**LISTADO DE CONCEPTOS**

**1.A** Exprese en complemento a 2 con 8 bits en base 2 los siguientes números: 1, 0, -1, -128, 127, 128, -130

1 = 0 0 0 0 0 0 0 1

0 = 0 0 0 0 0 0 0 0

-1 = 1 1 1 1 1 1 1 1

-128 = 1 0 0 0 0 0 0 0

127 = 0 1 1 1 1 1 1 1

128 = Not in Range

-130 = Not in Range

**1.B** ¿Que tienen en común las representaciones del número -1 con 8 y 16 bits?

En ocho bits -1 se representa con una secuencia de 8 unos seguidos y en 16 es una secuencia de 16 unos seguidos.

-1 en 8 bits = 1 1 1 1 1 1 1 1

-1 en 16 bits = 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

**1.C** ¿Cuál es el rango de números enteros con signo que se puede representar en complemento a 2 con 8, 16 y 32 bits?

En 8 bits el rango de números representables es [-128, 127]

En 16 bits el rango de números representables es [-32768, 32767]

En 32 bits el rango de números representables es [-4294967296, 4294967295]

**1.D** Haga una tabla con cuatro columnas. En la primer columna escriba los número en base 10 del 0 al 15, en la segunda lo propio en base 2, en la tercera igualmente pero en base 16 y en la cuarta en complemento a 2 con 4 bits.

Base 10 Base 2 Base 16 Complemento a 2 con 4 bits

0 0 0 0000

1 1 1 0001

2 10 2 0010

3 11 3 0011

4 100 4 0100

5 101 5 0101

6 110 6 0110

7 111 7 0111

8 1000 8 1000

9 1001 9 1001

10 1010 A 1010

11 1011 B 1011

12 1100 C 1100

13 1101 D 1101

14 1110 E 1110

15 1111 F 1111

**1.E** Represente los números -1 y 2 en complemento a 2 de 4 bits en base 2. Sume los números. ¿Cuál es el resultado?

-1 = 1111

2 = 0010

1 = 0001

**1.F** Escriba los números 15 y 2 en base 2 con 4 bits. Sume los números. ¿Cuál es el resultado?

1111 +

0010

= 0001

**1.G ¿**Qué relación encuentra entre los puntos 5 y 6? ¿Qué puede concluir?

El número 15 binario se usa para representar el -1 en complemento a 2

**1.H** ¿Por qué razón cree usted, en base a lo anterior, que los computadores utilizan la representación en complemento a 2? Justifique su respuesta en relación a la operación suma.

Una de las razones de utilizar complemento a 2 es la capacidad de representar restas usando sumas.

**1.I** ¿Cuál es la diferencia entre un Half adder y un full adder?

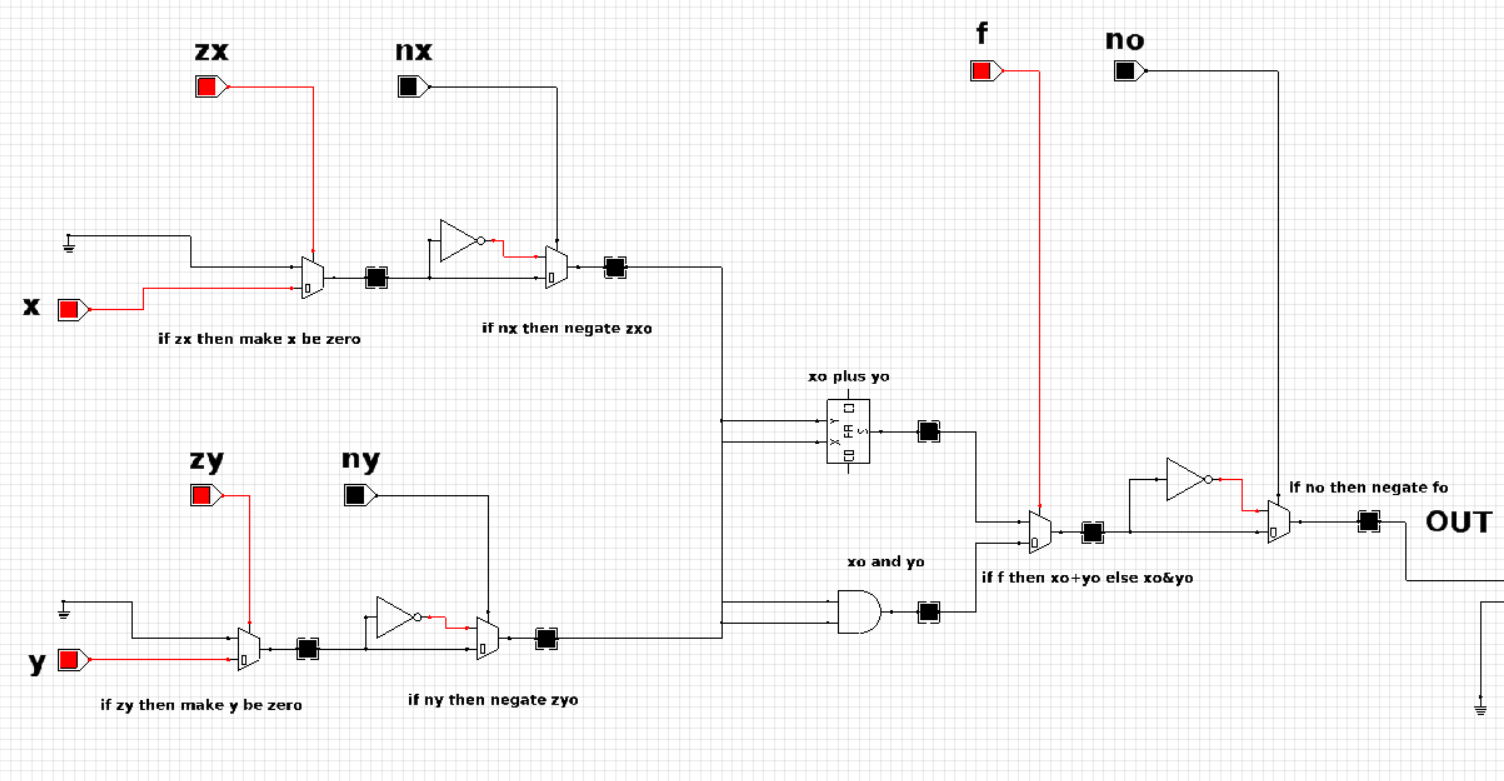
El half adder tiene dos buses de entrada mientras el full adder tres, siendo el bus adicional de gran utilidad para considerar el termino carry de otro half adder cualquiera. Dicha funcionalidad permite encadenar muchos half adders.

**1.J** ¿Qué significa que una suma ha generado un carry?

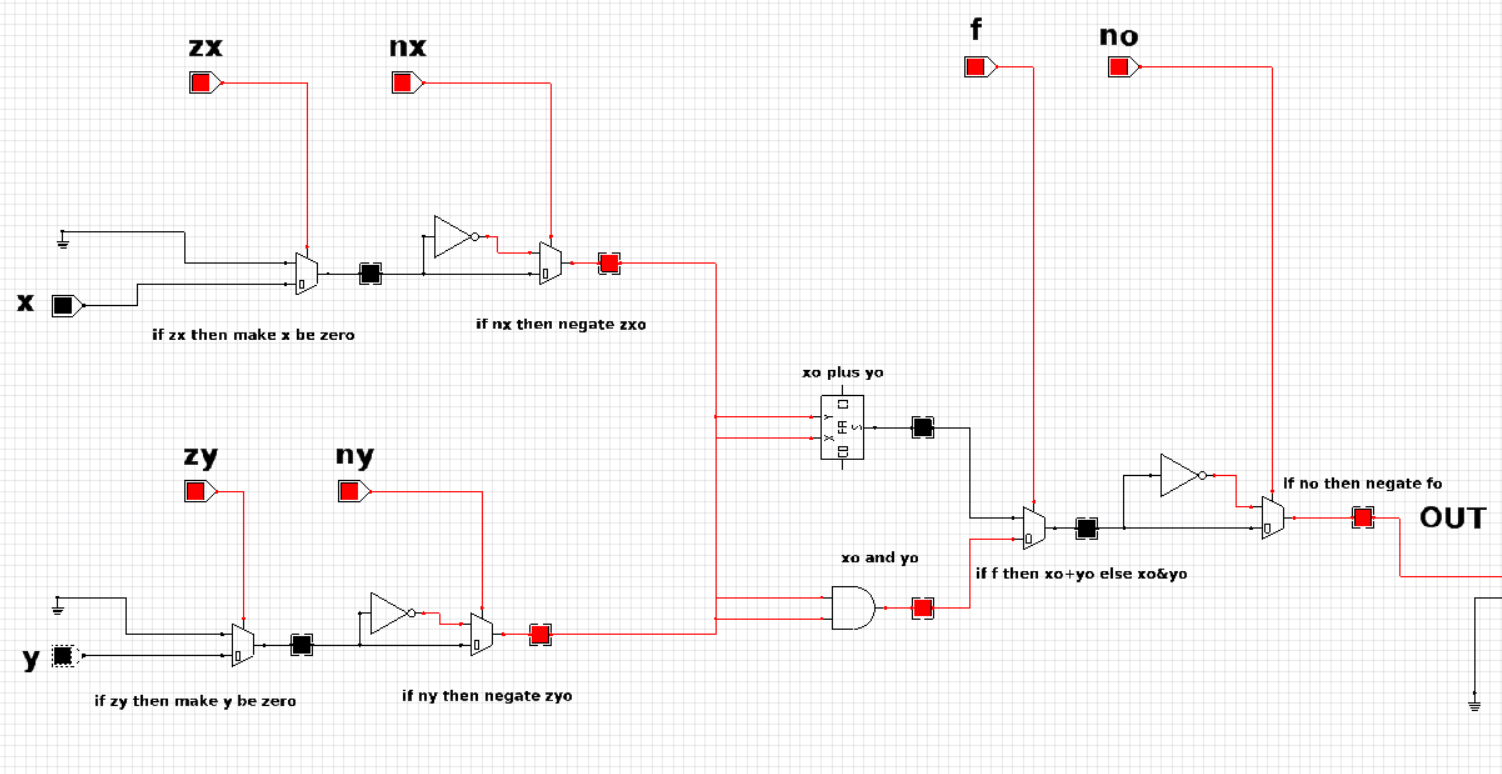
La suma no puede ser representada usando la misma cantidad de dígitos de los sumandos.

**1.K** En relación a la unidad aritmético lógica implementada, muestre un ejemplo que ilustre cada una de las operaciones que ésta puede realizar. Para ello, muestre en un diagrama en bloques de la ALU los valores intermedios de cada uno de los chips que la componen.

**FUNCION ZERO**

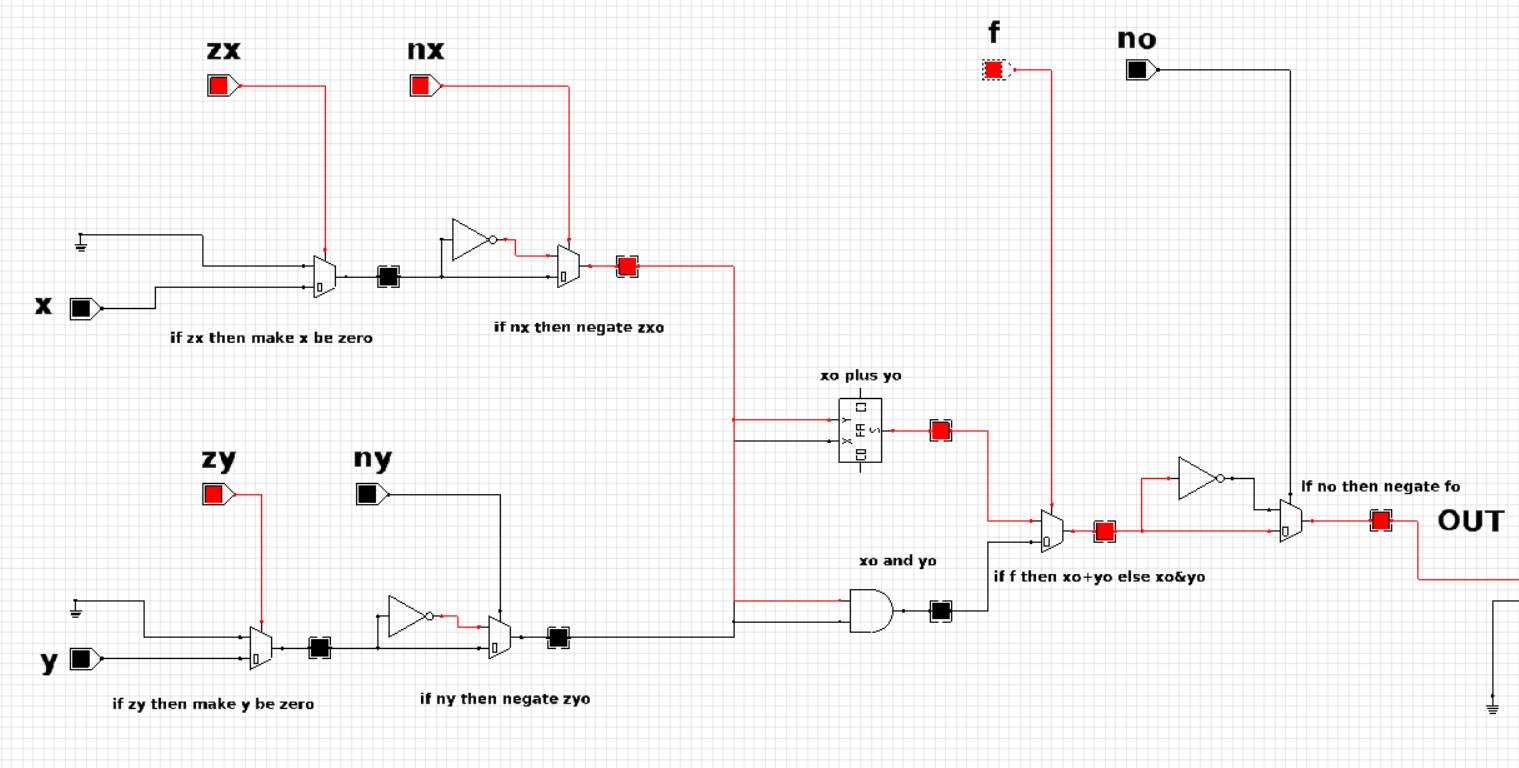


**FUNCION ONE**

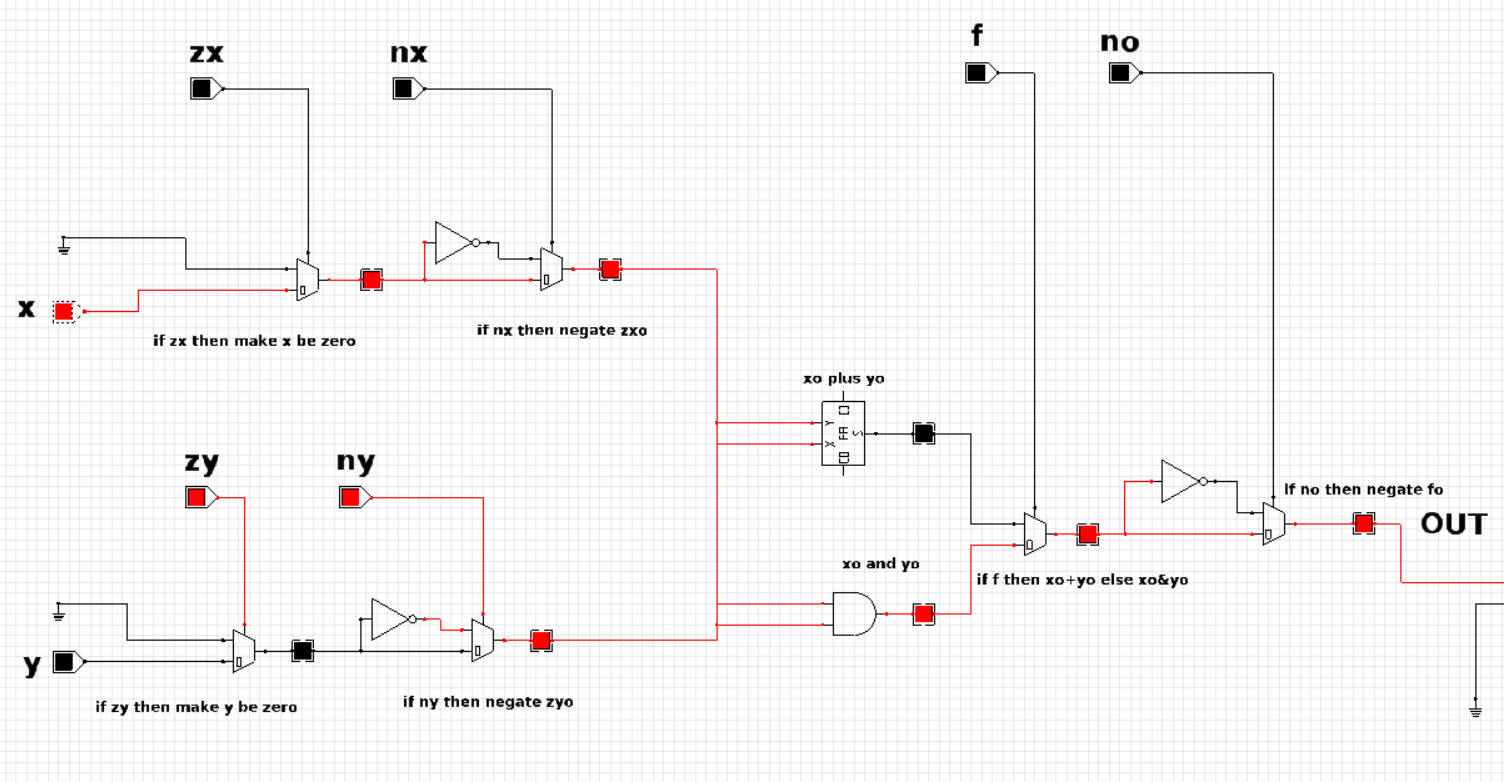


**FUNCION -1**

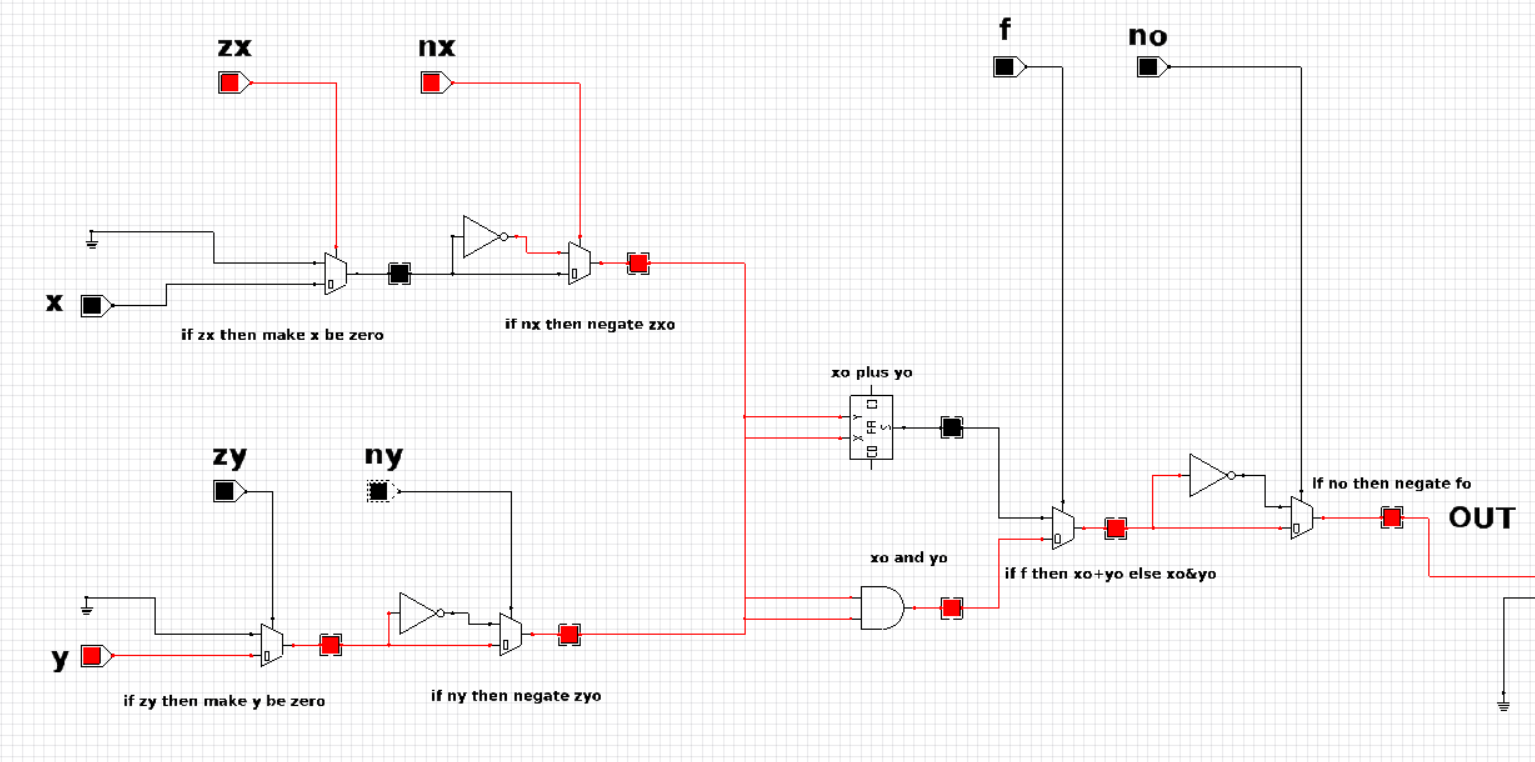
El output es una secuencia de bits de longitud n representando el uno, donde n el nùmero de bits de la arquitectura. Sin embargo la siguiente imagen solo puede representar un unico bit de dicha secuencia de bits.



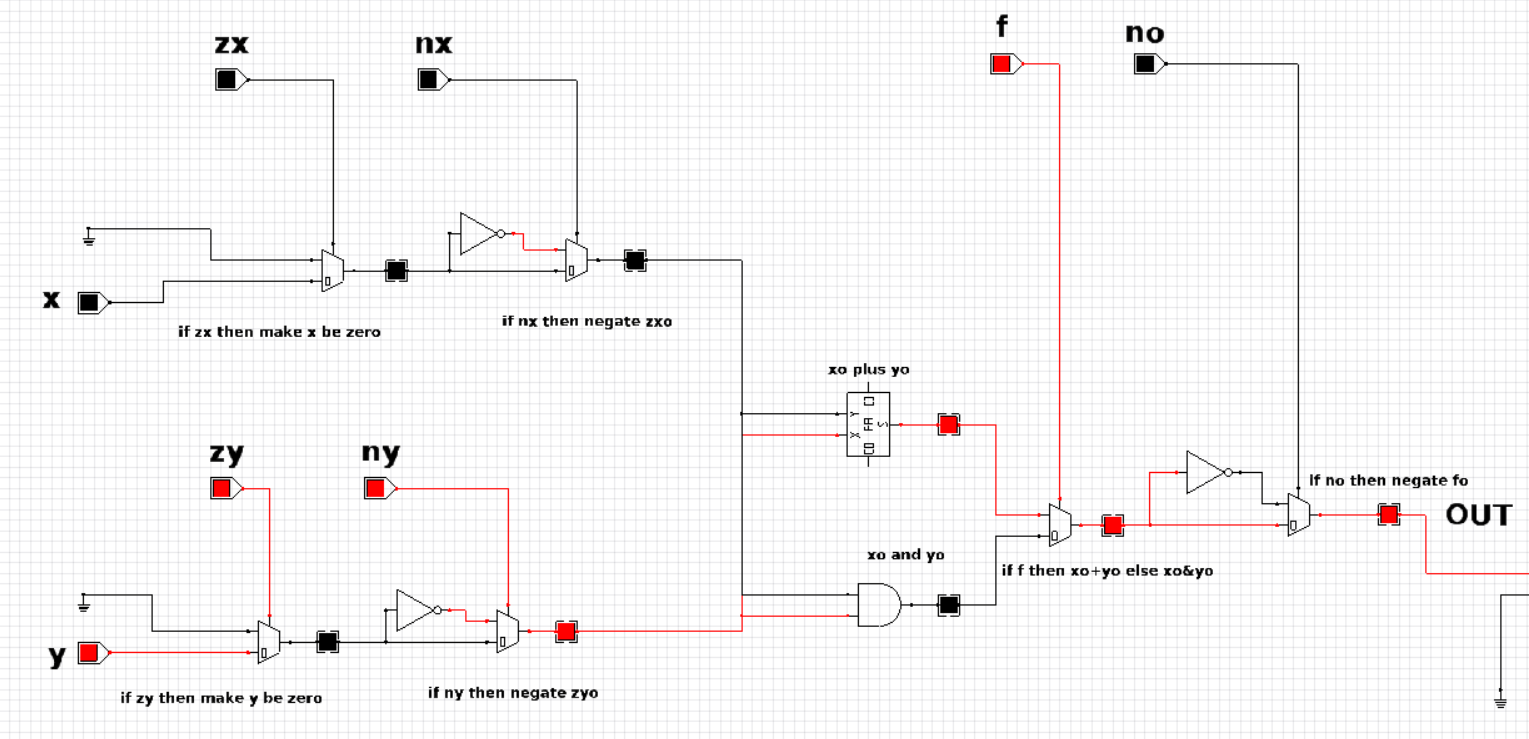
**FUNCION X**

****

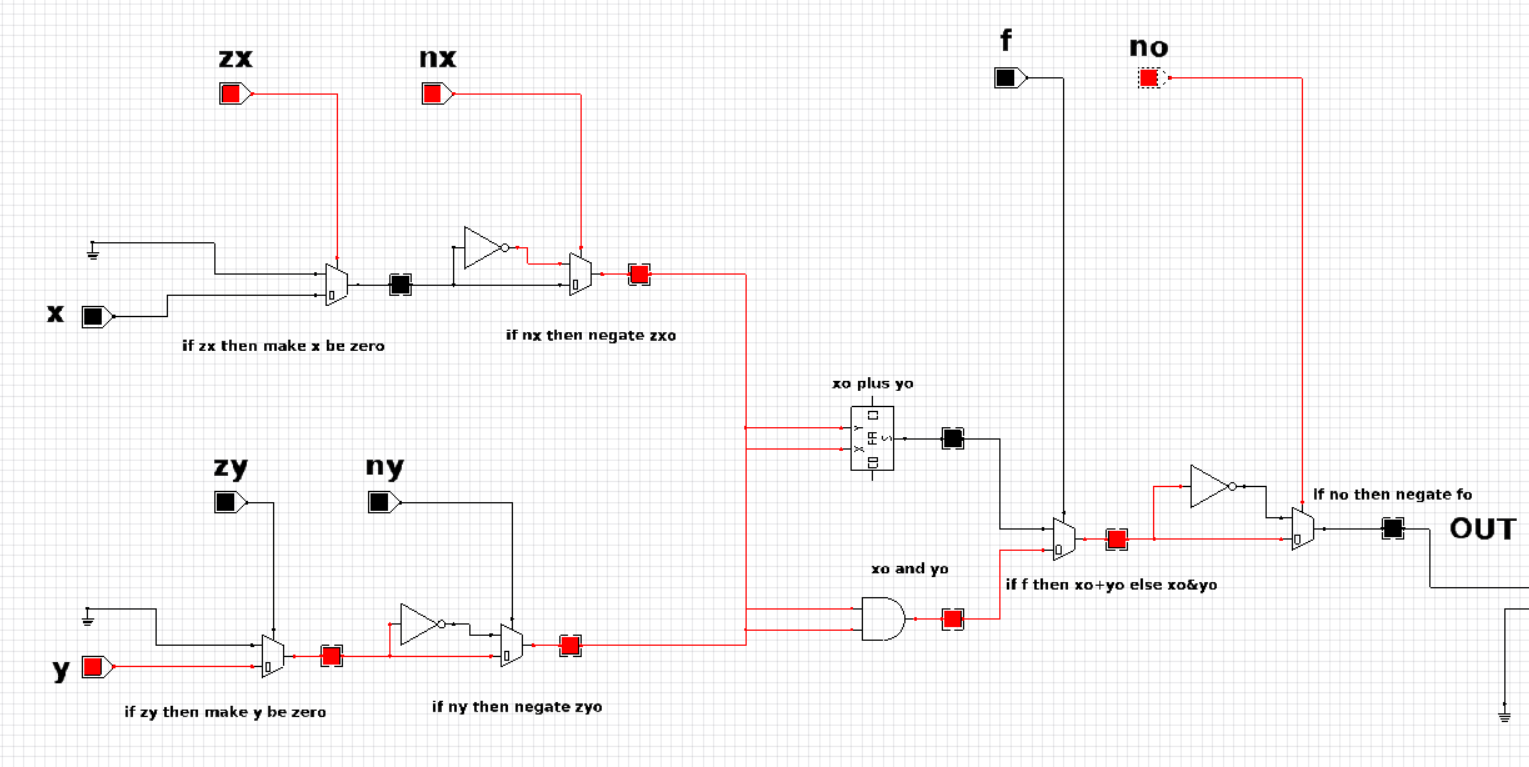
**FUNCION Y**

****

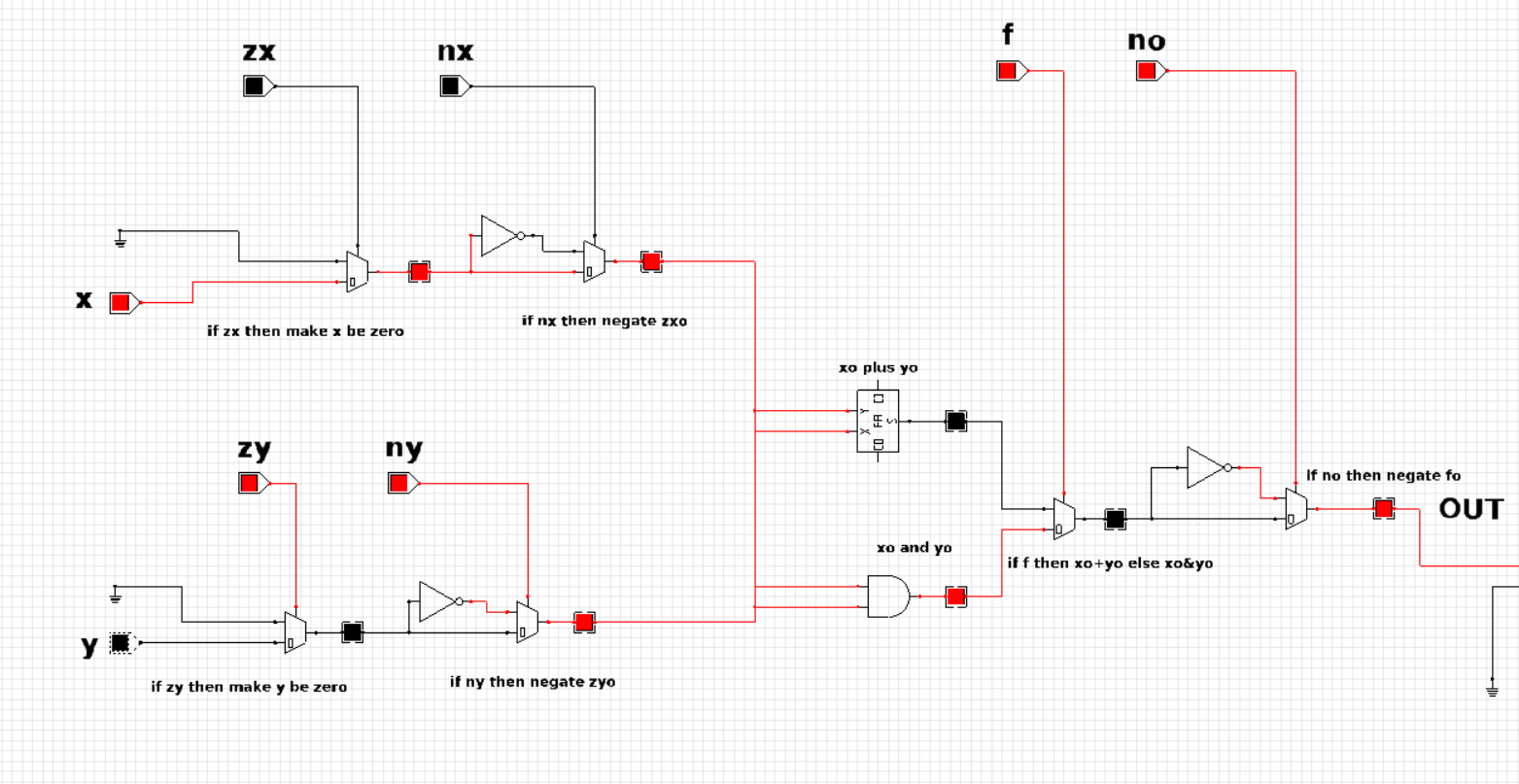
**FUNCION !X**



**FUNCION !Y**

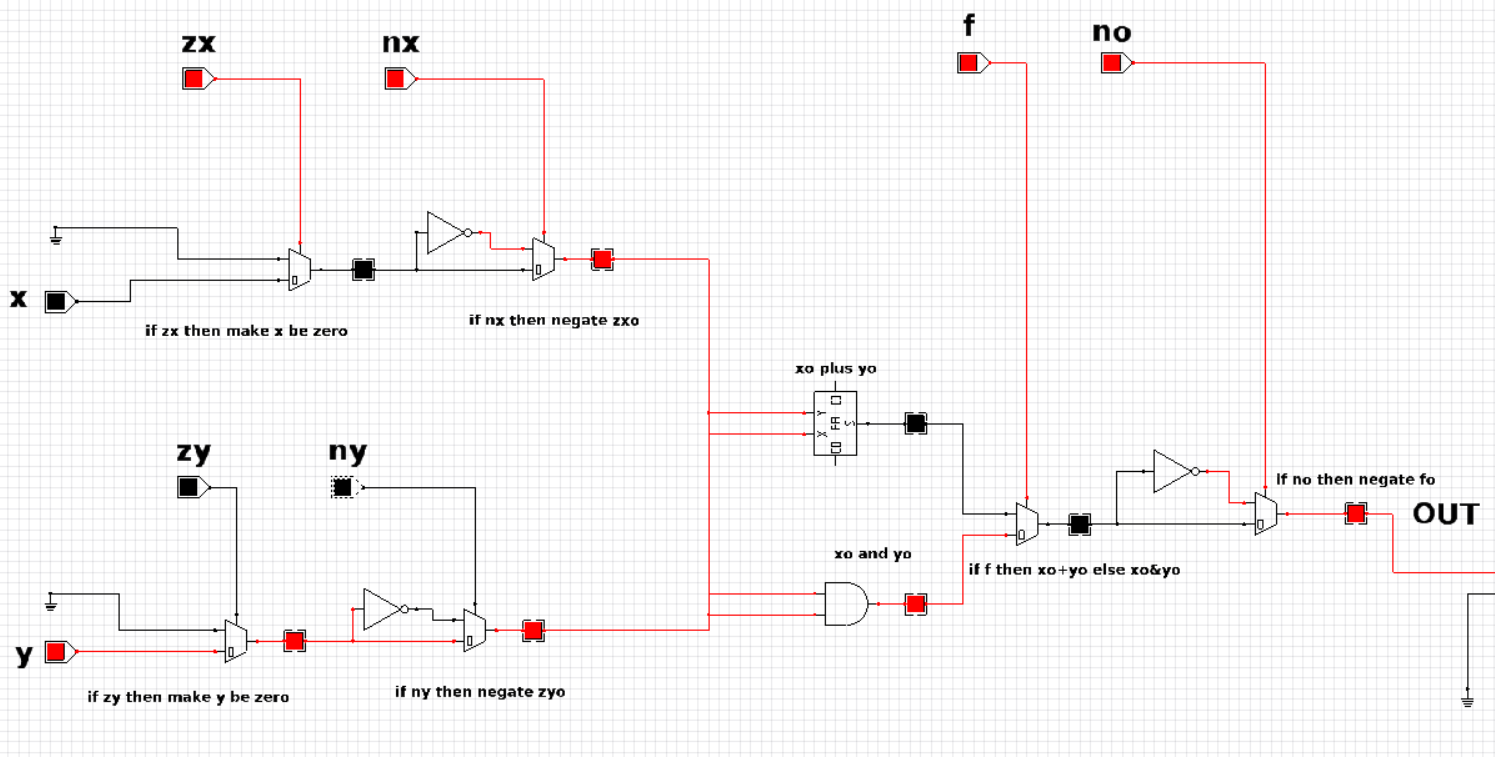


**FUNCION –X**



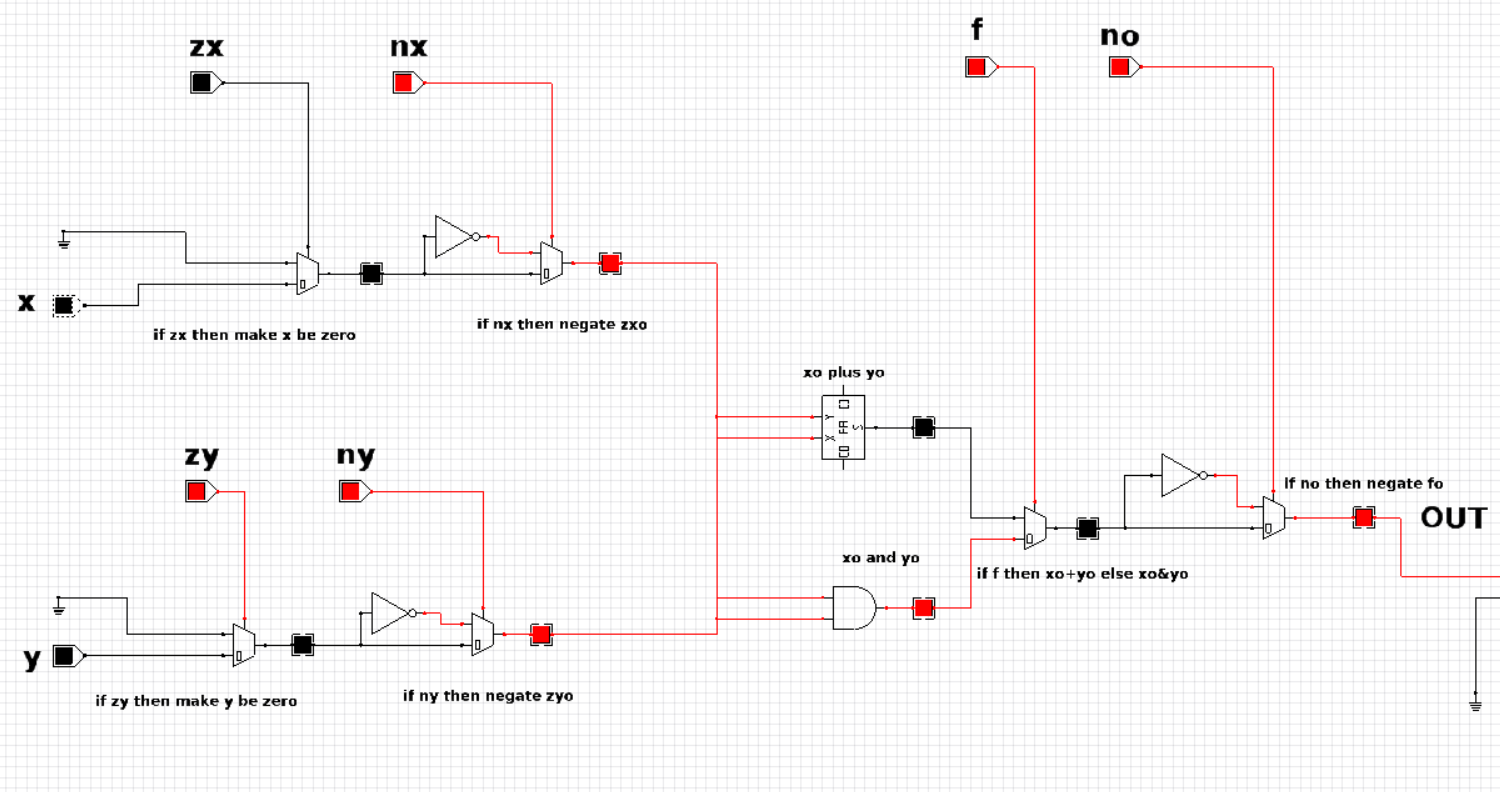
El esquema solo puede representar una entrada de un bit, y el bit menos significativo de la salida. Evaluando la función en 4 bits f(x = 0001) = 1111, tenemos que el bit menos significativo del resultado es efectivamente un uno. Sin embargo lo anteriormente mencionado es solo una prueba parcial del correcto funcionamiento de la ALU, para efectos prácticos consideraremos que la evaluación es correcta en su totalidad aunque no tengamos evidencia de los otros tres bits de la salida ni de la entrada.

**FUNCION –Y**



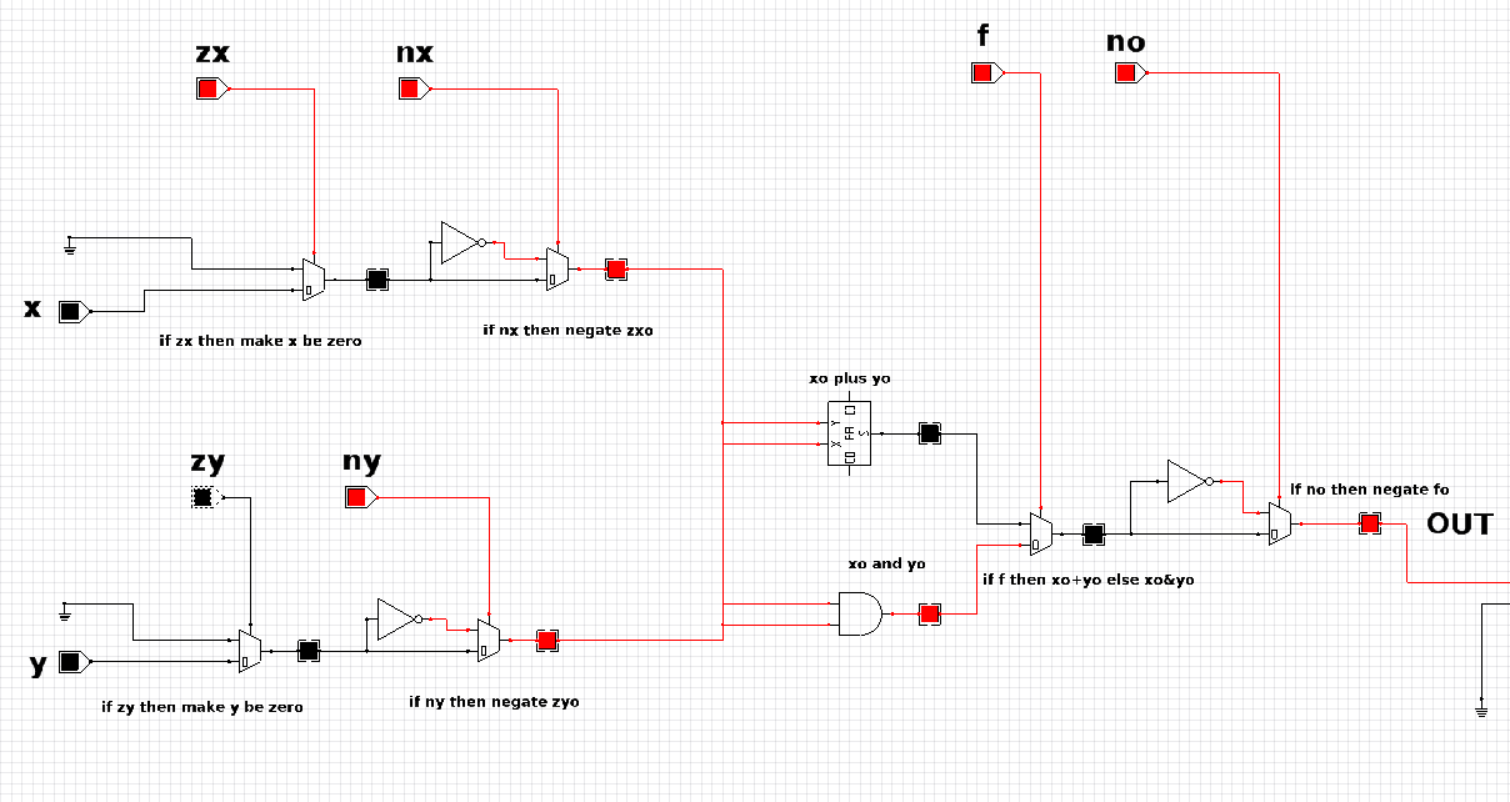
Las limitaciones del esquema no permiten evidenciar un funcionamiento completo de la ALU.

**FUNCION X+1**

****

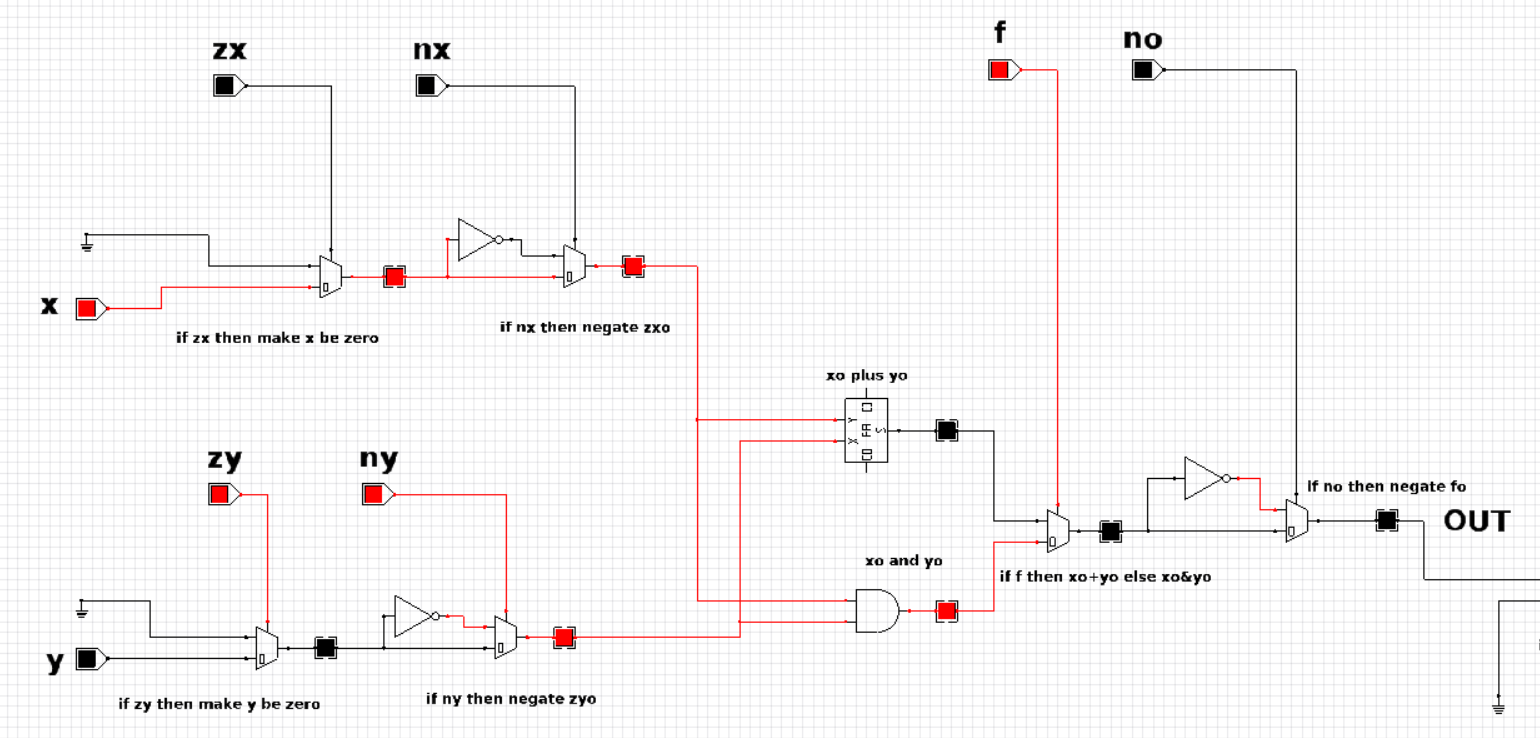
**f(x = 0) = 1**

**FUNCION Y+1**

****

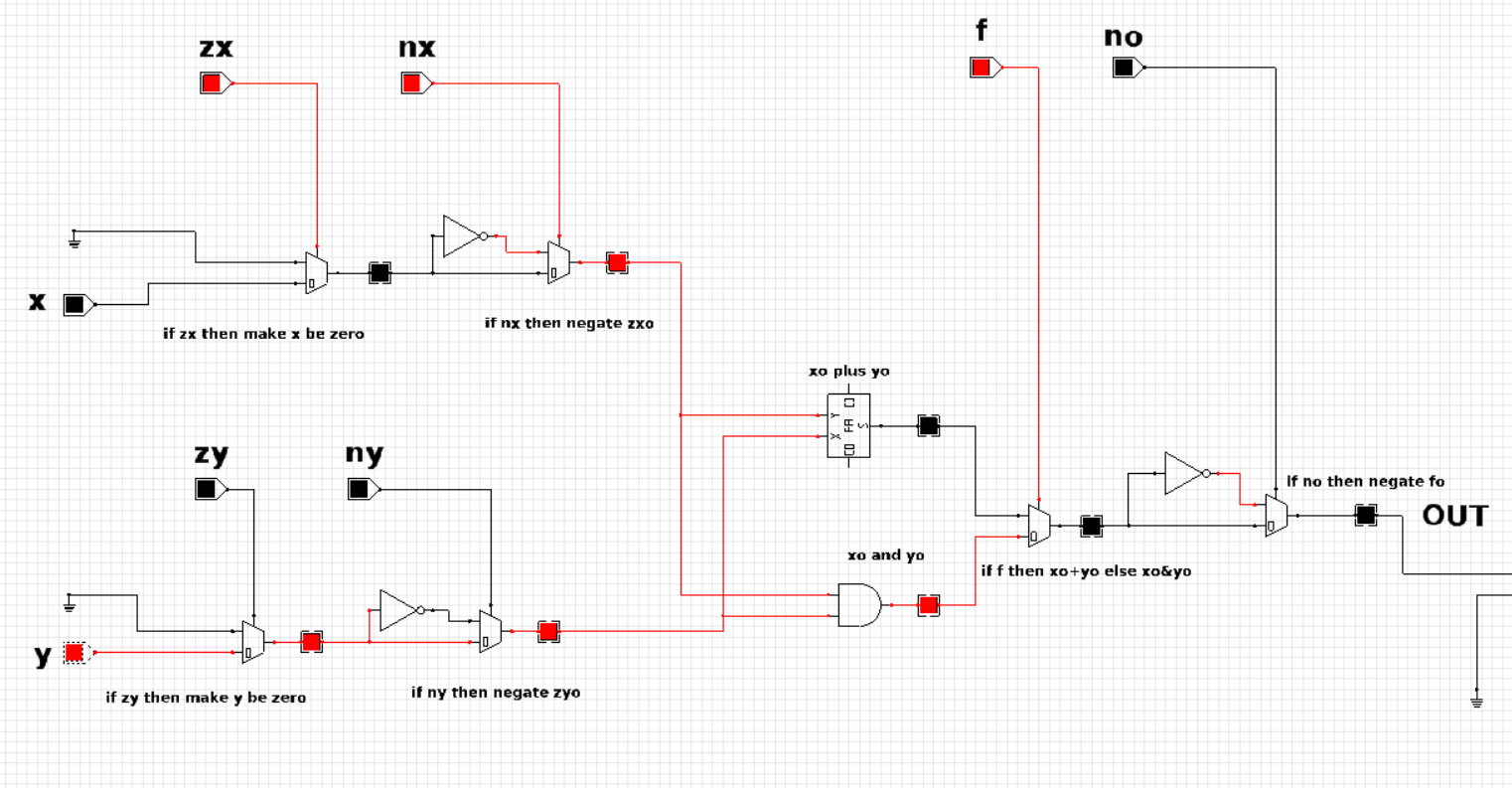
**f(y = 0) = 1**

**FUNCION X-1**

****

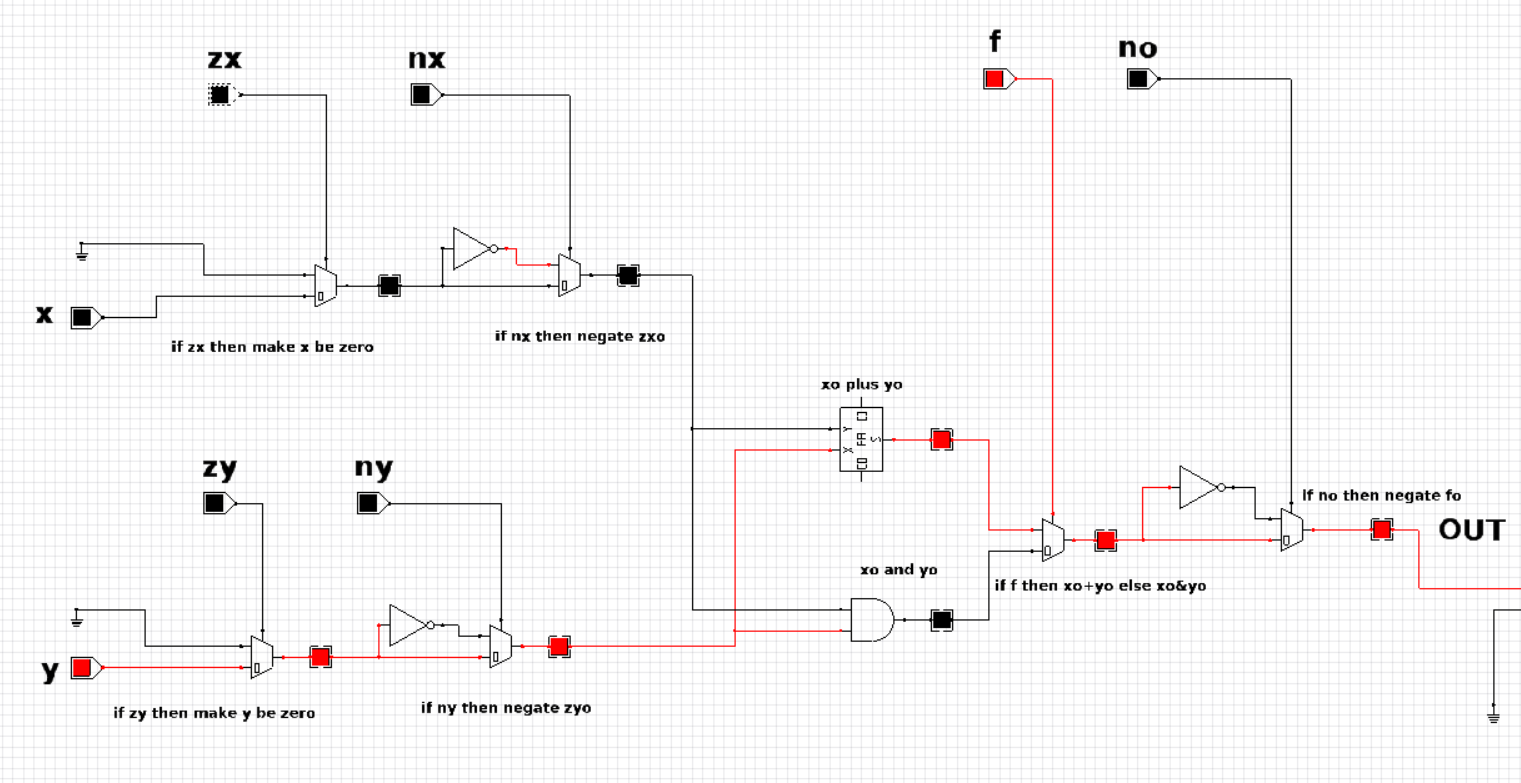
**f(x = 1) = 0**

**FUNCION Y-1**

****

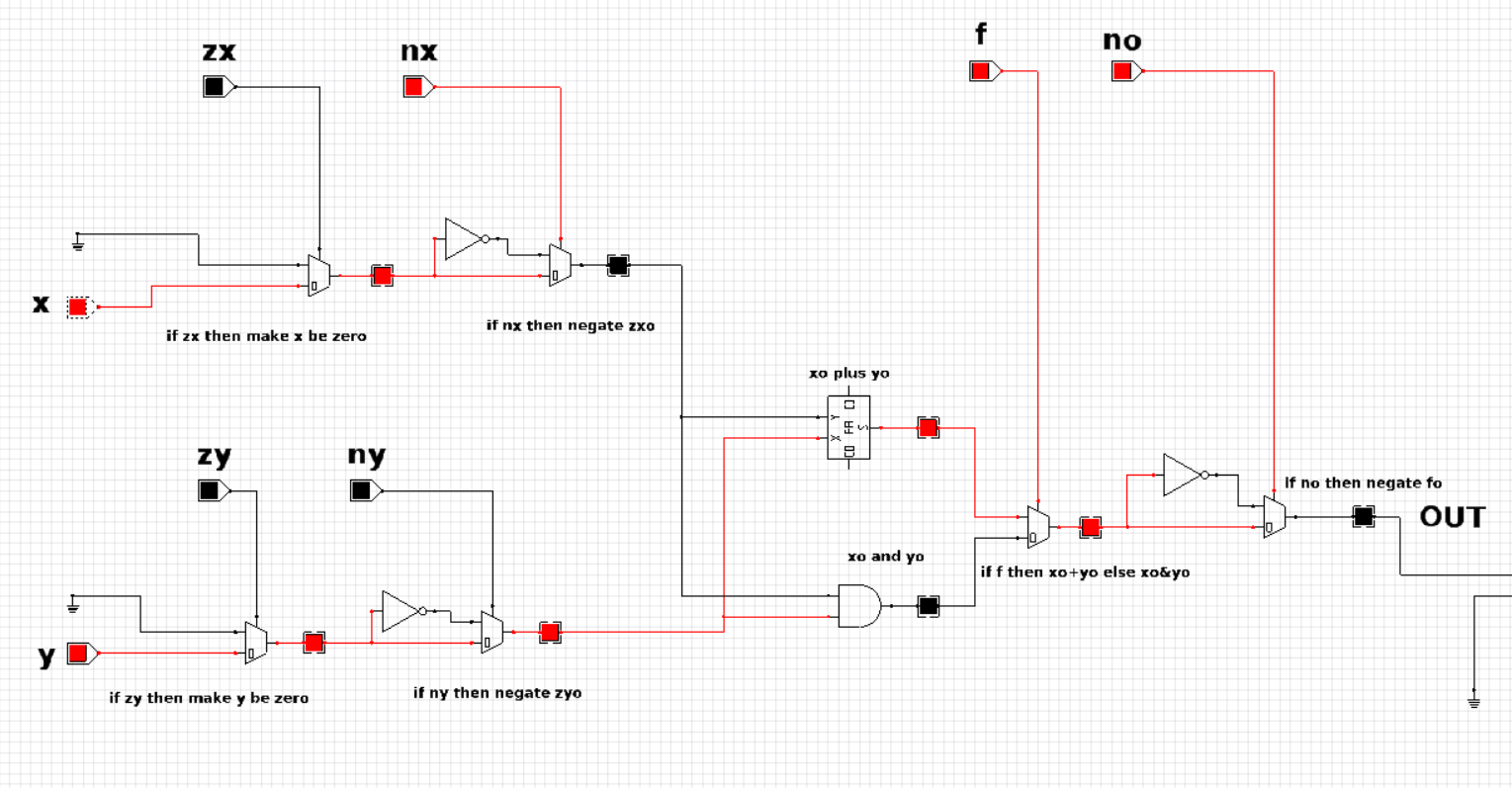
**f(y = 1) = 0**

**FUNCION X+Y**

****

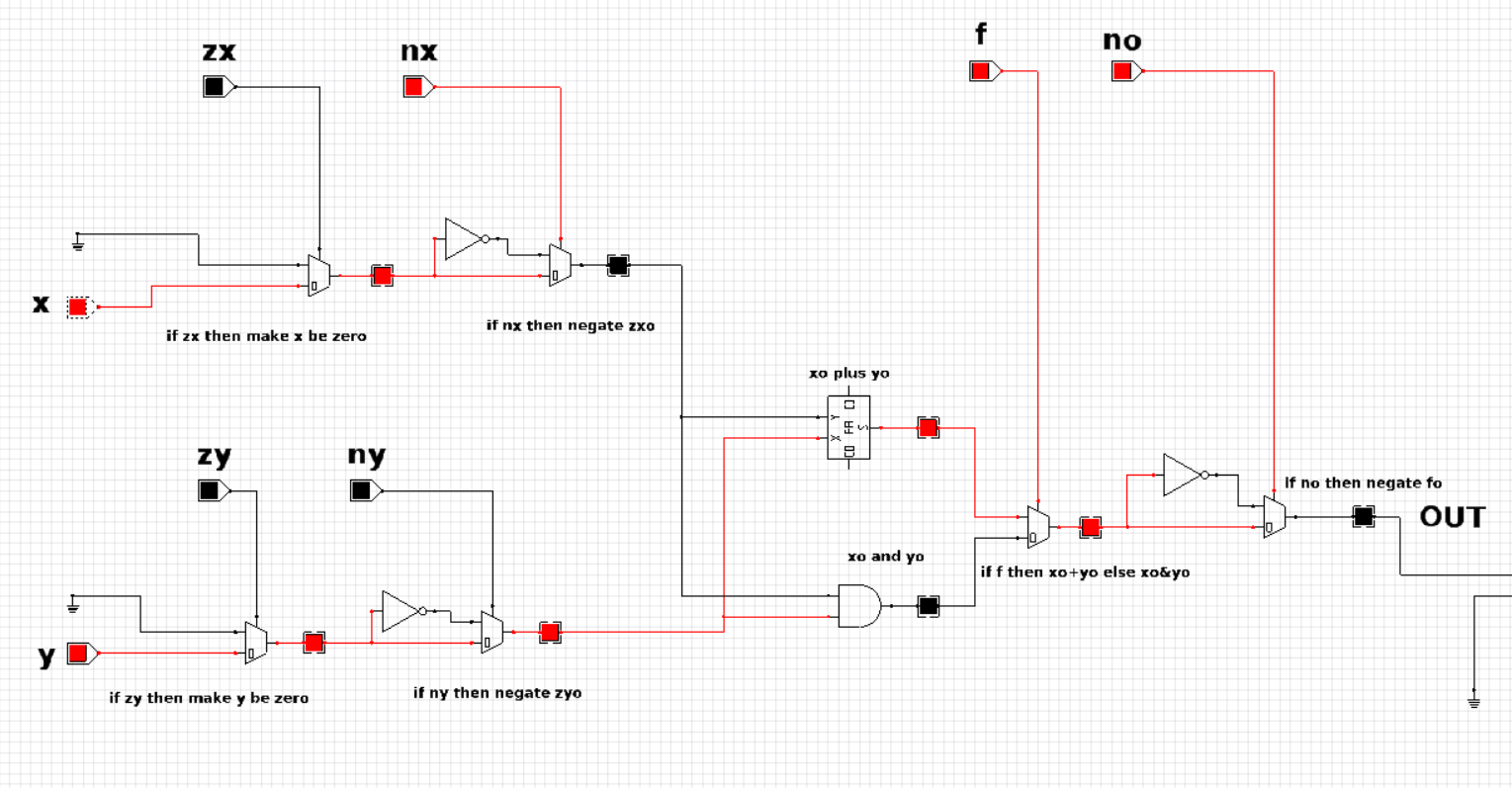
**f(x = 0, y = 1) = 1**

**FUNCION X-Y**

****

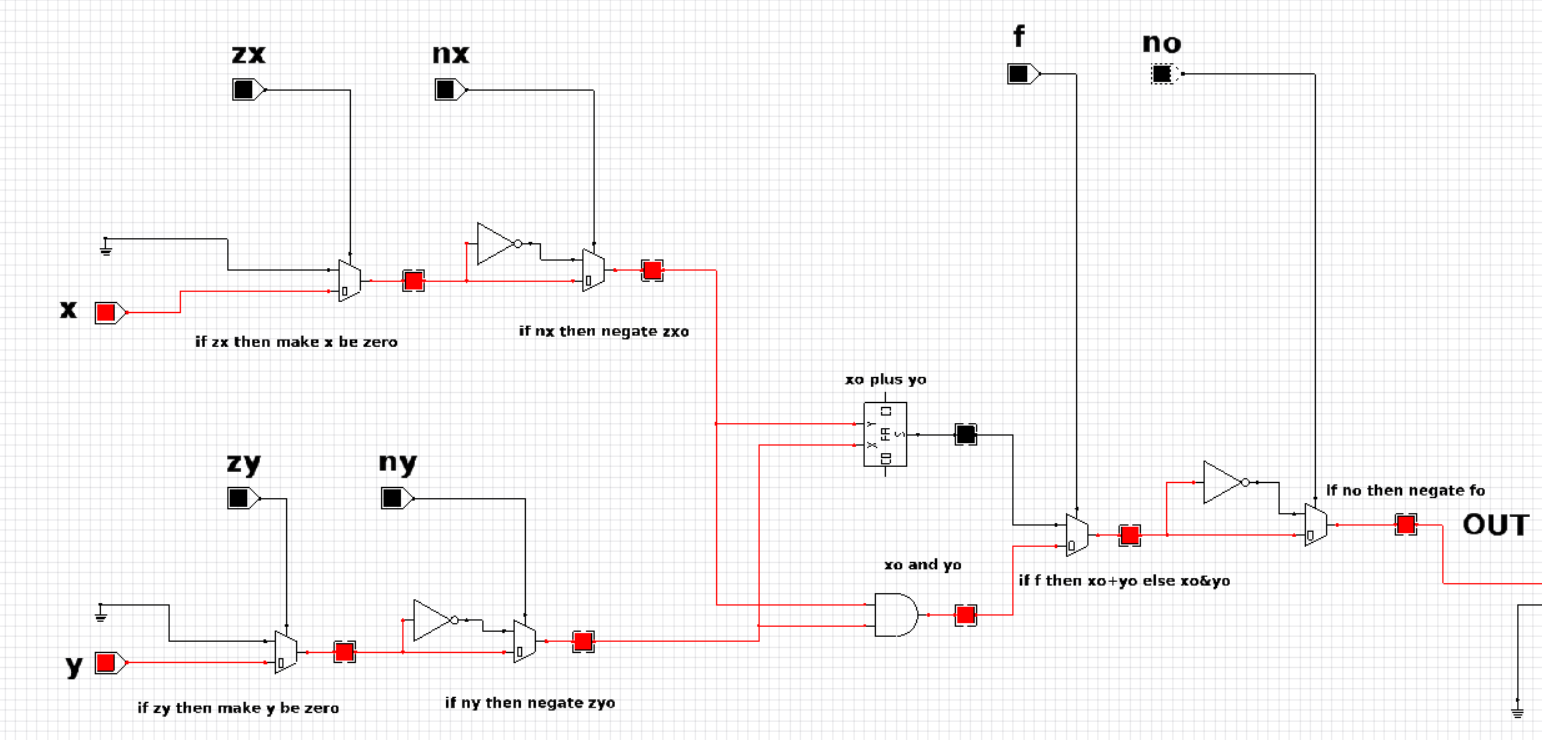
**f(x = 1, y = 1) = 0**

**FUNCION Y-X**

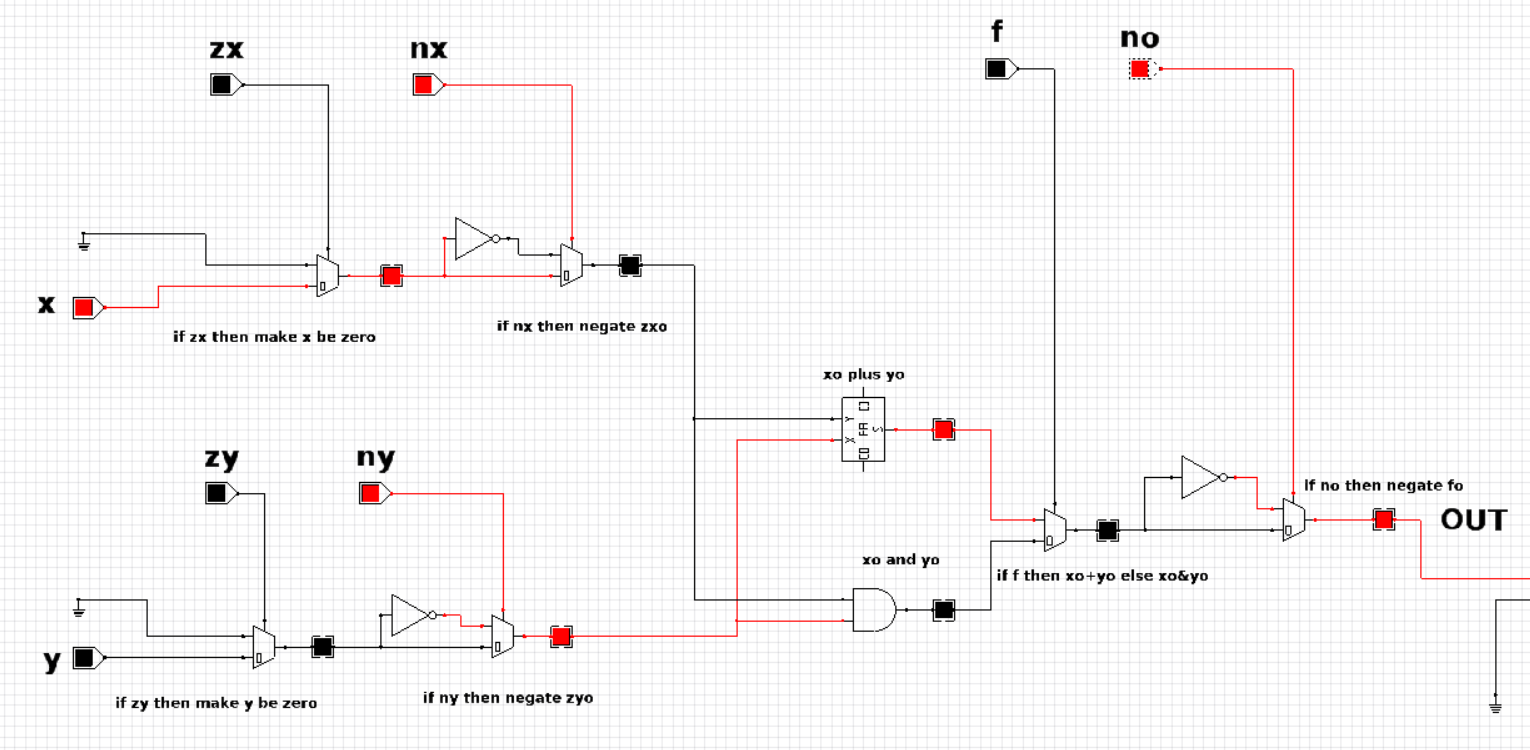
****

**f(x = 1, y = 1) = 0**

**FUNCION X&Y**



**FUNCION X|Y**

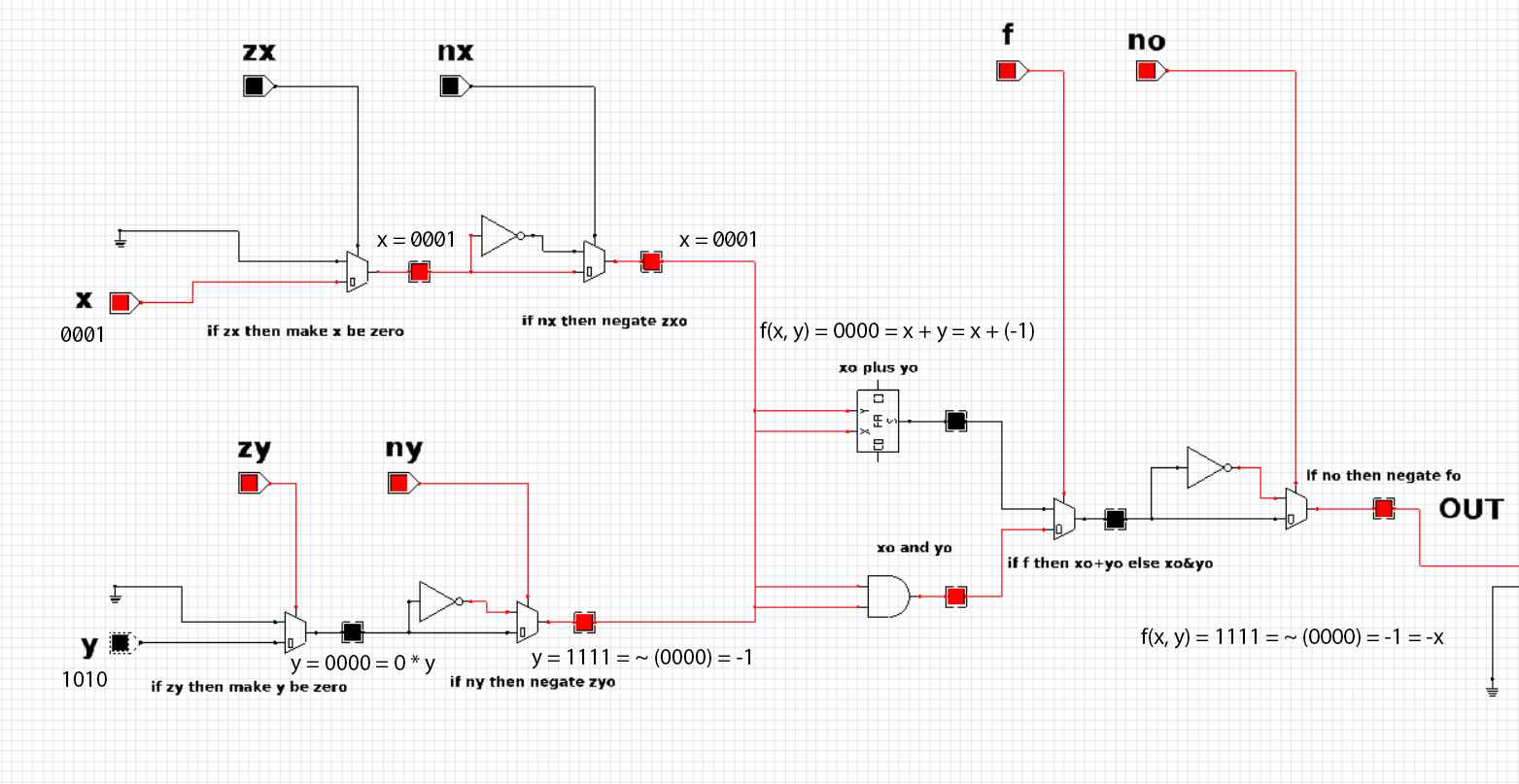
****

­**DESCIFRANDO LA ALU**

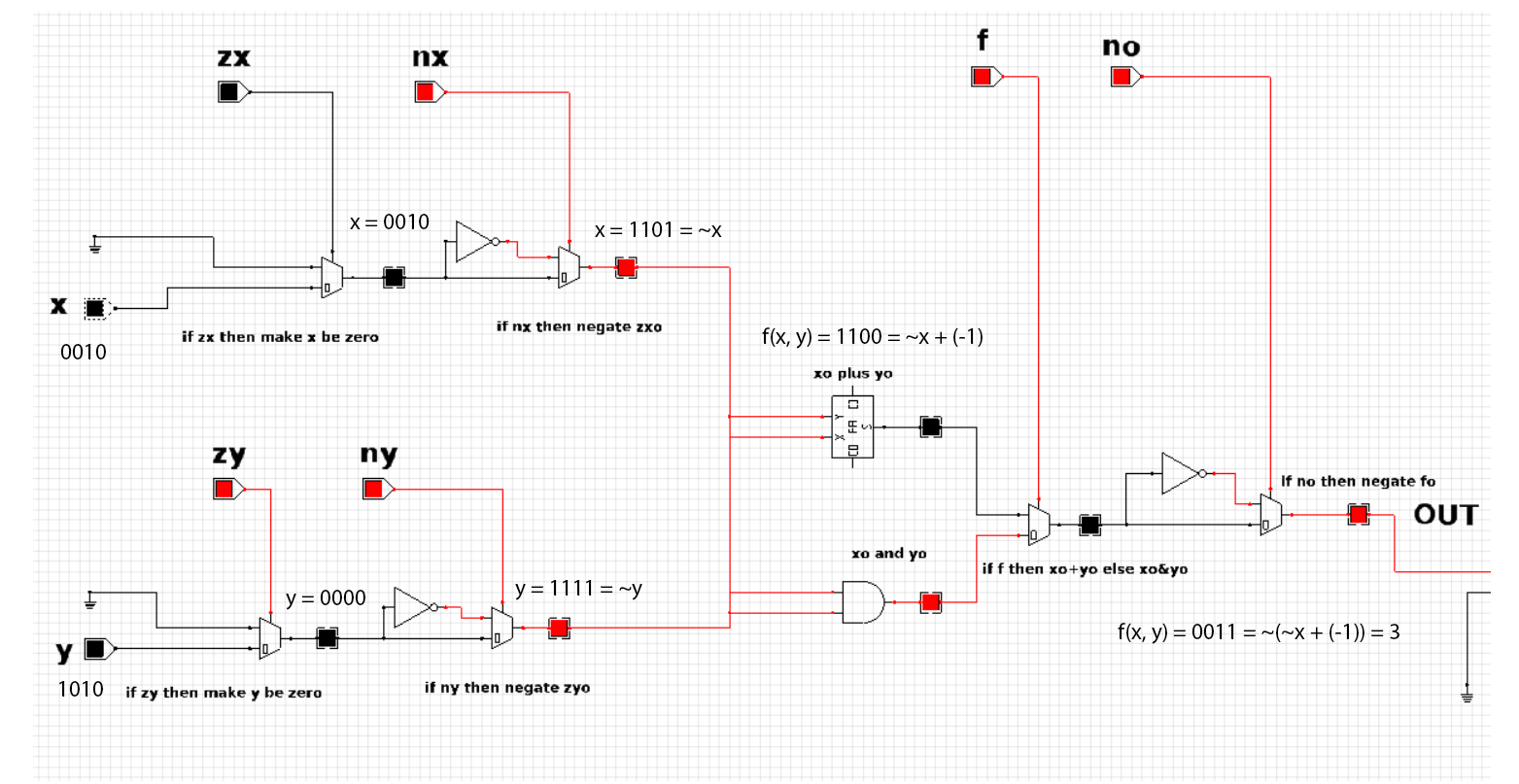
Demuestre cómo es posible que la ALU pueda realizar con el hardware que tenemos las siguientes operaciones: [Leer primero esto](https://es.wikipedia.org/wiki/Complemento_a_uno)

-X X+1 X-Y X or Y

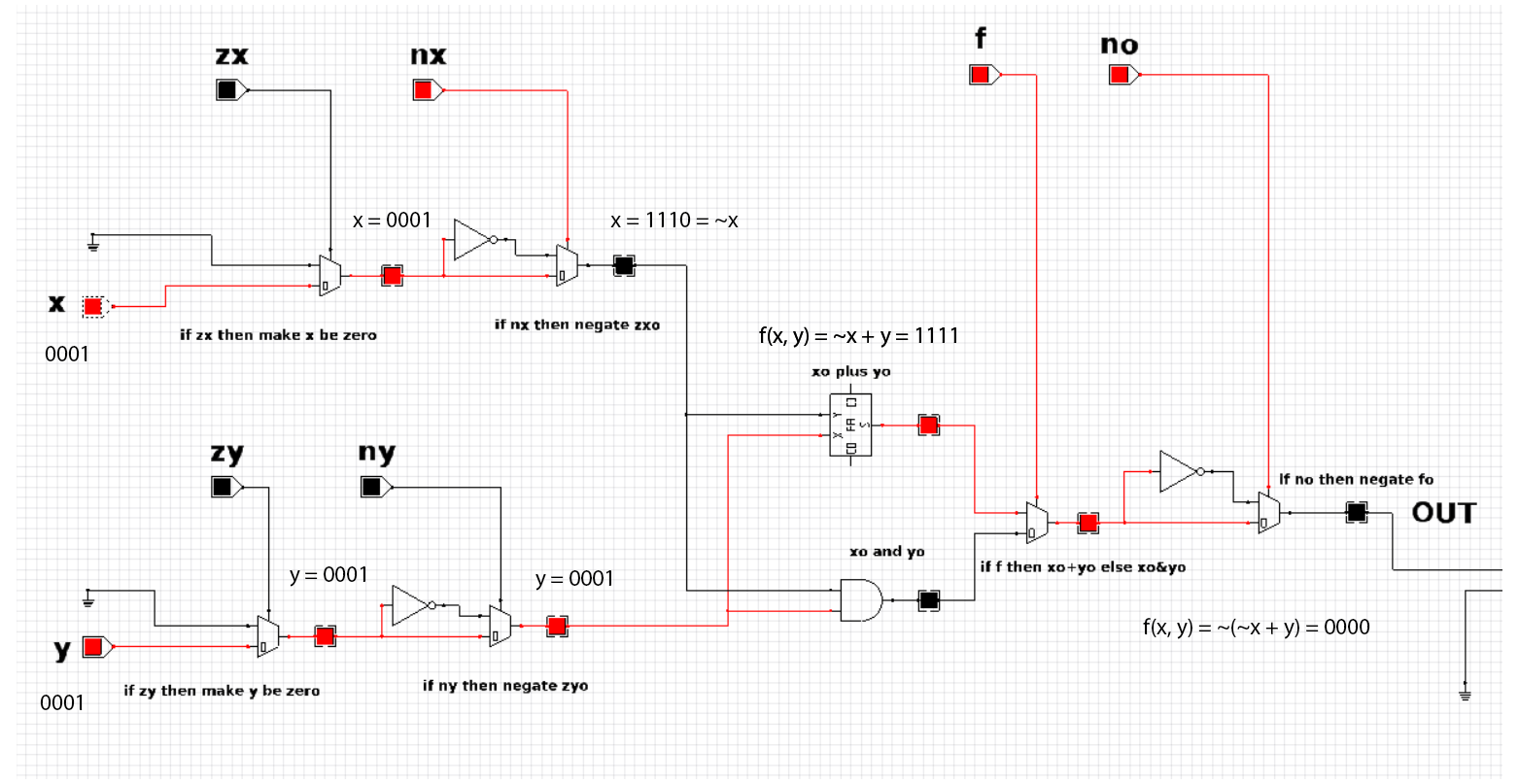
**DEMOSTRANDO -X**



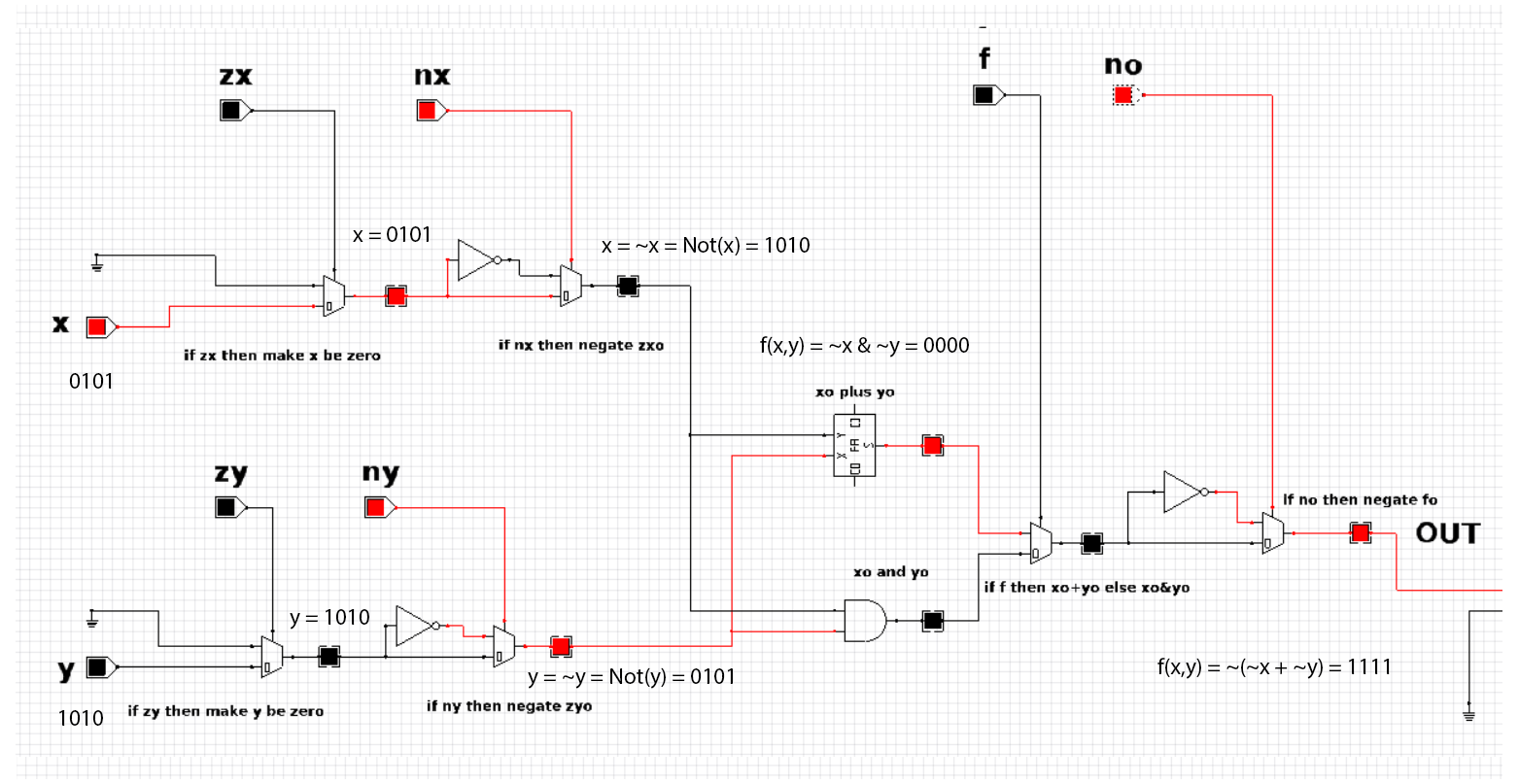
**DEMOSTRANDO X+1**



**DEMOSTRANDO X-Y**

****

**DEMOSTRANDO X OR Y**



# EXPLICACIÓN DE CHIPS

**HALF-ADDER**

Suma dos buses de n bits y devuelve el resultado junto al carry.

**FULL-ADDER**

Suma tres buses de n bits (el bus adicional se usa como carry in) y devuelve el respectivo resultado junto al carry.

**ADDER**

Suma dos buses de n bits, de tal forma que cada bus represente un número con signo en complemento a 2. Por lo tanto la función Adder posee solo una salida que representa el resultado de la suma sin tener en cuenta el overflow.

**INCREMENTER**