**FUNDAMENTOS DE LOS COMPUTADORES**

PRACTICA 2:

ALGEBRA DE BOOLE

Francisco Joaquín Murcia Gómez 48734281H

Grupo 3 practicas

**Descripción de la práctica:**

En esta práctica se llevará a la practica los contenidos asociados al tema 2 de teoría, con diversos ejercicios y utilizando el software de construcción de circuitos logisim.

**Objetivos de la práctica:**

* Simplificar funciones lógicas aplicando los postulados y teoremas del Álgebra de Boole.
* Obtener experimentalmente una tabla de verdad.
* Construir y utilizar tablas de Karnaugh para la simplificación de funciones.
* Combinar los elementos de una tabla de Karnaugh para obtener una expresión mínima.
* Transformar expresiones en forma de Suma de Productos a su forma dual en forma de Producto de Sumas.

**Ejercicios:**

1. Haciendo uso de puertas lógicas básicas (AND, OR, NOT), construye un circuito que implemente la función NOR Exclusiva. f(a,b)=~(a⊕b)=ab+~a~b. Utiliza los elementos “pin” para proporcionar las variables de entrada y el elemento “ver” visualizar la salida.

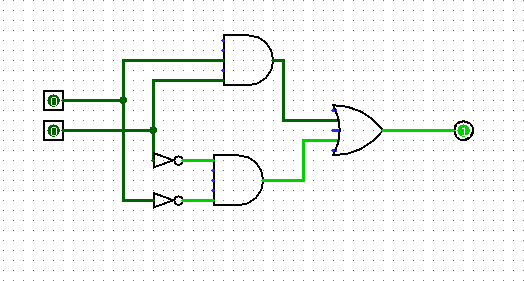
Para construir una puerta XNAND con puertas básicas necesitamos una puerta NOT, una AND y una OR.

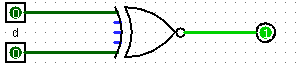
La puerta AND es el producto de la variable “a” y “b”.

La NOT es la negación de la OR.

Y la OR es la suma de las dos variables.

Por lo que la puerta XNOR seria f=ab+~a~b

El circuito seria así con puertas básicas: Y sin puertas básicas así:



1. Dada la expresión f=[a~b(c+~(b+d))+~(ab)]c~d, simplifícala haciendo uso de los postulados, teoremas y leyes del álgebra de Boole que consideres oportunos. Implementa la expresión original y la simplificada y comprueba que sus tablas de verdad son coincidentes.

Utilizando el teorema de DeMorgan:

f=[a~b(c+~(b+d))+~(ab)]c~d 🡪 f=[a~b(c+(~a~b))+~a+~b]c~d

Reduzco la función hasta que me queda la siguiente expresión:

f=a~bc~d+~ac~d+~bc~d

Saco factor común c~d:

f=(a~b+~a+~b) c~d

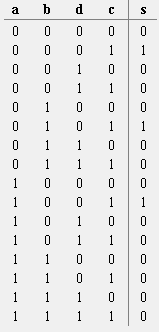
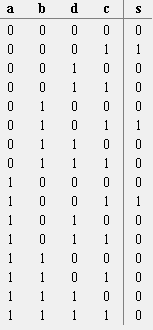
Utilizando la ley de absorción [~b+~ba=~b] sacamos:

f=(~a+~b)c~d

Finalmente tenemos como resultado:

**f=~ac~d+~bc~d**

Función sin simplificar Función simplificada

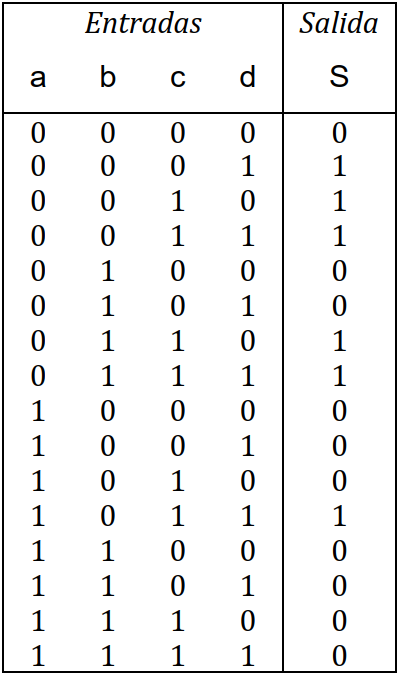


Como podemos observar es la misma función, pero simplificada.

1. Una función lógica f(a,b,c,d) presenta la tabla de verdad de la Tabla
2. Escribe su función algebraica en forma de suma de productos y de producto de sumas en su formato numérico.

Suma de productos

Donde la salida es 1, se coge el valor de las variables, cuando la entrada es 0 la variable se niega, una vez cogidos esos términos de suman, como podemos observar:



🡪~a~b~cd

🡪~abc~d

🡪~a~bcd

🡪~abc~d

🡪~abcd

🡪a~bcd

f=Σ4(1,2,3,6,7,11)

**f=~a~b~cd+~a~bc~d+~a~bcd+~abc~d+~abcd+a~bcd**

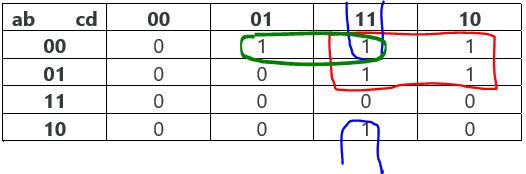
Productos de sumas

Se emplearía el mismo método, pero en vez de coger los términos cuando la salida es 1, es cuando es 0 y cundo la variable es 1 se niega la variable

f=∏4(0,4,5,8,9,10,12,13,14,15)

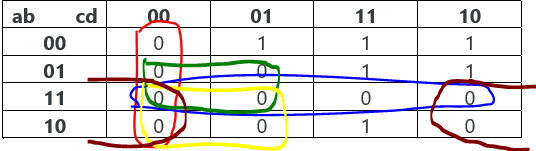
**f=(a+b+c+d)(a+~b+c+d)(a+~b+c+~d)(~a+b+c+d)(~a+b+c+~d) (~a+b+~c+d)(~a~b+c+d)(~a~b+c~d)(~a+~b~c+d)(~a+~b+~c+~d)**

1. Construye la tabla de Karnaugh asociada y simplifícala para obtener la expresión mínima en forma de suma de productos.



**f=~a~bd +~ac+~bcd**

1. Vuelve a simplificar la tabla de Karnaugh para obtener la expresión en forma de producto de sumas

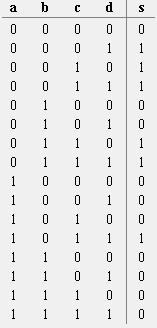
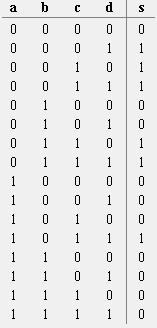


**f=(c+d)(~b+c)(~a+c)(~a+d)(~a+~b)**

1. Utilizando Logisim, implementa la función simplificada en los dos formatos y comprueba que las tablas de verdad coinciden con la original.

Suma de productos

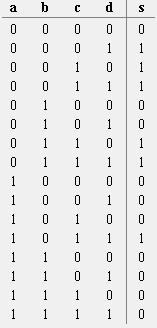
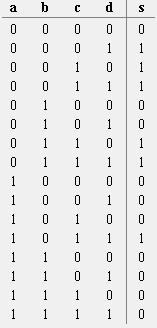
Simplificada sin simplificar



=

Producto de sumas

Simplificada sin simplificar

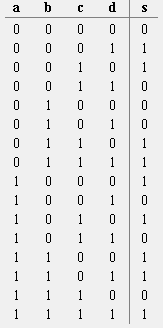


=

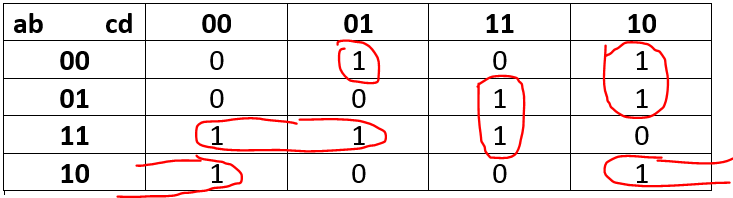
Ambas son la misma función, pero en formatos distintos

1. Sea la función f=∏4 (0,3,4,5,9,11,14).
   1. Simplifícala y obtén su expresión mínima en forma de suma de productos.

Como te dan el esquema del producto de sumas, he construido la tabla de verdad poniendo la salida como 0 en las filas que me indica.



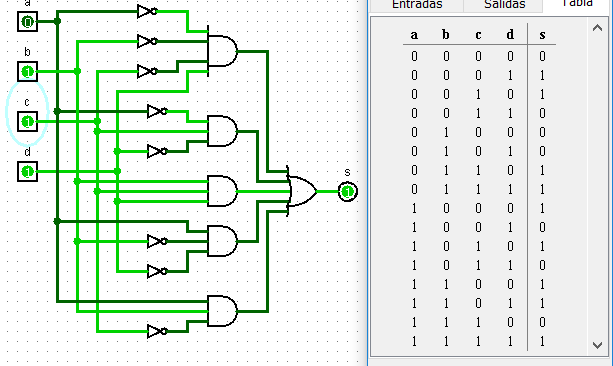
Con esta tabla he construido una tabla de Karnaugh y he agrupado los 1



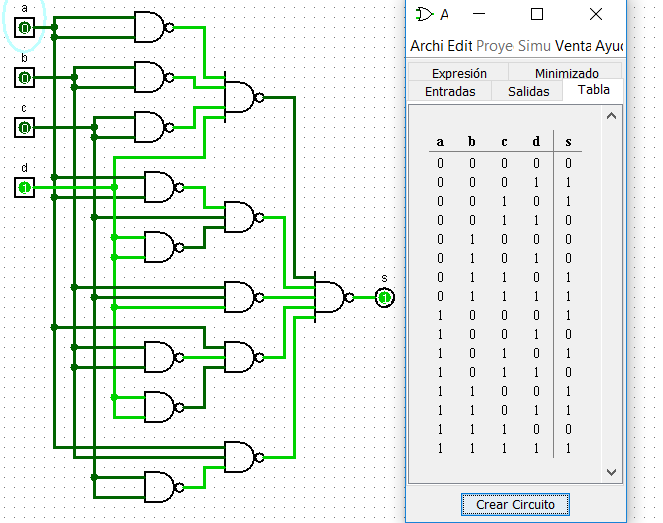
Como resultado:

**f=~a~b~cd+~ac~d+bcd+a~b~d+ab~c**

* 1. Implementa el circuito haciendo uso del menor número de puertas básicas posible y obtén su tabla de verdad.



* 1. Vuelve a implementar la función haciendo uso solamente de puertas NAND. Obtén su tabla de verdad y comprueba que la implementación es correcta.

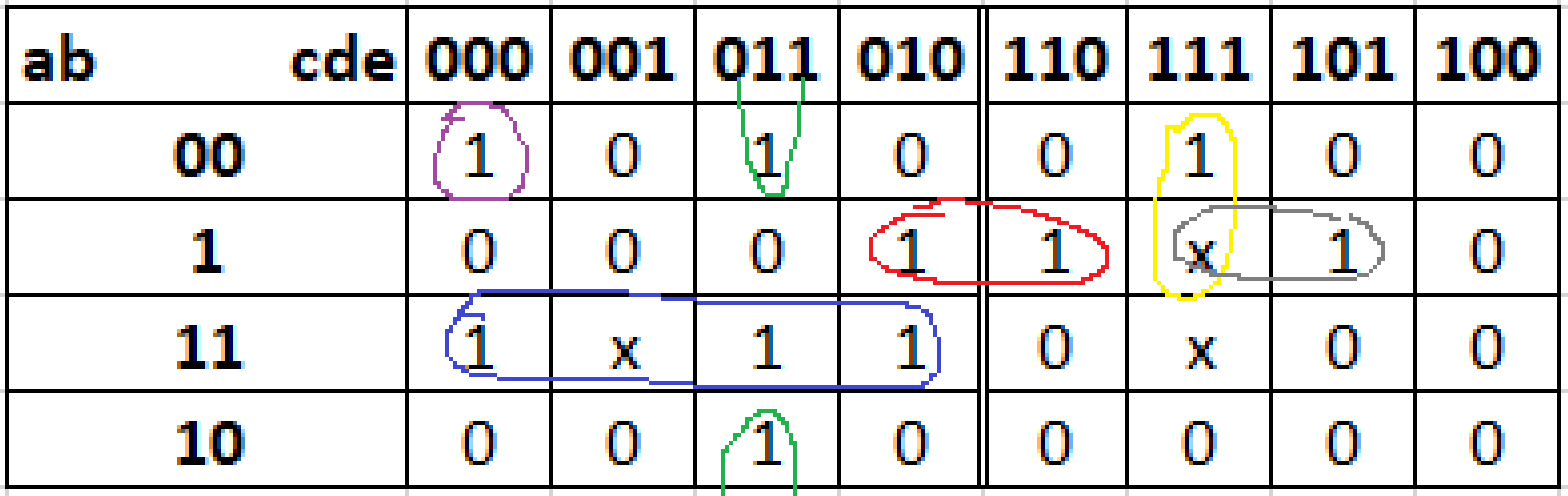


Podemos demostrar su validez debido a que su tabla de verdad coincide con la del apartado anterior

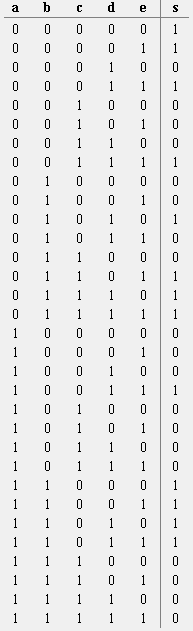
1. Simplifica e implementa, haciendo uso del mínimo número de puertas, la función:

f=Σ5 (10,3,19,24,14,13,0,7,26,27)+Σ0 (15,25,31)

Comprueba que la salida del circuito es correcta para cada una de las combinaciones. ¿Qué ocurre con las indiferencias al obtener la tabla de verdad?



**f=~a~b~c~d + ab~c + ~a~bde + ~b~cde + ~abd~e + ~abce**



Para los valores “x”, en la tabla coloca 1 si se coge la x en un grupo de Karnaugh y 0 si no, como el caso de la posición 31, como la x no se junta con ningún 1 se pone en la salida 0.

