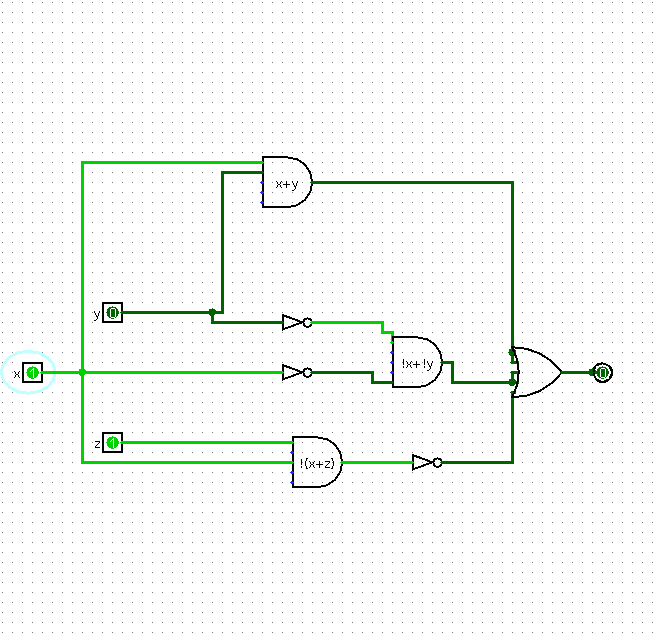
**Fundamentos de los computadores**

**Practica** **2:** Álgebra de Boole

**Ejercicio 1:**

Circuito implementado en Logisim (archivo ej1.circ) para la función 



**Ejercicio 2:**

Simplificamos la siguiente expresión*:*

Tabla de verdad para

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***a*** | ***b*** | ***c*** | ***d*** | ***S*** |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | **1** |
| 0 | 0 | 1 | 1 | **1** |
| 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 | 0 |
| 1 | 0 | 1 | 0 | **1** |
| 1 | 0 | 1 | 1 | **1** |
| 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 | 0 |

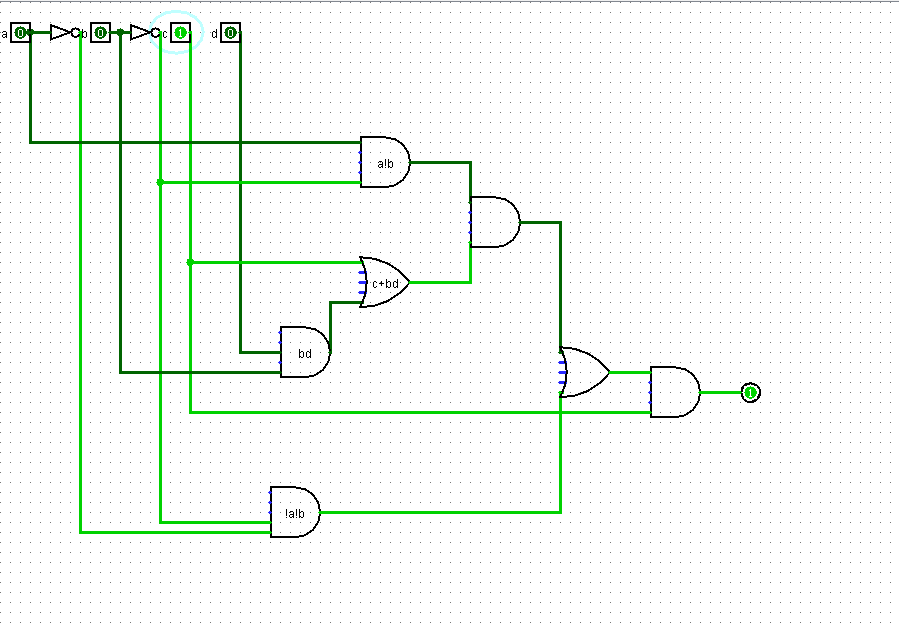
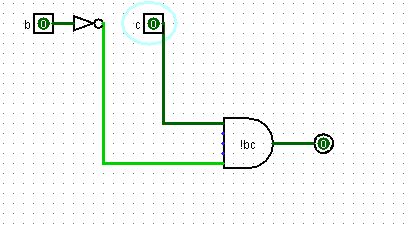
Circuito implementado en Logisim (archivo *ej2-original.circ)*

Tabla de verdad para

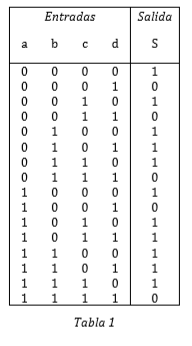
|  |  |  |
| --- | --- | --- |
| ***b*** | ***c*** | ***S*** |
| 0 | 0 | 0 |
| 0 | 1 | **1** |
| 1 | 0 | 0 |
| 1 | 1 | 0 |

Como Podemos ver en ambas tablas de verdad, la única situación que da como salida 1 es cuando b=0 y c=1, sin importar el valor de las otras variables.

Circuito implementado en Logisim (archivo *ej2-simplificada.circ)*

**Ejercicio 3:**

**Apartado a:**



Dada la siguiente tabla averiguamos:

La función algebraica en forma de suma de productos:

y en forma de producto de sumas:

**Apartado b:**

La tabla de Karnaugh y la expresión en forma de suma de productos:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ab\cd** | **00** | **01** | **11** | **10** |
| **00** | 1 | 0 | 0 | 1 |
| **01** | 1 | 1 | 0 | 1 |
| **11** | 1 | 1 | 0 | 1 |
| **10** | 1 | 0 | 1 | 1 |

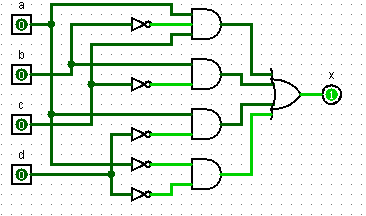
**Apartado c:**

La tabla de Karnaugh y la expresión en forma de producto de sumas:

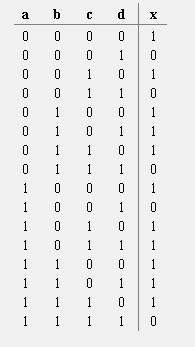
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ab\cd** | **00** | **01** | **11** | **10** |
| **00** | 1 | 0 | 0 | 1 |
| **01** | 1 | 1 | 0 | 1 |
| **11** | 1 | 1 | 0 | 1 |
| **10** | 1 | 0 | 1 | 1 |

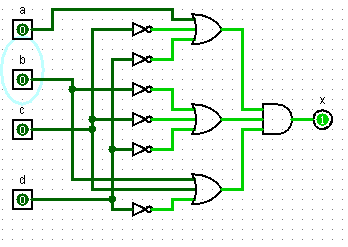
)

**Apartado d:**

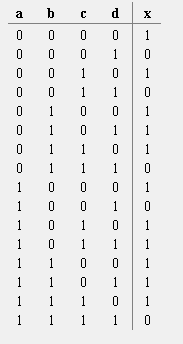
Circuito implementado en Logisim en forma de suma de productos (*archivo ej3d-sop.circ)* 

Y su tabla de verdad:



Circuito implementado en Logisim en forma de producto de sumas (archivo ej3d-pos.circ)

Y su tabla de verdad



Ya que ambas tablas de verdad coinciden, ambas formas de implementar el circuito son correctas.

**Ejercicio 4:**



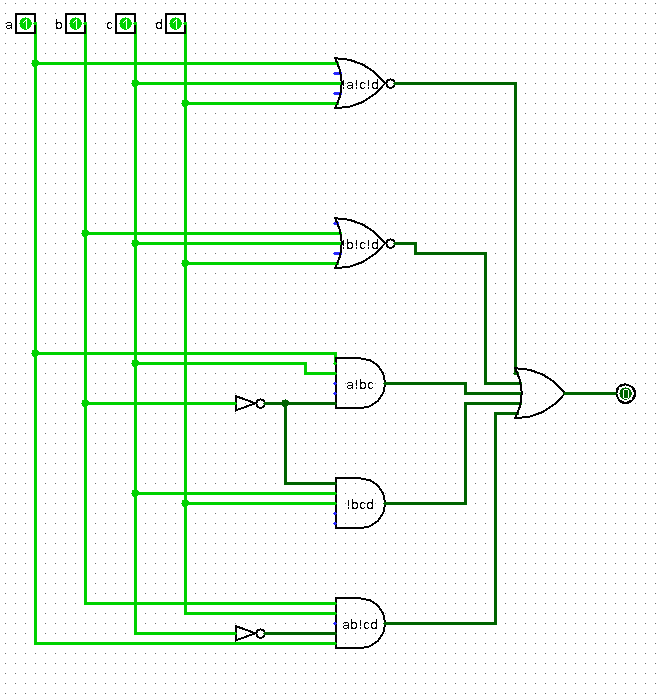
**Apartado a:**

Obtenemos la tabla de Karnaugh para obtener su expresión mínima en forma de suma de productos:

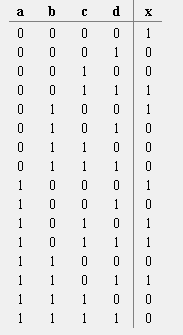
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ab\cd | 00 | 01 | 11 | 10 |
| 00 | 1 | 0 | 1 | 0 |
| 01 | 1 | 0 | 0 | 0 |
| 11 | 0 | 1 | 0 | 0 |
| 10 | 1 | 0 | 1 | 1 |

Su expresión en forma de suma de productos:

**Apartado b:**

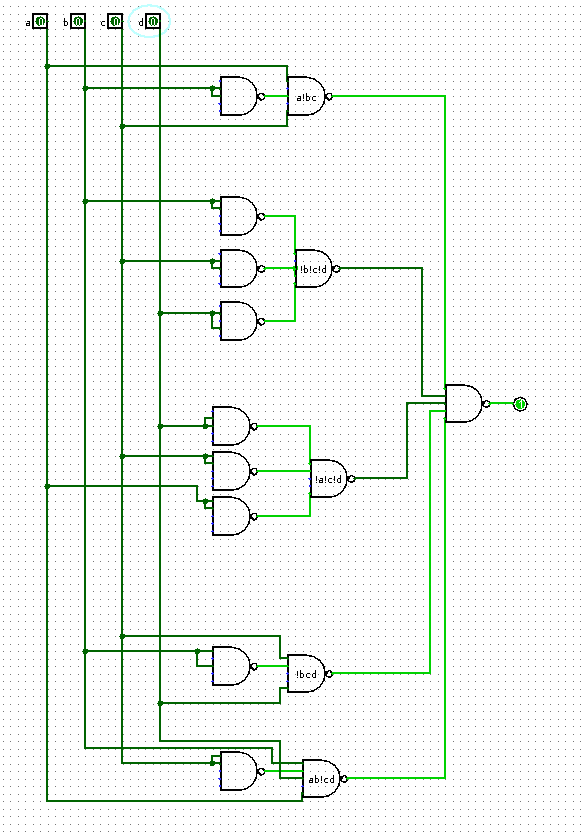
Circuito implementado en Logisim (archivo ej4b.circ) para la expresión anterior.

Y obtenemos su tabla de verdad:

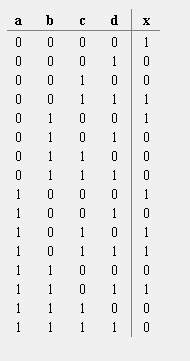


**Apartado c:**

Primero obtenemos la expresión para que pueda ser implementada solo con puertas NAND:

Circuito implementado en Logisim (archivo ej4c.circ) utilizando solo puertas NAND:

Y su tabla de verdad:



Ya que ambas tablas de verdad coinciden, ambas formas de implementar el circuito son correctas.

**Ejercicio 5:**

Para la función:



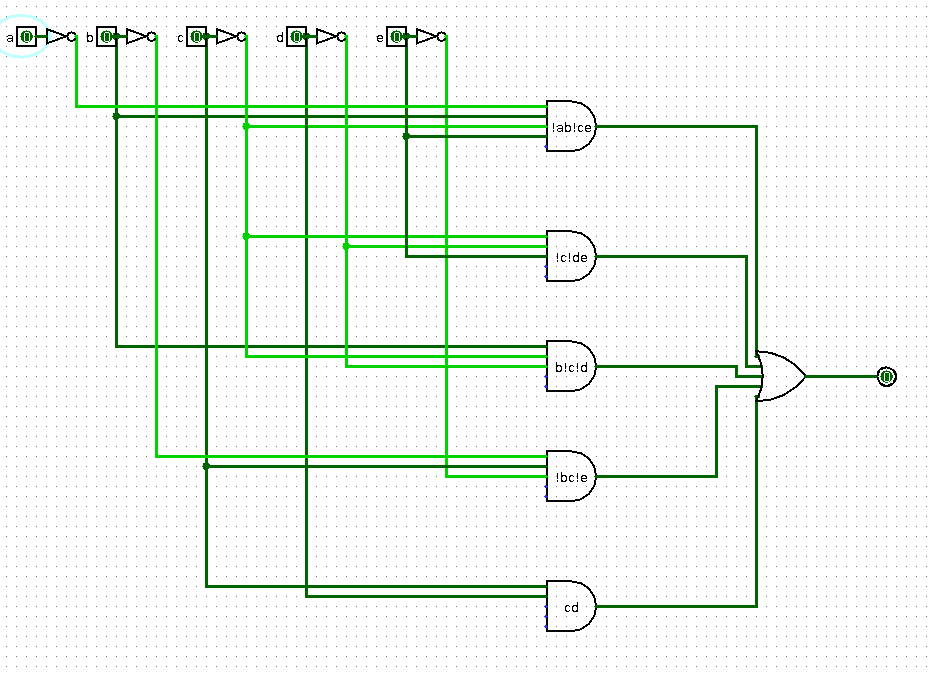
Primero obtenemos su tabla de verdad:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| a | b | c | d | e | S |
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 0 | 1 | 0 | **X** |
| 0 | 0 | 0 | 1 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 | 1 | 0 |
| 0 | 0 | 1 | 1 | 0 | 1 |
| 0 | 0 | 1 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 1 | 1 | 1 |
| 0 | 1 | 1 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 1 | **X** |
| 0 | 1 | 1 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 1 | **X** |
| 1 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 1 |
| 1 | 0 | 0 | 1 | 0 | **X** |
| 1 | 0 | 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 | 1 | 0 |
| 1 | 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 0 | **X** |
| 1 | 1 | 0 | 0 | 1 | 1 |
| 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 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 |

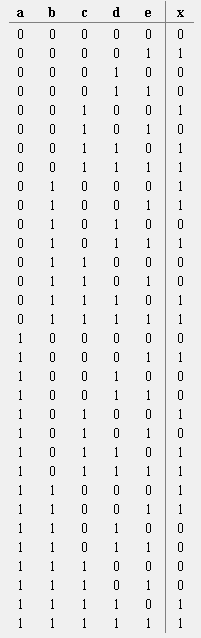
Lo que ocurre con las indiferencias (marcadas como X) es que pueden ser utilizadas tanto como 1 o 0 en la tabla de Karnaugh, lo cual nos facilita sacar las expresiones de suma de productos o producto de sumas.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ab\cde | 000 | 001 | 011 | 010 | 110 | 111 | 101 | 100 |
| 00 | 0 | 1 | 0 | x | 1 | 1 | 0 | 1 |
| 01 | 1 | **1** | **1** | 0 | 1 | x | x | 0 |
| 11 | x | 1 | 0 | 0 | 1 | 1 | 0 | 0 |
| 10 | 0 | 1 | 0 | x | 1 | 1 | 0 | 1 |

Y su expresión en forma de suma de productos:

Circuito implementado en Logisim (archivo ej5.circ)

Y su tabla de verdad:



Como lo podemos ver en esta tabla, los valores de salida X que hemos contado como 1 en la tabla de Karnaugh dan salida 1, y los que no hemos utilizado dan salida 0.