SIMPLIFICACION DE CIRCUITOS LOGICOS :

Una vez que se obtiene la expresión booleana para un circuito lógico, podemos reducirla a una forma más simple que contenga menos términos, la nueva expresión puede utilizarse para implantar un circuito que sea equivalente al original pero que contenga menos compuertas y conexiones.

SIMPLIFICACION ALGEBRAICA.

El álgebra booleana (Algebra de los circuitos lógicos tiene muchas leyes o teoremas muy útiles tales como :

1. Ley de Morgan :

    1. 1. $\overline{A} + \overline{B} = \overline{A \cdot B}$
       2. $\overline{A \cdot B} = \overline{A} + \overline{B}$

2. Ley Distributiva :

    1. 3. $A+(B \cdot C) = (A+B) \cdot (A+C)$
       4. $A \cdot (B+C) = A \cdot B + A \cdot C$

Ademas de las leyes formales para las funciones AND y OR :

1. 5. $A \cdot 0 = 0$; $A+0 = A$
   6. $A \cdot 1 = A$; $A+1 = 1$
   7. $A \cdot A = A$; $A+A = A$
   8. $A \cdot \overline{A} = 0$; $A+\overline{A} = 1$

y la Ley de la Involución:

1. 9. $\overline{A}$(negada) = A

Considerar la expresión booleana $A \cdot \overline{B} + \overline{A} \cdot B + A \cdot B = Y$, un diagrama lógico de ésta

expresión aparece en la Figura 1. Observar que deben utilizarse seis puertas para implementar este circuito lógico, que realiza la lógica detallada en la tabla de verdad (Tabla 1)

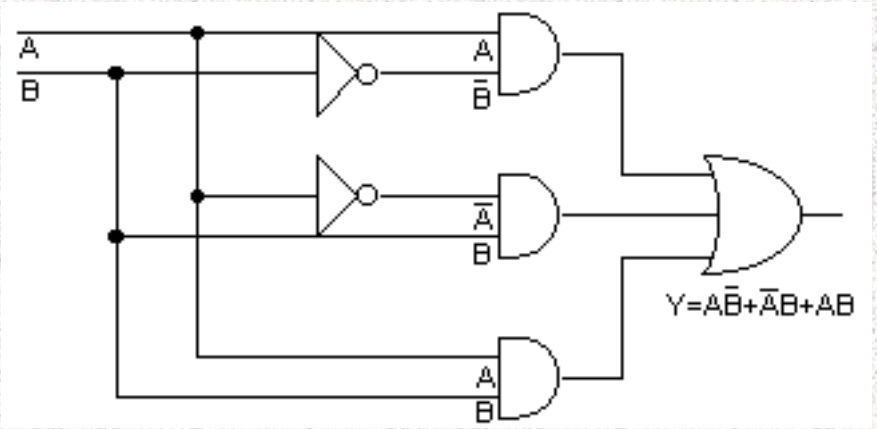


Figura 1: Circuito lógico no simplificado

ENTRADAS		SALIDA
B	**A**	**Y**
0	0	0
0	1	1
1	0	1
1	1	1

Tabla 1: Tabla de verdad de la función OR

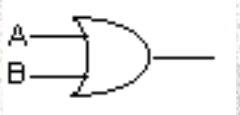


Figura 2: Circuito lógico simplificado

Aplicando el álgebra booleana :

$A \cdot \overline{B} + \overline{A} \cdot B + A \cdot B = Y$

RAZONES

$= A \cdot \overline{B} + (\overline{A} \cdot B + A \cdot B)$	,	Propiedad asociativa
$= A \cdot \overline{B} + B \cdot (\overline{A} + A)$	,	4. [A·(B + C) = A·B + A·C]
$= A \cdot \overline{B} + B \cdot 1$	,	8. [A + $\overline{A}$ = 1]
$= A \cdot B + B$	,	6. [B·1 = B]
$= B + A \cdot \overline{B}$	,	Propiedad conmutativa
$= (B + A) \cdot (B + \overline{B})$	,	3. [A + (B·C) = (A + B)·(A + C)]
$= (B + A) \cdot 1$	,	8. [A + $\overline{A}$ = 1]
$= B + A$	,	6. [A * 1 = A]

Concluimos entonces que una sola puerta OR de dos entradas realiza la misma función (¡ De hecho la tabla 1 corresponde a la función OR !)

EXPRESIONES BOOLENAS EN FORMA DE MINTERMS (SUMA DE PRODUCTOS).

Cuando se comienza un problema de diseño lógico, lo normal es construir primero una tabla de verdad, que detalle la operación exacta del circuito digital. Considerar la tabla de verdad 2, que contiene las variables C, B y y A. Observar que sólo dos combinaciones de variables generan una salida 1. Estas combinaciones se muestran en la lineas octava y segunda (sombreadas) de la tabla de verdad. La linea 2 se lee « una entrada no C Y (AND) una entrada no B Y (AND) una entrada A generan una salida I ». Esto se muestra en la parte derecha de la linea 2 con la expresión booleana $\overline{C} \cdot \overline{B} \cdot A$. La otra combinación de variables que genera un 1 se muestra en la linea 8 de la tabla de verdad. La linea 8 se lee «una entrada C Y (AND) una entrada B Y (AND) una entrada A generan una salida 1». La expresión booleana de la linea 8 aparece a la derecha y es $C \cdot B \cdot A$. Estas dos posible combinaciones se relacionan mediante el operador OR para formar la expresión booleana

completa de la tabla de verdad, que se muestra en la tabla 2, como $C \cdot B \cdot A + \underline{C} \cdot \underline{B} \cdot A = Y$. Esta expresión, a veces, se denomina forma en *suma de productos* de la expresión booleana. Los ingenieros también llaman a esta forma, *forma de minterms*.

Esta expresión puede traducirse al patrón AND-OR de puertas lógicas. El diagrama lógico de la Figura 5.3.c realiza la lógica descrita por la expresión booleana C · B · A + C ·B· A = Y , y genera la tabla de verdad 2.

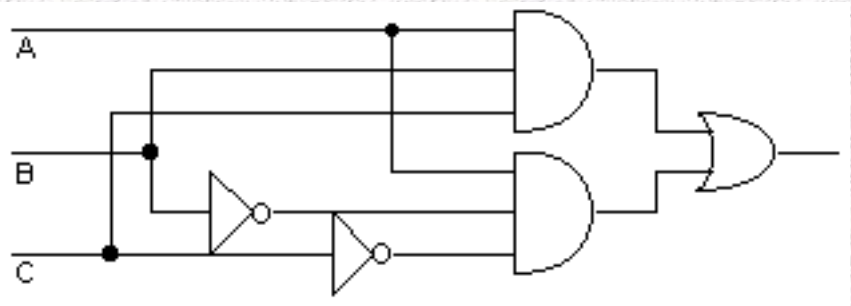


Figura 3: Circuito lógico equivalente AND-OR

ENTRADAS			**SALIDAS**	
C	**B**	**A**	**Y**	
0	0	0	0	
0	0	1	1	C·B·A
0	1	0	0	
0	1	1	0	
1	0	0	0	
1	0	1	0	
1	1	0	0	
1	1	1	1	C·B·A
C·B·A + C·B·A = Y				

Tabla 2: Expresión booleana

El procedimiento típico que se sigue en el trabajo de diseño lógico consiste en construir *primero* una tabla de verdad. *A continuación*, determinar una expresión booleana en forma de minterms a partir de la tabla de verdad. *Finalmente*, dibujar el circuito lógico AND-OR a partir de la expresión booleana en minterms.

EXPRESIONES BOOLENAS EN FORMA DE MAXTERMS (PRODUCTO DE SUMAS).

Considerar la tabla de verdad 3. La expresión booleana para esta tabla de verdad puede escribirse de dos formas, cómo se observó en la sección introductoria. La expresión booleana en minterms se obtiene de las salidas que son 1 en la tabla de verdad. Cada 1 en la columna de salida se convierte en un termino, que se relaciona con los demás, mediante el operador OR, en la expresión en forma de minterms. La expresión en minterms para esta tabla de verdad se da en la tabla 3, como :

$$B \cdot A + B \cdot \overline{A} + \overline{B} \cdot A = Y$$

(a) Expresión booleana en forma de maxterms : B + A = Y

TABLA DE VERDAD OR		
ENTRADAS		**SALIDA**
B	**A**	**Y**
0	0	0
0	1	1 -> $\overline{B} \cdot A$
1	0	1 -> $B \cdot \overline{A}$
1	1	1 -> $B \cdot A$
Expresión: $B \cdot A + B \cdot \overline{A} + \overline{B} \cdot A = Y$		

Tabla 3: Expresión booleana en forma de maxterms

La tabla de verdad 3 también puede describirse utilizando una expresión booleana en *forma de maxterms*. Este tipo de expresión se desarrolla a partir de los 0 de la columna de salida de la tabla de verdad. Por cada 0 de la columna de salida se realiza una operación OR. Observar que las *variables de entrada se invierten y después se realiza la operación OR*. La expresión booleana en maxterms de esta tabla de verdad aparece en la tabla 3. La expresión en maxterms para la tabla de verdad OR es *B + A = Y*. Esto significa lo mismo que la familiar expresión OR: *A + B = Y*. Para la tabla de verdad 3, la expresión booleana en

maxterms es la más simple, aunque ambas formas describen con precisión la lógica de dicha tabla de verdad.

ENTRADAS			SALIDA	
C	B	A	Y	
0	0	0	1	
0	0	1	1	
1	1	0	1	
0	1	1	1	
1	0	0	0	C+B+A
1	0	1	1	
1	1	0	1	
1	1	1	0	C+B+A
(C+B+A)·(C+B+A) = Y				

Tabla 4: Expresión booleana en Maxterms.

Considerar la tabla de verdad 4. La expresión en minterms para esta tabla es demasiado larga. La expresión booleana en *maxterms* se obtiene a partir de las variables de las lineas 5 y 8. Cada una de estas lineas tiene un 0 en la columna de salida. Las variables se invierten y se relacionan con operadores OR. Los términos así obtenidos se ponen entre paréntesis y se relacionan con operadores AND. La expresión booleana completa, en forma de maxterms, se da en la tabla 4, y también se la denomina *forma de producto de sumas* de la expresión booleana. El termino producto de sumas viene de la organización de los símbolos de suma (+) y producto (·).

Una expresión booleana en maxterms se implementa utilizando el patrón OR-AND de puertas lógicas según indica la figura 4. Observar que las salidas de las dos puertas OR están alimentando una puerta AND. La expresión en maxterms *(C + B + A) * (C + B + A) = Y*

, se implementa utilizando el patrón OR-AND de puertas lógicas de la Figura 4.

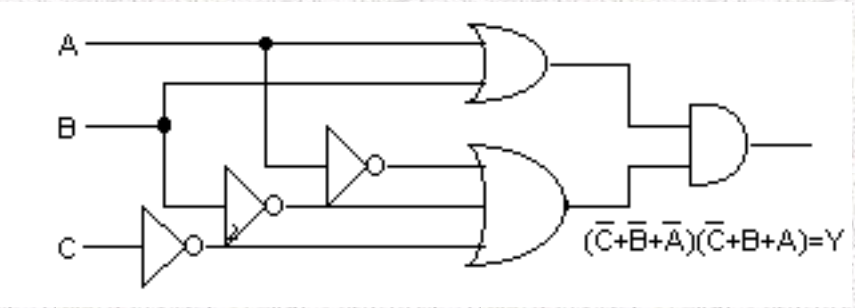


Figura 4:Expresión en forma de maxterms

Aplicando el álgebra booleana podemos pasar expresiones en forma de minterms a maxterms y viceversa. Ejemplo: Pasar la expresión booleana en forma de maxterms,

$$Y = (C + B + A)\cdot(C + \underline{B} + A)\cdot(C + \underline{B} + \underline{A})\cdot(\underline{C} + B + A)\cdot(\underline{C} + B + \underline{A})\cdot(\underline{C} + \underline{B} + A)$$

a su correspondiente en forma de mimterms, $Y = \underline{C}\cdot\underline{B}\cdot A + C\cdot B\cdot A$

tenemos :

$$Y = (C + B + A)\cdot(C + \underline{B} + A)\cdot(C + \underline{B} + \underline{A})\cdot(\underline{C} + B + A)\cdot(\underline{C} + B + \underline{A})\cdot(\underline{C} + \underline{B} + A)$$

$$= [(C + B + A)\cdot(C + \underline{B} + A)]\cdot[(C + \underline{B} + \underline{A})\cdot(\underline{C} + B + \underline{A})]\cdot[(\underline{C} + B + A)\cdot(\underline{C} + \underline{B} + A)],\ \text{Propiedad asociativa y conmutativa}$$

$$= \{[(C + A) + B]\cdot[(C + A) + \underline{B}]\}\cdot\{[(C + \underline{B}) + \underline{A}]\cdot[(\underline{C} + B) + \underline{A}]\}\cdot\{[(\underline{C} + A) + B]\cdot[(\underline{C} + A) + \underline{B}]\},\ \text{Propiedad asociativa y conmutativa.}$$

$$= [(C + A) + B\cdot\underline{B}]\cdot[(C + \underline{B})\cdot(\underline{C} + B) + \underline{A}]\cdot[(\underline{C} + A) + B\cdot\underline{B}] - - - - ,\ [A + (B\cdot C) = (A + B)\cdot(A + C)]$$

$$(C + A)\cdot[(C + \underline{B})\cdot(\underline{C} + B) + \underline{A}]\cdot(\underline{C} + A) - - - ,\ \ [A\cdot\underline{A} = 0]\ y\ [A + 0 =$$

A]

(C + A)·(C̲ + A)·[(C + B̲)·(C̲ + B) + A̲] - - - , Propiedad conmutativa

(C·C̲ + A)·[(C + B̲)·C̲ + (C + B̲)·B + A̲] , [A + (B·C) = (A + B)(A + C)], [A·(B+C) = A·B + A·C]

A·[C̲·C + C̲·B̲ + C·B + B·B̲ + A̲], [A·A̲ = 0],[A·(B + C) = A·B + A·C] y [A + 0 = A]

A·[C̲·B̲ + C·B + A̲] - - - - - - - - - - - ,[A·A̲ = 0] y [A + 0 = A]

A·C̲·B̲ + A·C·B + A·A̲] - - - - - , [A·(B + C) = A·B + A·C]

A·C̲·B̲ + A·C·B - - - - - - - - - - , [A·A̲ = 0] y [A + 0 = A]

C̲·B̲·A + C·B·A - - - - - - - - , Propiedad conmutativa

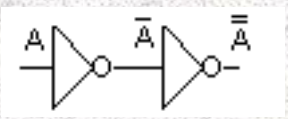
Otra forma de pasar una expresión booleana en forma de minterms a maxterms y viceversa es utilizando únicamente el teorema de D'Morgan. El ejemplo anterior quedaría :

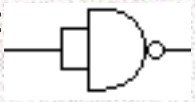
Y = (C + B + A)·(C + B̲ + A)·(C + B̲ + A̲)·(C̲ + B + A)·(C̲ + B + A̲)·(C̲ + B̲ + A)

= (C̲·B̲·A̲)·(C̲·B·A̲)·(C̲·B̲·A̲)·(C̲·B·A)·(C̲·B·A̲)·(C̲·B̲·A̲),

= C̲·B̲·A̲ + C̲·B·A̲ + C̲·B·A + C·B̲·A̲ + C·B̲·A + C·B·A,

UTILIZACION DE LA LOGICA NAND Y NOR.

La lógica NAND y NOR se utiliza para simplificar circuitos compuestos, por puertos AND, OR y NOT, en circuitos compuestos únicamente por puertas NAND o únicamente por puertas NOR. Esta lógica se fundamenta en la ley de la Involución (A̲ = A), la cual puede representarse por : 

, teniendo en cuenta que una puerta NOT es equivalente a : 

la lógica NAND se utiliza para simplificar circuitos AND-OR como se ilustra en el siguiente ejemplo :

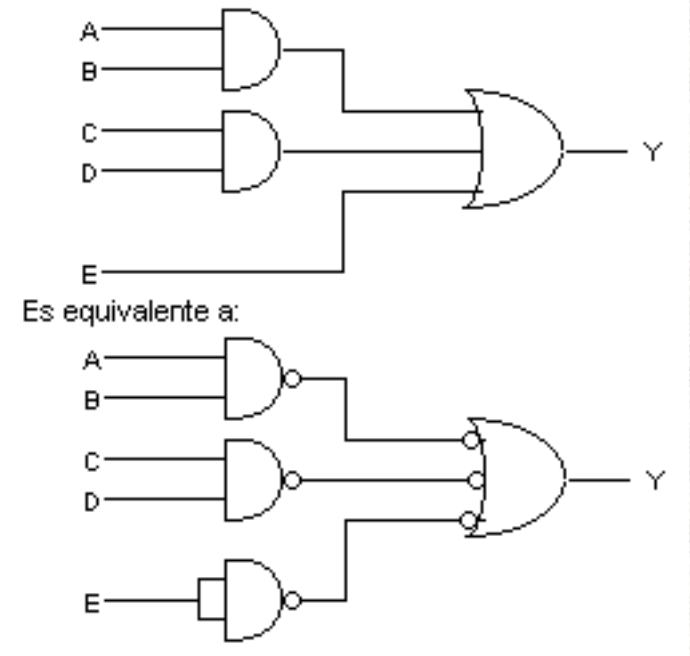


Figura 5: Circuito lógico NAND

Observar que negamos las entradas de la puerta OR, al igual que las salidas de las puertas AND (1 y 2). Dado que la linea E solo se negó una sola vez (A la entrada de la puerta OR), la negamos otra vez con una puerta NOT, para que el circuito no se altere, y teniendo en cuenta la ley de la Involución; es decir E = E.

De manera similar la lógica NOR se utiliza para simplificar circuitos OR-AND como se ilustra en el siguiente ejemplo :

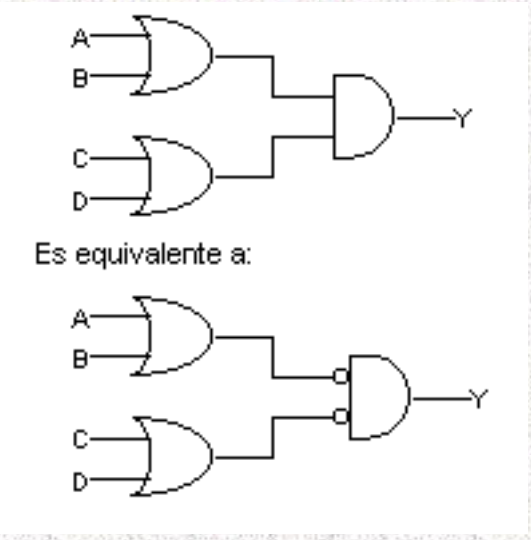
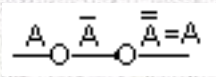


Figura 6: Circuito lógico NOR

Observar que tanto para la utilización de la lógica NAND como para la NOR, sobre cualquier linea se niega dos veces : 

,lo cual es consistente con la ley de la Involución.

DIAGRAMAS DE KARNAUGH

Es un metodo grafico que se utiliza para simplificar circuitos logicos en un proceso simple y ordenado. Es metodo que se basa en los teoremas booleanos estudiados anteriormente y su utilidad practica se limita a 5 variables. Las reglas a seguir son las siguientes:

1. A partir de la tabla de verdad sacar las expresiones booleanas en forma de minterns o maxterms.
2. Colocar los 1 corespondientes en el diagrama por cada grupo de variables operadas por AND si es en forma de minterns u operadas por OR si es en forma de maxterms.
3. Agrupar los 1 adyacentes (las agrupaciones se realizan en grupos de 2, 4, 8 1)
4. Eliminar las variables que aparezcan con su complemento.
5. Enlazamos con OR los resultados obtenidos (si es en forma de minterns) o con AND (si es en forma de maxterms).

Tomemos la tabla de verdad 5. Lo primero que debemos hacer es sacar las expresiones booleanas correspondientes:

A	B	Q	
0	0	0	
0	0	1	$\underline{A}\cdot B$
1	0	1	$A\cdot\underline{B}$
1	1	1	$A\cdot B$
$Q=(\underline{A}\cdot B)+(A\cdot\underline{B})+(A\cdot B)$			

Tabla 5

Luego procedemos a colocar cada 1 correspondiente en el diagrama por cada grupo de variables operadas con AND (para nuestro ejemplo). Los diagramas de Karnaugh pueden presentarse de dos maneras diferentes: la americana y la alemana, demos un vistazo a dichas presentaciones:

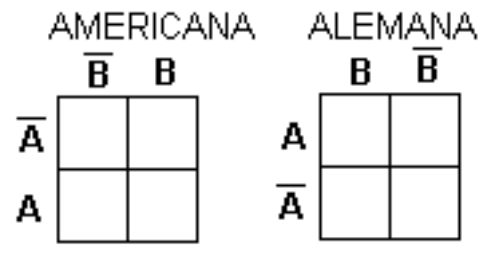


Figura 7: Diagramas de Karnaugh para 2 variables

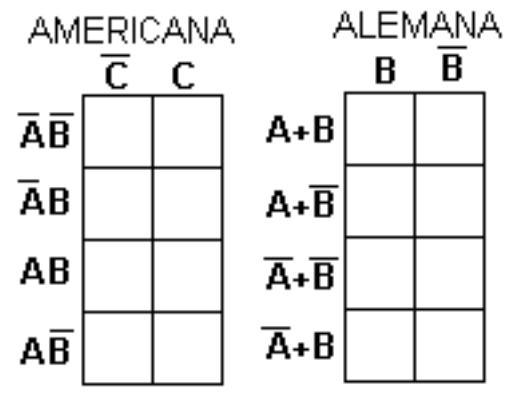


Figura 8: Diagramas de Karnaugh para 3 variables

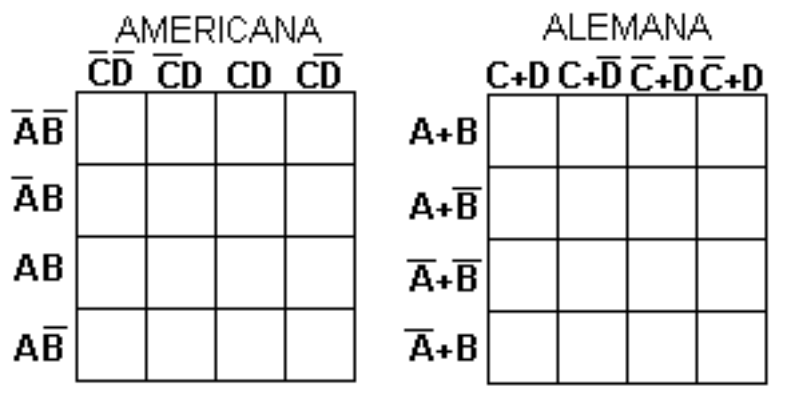


Figura 9: Diagramas de Karnaugh para 4 variables

Ahora que conocemos las maneras en que se pueden presentar las diagramas procedemos a colocar los 1 correspondientes por cada grupo de variables operadas con AND (en nuestro ejemplo)

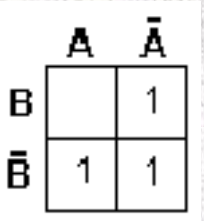


Figura 10: Colocación de los unos en el mapa de Karnaugh

Luego procedemos a agrupar los 1 adyacentes que se encuentren en el diagrama, estas agrupaciones se realizan en grupos de 2, 4, o de 8 "1" . Debemos tratar en lo posible de no realizar tantas agrupaciones.

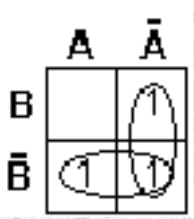


Figura 11: Agrupación de términos

Despues de realizar las agrupaciones eliminanos por cada grupo las variables que aparezcan con su complemento. En el agrupamiento de 2 "1" se elimina una variable; en el agrupamiento de 4 "1" se eliminan 2 variables y en el agrupamiento de 8 "1" se eliminan 3 variables.

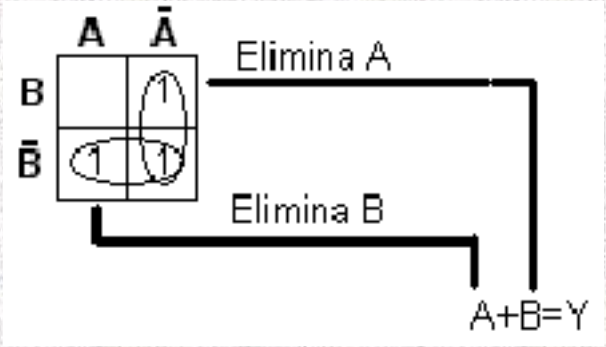


Figura 12: Eliminación de términos

Por ultimo enlazamos con OR (ya que nuestro ejemplo es en forma de minterns) los resultados que obtuvimos de la eliminacion de variables.

$$Q = A + B$$

De esta manera la ecuacion logica $Q=(\underline{A}\cdot B)+(A\cdot B)+(A\cdot \underline{B})$ nos quedaría reducida a una puerta OR

DIAGRAMAS DE KARNAUGH CON 5 VARIABLES

Para realizar simplificaciones con 5 variables se utilizan los llamados diagramas bidimensionales, en donde un plano nos indica la quinta variable y el otro plano su complemento, veamos:

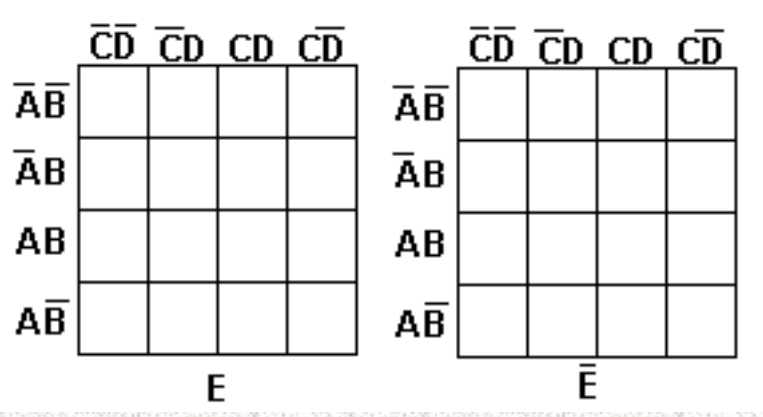


Figura 13: Diagrama de Karnaugh para 5 variables

Realicemos un ejercicio para asimilar la simplificacion con 5 variables. Tomemos la siguiente tabla de verdad:

A	B	C	D	E	Q
0	0	0	0	0	0
0	0	0	0	1	0
0	0	0	1	0	1
0	0	0	1	1	1
0	0	1	0	0	0
0	0	1	0	1	0
0	0	1	1	0	0
0	0	1	1	1	0
0	1	0	0	0	0
0	1	0	0	1	0
0	1	0	1	0	1
0	1	0	1	1	1
0	1	1	0	0	0
0	1	1	0	1	0
0	1	1	1	0	0
0	1	1	1	1	0

1	0	0	0	0	1
1	0	0	0	1	1
1	0	0	1	0	0
1	0	0	1	1	0
1	0	1	0	0	0
1	0	1	0	1	0
1	0	1	1	0	1
1	0	1	1	1	0
1	1	0	0	0	0
1	1	0	0	1	0
1	1	0	1	0	0
1	1	0	1	1	0
1	1	1	0	0	0
1	1	1	0	1	0
1	1	1	1	0	0
1	1	1	1	1	0

Tabla 6: Tabla de verdad de cinco variables

Luego procedemos a sacar la ecuacion no simplificada

$$Q = \overline{A}\overline{B}\overline{C}\overline{D}\overline{E} + \overline{A}\overline{B}\overline{C}\overline{D}E + \overline{A}\overline{B}\overline{C}D\overline{E} + \overline{A}\overline{B}\overline{C}DE + \overline{A}B\overline{C}\overline{D}\overline{E} + \overline{A}B\overline{C}\overline{D}E + \overline{A}B\overline{C}D\overline{E}$$

Despues que obtenemos la ecuacion no simplificada pasamos los 1 correspondientes al diagrama y realizamos las agrupaciones. Si existen agrupaciones que ocupan el mismo lugar en ambos planos, los reflejamos para obtener una ecuación más simplificada. El proceso de simplificación es el mismo que utilizamos anteriormente.

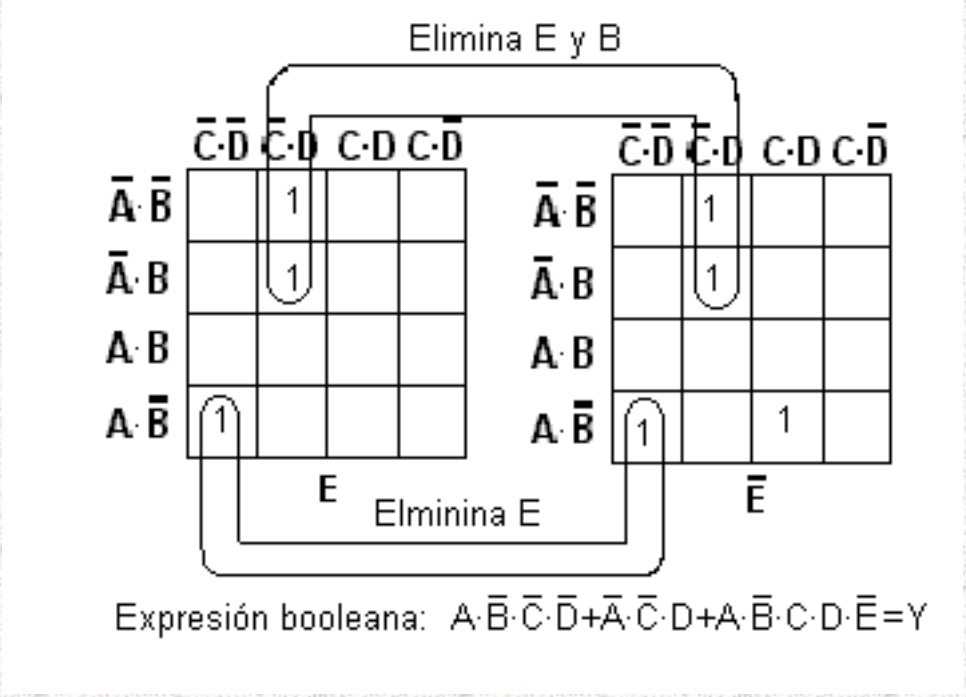


Figura 14: Simplificación de diagramas de Karnaugh de 5 variables

De esta manera obtenemos la siguiente ecuación:

$$Q = ABCD + ACD + ABCDE$$

CONDICIONES NO IMPORTA

En muchos circuitos logicos hay condiciones de entrada para las que no se especifican los niveles de salida,en la mayoria de los casos es por que estas condiciones nunca se presentaran o simplemente el nivel logico de la salida es irrelevante.

A	B	C	Q	
0	0	0	0	
0	0	1	0	
0	1	0	0	
0	1	1	X	

1	0	0	X	
1	0	1	1	A·B·C
1	1	0	1	A·B·C
1	1	1	1	A·B·C

Tabla 7

En la tabla de verdad no se especifica el nivel de salida para las condiciones "0,1,1" y "1,0,0". En su lugar se coloca una x que representa la condicion no importa. La persona que este realizando la simplificacion tiene la libertad de determinar el nivel logico para la salida de la condicion "no importa", con el fin de producir la expresion mas simple. Realicemos la simplificacion:

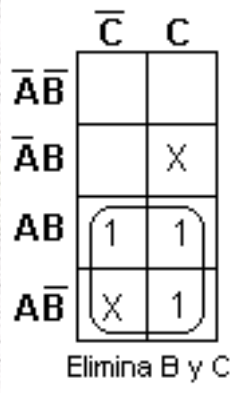


Figura 15: Simplificación de diagramas de Karnaugh con condiciones "no importa"

de esta manera obtenemos que: Q = A.

En muchos casos se trabaja con el código BCD, sabemos que en este codigo existen 6 cobinaciones que son prohibidas (1010,1011,1101, 1110,1111), estas condiciones tambien son llamadas condiciones no importa.

8	4	2	1	Q

0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	1
1	0	1	0	X
1	0	1	1	X
1	1	0	0	X
1	1	0	1	X
1	1	1	0	X
1	1	1	1	X

Tabla 8: Términos irrelevantes en los números BCD

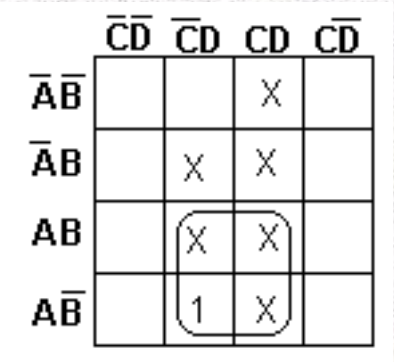


Figura 16: Simplificación

NÚMEROS UTILIZADOS EN ELECTRÓNICA DIGITAL

Los sistemas de numeración utilizados en electrónica digital son los siguientes: sistema decimal, sistema binario, sistema octal y sistema hexadecimal

SISTEMA DECIMAL

Este sistema consta de diez símbolos que van desde el numero 0 hasta el numero 9, los cuales le dan la característica principal a este sistema conocido por todo el mundo. Estos símbolos numéricos también forman unidades numéricas compuestas, al tomarlos como exponentes de un número que se encargará de regular el procedimiento, este número es llamado base. El numero base va a ser 10, por tal motivo también es conocido como "sistema de numeración en base 10".

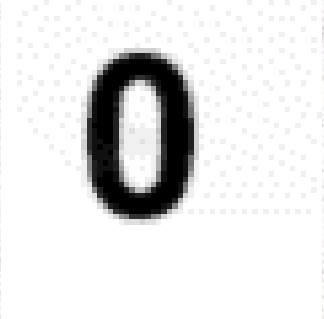


Figura 1: Sistema decimal

SISTEMAS DE NÚMEROS BINARIOS

Figura 2: Sistema de numeros binarios

Este es el sistema numérico que utilizan los sistemas digitales para contar y es el código al que traduce todas las informaciones que recibe. Se dice "Binario" a todo aquello que tiene dos partes, dos aspectos, etc. Muchas cosas en los sistemas digitales son binarias: Los

impulsos eléctricos que circulan en los circuitos son de baja o de alta tensión, los interruptores biestables están encendidos o apagados, abiertos o cerrados, etc. A diferencia del sistema decimal al que estamos habituados, y que utiliza diez cifras, del 0 al 9, el sistema numérico binario utiliza solo dos cifras, el 0 y el 1. En el sistema binario las columnas no representan la unidad, la decena, la centena, como en el sistema decimal, sino la unidad (2^0), el doble (2^1), el doble (2^2), etc. De modo que al sumar en la misma columna 1 y 1, dará como resultado 0, llevándonos 1 a la columna inmediatamente a la izquierda. Para los sistemas digitales es fácil, hasta el punto que reduce todas las operaciones a sumas y restas de números binarios.

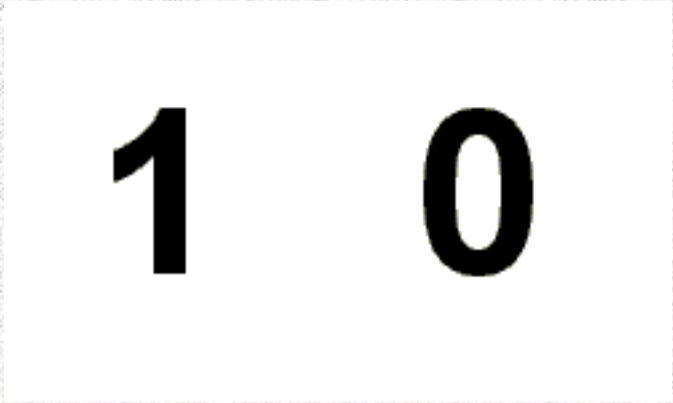


Figura 3: Sistema binario

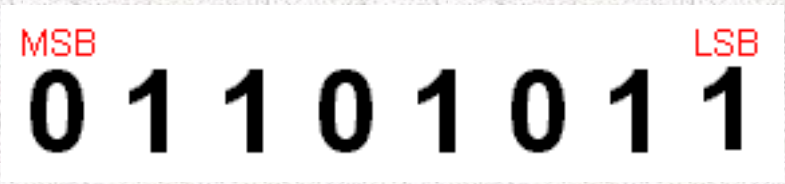


Figura 4: Números binarios

También las palabras, los números y los dibujos se traducen en el ordenador en secuencias de 1 y 0. De hecho toda letra, cifra o símbolo gráfico es codificado en una secuencia de 0 y 1. Si, por ejemplo, nuestro nombre tiene cinco letras, la representación para el ordenador constara de cinco bytes. La palabra bit deriva de las dos palabras inglesas "binary digit" cifra binaria, y designa a las dos cifras 0 y 1, que se utilizan en el sistema binario. Un bit es también, la porción más pequeña de información representable mediante un número, e indica si una cosa es verdadera o falsa, alta o baja, negra o blanca, etc.
Un byte es generalmente una secuencia de 8 bits. Ocho ceros y unos se pueden ordenar de 256 maneras diferentes ya que cada bit tiene un valor de posición diferente, donde el bit

numero 1 le corresponderá un valor de posición de 2^0(1), el siguiente bit tendrá un valor de 2^1(2), el siguiente 2^2(4), el siguiente 2^3(8), el siguiente 2^4(16), el siguiente un valor de 2^5(32), y así sucesivamente hasta llegar la ultima posición, o ultimo bit, en este caso el numero 8, que también es llamado el MSB (Bit Mas Significativo) y el LSB (Bit Menos Significativo) correspondiente a la primera posición o bit numero 1. Ejemplo:

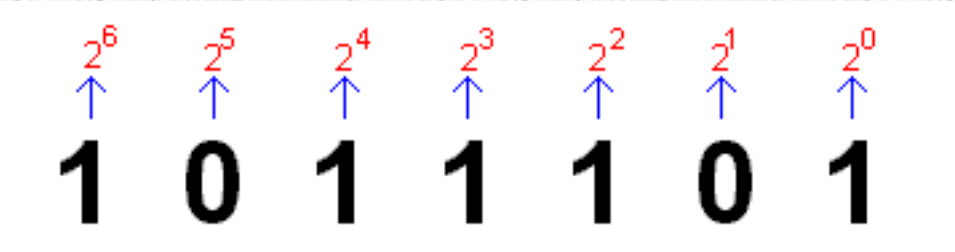


Figura 5: Valores de las posiciones de los números binarios

SISTEMA DE NUMERACIÓN OCTAL

Figura 6: Sistema de numeración octal

Este sistema consta de 8 símbolos desde el 0 hasta el 7, es muy poco utilizado en los computadores. La facilidad con que se pueden convertir entre el sistema Octal y el binario hace que el sistema Octal sea atractivo como un medio "taquigráfico" de expresión de números binarios grandes. Cuando trabajamos con una gran cantidad de números binarios de muchos bits, es mas adecuado y eficaz escribirlos en octal y no en binarios. sin embargo, recordemos los circuitos y sistemas digitales trabajan eléctricamente en binario, usamos el sistema Octal solo por conveniencia con los operadores del sistema

SISTEMA DE NUMERACIÓN HEXADECIMAL

Este sistema consta de 16 símbolos donde desde el 0 hasta el 9 son números y del 10 hasta el 15 son letras, las cuales se encuentran distribuidas en la siguiente forma:

Hexadecimal	Decimal	Hexadecimal	Decimal
0	0	8	8
1	1	9	9
2	2	A	10
3	3	B	11
4	4	C	12
5	5	D	13
6	6	E	14
7	7	F	15

Tabla 1: Símbolos utilizados en el sistema de numeración hexadecimal

La ventaja principal de este sistema de numeración es que se utiliza para convertir directamente números binarios de 4 bits. En donde un solo dígito hexadecimal puede representar 4 números binarios o 4 bits.

CONVERSIONES DE SISTEMAS DE NUMERACIÓN

CONVERSIÓN DE UN NUMERO DECIMAL A BINARIO

Para esta transformación es necesario tener en cuenta los pasos que mostraremos en el siguiente ejemplo: Transformemos el numero 42 a numero binario

1. Dividimos el numero 42 entre 2
2. Dividimos el cociente obtenido por 2 y repetimos el mismo procedimiento hasta que el cociente sea 1.
3. El numero binario lo formamos tomando el primer dígito el ultimo cociente, seguidos por los residuos obtenidos en cada división, seleccionándolos de derecha a izquierda, como se muestra en el siguiente esquema.

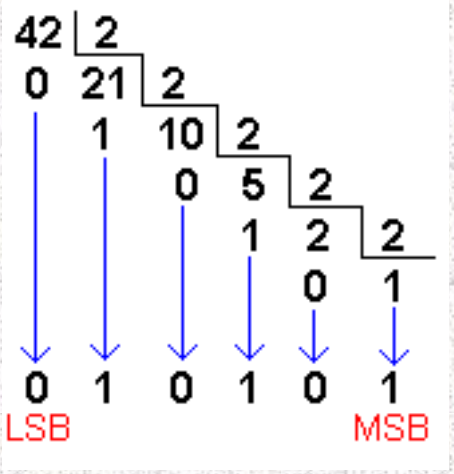


Figura 7: Conversión de decimal a binario

CONVERSIÓN DE UN NUMERO DECIMAL FRACCIONARIO A UN NUMERO BINARIO

Para transformar un número decimal fraccionario a un numero binario debemos seguir los pasos que mostramos en el siguiente ejemplo: transformemos el numero 42,375.

1. la parte entera se transforma de igual forma que el ejemplo anterior.
2. La parte fraccionaria de la siguiente manera:

    - Multiplicamos por el numero 2 y tomamos la parte entera del producto que ira formando el numero binario correspondiente
    - Tomamos nuevamente la parte entera del producto, y la parte fraccionaria la multiplicamos sucesivamente por 2 hasta llegar a 0
    - Tomamos nuevamente la parte entera , y como la parte fraccionaria es 0, indica que se ha terminado el proceso .El numero binario correspondiente a la parte decimal será la unión de todas las partes enteras, tomadas de las multiplicaciones sucesivas realizadas durante el transcurso del proceso , en donde el primer dígito binario corresponde a la primera parte entera , el segundo dígito a la segunda parte entera , y así sucesivamente hasta llegar al ultimo .Luego tomamos el numero binario , correspondiente a la parte entera , y el numero binario , correspondiente a la parte fraccionaria y lo unimos en un solo numero binario correspondiente a el numero decimal.

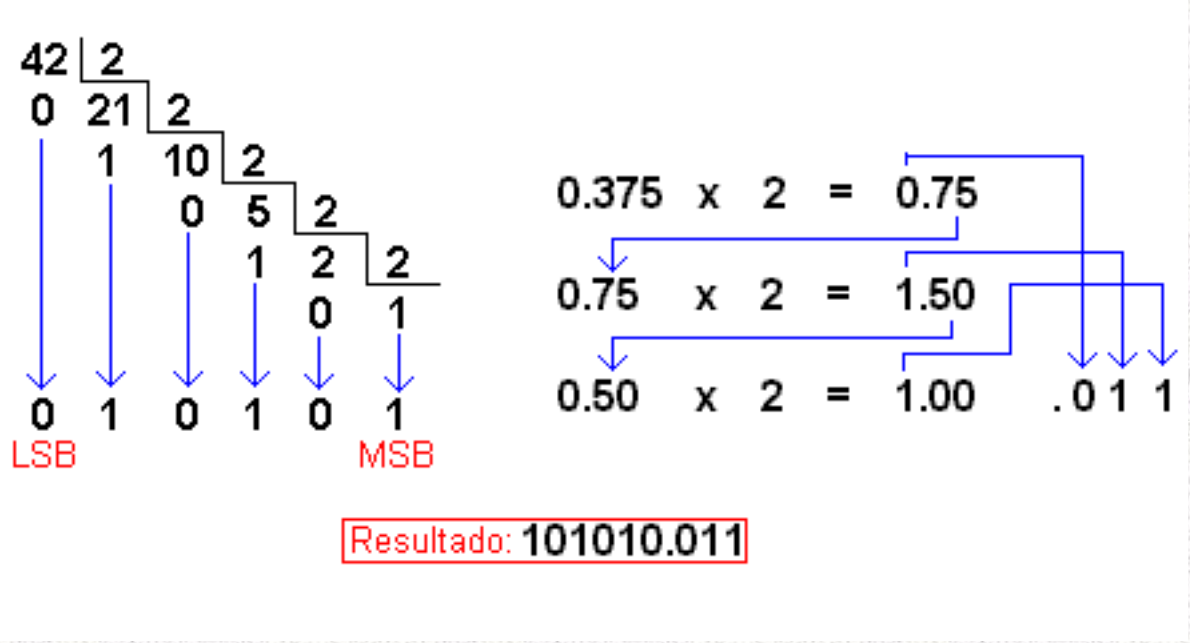


Figura 8: Conversión de decimal fraccionario a binario

CONVERSIÓN DE UN NUMERO BINARIO A UN NUMERO DECIMAL

Para convertir un número binario a decimal, realizamos los siguientes pasos:

1. Tomamos los valores de posición correspondiente a las columnas donde aparezcan únicamente unos
2. Sumamos los valores de posición para identificar el numero decimal equivalente

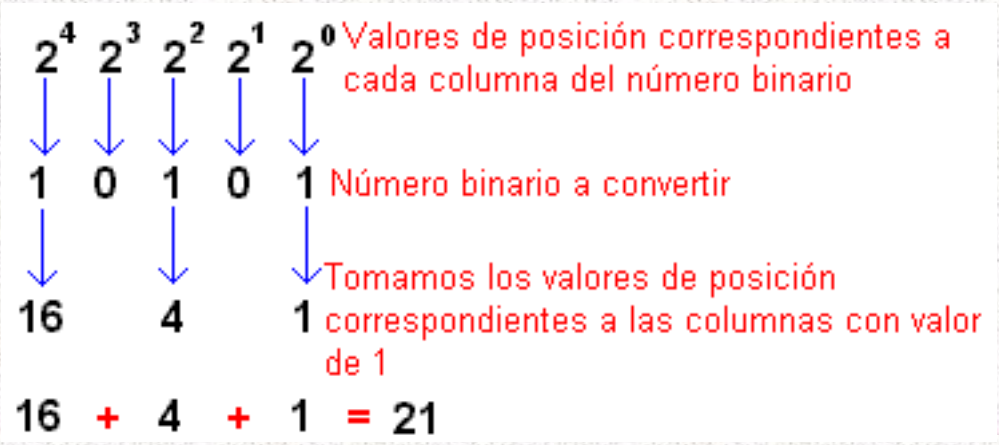


Figura 9: Conversión de binario a decimal

CONVERSIÓN DE UN NUMERO DECIMAL A OCTAL

Para convertir un numero en el sistema decimal al sistema de numeración Octal, debemos seguir los pasos que mostraremos en el siguiente ejemplo Convertir el numero decimal 323.625 a el sistema de numeración Octal

1. Se toma el numero entero y se divide entre 8 repetidamente hasta que el dividendo sea menor que el divisor, para colocar entonces el numero 0 y pasar el dividendo a formar el primer dígito del numero equivalente en decimal
2. Se toma la parte fraccionaria del numero decimal y la multiplicamos por 8 sucesivamente hasta que el producto no tenga números fraccionarios
3. Pasamos la parte entera del producto a formar el dígito correspondiente
4. Al igual que los demás sistemas , el numero equivalente en el sistema decimal , esta formado por la unión del numero entero equivalente y el numero fraccionario equivalente.

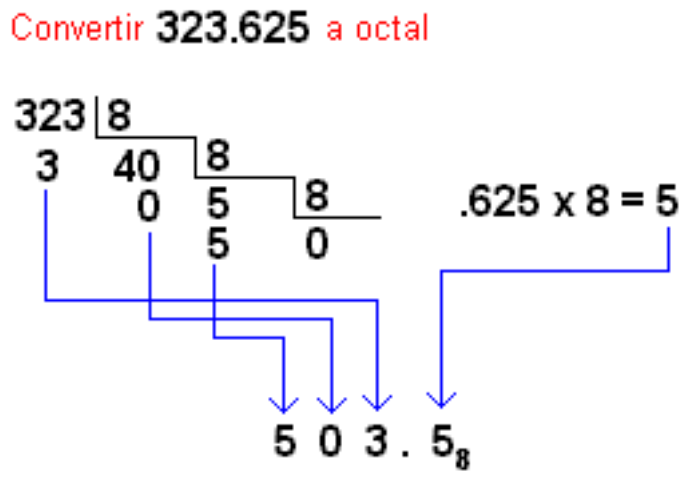


Figura 10: Conversión de decimal a octal

CONVERSIÓN DE UN NUMERO OCTAL A BINARIO

La ventaja principal del sistema de numeración Octal es la facilidad conque pueden realizarse la conversión entre un numero binario y octal. A continuación mostraremos un ejercicio que ilustrará la teoría. Por medio de este tipo de conversiones, cualquier numero Octal se convierte a binario de manera individual. En este ejemplo, mostramos claramente el equivalente **100 111 010** en binario de cada numero octal de forma individual.

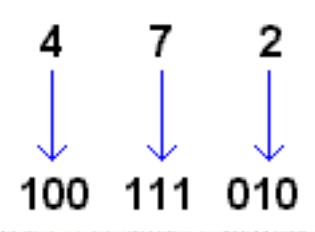


Figura 11: Conversión de octal a binario

CONVERSIÓN DE UN NUMERO DECIMAL A UN NUMERO HEXADECIMAL

Convertir el numero 250.25 a Hexadecimal

1. Se toma la parte entera y se divide sucesivamente por el numero decimal 16 (base) hasta que el cociente sea 0
2. Los números enteros resultantes de los cocientes, pasarán a conformar el numero hexadecimal correspondiente, teniendo en cuenta que el sistema de numeración hexadecimal posee solo 16 símbolos, donde los números del 10 hasta el 15 tienen símbolos alfabéticos que ya hemos explicado
3. La parte fraccionaria del numero a convertir se multiplica por 16 (Base) sucesivamente hasta que el producto resultante no tenga parte fraccionaria
4. Al igual que en los sistemas anteriores, el numero equivalente se forma, de la unión de los dos números equivalentes, tanto entero como fraccionario, separados por un punto que establece la diferencia entre ellos.

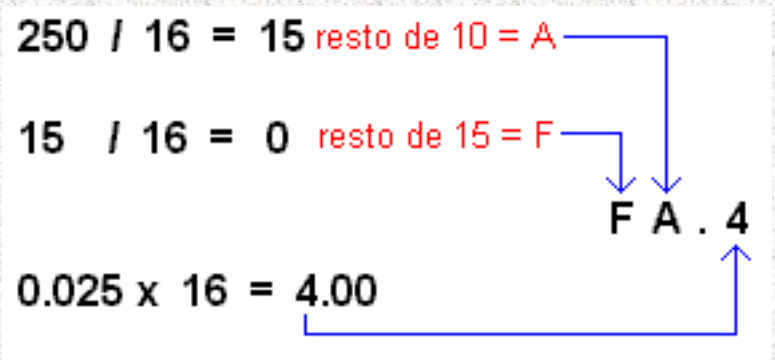


Figura 12: Conversión de decimal a hexadecimal

CONVERSIÓN DE UN NUMERO HEXADECIMAL A UN NUMERO DECIMAL

Como en los ejemplos anteriores este también nos ayudará a entender mejor este procedimiento: Convertir el numero hexadecimal 2B6 a su equivalente decimal.

1. Multiplicamos el valor de posición de cada columna por el dígito hexadecimal

correspondiente.

2. El resultado del número decimal equivalente se obtiene, sumando todos los productos obtenidos en el paso anterior.

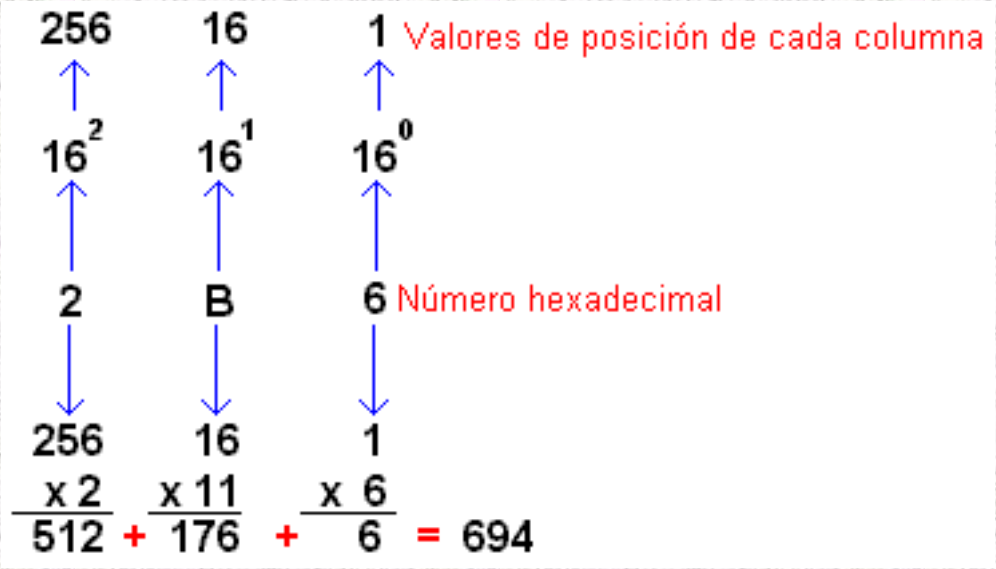


Figura 13: Conversión de hexadecimal a decimal

SISTEMA DE NÚMEROS EN COMPLEMENTO A 2

Este es un sistema que nos permite representar números binarios de forma negativa, en donde el MSB (Bit mas Significativo) es el bit del signo. Si este bit es 0 entonces el numero binario es positivo (+), si el bit del signo es 1, entonces el numero es negativo(-) los siete bits restantes del registro representan la magnitud del numero **1010110**, para complementar mejor la explicación tendremos que dedicarle mucha atención a la explicación de conversiones donde interviene este tipo de numeración, que es bastante utilizado en los microprocesadores, ya que estos manejan tanto números positivos como números negativos.

Para comprender mejor la conversión de sistema de numeración de este sistema de numeración, hay que tener en cuenta las siguientes definiciones

FORMA COMPLEMENTO A 1

El complemento a 1 de un numero binario se obtiene cambiando cada 0 por 1 y viceversa. En otras palabras, se cambia cada bit del numero por su complemento.

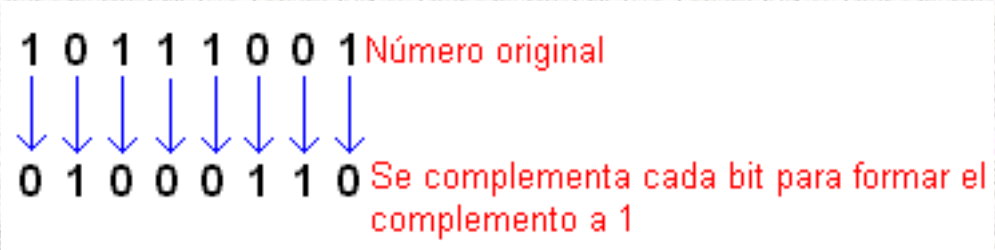


Figura 14: Complemento a uno

FORMA COMPLEMENTO A 2

El complemento a 2 de un numero binario se obtiene tomando el complemento a 1, y sumándole 1 al bit menos significativo. A continuación se ilustra este proceso para el numero 1001 = 9

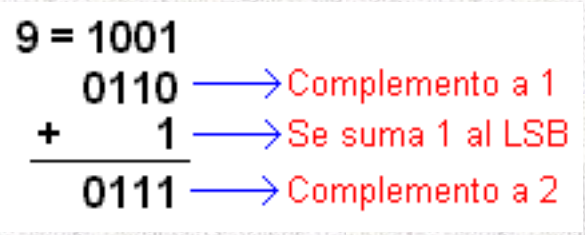


Figura 15: Complemento a 2

Cuando se agrega el bit de signo 1 al MSB, el numero complemento a 2 con signo se convierte en **10111** y es el numero equivalente al - 9.