bar{B}) + \bar{A}B$ APLICANDO LA LEY DISTRIBUTIVA

$= (A\bar{B} + \bar{A})(A\bar{B} + B)$ DE NUEVO LA LEY DISTRIBUTIVA

$= (A + \bar{A})(\bar{B} + \bar{A})(A+B)(\bar{B}+B)$ $a + \bar{a} = 1$

$F = (\bar{B} + \bar{A})(A + B)$ FORMA (OR/AND)

PROBLEMAS PROPUESTOS

1.-¿Cuál es la diferencia entre el Álgebra Normal y el Álgebra Booleana?

2.-¿Cuándo es verdadero el resultado de una operación AND?

3.-¿Cuantas combinaciones de entrada puede tener una función si n = al número de variables de entrada?

4.-¿Qué es una compuerta?

5.-¿Cuando es verdadero el resultado de una operación OR?

6.-¿Cuál es la función de un inversor? Y escriba su símbolo.

7.-¿Cuando se cumple una función?

a) EX-OR b) NAND c) NOR

8.- Explique el funcionamiento de un operador Concidence.

9.-¿Como se complementa una función por medio del Teorema de D'MORGAN?

10.-¿A que se le llama minitérmino y cuantos minitéminos tiene una función de “n“variables de entrada?

11.-¿A que se le llama maxitérminos?

12.- Encontrar el circuito de las siguientes ecuaciones:

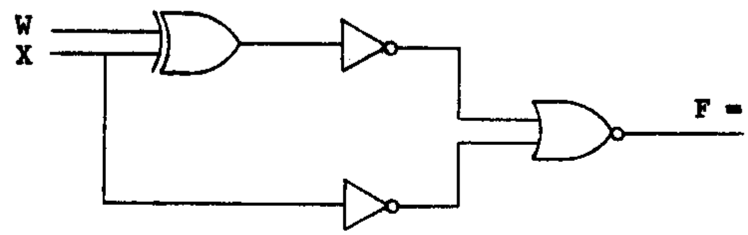
a) $F = X \oplus (y \odot Z)$

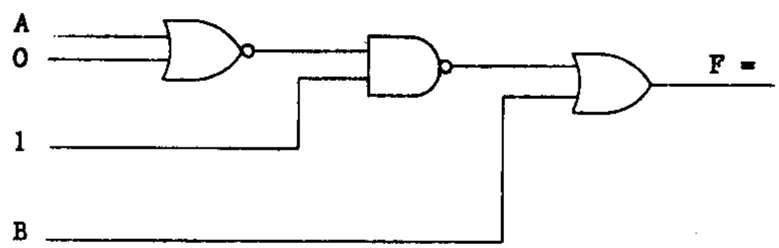
b) $F = \overline{\overline{\overline{W} + (X} + \overline{Y})}$

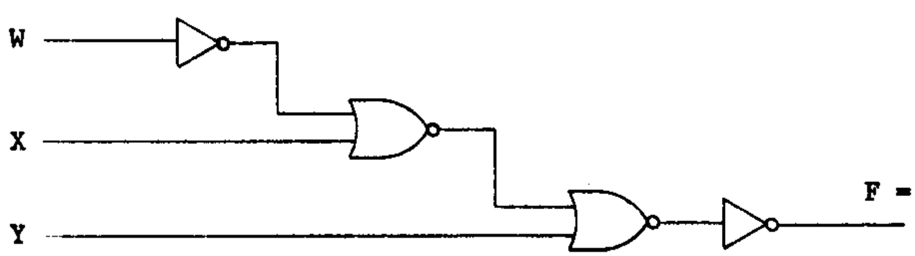
c) $F = (A.B) + (C+\overline{B}) \cdot \overline{(\overline{A}.C)} + \overline{(B+\overline{C})}$

d) $F = \overline{(\overline{PE} + \overline{PE}) \cdot (\overline{P} + \overline{A}) \oplus (E \odot A)}$

13.- Encontrar las ecuaciones de los siguientes circuitos:







14.-Implementar un circuito EX-OR y un Concidence con compuertas

a) NAND b) NOR

15.-Indique cual de las funciones están expresada en minitérminos.

$$F(ABC) = AC + \bar{A}B + C\bar{B} + ABC$$

$$F(ABC) = B + A\bar{B} + \overline{BC} + \bar{A}BC$$

$$F(ABC) = \bar{A}\bar{B}\bar{C} + A\bar{B}\bar{C} + BA\bar{C} + ABC$$

16.-a) Representar la Tabla de Verdad de las siguientes funciones:

$$F_1 = AB + A\bar{B}C + \bar{A}\bar{B}$$

$$F_2 = ABC\bar{D} + AB\bar{C} + \bar{A}BD$$

b) Hallar la forma canónica de suma de productos y producto de sumas de las dos funciones del inciso a).

17.-Dada la función F (ABCD) representada mediante la forma canónica de Suma de Productos.

F (ABCD)=Σm(0, 1, 2, 3, 12, 15)

a) Representar la tabla de verdad de esta función

b) Obtener la forma canónica de Producto de Sumas

c) Obtener las dos formas canónicas algébricas de esta función

18.-La función F(ABCD) cumple la siguiente tabla de verdad.

D	C	B	A	F
0	0	0	0	1
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	1
0	1	0	1	0
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	1
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

a) Obtener las ecuaciones booleanas de suma de productos y productos de sumas

b) Obtener las formas canónicas de minitérminos y maxitérminos

19.- Una función de tres variables F(ABC) ha de tomar el valor cero cuando la variable B se encuentre en estado uno y la variable A no este en estado uno

a) Realizar la tabla de verdad de esta función

b) Obtener las formas canónicas de suma de productos y producto de sumas

4 CÓDIGOS Y REPRESENTACIÓN DE INFORMACIÓN

4.0 Introducción

En el capítulo1 vimos como la información y la cantidad se pueden representar por medio de Unos y Ceros.

Conforme aumenta la complejidad de la información y de los datos se hace necesario el uso de Códigos que faciliten su representación.

El término Código se usa aquí para designar a un conjunto de símbolos o combinaciones de Unos y Ceros que sirven para representar información numérica o alfabética.

Los sistemas digitales generalmente representan la información numérica y efectúan sus operaciones internas en Código Binario. Sin embargo, para poder entablar protocolos que interactúen con el mundo exterior se recurre al uso de otros códigos.

En la figura 4.0 se indican los códigos más comunes empleados en la comunicación de un sistema digital con el mundo exterior.

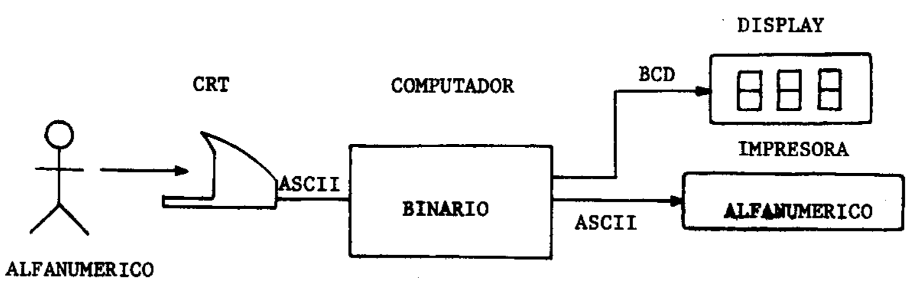


Figura 4.0 Comunicación de un sistema digital con el mundo exterior

Antes de continuar daremos las definiciones de algunos términos que usaremos en este capítulo.

BIT: Contracción de BINARY-DIGIT - dígito binario.

BYTE: Grupo de 8 bits.

CARÁCTER: Cualquier letra, número o símbolo que un computador pueda entender, almacenar o procesar.

WORD PALABRA: Grupo de bits utilizados para representar una información. No existe restricción para la cantidad de bits que forman una palabra.

4.1 Códigos Pesados

Cuando no es posible usar el código binario para la representación de una cantidad, se utilizan los llamados Códigos Pesados. Se dice que un código es Pesado cuando en correspondencia con la posición de cada bit, en una palabra, existen valores numéricos, que tienen la siguiente propiedad:

La suma de los productos de cada bit por su correspondiente valor de posición w, es igual al valor equivalente de la palabra. Esto puede representarse mediante la siguiente expresión:

$$N = \sum_{i=1}^{n} w_i a_i + C \; . \; . \; . \; . \; . \; . \qquad (4.0)$$

Donde N es la cantidad

n - Número de bits

w_i=Peso de cada bit

a_i= coeficientes

C = Base constante del código

EJEMPLO 4.0

Determine si el siguiente código es un código pesado.

ABCD	CANTIDAD
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	10
1011	11
1100	12
1101	13
1110	14
1111	15

El primer paso es encontrar los valores de w y c para el código y aplicar la ecuación (4.0) a cada combinación. Si la ecuación es válida para todas las combinaciones el código es un código pesado.

De la primera combinación podemos determinar el valor de C aplicando la ecuación (4.0)

$$N = \sum_{i=1}^{n} w_i a_i + C$$

Los valores de

$$0.w_A + 0.w_B + 0.w_C + 0.w_D + C = 0$$

$$\therefore C = 0$$

Los pesos w_A, w_B se pueden determinar de las siguientes combinaciones

DE LA SEGUNDA COMBINACION

$$0.w_A + 0.w_B + 0.w_C + 1.w_D + 0 = 1$$

$$\therefore W_D = 1$$

DE LA NOVENA COMBINACION

$$1.W_A + 0(4) + 0(2) + 0(1) + 0 = 8$$

$$\therefore W_A = 8$$

DE LA TERCERA COMBINACION

$$0.w_A + 0.w_B + 1.w_C + 0.(1) + 0 = 2$$

$$\therefore w_C = 2$$

DE LA QUINTA COMBINACION

$$0.w_A + 1.w_B + 0.(2) + 0(1) + 0 = 4$$

$$\therefore W_B = 4$$

Los pesos de este código son 8, 4, 2,1 el siguiente paso es aplicar estos valores en todas las demás combinaciones, por ejemplo;

ABCD

$$1100 = 1(8) + 1(4) + 0(2) + 0(1) = 12.$$

$$1111 = 1(8) + 1(4) + 1(2) + 1(1) = 15.$$

En este caso todos los códigos coinciden, podemos decir que se trata de un código pesado.

Ejemplo 4.1

Determine si el siguiente código es un código pesado.

ABCD	CANTIDAD
0000	0
0001	1
0010	2
0011	3
0100	4
1011	5
1100	6
1101	7
1110	8
1111	9

Paso 1

Determinar w_A, w_B, w_C, w_D y C.

$0.w_A + 0.w_B + 0.w_C + 0.w_D + C = 0$

$\therefore C = 0$

$0.w_A + 0.w_B + 0.w_C + 1(w_D) + 0 = 1$

$\therefore w_D = 1$

$0.w_A + 0.w_B + 1\ (w_C) + 0(1) + 0 = 2$

$\therefore w_C = 2$

$0.w_A + 1.(w_B) + 0(2) + 0(1) + 0 = 4$

$\therefore w_B = 4$

$1(w_A) + 0(4) + 1(2) + 1(1) + 0 = 5$

$\therefore w_A = 2$

$C = 0$

$w_A = 2$

$w_B = 4$

$w_C = 2$

$w_D = 1$

Aplicamos los valores de los pesos 2421 en otras combinaciones.

ABCD

1110 = $1(2) + 1(4) + 1(2) + 0(1) = 8$

1111 = $1(2) + 1(4) + 1(2) + 1(1) = 9$

El código 2421 es un código pesado

4.2 Códigos numéricos más usados

En la siguiente tabla se listan algunos de los códigos numéricos 4 de- bits más utilizados. Figura4.1

VALOR	BCD 8421	2,4,2,1	EXCESO 3	GRAY	BINARIO NATURAL 8421
0	0000	0000	0011	0000	0000
1	0001	0001	0100	0001	0001
2	0010	0010	0101	0011	0010
3	0011	0011	0110	0010	0011
4	0100	0100	0111	0110	0100
5	0101	1011	1000	0111	0101
6	0110	1100	1001	0101	0110
7	0111	1101	1010	0100	0111
8	1000	1110	1011	1100	1000
9	1001	1111	1100	1101	1001
10				1111	1010
11				1110	1011
12				1010	1100
13				1011	1101
14				1001	1110
15				1000	1111

Figura 4.1 Códigos numéricos más usados.

Código BCD

El código BCD cuyas siglas tienen su origen del nombre en inglés (Binary, Coded, Decimal) Decimal Codificado en Binario, es precisamente eso, un número decimal del 0 al 9 representado en 4 bits. Los números del 10-al 15 no se incluyen este código.

Es importante notar que un número codificado en BCD no es lo mismo que un número codificado en binario natural como se puede observar en la figura 4.1.

Para expresar un número de 2 dígitos decimales en BCD es necesario usar 2 décadas de BCD como se muestra en el ejemplo 4.2.

EJEMPLO 4.2

Represente en BCD el número 10_{10}.

El código BCD se usa en dispositivos digitales en donde los datos de entrada se generan en un teclado decimal y las salidas se muestran en una pantalla numérica. Por ejemplo en calculadoras digitales, relojes, multímetros, contadores de frecuencia, etc.

Las computadoras digitales modernas no procesan en BCD por dos motivos: El primero es que para representar un número en BCD se requieren más bits que un número representado en binario natural. Y el segundo motivo es que las operaciones aritméticas son más complicadas que en binario. Imaginemos una suma de 0110 + 0111, 6 + 7.

$$\begin{array}{r} 0110 \\ +\ 0111 \\ \hline 1101 \end{array} \quad \longrightarrow \quad \begin{array}{r} 6 \\ +\ 7 \\ \hline 13 \end{array}$$

El número 1101 no existe en BCD, por lo tanto es necesario una operación extra para corregir el resultado, un método simple es sumarle 6 0110_2 que es el número de combinaciones que no existen en BCD.

Entonces:

$$\begin{array}{rl} 0110 & \\ +\ 0111 & \\ \hline 1101 & \longrightarrow \text{SUMA BINARIA} \\ +\ 0110 & \\ \hline 1\ \ 0011 & \longrightarrow \text{CORRECCION A BCD} \end{array}$$

CÓDIGO 2421

El código 2421 es un código BCD que tiene un paso diferente al usual.

En vez de que la posición del bit de mayor peso MSB tenga un peso de 8, como sucede en el BCD 8421, tiene un peso de 2.

EXCESO-3

Es otro código BCD común, a menudo se abrevia como XS3. Este código representa a un número decimal en 4 bits, solo que se le añade 3 a cada dígito decimal antes de efectuar la conversión, por ejemplo el cero se encodifica en EXCESO-3 como 0011. Este código tiene propiedades aritméticas útiles, para encontrar el 9 complemento de un número solo se cambian los unos por ceros y viceversa. El método del 9 complemento sirve para hacer restas base 10 y es semejante al método del 2 complemento. En la figura 4.1 aparece el código XS3.

4.3 Códigos no pesados-código GRAY

En la tabla de la figura 4.1 aparece el código GRAY. En este código existe solo un cambio de un bit entre dos números sucesivos. Los códigos que tienen esta característica generalmente son Códigos no Pesados y su aplicación se extiende en los campos de la instrumentación, transductores, convertidores analógica/digital, encodificadores de desplazamiento lineal y angular, etc.

En la figura 4.2 se muestra parcialmente el disco de un encodificador de posición angular. Cada uno de los 4 anillos concéntricos representa un peso binario y las partes oscuras y blancas representan ceros y unos respectivamente. Sobre el disco se hayan colocados radialmente 4 transductores mecánicos u ópticos que detectan cada combinación binaria correspondiente a una posición del 0 al 15.

Supongamos que el detector está leyendo el número 8 (1000_2) y la posición que sigue según el movimiento del disco es la 7 (0111_2) Por más delgada que pueda ser la zona sensor del detector al pasar del 1000 al 0111 detectará un 1111, que para este caso es precisamente el número del extremo opuesto del disco.

La decisión de usar el código GRAY en vez del binario es la mejor solución al problema de la ambigüedad de lectura en un encodificador óptico.

En la figura 4.3 se muestra parcialmente un disco codificado en código GRAY.

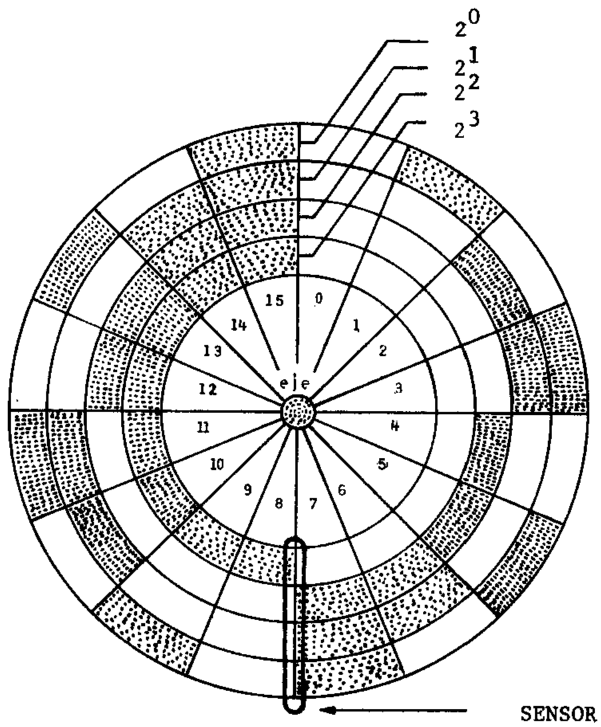


Figura 4.2 Encodificador de desplazamiento angular codificado en Binario.

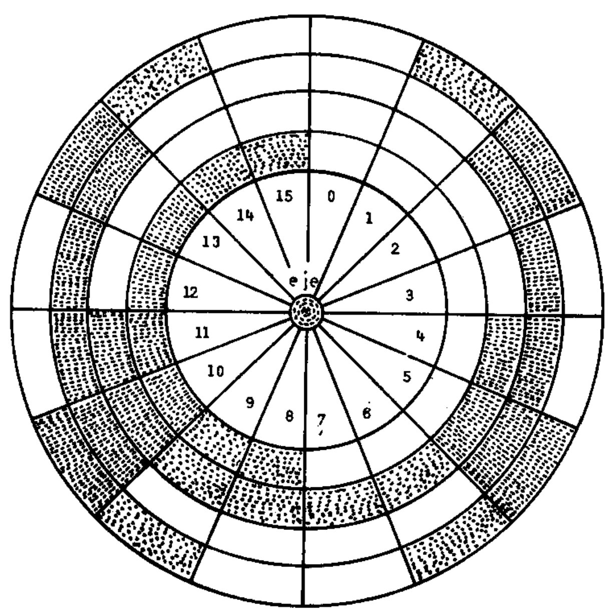


Figura 4.3 Encodificador de desplazamiento angular codificado en GRAY.

Conversión de código GRAY a BINARIO

Pasos para la conversión de binario a código GRAY.

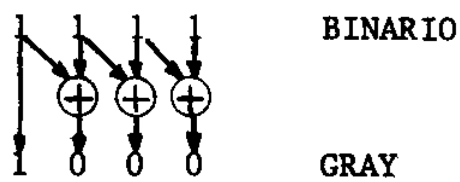
1- El bit de mayor peso del código GRAY es el mismo que el de código binario.

2- El segundo bit del código GRAY es igual a la operación EX-OR del primer y segundo bits del número binario y así sucesivamente.

3.- El tercer bit del código GRAY es igual al EX-OR del segundo y tercer bits del número binario y así sucesivamente.

EJEMPLO 4.3

Convierta el número binario 1111_2 a código GRAY



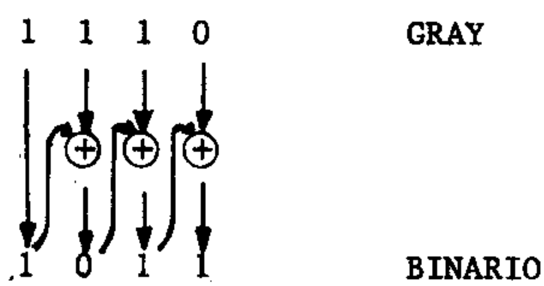
Pasos para la conversión del código GRAY a BINARIO.

1- El bit de mayor peso BINARIO es el mismo que el de código GRAY.

2- El segundo bit de código binario es igual a la operación EX-OR del primer bit de código binario y el siguiente GRAY y así sucesivamente.

EJEMPLO 4.4

Convierta el número Gray 1110 a Binario



4.4 Códigos Alfanuméricos

Un computador utiliza tanto datos alfabéticos, como caracteres especiales, tales como signos de puntuación y símbolos matemáticos. A los códigos que representan letras, caracteres y números se les llama Códigos Alfanuméricos. Generalmente estos códigos tienen un promedio de 64 caracteres, para representarlos en forma binaria se necesitan 6 bits, 26 = 64.

Código EBCDIC

El código EBCDIC (Extended-Binary-Coded-Decimal-Interchange-Code) Código de Intercambio Decimal Codificado En Binario Extendido, puede representar hasta 256 diferentes caracteres. Todos los caracteres están representados por 8 bits o dos números hexadecimales. Este código permite el uso de letras mayúsculas y minúsculas así como caracteres especiales y de control tales como NULL y PF. Estos caracteres de control los interpretan los dispositivos periféricos como las impresoras y terminales de video. Muchas combinaciones no tienen asignado un carácter. En la figura 4.4 aparece la tabulación del código EBCDIC.

EBCDIC	Bit Configuration	Hex	EBCDIC	Bit Configuration	Hex	EBCDIC	Bit Configuration	Hex	EBCDIC	Bit Configuration	Hex
NULL	0000 0000		b (blank)	0100 0000	40		1000 0000	.		1100 0000	C0
	0000 0001			0100 0001		a	1000 0001	81	A	1100 0001	C1
	0000 0010			0100 0010		b	1000 0010	82	B	1100 0010	C2
	0000 0011			0100 0011		c	1000 0011	83	C	1100 0011	C3
PF	0000 0100	04		0100 0100		d	1000 0100	84	D	1100 0100	C4
HT	0000 0101	05		0100 0101		e	1000 0101	85	E	1100 0101	C5
LC	0000 0110	06		0100 0110		f	1000 0110	86	F	1100 0110	C6
DEL	0000 0111	07		0100 0111		g	1000 0111	87	G	1100 0111	C7
	0000 1000		C	0100 1000		h	1000 1000	88	H	1100 1000	C8
	0000 1001			0100 1001	49	i	1000 1001	89	I	1100 1001	C9
	0000 1010		¢	0100 1010	4A		1000 1010			1100 1010	
	0000 1011			0100 1011	4B		1000 1011			1100 1011	
	0000 1100		<	0100 1100	4C		1000 1100			1100 1100	
	0000 1101		(	0100 1101	4D		1000 1101			1100 1101	
	0000 1110		+	0100 1110	4E		1000 1110			1100 1110	
	0000 1111		!	0100 1111	4F		1000 1111			1100 1111	
	0001 0000		&	0101 0000			1001 0000			1101 0000	D0
	0001 0001			0101 0001		j	1001 0001	91	J	1101 0001	D1
	0001 0010			0101 0010		k	1001 0010	92	K	1101 0010	D2
	0001 0011			0101 0011		l	1001 0011	93	L	1101 0011	D3
RES	0001 0100	14		0101 0100		m	1001 0100	94	M	1101 0100	D4
NL	0001 0101	15		0101 0101		n	1001 0101	95	N	1101 0101	D5
BS	0001 0110	16		0101 0110		o	1001 0110	96	O	1101 0110	D6
IDL	0001 0111	17		0101 0111		p	1001 0111	97	P	1101 0111	D7
	0001 1000			0101 1000		q	1001 1000	98	Q	1101 1000	D8
	0001 1001			0101 1001		r	1001 1001	99	R	1101 1001	D9
	0001 1010		!	0101 1010	5A		1001 1010			1101 1010	
	0001 1011		$	0101 1011	5B		1001 1011			1101 1011	
	0001 1100		*	0101 1100	5C		1001 1100			1101 1100	
	0001 1101		)	0101 1101	5D		1001 1101			1101 1101	
	0001 1110		;	0101 1110	5E		1001 1110			1101 1110	
	0001 1111		¬	0101 1111	5F		1001 1111			1101 1111	
	0010 0000		-	0110 0000	60		1010 0000			1110 0000	E0
	0010 0001		/	0110 0001	61		1010 0001			1110 0001	
	0010 0010			0110 0010		s	1010 0010	A2	S	1110 0010	E2
	0010 0011			0110 0011		t	1010 0011	A3	T	1110 0011	E3
BYP	0010 0100	24		0110 0100		u	1010 0100	A4	U	1110 0100	E4
LF	0010 0101	25		0110 0101		v	1010 0101	A5	V	1110 0101	E5
EOB	0010 0110	26		0110 0110		w	1010 0110	A6	W	1110 0110	E6
PRE	0010 0111	27		0110 0111		x	1010 0111	A7	X	1110 0111	E7
	0010 1000			0110 1000		y	1010 1000	A8	Y	1110 1000	E8
	0010 1001			0110 1001		z	1010 1001	A9	Z	1110 1001	E9
	0010 1010			0110 1010			1010 1010			1110 1010	
	0010 1011		,	0110 1011	6B		1010 1011			1110 1011	
	0010 1100		%	0110 1100	6C		1010 1100			1110 1100	
	0010 1101		_	0110 1101	6D		1010 1101			1110 1101	
	0010 1110		>	0110 1110	6E		1010 1110			1110 1110	
	0010 1111		?	0110 1111	6F		1010 1111			1110 1111	
	0011 0000			0111 0000			1011 0000		0	1111 0000	F0
	0011 0001			0111 0001			1011 0001		1	1111 0001	F1
	0011 0010			0111 0010			1011 0010		2	1111 0010	F2
	0011 0011			0111 0011			1011 0011		3	1111 0011	F3
PN	0011 0100	34		0111 0100			1011 0100		4	1111 0100	F4
RS	0011 0101	35		0111 0101			1011 0101		5	1111 0101	F5
UC	0011 0110	36		0111 0110			1011 0110		6	1111 0110	F6
EOT	0011 0111	37		0111 0111			1011 0111		7	1111 0111	F7
	0011 1000			0111 1000			1011 1000		8	1111 1000	F8
	0011 1001			0111 1001	79		1011 1001		9	1111 1001	F9
	0011 1010			0111 1010	7A		1011 1010			1111 1010	
	0011 1011		#	0111 1011	7B		1011 1011			1111 1011	
	0011 1100		@	0111 1100	7C		1011 1100			1111 1100	
	0011 1101		'	0111 1101	7D		1011 1101			1111 1101	
	0011 1110		=	0111 1110	7E		1011 1110			1111 1110	
	0011 1111		"	0111 1111	7F		1011 1111			1111 1111	

Figura 4.4 Código (EBCDIC)

Código ASCII

En un esfuerzo por estandarizar los códigos de intercambio de información los fabricantes de equipo relacionado a esta rama acordaron usar el código ASCII, siglas del inglés (American Standard Code for Information Interchange). Este código puede representar hasta 128 caracteres diferentes y usa 7 bits. El listado está dividido en zonas, por ejemplo la zona 011 (de los bits de mayor peso) contiene todos los caracteres numéricos más 6 caracteres especiales así el número 0 es un 30 HEX ó 011 0000.

La letra A es un 41 HEX o un 100 0001 este código también incluye los caracteres de control. En la figura 4.5 aparece el listado del código ASCII y el significado de las abreviaciones para los caracteres de control.

NOTA. No se acostumbra usar traducción para estos términos

	CONTROL		CARACTERES ALFANUMERICOS					
765 → / 4321 ↓	0	1	2	3	4	5	6	7
	000	001	010	011	100	101	110	111
0000	NUL	DLE	SP	Ø	@	P	`	p
0001	SOH	DC1	!	1	A	Q	a	q
0010	STX	DC2	"	2	B	R	b	r
0011	ETX	DC3	#	3	C	S	c	s
0100	EOT	DC4	$	4	D	T	d	t
0101	ENQ	NAK	%	5	E	U	e	u
0110	ACK	SYN	&	6	F	V	f	v
0111	BEEP	ETB	'	7	G	W	g	w
1000	BS	CAN	(	8	H	X	h	x
1001	HT	EM	)	9	I	Y	i	y
1010	LF	SUB	*	:	J	Z	j	z
1011	VT	ESC	+	;	K	[	k	{
1100	FF	FS	,	<	L	\	l	:
1101	CR	GS	–	=	M	]	m	}
1110	SO	RS	.	>	N	Λ	n	~
1111	SI	US	/	?	O	–	o	DEL

NULL	Null Idle	CR	Carriage return
SOM	Start of message	SO	Shift out
EOA	End of address	SI	Shift in
EOM	End of message	DC_0	Device control ①
			Reserved for data link escape
EOT	End of transmission	DC_1-DC_3	Device control
WRU	"Who are you?"	ERR	Error
RU	"Are you ...?	SYNC	Synchronous idle
BELL	Audible signal	LEM	Logical end of media
FE	Format effector	SO_0-SO_7	Separator (information)
HT	Horizontal tabulation		Word separator (blank,
			normally non-printing)
SK	Skip (punched card)	ACK	Acknowledge
LF	Line feed	②	Unassigned control
V/TAB	Vertical tabulation	ESC	Escape
FF	Form feed	DEL	Delete Idle

Figura 45 Listado del Código ASCII

Código TTY

El código TTY (Tele Type) a menudo llamado BAUDOT usa solamente 5 bits por carácter como resultado algunas palabras del código se emplean para representar más de un carácter. El código TTY ha sido extendido para representar 32 caracteres diferentes usando el carácter especial MODE-CHANGE.

El transmisor y el receptor que manejan este tipo de código deben comenzar con el mismo modo, generalmente el modo alfabético. Los cambios en el modo se introducen en la secuencia de las palabras del código siempre que sea necesario.

El número efectivo de bits por carácter se incrementan por encima de 5, desde que los caracteres de Mode-Change adicionan los bits al dato que se están transmitiendo o almacenando y que sin embargo no llevan información. Una variedad del código de 5 bits, usa dos caracteres de Mode-Change, uno para hacer la transferencia a un modo alfabético y otro para la transferencia al modo numérico. En la figura 4.6 aparece el listado Para el código TTY.

PALABRA	MODO ALFABETICO	MODO NUMERICO
00	Blank	Blank
01	E	3
02		
03	A	-
04		
05	S	
06	I	8
07	U	7
10		
11	D	$
12	R	4
13	J	,
14	N	
15	F	
16	C	
17	K	(
20	T	5
21	Z	
22	L	)
23	W	2
24	H	
25	Y	6
26	P	0
27	Q	1
30	O	9
31	B	
32	G	
33		
34	M	.
35	X	/
36	V	
37	Mode change	Mode change

Figura 4.6 CÓDIGO TTY

4.5 Detección de errores (Paridad)

Una de las propiedades de los códigos que hemos discutido en este capítulo es la capacidad que tienen para detectar errores cuando alguna información codificada se transmite de un dispositivo a otro, incluso cuando esa información se almacena en memoria. Los errores consisten en la pérdida o alteración de uno o más bits de una palabra manipulada o transmitida.

Uno de los formatos más utilizados para la detección de errores es el método de PARIDAD. Este método consiste en agregar a la palabra codificada un bit extra llamado precisamente BIT DE PARIDAD que se usa para determinar si el dato transmitido ha sido alterado durante el proceso de transmisión. El bit de paridad se establece como 0 ó 1 dependiendo del número de UNOS que contiene la palabra. Este bit se usa en 2 formas diferentes, una para indicar una PARIDAD PAR y otra para indicar una PARIDAD IMPAR.

En el método de PARIDAD PAR el bit de paridad se escoge de tal manera que el número total de UNOS en la palabra, incluyendo el bit de paridad sea un número par. Por ejemplo supongamos una C, codificada en ASCII como 100 0011 el grupo tiene 3 UNOS por lo tanto añadiremos un bit de paridad igual a 1 para hacer que el número total de UNOS tenga un valor PAR. Entonces la palabra quedaría como:

100 0011 1

Bit de PARIDAD

Supongamos ahora que deseamos incluir un bit de paridad par en una A codificada en ASCII como 100 0001. El grupo tiene 2 Unos por lo tanto el bit de paridad par debe ser igual a 0 para hacer que el número total de unos tenga un valor par entonces la palabra quedaría:

100 0001 0

Bit de PARIDAD

El método de Paridad Impar se usa de la misma forma, excepto que el bit de paridad toma un valor tal que el número total de unos, incluyendo el bit de paridad sea un número Impar.

C en ASCII	=	100	0011	
Paridad Impar	=	100	0011	0
A en ASCII	=	100	0001	
Paridad Impar	=	100	0001	1

El método de Paridad no puede usarse para detectar errores dobles, es decir si en una palabra que tenga 4 unos, 2 unos se convierten en ceros, la palabra seguirá teniendo Paridad Par. En este caso el método de detección de error se sofistica más, generalmente estos métodos de detección de errores dobles señalan el bit equivocado e incluso lo corrigen.

4.6 Números con signo

En cualquier sistema numérico existen números positivos (+) y números negativos (-), estos números reciben el nombre genérico de números con signo.

En la representación aritmética ordinaria un número positivo o negativo se indica precediendo a la magnitud por un (+) o un (-) por ejemplo un + 48 o un -56. En un computador una magnitud se representa en binario y el bit de mayor peso MSB se reserva para indicar el signo del número. Si el MSB es CERO el número es positivo y si el MSB es UNO el número es negativo. En la figura 4.7 aparecen varios números con signo, expresados en 8 bits binarios

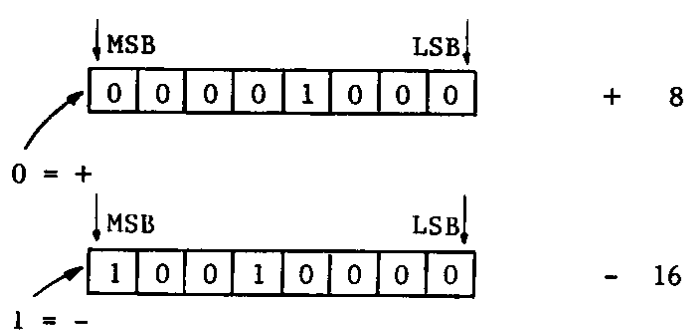


Figura 4.7 Números binarios positivos y negativos

Los números negativos pueden expresarse en forma afirmada como el 16 de la figura 4-7, sin embargo muchas computadoras manejan los números negativos en la forma de uno y Dos Complemento. En la figura 4.8 se muestra un -16 expresado en la forma del UNO y DOS complemento.

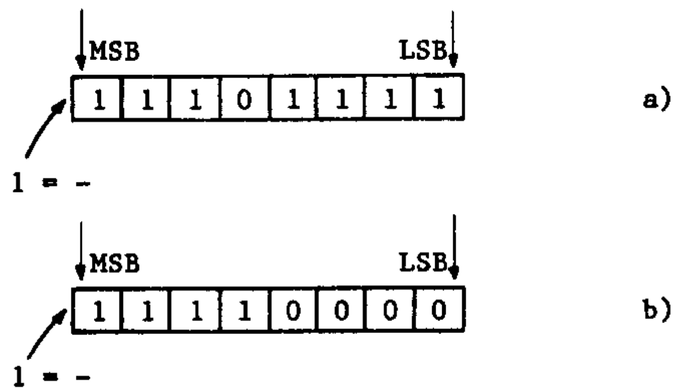


Figura 4.8 Números negativos expresados por medio de : a) Uno complemento, b) Dos complemento.

4.7 Sumas y Restas con números con signo

Las reglas para sumar o restar números binarios con signo son las mismas que las que se usan en decimal.

1- Para sumar números con el mismo signo, se le da al resultado el mismo signo, es decir, dos números positivos generan una suma positiva, dos números negativos generan una suma negativa.

2- Para sumar números con signos diferentes, se obtiene la diferencia entre ambos y el signo del resultado es el del número mayor.

3- Para restar números con signos, se cambia el signo del sustraendo y se suma el sustraendo al minuendo de acuerdo con las reglas 1 y 2.

Ejemplo 4.5

Sumas

	0000 0011	3	Números Positivos
+	0000 1000	+ 8	
+	0000 1011	11	

1111 1101	- 3	Números Negativos en
1111 1000	- 8	2 complemento.
1111 0101	- 11	Resultado en 2 complemento

1111 1101	- 3	2 Complemento
0000 1000	+ 8	Afirmado
0000 0101	+ 5	Afirmado

0000 0011	+ 3	Afirmado
1111 1000	- 8	2 Complemento
1111 1011	- 5	2 Complemento

RESTAS

8-3=+5

0000 1000	+ 8	Afirmado
1111 1101	(-)+3	2 Complemento
0000 0101	+ 5	Afirmado

3-8=-5

0000 0011	+ 3	Afirmado
1111 1000	(-)+8	2 Complemento
1111 1011	- 5	2 Complemento

Generalmente un computador efectúa la suma y la resta con un circuito únicamente sume, la multiplicación y la división con subrutinas que usan la suma y resta.

PROBLEMAS PROPUESTOS

1. ¿Qué es Código?

2. ¿Qué es un Byte y que es un Carácter?

3. ¿Cuántos tipos de códigos hay?

4. ¿Qué es un Código pesado?

5. ¿Cuál es la diferencia entre un código BCD y un 2421?

6. ¿Para qué nos puede servir un Código de Exceso 3?

7. ¿Qué es un Código no pesado?

8. ¿Cómo se convierte de un Código Gray a un Código Binario?

9. ¿Cómo se convierte de un Código Binario a un Código Gray?

10. ¿Que son los Códigos Alfanuméricos?

11. ¿Cuántos caracteres y cuántos bits representa un Código EBCDIC?

12. ¿Cómo se representan los siguientes caracteres en Código ASCII?

Y	____________	**ó**	____________
EM	____________	**ó**	____________
SP	____________	**ó**	____________
a	____________	**ó**	____________
NULL	____________	**ó**	____________
y	____________	**ó**	____________
0	____________	**ó**	____________
:	____________	**ó**	____________
4	____________	**ó**	____________
?	____________	**ó**	____________
(	____________	**ó**	____________

13. ¿Cómo se representan los siguientes caracteres en Código TTY?

$	__________
G	__________
-	__________
D	__________
/	__________

14. ¿Para qué nos sirve el método de Paridad y en qué consiste?

15. Determine si el siguiente código es un código pesado.

ABCD	CANTIDAD
0000	0
0001	1
0010	2
0011	3
0100	4
1000	5
1001	6
1010	7
1011	8
1100	9

16. Convierta los siguientes números binarios al Código Gray.

1100_2	__________
0111_2	__________
1010_2	__________
1000_2	__________
0100_2	__________

17. Convierta los siguientes números Gray a Binario.

1001 ____________________

1110 ____________________

0101 ____________________

0010 ____________________

1011 ____________________

18. Obtener el número decimal equivalente al número 0110 1000 0100 en BCD 8421.

19. Obtener a partir del Código 2421, un Código de Paridad Par.

20. Convertir el número 1100 1000 0011 perteneciente al Código BCD exceso tres a:

a) El código BCD 8421

b) E1 código BCD 2421

c) El sistema binario natural

d) El sistema decimal

21. Efectúe las siguientes operaciones de números con signo.

0000 1101 + 0000 0101	0011 0000 - 0100 1111
1110 0011 + 1001 1110	1001 1111 - 1111 1111
0100 0011 + 1111 1101	1110 1010 - 1111 1000

5 MINIMIZACIÓN DE FUNCIONES BOOLEANAS

5.0 Introducción

En el capítulo tres observamos cómo, a partir de una tabla de verdad, se puede obtener la expresión booleana que representa el comportamiento de un bloque digital. Esta expresión no siempre está en su forma más simple.

Ejemplo 5.0

Obtenga la función del bloque digital cuya tabla de verdad se muestra a continuación.

AB	F(AB)
00	0
01	1
10	0
11	1

$$F(AB) = \bar{A}B + AB$$

La expresión $F(AB) = \bar{A}B + AB$ no está en su forma más reducida. Por simple inspección visual podemos notar que los valores de F(AB) son iguales a los valores de la variable B.

En ambos minitérminos B permanece constante, si la tomamos como factor común y aplicamos después la propiedad del álgebra booleana que dice a+a =a, tenemos:

$$F(AB) = \bar{A}B + AB$$
$$F(AB) = B(\bar{A} + A)$$
$$F(AB) = B$$

En este capítulo discutiremos las técnicas de minimización que nos lleven a reducir el número de los componentes necesarios para implementar una función booleana. Esta minimización es importante debido a que la cantidad de elementos impacta en el costo, complejidad y mantenimiento de un circuito digital.

5.1 Criterio de costo

Cuando se implementaban las compuertas lógicas por medio de elementos discretos como, diodos, resistencias y transistores, el COSTO estaba relacionado con la cantidad de entradas a las compuertas que formaban un circuito.

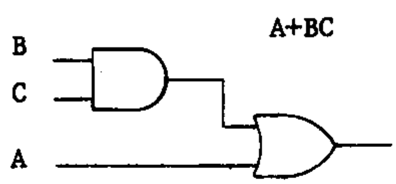
EL Costo Unitario C.U. = # total de entradas a las compuertas del circuito.

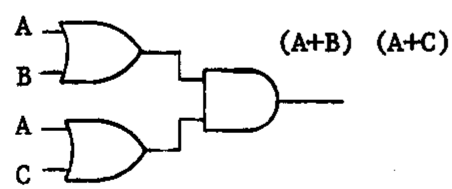
En la actualidad el costo de un Circuito Integrado no depende tanto del bloque que se encuentra en su interior, sino en gran parte del número de entradas y salidas, que determinan la cantidad de terminales en el chip. Por lo tanto el costo de un bloque digital depende del costo de los circuitos integrados empleados para su implementación. Sin embargo es recomendable comparar el C.U. con el costo total de los circuitos integrados.

Ejemplo 5.1

De la siguiente igualdad. Compare las dos expresiones en términos del costo unitario necesario para implementar ambas partes.

$$A+BC = (A+B)\ (A+C)$$





CU = 4

CU = 6

5.2 Manipulación Algebraica

En este punto aplicaremos las propiedades y las leyes del álgebra Booleana para la simplificación de expresiones booleanas llamadas también funciones de interrupción.

Existen básicamente cuatro métodos de minimización algebraica que consisten en:

1.- Factorizar términos para lograr las formas a+a=a

2.- Duplicado de un término ya existente

3.- Multiplicar por un término del tipo(a + **a**)

4.- Aplicar la Ley Distributiva.

5.2.1 Factorización

Cuando una expresión booleana en la forma de sumatoria de productos contiene dos minitérminos que difieren solo en una variable, esta puede eliminarse factorizando los términos comunes.

Ejemplo 5.2

Simplifique la función $F(ABC) = ABC + AB\bar{C}$

$$F(ABC) = ABC + AB\bar{C}$$

$$F(ABC) = AB\ (C + \bar{C}) \quad \text{- Dado} \quad C + \bar{C} = 1$$

$$F(ABC) = AB$$

La expresión $A(B + \bar{B}) = A$, tiene su equivalente en la forma de maxitérminos.

$$(A+B)\ (A+\bar{B}) = A$$

Ejemplo 5.3 Simplifique la siguiente función:

$$F(ABC) = (A+B+C)\ (A+B+\bar{C})$$
$$F(ABC) = \left[(A+B)+C\right]\ \left[(A+B)+\ \bar{C}\right] \qquad (A+B)\quad (A+\bar{B}) = A$$
$$F(ABC) = A+B$$

Al usar el método de factorización pueden aparecer minitérminos que compartan una misma literal y que tomen la forma a + 1 = 1.

Ejemplo 5.4

Simplifique la siguiente expresión:

$$F(ABC) = A + AB + A\bar{C} + A\bar{B}C$$
$$F(ABC) = A\ (1 + B + \bar{C} + \bar{B}C) \qquad \textbf{Dado} \qquad a + 1 = 1$$
$$F(ABC) = A$$

5.2.2 Duplicando un término ya existente

La propiedad del álgebra booleana que dice a=a + a +……+ a +a, pude usarse en la simplificación de funciones booleanas. Un término que aparece en una suma de minitérminos o un producto de sumatorias puede duplicarse tantas veces como sea necesario, para su combinación con otros términos.

Ejemplo 5.5

Minimizar la siguiente expresión duplicando término

$$F = AB\bar{C} + ABC + A\bar{B}C$$

$$F = AB\bar{C} + ABC + ABC + A\bar{B}C \qquad \textbf{Duplicando ABC}$$

$$F = AB\ (\bar{C}+C) + AC\ (B+\bar{B}) \quad \textbf{Dado} \quad \mathbf{a + \bar{a} = 1}$$

$$F = AB + AC$$

5.2.3 Multiplicando por un término del tipo (a + $\bar{a}$)

En ciertas ocasiones se presentan funciones tales como:

$$F = XY + \bar{X}Z + YZ$$

Aquí no se visualiza una posible simplificación por factorización. En tal caso podemos multiplicar el término YZ por(X+X) donde ($\bar{X}$ +X) = 1.

Ejemplo 5.6

$$F = XY + XZ + YZ$$

$$F = XY + \bar{X}Z + (X+\bar{X})YZ \quad \textbf{Dado} \quad \text{multiplicando por } (\bar{X} + X)$$

$$F = XY + \bar{X}Z + XYZ + \bar{X}YZ$$

$$F = XY + XYZ + \bar{X}Z + \bar{X}YZ$$

$$F = XY\ (1+Z) + \bar{X}Z\ (1+Y)$$

$$F = XY + \bar{X}Z$$

5.2.4 Aplicando la Ley Distributiva

La Ley Distributiva a + bc = (a+b) (a+c), se presenta como una posible solución para expresiones de la forma, a + ab.

Ejemplo 5.7

Simplifique la expresión $A + \bar{A}B$

$F = A + \bar{A}B$

$F = (A + \bar{A})\ (A + B)$ $\quad$ $a + \bar{a}b$ Ley Distributiva

$F = 1\ (A+B)$ $\quad$ **Dado** $\quad$ $a + \bar{a} = 1$

$F = A + B$

5.3 Mapas de Karnaugh

El mapa de Karnaugh es un método gráfico para la representación y minimización de funciones booleanas. Se usa para simplificar funciones de 2, 3 y 4 variables, pero puede extenderse satisfactoriamente a funciones de 5 y 6 variables.

Su operación se basa en la combinación de minitérminos los cuales difieren en solo una variable, como $AB + A\bar{B} = A(B + \bar{B}) = A$

Un mapa para una función de N variables consiste de 2^n cuadros. Donde cada cuadro representa a un minitérmino, además entre los minitérminos de cuadros adyacentes debe haber un solo cambio en una de sus variables.

Un mapa para una función de 2 variables tiene 2^2=4 cuadros, para 3 variables 2^3=8 cuadros, para 4 variables 2^4=16 y así sucesivamente. El mapa de muestra en la figura. 5.1

f(AB)	$\bar{A}$	A
$\bar{B}$	$\bar{A}\bar{B}$	$A\bar{B}$
B	$\bar{A}B$	AB

Figura. 5.1 Mapa de Karnaugh para una función de dos variables

La función se encuentra graficada en una cuadrícula donde las coordenadas son A y B. En el eje horizontal la mitad derecha del mapa corresponde a la variable afirmada A y la izquierda a su complemento $\overline{A}$. Lo mismo sucede con la variable B graficada en el eje vertical. Generalmente se acostumbra marcar la zona para cada variable con su etiqueta correspondiente figura.5.1

Si se desea graficar la expresión AB en el mapa se indica escribiendo un 1 en el cuadro donde las variables A y B son comunes, como se muestra en la figura.5.2 en general cada cuadro impreso representa un término formado por el producto de las variables comunes al cuadro.

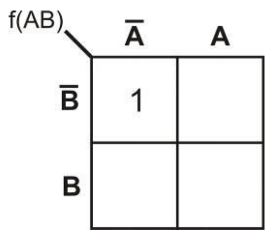


Figura. 5.2 Gráfica en un mapa para la expresión AB.

Para simplificar el acomodo de las etiquetas correspondientes a cada zona se indican las variables alfabéticas en la parte superior izquierda del mapa, para el ejemplo de una función de dos variables, A se grafica en el eje horizontal y B en el eje vertical. Por último las zonas se marcan con un número 0 ó 1 como se muestra en la figura. 5.3

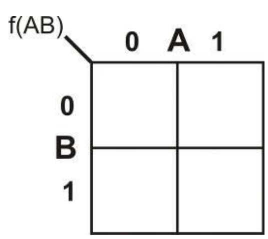


Figura. 5.3 Para indicar cada zona, las variables pueden sustituirse por números.

Un mapa para una función de tres variables se muestra en la figura 5.4 se puede observar que existe físicamente una variable modificada entre dos cuadros adyacentes.

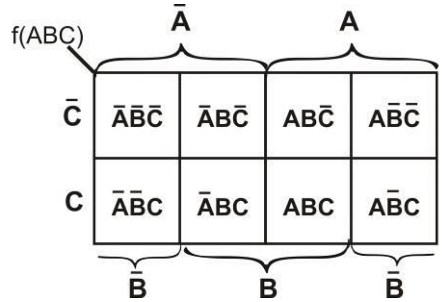


Figura. 5.4 Mapa de karnaugh para una función de 3 variables indicando la zona correspondiente a cada variable.

En la figura.5.5 aparece el mapa de Karnaugh con la distribución acostumbrada. En el eje horizontal se grafican simultáneamente las variables A y B por este motivo la etiqueta que aparece en la parte superior de cada columna es de dos dígitos y dan las combinaciones 00,01,11 y10.

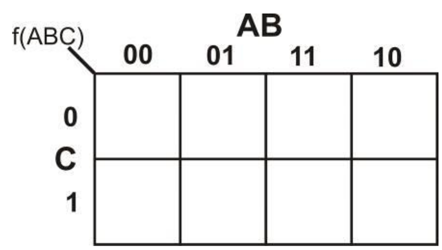


Figura. 5.5 Mapa de karnaugh para una función de 3 variables

Para graficar una expresión de cuatro variables tenemos que utilizar un mapa de 2^4=16 cuadros. En el eje horizontal se colocan las variables A y B, y en el eje vertical las variables C y D. En la figura 5.6 se muestra un mapa de Karnaugh para esta función, indicando la zona correspondiente a cada variable y sus etiquetas numéricas.

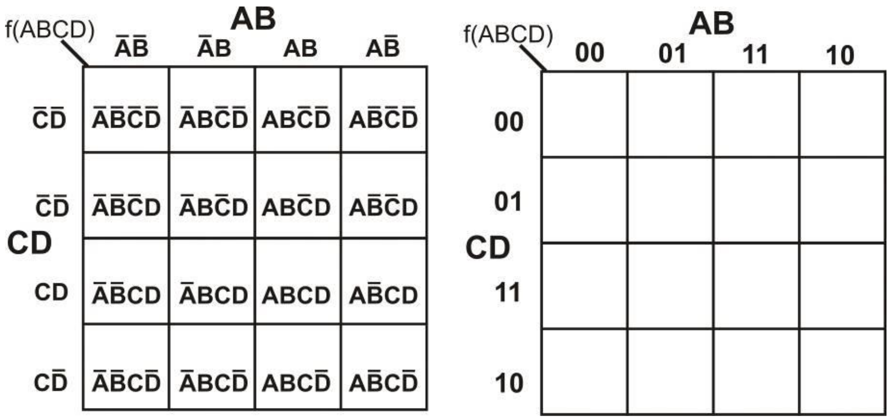


Figura. 5.6 Mapas de Karnaugh para una función de 4 variables

5.3.1 Reducción de expresiones Booleanas usando el mapa de karnaugh

La utilidad del mapa de Karnaugh se basa en que el acomodo de las áreas para cada variable, permite minimizar una expresión lógica por simple inspección. Veamos qué relación existe entre un mapa de Karnaugh y una tabla de verdad.

En la figura.5.7 se muestra una tabla de verdad para una función de 2 variables y el acomodo para cada minitérminos de la función en el mapa.

m	A	B	f(A,B)
0	0	0	m0
1	0	1	m1
2	1	0	m2
3	1	1	m3

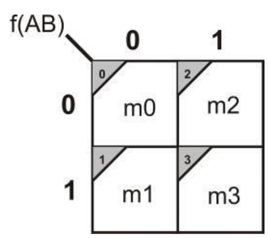


Figura. 5.7 Tabla de verdad para una función de 2 variables y Mapa de Karnaugh conteniendo los valores de la tabla.

A cada cuadro se le asigna un número en decimal que corresponde al nombre de cada minitérmino. Generalmente se escriben estos números en la parte superior derecha del cuadro pare facilitar la transferencia de los datos de la tabla, ver figura. 5.8

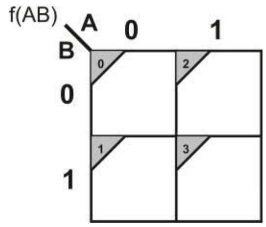


Figura. 5.8 A cada cuadro se le asigna un número decimal correspondiente al minitérmino de ese cuadro.

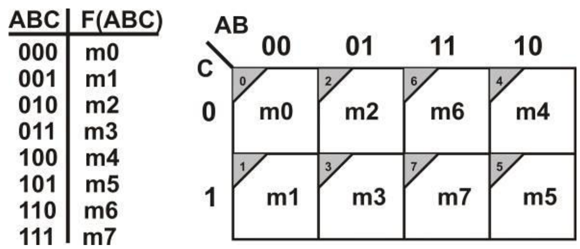


Figura 5.9 Distribución de los Minitérminos para un mapa de Karnaugh de 3 variables.

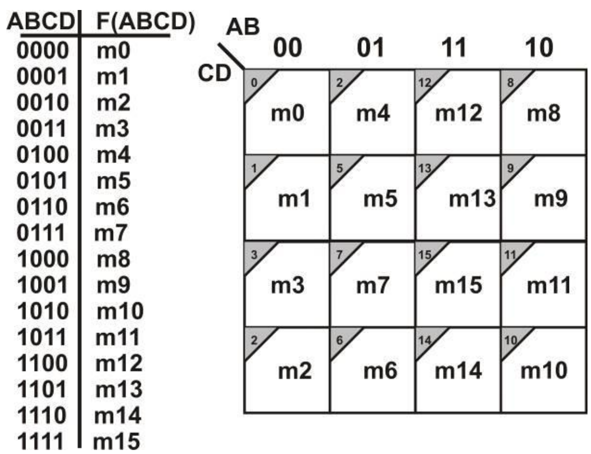


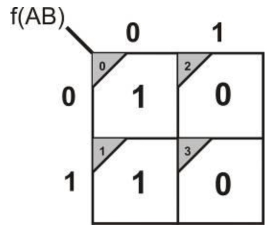
Figura. 5.10 Acomodo para los minitérminos de un mapa de Karnaugh de 4 variables.

Para transferir el contenido de la tabla al mapa de Karnaugh se colocan en su cuadro correspondiente los minitérminos pare los cuales la función es verdadera. Con el propósito de facilitar la transferencia, estos minitérminos se sustituyen por los unos. Los cuadros restantes pueden llenarse con ceros e indican los minitérminos que no aparecen en la función. Los ceros pueden omitirse si se desea.

Ejemplo 5.8

Transfiera el contenido de la siguiente tabla de verdad a su mapa de Karnaugh.

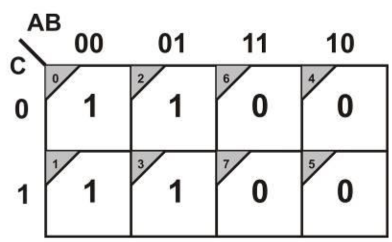
m	A	B	f(A,B)
0	0	0	1
1	0	1	1
2	1	0	0
3	1	1	0



Ejemplo 5.9

Transfiera el contenido de la siguiente tabla de verdad, de una función de 3 variables a su mapa de Karnaugh correspondiente.

ABC	F(ABC)
000	1
001	1
010	1
011	1
100	0
101	0
110	0
111	0

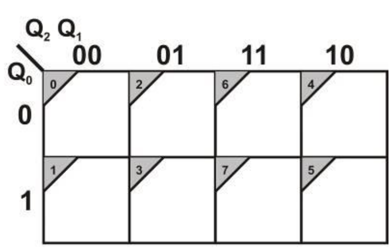


En los arreglos que hemos visto anteriormente la letra A representa la variable de mayor peso. Obviamente el orden de las variables puede cambiarse, por ejemplo que sea A la variable con un peso de 2^0 (menor peso).

Cuando se usan nombres con subíndices numéricos para las variables de un sistema, esto cambia. La variable X^o siempre será la de menor peso, como se muestra en el ejemplo 5.10

Ejemplo 5.10

$Q_2Q_1Q_0$	F
0 0 0	
0 0 1	
0 1 0	
0 1 1	
1 0 0	
1 0 1	
1 1 0	
1 1 1	



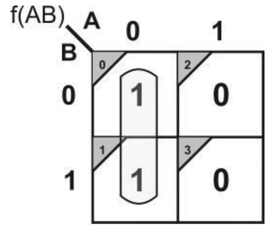
Cuando aparecen unos en cuadros adyacentes significa que existe entre ellos una variable redundante, es decir que al agruparlos se elimina una variable, usando la siguiente propiedad del álgebra booleana:

$A\overline{B} + AB = A(\overline{B} + B) = A$

Ejemplo 5.11

Simplificar la función cuya tabla de verdad aparece a continuación.

m	A	B	f(A,B)
0	0	0	**1**
1	0	1	**1**
2	1	0	**0**
3	1	1	**0**



Del grupo formado se observa que la variable B es redundante ya que adquiere el valor de B y $\overline{B}$ a lo largo del grupo, mientras que A permanece constante.

Por lo tanto:

F(AB) = A

De este ejemplo podemos deducir que el nombre que toma un grupo es igual al de la variable o variables que no cambian.

Un mismo Uno puede agruparse una o varias veces con diferentes unos adyacentes, y así sintetizar el método de "Duplicación de un Minitérmino ya existente" discutido en la sección 5.2.2 de este capítulo.

Ejemplo 5.12

Obtenga la forma más simple de la función cuya tabla de verdad aparece a continuación.

AB	F
00	1
01	1
10	0
11	1

$$F(AB) = \bar{A}\bar{B} + \bar{A}B + AB$$

1- Algebraicamente se puede lograr una máxima simplificación duplicando el término AB

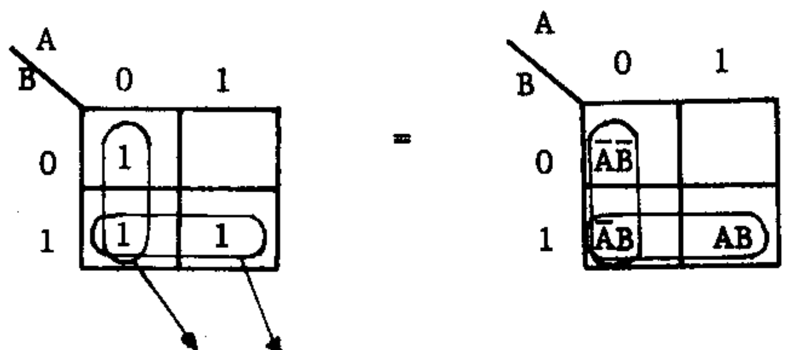
$$F(AB) = \bar{A}\bar{B} + \bar{A}B + \bar{A}B + AB$$

$$= \bar{A}\ (\bar{B}+B) + B(\bar{A}+A)$$

$$F(AB) = \bar{A}+B$$

2- Por medio del Mapa de Karnaugh el minitérmino AB puede formar parte de dos grupos distintos

AB	F
00	1
01	1
10	0
11	1



$$F(AB) = \bar{A} + B$$

Ejemplo 5.13

Simplifique la siguiente función a partir de su tabla de verdad,

ABC	F
0 0 0	0
0 0 1	0
0 1 0	1
0 1 1	1
1 0 0	0
1 0 1	1
1 1 0	1
1 1 1	1

$$F = \bar{A}B\bar{C} + \bar{A}BC + A\bar{B}C + AB\bar{C} + ABC$$

$$F = \bar{A}B\bar{C} + \bar{A}BC + A\bar{B}C + ABC + AB\bar{C}$$

$$F = \bar{A}B(\bar{C} + C) + AB(C + \bar{C}) + AC(\bar{B} + B)$$

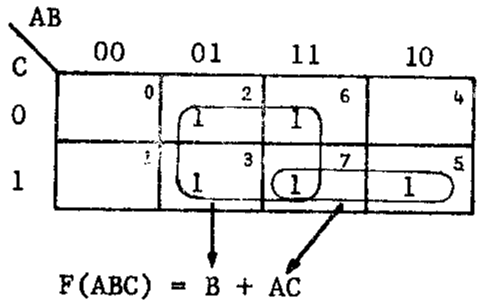
$$F = \bar{A}B + AB + AC$$

$$F = B(\bar{A} + A) + AC$$

$$\boxed{F = B + AC}$$

En la simplificación algebraica anterior podemos observar que los minitérminos 2, 3,6 y 7 se agruparon en forma separada para posteriormente llegar a solo la variable B. Por otro lado los minitérminos 2, 3, 6 y 7 se agruparon reduciéndose a la combinación AC.

Veamos la solución por medio del mapa de Karnaugh



El nombre del grupo formado por los minitérminos 2, 3, 6 y 7 es igual al de la variable que no cambia B. Y el nombre del grupo formado por los minitérminos 5 y 7 es AC.

REGLAS PARA EL USO DEL MAPA DE KARNAUGH

1- Formar el menor número de grupos

2- Formar cada grupo con la mayor cantidad de unos posible.

3- Todos los unos deberán agruparse, tomando en cuenta que un solo minitérmino puede formar un grupo.

4- El número de unos agrupados en un lazo debe ser una cantidad potencia entera de 2, (2^n) por ejemplo: 1,2, 4, 8, 16, etc.

Estas reglas se acompañan de los llamados, grupos típicos de Unos adyacentes. Un par de unos se consideran adyacentes entre sí, cuando son contiguos en forma horizontal o vertical, pero no diagonalmente. Todos los cuadros de un mapa de Karnaugh son adyacentes entre sí, esto puede manifestarse en forma más clara en un mapa de 3 variables en adelante.

Para un mapa de 3 variables 0 y 1 y 4,5 son adyacentes entre sí puesto que existe un solo cambio de un extremo a otro del mapa (La variable A).

Para indicar un grupo de un extremo a otro de un mapa se marca como se muestra en la figura. 5.11.

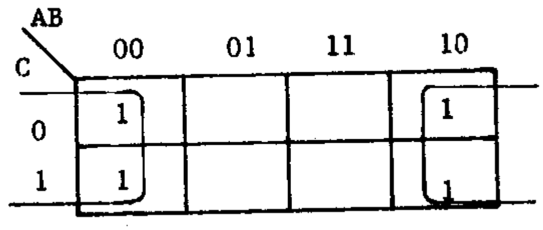
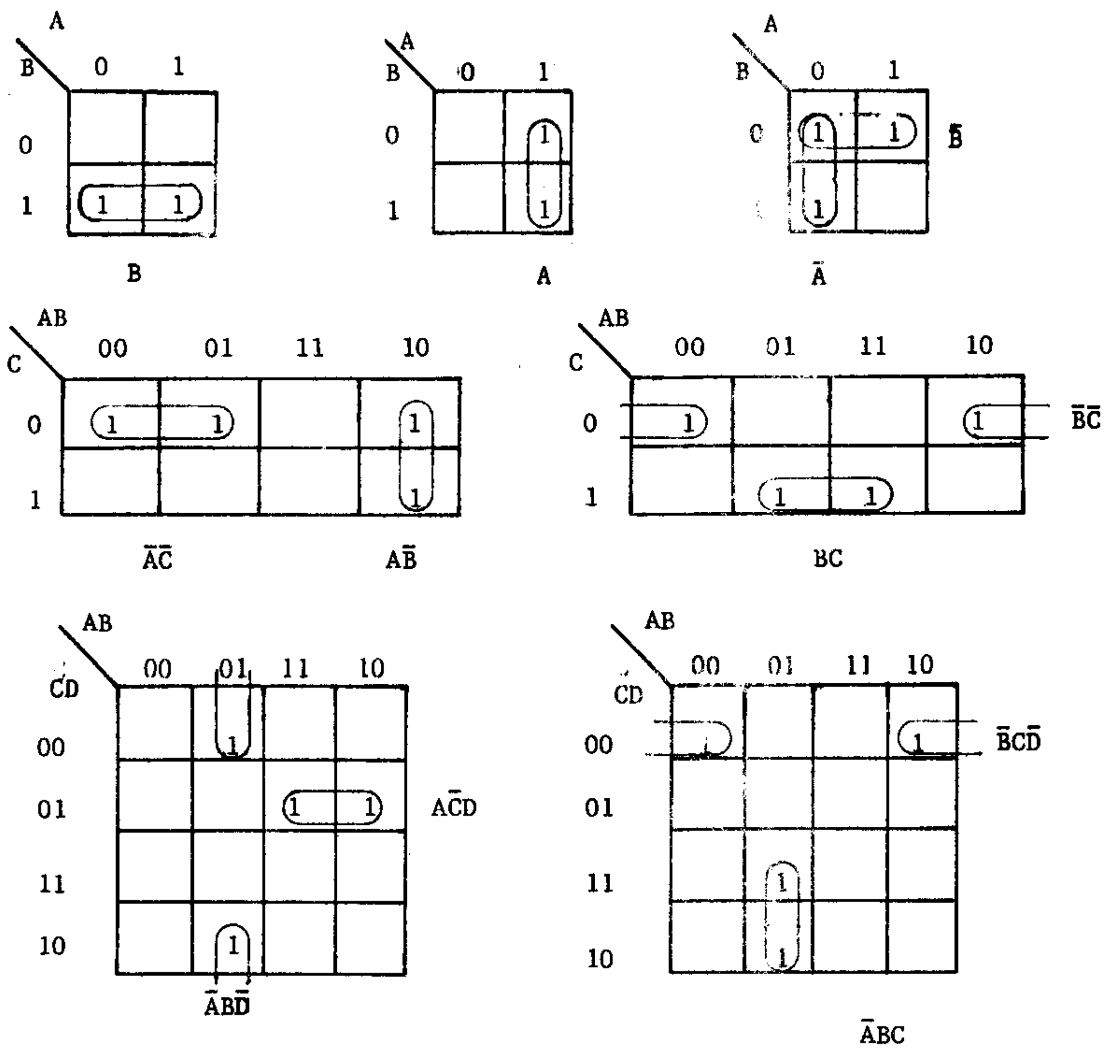
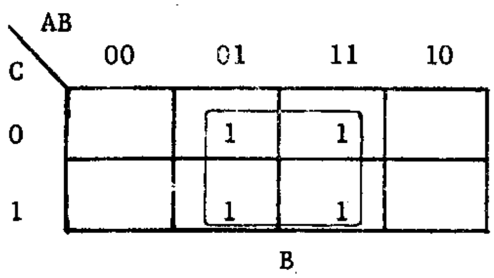


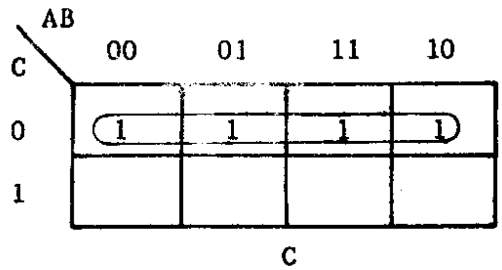
Figura. 5.11 Los grupos de un extremo a otro de un mapa se marcan con lazos abiertos

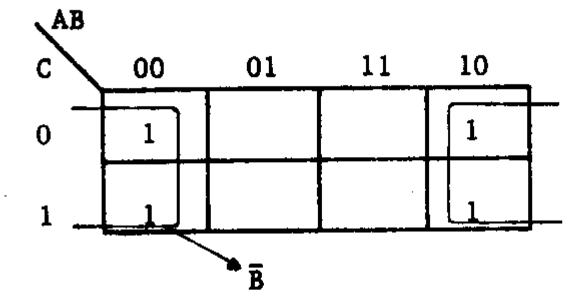
Grupos Típicos de 2 minitérminos

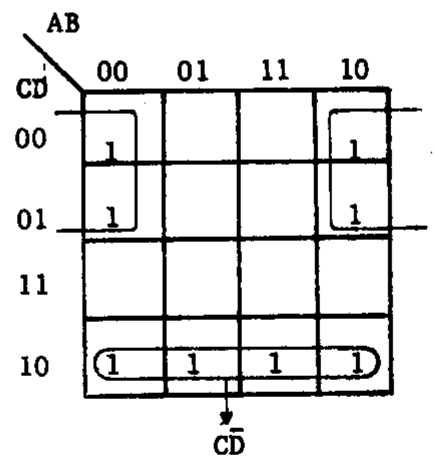


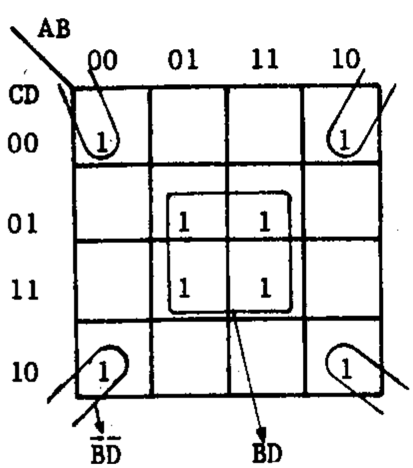
Grupos Típicos de 4 minitérminos

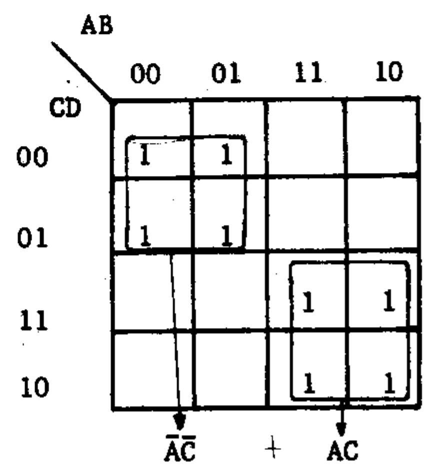

AB
C
00 01 11 10
0 1 1
1 1 1
B


AB
C
00 01 11 10
0 1 1 1 1
1
C

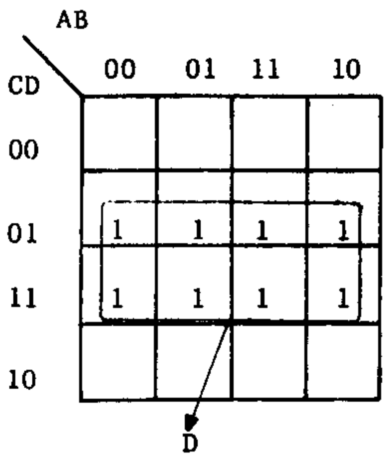

AB
C
00 01 11 10
0 1 1
1 1 1
B̄

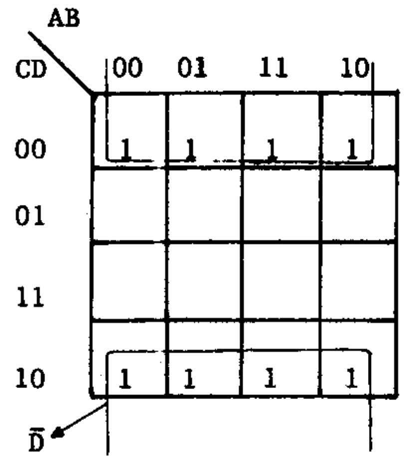

AB
CD
00 01 11 10
00 1 1
01 1 1
11
10 1 1 1 1
CD̄


AB
CD
00 01 11 10
00 1 1
01 1 1
11 1 1
10 1 1
B̄D̄
BD

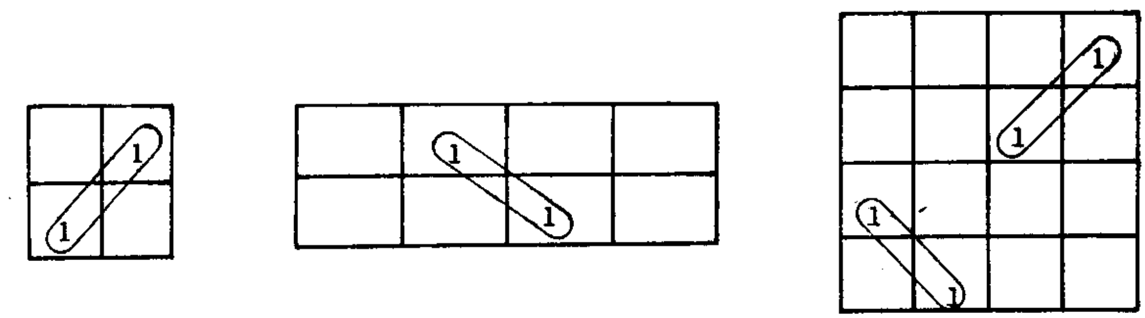

AB
CD
00 01 11 10
00 1 1
01 1 1
11 1 1
10 1 1
ĀC̄ + AC

Grupos de 8 Minitérminos

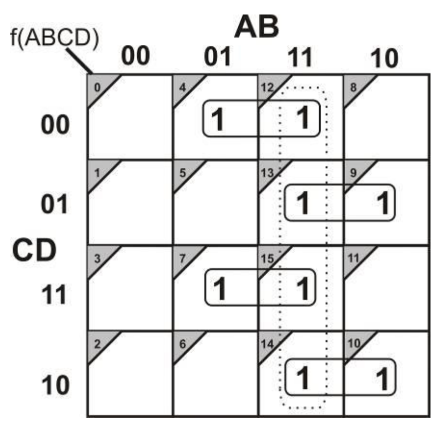




Grupos no permitidos



Lazo redundante

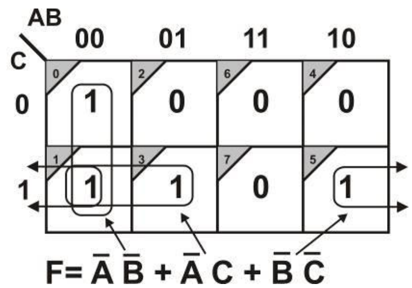


Ejemplo 5.14

Simplifique la siguiente función booleana, por medio del mapa de Karnaugh.

F(A, B, C) = Σ (0, 1, 3, 5)

ABC	F(ABC)
000	1
001	1
010	0
011	1
100	0
101	1
110	0
111	0

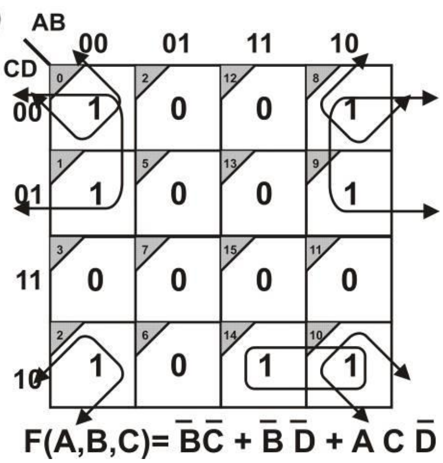


Ejemplo 5.15

Simplifique la siguiente función utilizando el mapa de Karnaugh.

F(A, B, C, D) = Σ (0, 1, 2, 8, 9, 10, 14)

ABCD	F(ABCD)
0000	1
0001	1
0010	1
0011	0
0100	0
0101	0
0110	0
0111	0
1000	1
1001	1
1010	1
1011	0
1100	0
1101	0
1110	1
1111	0



5.3.2 Productos de sumatorias a partir de un mapa de KARNAUGH

A partir de un mapa de Karnaugh, es posible obtener una función booleana expresada en la forma de productos de sumatorias. El primer paso es encontrar la forma de sumatoria de productos para $\bar{\mathbf{F}}$. Esto se logra agrupando Ceros en vez de Unos en el mapa y después aplicar el teorema de D'Morgan para convertir la **F** en F. En algunas ocasiones la función expresada a partir de F es más compacta.

Ejemplo 5.16

Dada $F_{(A, B, C)}=\Sigma(4, 5, 7)$.

Simplificar F en la forma de sumatoria de productos y productos de sumatorias a partir del mapa de Karnaugh.

a) A partir de F.

	00	01	11	10
0	0	0	0	1
1	0	0	1	1

$$F_{(ABC)} = A\bar{B} + AC \; . \; . \; . \; CU = 7$$

b) A partir de $\bar{F}$.

	00	01	11	10
0	0	0	0	1
1	0	0	1	1

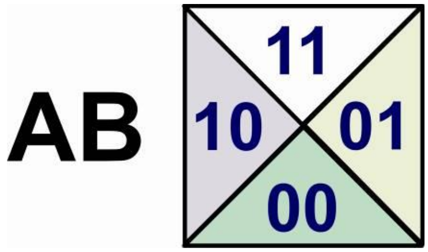
$$\bar{F}_{(ABC)} = \bar{A} + B\bar{C}$$

$$F_{(ABC)} = \overline{\bar{A} + B\bar{C}}$$

$$F_{(ABC)} = A\,(\bar{B} + C) \; . \; . \; . \; CU = 5$$

5.3.3 Mapas de KARNAUGH de 5 y 6 variables

Un mapa de Karnaugh de 5 variables puede construirse en tres dimensiones. Colocando un mapa de 4 variables sobre otro mapa también de 4 variables, los términos de la parte inferior se listan del 0 al 15 y corresponden a la zona de la variable de mayor peso negada, $\bar{A}$. Los términos de la parte superior, corresponden a la variable A afirmada y se listan del 16 al 31.



Para representar el mapa en dos dimensiones, se dividen los cuadros de un mapa de 4 variables por medio de una línea diagonal. Para colocar en la parte inferior de la línea, el mapa de A y en la parte superior el mapa de A, como se muestra en la figura 5.12.

Los términos de la parte superior e inferior de la línea diagonal combinan igual que un mapa de 4 variables.

Los términos de un mismo cuadro pueden combinarse puesto que difieren únicamente en una variable.

Los términos que aparentemente son adyacentes, no lo son. Por ejemplo, los términos 4 y 28 no son adyacentes, porque aparecen en diferente columna y en diferente posición respecto a la diagonal, además existe más de un cambio entre sus variables.

Cada término puede ser adyacente a otros 5 términos, 4 en la misma posición respecto a la diagonal y uno en el mismo cuadro. Como aparece en la figura. 5.13.

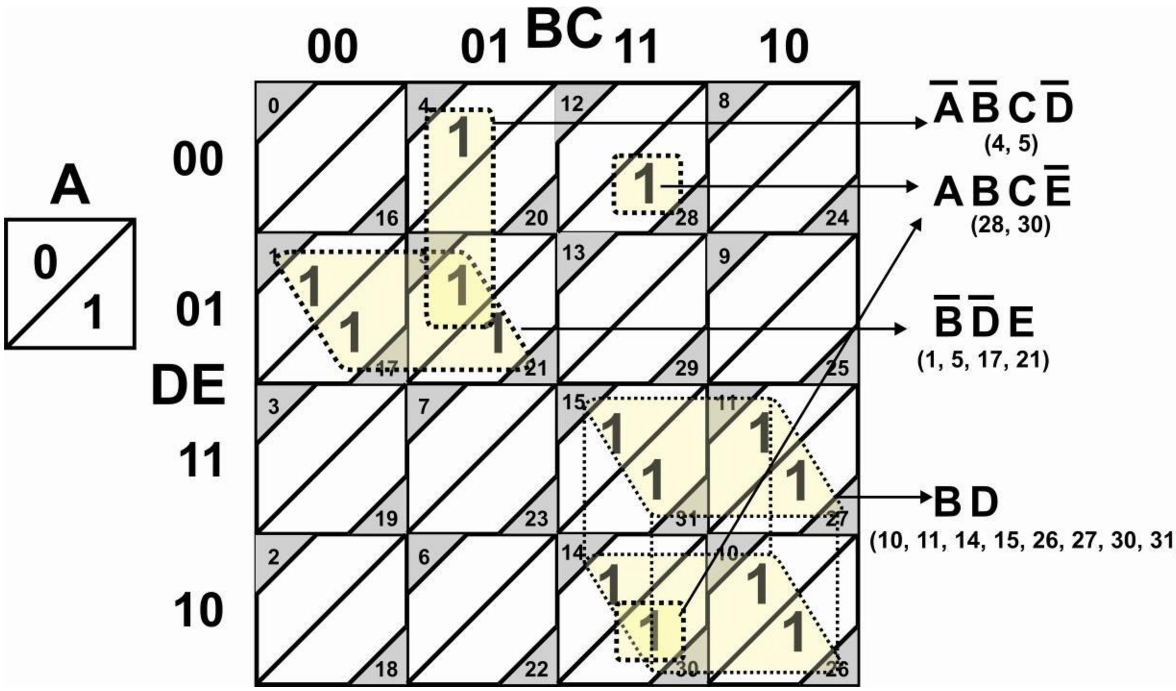


Figura. 5.12 Mapa de Karnaugh para una función de 5 variables

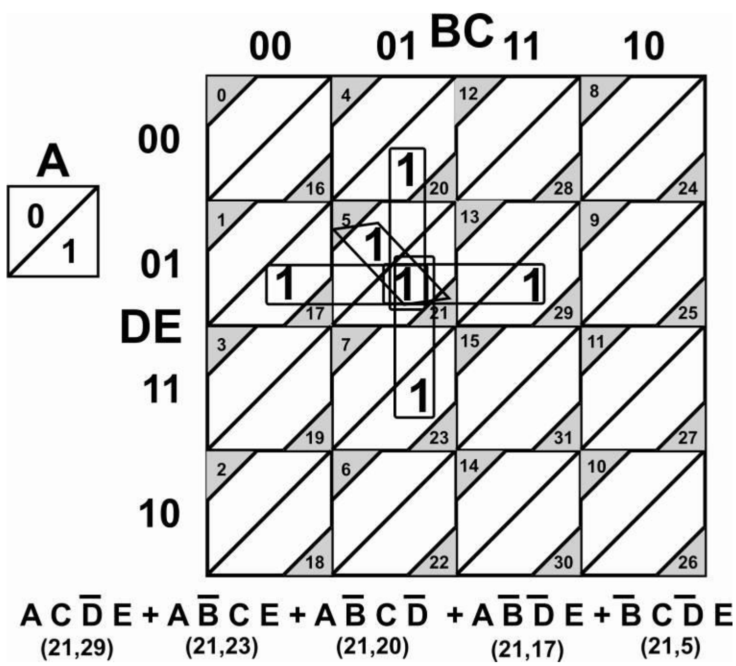
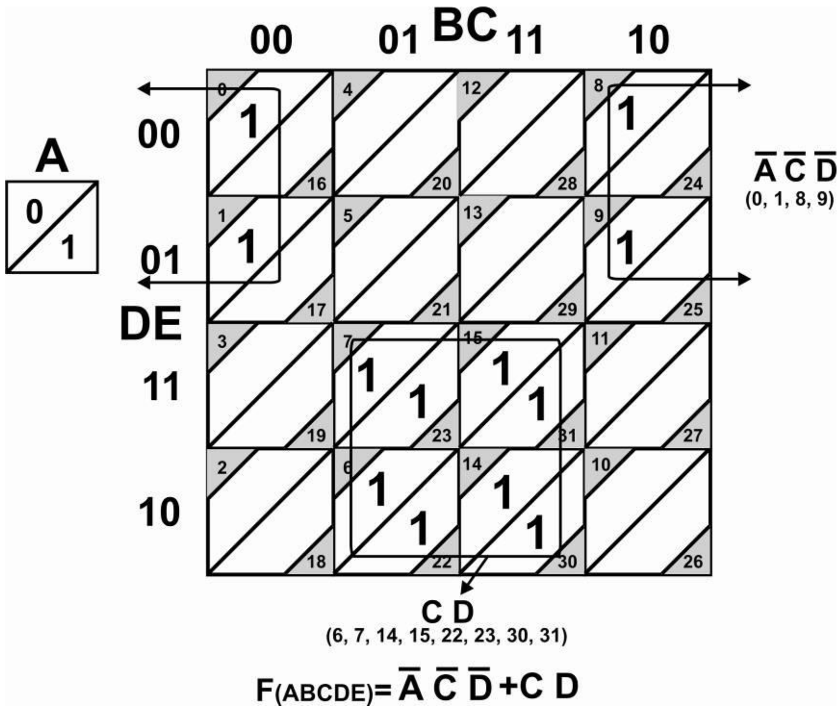


Figura.5.13. Un mismo uno puede ser adyacente a otros 5 unos

Ejemplo 5.17

Simplifique la siguiente función usando un mapa de Karnaugh de 5 variables.

$$F_{(ABCDE)} = \Sigma\ 0,\ 1,\ 6,\ 7,\ 8,\ 9,\ 14,\ 15,\ 22,\ 23,\ 30,\ 31.$$



Un mapa de Karnaugh de 6 variables puede construirse dividiendo el cuadro de un mapa de 4 variables en 4 partes como se muestra en la figura 5.14, asignando los valores de A y B (variables de mayor peso) a cada parte de un cuadro y las variables C D E y F a las hileras y columnas.

A y B se distribuyen en la siguiente forma:

Los minitérminos del 0-15 se grafican en la posición AB - 00.

Los minitérminos del 16-31 se grafican en la posición AB - 01.

Los minitérminos del 32-47 se grafican en la posición AB - 10.

Los minitérminos del 48-63 se grafican en la posición AB - 11.

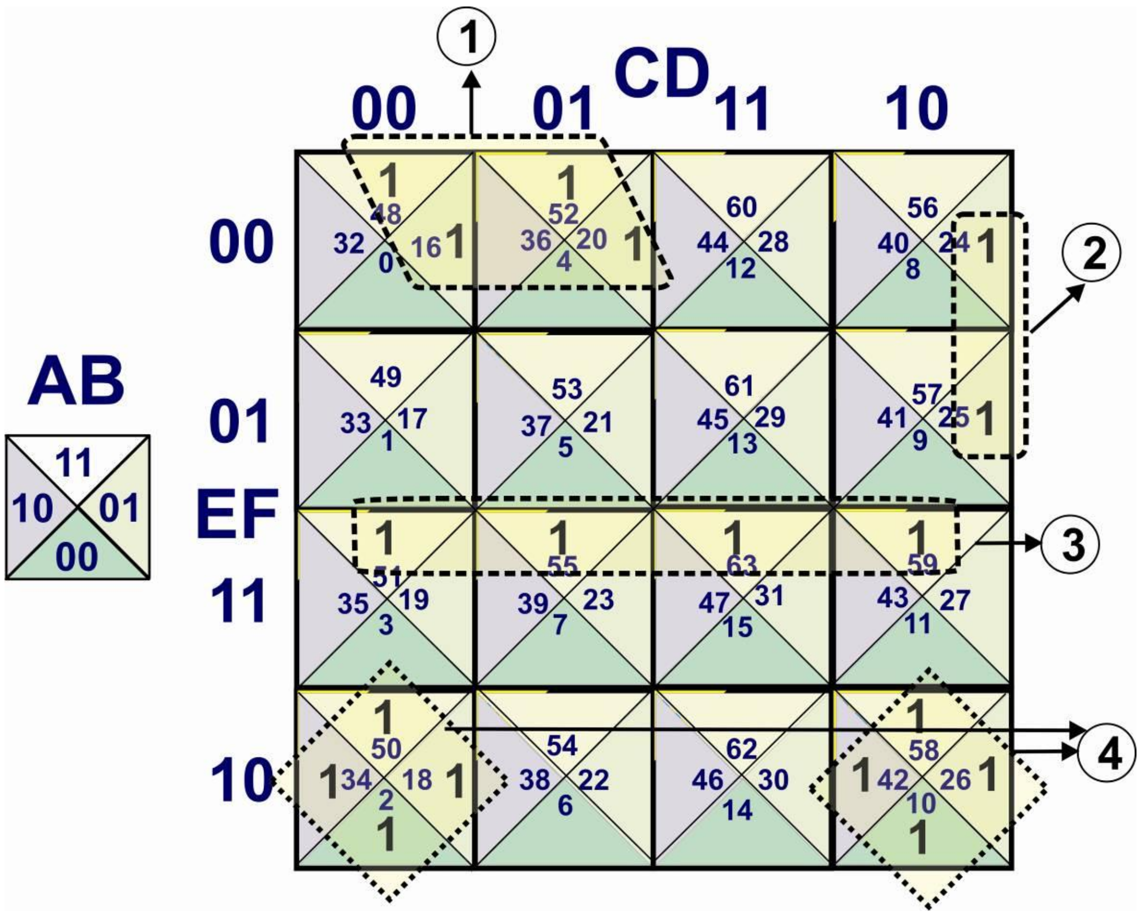


Figura. 5.14 Mapa de Karnaugh para una función de 6 variables.

Los grupos que aparecen en el mapa de la figura. 5.13 son los siguientes:

1 - $B\bar{C}\bar{E}\bar{F}$

2 - $\bar{A}BC\bar{D}\bar{E}$

3 - $ABEF$

4 - $\bar{D}E\bar{F}$

PROBLEMAS PROPUESTOS

CAPITULO 5

1.-¿Para qué nos sirve la minimización de las funciones booleanas?

2.- Mencione los cuatro métodos de minimización.

3.-¿Para qué nos sirven los Mapas de Karnaugh y cuántos cuadros tiene un Mapa de N variables?

4.-¿Cuáles son las reglas para usar el mapa de Karnaugh?

5.-Minimice algebraicamente las siguientes funciones:

$F = XY(X+Y)$

$F = A + \bar{A}B$

$F = \bar{A}BC + AB\bar{C} + ABC$

$F = A(A+B)$

$F = (a+b)\ (a+c)\ (a+d)$

6.- Simplifique las siguientes funciones utilizando el Mapa de Karnaugh.

$F(ABC) = \sum 1,2,6,$

$F(ABCD) = \sum 1,2,3,8,9,12,14$

$F(ABC) = \sum 1,3,4,5,$

$F(ABC) = \sum 0,1,4,5,6,7$

$F(ABC) = \sum 0,1,4,5,7$

$F(ABCD) = \sum 1,3,6,7,10,11,13,15$

$F(ABC) = \sum 0,1,2,3,4,5$

$F(ABCD) = \sum 0,1,2,8,9,10,11,14,15$

$F(ABCD) = \sum 2,3,4,5,6,8,9,10,11,12,13,14$

$F(ABCD) = \sum 0,1,2,3,8,9,10,11,12,13,14,15$

$F(ABCDE) = \sum 0,2,5,7,8,10,13,15,16,18,21,23,24,26,29,31$

$F(ABCDE) = \sum 1,3,4,6,9,11,12,14,17,19,22,30$

$F(ABCDEF) = \sum 0,2,8,10,16,18,24,26,32,34,40,42,48,50,53,55,56,58,61,63$

$F(ABCDEF) = \sum 16,17,18,19,20,23,24,25,26,27,28,31,32,33,34,35,36,40,41$
$42,43,44,55,63$

6 Diseño Combinacional

6.0. Definición de un bloque Combinacional

El término **Sistema Combinacional** describe a un bloque digital cuya salida es una función booleana de sus entradas. En otras palabras, los valores de la salida (0 o 1) de un bloque combinacional dependen únicamente de la combinación que tomen los valores (0 o 1) de sus variables de entrada.

Un sistema combinacional puede tener una o más entradas y una o más salidas. Estas salidas no pueden ser retroalimentadas a la entrada.

En la figura 6.0 aparece la representación de un sistema combinacional generalizado.

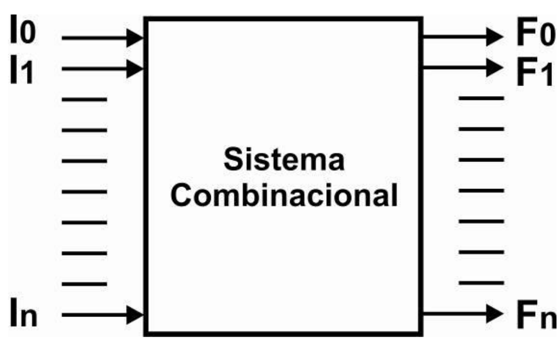


Figura 6.0 Representación de un Sistema Combinacional

Cuando se presenta un cambio en la combinación de las variables de entrada de un sistema combinacional, las salidas toman nuevos valores, estos nuevos valores aparecen con un intervalo de tiempo, determinado por los tiempos de propagación inherentes a cada compuerta usada para implementar el circuito.

El hecho de no tener retroalimentación asegura que los cambios en las entradas produzcan cambios en las salidas sin generar inestabilidad en el bloque.

Las aplicaciones típicas de los sistemas combinacionales son, en bloques de control digital, convertidores de código, circuitos que efectúan operaciones aritméticas, como sumas, comparaciones, etc. y que forman la estructura básica de calculadoras y computadoras digitales. Algunos de estos ejemplos se discuten en el punto 6.2 de este capítulo.

6.1 Metodología de Diseño Combinacional

El diseño de un sistema combinacional se puede resumir básicamente en los siguientes pasos:

1. Establecer las funciones especificas del bloque combinacional.

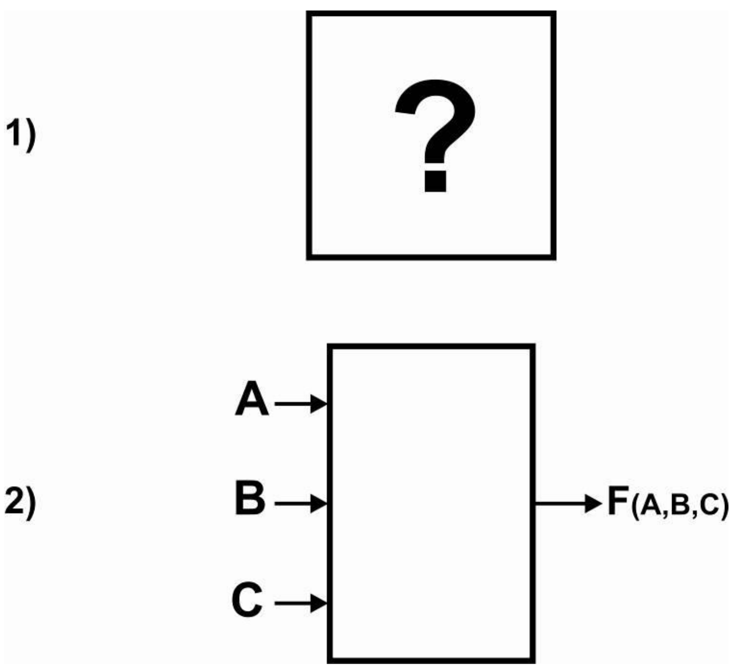
2. Determinar la cantidad de entradas y salidas al sistema.

3. Representar el comportamiento del sistema por medio de una tabla de verdad.

4. Obtener la función booleana de salida del sistema a partir de la tabla de verdad, usando el método de minimización algébrica o del mapa de Karnaugh.

5. Implementar el sistema con elementos lógicos.

Estos primeros pasos de diseño nos conducirán a la obtención del prototipo de prueba en el laboratorio. Posteriormente se discutirán los detalles para efectuar las pruebas de campo del prototipo.



3)

A	B	C	F(A,B,C)
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

4) $\boxed{F_{(A,B,C)} = \quad\quad}$

5)

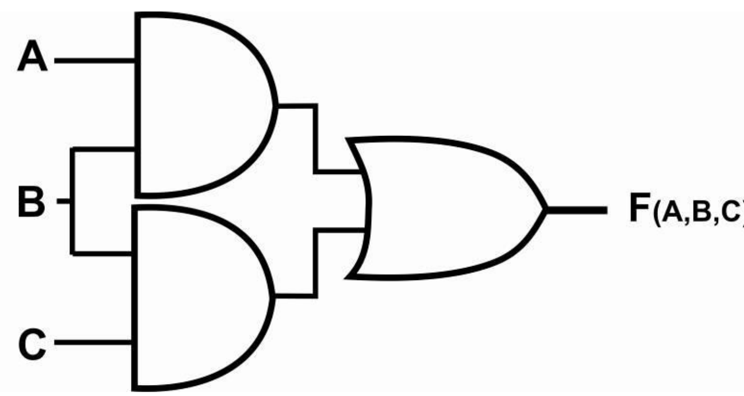


Figura 6.1Pasos del Diseño Combinacional

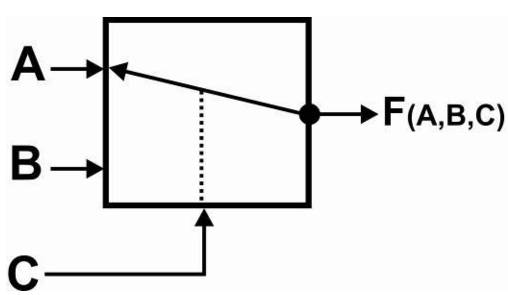
6.2 Ejemplos de diseño

Ejemplo 6.0 Selector (MULTIPLEXER)

Diseñe un circuito combinacional que tenga tres entradas denominadas A, B y C, una salida denominada F. Si la entrada C es igual a cero lógico, la salida debe ser igual a la entrada A y si C =1, la salida debe ser igual a B.

1. La descripción anterior cumple con el primer paso de diseño.

2. Determinar el número de entradas y salidas.

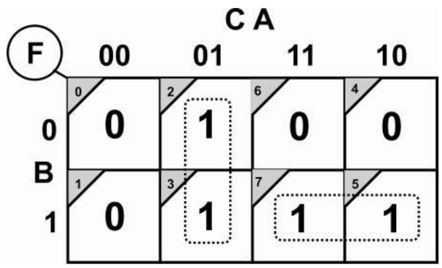


C	F
0	A
1	B

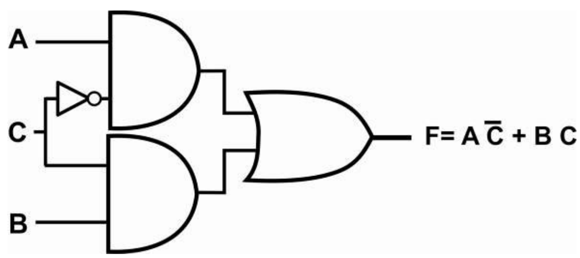
3. Se puede tener una mayor visualización del problema si se acomoda la tabla de verdad en la siguiente forma.

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

4. Obtener F(CAB)



Implementación



Ejemplo 6.1

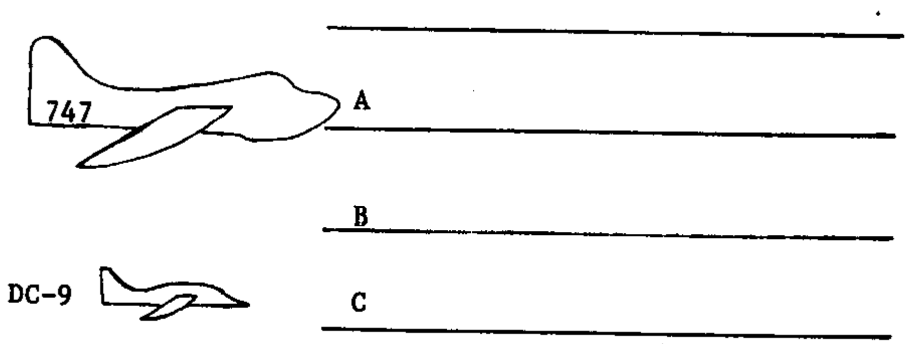
Diseñe un circuito que indique al operador de la torre de control de un aeropuerto, qué tipo de avión puede aterrizar cuando alguna de las pistas este ocupada.

a) El aeropuerto tiene 3 pistas, A, B y C

b) Pueden aterrizar Jumbos 747 ó DC-9's

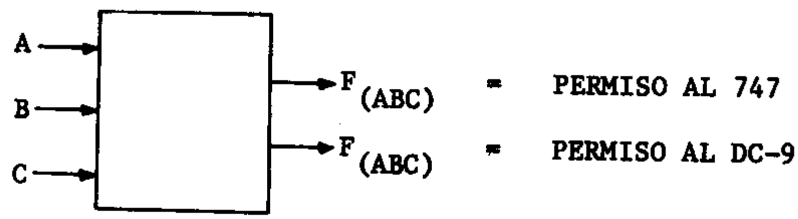
c) Un 747 necesita dos pistas contiguas para aterrizar y un DC-9 solo una

d) El 747 DC-9 Tiene mayor prioridad que el DC-9



1

2



3

ABC	747	DC-9
000	0	0
001	0	1
010	0	1
011	1	0
100	0	1
101	0	1
110	1	0
111	1	1

un cero en A, B o C indican que la pista no está disponible.

4

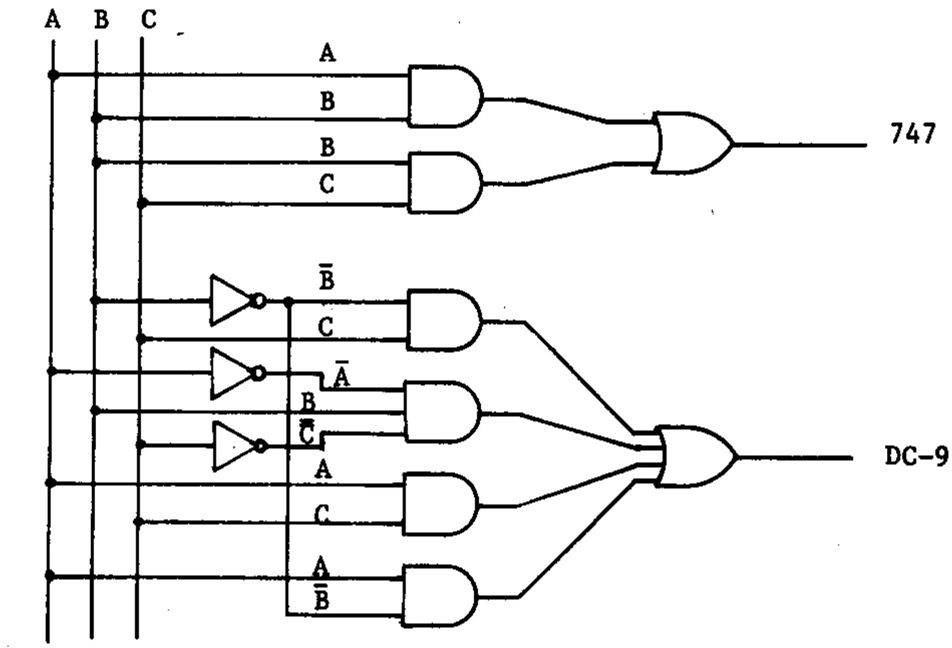
C \ AB	00	01	11	10
0	0	0	1	0
1	0	1	1	0

$$747\ (ABC) = AB + BC$$

C \ AB	00	01	11	10
0	0	1	0	1
1	1	0	1	1

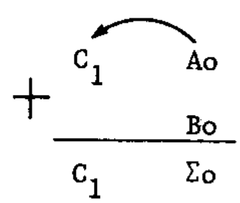
$$DC\text{-}9\ (ABC) = \bar{B}C + \bar{A}B\bar{C} + AC + A\bar{B}$$

5 Implementación



Ejemplo 6.2

Diseño de un sumador completo. En el punto 2.8.1 se discutió el procedimiento de la suma binaria. Ahora diseñaremos un dispositivo digital que efectué la suma entre 2 palabras binarias de 1 bit cada una. Imaginemos la palabra A de un solo bit y la palabra B también de un solo bit.



La suma de Ao + Bo da como resultado Σo y un acarreo C1, cuando la suma de A y B exceda la base (2).

Se puede observar que este bloque tiene solamente dos entradas Ao y Bo, y dos salidas Σo y Cl como se muestra en la figura 6.2

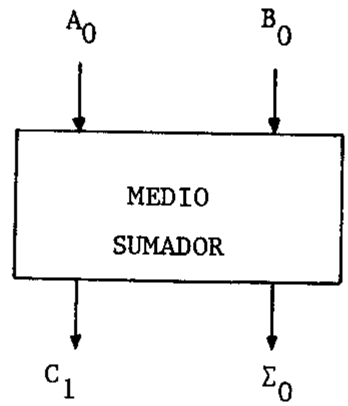


Figura 6.2 Símbolo para un Medio Sumador

Al bloque anterior se le conoce como medio sumador y no puede usarse para sumar palabras de más bits. Para poder sumar una palabra multi-bit es necesario diseñar un bloque que pueda conectarse en cascada y que considere como entrada el acarreo que genera cada par de bit's sumados anteriormente además de A y B como se muestra en la Figura 6.3

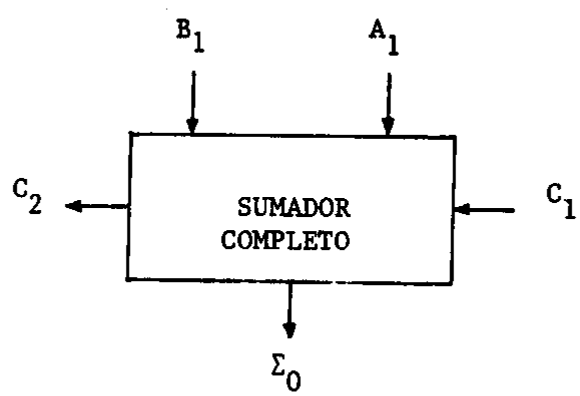


Figura 6.3 Símbolo para un Sumador Completo

Para efectuar la suma de una palabra de 4 bit's por ejemplo se pueden usar 3 sumadores completos y un medio sumador como se muestra en la figura 6.4

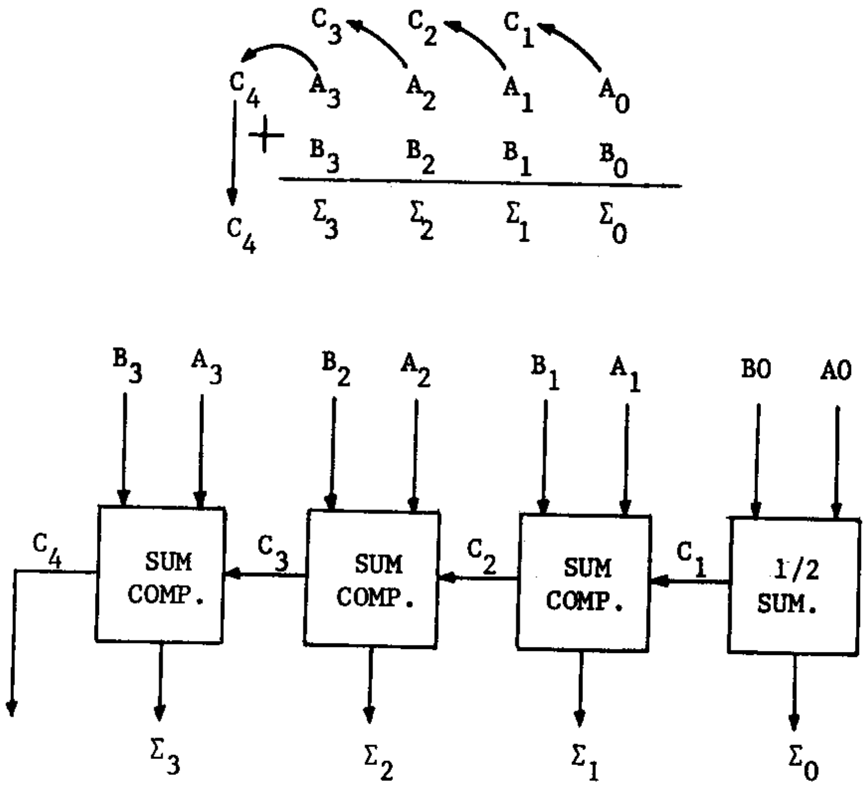


Figura 6.4 Suma Binaria de 4 bits, al sumar los bits de menor peso no se toma en cuenta el acarreo anterior puesto que no existen bits anteriores.

Entonces el bloque combinacional a diseñar tiene 3 entradas A, B y C y 2 salidas Σ1 y C2. La Tabla de Verdad del Medio sumador aparece a continuación.

$A_1B_1C_1$	Σ_1	C_2
000	0	0
001	1	0
010	1	0
011	0	1
100	1	0
101	0	1
110	0	1
111	1	1

C \ AB	00	01	11	10
0	0	1	0	1
1	1	0	1	0

C \ AB	00	01	11	10
0	0	0	1	0
1	0	1	1	1

$$C_{(ABC)} = AB + BC + AC$$

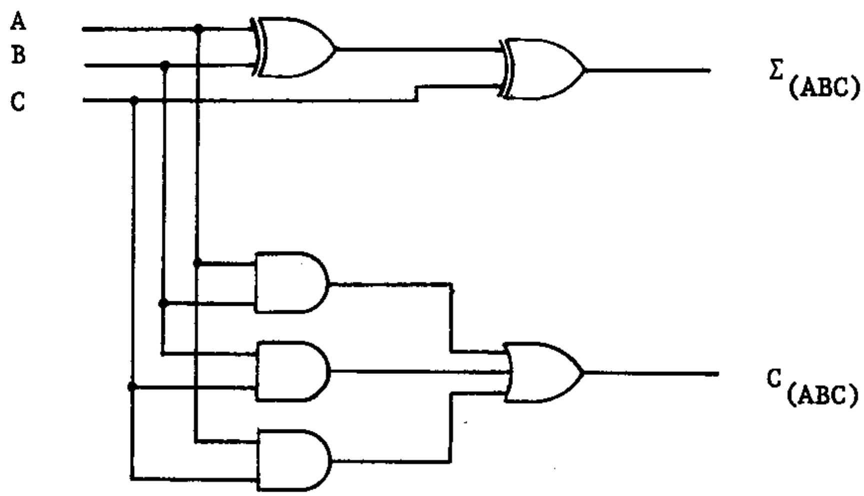
$$\Sigma_{(ABC)} = \bar{A}\bar{B}C + \bar{A}B\bar{C} + ABC + A\bar{B}\bar{C}$$

$$\Sigma_{(ABC)} = \bar{A}(\bar{B}C + B\bar{C}) + A(BC + \bar{B}\bar{C})$$

$$\Sigma_{(ABC)} = \bar{A}(B \oplus C) + A\ (\overline{B \oplus C})$$

$$\Sigma_{(ABC)} = A \oplus (B \oplus C)$$

$$\Sigma_{(ABC)} = A \oplus B \oplus C$$



6.3 Sistemas que no están completamente especificados

En los ejemplos de diseño combinacional que hemos visto anteriormente podemos notar que para cada combinación de las variables de entrada existe un valor definido pare la salida o salidas, a este tipo de sistemas combinacionales se les da el nombre de Sistemas Completamente Especificados.

En muchos casos se pueden presentar bloques en que sus combinaciones de entrada por alguna u otra razón no requieren a su salida un valor especifico, es decir para esa combinación de entrada la salida puede ser Cero o Uno, no importa cuál.

Estos casos se clasifican de dos formas. La primera se conoce como Don't Care y describe a una combinación de las variables de entrada para la cual no interesa que valor pueda tomar la salida.

El segundo caso es el Can't Happen y se refiere a una salida cuya combinación de entrada jamás llega a presentarse.

A un sistema que contenga Don't Care's o Can't Happen's se le denomina Sistema Combinacional que no está Completamente Especificado.

Para motivos de diseño el Don't Care o Can't Happen puede tomarse como cero o como uno según convenga a la solución del mapa de Karnaugh, y se indican can una X. Esto es muy útil, en la figura 6.5 se muestra cuando se toma una X como "1" o como "0".

ABC	F
000	0
001	0
010	1
011	1
100	0
101	X
110	1
111	X

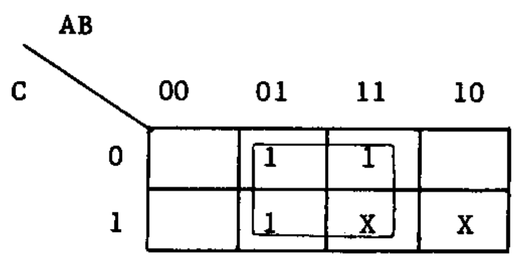


Figura 6.5 LA "X" en 7 conviene tomarla como uno la que esta en 5 conviene tomarla como cero

Si por algún motivo la combinación de entrada considerada como Don't Care o Can't Happen llegara a presentarse, el valor de la salida para esa combinación será igual al valor que se le impuso a la X en el mapa en el ejemplo de la Figura 6.5 si se llegara presentar la combinación ABC la salida será igual a uno porque esa X fue tomada como uno, por el contrario si se presenta la combinación ABC la salida será igual a cero porque la X fue tomada como cero.

Una última observación sobre las X's en un mapa, es que pueden formar grupos tomados como unos o como ceros, pero no se deben formar grupos de X'S solas.

Ejemplo 6.3

En la figura 6.6 se muestra un dispositivo empleado para la detección de tres tipos de monedas que, pasan por un plano inclinado. Consta de tres rayos de luz que inciden sobre tres fotoceldas marcadas como A, B y C. Al incidir un rayo de luz sobre una foto celda se genera un cero lógico a su salida, al interrumpirse un haz de luz la fotocelda genera un uno lógico.

El problema es entonces diseñar un circuito cuyas entradas sean A, B y C y sus salidas indiquen si pasó una moneda de .20, .50 ó 1.00.

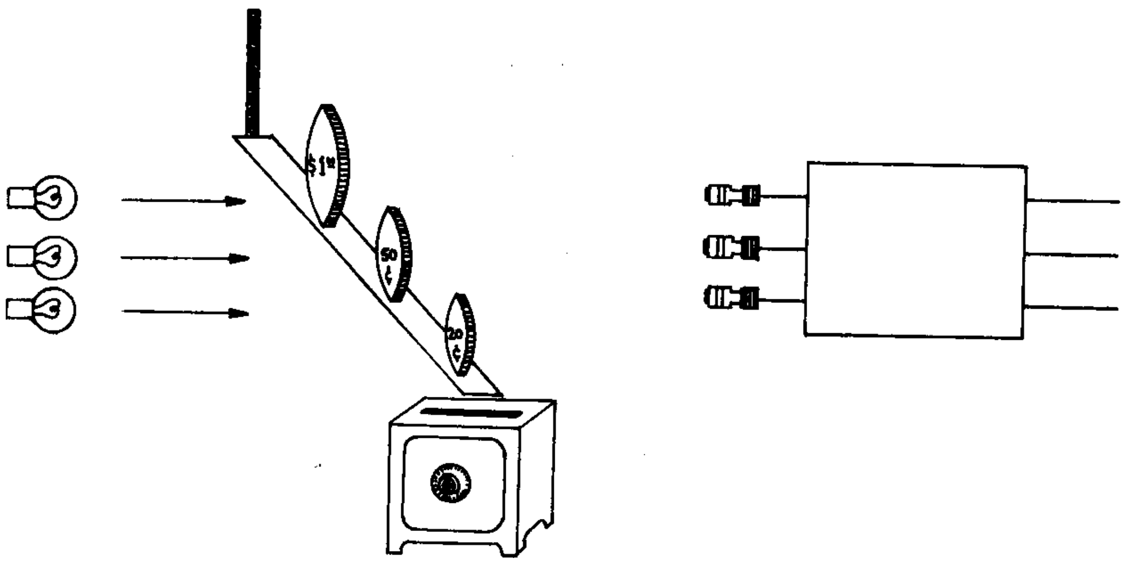
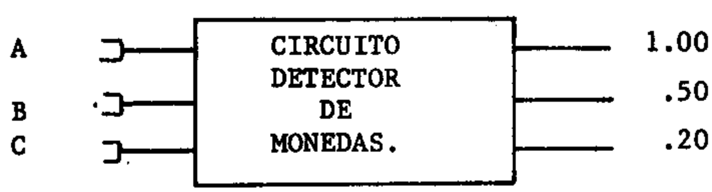


Figura 6.6 Sistema detector de tres tipos de monedas.

1. El bloque queda descrito por la redacción anterior.

2. Determinar el # de entradas y salidas



3. Tablas de verdad

ABC	100	.50	.20
000	0	0	0
001	0	0	1
010	X	X	X
011	0	1	0
100	X	X	X
101	X	X	X
110	X	X	X
111	1	0	0

ciertas combinaciones no pueden presentarse puesto que no es posible que una- moneda se despegue del plano inclinado- al bajar. En esta caso esas combinaciones son can't happen's.

4. Obtener las salidas

1.00

C \ AB	00	01	11	10
0	0	X	X	X
1	0	0	1	X

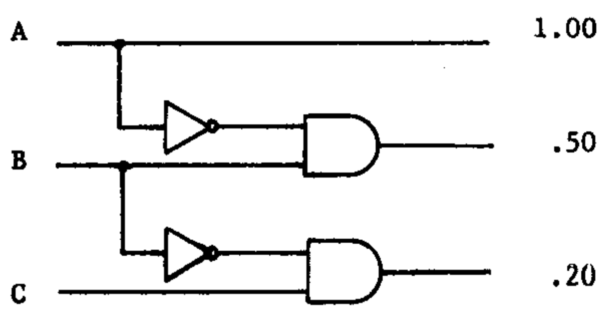
.50

C \ AB	00	01	11	10
0	0	X	X	X
1	0	1	0	X

.20

C \ AB	00	01	11	10
0	0	X	X	X
1	1	θ	0	X

5. Implementación



Ciertas combinaciones no pueden presentarse puesto que no es posible que una moneda se despegue del plano inclinado al bajar. En este caso esas combinaciónes son Can't Happens.

6.4 Display de 7 Segmentos

El término Display se usa aquí, con otros tantos sin traducción. Una definición aproximada sería la de un dispositivo óptico empleado para visualizar en forma alfanumérica o gráfica una información expresada generalmente a partir de unos y ceros.

El uso principal de un display se presenta cuando es necesario mostrar información a partir de un dispositivo digital. En este caso discutiremos un arreglo típico formado por siete segmentos con capacidad de mostrar caracteres numéricos decimales y algunos caracteres alfabéticos el nombre que recibe es precisamente Display de 7 Segmentos Figura 6.7 .

Estos display's se presentan comercialmente en una amplia variedad de tamaños, colores y tipos. La iluminación de cada segmento la producen, focos incandescentes, efectos fluorescentes sobre segmentos móviles, diodos emisores de luz (led's), cristal liquido de cuarzo y otras técnicas.

Para mostrar información en el display es necesario diseñar un sistema combinacional que convierta un código BCD a 7 salidas que enciendan o apaguen cada segmento a fin de desplegar el carácter apropiado.

A este tipo de bloques convertidores de código se les llama también Decodificadores.

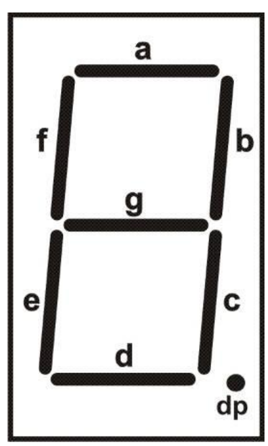
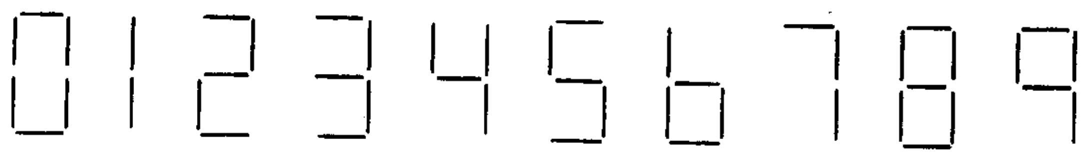


*Figura 6.7 Display de 7 segmentos, cada segmento se marca con una letra minúscula de la **a** hasta la **g.***



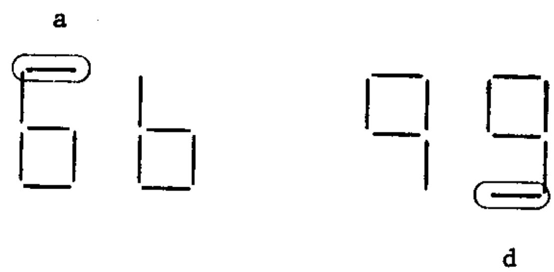
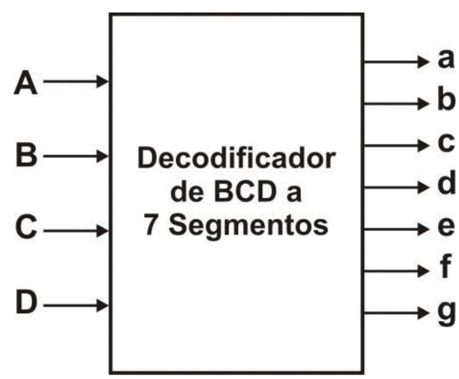


Figura 6.8 Configuración para los números del 0 al 9 en un display de 7 segmentos. Existen caracteres como el 6 y 9 que pueden configurarse indistintamente.

Ejemplo 6.4

Diseñe un decodificador de BCD a 7 SEGMENTOS

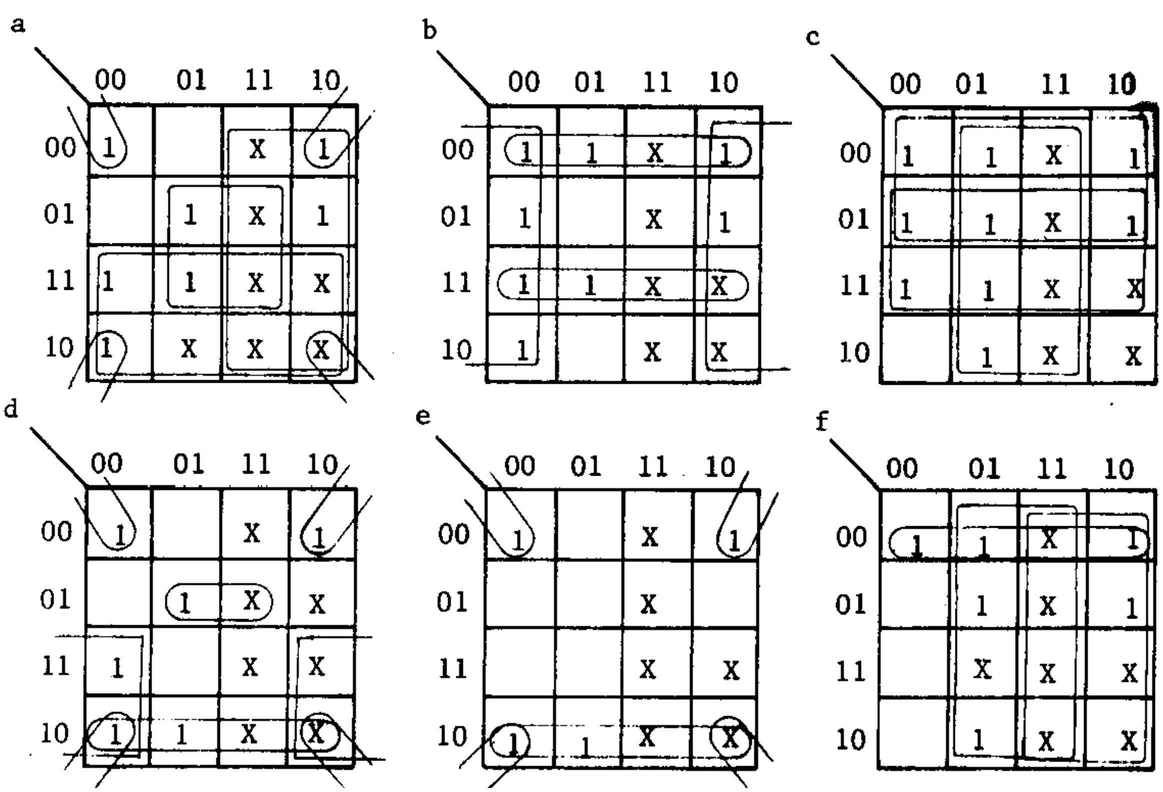
1. El bloque es un convertidor de código cuya entrada es BCD de 0 a 9, y la salida un código para manejar 7 segmentos.
2. # de entradas y salidas.

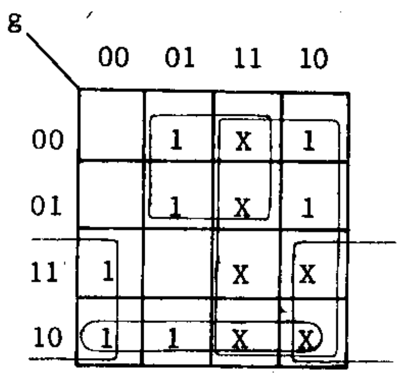


3. TABLA DE VERDAD

ABCD	a	b	c	d	e	f	g
0000	1	1	1	1	1	1	0
0001	0	1	1	0	0	0	0
0010	1	1	0	1	1	0	1
0011	1	1	1	1	0	0	1
0100	0	1	1	0	0	1	1
0101	1	0	1	1	0	1	1
0110	X	0	1	1	1	1	1
0111	1	1	1	0	0	X	0
1000	1	1	1	1	1	1	1
1001	1	1	1	X	0	1	1
1010	X	X	X	X	X	X	X
1011	X						X
1100	X						X
1101	X	CAN'T HAPPEN					X
1110	X						X
1111	X	X	X	X	X	X	X

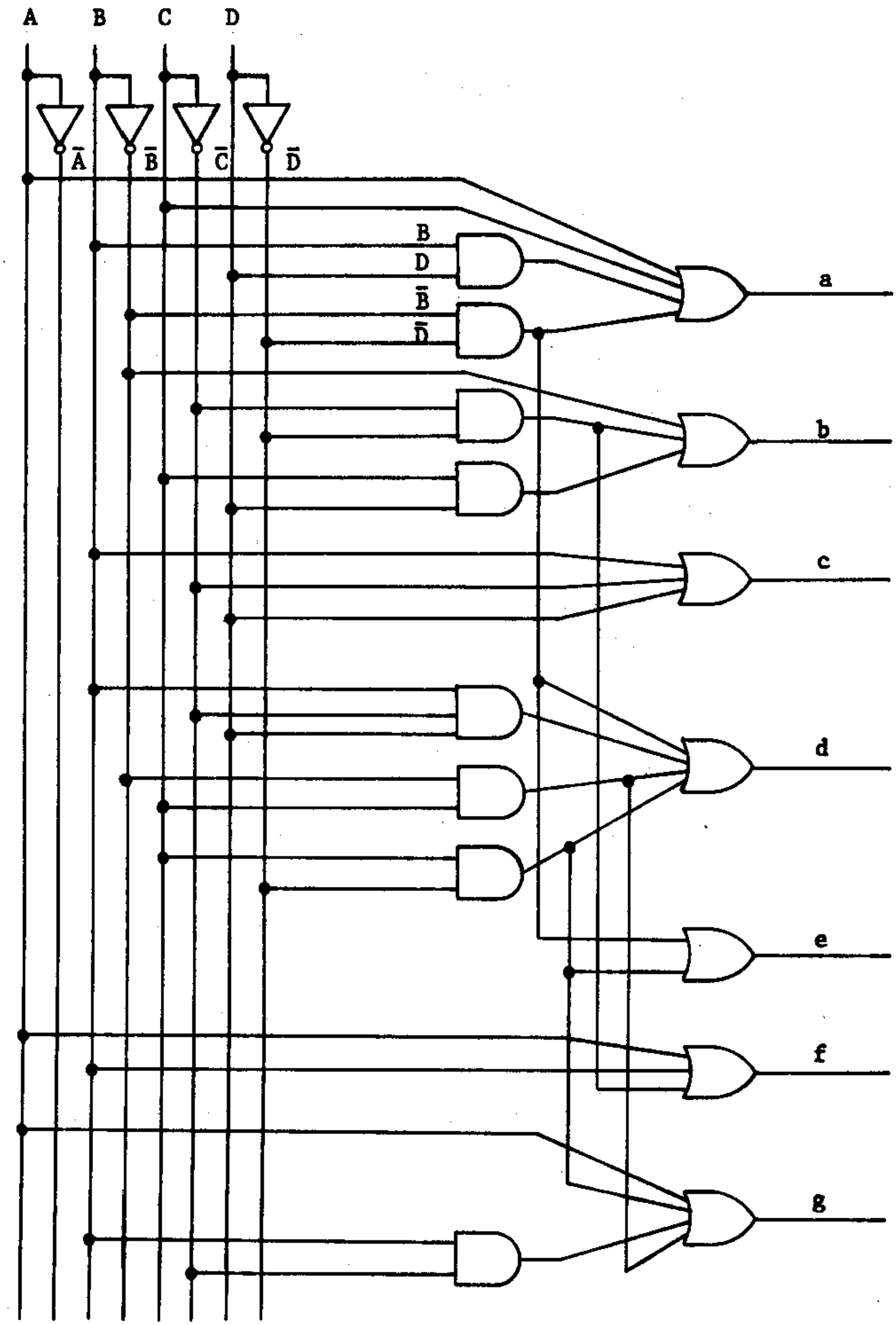
4. MAPAS





$$
\begin{aligned}
a &= A+C+BD+\overline{B}\,\overline{D}\\
b &= \overline{B}+\overline{C}\,\overline{D}+CD\\
c &= B+\overline{C}+D\\
d &= \overline{B}\,\overline{D}+B\overline{C}D+\overline{B}C+C\overline{D}\\
e &= \overline{B}\,\overline{D}+C\overline{D}\\
f &= A+B+\overline{C}\,\overline{D}\\
g &= A+C\overline{D}+B\overline{C}+\overline{B}C
\end{aligned}
$$

5. Implementación


A
B
C
D
Ā
B̄
C̄
D̄
B
D
B̄
D̄
a
b
c
d
e
f
g

6.5 Decodificadores

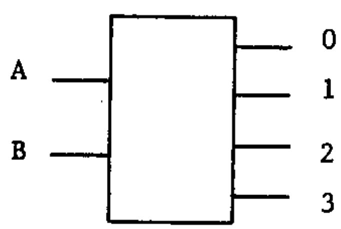
Un decodificador es un circuito lógico que convierte un código de entrada de n bits a un código de salida de una cantidad menor o igual a 2^n bits. La cantidad de combinaciones que se pueden presentar a la salida es igual a 2^n bits de entrada.

El decodificador de BCD a 7 segmentos es un bloque donde la salida no tiene una relación directa con la entrada sin embargo existen decodificadores donde si se presenta esta relación. Este tipo de decodificadores son los llamados de "n líneas de entrada a 2^n líneas de salida" para cada combinación de las líneas de entrada se habilita una sola salida a la vez, ya sea con niveles altos o bajos.

Ejemplo 6.5, Diseñar un decodificador de nivel activo alto.

1- Ok

2- Dos entradas y 4 salidas



3 - Tabla de Verdad

AB	0	1	2	3
00	1	0	0	0
01	0	1	0	0
10	0	0	1	0
11	0	0	0	1

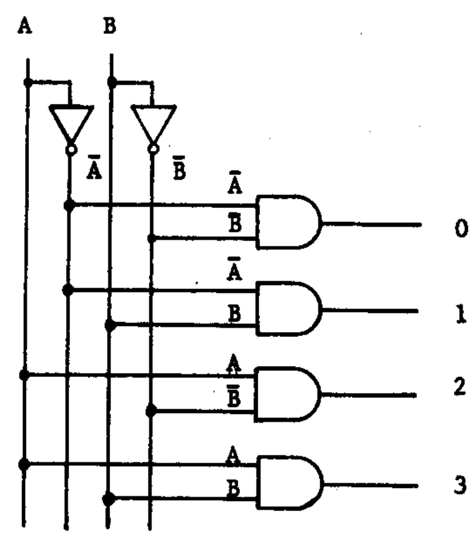
4- Cada salida es igual a un minitérmino y no existen X's es innecesario recurrir a los mapas.

$0_{(AB)} = \bar{A}\bar{B}$

$1_{(AB)} = \bar{A}B$

$2_{(AB)} = A\bar{B}$

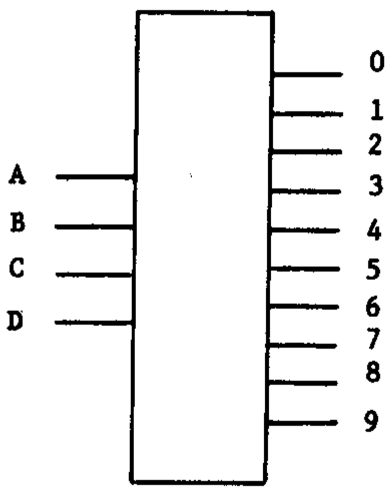
$3_{(AB)} = AB$



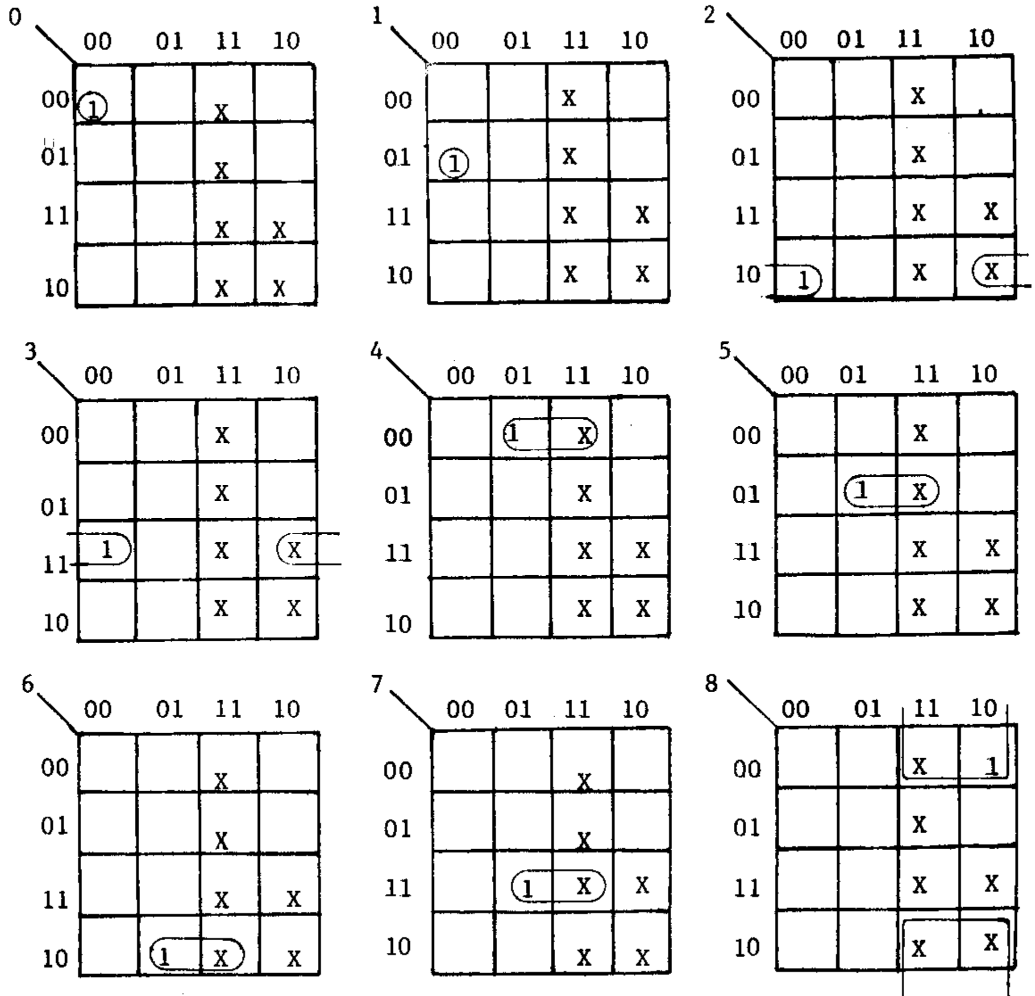
Algunos decodificadores no usan todos las 2^n posibles combinaciones de salida, sino solo algunos. Por ejemplo, un decodificador de BCD a Decimal tiene un código de entrada de 4 bits y 10 salidas. Las cuales tienen valores solo para las combinaciones de entrada del 0 al 9.

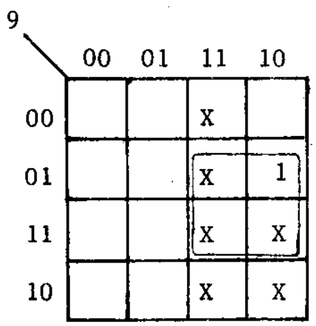
Ejemplo 6.6

Diseñe un decodificador de 4 a 10 líneas



ABCD	0	1	2	3	4	5	6	7	8	9
0000	1	0	0	0	0	0	0	0	0	0
0001	0	1								
0010	0		1							
0011	0			1						
0100	0				1					
0101	0					1				
0110	0						1			
0111	0							1		
1000	0								1	
1001	0									1
1010	X	X	X	X	X	X	X	X	X	X
1011	X									
1100	X									
1101	X									
1110	X									
1111	X									





$0_{(ABCD)} = \bar{A}\bar{B}\bar{C}\bar{D}$

$1_{(ABCD)} = \bar{A}\bar{B}\bar{C}D$

$2_{(ABCD)} = \bar{B}C\bar{D}$

$3_{(ABCD)} = \bar{B}CD$

$4_{(ABCD)} = B\bar{C}\bar{D}$

$5_{(ABCD)} = B\bar{C}D$

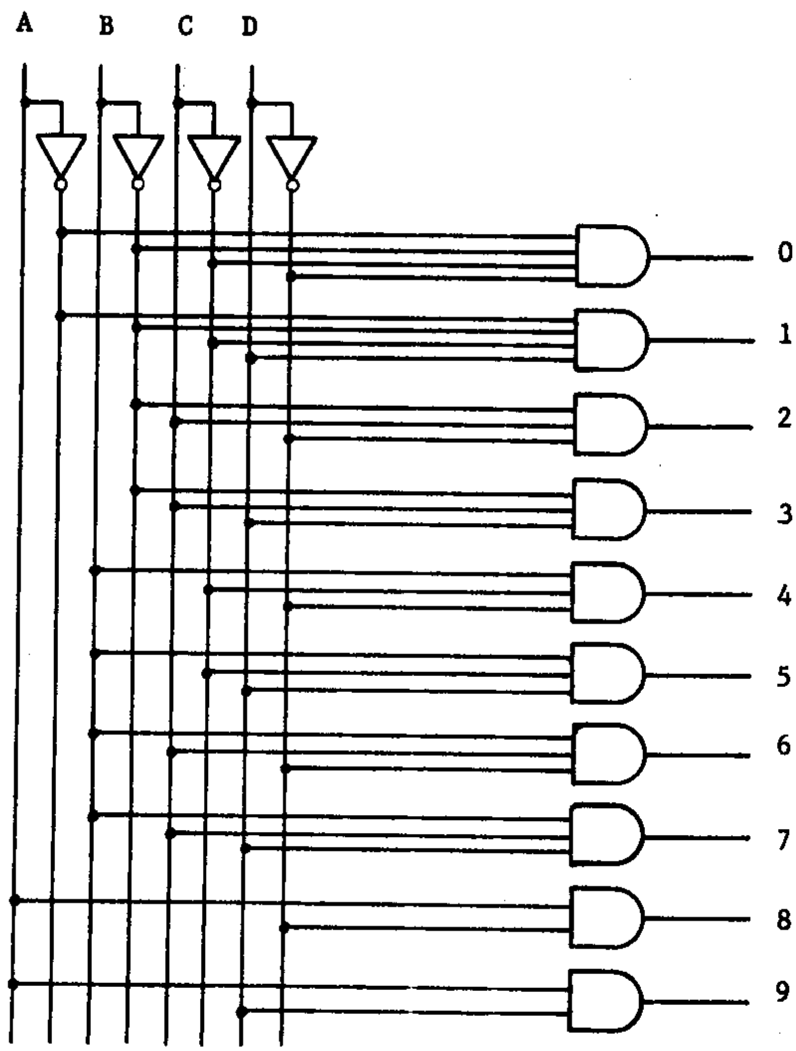
$6_{(ABCD)} = BC\bar{D}$

$7_{(ABCD)} = BCD$

$8_{(ABCD)} = A\bar{D}$

$9_{(ABCD)} = AD$

5.- Implementación



Antes de que aparecieran los displays de 7 segmentos en el mercado, se fabricaba un display encapsulado en un bulbo de cristal transparente. Cada dígito se configuraba por un filamento muy delgado. Todos los filamentos con forma de números del 0 al 9 estaban colocados uno detrás de otro. Si un filamento se iluminaba, opacaba a todos los demás, notándose claramente un dígito.

Un decodificador de 4 a 10 líneas puede servir como un decodificador de BCD a Decimal pare manejar este tipo de display, como se muestra en la Figura 6.9.

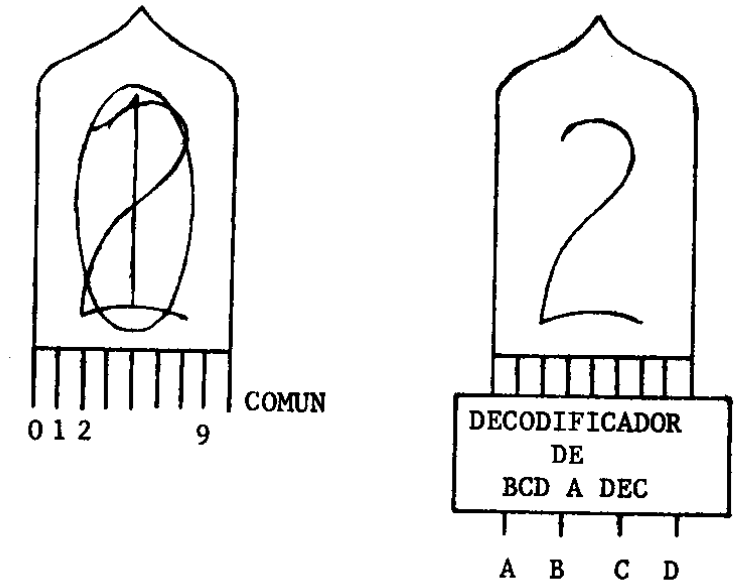
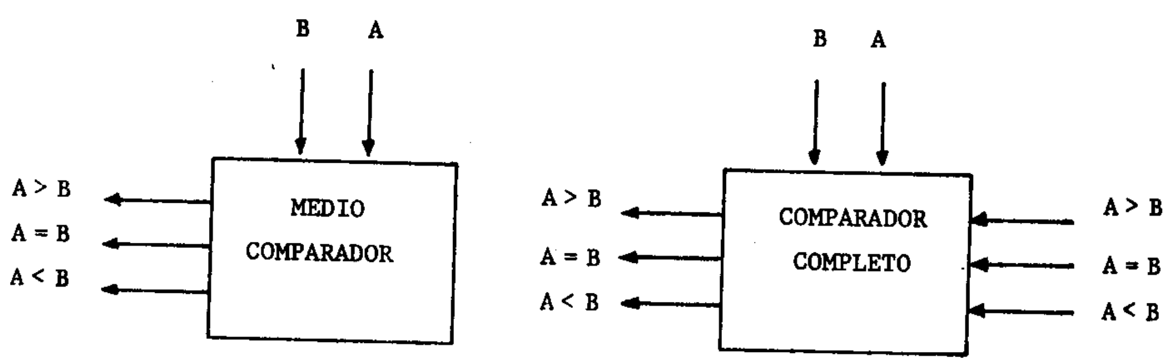


Figura 6.9 a) Display Decimal, b) Decodificador de BCD a decimal manejando un display decimal.

Ejemplo 6.7

Diseñe un circuito que efectué la comparación en magnitud de 2 palabras binarias de un solo bit. Además que pueda ser expandido para comparar palabras de más bits.

Un comparador completo. A diferencia de un medio comparador, es un bloque que puede conectarse en cascada para efectuar comparaciones multibit y que considera el resultado de la comparación de los bits anteriores. Como se muestra en la Figura 6.10.



6.10 a) Medio Comparador, b) Comparador Completo

Como podemos observar en la figura 6.10 b) el bloque tiene 5 entradas y 3 salidas. Para reducir el número de entradas es preferible sustituir las entradas a solamente 2, cuyas combinaciones formen un código que de los 3 valores de comparación, A> B, A = B, A < B. Figura 6.11

A> B	A < B	FUNCION
0	0	A = B
0	1	A < B
1	0	A > B
1	1	can't happen

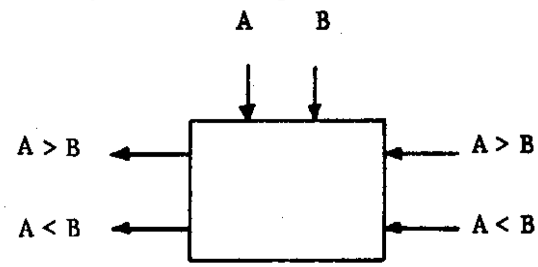
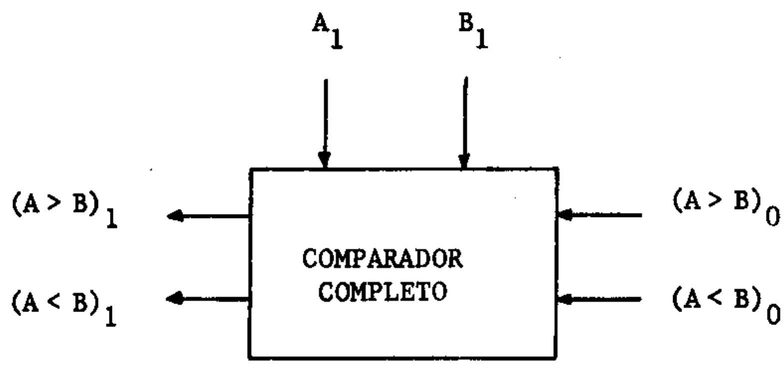


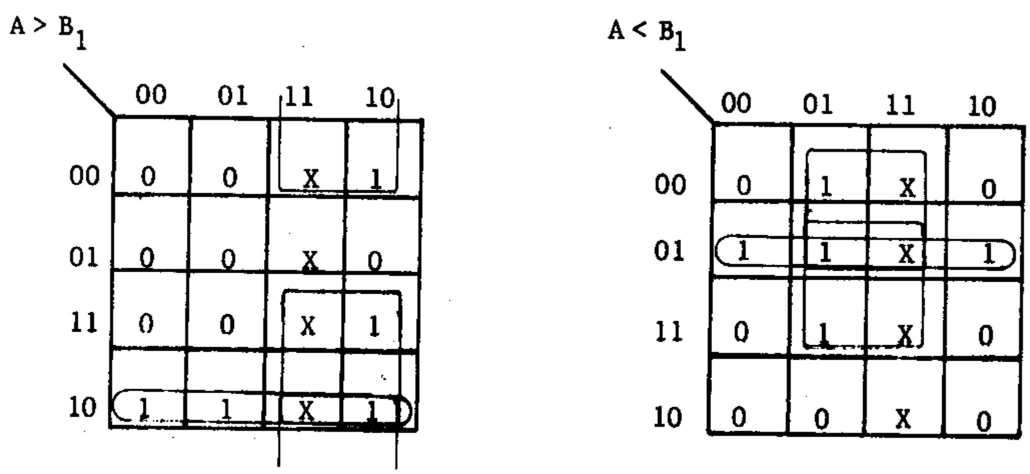
Figura 6.11 Las entradas y salidas a un comparador pueden reducirse.

Entonces el bloque a diseñar queda de la siguiente forma:



Los valores de $(A> B)_o$ y $(A < B)_o$ son el resultado de la comparación del bit anterior.

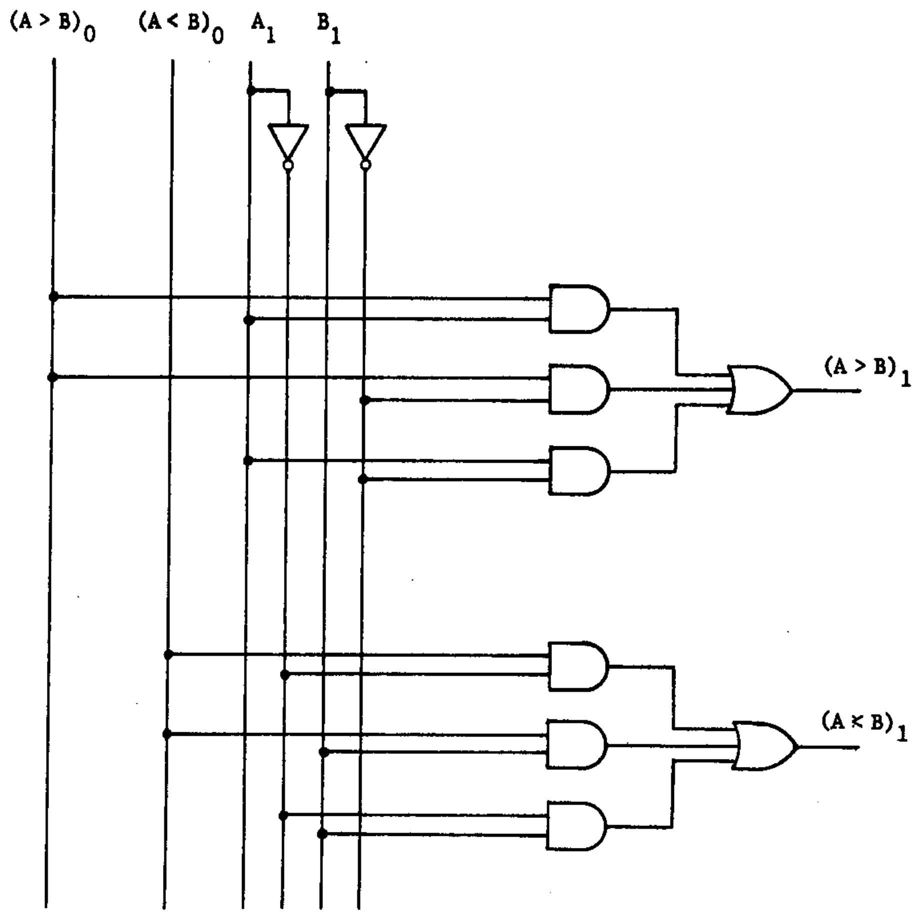
$(A>B)_0$	$(A<B)_0$	A_1	B_1	$(A>B)_1$	$(A<B)_1$
0	0	0	0	0	0
0	0	0	1	0	1
0	0	1	0	1	0
0	0	1	1	0	0
0	1	0	0	0	1
0	1	0	1	0	1
0	1	1	0	1	0
0	1	1	1	0	1
1	0	0	0	1	0
1	0	0	1	0	1
1	0	1	0	1	0
1	0	1	1	1	0
1	1	0	0	X	X
1	1	0	1	X	X
1	1	1	0	X	X
1	1	1	1	X	X



$$(A>B)_1 = (A>B)_0 \cdot \bar{B}_1 + (A>B)_0 \cdot A_1 + A_1\bar{B}_1$$

$$(A<B)_1 = \bar{A}_1 B_1 + (A<B)_0 \cdot \bar{A}_1 + (A<B)_0 \cdot B_1$$

5. Implementación



6.6 Sistemas Combinacionales con salidas múltiples

En los ejemplos de diseño combinacional que hemos tratado en los puntos anteriores de este mismo capítulo, observamos qua existen bloques con varias salidas que se usan en forma simultánea, es decir sistemas constituidos por varias funciones lógicas que dependen de las mismas variables booleanas de entrada. A estos bloques se les da el nombre de sistemas combinacionales con salidas múltiples.

La minimización de una función múltiple se puede efectuar tratando cada una de las funciones en forma independiente como en los ejemplos del punto 6.2, sin embargo no se tiene la completa seguridad de obtener la forma más simple del circuito.

Un método sencillo de minimización consiste en comparar los mapas de Karnaugh de todas las funciones a identificar los grupos que sean comunes a más de una función. Estos términos comunes o repetidos tienen que ser implementados una sola vez. Una de las precauciones que es necesario considerar que la salida de una compuerta puede conectarse a un número limitado de entradas (Fan-out, ver terminología de los circuitos integrados,).

Pasos para la minimización de funciones múltiples por medio del mapa de Karnaugh:

1.- Configurar el mapa de Karnaugh para cada función.

2.- Buscar el grupo más pequeño que aparece en cada uno de los mapas e indicarlo con un círculo.

3.- Continúe el proceso de agrupación partiendo del grupo más pequeño hasta el más grande.

4.- Al seleccionar un grupo considerar su utilidad de acuerdo a que sea común a la mayoría de las funciones.

5.- Seleccionar el mejor juego de grupos comunes a cada función.

Ejemplo 6.8

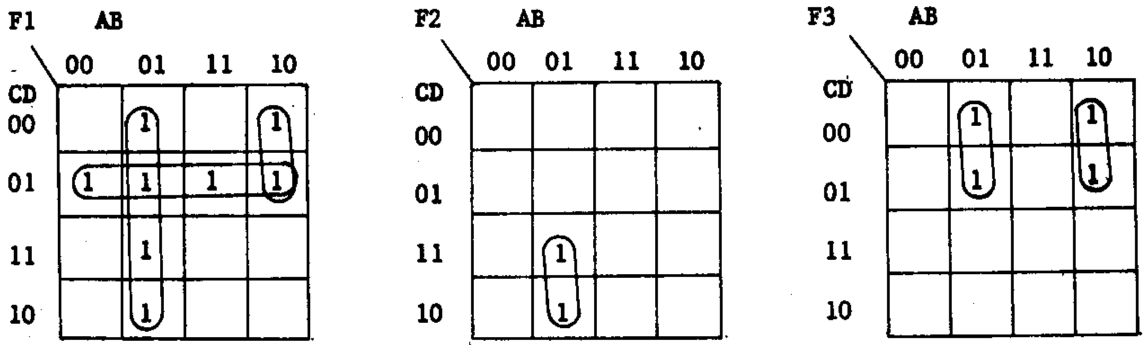
Implemente la función lógica que se presenta a continuación. a) En forma independiente, b) Como una función múltiple.

$F_1(ABCD) = \Sigma\ 1, 4, 5, 6, 7, 8, 9, 13$

$F_2(ABCD) = \Sigma\ 6, 7$

$F_3(ABCD) = \Sigma\ 4, 5, 8, 9$

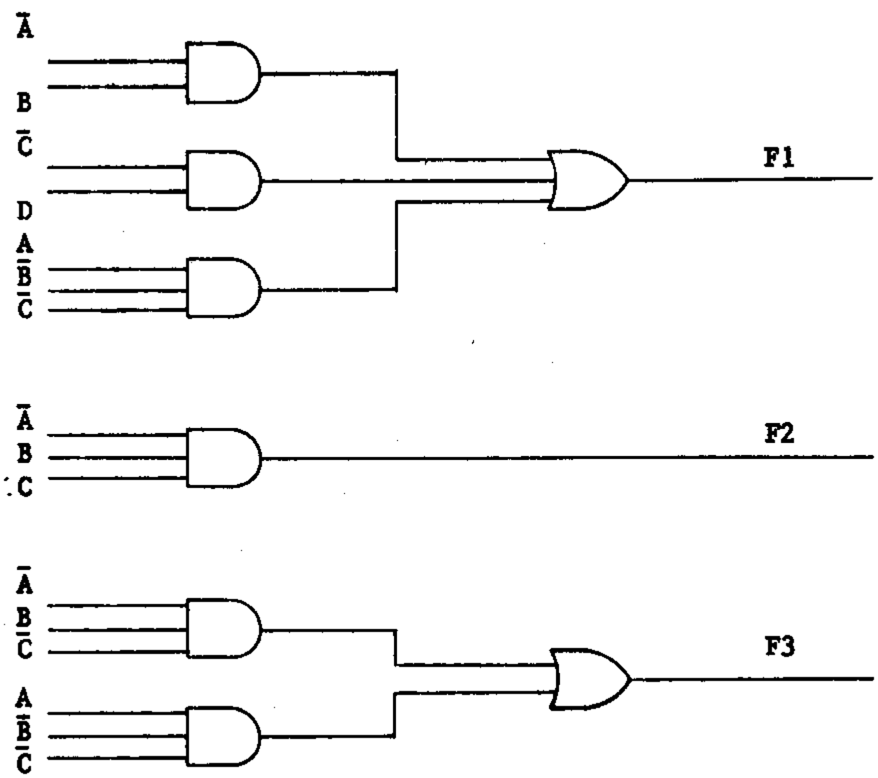
a) En forma independiente



$F1 = \bar{A}B + \bar{C}D + A\bar{B}\bar{C}$

$F2 = \bar{A}BC$

$F3 = \bar{A}B\bar{C} + A\bar{B}\bar{C}$



Sin tomar en cuenta los NOT'S de entrada el costo unitario CU= 21

b) Como una función múltiple:

F1

CD \ AB	00	01	11	10
00		1		1
01	1	1	1	1
11		1		
10		1		

F2

CD \ AB	00	01	11	10
00				
01				
11		1		
10		1		

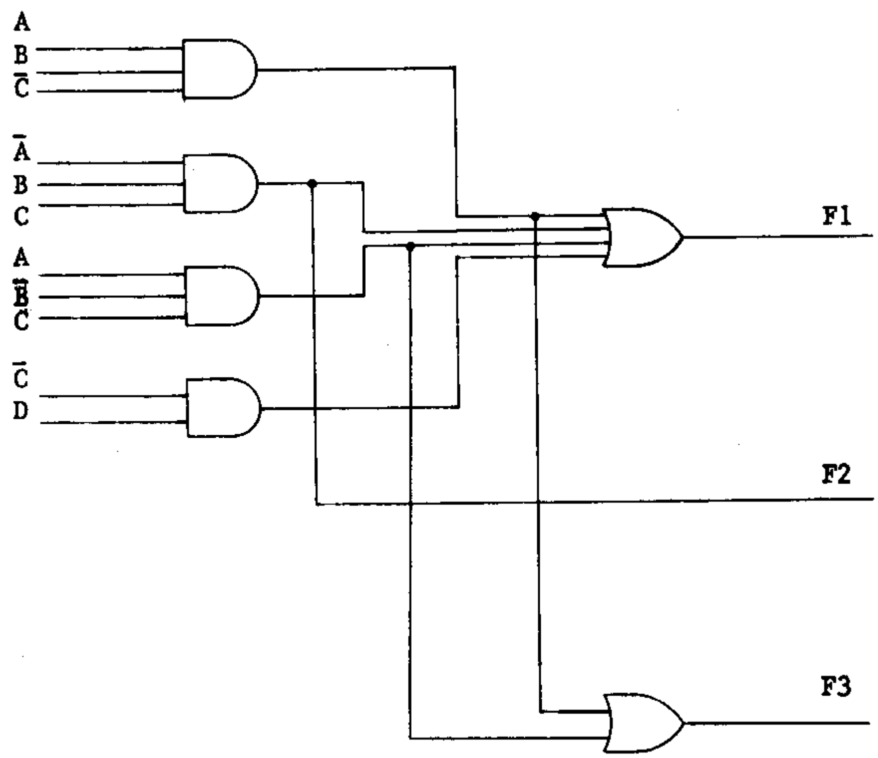
F3

CD \ AB	00	01	11	10
00		1		1
01		1		1
11				
10				

$F1 = \bar{A}B\bar{C} + \bar{A}BC + A\bar{B}\bar{C} + \bar{C}D$

$F2 = \bar{A}BC.*$

$F3 = \bar{A}B\bar{C} + A\bar{B}\bar{C}.*$



Sin tomar en cuenta los Not's de entrada el costo unitario CU= 17

PROBLEMAS PROPUESTOS

1.- Defina los Sistemas Completamente Especificados.

2.-¿Qué es el Don't Care?

3.-¿Qué es el Can't Happen?

4.- Defina los sistemas que no están completamente especificados.

5.-¿Para qué nos sirve un Display?

6.-¿Qué son los Decodificadores?

7.-Diseñe un circuito Combinacional que desarrolle la multiplicación binaria de 2 palabras de 2 bits. La palabra A, (A_1, A_0) y la palabra B,(B_1, B_0). El resultado o salida del circuito marcarlo con la letra M (M... ..M_1 M_o).

8.-Diseñar un circuito combinacional al que lleguen 4 líneas de entrada codificadas en código binario natural y cuya salida esté codificada en BCD.

9.-Detector de errores en BCD. Diseñar un circuito al que lleguen 4 líneas de entrada a indique cuando alguna de las combinaciones no sea BCD.

10.-Cuando se requiera extrema confianza en el control de algún proceso, se usan 2 ó 3 sistemas de control que operen simultáneamente. Tal es el caso de un sistema que opera por triplicado, nuestro interés es que cuando menos 2 de los 3 sistemas operen satisfactoriamente. Por lo tanto se requiere señalizar la confiabilidad del sistema de control, por medio de una sola salida.

11.-En el ejemplo 6.4 el diseño de un decodificador de BCD a 7 segmentos, las combinaciones de 10 a 15 se tomaron como CAN'T HAPPEN.

a) Diseñe el mismo sistema con la variante de que una combinación que no sea BCD muestre una E de ERROR

b) Diseñe el mismo sistema, pero que muestre además los siguientes caracteres A,b,C,d,E,F para las combinaciones del 10 al 15.

12.-a) Diseñe un circuito que tenga 4 entradas I_3, I_2, I_1, I_0 y 2 salidas O_0, O_1. El estado de las salidas mostrara cual línea de entrada tiene en uno lógico, es decir la salida es un código en binario natural correspondiente a cada una de las entradas.

b) Diseñe el mismo circuito tomando en cuenta que pueden presentarse unos en todas las entradas a la vez. En este caso la salida será un código que representa a la entrada de mayor peso. Este circuito recibe el nombre de codificador de prioridad.

13.- Diseñe un circuito combinacional que convierta código Gray de 4 bits a código binario natural.

Bibliografía

1.- Digital Computer Design Fundamentals, Chu Yaohan, Mc Graw Hill, 1962, ISBN 07-010800-5

2.-Introduction to switching circuit theory, Givone Donald, Mc Graw Hill,1970 LCCCN 72-95802

3.- Designing with TTL Integrated Circuits, Morris Robert l., Texas Instruments Incorporated, 1971, ISBN 07-063745-8

4.- Fundamentals of Digital Systems Design, Rhyne, V. Thomas Prentice-Hall, 1973 ISBN-13: 978-0133361568

5.- Sistemas Digitales Principios y Aplicaciones,Tocci R, Prentice Hall, 2004, ISBN 970-26-0297-1

6.- Fundamentos de Sistemas Digitales, T.L. Floyd, Prentice Hall, ISBN z84-205-2994-X

7.- Diseño Digital Principios y Prácticas, John F. Wakerly, Prentice Hall, ISBN 70-17-0404-5

8.- Sistemas Digitales y Electrónica Digital,Garza G. Juan, Prentice Hall, 2006, ISBN 970-26–0719-1

9.- Fundamentos de Diseño Lógico,Charles H. Roth, Jr., THOMSON, ISBN 970-686-373-7