Universidad Rafael Landívar
Tradición Jesuita en Guatemala

Matemática Discreta I

Ing. Juan Carlos Soto

POTENCIA BINARIA

PROYECTO: DOCUMENTACION DEL PROYECTO

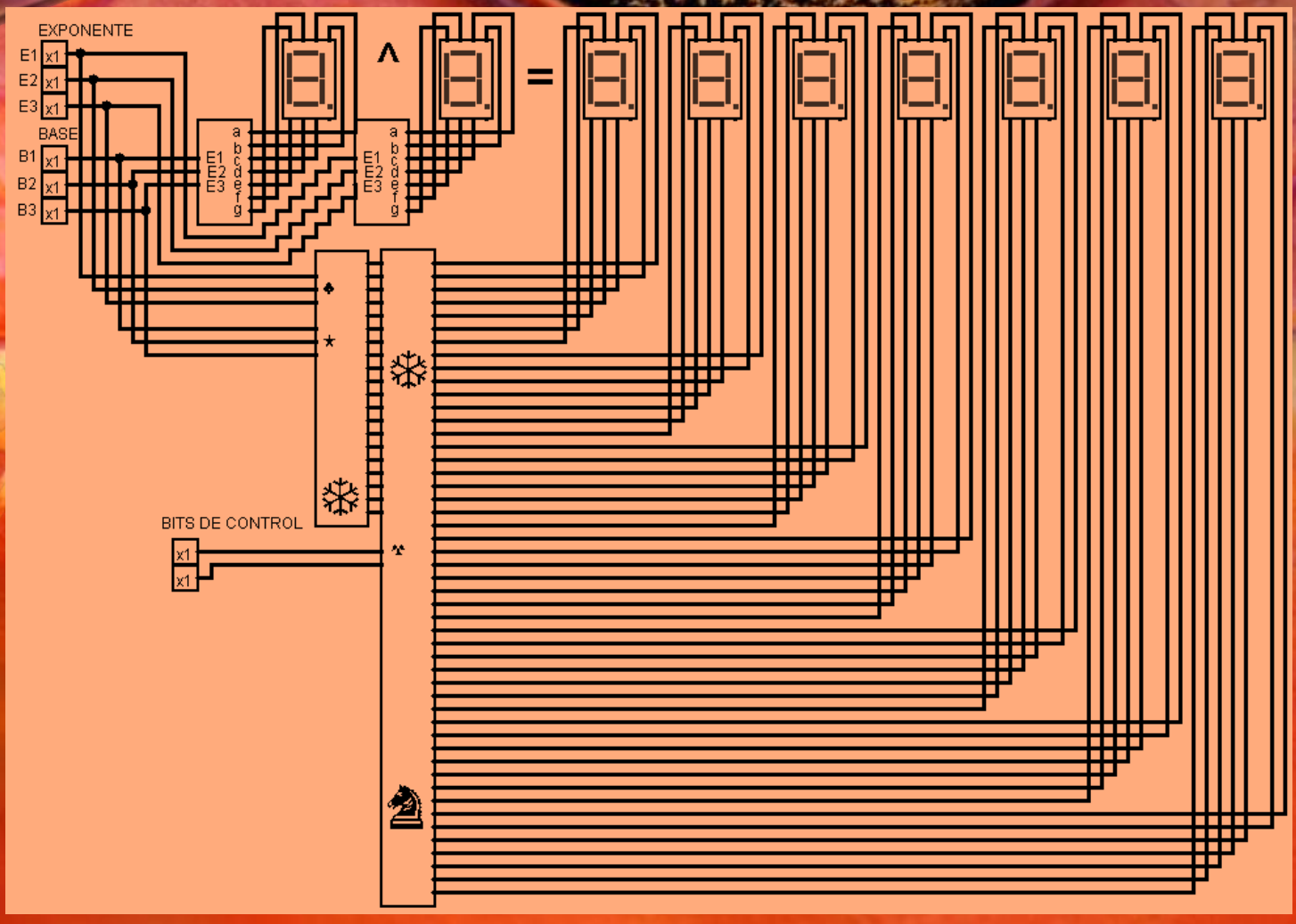
Luis Fernando Guevara Toledo (1195513)

Luis Pedro González Morales (1086415)

Francisco Josué Solís Ruano (1050014)

Ana Cecilia Ávila López (1121714)

POTENCIA BINARIA



Introducción

La herramienta logisim está diseñada para crear circuitos digitales. Tiene un gran poder que se expande junto a la imaginación del diseñador sin embargo cuenta con algunos desperfectos que van ligados al rendimiento del ordenador donde se esta ejecutando. A pesar de todo ayuda al usuario por su interfaz y la abundancia de tutoriales que facilitan su uso. Un circuito digital usa el sistema binario que son precisamente los digitos 0's y 1's por ende es más fácil expresar su notación en los sistemas hexadecimal u octal debido a las bases son múltiplos del sistema binario; utilizando la técnica de agrupación es posible la relación entre un sistema y otro. Como reto adicional nos propusimos entregarle al usuario la posibilidad de devolver el resultado en el sistema más usado, es decir el sistema decimal. Para que esta meta se lograra, usando el alto conocimiento de los integrantes del grupo y la teoría BCD (Binary Coded Decimal) se relaciono y creo un circuito convertor del sistema binario tradicional al BCD.

El circuito esta conformado de operaciones básicas como son la suma y multiplicación, que van aumentando progresivamente sus entradas, y la agrupación de operaciones que en determinado momento ayudaban a simplificar la digitalización de las operaciones más complicados. Es decir, la potencia contenia dentro de si multiplicaciones y sumas. Una potencia mayor utilizaba el circuito de la potencia anterior para que esta fuera reutilizable e hiciera más fácil su edición.

Investigación del tema

Circuito digital:

- **Operan en modo binario donde cada voltaje de entrada y de salida es un 0 y un 1; las designaciones 0 y 1 representan intervalos predefinidos de voltaje. Esta característica de los circuitos lógicos nos permite utilizar el *álgebra booleana* como herramienta de para el análisis y diseño de sistemas digitales. Las *compuertas lógicas,* que son los circuitos lógicos más fundamentales, y observaremos cómo puede describirse su operación mediante el uso del álgebra booleana.**

Álgebra Booleana:

Algebra booleana es un álgebra que le permite abstraer las principales operaciones algebraicas en un sistema binario.

Esta fue formulada por George Bool en su libro: "An Investigation of the laws of thought" que significa: "Una investigación sobre las leyes del pensamiento".

Se basa en un conjunto de valores $N = \{0, 1\}$. Los cuales están estrechamente relacionados a los de la lógica siendo F = 0 y 1 = V.

Como las operaciones del álgebra ordinaria van sobre los números reales, el álgebra de Boole se lleva sobre números binarios.

Se permiten las siguientes operaciones:

- Suma (+) como un "O" (V) en leyes de la lógica.
- Multiplicación (×) como un "Y" (^) en leyes de la lógica.
- Complemento ($\bar{x}, x'$) como "NOT" (¬) en leyes de la lógica.

Las cuales están definidas según la siguiente tabla de verdad:

H	J	H + J	H*J	$\bar{H}$	$\bar{J}$
0	0	0	0	1	1
0	1	1	0	1	0
1	0	1	0	0	1
1	1	1	1	0	0

- **La Lógica Proposicional**:

El álgebra booleana le permite procesar las expresiones y la forma algebraica siguiendo una lógica proposicional en donde las funciones devuelven sólo los resultados cero o uno.

- **Forma normal Disyuntiva (FND):**

Para cualquier n ϵ $\mathbb{Z}^+$, si f es una función booleana sobre las variables $x_1, x_2, x_3 \ldots x_n$.

a) Cada termino xi o su complemento xi para $1 \leq i \leq n$ es una literal.

b) Cada término de la forma $y_1, y_2, y_3 \ldots y_n$ donde cada $y_i = x_i$ o x_i, es una conjunción fundamental.

c) Una representación de suma de conjunciones.

FND se liga directamente a los 1's.

Ejemplo:

x	y	z	f	
0	0	0	1	$= \bar{x}\bar{y}\bar{z}$
0	0	1	0	
0	1	0	0	
0	1	1	1	$= \bar{x}yz$
1	0	0	0	
1	0	1	0	
1	1	0	1	$= xy\bar{z}$
1	1	1	0	

Cada **1** se transforma a una multiplicación de las variables y se suman cada uno de los términos. Obteniendo así la función resultante. Cuando se trata de FND, **si la variable posee un valor de 0 se dispone como negada.**

$$f = \bar{x}\bar{y}\bar{z} + \bar{x}yz + xy\bar{z}$$

- **Forma normal conjuntiva (FNC):**

Para cualquier n ϵ $\mathbb{Z}^+$, si f es una función booleana sobre las variables $x_1, x_2, x_3 \ldots x_n$.

a) Cada término x_i o su complemento x_i para $1 \leq i \leq n$ es una literal.

b) Cada término de la forma $y_1, y_2, y_3 \ldots y_n$ donde cada $y_i = x_i$ o x_i, es una disyunción fundamental.

c) Una representación de multiplicación de disyunciones.

FNC se liga directamente a 0's.

Ejemplo:

x	y	z	f	
0	0	0	0	$= (x + y + z)$
0	0	1	1	
0	1	0	1	
0	1	1	0	$= (x + \bar{y} + \bar{z})$
1	0	0	1	
1	0	1	1	
1	1	0	0	$= (\bar{x} + \bar{y} + z)$
1	1	1	1	

Cada **0** se transforma a una suma de las variables y se multiplican cada uno de los términos. Obteniendo así la función resultante. Cuando se trata de FNC, **si la variable posee un valor de 1 se dispone como negada.**

$$f = (x + y + z)(x + \bar{y} + \bar{z})(\bar{x} + \bar{y} + z)$$

- **Compuertas Lógicas:**

Representación gráfica de las operaciones del álgebra booleana.

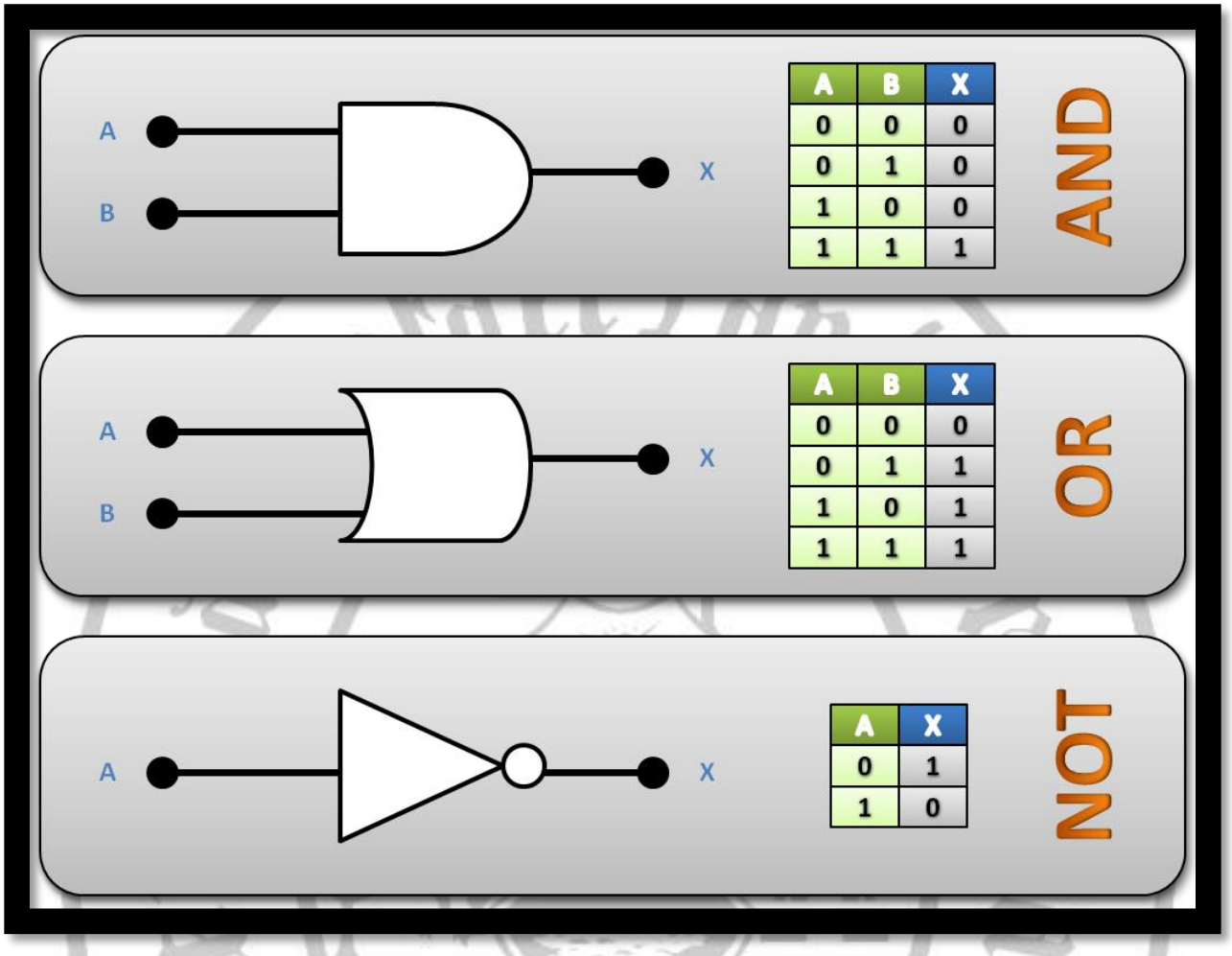


Ilustración 1. Resumen de compuertas lógicas.

Se pueden unir múltiples variables en una sola compuerta (no es exclusivamente de 2 variables), se basa en 2 debido a su relación directa con chips de electrónica que ejecutan una determinada acción.

En general, cuando obtenemos expresiones mediante la simplificación por mapas de Karnaugh, quedan **dos pasos** en el diagrama.

- Si es de **mintérminos**, quedarían X compuertas **AND** unidas por una gran compuerta **OR**.

- Si es de **maxtérminos**, quedarían Y compuertas **OR** unidas por una gran compuerta **AND**.

- **Mapa de Karnaugh:**

Es un diagrama basado en las tablas de verdad que nos ayuda a simplificar funciones del álgebra booleana. La base para simplificar es agrupar 1's o 0's en potencias de 2 (1, 2, 4, etc.) e identificar qué es lo común a dichos elementos en sus representaciones binarias.

$2^0 = 1$
$2^1 = 2$
$2^2 = 4$
$2^3 = 8$

} Indica la cantidad que puedo agrupar, 1, 2, 4 u 8.
Solo se pueden agrupar horizontal y verticalmente.

Indica la cantidad de variables que se van a eliminar, las variables que no están en común, u, v, w, x, y, z.

- Cuando se manipula **1's (mintérminos),** si el valor común de una variable es 1 se dispone la variable normal (x), en caso sea 0 se dispone la variable negada $(\bar{x})$, todas las variables que no cambian se unen por medio de la multiplicación.

- Cuando se manipula **0's (maxtérminos)**, si el valor común de una variable es 1 se dispone la variable negada $(\bar{x})$, en caso sea 0 se dispone la variable normal(x), todas las variables que no cambian se unen por medio de la suma.

Ejemplo:

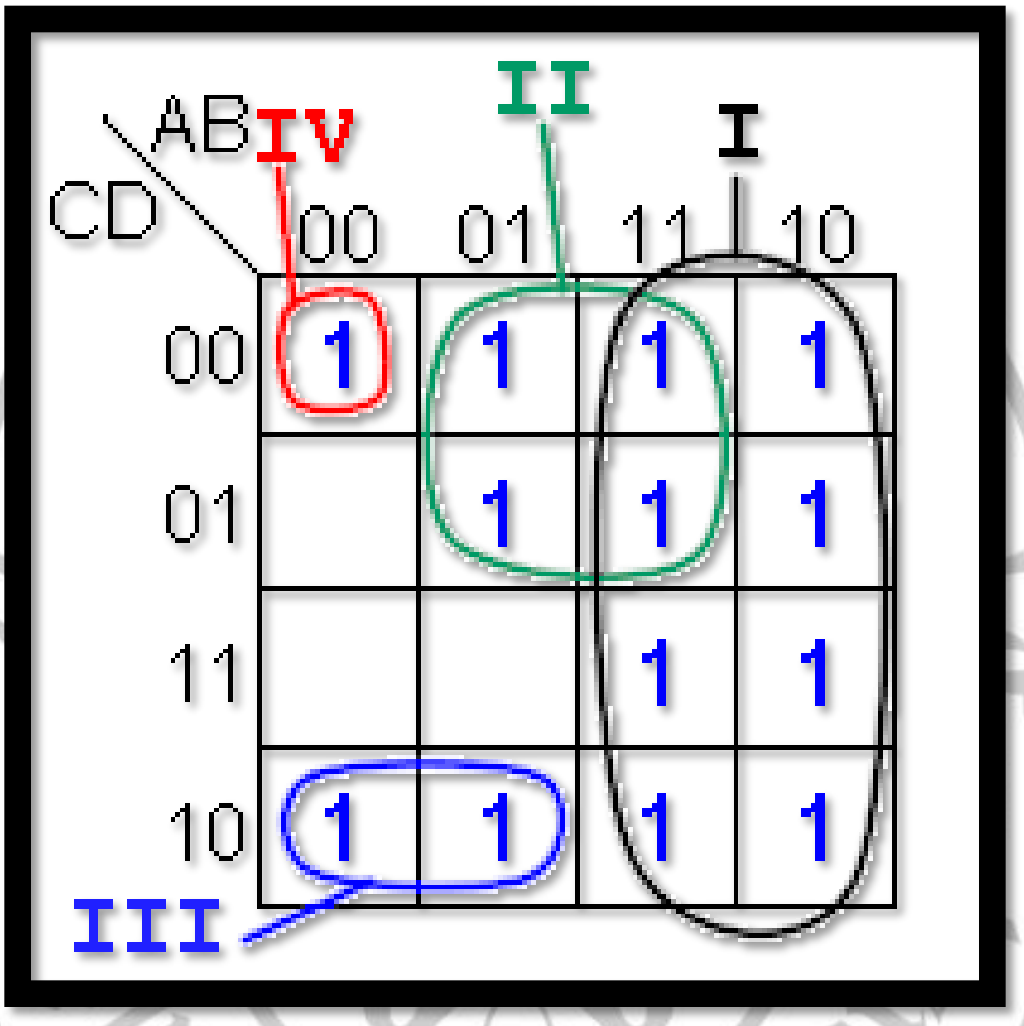


Ilustración 2. Ejemplo de un mapa de Karnaugh.

1) A
2) B¬C
3) ¬AC¬D
4) ¬A¬B¬C¬D

R/ (A) + (B¬C) + (¬AC¬D) + (¬A¬B¬C¬D)

- **Compuertas Lógicas:**

Son circuitos que generan voltajes de salida en función de la combinación de entrada correspondientes a las funciones lógicas.

Trabajan con dos estados lógicos (0, 1) los cuales pueden asignarse de acuerdo a la lógica o a la lógica negativa.

- **Lógica Positiva:**

En la lógica positiva una tensión alta representa un 1 y una tensión baja representa un 0.

- **Lógica Negativa:**

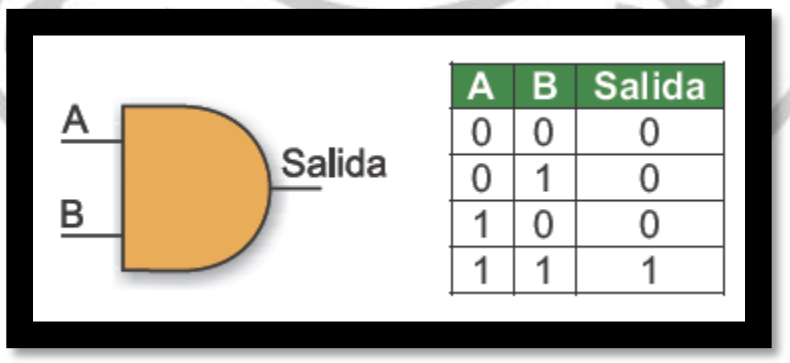
En la lógica negativa una tensión alta equivale a un 0 y una tensión baja representa a un 1.

Por lo general se suele trabajar con lógica positiva.

1) **Compuerta AND:** (p^q – xy)

Puede tener varias entradas pero solo tiene una salida.

Ejemplo:



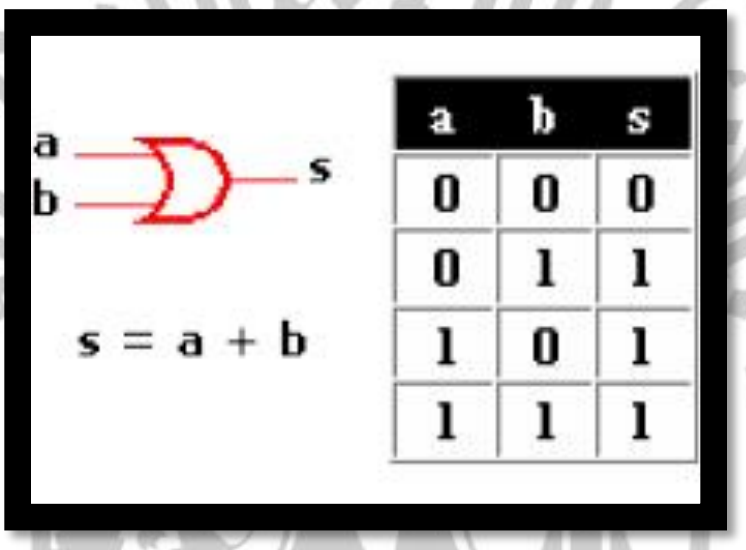
A	B	Salida
0	0	0
0	1	0
1	0	0
1	1	1

Ilustración 3. Compuerta lógica AND.

2) **Compuerta OR:** (pVq – x+y)

Entrega una salida positiva si en sus entradas está presente al menos un 1.

Ejemplo:



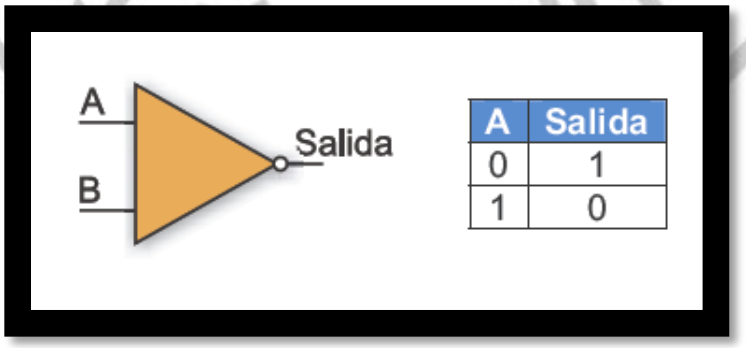
a	b	s
0	0	0
0	1	1
1	0	1
1	1	1

Ilustración 4. Compuerta lógica OR.

3) **Compuerta NOT:** (¬p – x')

La compuerta NOT (compuerta NO) también es llamada Compuerta Inversora.

Ejemplo:



A	Salida
0	1
1	0

Ilustración 5. Compuerta lógica NOT.

4) **Compuerta NAND:** ¬(a*b)

La compuerta NAND (No Y) opera de forma contraria a un AND, es su negación.

Ejemplo:

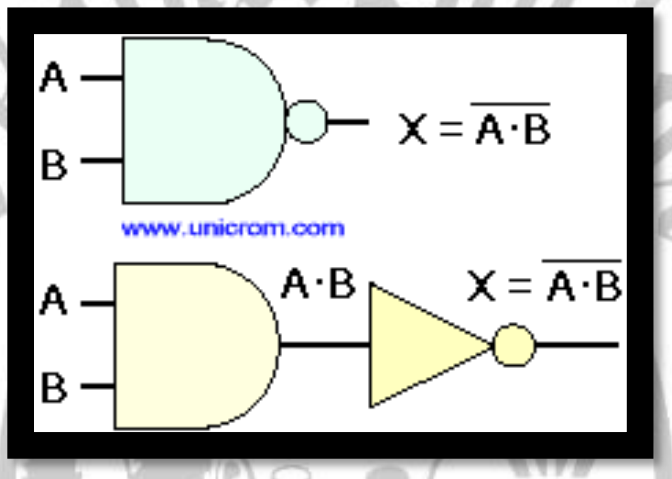


Ilustración 6. Compuerta lógica NAND.

5) **Compuerta NOR:** ¬(a+b)

Esta compuerta es el resultado de invertir la salida de una compuerta OR.

Ejemplo:

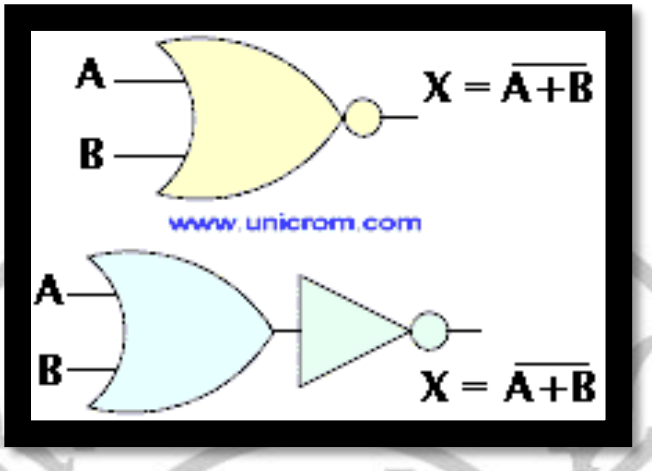


Ilustración 7. Compuerta lógica NOR.

6) **Compuerta OR exclusiva, XOR:** (a¬b + b¬a)

Esta compuerta realiza una suma lógica entre A por B invertida y A invertida por B. Por lo cual se le denomina OR exclusivo.

Esta compuerta tendrá una salida alta siempre y cuando sus entradas tengan niveles distintos.

Ejemplo:

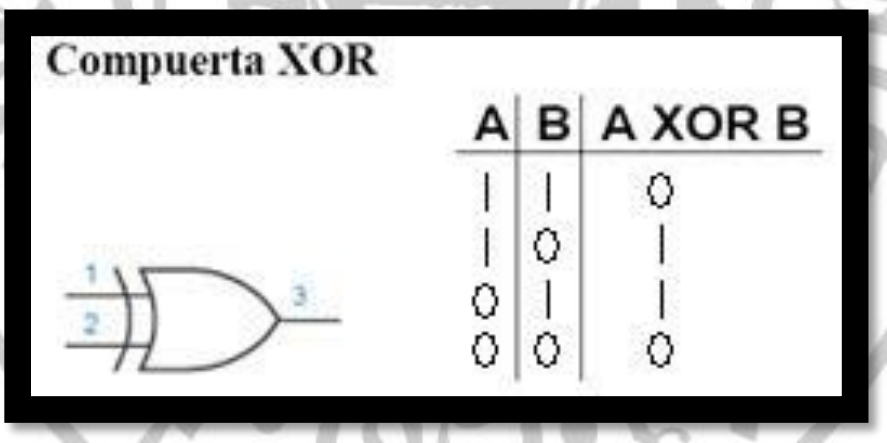


Ilustración 8. Compuerta lógica XOR.

7) **Compuerta NOR exclusiva, XNOR:** ¬(¬(a+b))

SU valor de verdad es igual a la doble condicional. La compuerta XNOR opera en forma opuesta a la XOR. Entrega una salida alta

cuando sus entradas tienen el mismo nivel. Es ideal para su aplicación en comparadores.

Ejemplo:

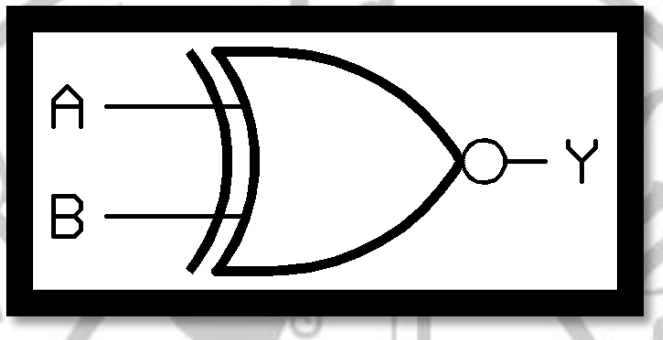


Ilustración 9. Compuerta lógica XNOR.

- **Sistemas de Numeración:**

Conjunto de reglas y símbolos que sirve para representar números.

Surgieron por la necesidad de contar.

- **Digito:**

Deriva del latín "Digitus" que quiere decir dedo. Representa cantidades.

- **Número:**

Idea o sensación de cantidad.

- **Numeral:**

Representación de un número.

Hay 3 tipos de sistemas:

- Aditivo: Símbolos y la suma del valor.
- Posicional: El calor es de la posición.
- Hibrido: Aditivo y Posicional.

1. **Sistema Aditivo:**

El valor se representa en figuras y el total se obtiene sumando la cantidad de figuras por el valor de cada una. (Palitos, piedras, etc.)

- Lo usaban los egipcios.
- No se necesita el 0.

No importa el orden, cada vez que aparezca se suma su valor.

2. **Sistema Posicional:**

Es la solución más práctica para representar grandes cantidades. Cuando se alcanzaba un determinado número se hacía una marca distinta que representaba a todos ellos. Este número se denominó "la Base".

3. **Sistema Híbrido:**

Son la unión del sistema aditivo y el sistema posicional. Utiliza tanto la suma como la multiplicación.

Ejemplos de sistemas híbridos son los números romanos, mayas, babilónicos y chino.

- **Notación Posicional:**

La notación posicional es un sistema de numeración en el cual cada digito posee un valor que depende de su posición relativa, la cual

está determinada por la base, que es el número de dígitos necesarios para escribir cualquier número.

Se utiliza la notación posicional usando la base en notación exponencial.

En Binario:

$$2^5, 2^4, 2^3, 2^2, 2^1, 2^0 = 32, 16, 8, 4, 2, 1$$

A. SISTEMA DECIMAL:

Emplea 10 símbolos llamados dígitos.

0, 1, 2, 3, 4, 5, 6, 7, 8, 9.

Emplea la base de 10.

10^4	10^3	10^2	10^1	10^0
10,000	1,000	100	10	1

B. SISTEMA BINARIO:

Emplea 2 símbolos llamados dígitos.

0, 1.

Emplea la base de 2.

2^4	2^3	2^2	2^1	2^0
16	8	4	2	1

C. SISTEMA HEXADECIMAL:

Emplea 16 símbolos llamados dígitos.

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F.

Emplea la base de 16.

16^3	16^2	16^1	16^0
4096	256	16	1

V. POTENCIA BINARIA

La potencia binaria se basa en la multiplicación binaria continua, la cual es la más sencilla que en cualquier otro sistema de numeración. Ya que los factores de la multiplicación solo pueden ser ceros o unos, el producto únicamente puede ser cero o uno.

X	Y	X * Y
0	0	0 x 0 = 0
0	1	0 x 1 = 0
1	0	1 x 0 = 0
1	1	1 x 1 = 1

En un ordenador, la operación de multiplicar se realiza por medio de sumas repetitivas. Aunque puede generar algunos problemas en la programación ya que cada suma de dos unos origina un arrastre. Por lo que para evitar el problema, es necesario contar el número de unos y arrastres en cada columna. Si el número de unos es par, la sumas será cero y si es impar será uno.

Ejemplo

$$101 \; x \; 111$$

$$\begin{array}{r} 111 \\ x \quad 101 \\ \hline 111 \\ 000 \end{array}$$

Conclusiones:

- Analizar y comprender las compuertas lógicas y circuitos digitales para usarlos en la práctica.
- El uso de software de compuertas lógicas como (logisim) se comprendió su funcionamiento.
- Se determinó que una tabla de verdad es la base de todas las compuertas lógicas.
- Se concluyó que el sistema binario es fundamental para lograr hacer un circuito digital.

Recomendaciones

- El programa logisim cuando tiene muchos circuitos se traba debido a la gran carga que generan o poca RAM del sistema. (usar una computadora con mayor poder)
- Que se ayude al estudiante con el inicio del proyecto para tener una mejor visión del mismo.

Análisis del Problema

Entrada: La Base se representa como un número binario de 3 bits y el Exponente como un número binario de 3 bits, ambos se encuentran en el rango decimal de 0 a 7 siendo un total de 6 entradas.

- Proceso: Encontrar la función que lograra multiplicar la base las veces necesarias que indica el exponente realizando dentro de sus circuitos internos las multiplicaciones continuas de los números, sumando los resultados temporales y multiplicando ese pre-resultado con la base hasta que llegue al límite estipulado para finalmente desplegar el resultado definitivo en 7 displays de 7 segmentos. Además de determinar la función que transformara las entradas y las salidas en hexadecimal y octal, que en este caso son: la base, el exponente y el resultado. También crear una función que deberá desplegar los números en hexadecimal y octal en un display, esta se realizará por medio de una tabla de verdad para saber que segmento o segmentos del display deberá encenderse para cada número y una entrada de cambio de base que permitirá escoger si el resultado se despliega en el sistema de base 8 o 16 según criterio del usuario. Las tablas de verdad se utilizarán para determinar todas las posibilidades de casos a los que se les añadirán las condiciones específicas que se añadirán a los mapas de Karnaugh y a partir de la simplificación de estos se obtendrá la función con la que se diseñará el circuito.

$$M^n = M * M * M * M * M \ldots . = RF$$

Para simplificar esta expresión usaremos lo más común de la programación: ***divide y vencerás*** evaluaremos M*M para llegar a un pre-resultado y volverlo a multiplicar por M (R_A*M) hasta n veces para llegar al resultado RF.

M^n =

```
    M
  * M
  _____
   A…A
+  B…B
  _____
    RA
*    M
  _____
   AA…A
+   B…B
  _____
    RA2
```

Con este procedimiento consecutivo se puede llegar al resultado deseado.

Caso especial: x^0
Sea x cualquier número real excepto el 0:

$$x^0 = X^{a-a}$$

Tomando “a” como cualquier número real. La segunda ecuación se cumple por propiedades de las potencias:

$$x^{a-a} = \frac{x^a}{x^a}$$

Por tanto, relacionando ambas ecuaciones:

$$x^0 = \frac{x^a}{x^a}$$

Como el numerador y el denominador son iguales, podemos simplificar:

$$\frac{x^a}{x^a} = 1$$

Es decir:

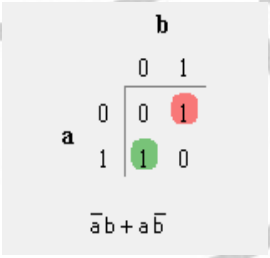
$$x^0 = 1$$

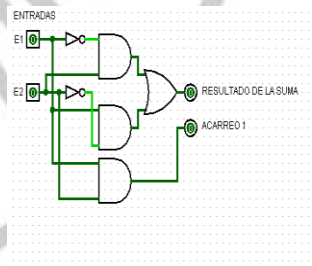
- Salida: Resultado de la operación exponencial mostrada en 7 displays de 7 segmentos. Al Display ira conectado una “Caja” llamada “Display Principal” que se encargara de entregarle 7 salidas para (a, b, c, d, e, f, g) de 4 entradas (A, B, C, D) dentro de esta entrada irán 7 sub “Cajas” que representaran los segmentos a, b, c, d, e, f y g que dentro de cada uno contiene el circuito que se puede obtener mediante la simplificación de un mapa de Karnaugh relacionando las 4 entradas y el resultado que se desplegara en el Display.

Diagramación del circuito

Sumador 2 variables

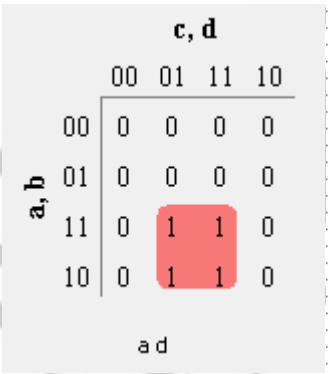
X	Y	Resultado	Acarreo1
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

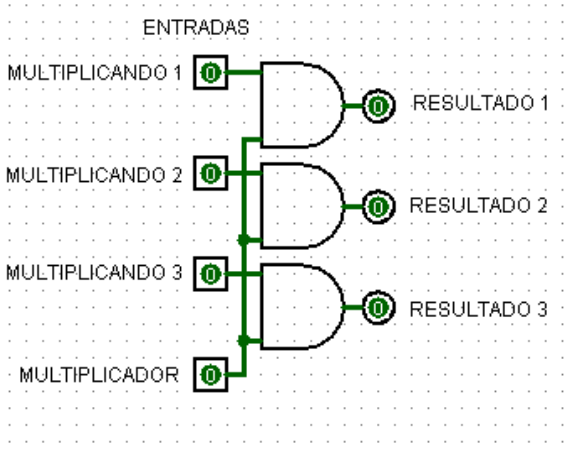




Multiplicador 3 entradas:

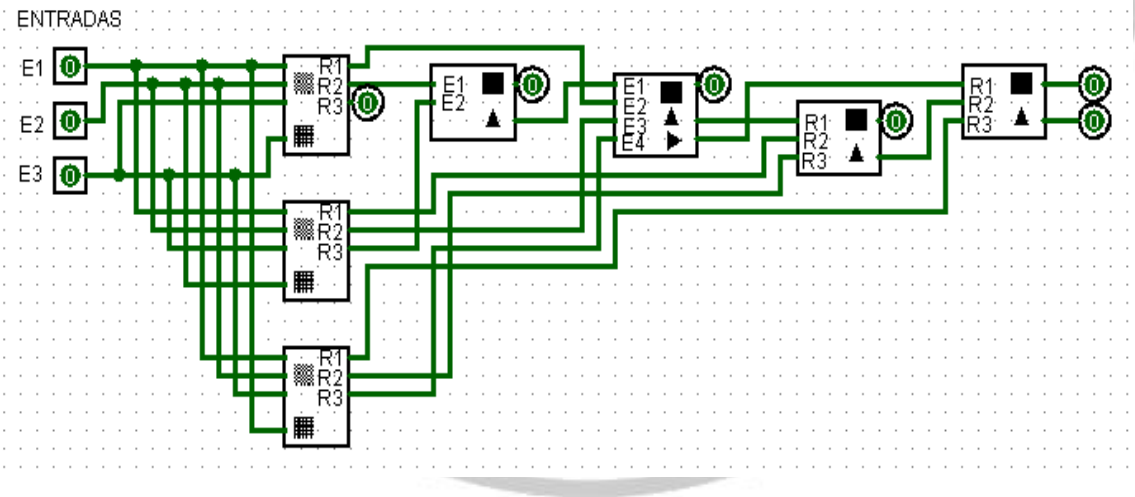
w	x	y	z				
0	0	0	0	0	0	0
0	0	0	1	0	0	0
0	0	1	0	0	0	0
0	0	1	1	0	0	1
0	1	0	0	0	0	0
0	1	0	1	0	1	0
0	1	1	0	0	0	0
0	1	1	1	0	1	1
1	0	0	0	0	0	0
1	0	0	1	1	0	0
1	0	1	0	0	0	0
1	0	1	1	1	0	1
1	1	0	0	0	0	0
1	1	0	1	1	1	0
1	1	1	0	0	0	0
1	1	1	1	1	1	1





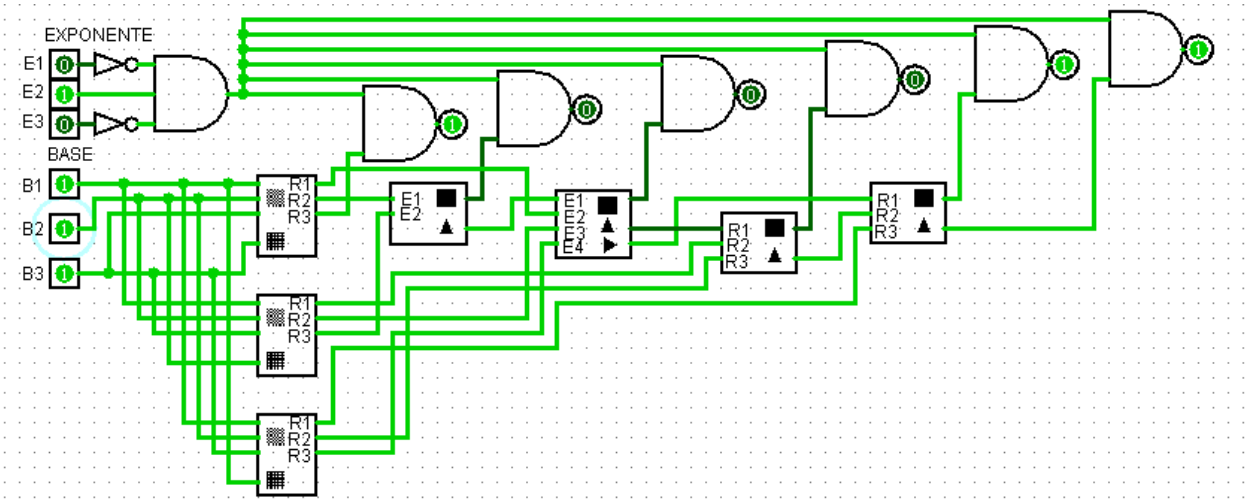
Para lograr multiplicar el número, cada bite de entrada se puso en un and de 2 entradas y la otra entrada es su multiplicador y el circuito se hizo 3 veces por el motivo de que son 3 entradas.

Potenciador p*p



Tiene 3 entradas y 6 salidas que multiplica y suma todas las variables.

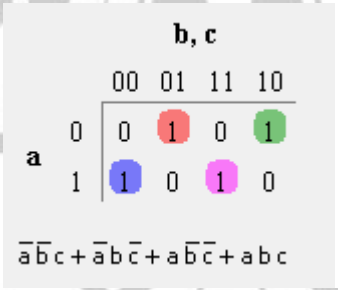
Elección potencia p*p

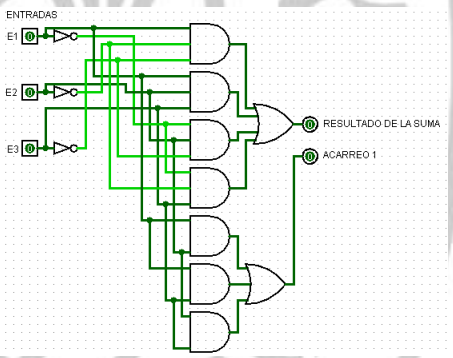


Se utilizó un bit de control para que solo en el caso que el exponente sea 2 muestre el resultado en caso contrario todo es 0

Sumador 3 variables

X	Y	Z	Resultado	Acarreo1
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

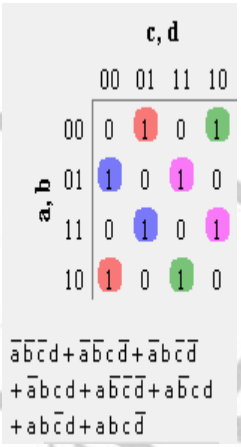


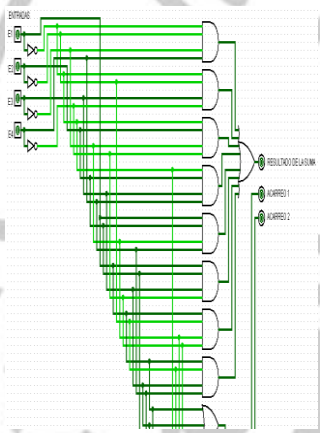


Sumador 4 variables

X	Y	Z	W	Resultado	Acarreo1	Acarreo2
0	0	0	0	0	0	0
0	0	0	1	1	0	0
0	0	1	0	1	0	0
0	0	1	1	0	1	0
0	1	0	0	1	0	0
0	1	0	1	0	1	0
0	1	1	0	0	1	0

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





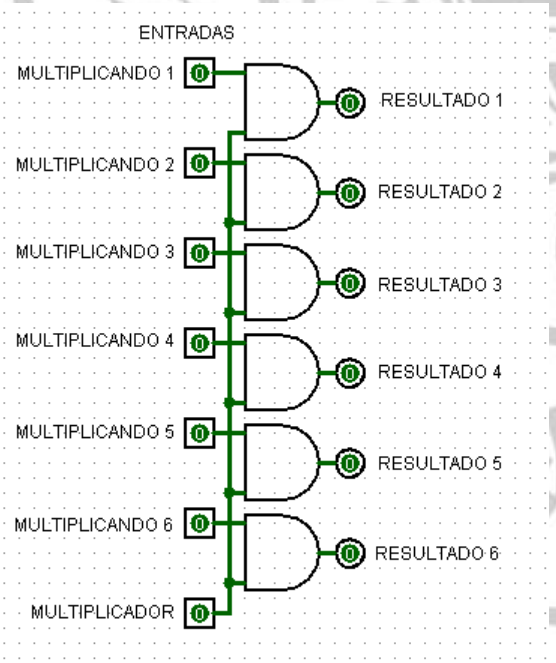
Multiplicador 6 multiplicando *1 multiplicador

U	V	W	X	Y	Z						
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	1	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0	0	0
0	0	0	0	1	1	0	0	0	0	0	1
0	0	0	1	0	0	0	0	0	0	0	0
0	0	0	1	0	1	0	0	0	0	1	0
0	0	0	1	1	0	0	0	0	0	0	0
0	0	0	1	1	1	0	0	0	0	1	1
0	0	1	0	0	0	0	0	0	0	0	0
0	0	1	0	0	1	0	0	0	1	0	0
0	0	1	0	1	0	0	0	0	0	0	0
0	0	1	0	1	1	0	0	0	1	0	1
0	0	1	1	0	0	0	0	0	0	0	0
0	0	1	1	0	1	0	0	0	1	1	1
0	0	1	1	1	0	0	0	0	0	0	0
0	0	1	1	1	1	0	0	1	0	0	0

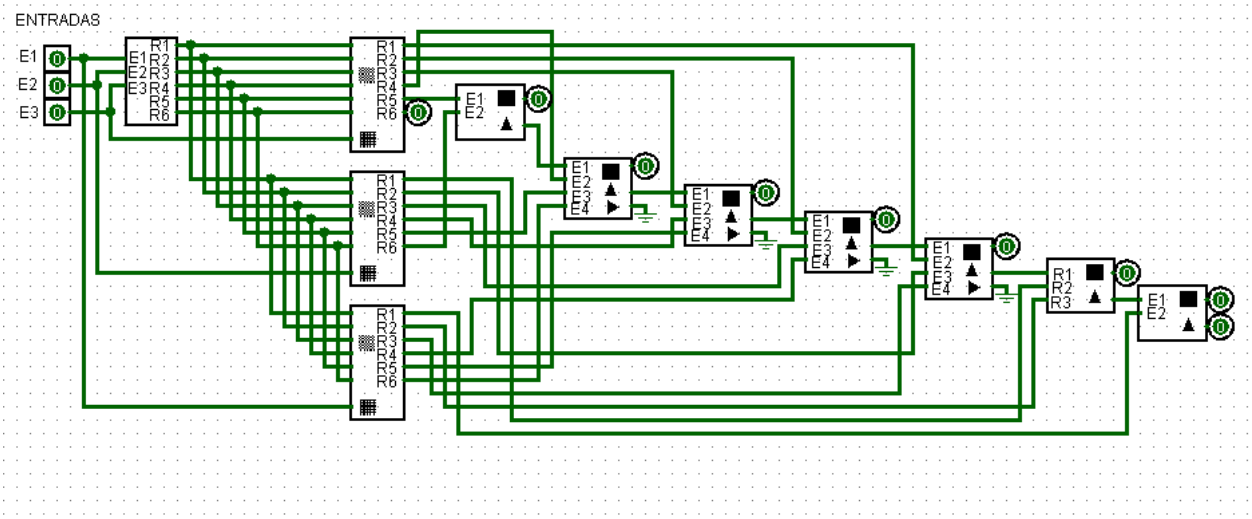
0	1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	1	0	0	1	0	0	1
0	1	0	0	1	0	0	0	0	0	0	0
0	1	0	0	1	1	0	0	1	0	1	1
0	1	0	1	0	0	0	0	0	0	0	0
0	1	0	1	0	1	0	0	1	1	0	0
0	1	0	1	1	0	0	0	0	0	0	0
0	1	0	1	1	1	0	0	1	1	0	1
0	1	1	0	0	0	0	0	0	0	0	0
0	1	1	0	0	1	0	0	1	1	1	0
0	1	1	0	1	0	0	0	0	0	0	0
0	1	1	0	1	1	0	0	1	1	1	1
0	1	1	1	0	0	0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

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

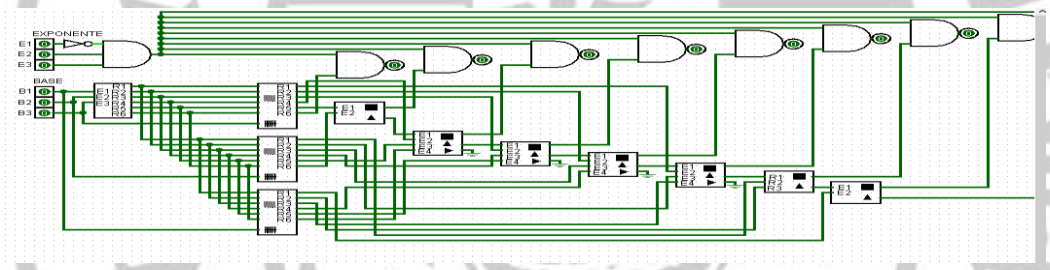
1	0	1	0	0	0	1	1	0	1	1	0
1	0	1	0	0	1	1	0	0	0	0	0
1	0	1	0	1	0	1	1	0	1	1	1
1	0	1	0	1	1	1	0	0	0	0	0
1	0	1	1	0	0	1	1	1	0	0	0
1	0	1	1	0	1	1	0	0	0	0	0
1	0	1	1	1	0	1	1	1	0	0	1
1	0	1	1	1	1	1	0	0	0	0	0
1	1	0	0	0	0	1	1	1	0	1	0
1	1	0	0	0	1	1	0	0	0	0	0
1	1	0	0	1	0	1	1	1	0	1	1
1	1	0	0	1	1	1	0	0	0	0	0
1	1	0	1	0	0	1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



Potencia P*P*P



Elección potencia P*P*P

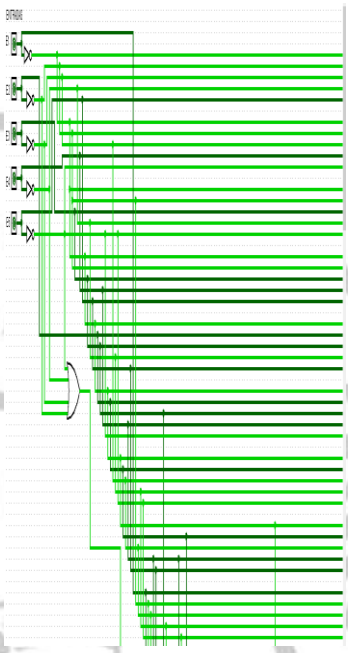


Para que escoja p*p*p pusimos un bit de control para 011 donde solo en ese bit se encienda

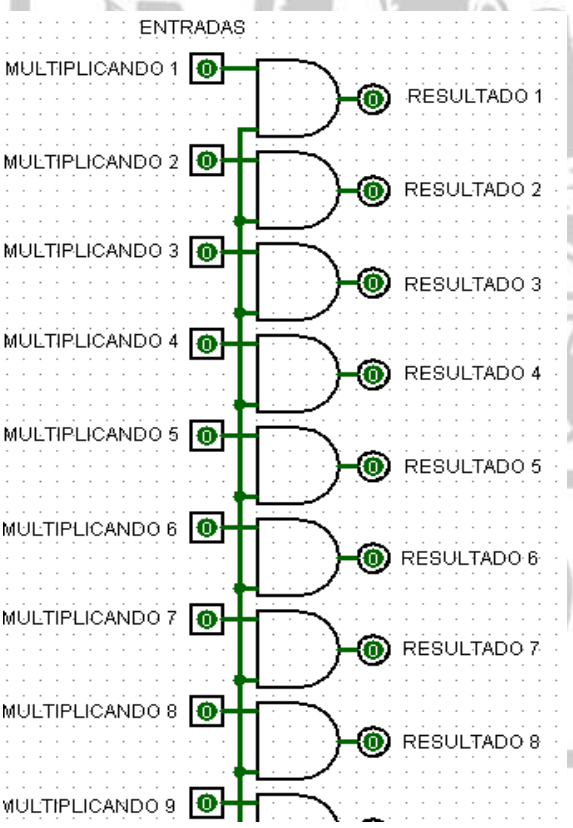
Sumador 5 Variables

V	W	X	Y	Z	Resultado	Acarreo1	Acarreo2
0	0	0	0	0	0	0	0
0	0	0	0	1	1	0	0
0	0	0	1	0	1	0	0
0	0	0	1	1	0	1	0
0	0	1	0	0	1	0	0
0	0	1	0	1	0	1	0
0	0	1	1	0	0	1	0
0	0	1	1	1	1	1	0
0	1	0	0	0	1	0	0
0	1	0	0	1	0	1	0
0	1	0	1	0	0	1	0
0	1	0	1	1	1	1	0

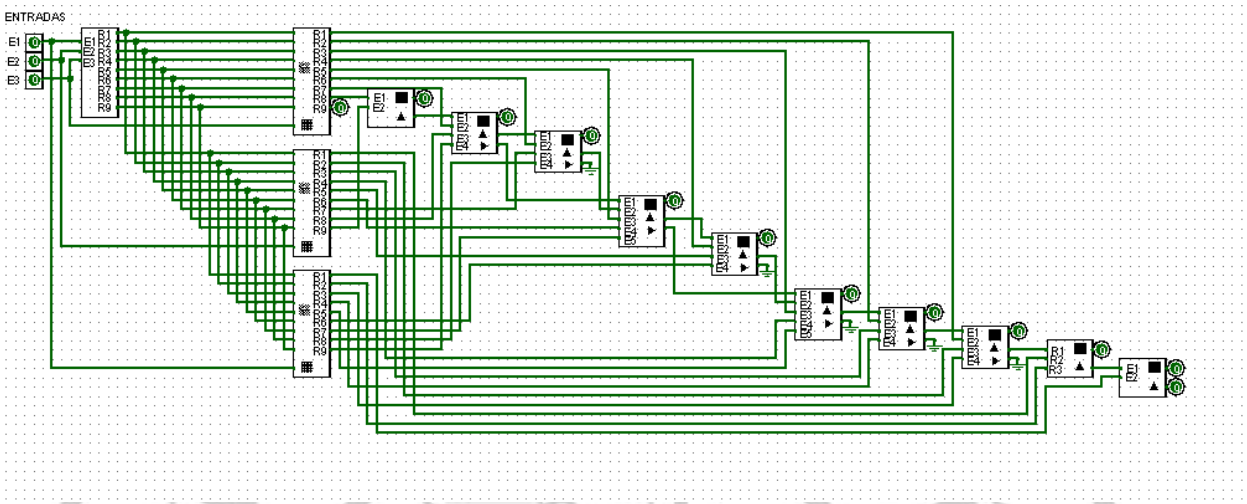
0	1	1	0	0	0	1	0
0	1	1	0	1	1	1	0
0	1	1	1	0	1	1	0
0	1	1	1	1	1	1	0
1	0	0	0	0	1	0	0
1	0	0	0	1	0	1	0
1	0	0	1	0	0	1	0
1	0	0	1	1	1	1	0
1	0	1	0	0	0	1	0
1	0	1	0	1	1	1	0
1	0	1	1	0	1	1	0
1	0	1	1	1	0	0	1
1	1	0	0	0	0	1	0
1	1	0	0	1	1	1	0
1	1	0	1	0	1	1	0
1	1	0	1	1	0	0	1
1	1	1	0	0	1	1	0
1	1	1	0	1	0	0	1
1	1	1	1	0	0	0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



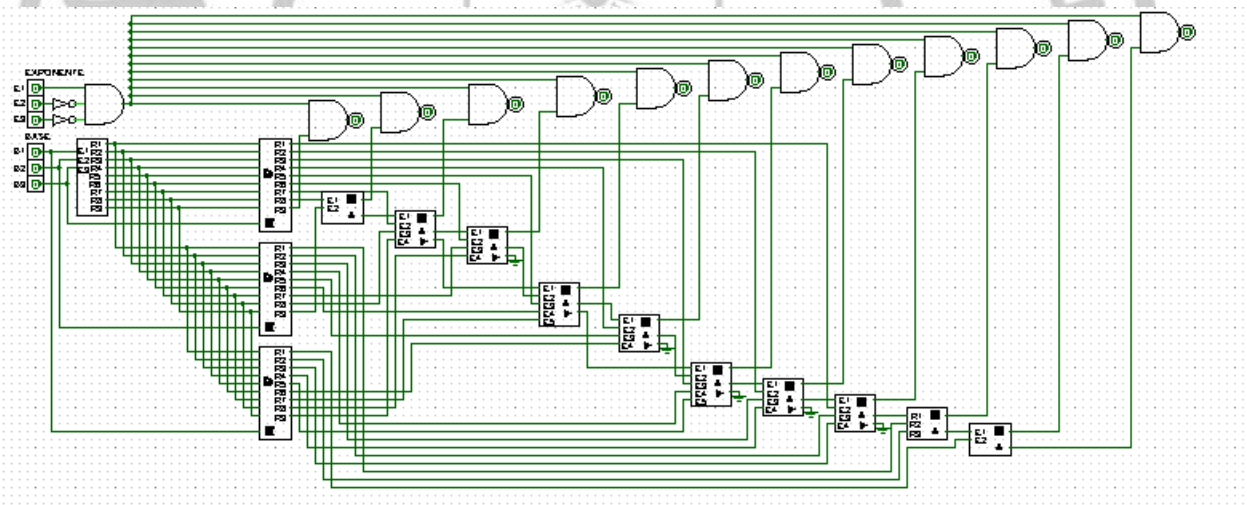
Multiplicando 9 * 1 multiplicador



Potenciador P*P*P*P

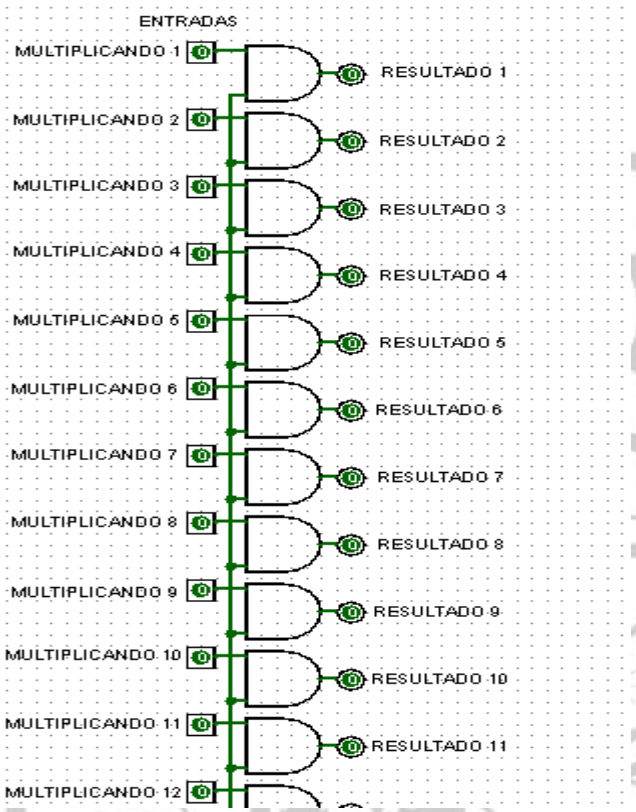


Elección Potencia P*P*P*P

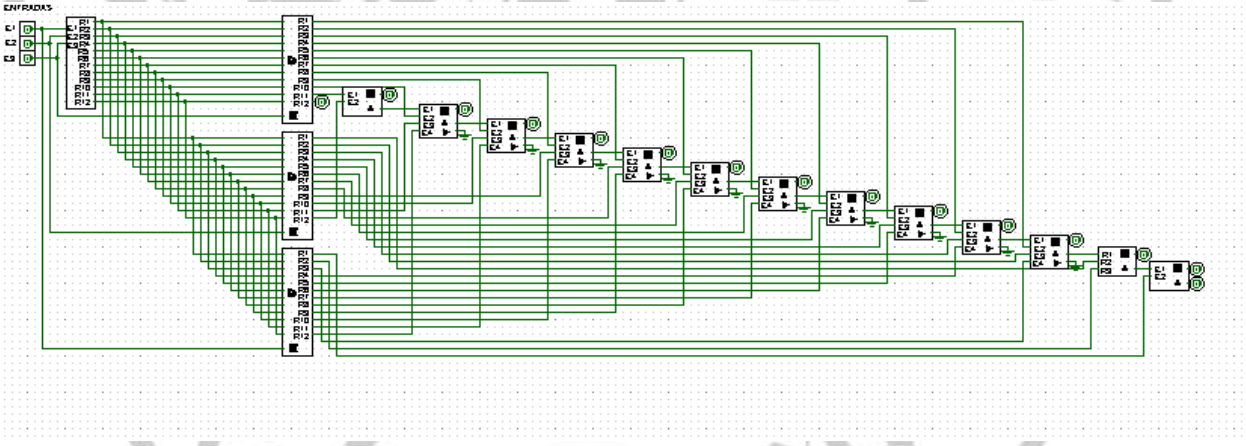


Hicimos un bit de control para que solo en 100 se encienda y escoja la potencia 4

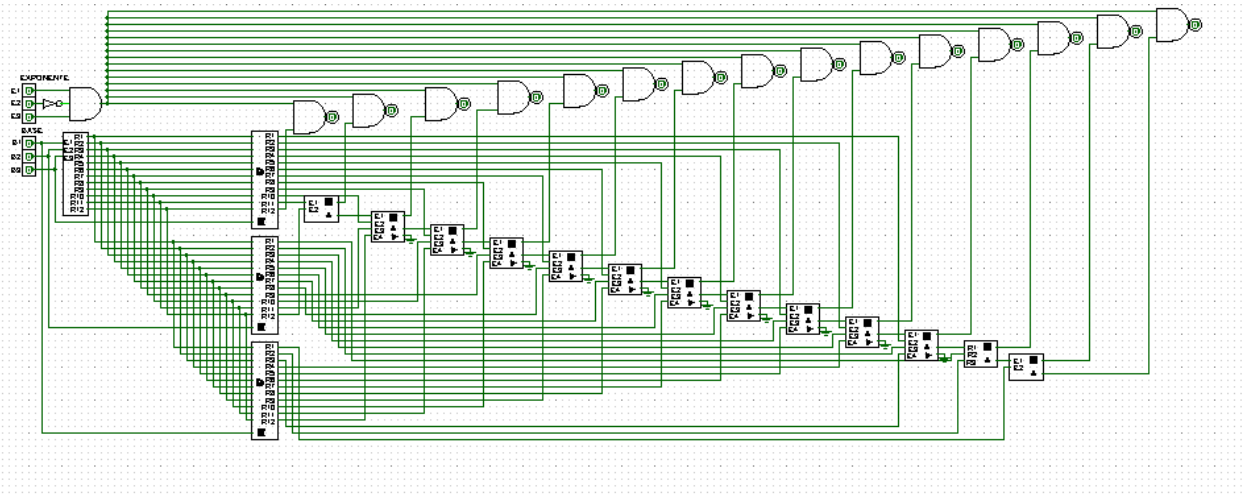
Multiplicando 12 * 1 Multiplicador



Potencia P*P*P*P*P

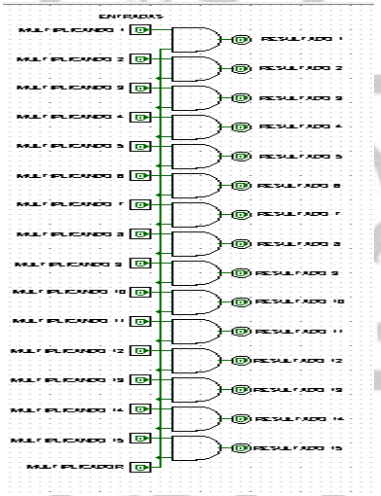


Elección potencia P*P*P*P*P

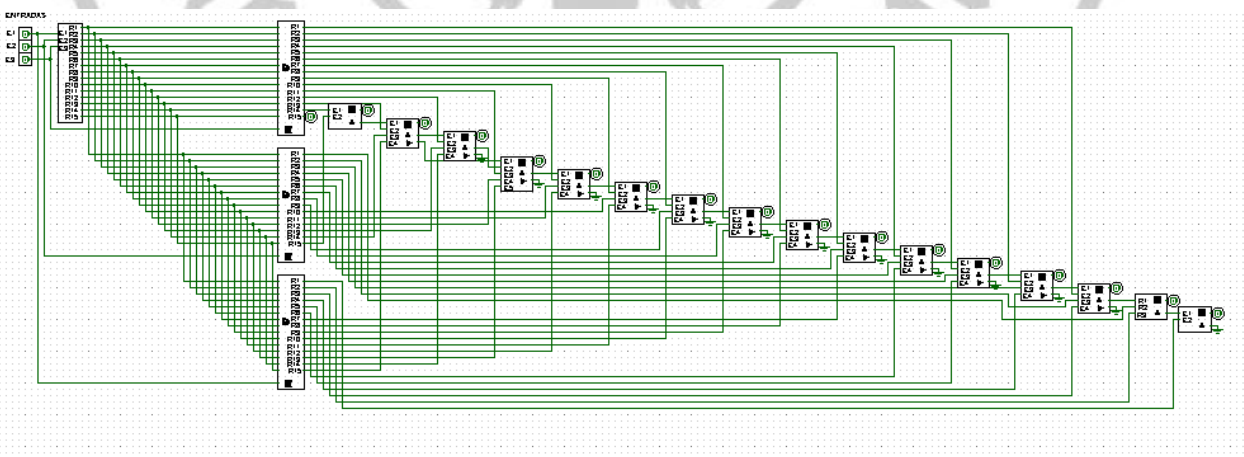


Bit de control donde solo se enciende cuando es 101

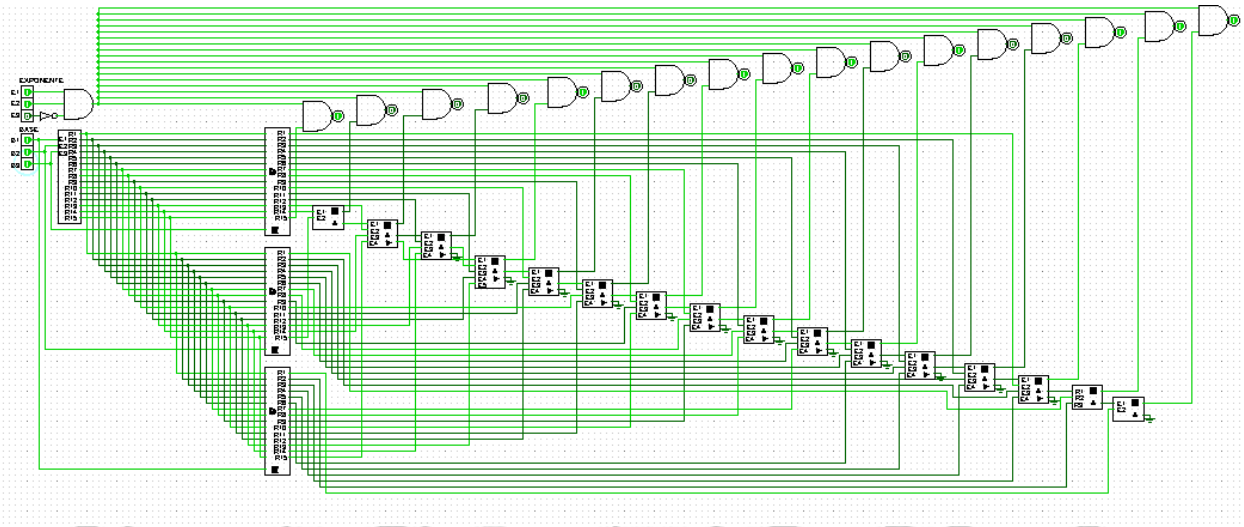
Multiplicador 15 * 1 multiplicando



Potencia P*P*P*P*P*P

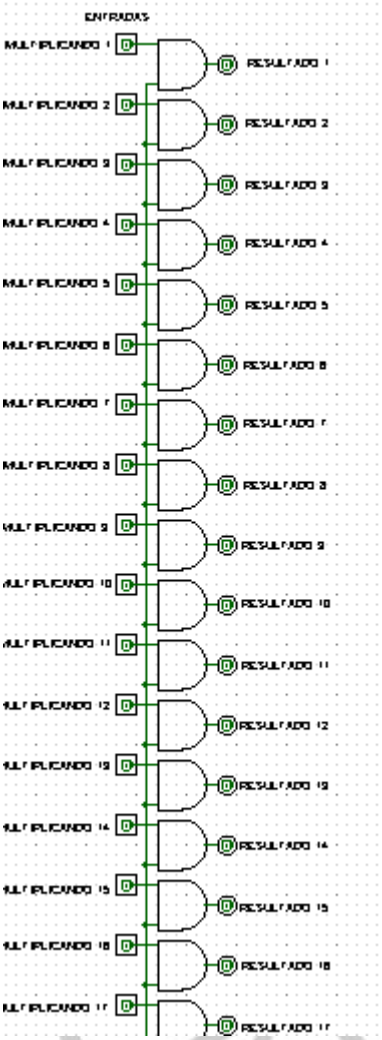


Elección potencia P*P*P*P*P*P*P

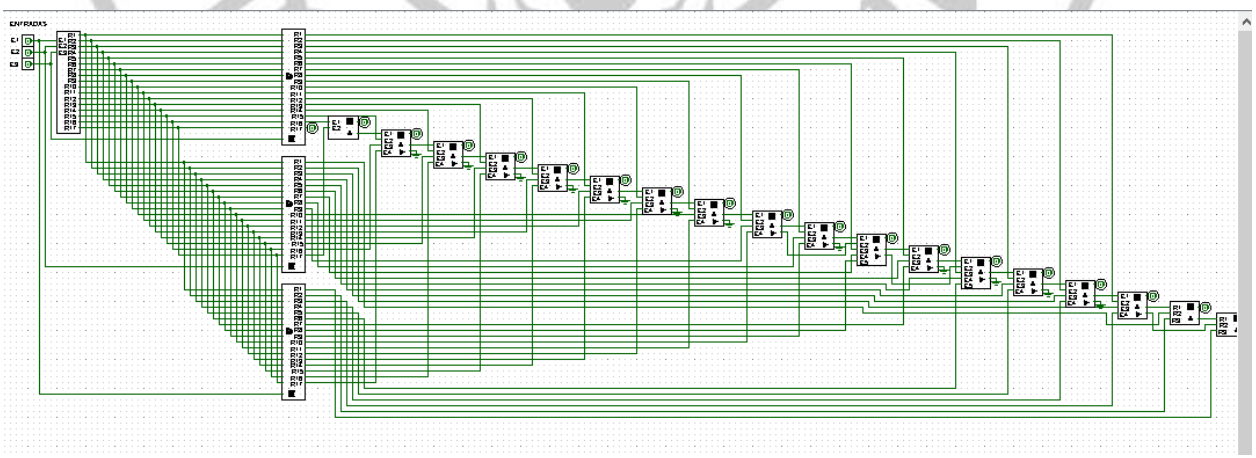


Bit de control donde funciona esta potencia cuando es 110 y los demás están apagados.

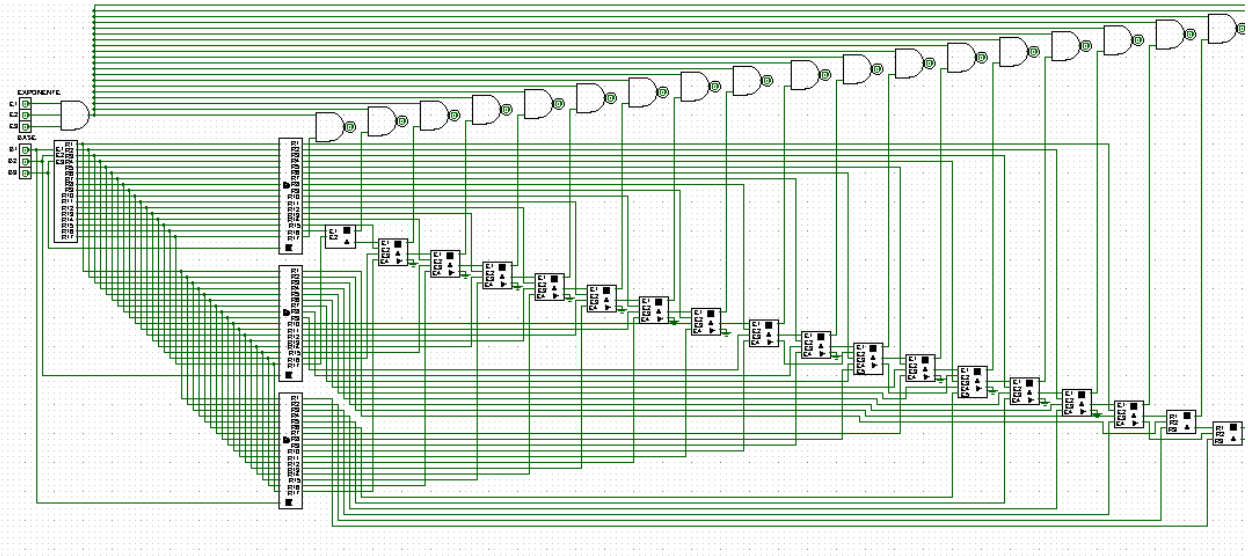
Multiplicador 17*1 multiplicando



Potencia P*P*P*P*P*P*P



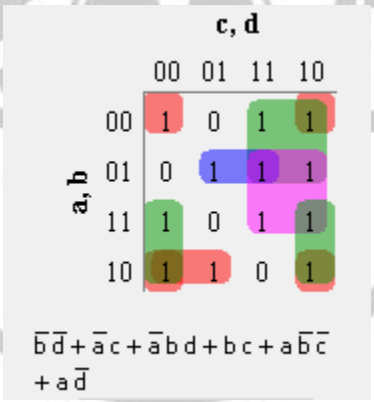
Elección Potencia P*P*P*P*P*P*P

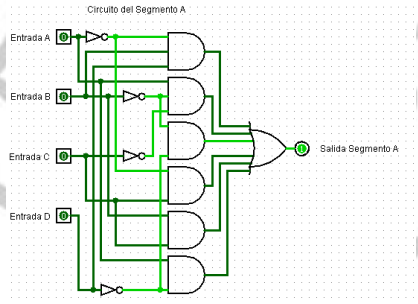


Un Bit de control donde solo se enciende cuando es 111 y todos los demás están apagados

Display segmento A

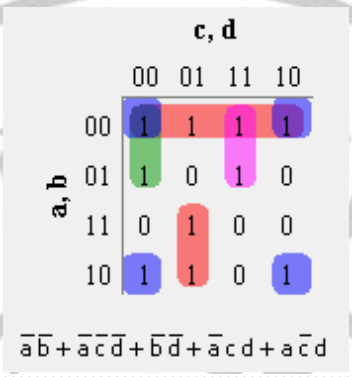
W	X	Y	Z	Resultado
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	0
1	1	0	0	1
1	1	0	1	0
1	1	1	0	1
1	1	1	1	1

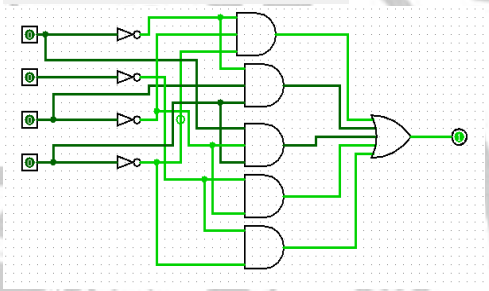




Display del segmento B

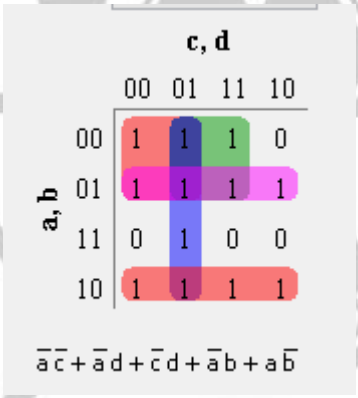
W	X	Y	Z	Resultado
0	0	0	0	1
0	0	0	1	1
0	0	1	0	1
0	0	1	1	1
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	0
1	1	0	0	0
1	1	0	1	1
1	1	1	0	0
1	1	1	1	0

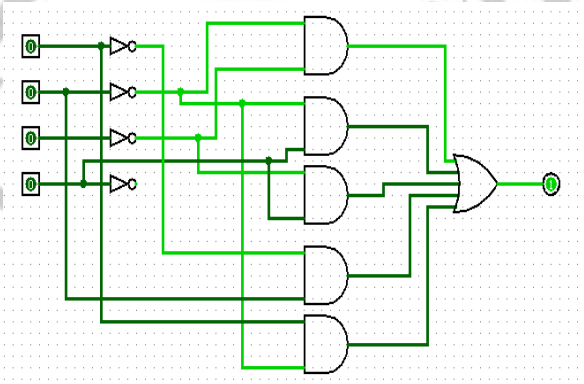




Display Segmento C

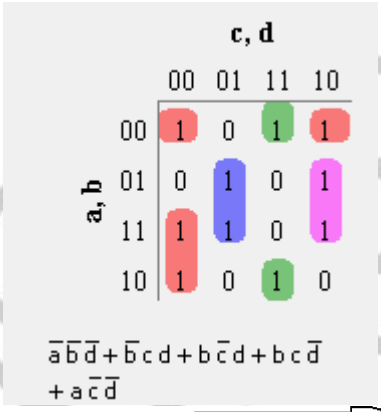
W	X	Y	Z	Resultado
0	0	0	0	1
0	0	0	1	1
0	0	1	0	0
0	0	1	1	1
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	0
1	1	1	1	0

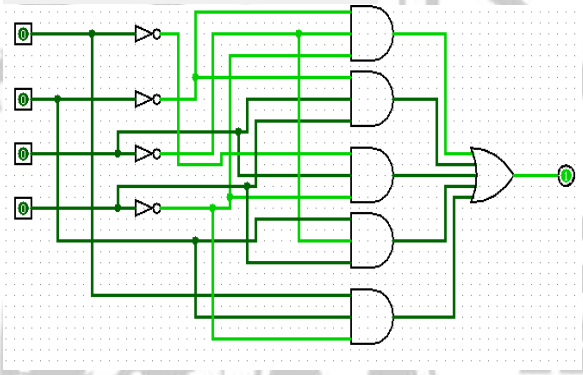




Display segmento D

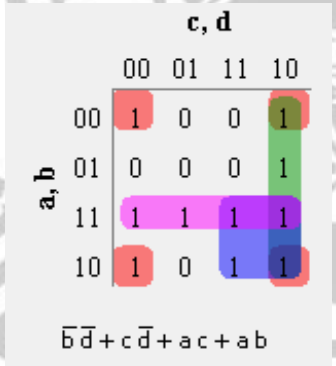
W	X	Y	Z	Resultado
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	0
1	0	0	0	1
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0

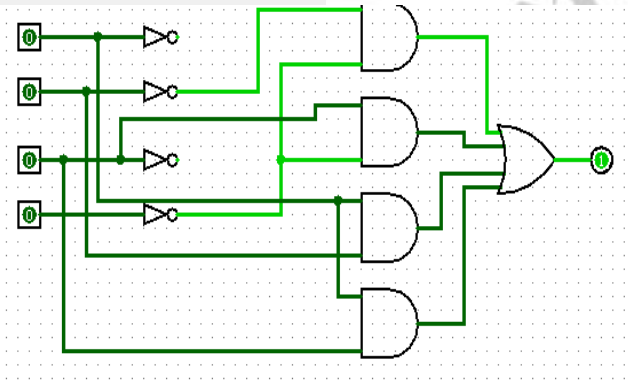




Display segmento E

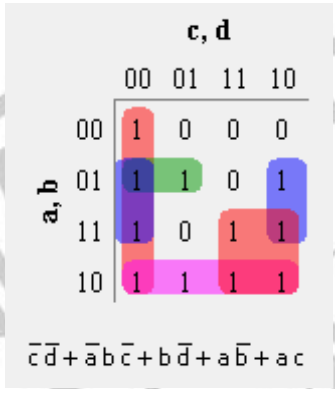
W	X	Y	Z	Resultado
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	1
0	1	1	1	0
1	0	0	0	1
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

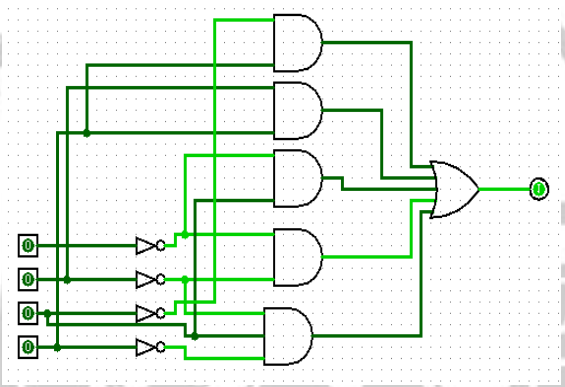




Display segmento F

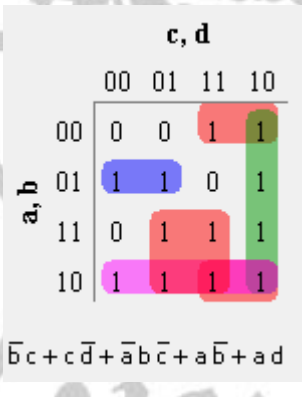
W	X	Y	Z	Resultado
0	0	0	0	1
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1
0	1	1	1	0
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	0
1	1	1	0	1
1	1	1	1	1

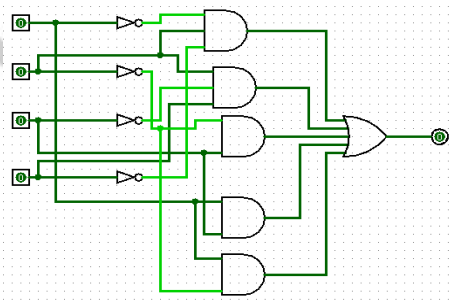




Display Segmento G

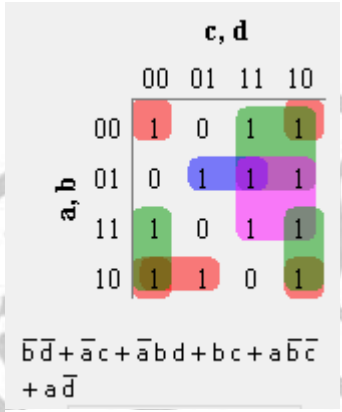
W	X	Y	Z	Resultado
0	0	0	0	0
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1
0	1	1	1	0
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

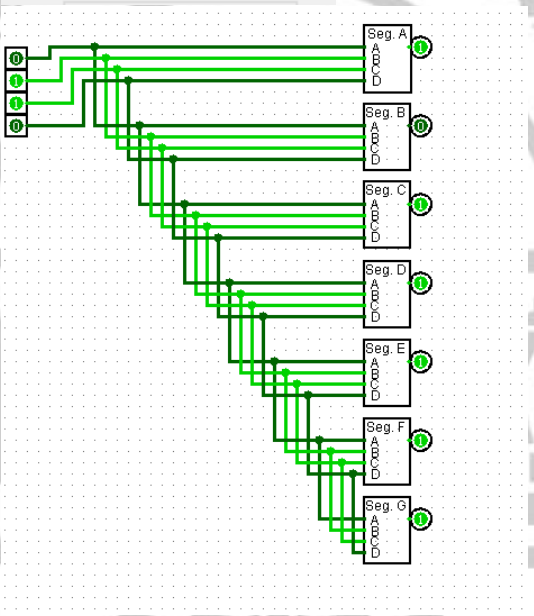




Display 7 Segmentos

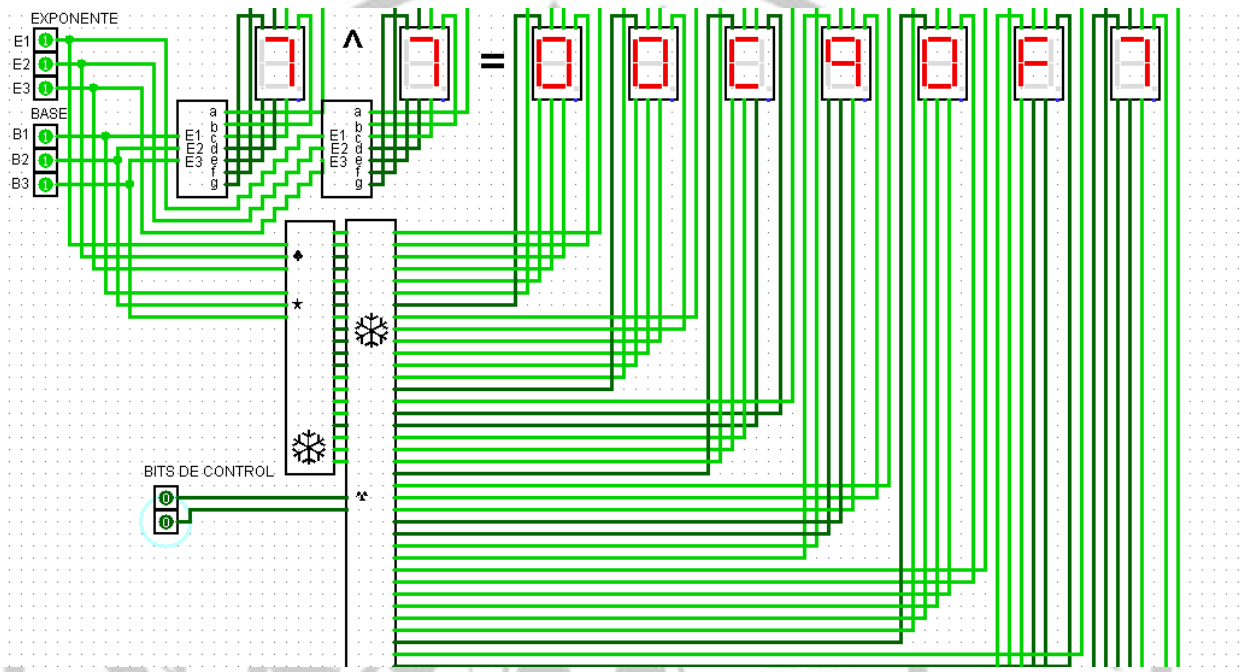
W	X	Y	Z	A	B	C	D	E	F	G
0	0	0	0	1	1	1	1	1	1	0
0	0	0	1	0	1	1	0	0	0	0
0	0	1	0	1	1	0	1	1	0	1
0	0	1	1	1	1	1	1	0	0	1
0	1	0	0	0	1	1	0	0	1	1
0	1	0	1	1	0	1	1	0	1	1
0	1	1	0	1	0	1	1	1	1	1
0	1	1	1	1	1	1	0	0	0	0
1	0	0	0	1	1	1	1	1	1	1
1	0	0	1	1	1	1	0	0	1	1
1	0	1	0	1	1	1	0	1	1	1
1	0	1	1	0	0	1	1	1	1	1
1	1	0	0	1	0	0	1	1	1	0
1	1	0	1	0	1	1	1	1	0	1
1	1	1	0	1	0	0	1	1	1	1
1	1	1	1	1	0	0	0	1	1	1





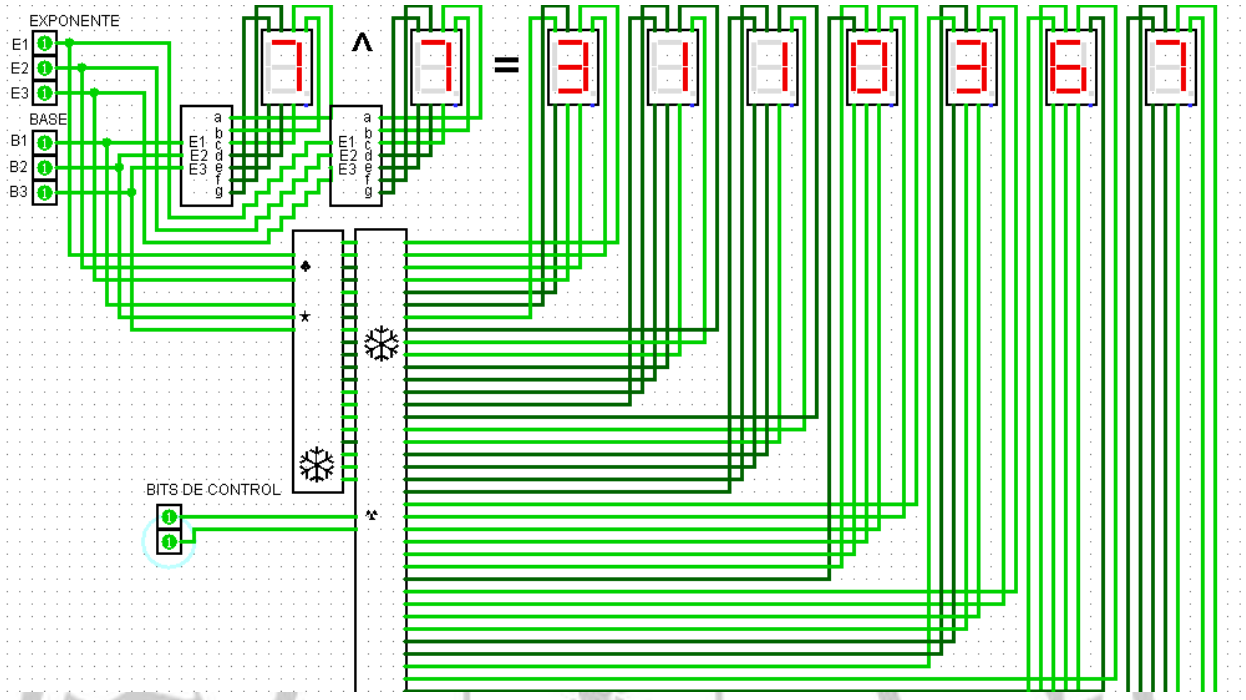
Diseño del circuito

Main de hexadecimal, octal y decimal



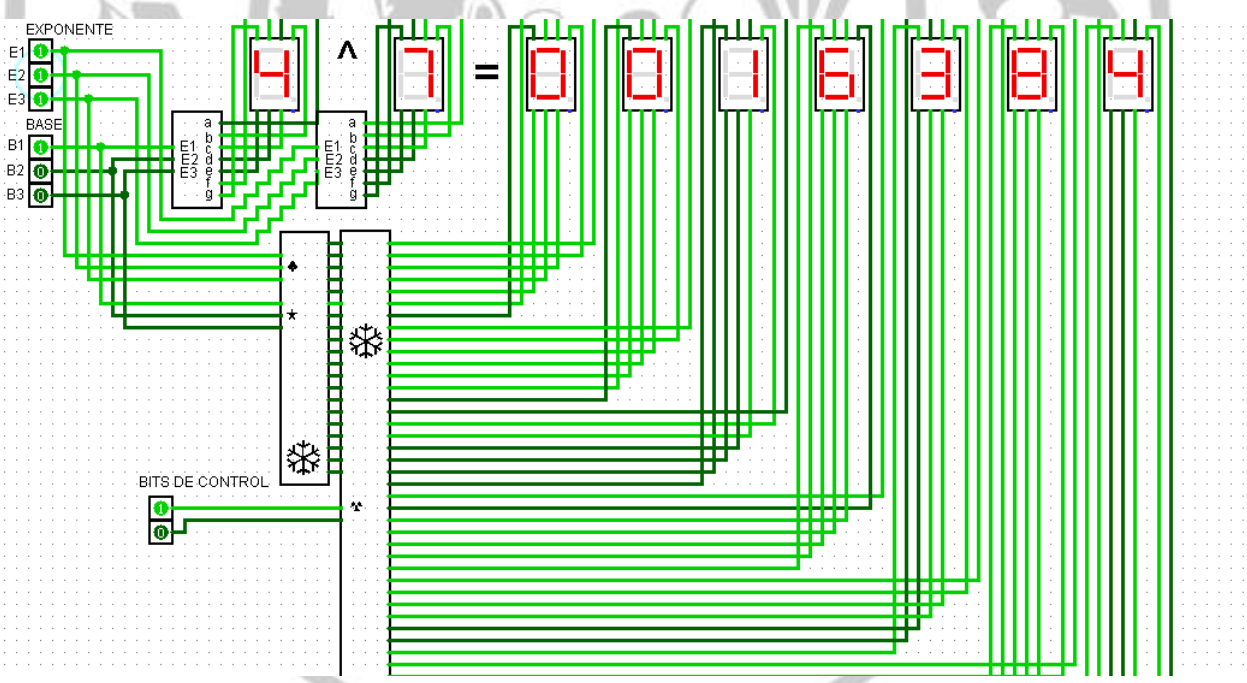
Para poner el circuito en hexadecimal poner el bit de control en 00

Sistema octal



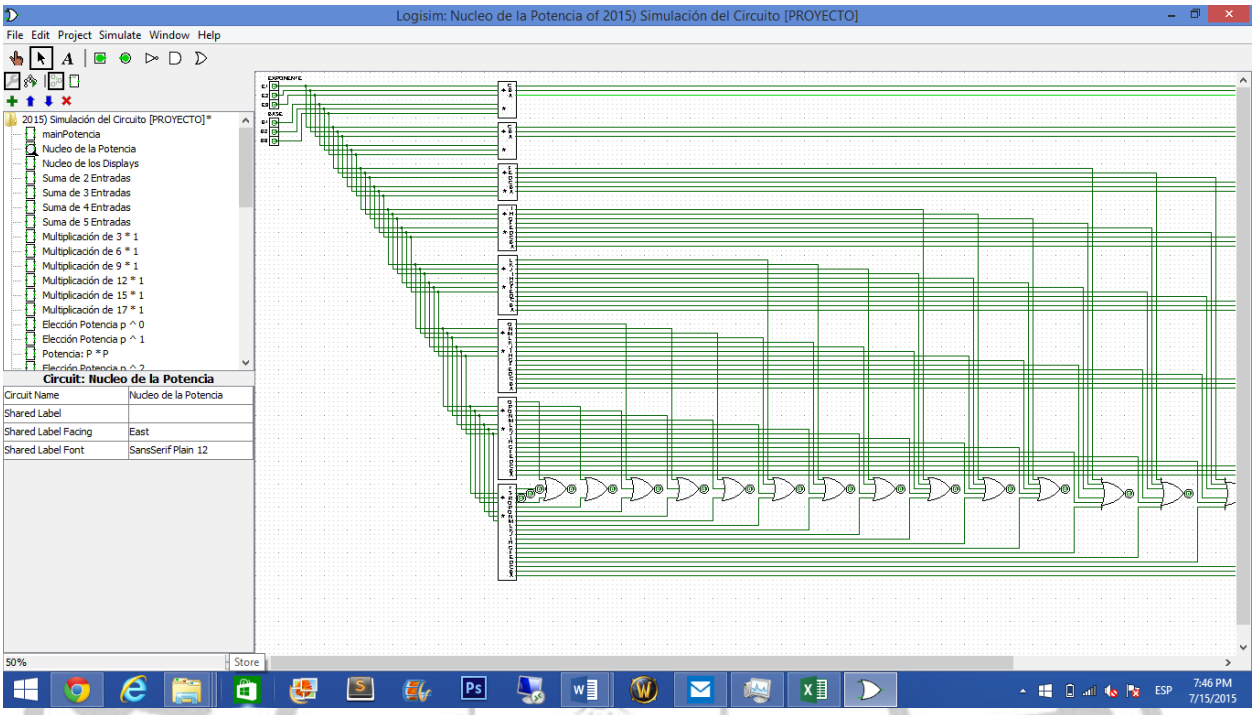
Para poner el circuito en sistema octal cambiar el bit de control a 11

Sistema decimal

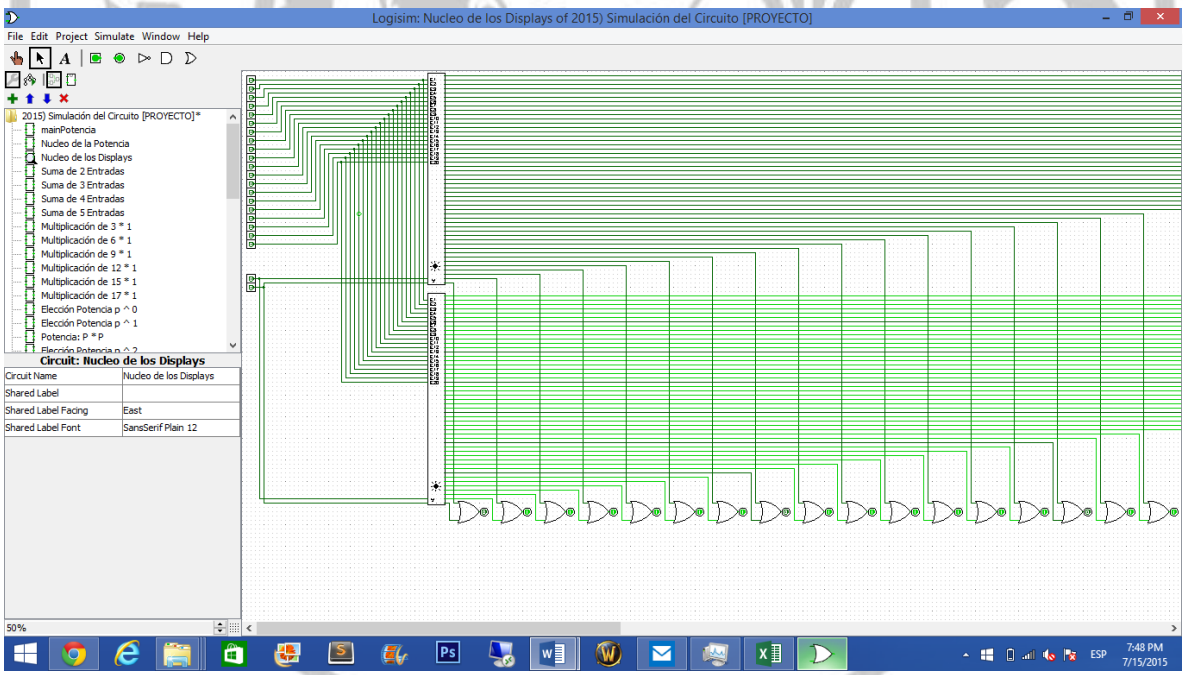


Para poner el circuito en decimal poner el bit de control en 10

Núcleo de la potencia



Núcleo del display



Simulación del circuito

20150715_2044_34.
avi