10-2021 GRADO EN INGENIERÍA INFORMÁTICA

05GIIN
FUNDAMENTOS DE COMPUTADORES

ACTIVIDAD EVALUATIVA 2

09/12/2021

Juan Diego Tomás Nolan

Miguel Ángel Gagliardo

Salceda Fernández-Barredo del Amo

Verónica Romero Márquez

ÍNDICE

1. Introducción

Un sistema combinacional es todo sistema digital en el que sus salidas son función exclusiva del valor de sus entradas en un momento dado, sin que intervengan en ningún caso estados anteriores de las entradas o de las salidas. Las funciones booleanas –compuestas por operadores OR, AND, NAND, XOR– se pueden representar íntegramente mediante una tabla de verdad. Por tanto, carecen de memoria y de retroalimentación.

Existen una serie de circuitos combinacionales que son muy comunes y aparecen o bien aisladamente o formando parte de otros circuitos más complejos de aplicación general, y que se repiten un número de veces tan considerable que se hace aconsejable su fabricación en serie.

Entre los varios elementos que existen, en la práctica que vamos a desarrollar usaremos los siguientes:

- Codificador
- Sumador
- Conversor
- Comparador

2. Objetivos

Diseñar un sistema combinacional utilizando tablas de la verdad, álgebra de Boole y mapas de Karnaugh e implementar el sistema con la ayuda de la herramienta Logisim.

El sistema digital que vamos a diseñar tiene el objetivo de visualizar el resultado de una pelea de boxeo mediante displays de siete segmentos e indicar mediante un LED quién ha ganado el combate.

La puntuación será dada por 3 jueces mediante dos teclados por juez, pudiendo dar una puntuación de 0 a 4 puntos por boxeador. En caso de empate dispondrán de dos pulsadores adicionales para determinar quién es el ganador.

3. Desarrollo

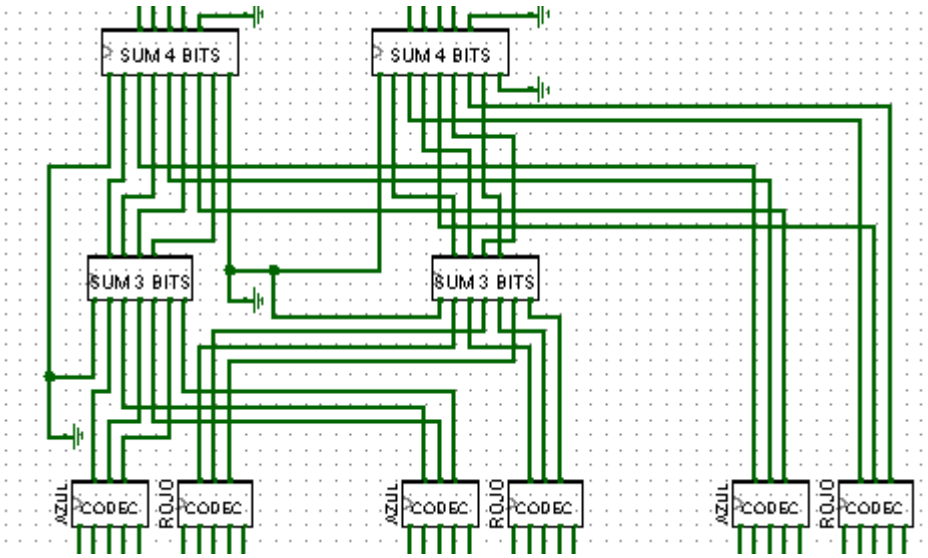
Para implementar el diseño deberemos en primer lugar convertir las señales de los pulsadores dadas por los jueces de sistema decimal a binario mediante codificadores para a continuación poder usar sumadores.

Una vez obtenido el resultado del sumador, por un lado, lo tenemos que convertir el código binario en código BCD, y éste a su vez en código 7 segmentos, para conectarlo a un display LED de 7 segmentos. Por otro lado, con el resultado de la suma en binario deberemos usar un comparador, para reflejar mediante un LED cuál de las dos puntuaciones es mayor.

Finalmente usaremos la señal del pulsador extra de los jueces para introducirla en el comparador, en caso de empate.

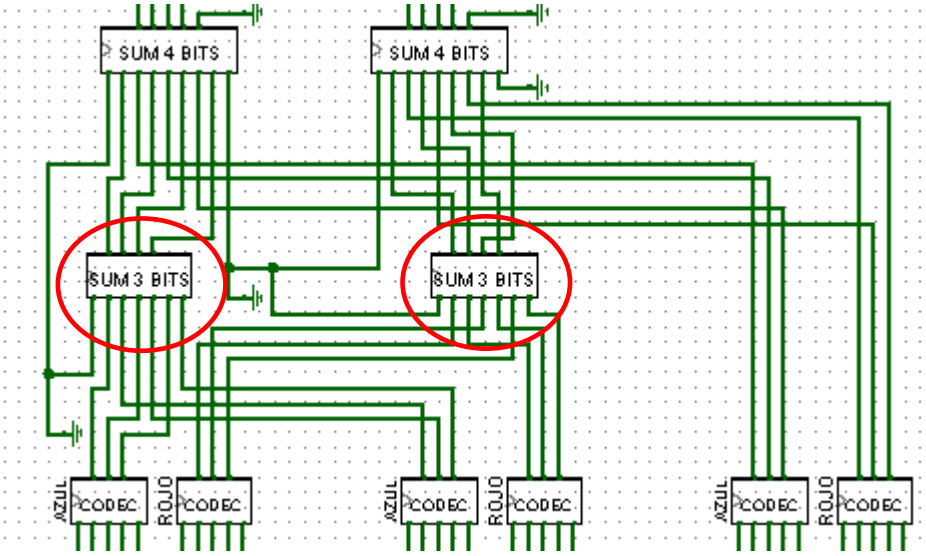
A continuación, detallaremos la tabla de la verdad y los mapas de Karnaugh o el álgebra de Boole empleado para cada uno de los elementos anteriores: codificadores, sumadores, conversores, y comparador.

El circuito se ha diseñado de la siguiente manera:



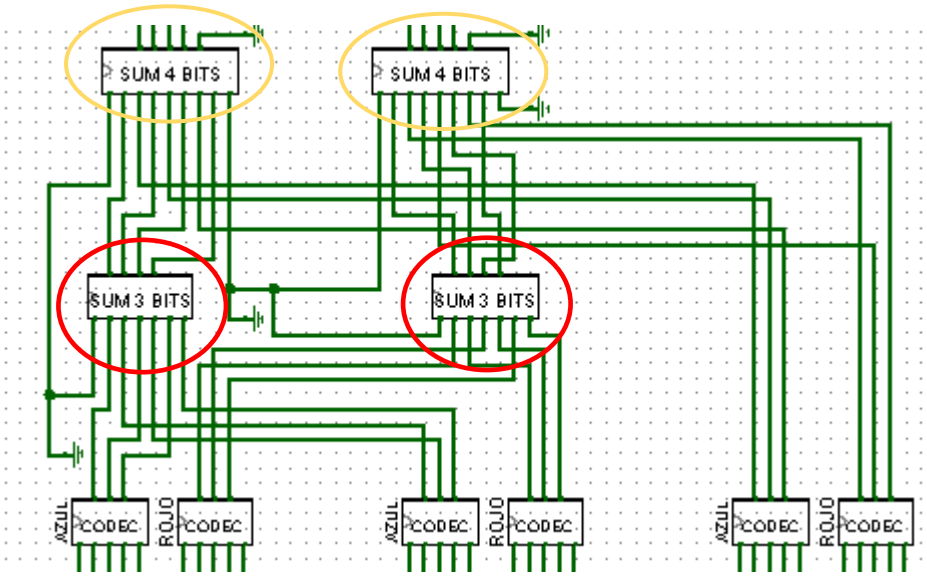
El bloque del CODIFICADOR se ha repetido dos veces por cada Juez, uno para el boxeador rojo, y otro para el boxeador azul. Cada codificador admite 5 entradas, de 0 a 4 con prioridad sobre la puntuación más alta (en caso de que se pulse accidentalmente dos teclas a la vez, prevalece la puntuación más alta). A su vez, cada codificador tiene 3 salidas, que se corresponden con un número binario de 3 bits, que codifica hasta el número 7 en base decimal, aunque cada juez solamente puede puntuar has el número 4.

Las 3 puntuaciones de cada boxeador correspondientes a cada juez (18 salidas en total), van a dos sumadores de 3 bits (círculo rojo), distribuidos de la forma siguiente: en uno de los dos sumadores de 3 bits entran 6 bits, 3 de cada juez para un solo boxeador (boxeador rojo). En el otro sumador de 3 bits ocurre lo mismo para el boxeador azul.



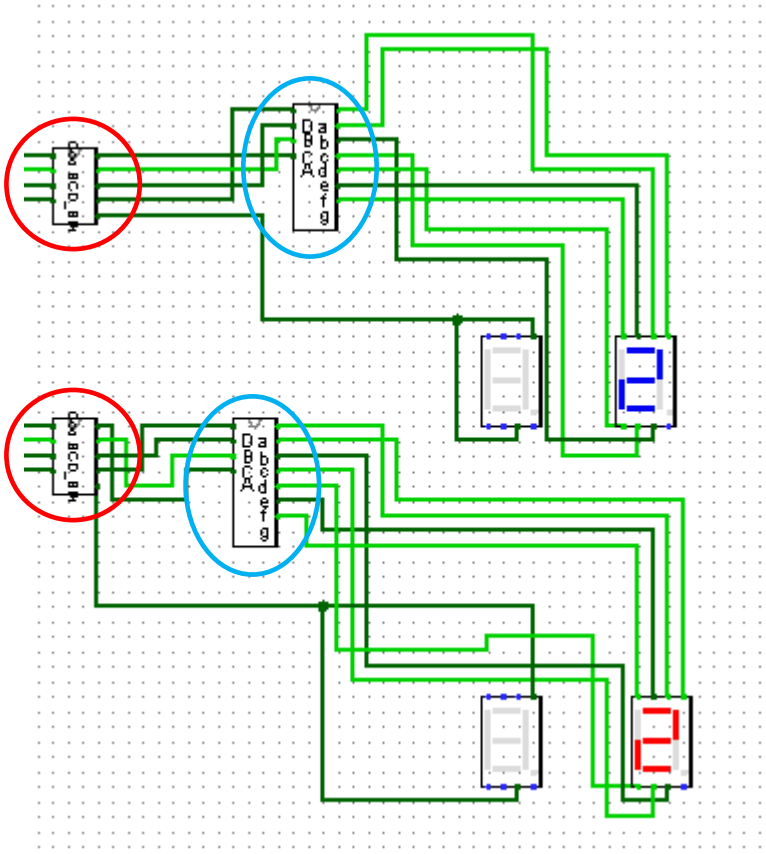
Las salidas de los dos sumadores de 3 bits más el resultado para cada boxeador del tercer juez, entran en dos sumadores de 4 bits (círculo amarillo), que se distribuyen de la siguiente forma:

En el sumador de la izquierda del dibujo, entran el resultado de la suma del boxeador azul de los dos primeros jueces, sumada a la puntuación del boxeador azul del tercer juez. En el sumador de la derecha, entran las dos puntuaciones de los dos primeros jueces del boxeador rojo, más la puntuación del tercer juez para el boxeador rojo.

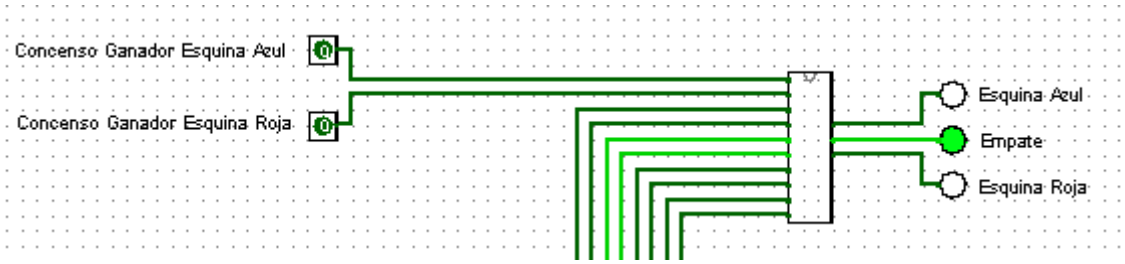


Las salidas para los dos sumadores de 4 bits, son dos números binarios de 4 bits que corresponden al boxeador azul y al rojo, y que a su vez entrarán cada uno en un conversor binario a BCD (círculo rojo), ya que necesitamos codificación BCD para poder iluminar los displays de 7 segmentos.

De este conversor binario a BCD se conectan las 4 de las 5 salidas al conversor BCD de 7 segmentos (círculo azul), que mostrarán la puntuación en código decimal en el display. La quinta salida que pertenece al bit de mayor peso, va directamente a otro display habilitado para las decenas. Como la puntuación máxima para cada jugador es de 12, el display de las decenas no pasará de 1, por lo que las entradas de éste están marcadas para que se iluminen los segmentos b y c en caso de que haya un 1 en esa quinta salida.



Los 4 bits de salida del sumador de 4 bits que corresponden a la puntuación del boxeador rojo y azul, también van a un comparador (siguiente figura). Las 3 salidas del comparador marcarán A<B que gana la esquina roja, y viceversa para la esquina azul. También hay una salida de empate, A=B. Además de las entradas de los sumadores, se han añadido dos entradas más al comparador, que son las entradas de consenso para cada jugador, y que se deciden de forma unitaria ante un eventual empate.



CODIFICADORES:

Tenemos que generar 2 codificadores por juez, uno para puntuar el boxeador azul y otro para el boxeador rojo. Como son tres jueces habrá que generar en total 6 codificadores idénticos.

Cada codificador tendrá 5 entradas que corresponderán a la puntuación asignada a los boxeadores y tres salidas, ya que la puntuación máxima que puede dar cada juez será de 4 (el número 4 en binario es “100”, con lo cual necesitaremos 3 bits).

Para empezar, generaremos la tabla de la verdad con prioridad. Como la tecla de mayor peso es la A y la de menor peso es la E, no nos afecta el valor de la tabla que encontremos detrás de los 1, así que pondremos una X y podremos obtener la siguiente tabla simplificada:

ENTRADAS					SALIDAS		
A	**B**	**C**	**D**	**E**	**S3**	**S2**	**S1**
0	0	0	0	0	X	X	X
0	0	0	0	1	0	0	0
0	0	0	1	X	0	0	1
0	0	1	X	X	0	1	0
0	1	X	X	X	0	1	1
1	X	X	X	X	1	0	0

Las fórmulas sin simplificar extraídas de la tabla anterior serían las siguientes:

S3: A (MSB)

$S2: \bar{A}\bar{B}C + \bar{A}B$

$S1: \bar{A}\bar{B}\bar{C}D + \bar{A}B$ (LSB)

Utilizaremos el álgebra de Boole para simplificar las fórmulas de S2 y S1:

$S2 = \bar{A}(\bar{B}C + B)$ Aplicamos ley distributiva para 'A. Después aplicamos teorema de consenso $(\bar{B}C + B) = C + B$

$S2 = \bar{A}(C + B)$

$S1 = \bar{A}\bar{B}\,\bar{C}\,D + \bar{A}B$ Aplicamos ley distributiva $\bar{A}(\bar{B}\,\bar{C}\,D + B)$ aplicamos Teorema de consenso $(\bar{B}\bar{C}D + B) = B + \bar{C}D$

$S1 = \bar{A}(\bar{C}\,D + B)$

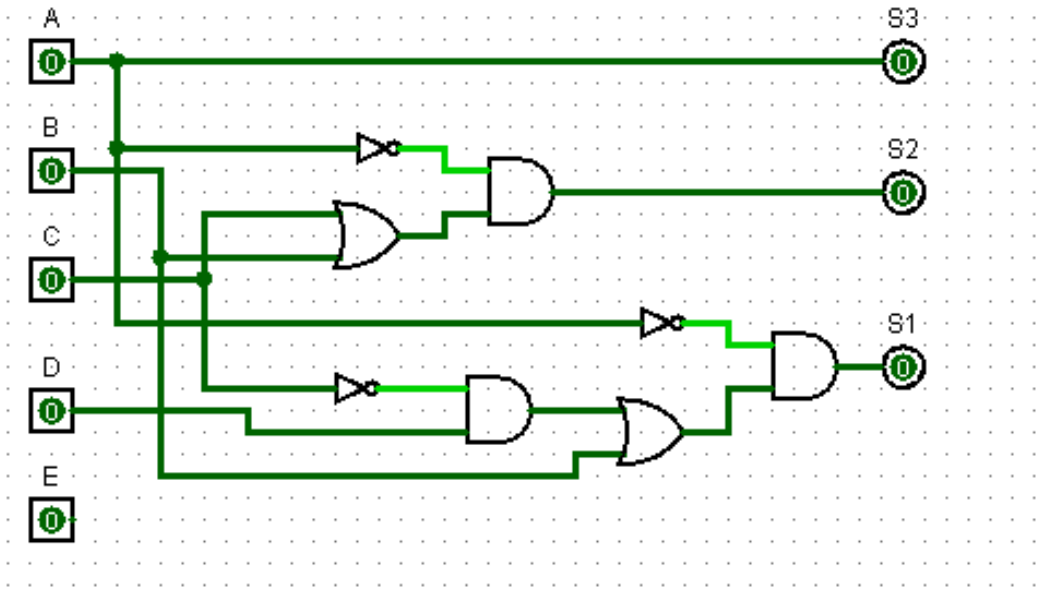
Las fórmulas simplificadas quedarían así:

$S3 = A$

$S2 = \bar{A}\,(C + B)$

$S1 = \bar{A}\,(\bar{C}\,D + B)$

El circuito generado por las ecuaciones anteriores sería el siguiente:



SUMADORES:

Para simplificar el diseño de cada sumador de 9 entradas y 4 salidas, se han diseñado primeramente dos sumadores de 6 entradas (llamados SR-6 y SA-6), compuestos a su vez por dos sumadores de 3 dígitos en paralelo, uno para el boxeador rojo y otro para el boxeador azul, que suman las puntuaciones de dos de los tres jueces y cuyo resultado envían a los sumadores que se describen a continuación. Para recoger la suma de los tres jueces, se han diseñado dos sumadores de 9 entradas y cuatro salidas (SR-9 y SA-9), conectados en su entrada, a la salida de los sumadores SR-6 y SA-6, y a la salida del codificador del tercer juez, cuya puntuación de 3 dígitos para cada boxeador entra en el SR-9 o el SA-9, según corresponde al boxeador rojo o al azul.

Con el fin de no generar una tabla de la verdad de 26 combinaciones, se ha optado por realizar 3 sumadores en paralelo, uno para cada posición de los 3 dígitos (A2 con B2 e igual para A1 y A0). El sumador A0 B0 genera un S0 que corresponde al LSB y un acarreo, que se introduce en el sumador A1B1, para dar un S1 junto con un acarreo de salida, que será el acarreo de entrada para el sumador A2B2 (MSB inicial) que finalmente dará un S2 con un acarreo de salida que será el MSB del producto de la suma, siendo éste de 4 bits.

Para el sumador de A0B0 (semisumador), la tabla de la verdad es:

A0	B0	C_{out}	S0 (LSB)	número binario resultante
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	0	10

Para el resto de los sumadores (sumadores completos), la tabla sigue el mismo esquema que se muestra a continuación, por ejemplo, para A1B1:

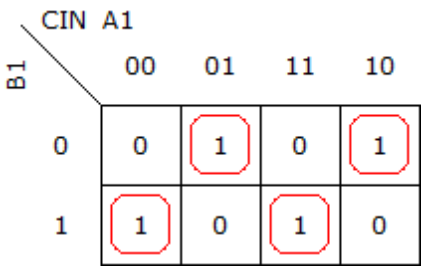
A1	B1	C_{in}	S1	C_{out}	número binario resultante
0	0	0	0	0	0
0	0	1	1	0	1
0	1	0	1	0	1
0	1	1	0	1	10
1	0	0	1	0	1
1	0	1	0	1	10
1	1	0	0	1	10
1	1	1	1	1	11

El mapa de Karnaugh para esta tabla es el siguiente:

Para S1: no se puede minimizar más, la expresión que queda es:

$S1 = (A1 \oplus B1) \oplus (C_{in})$

(para negar el término, se usa la comilla delante de éste. Así, A negada se escribirá como ‘A)



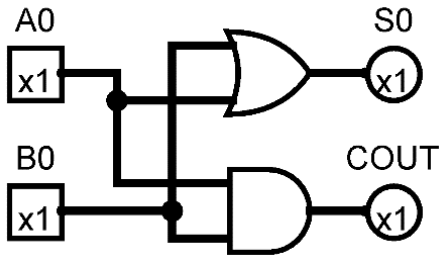
Para C_{out}, la tabla es:

CIN A1 / B1	00	01	11	10
0	0	0	1	0
1	0	1	1	1

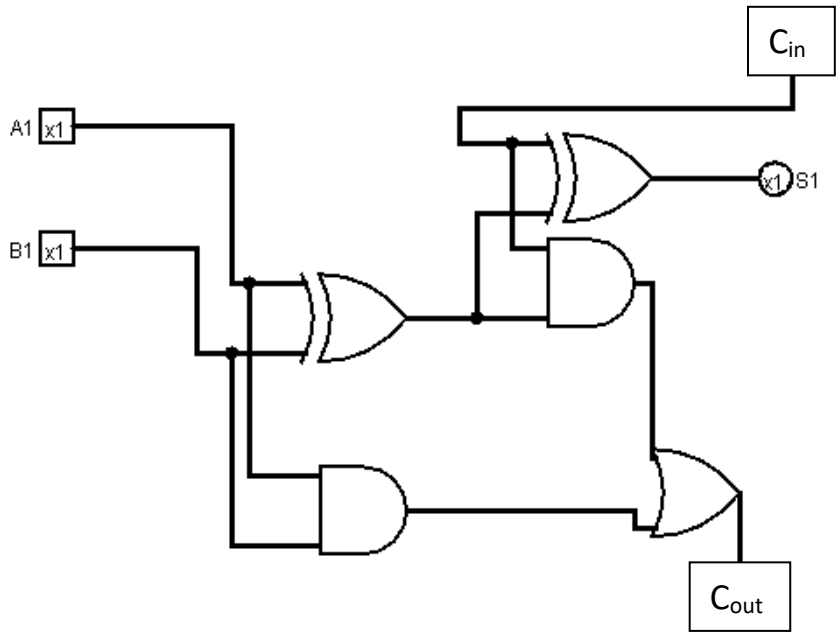
y la expresión que resulta del mapa es:

$C_{out} = C_{in}\,(A1 \oplus B1) + (A1\ B1)$

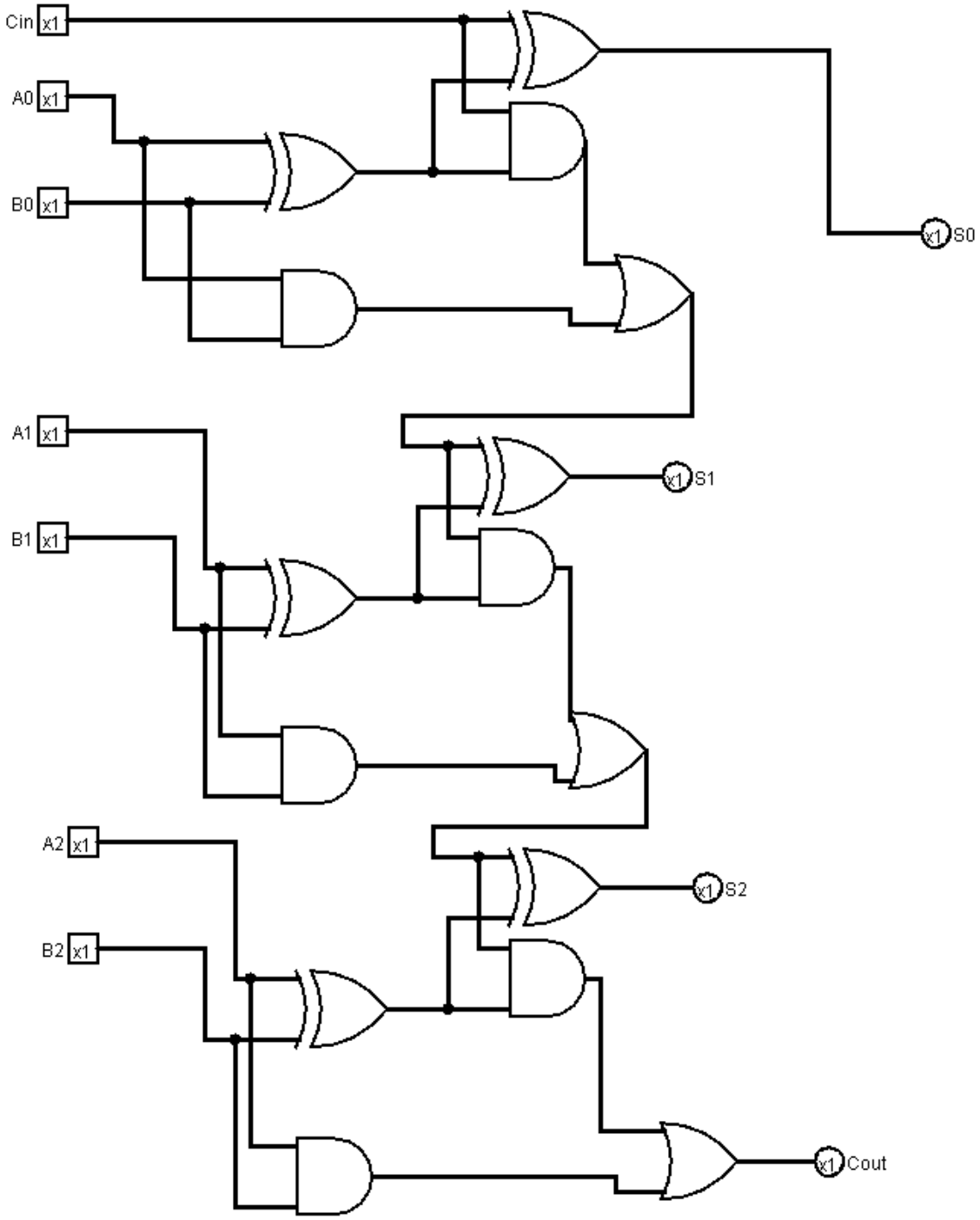
El circuito que resulta del primer sumador (semisumador A0B0), es:



Los siguientes circuitos de sumadores completos por separado para cada bit, serían:



Finalmente, el sumador de 3 bits que es la unidad que hemos usado queda reflejado como circuito de la siguiente manera, usando el C_{out} del sumador anterior de los bits de menor peso como C_{in} del sumador de los siguientes bits de más peso:

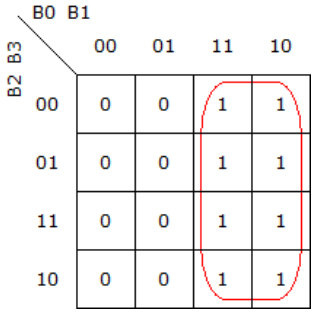


CONVERSORES:

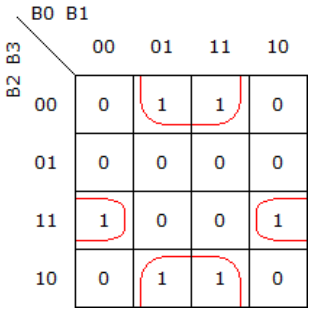
Tenemos que crear dos conversores (para el resultado de cada suma, que está en código binario) uno para pasar el **código binario a código BCD,** y el otro para convertir el **BCD a 7 segmentos**. Cada conversor tendrá 4 entradas (B3 ES EL LSB Y B0 EL MSB) y 5 salidas (C4 ES EL LSB Y C0 EL MSB). Para ello generaremos la siguiente tabla de la verdad para el **conversor de código binario a código BCD**:

ENTRADAS				SALIDAS				
B0	**B1**	**B2**	**B3**	**C0**	**C1**	**C2**	**C3**	**C4**
0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1	0
0	0	1	0	0	0	1	0	0
0	0	1	1	0	1	0	0	1
0	1	0	0	0	1	0	0	0
0	1	0	1	0	0	0	0	1
0	1	1	0	0	1	1	0	0
0	1	1	1	0	0	1	0	1
1	0	0	0	1	0	0	0	0
1	0	0	1	1	0	0	1	0
1	0	1	0	1	0	1	0	0
1	0	1	1	1	1	0	0	1
1	1	0	0	1	1	0	0	0
1	1	0	1	1	0	0	0	1
1	1	1	0	1	1	1	0	0
1	1	1	1	1	0	1	0	1

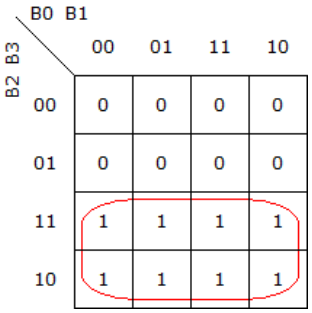
Los mapas de Karnaugh que usaremos y las expresiones resultantes quedarían de la siguiente manera:



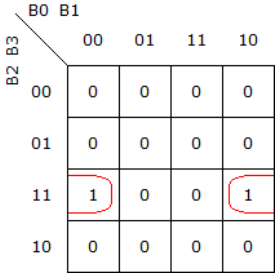
C0 = B0



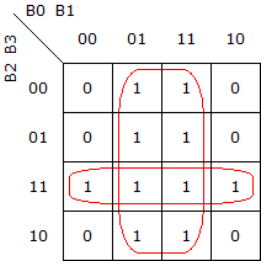
$$C1 = \overline{B1}B2B3 + B1\overline{B3}$$



$$C2 = (B2 + \overline{B3})\,B2$$

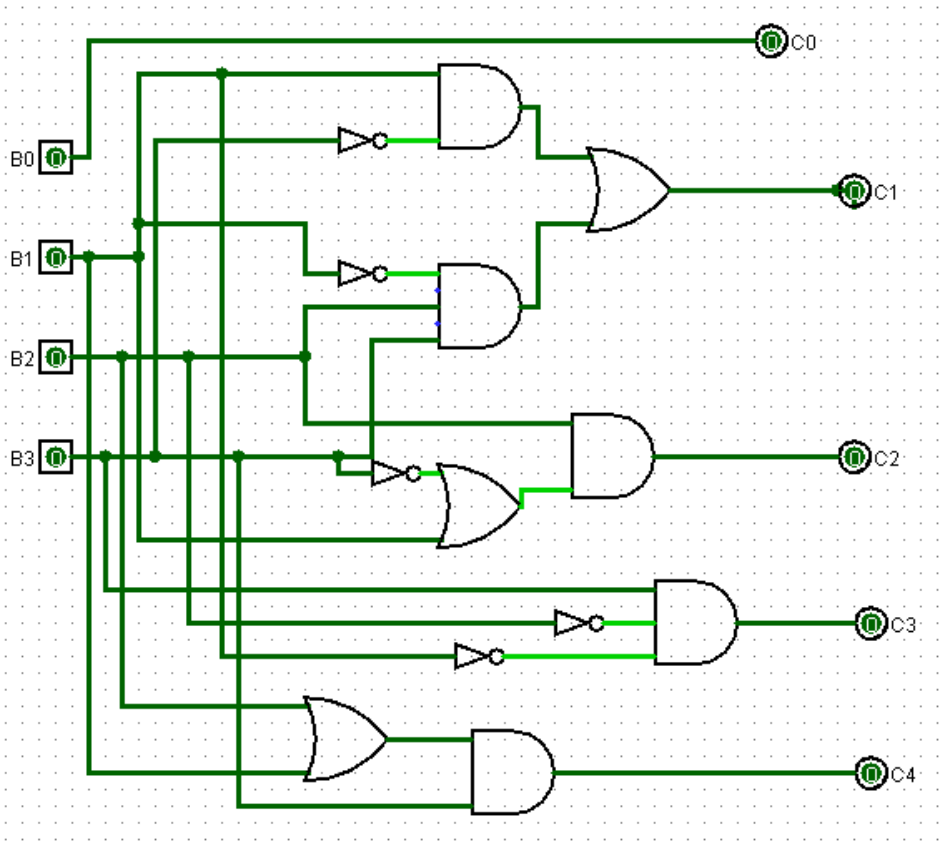


$$C3 = \overline{B1}\,\overline{B2}B3$$



$$C4 = B2B3 + B1$$

El circuito generado por las ecuaciones anteriores sería el siguiente:



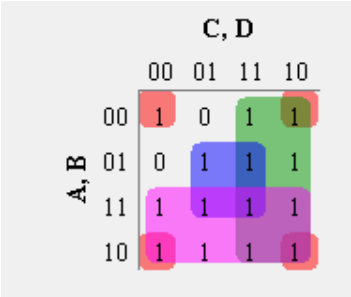
Conversor de BCD a 7 segmentos:

Usaremos la salida del conversor anterior para pasarlo a código 7 segmentos. La tabla de la verdad será la siguiente:

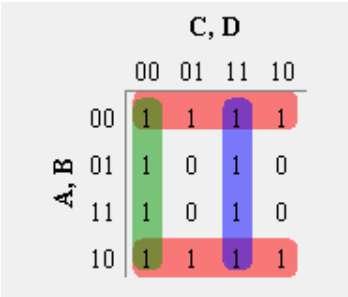
ENTRADAS				SALIDAS						
A	B	C	D	a	b	c	d	e	f	g
0	0	0	0	1	1	1	1	1	1	0
0	0	0	1	0	1	1	0	0	0	0
0	0	1	0	1	1	0	1	1	0	1
0	0	1	1	1	1	1	1	0	0	1
0	1	0	0	0	1	1	0	0	1	1
0	1	0	1	1	0	1	1	0	1	1
0	1	1	0	1	0	1	1	1	1	1
0	1	1	1	1	1	1	0	0	0	0
1	0	0	0	1	1	1	1	1	1	1
1	0	0	1	1	1	1	1	0	1	1
1	0	1	0	1	1	0	1	1	1	1
1	0	1	1	1	1	1	1	0	1	1
1	1	0	0	1	1	1	1	0	1	1
1	1	0	1	1	0	1	1	0	1	1
1	1	1	0	1	0	1	1	1	1	1
1	1	1	1	1	1	1	1	0	1	1

En las entradas, la A corresponde al MSB y la D al LSB

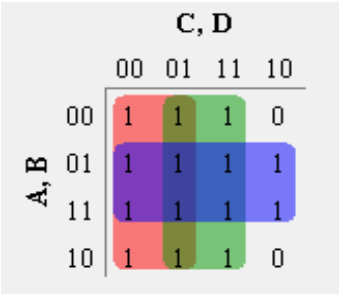
Los mapas de Karnaugh que usaremos y las expresiones resultantes quedarían de la siguiente manera:



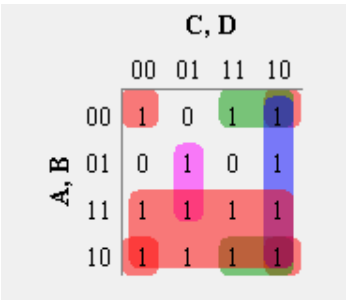
$a = \bar{B}\bar{D} + C + BD + A$



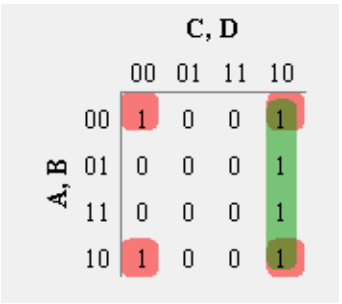
$b = \bar{B} + \bar{C}\bar{D} + CD$



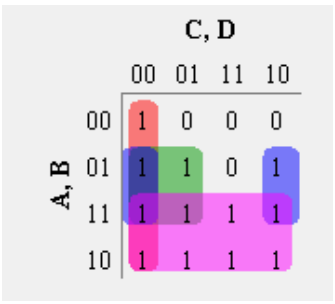
$c = B + \bar{C} + D$



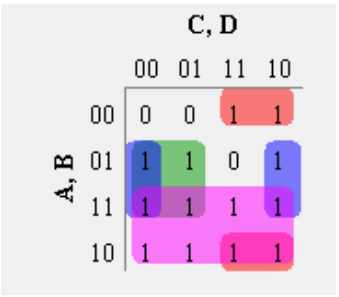
$d = \bar{B}\bar{D} + \bar{B}C + C\bar{D} + B\bar{C}D + A$



$e = \bar{B}\bar{D} + C\bar{D}$

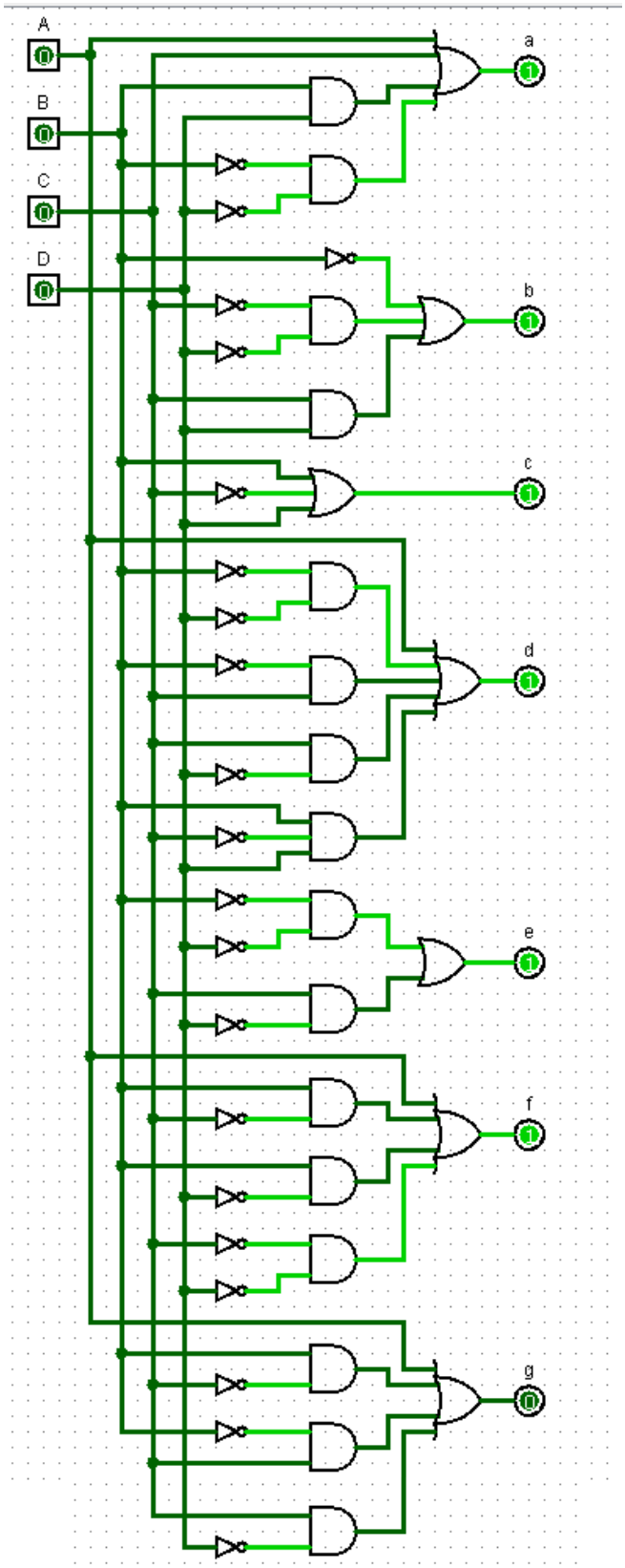


$f = \bar{C}\bar{D} + B\bar{C} + B\bar{D} + A$



$g = \bar{B}C + B\bar{C} + B\bar{D} + A$

El circuito del conversor quedaría de la siguiente manera:



COMPARADOR:

La función del comparador en nuestro circuito es la de decidir si hay empate, ganador azul o ganador rojo.

Para ello utilizamos:

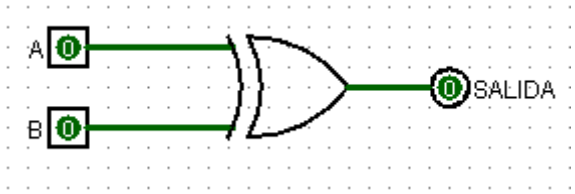
A=B --→ empate entre boxeadores.

A>B --→ gana boxeador rojo.

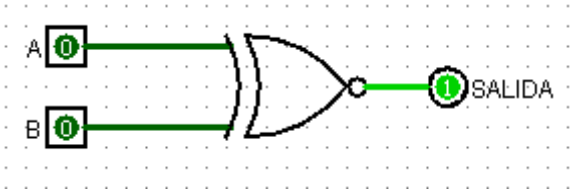
A<B --→ gana boxeador azul.

Sabemos que **A=B** es una puerta lógica XOR, podemos ver su tabla de la verdad a continuación:

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



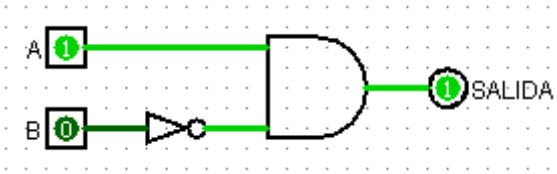
Las entradas que son iguales 00 y 11 nos da 0 pero nosotros en vez de utilizar una XOR utilizamos un XNOR porque preferimos tener 1 en la salida cuando los números son iguales (AyB).



Lo siguiente será definir si A>B, para ello utilizamos una puerta lógica AND en la que la entrada B esta negada, es decir seria $A\bar{B}$

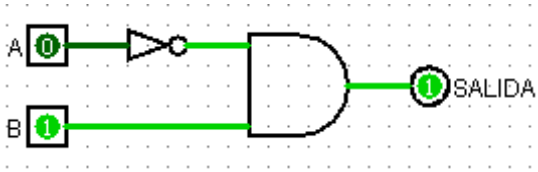
Esto lo podemos ver en la tabla de la verdad:

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



Por último, seria definir A<B que como podemos observar a continuación seria $\bar{A}B$

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

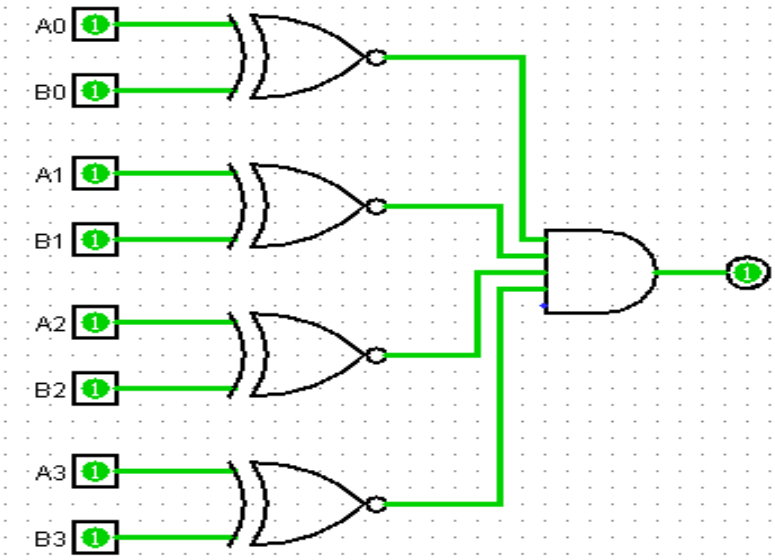


Como nuestro circuito necesita de 4 bits frente a otros 4 bits para comparar (esto viene del resultado dado por el sumador), es decir seria comparar la puntuación del boxeador rojo (4bits) con la puntuación del boxeador azul (4bits). En resumen, necesitamos comparar 2 números de 4 bits cada uno.

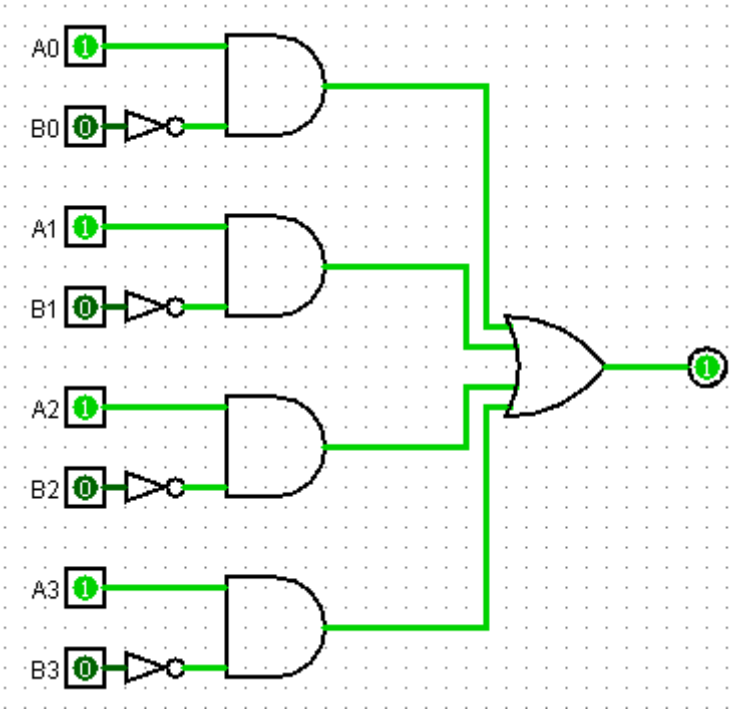
Como hacer la tabla de la verdad es una locura porque estaríamos hablando de 2^8=256 números = líneas de la tabla, haremos nuestro comparador sabiendo lo anterior y uniéndolos en cascada.

Por lo tanto, independientemente del número de bits que tenga nuestros números el comparador siempre nos dará A=B utilizando una XNOR por numero de un bit.

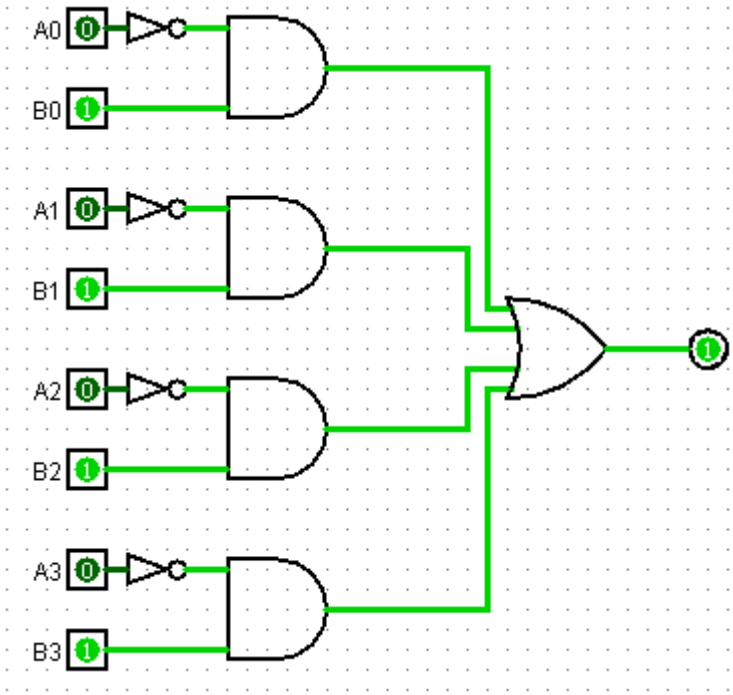
Es decir, quedaría así:



Para la parte en la que tenemos que comprobar que **A>B** utilizamos también lo que hemos mencionado antes, pero uniéndolo en cascada y poniendo una puerta OR a la cual todas las salidas van conectadas, esto lo que hace es que cuando haya un bit de A mayor que el de B nos de 1 en su salida.



Para **A<B** hacemos lo mismo, pero negando A y uniéndolos en cascada.



Ya tenemos todos nuestros circuitos definidos partiendo de su nomenclatura más básica de 1 bit y extrapolándolo a más bits poniendo las entradas en cascada.

Ahora falta unir todo esto para que funcione todo de una forma sinérgica y se activen las salidas correspondientes dependiendo de sus entradas sin olvidar que las salidas serán **A=B, A>B** y **A<B**.

Recordemos que todo esto lo hacemos así porque montar la tabla de la verdad y sacar Karnough es inviable o al menos muy poco práctico de manera manual(humana). Por ello partimos de la base, de números 1 bits para hacerlo ya que extrapolarlo a más bits es igual, pero conectándolo en cascada y modificando alguna entrada con alguna salida.

Por lo tanto, tenemos 2 números de 4 bits que llamaremos **A0**, **A1**, **A2**, **A3** y el otro número será **B0**, **B1**, **B2**, **B3**, siendo **A0** y **B0** los bits de menor peso.

Sabemos que **A=B,** cuando (**A0**= **B0)** y (**A1**=**B1)** y (**A2 =B2)** y (**A3=B3).**

Y sabemos que **A=B** es una puerta lógica **XOR** (en nuestro caso hemos utilizado XNOR) para obtener 1 cuando los bits que se comparan son iguales.

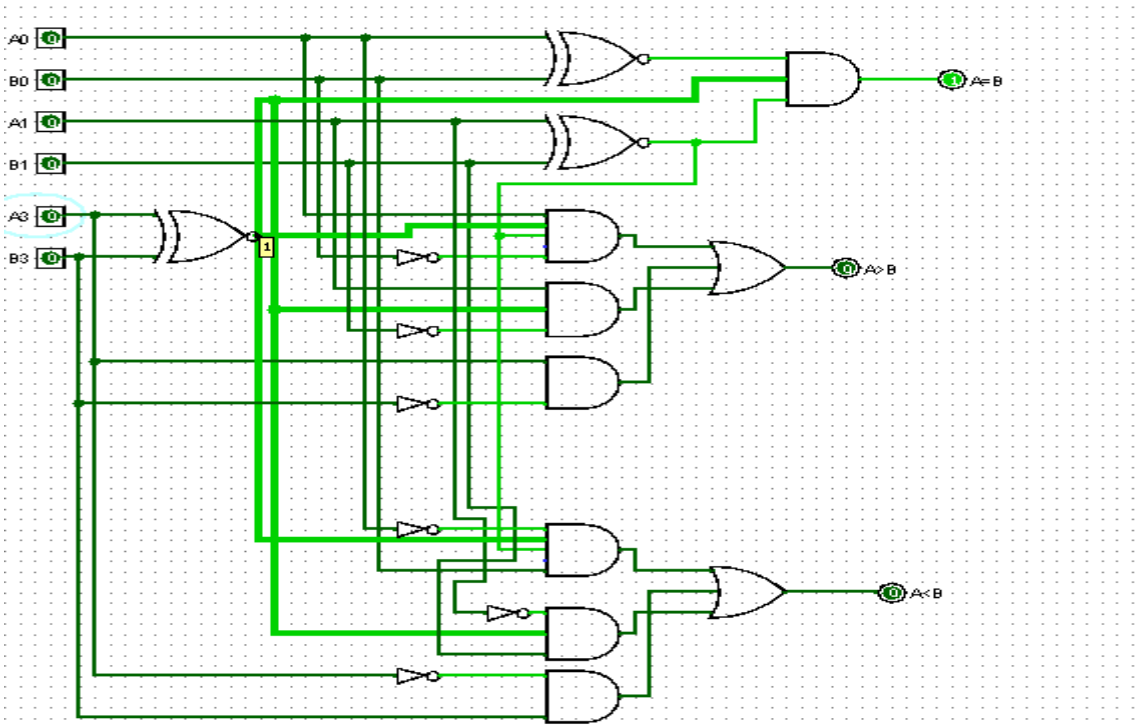
Por lo tanto, la función **(A=B)** = $\overline{(A0 \oplus \mathbf{B0})}$ * $\overline{(A1 \oplus \mathbf{B1})}$ * $\overline{(A2 \oplus \mathbf{B2})}$ * $\overline{(A3 \oplus \mathbf{B3})}$

Sabemos que **A>B** = $A0\overline{B0} + A1\overline{B1} + A2\overline{B2} + A3\overline{B3}$

Sabemos que **A<B** = $\overline{A0}B0 + \overline{A1}B1 + \overline{A2}B2 + \overline{A3}B3$

Por lo tanto, podremos montar nuestro comparador en cascada para comparar 2 números de 4 bits cada uno.

Todo esto tiene que estar conectado de una manera especial para que no se activen a la vez varias salidas, me refiero a si comparamos un bit de **A3** con un bit de **B0**, como nuestros comparadores comparan bit a bit debemos utilizar las salidas de las XNOR e introducirlas en las **ANDs** de menor peso para que no se nos activen las salidas de **A>B** y **A<B** a la vez.



(Ejemplo diseñado con logiSIM de comparador de 3 bits para verlo más claro)

Aquí en verde remarcado podemos ver como la salida de **A3=B3(XNOR)** de los bits de mayor peso(A3B3) van a las entradas de las puertas AND de **A<B** y **A>B** de menor peso, para evitar que se activen las salidas a la vez.

Por lo tanto, si metemos más bits de peso hay que modificar las entradas con las salidas de XNOR de los bits de mayor peso sobre los bits de menor peso.

En nuestro circuito que es de 4 bits seria para la puerta AND de menor peso de bits(A0B0) que hace la comparación de A>B entre los bits de menor peso

Puerta AND que compara A0>B0 = $(A0\overline{B0})(\overline{A1 \oplus \mathbf{B1}})(\overline{A2 \oplus \mathbf{B2}})(\overline{A3 \oplus \mathbf{B3}})$

Para la puerta AND que compara A1>B1 = $(A1\overline{B1})(\overline{A2 \oplus \mathbf{B2}})(\overline{A3 \oplus \mathbf{B3}})$

Lo mismo para las puertas AND que activan la función A<B

Puerta AND que compara A0<B0 = $(\overline{A0}B0)(\overline{A1 \oplus \mathbf{B1}})(\overline{A2 \oplus \mathbf{B2}})(\overline{A3 \oplus \mathbf{B3}})$

Para la puerta AND del siguiente BIT de más peso seria $(\overline{A1}B1)(\overline{A2 \oplus \mathbf{B2}})(\overline{A3 \oplus \mathbf{B3}})$

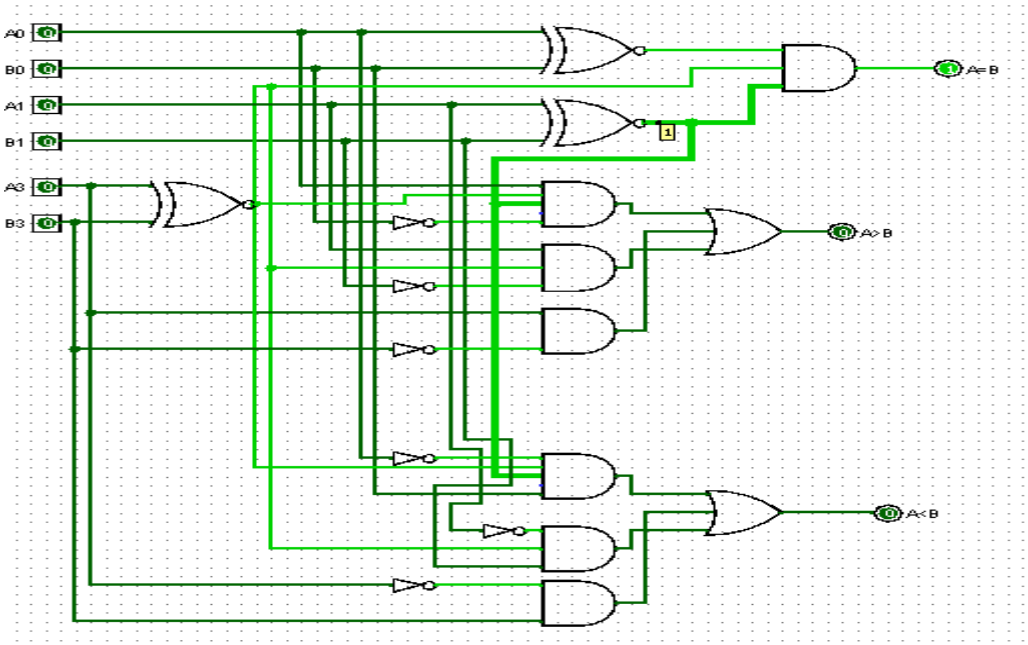
Y así sucesivamente si vamos añadiendo más bits a los números, es decir más comparadores las salidas XNOR de los comparadores de mayor peso de bits van a las entradas A>B y A<B de menor peso.

Para la igualdad A=B no es necesario hacer nada la función seria:

$$A = B = \overline{(A0 \oplus \mathbf{B0})}\,\overline{(A1 \oplus \mathbf{B1})}\,\overline{(A2 \oplus \mathbf{B2})}\,\overline{(A3 \oplus \mathbf{B3})}$$

Si esto es igual a 1 en la salida es que **A=B**

Aquí tenemos otro ejemplo de lo que he explicado antes, pero con una salida **XNOR** de menor peso, en este caso **A1B1** y vemos que va a las entradas **A>B** y **A<B** de los bits de menor peso(**A0B0**) se introduce en sus entradas la salida de la **XNOR** de **A1B1**.



Circuito montado en logiSim para ver lo que ocurre con las salidas de **A=B(XNOR)** de mayor peso frente a las de menor peso)

Y con todo esto explicado ya podemos montar nuestro circuito comparador de 2 entradas de 4 bits para así averiguar que boxeador gana o si hay empate.

4. Conclusiones

En algunos casos, al ver que la tabla de la verdad era demasiado extensa, sobre todo en el sumador y el codificador, se ha optado por conectar en cascada unidades más pequeñas y manejables.

Una dificultad añadida, ha sido la detección de bugs en Logisim, en las que se ha visto que algunas puertas tipo AND de 4 entradas fallaban y se ha tenido que poner puertas de 5 entradas.

Logisim tampoco implementa circuitos con puertas XOR o XNOR de forma automática a partir de la tabla de la verdad, por pequeña que sea, y tampoco deja meter el símbolo en la zona de las ecuaciones, por lo que se han tenido que crear directamente en el circuito.