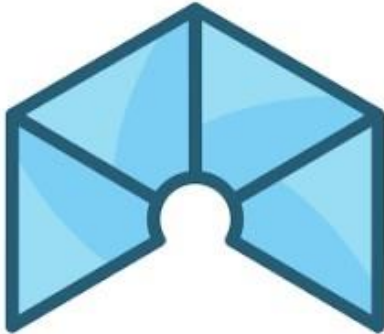


# FPGA Libres con ICEstudio

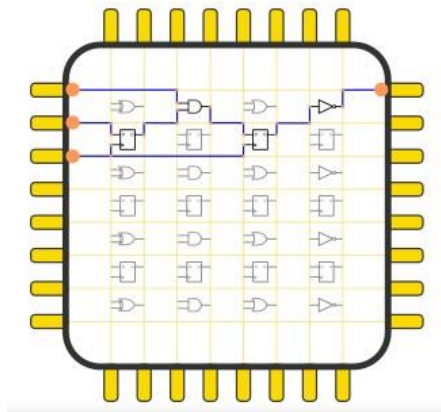


# FPGA (“Field Programmable Gate Array”)

## Qué son las FPGAs?

Las FPGAs son como **chips en blanco**. Chips constituidos por puertas lógicas, cables y biestables, elementos básicos de los circuitos digitales, pero estos elementos están sin conectar.

Las FPGAs se programan normalmente usando un lenguaje específico (los dos más comunes son VHDL y Verilog) y que tras cargarlo en el integrado, es creado físicamente en el chip, logrando construir cualquier circuito digital.



## Ventajas

- Son más rápidos que los sistemas programables.
- Pueden trabajar en frecuencias altas.
- Las FPGAs se ejecutan de forma paralelo como circuitos independientes, no como los sistemas microprocesados que ejecutan un programa de forma secuencial.
- En el año 2015 se liberó el hardware (Clifford Wolf)

# Proyecto ICESTORM

Proyecto creado por Clifford Wolfek (2015) para liberar la FPGA Lattice.

Gracias a las herramientas libres que existen en este proyecto, se han creado otras herramientas, como, **Icestudio**, Software que deja accesible para muchas personas la síntesis y el diseño de FPGAs.

En España, la comunidad FPGA Wars, con la intención de desarrollar software y hardware libre, ha desarrollado el software Icestudio y la tarjeta electrónica Alhambra II.

## Comunidad FPGA Wars

En España la comunidad FPGA Wars ha sido creada para promover el desarrollo de software y hardware libre.

<http://fpgawars.github.io/>

### **Participantes**

Juan González (Obijuan)

Jesús Arroyo Torrens

Eladio Delgado

Andrés Prieto-Moreno

### **Proyectos**

Entre los proyectos que nos ayudaran a trabajar con las FPGAs destacamos los siguientes dos proyectos:

1. Icezun Alhambra II: Placa de desarrollo de FPGA (FPGA de Lattice iCE40HX1K-TQ144), placa diseñada por Eladio González en colaboración con Juan González.
2. ICEstudio: Software creado por Juan Arroyo que nos permitirá diseñar gráficamente los circuitos necesarios para programar la FPGA.

# ICEstudio

El software ICEstudio, nos permitirá crear circuitos digitales, sintetizarlo y descargarlo a la placa Alhambra II donde esta nuestra FPGA.

Este software es multiplataforma y software libre.

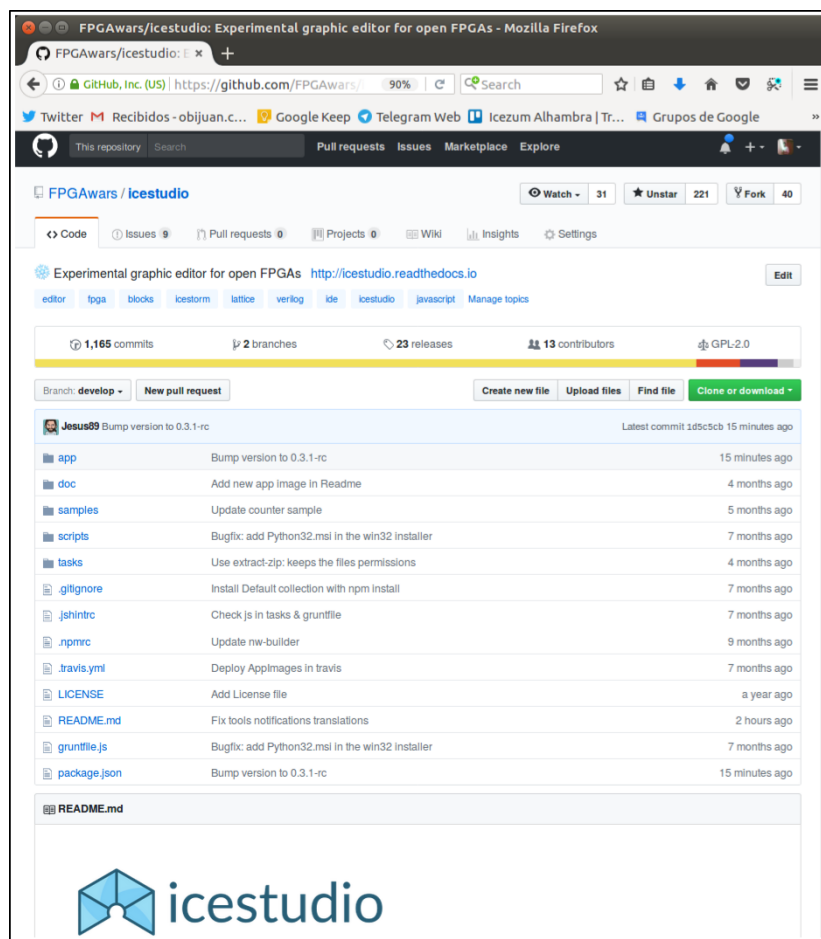
Nosotros explicaremos los pasos a seguir para instalarlo en el sistema operativo de windows

## Instalación del ICEstudio

### 1. Descargar el instalador de ICEstudio

Vamos a la página del ICEstudio en github

<https://github.com/FPGAwards/icestudio>

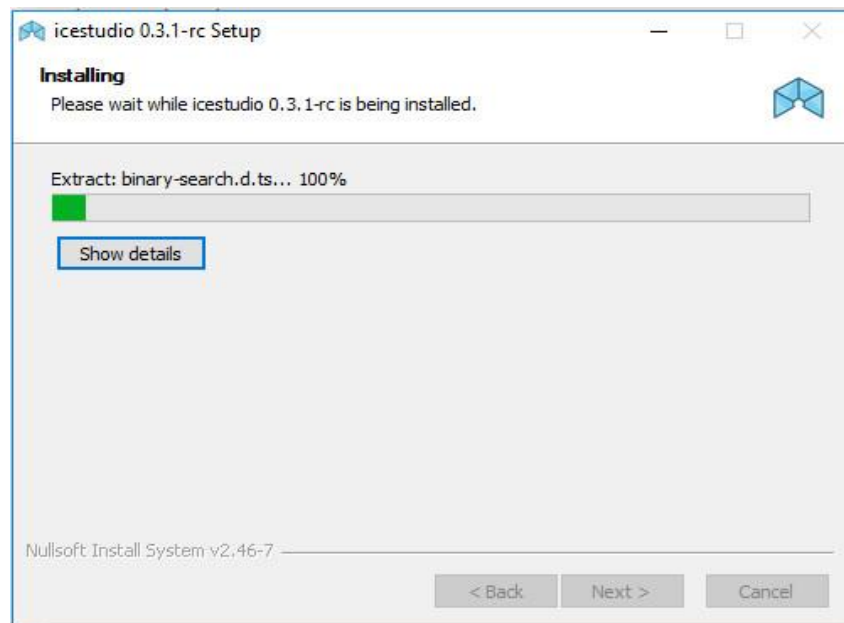


Y pinchamos en la opción **Release** donde nos aparecerán todas las versiones liberadas de ICEstudio existentes. Elegimos la versión que nos interese. En nuestro caso hemos utilizado la versión 0.4.0-dev para Windows de 64 bits.

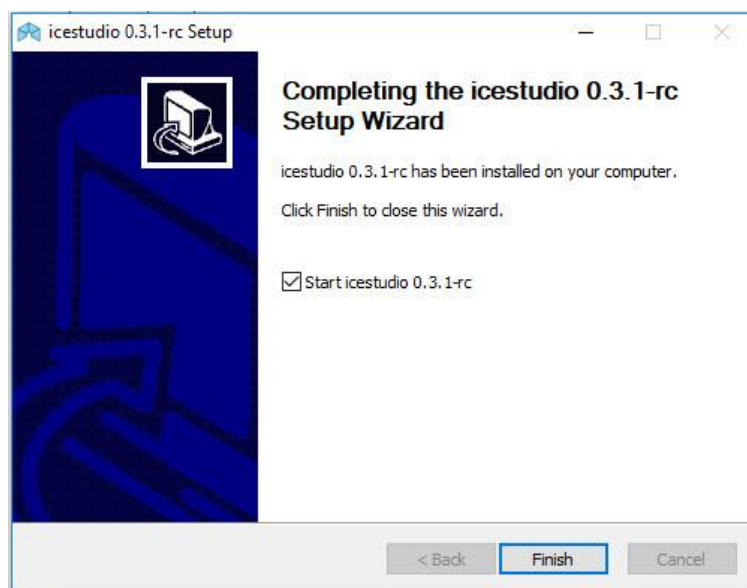
## 2. Ejecutar el instalador

Una vez descargado el instalador, lo ejecutaremos haciendo doble click. Dependiendo el antivirus que tengamos instalados nos podrán aparecer distintos mensajes que tendremos que obviar. También nos puede preguntar si permitimos que el instalador realice cambios, en ese caso le indicaremos que sí.

Se nos abre el instalador y le damos a **Siguiente**. El instalador detecta automáticamente si tenemos instalado **Python 2.7**. Si no lo está, se procede a su instalación, y luego a la de Icestudio.

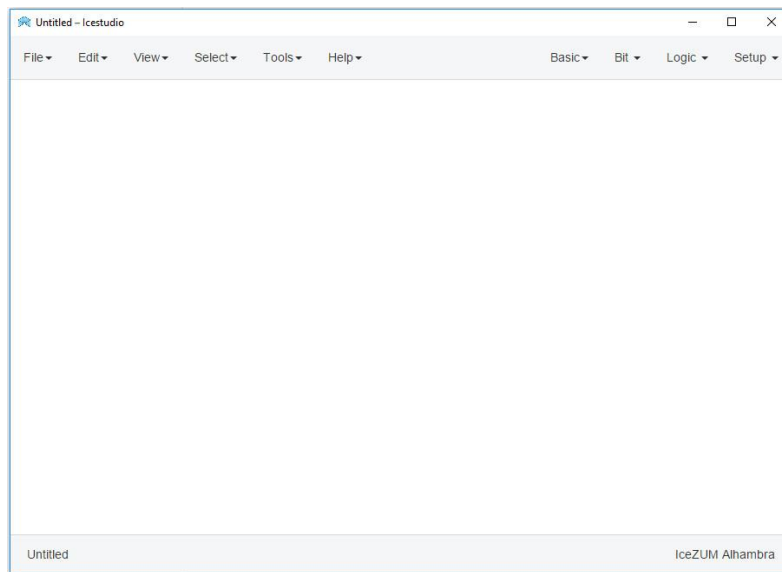


Al terminar la instalación nos saldrá un mensaje parecido al siguiente:



Clickaremos en Finalizar

Una vez Finalizado se abrirá el software ICestudio y aparecerá una ventana similar a la que a continuación se muestra.

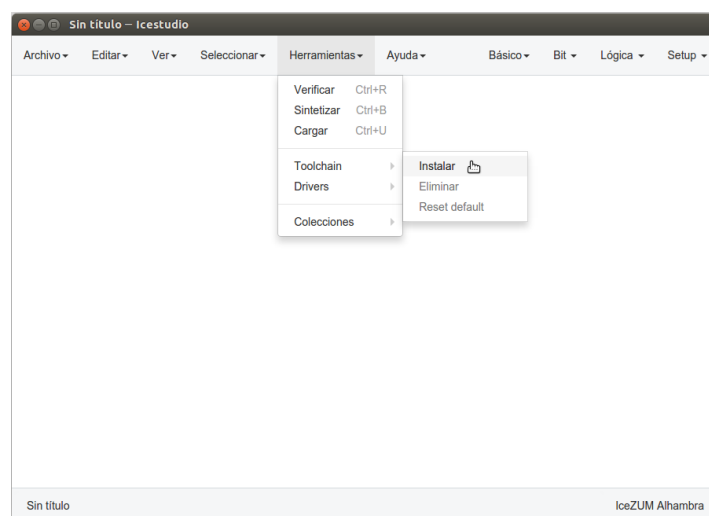


### 3. Configurar el idioma

Arrancamos ICestudio y lo primero que debemos hacer es configurar el idioma. Nos vamos al menú **Edit/Preferences/Language** y seleccionamos el idioma que queramos.

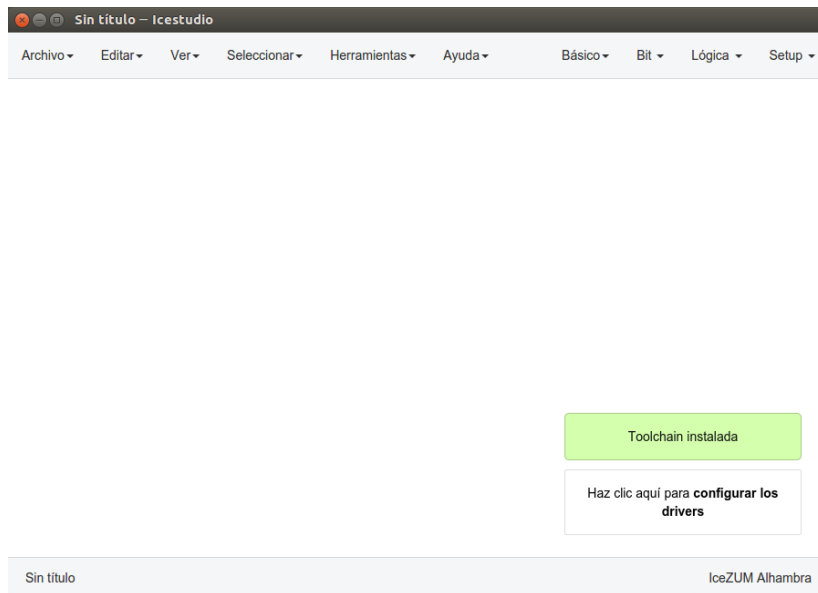
### 4. Instalar el toolchain

Con lo que tenemos hasta ahora podemos abrir ejemplos y modificarlos pero no podemos sintetizarlos en la FPGA, para ello tenemos que instalar las herramientas (la toolchain). Para hacerlo nos vamos a **Herramientas/Toolchain/Instalar**



Comienza su instalación. Nos aparecerá una ventana con una **barra de progreso**. Este proceso puede durar unos minutos.

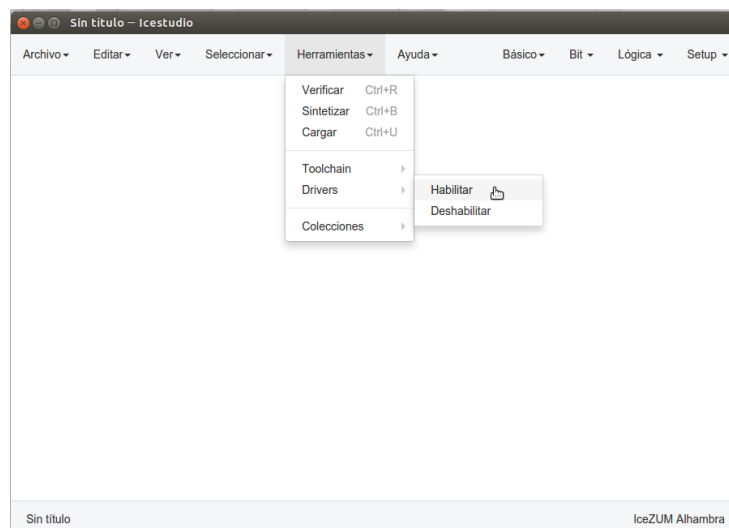
Al terminar, nos aparecerá una **notificación** verde en la parte inferior derecha de la pantalla indicando que la **toolchain** está instalada.



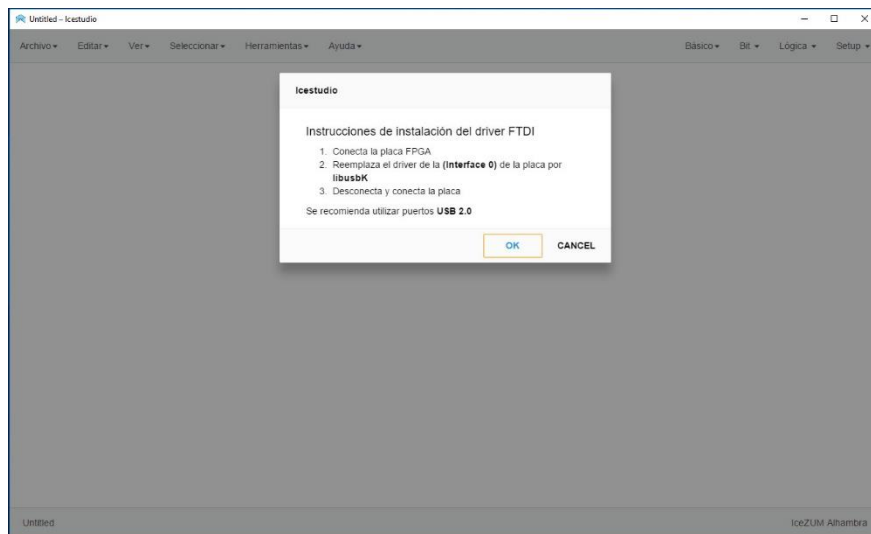
Alguna vez hemos tenido algún problema a la hora de instalar el toolchain al estar conectados por wifi. Lo hemos solucionado conectando el ordenador a red mediante cable.

## 5. Instalar los drivers

En el punto en el que estamos ya podemos sintetizar el circuito (se generara el bitstream) pero no podemos cargarlo en la placa con la FPGA libres, para poder hacerlo deberemos habilitar los drivers.



Nos vamos a la opción del menú **Herramientas/Drivers/Habilitar**. Para la plataforma de Windows, hay que tener la placa de FPGA conectada y nos aparecerán los pasos a seguir.



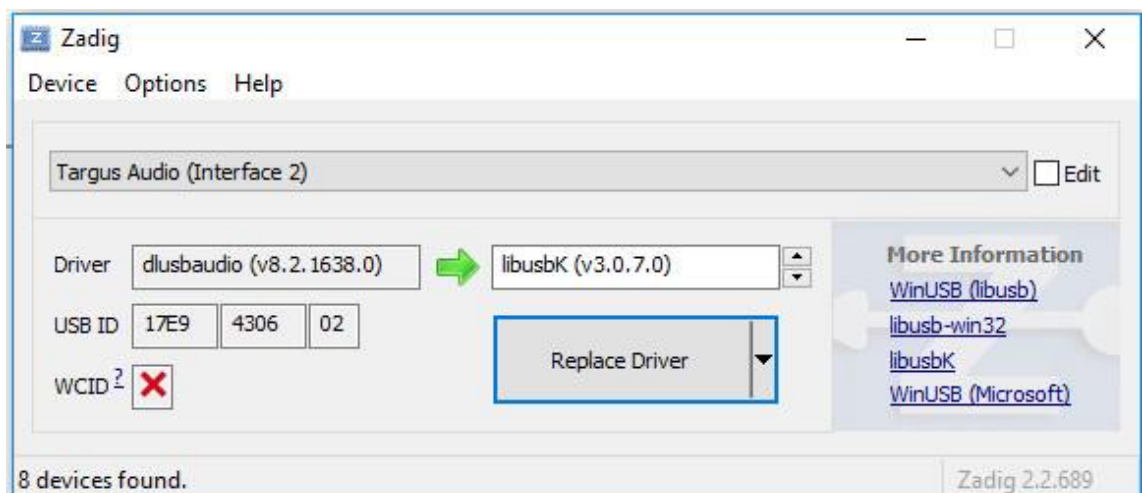
Una vez en este punto los pasos a seguir para instalar los drivers son los que a continuación enumeramos

1. Conectar la placa de la FPGA al USB

Se recomienda conectar la placa de la FPGA a un **USB 2.0**. Una vez instalado el driver, la placa de FPGA siempre deberá de conectarse al USB donde teníamos conectada la placa a la hora de instalar los drivers.

2. Seleccionar el driver

Cuando pulsamos Ok, ICEstudio lanzara la aplicación Zadig (que se ha instalado al instalar ICEstudio) que nos permitirá instalar el driver de la FPGA. Aparecerá una ventana como la siguiente:





Al pulsar en el desplegable, deberemos elegir la tarjeta de FPGA que utilizaremos, en nuestro caso, Alhambra II y dentro de las dos opciones que nos aparecen con la Alhambra II elegiremos la que pone **Interface 0**.

El desplegable se cerrará y nos aparecerá lo que hemos seleccionado. En la casilla que está encima del botón de "Replace driver" debemos seleccionar el **driver libusK** (es muy probable que sea el que aparece por defecto), pero si no, lo seleccionamos con las flechas. Ahora apretamos **Replace driver** y comienza la instalación del driver.

En unos segundos nos aparecerá el mensaje de que se ha **instalado correctamente**.

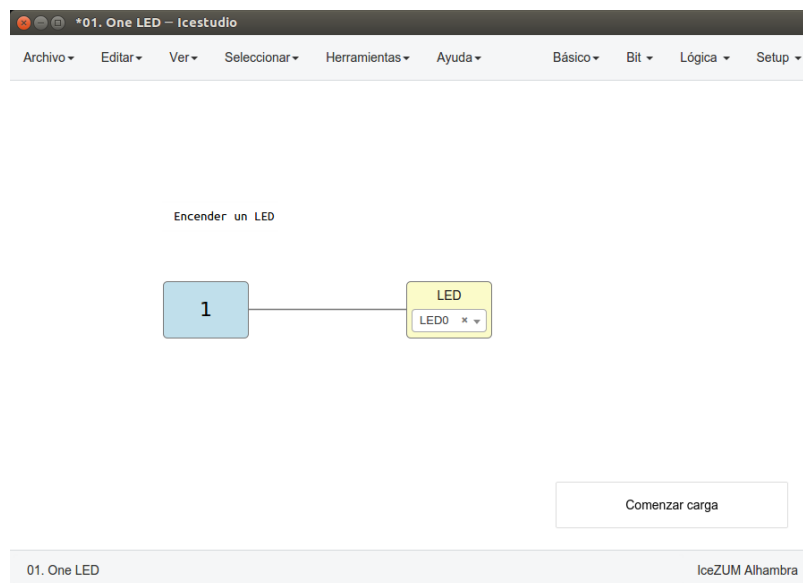
### 3. Desconectar y conectar la placa

Una vez instalado el driver, deberemos de desconectar y volver a conectar la placa de FPGA para que el driver funcione correctamente.

### 6. Cargar el circuito

Ya podemos cargar el circuito sintetizado, para ello dibujaremos un circuito o cargaremos uno de los ejemplos existentes.

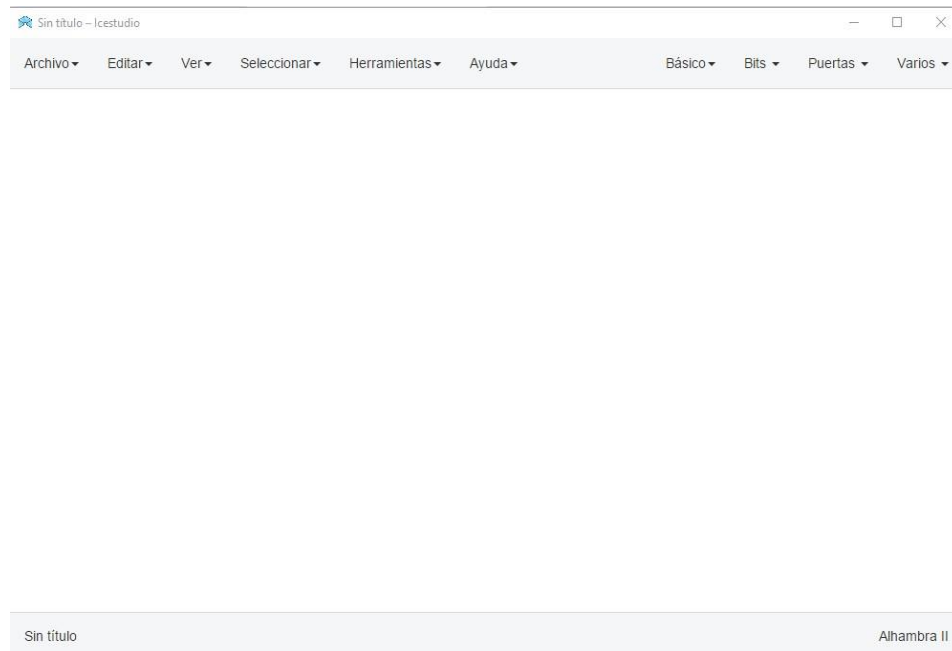
Una vez que tenemos el circuito que queremos cargar, deberemos de elegir la opción **Herramientas/Cargar** y nos aparecerá una notificación indicando que comienza la carga.



Una vez terminada la carga nos aparece una notificación verde indicándonos que la carga se ha realizado correctamente y el circuito se puede comprobar.

## Software ICEstudio

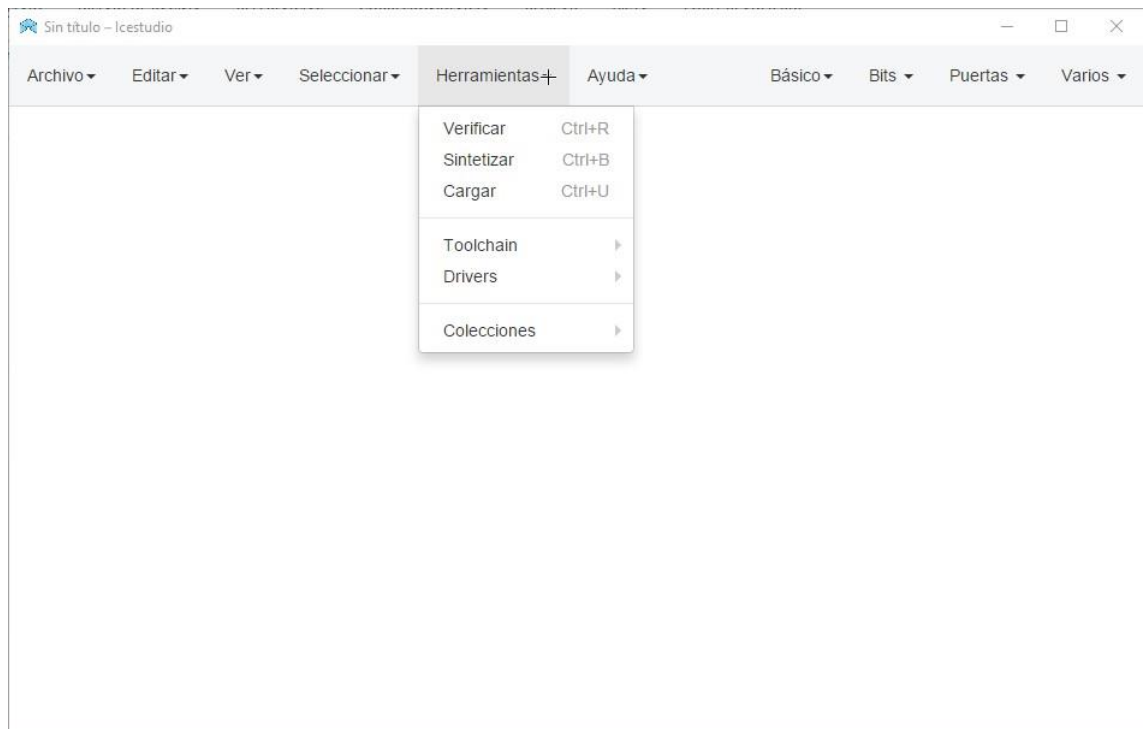
En la imagen que se muestra a continuación nos fijaremos en las opciones de **seleccionar** y **herramientas** del software ICEstudio.



**Seleccionar:** En esta opción seleccionaremos la placa con la que trabajaremos que en nuestro caso es la Alhambra II y la colección que queremos cargar.

A la hora de programar la FPGA nos basaremos en distintos elementos gráficos que al conectarse entre ellos crearemos el circuito que nos hace falta. Estos elementos estarán disponibles en colecciones que podemos crear o cargar. Una vez seleccionado la colección necesaria, nos aparecerá en la parte derecha del menú una organización de opciones que dependerá de la colección seleccionada. En la imagen superior nos aparecen (Básico, Bits, Puertas, Varios) por que la colección seleccionada está organizada de esa forma.

**Herramientas:** Tal como se muestra en la siguiente imagen, en la opción Herramientas hay tres grupos de opciones.



1. Opciones relacionadas con verificar el circuito que hemos dibujado, sintetizar el circuito y cargarlo en la FPGA.
2. Opción de cargar, actualizar o eliminar el toolchain así como habilitar o deshabilitar el driver.
3. Añadir, borrar colecciones.

## **Placa Alhambra II**

La placa Alhambra II ha sido desarrollada por Eladio Delgado Mingorance, y es una placa que nos permitirá probar nuestros circuitos diseñados con Icestudio.

La placa ha sido diseñada con herramientas de *código abierto* (FreeCad, KiCad, InkScape y LibreOffice) y es compatible con la cadena de herramientas (toolchain) de fuente abierta de Clifford Wolf.

Las características de la placa Alhambra son las que a continuación enumeramos:

- Placa de desarrollo FPGA (iCE40HX4K-TQ144 de Lattice)
- Hardware abierto
- Compatible con opensource icestorm toolchain e Icestudio
- Pinout similar a Arduino Uno
- Puedes utilizar la mayoría de los Shields
- Oscilador MEMS de 12 MHZ
- Interruptor ON / OFF (apaga tu robot móvil fácilmente)
- Fuente de alimentación USB. Dos conectores. Hasta 4.8A
- Pines analógicos (aunque el convertidor A / D incrustado i2c)
- 20 pines de entrada / salida 3.3v (5v tolerante)
- El dispositivo USB FTDI 2232H permite la programación FPGA y la interfaz UART a una PC
- Botón de reinicio
- 8 leds de propósito general (leds de usuario)
- 2 pulsadores de uso general
- Memoria flash de 4MB



Desde la siguiente dirección se puede comprar la placa y/o consultar la información sobre la placa Alhambra II:

<https://alhambrabits.com/alhambra/>

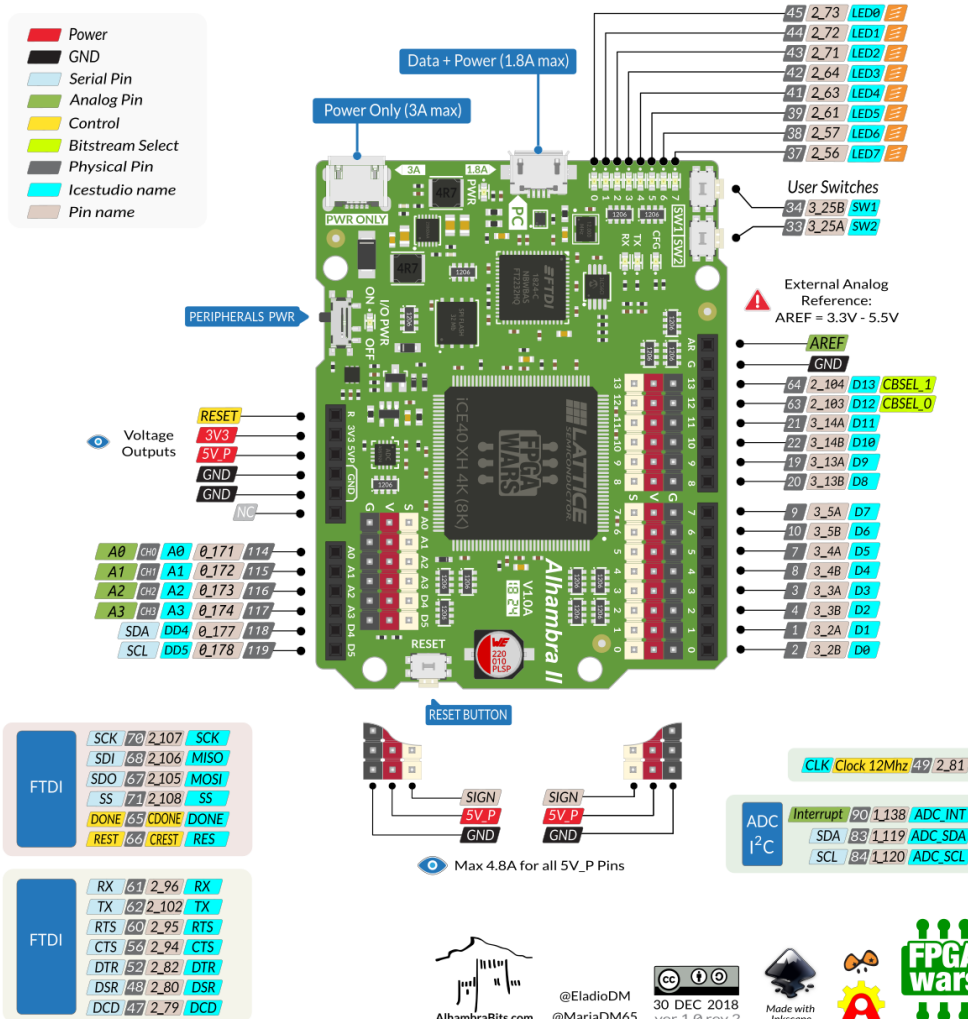
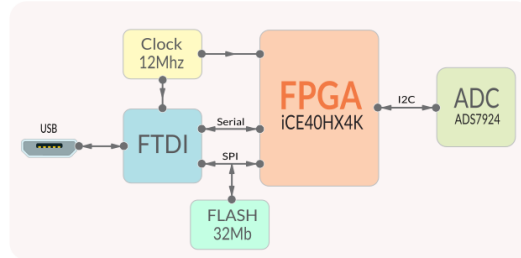
## Pinout Alhambra II

En la siguiente imagen mostramos el Pinout de la Alhambra II.

# Alhambra II<sup>V1.0A</sup>

## Open FPGA Board

- ⚠ Maximum Supply Voltage: 5.5V
- 🔌 Operating Supply Voltage: 3.5 - 5.5V
- ✅ 3.3V GPIO, 5V Compatible
- 🔌 Each GPIO Includes a 200 ohm Serial Resistor
- ✅ ADC Internal to External Ref. Automatic Switching

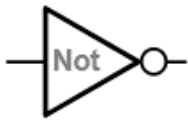
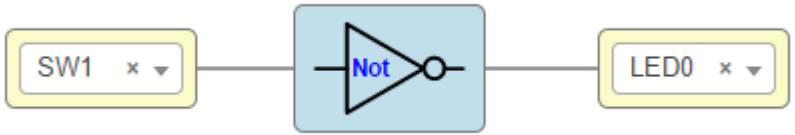
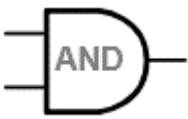
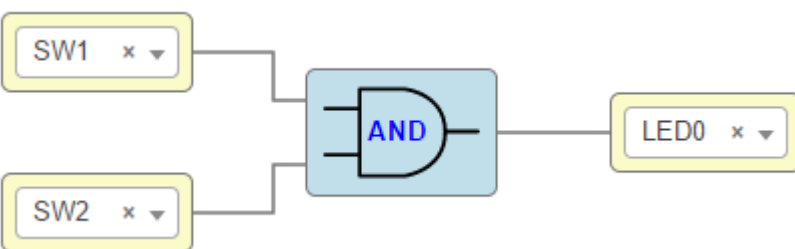
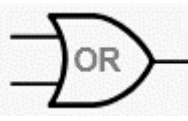
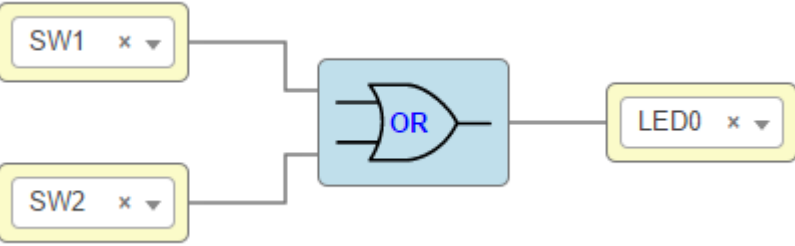



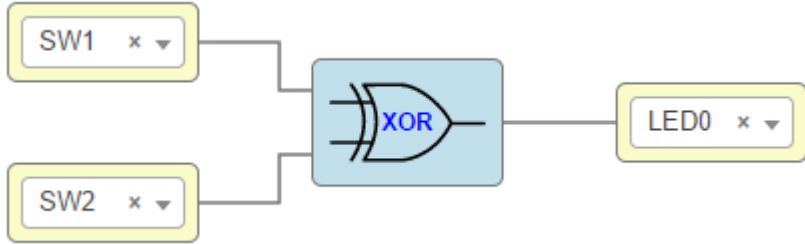
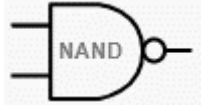
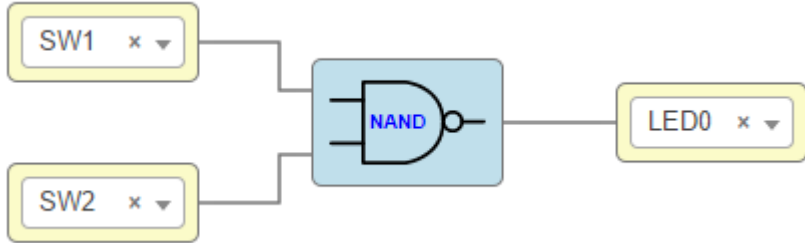

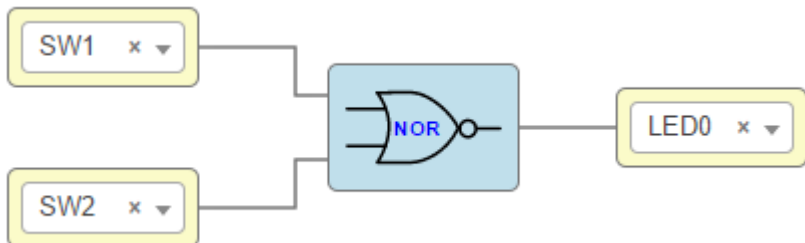

## Programación de las FPGA

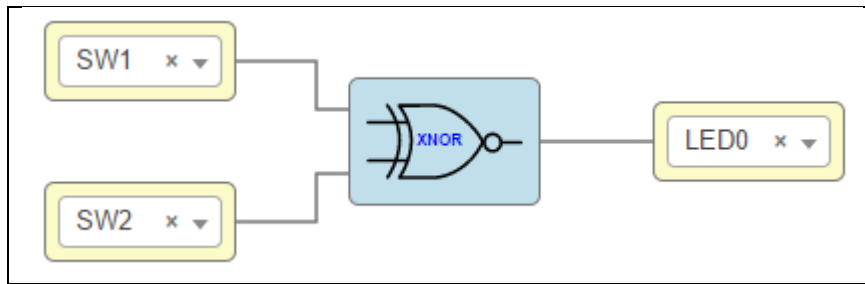
Vamos a empezar a programar nuestra FPGA. Para ello realizaremos diversos ejercicios que permitirán al alumnado y profesor poder realizar distintos circuitos digitales que probaremos en nuestra FPGA. Estos ejercicios nos servirán para practicar y conocer los distintos elementos que se pueden utilizar en circuitos digitales y a su vez aprenderemos a programar nuestra FPGA.

### Ejercicio 1

Rellenar la tabla de la verdad de las puertas lógicas básicas y verificar en Icestudio.

	<table border="1"><thead><tr><th>SW1</th><th>LED0</th></tr></thead><tbody><tr><td>0</td><td></td></tr><tr><td>1</td><td></td></tr></tbody></table>	SW1	LED0	0		1										
SW1	LED0															
0																
1																
																
	<table border="1"><thead><tr><th>SW1</th><th>SW2</th><th>LED0</th></tr></thead><tbody><tr><td>0</td><td>0</td><td></td></tr><tr><td>0</td><td>1</td><td></td></tr><tr><td>1</td><td>0</td><td></td></tr><tr><td>1</td><td>1</td><td></td></tr></tbody></table>	SW1	SW2	LED0	0	0		0	1		1	0		1	1	
SW1	SW2	LED0														
0	0															
0	1															
1	0															
1	1															
																
	<table border="1"><thead><tr><th>SW1</th><th>SW2</th><th>LED0</th></tr></thead><tbody><tr><td>0</td><td>0</td><td></td></tr><tr><td>0</td><td>1</td><td></td></tr><tr><td>1</td><td>0</td><td></td></tr><tr><td>1</td><td>1</td><td></td></tr></tbody></table>	SW1	SW2	LED0	0	0		0	1		1	0		1	1	
SW1	SW2	LED0														
0	0															
0	1															
1	0															
1	1															
																

	<table><tr><th>SW1</th><th>SW2</th><th>LED0</th></tr><tr><td>0</td><td>0</td><td></td></tr><tr><td>0</td><td>1</td><td></td></tr><tr><td>1</td><td>0</td><td></td></tr><tr><td>1</td><td>1</td><td></td></tr></table>	SW1	SW2	LED0	0	0		0	1		1	0		1	1	
SW1	SW2	LED0														
0	0															
0	1															
1	0															
1	1															
																
	<table><tr><th>SW1</th><th>SW2</th><th>LED0</th></tr><tr><td>0</td><td>0</td><td></td></tr><tr><td>0</td><td>1</td><td></td></tr><tr><td>1</td><td>0</td><td></td></tr><tr><td>1</td><td>1</td><td></td></tr></table>	SW1	SW2	LED0	0	0		0	1		1	0		1	1	
SW1	SW2	LED0														
0	0															
0	1															
1	0															
1	1															
																
	<table><tr><th>SW1</th><th>SW2</th><th>LED0</th></tr><tr><td>0</td><td>0</td><td></td></tr><tr><td>0</td><td>1</td><td></td></tr><tr><td>1</td><td>0</td><td></td></tr><tr><td>1</td><td>1</td><td></td></tr></table>	SW1	SW2	LED0	0	0		0	1		1	0		1	1	
SW1	SW2	LED0														
0	0															
0	1															
1	0															
1	1															
																
	<table><tr><th>SW1</th><th>SW2</th><th>LED0</th></tr><tr><td>0</td><td>0</td><td></td></tr><tr><td>0</td><td>1</td><td></td></tr><tr><td>1</td><td>0</td><td></td></tr><tr><td>1</td><td>1</td><td></td></tr></table>	SW1	SW2	LED0	0	0		0	1		1	0		1	1	
SW1	SW2	LED0														
0	0															
0	1															
1	0															
1	1															



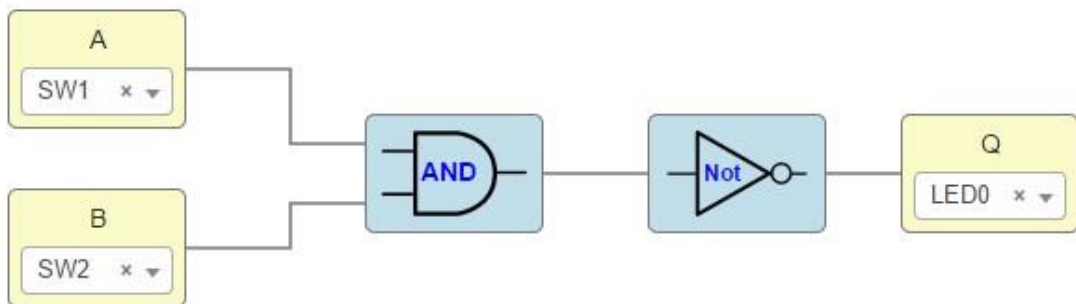
## Ejercicio 2

Dibuja las siguientes puertas utilizando las puertas lógicas básicas AND, OR y NOT.

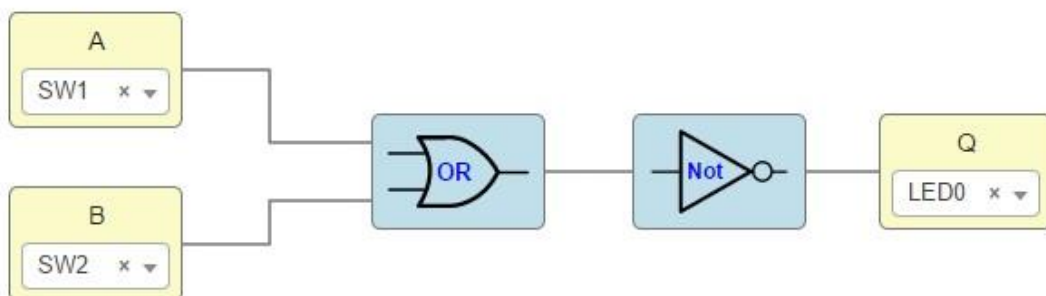
- a) NAND
- b) NOR
- c) XOR
- d) XNOR

## Ejercicio 2 Resuelto

- a) NAND

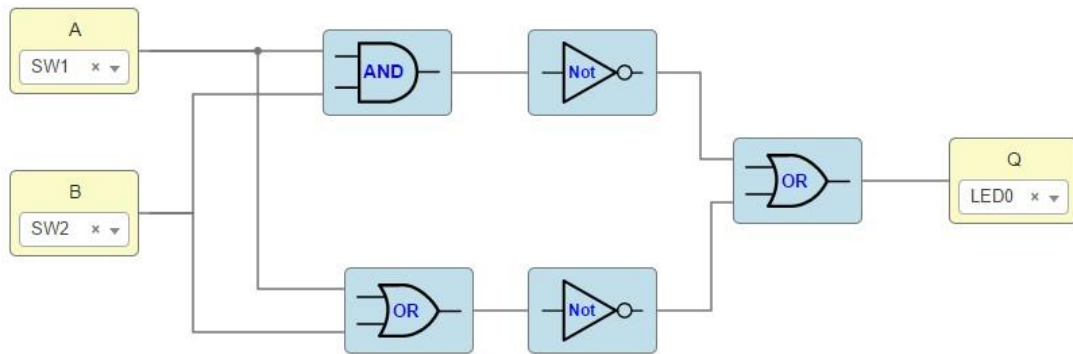


- b) NOR

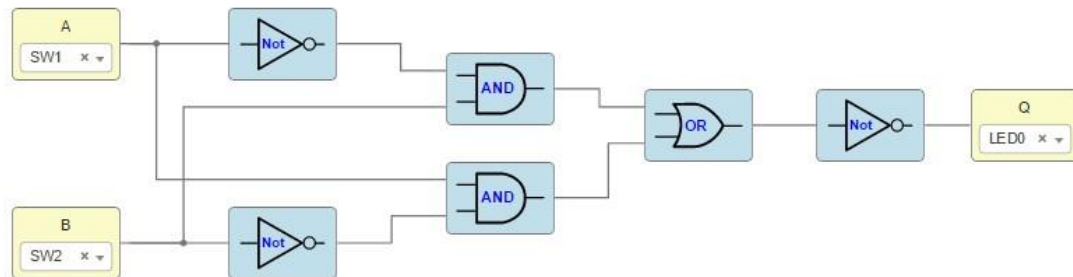




c) XOR



d) XNOR

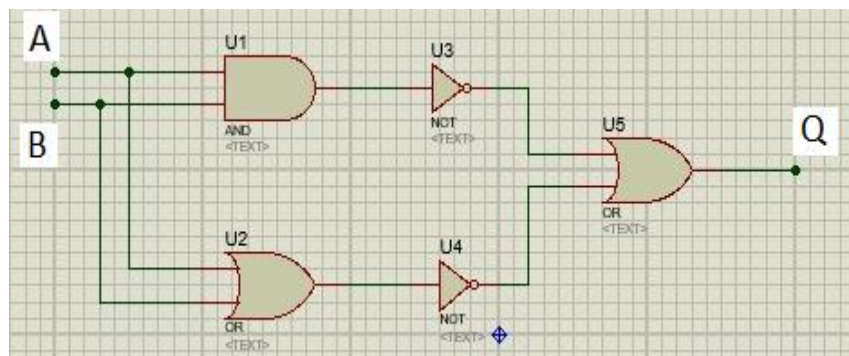


### Ejercicio 3

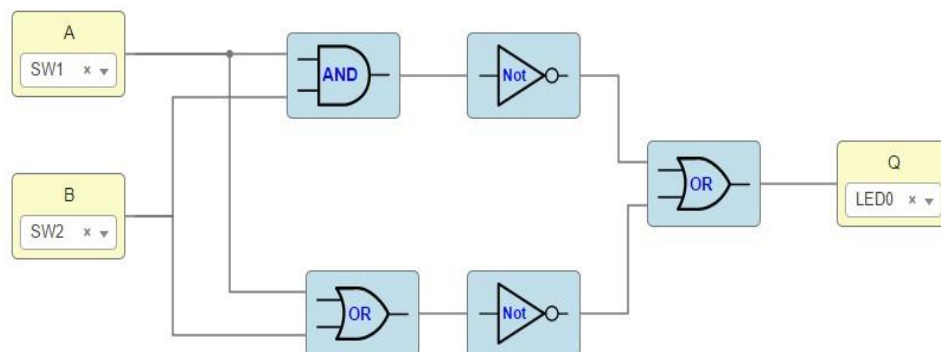
Validar los siguientes circuitos digitales en Icestudio y rellenar las tablas de la verdad.

A	B	Q
0	0	
0	1	
1	0	
1	1	

#### Circuito 1

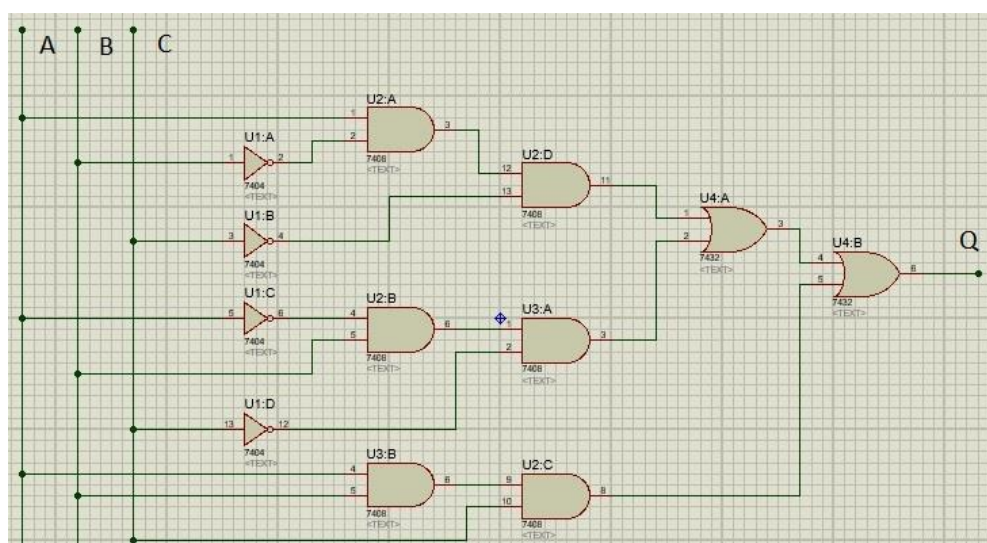


#### Circuito 1 Resuelto



A	B	Q
0	0	1
0	1	1
1	0	1
1	1	0

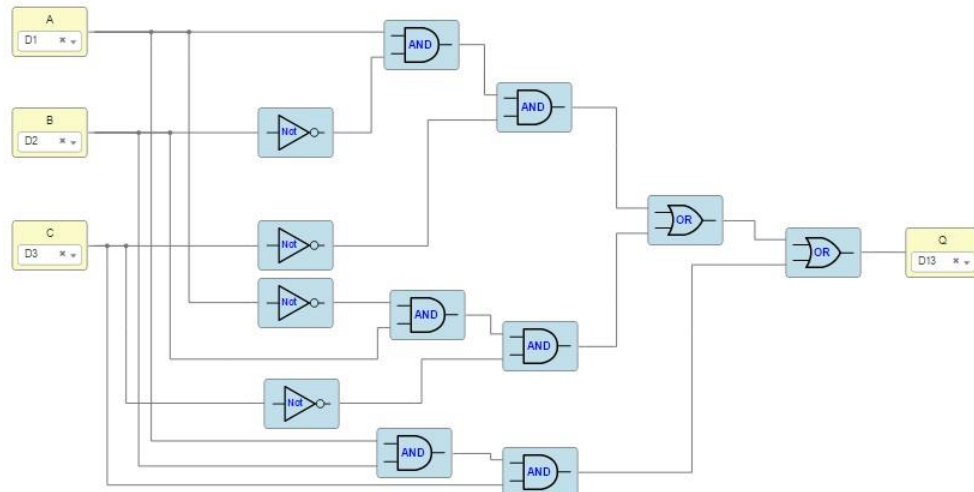
## Circuito 2



A	B	C	Q
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

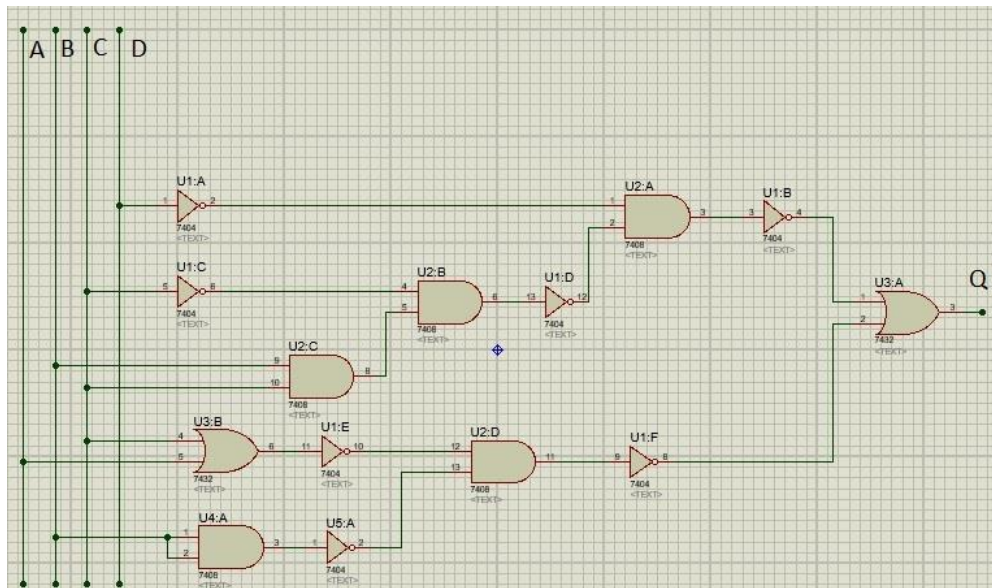
## Circuito 2 Resuelto

Para resolver el siguiente circuito en la FPGA conectaremos las tres entradas (3 interruptores) en tres pines digitales de la FPGA, en nuestro caso en D1, D2 y D3. La salida (mediante un led) la conectaremos al pin digital D13.



A	B	C	Q
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

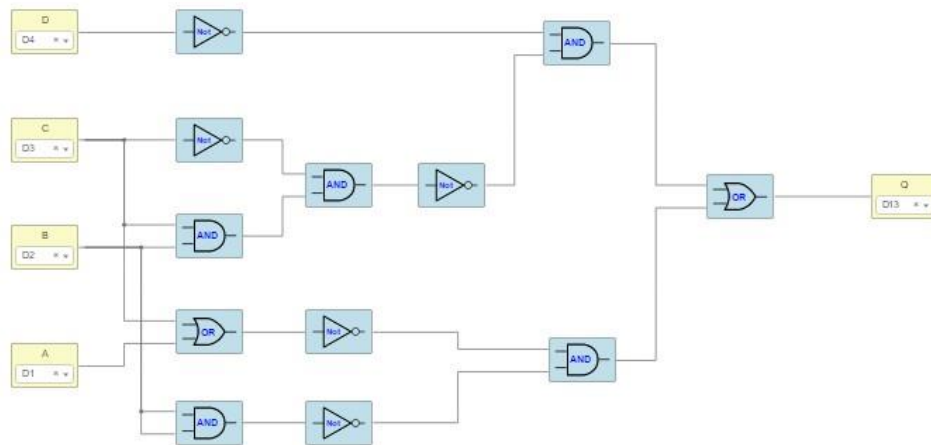
### Circuito 3



A	B	C	D	Q
0	0	0	0	
0	0	0	1	
0	0	1	0	
0	0	1	1	
0	1	0	0	
0	1	0	1	
0	1	1	0	
0	1	1	1	
1	0	0	0	
1	0	0	1	
1	0	1	0	
1	0	1	1	
1	1	0	0	
1	1	0	1	
1	1	1	0	
1	1	1	1	

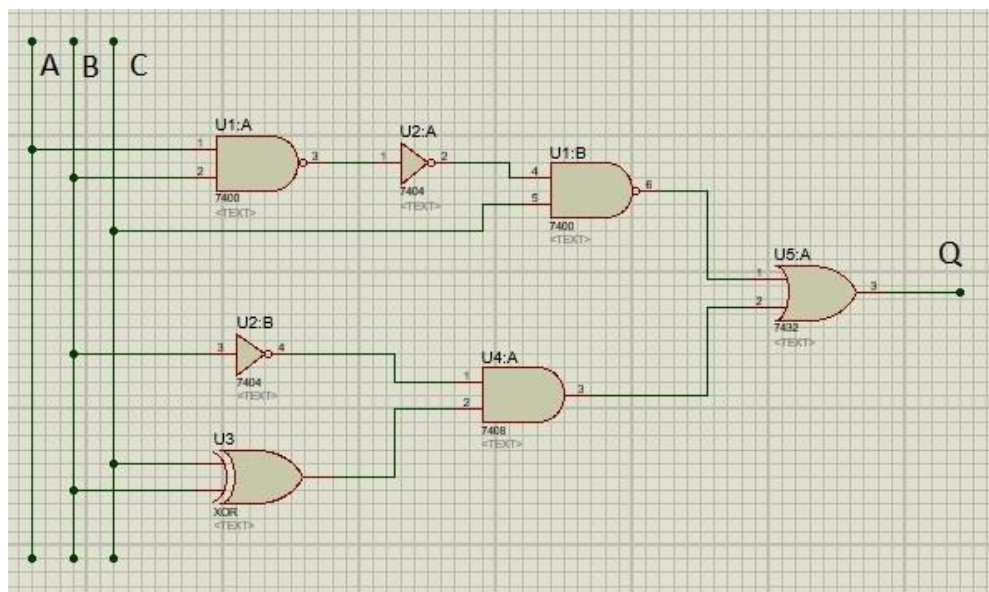
### Circuito 3 Resuelto

Para resolver el siguiente circuito en la FPGA conectaremos las cuatro entradas (4 interruptores) en cuatro pines digitales de la FPGA, en nuestro caso en D1, D2, D3 y D4. La salida (mediante un led) la conectaremos al pin digital D13.



A	B	C	D	Q
0	0	0	0	0
0	0	0	1	1
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	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	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

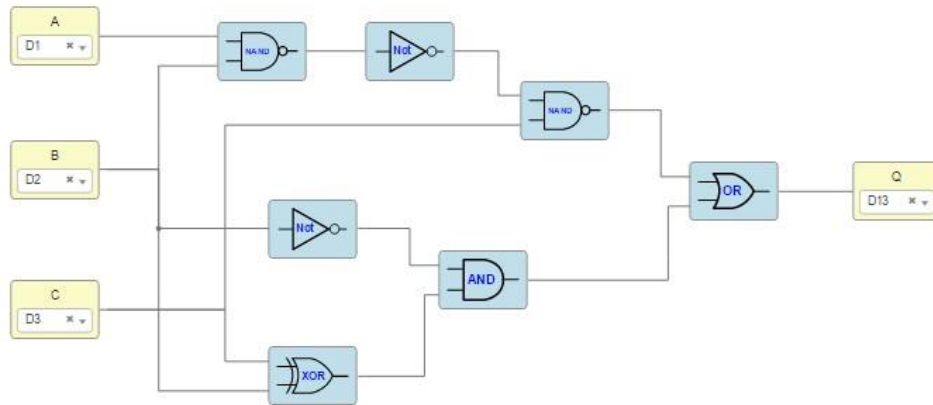
## Circuito 4



A	B	C	Q
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

#### Circuito 4 Resuelto

Para resolver el siguiente circuito en la FPGA conectaremos las tres entradas (3 interruptores) en tres pines digitales de la FPGA, en nuestro caso en D1, D2 y D3. La salida (mediante un led) la conectaremos al pin digital D13.



A	B	C	Q
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0



#### Ejercicio 4

Obtener las funciones de los circuitos anteriores mediante suma de productos (Minterm) y producto de sumas (Maxterm).

##### Circuito 1 Resuelto

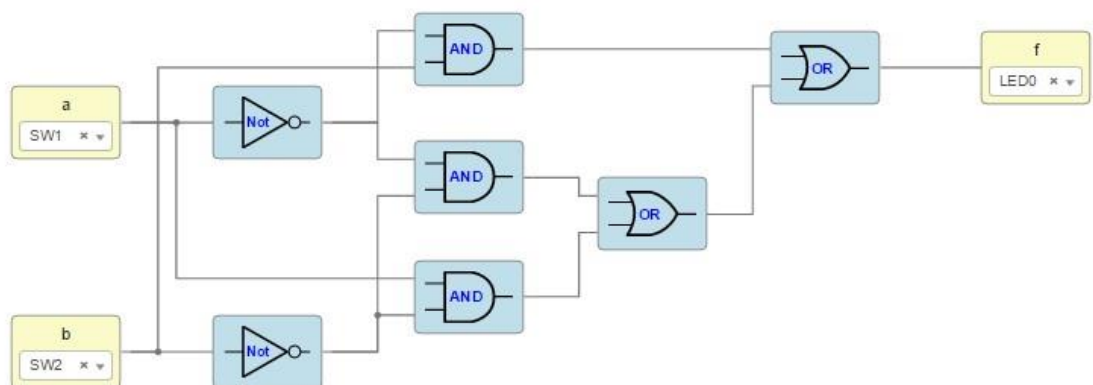
Teniendo en cuenta que el circuito 1 tiene la siguiente tabla de la verdad:

A	B	Q
0	0	1
0	1	1
1	0	1
1	1	0

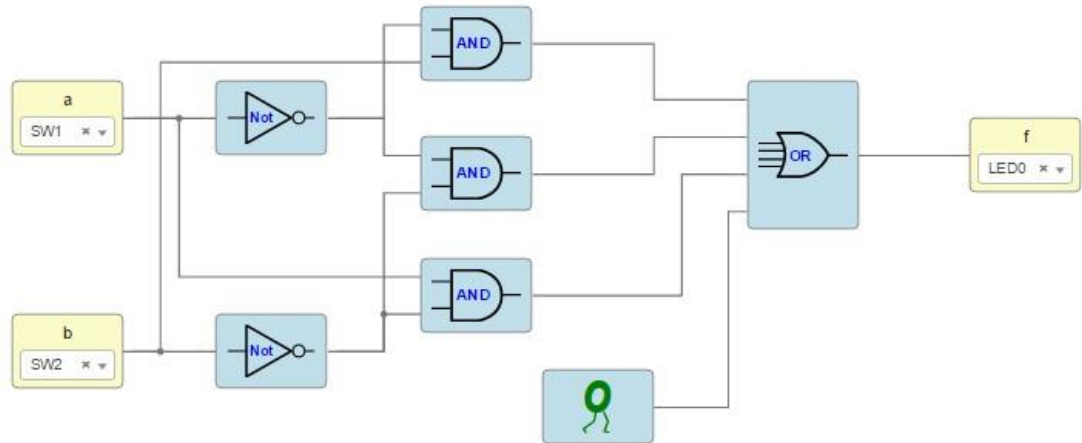
La ecuación Minterm queda de la siguiente manera:

$$f = \bar{a} \bar{b} + a \bar{b} + \bar{a} b$$

El circuito

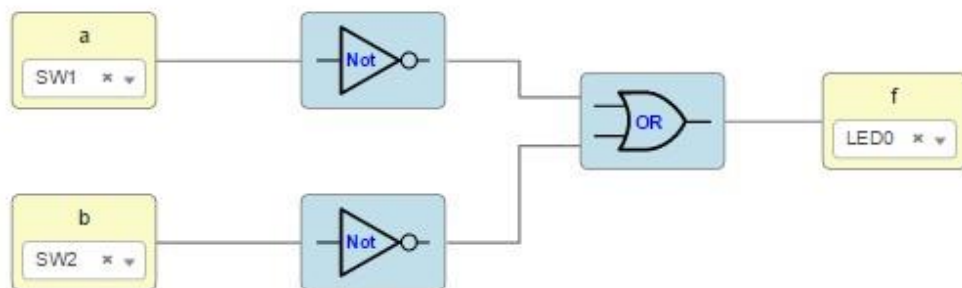


También puedo hacer el mismo ejercicio utilizando puertas OR de 4 entradas. Como tengo que sumar tres términos en la cuarta entrada debería de introducir un "0" para que la función no varié. El circuito utilizando puertas OR de cuatro entradas nos queda de la siguiente manera:



La ecuación de forma maxterm queda de la siguiente manera:

$$f = \bar{a} + \bar{b}$$

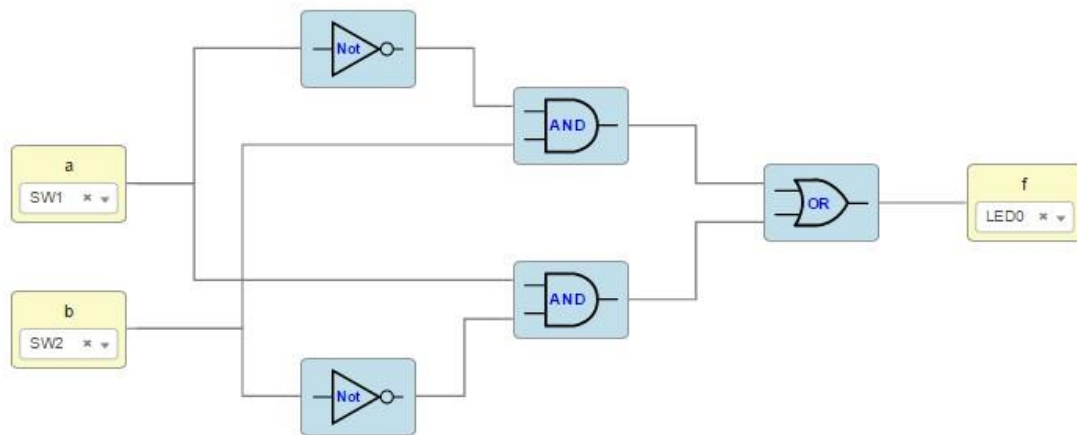


### Ejercicio 5

Dibuja los esquemas digitales de las siguientes funciones.

$$a) Y_{SOP} = (\bar{A} \cdot B) + (A \cdot \bar{B})$$

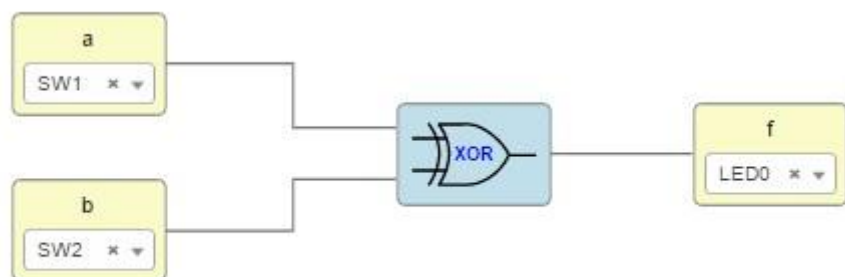
Circuito Resuelto



Según este circuito la tabla de la verdad queda de la siguiente manera:

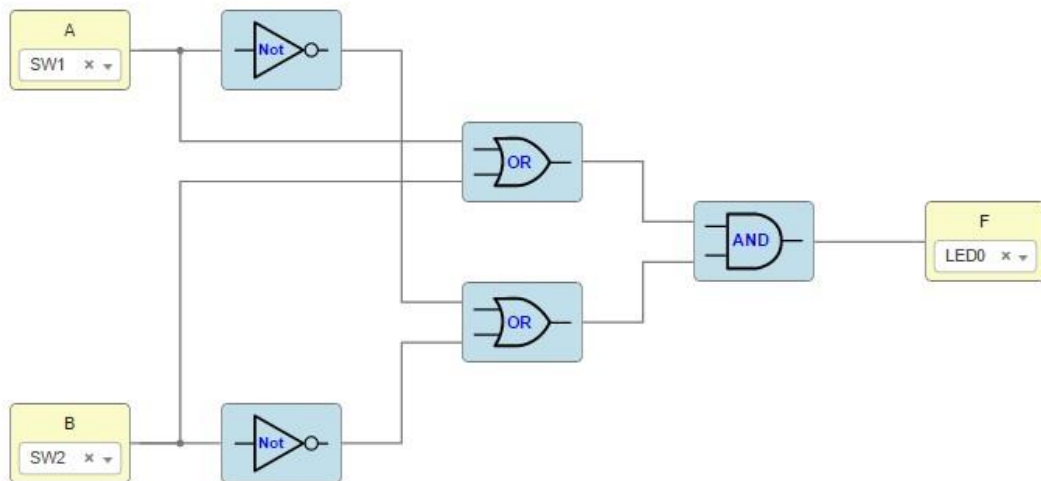
A	B	Q
0	0	0
0	1	1
1	0	1
1	1	0

Esta tabla de la verdad corresponde a la puerta lógica XOR, por lo tanto otra opción es la que a continuación se dibuja:



b)  $Y_{POS} = (A + B) \cdot (\bar{A} + \bar{B})$

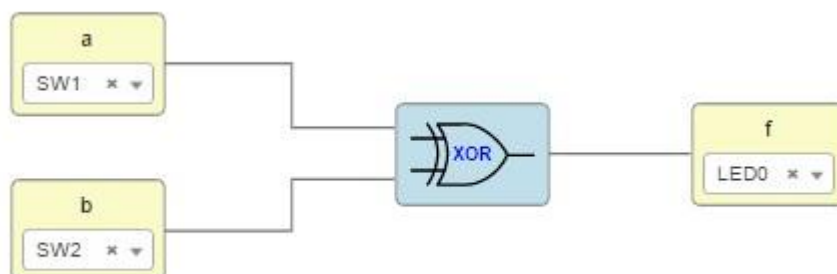
Circuito Resuelto



Según este circuito la tabla de la verdad queda de la siguiente manera:

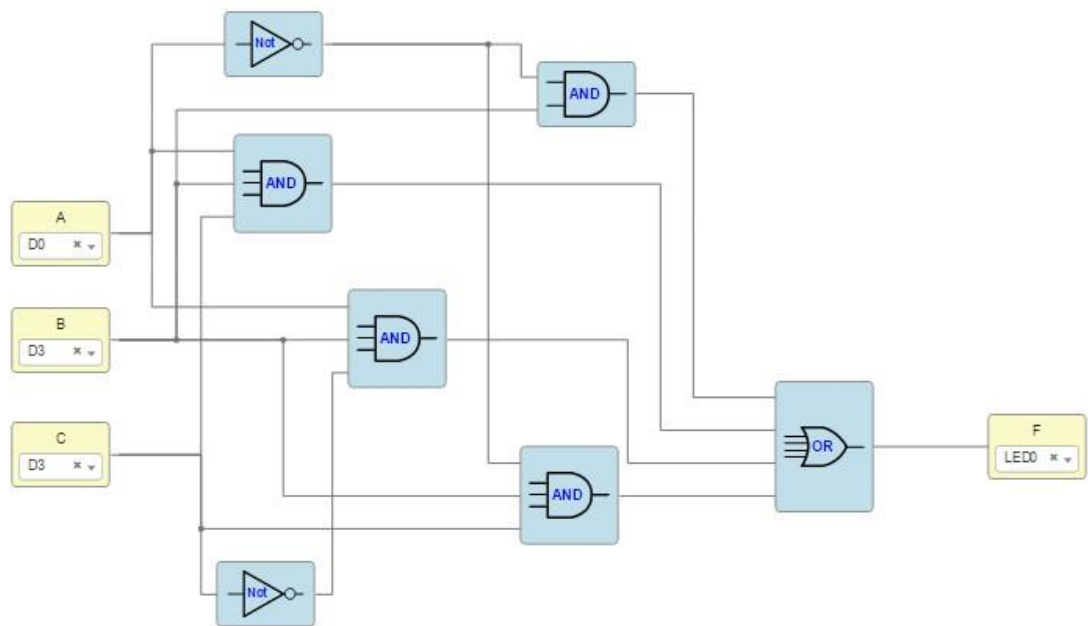
A	B	Q
0	0	0
0	1	1
1	0	1
1	1	0

Tal como se puede ver este circuito es equivalente al anterior, por lo tanto también se puede realizar el montaje con una puerta lógica XOR.



c)  $Y = A \cdot B \cdot C + A \cdot B \cdot \bar{C} + \bar{A} \cdot B \cdot C + \bar{A} \cdot B$

Circuito Resuelto



Según este circuito la tabla de la verdad queda de la siguiente manera:

A	B	C	Q
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

### Ejercicio 6

¿Es posible simplificar la última función del ejercicio anterior? Investiga que es un mapa de Karnaugh y el teorema de DeMorgan.

Solución:

$$Y = A \cdot B \cdot C + A \cdot B \cdot \bar{C} + \bar{A} \cdot B \cdot C + \bar{A} \cdot B$$

$$Y = A \cdot B \cdot (C + \bar{C}) + \bar{A} \cdot B \cdot (C + 1)$$

$$Y = A \cdot B + \bar{A} \cdot B$$

$$Y = (A + \bar{A}) \cdot B$$

$$Y = B$$

Tal como puedes observar, la salida de la tabla de la verdad da exactamente igual que la entrada B, por lo tanto todo el circuito anterior se puede montar como en la imagen siguiente:



### Ejercicio 7

Dar solución a los siguientes problemas utilizando puertas lógicas.

a) problema:

Queremos implementar dos alarmas. A las luces del sistema les llamaremos A y B. El sistema tendrá tres entradas X, Y y Z. El sistema funcionara con la siguiente lógica:

- La alarma A se activara cuando se active la entrada X exclusivamente.
- La alarma B se activara cuando se active la entrada Z exclusivamente.
- Las dos alarmas se activaran a la vez cuando se activen dos entradas cualesquiera a la vez.

Rellenaremos la tabla de la verdad con los condicionantes marcados por la lógica, quedando la tabla de la verdad de la siguiente manera:

X	Y	Z	A	B
0	0	0	0	0
0	0	1	0	1
0	1	0	0	0
0	1	1	1	1
1	0	0	1	0
1	0	1	1	1
1	1	0	1	1
1	1	1	1	1

Utilizando las tablas de Karnaugh y simplificando por minterm la ecuación queda de la siguiente manera:

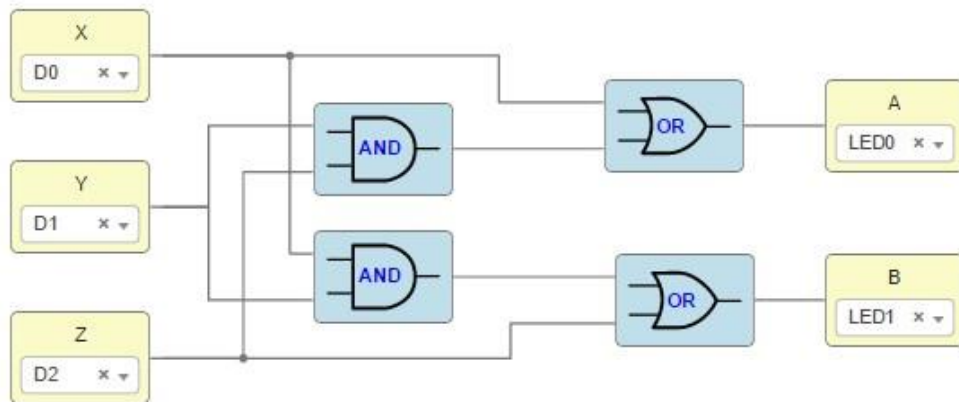
z\xy	00	01	11	10
0	0	0	1	1
1	0	1	1	1

$$A = X + (YZ)$$

z\xy	00	01	11	10
0	0	0	1	0
1	1	1	1	1

$$B = Z + (XY)$$

Quedando el circuito de la siguiente manera:



Utilizando las tablas de Karnaugh y simplificando por maxterm la ecuación queda de la siguiente manera:

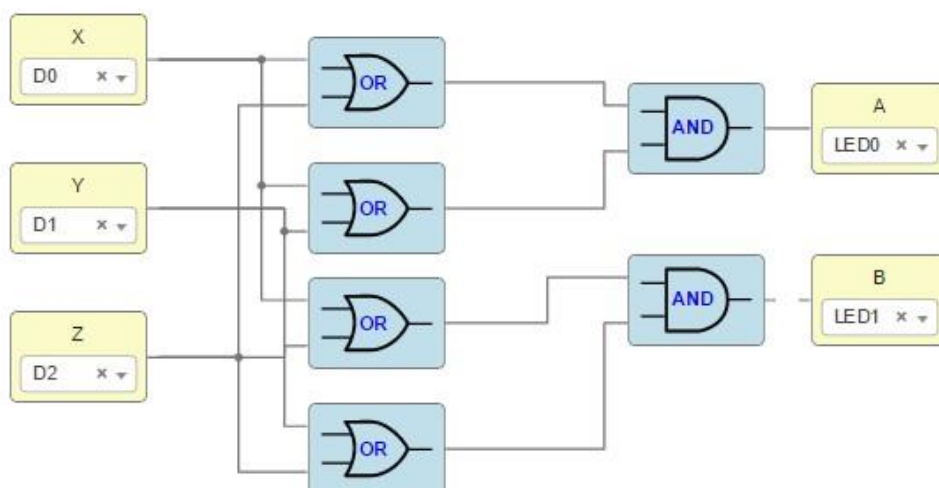
z\xy	00	01	11	10
0	0	0	1	1
1	0	1	1	1

$$A = (X + Z)(X + Y)$$

z\xy	00	01	11	10
0	0	0	1	0
1	1	1	1	1

$$B = (X + Z)(Y + Z)$$

Quedando el circuito de la siguiente manera:





### Ejercicio 8

Conseguir las siguientes puertas equivalentes utilizando las puertas NAND.

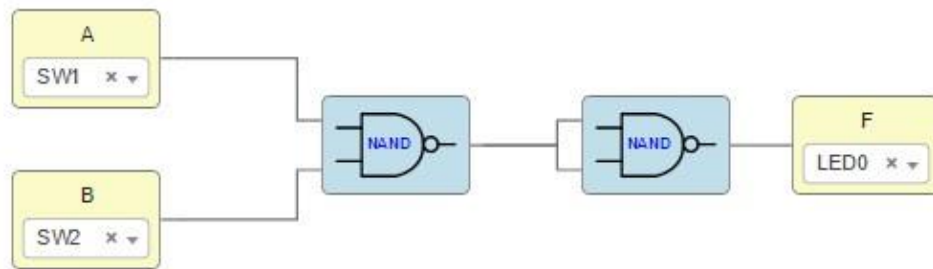
a) NOT

La puerta NOT se puede conseguir con una NAND cuyas entradas están unidas.



b) AND

La puerta AND es una NAND invertida (Invertir la puerta lo realizaremos con una NAND cuyas entrada están unidas).



c) OR

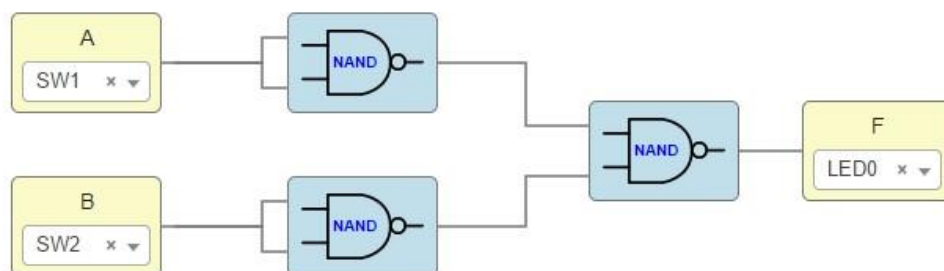
Aplicando las leyes de Morgan la ecuación nos queda de la siguiente manera:

1.-  $F = a + b$

2.-  $F = \overline{\overline{a + b}}$

3.-  $F = \overline{\overline{a} \cdot \overline{b}}$

El circuito queda de la siguiente manera:

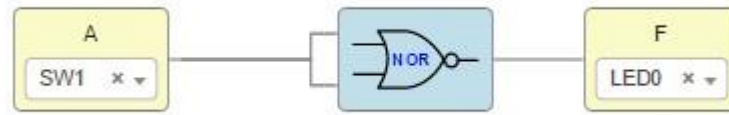


### Ejercicio 9

Conseguir las siguientes puertas equivalentes utilizando las puertas NOR.

a) NOT

La puerta NOT se puede conseguir con una NOR cuyas entradas están unidas.



b) AND

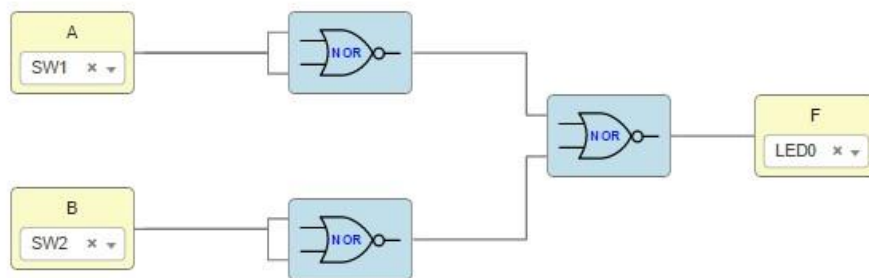
Aplicando las leyes de Morgan la ecuación nos queda de la siguiente manera:

$$1.- F = a \cdot b$$

$$2.- F = \overline{\overline{a \cdot b}}$$

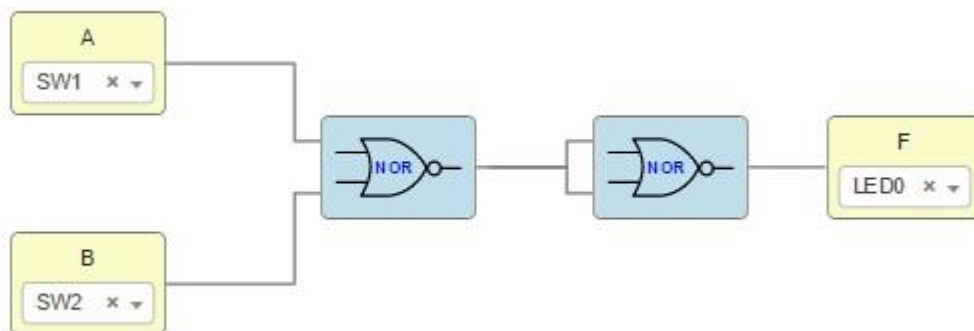
$$3.- F = \overline{\overline{a} + \overline{b}}$$

El circuito queda de la siguiente manera:



c) OR

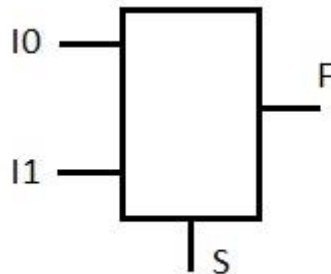
La puerta OR es una NOR invertida (Invertir la puerta lo realizaremos con una NOR cuyas entrada están unidas).



## Ejercicio 12

Diseñar un multiplexor de dos entradas.

El multiplexor es un elemento en el que sacamos a la salida el valor de la entrada correspondiente a la entrada que nos indica la entrada de selección.



Si la entrada de selección S es 0, la salida F tendrá el valor de la entrada I0 y si la entrada de selección S es 1, la salida F tendrá el valor de la entrada I1.

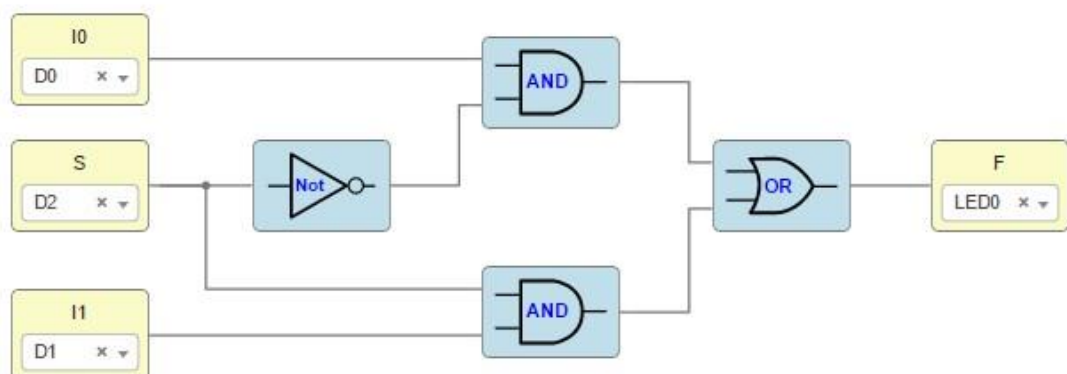
La tabla de la verdad nos quedara de la siguiente manera:

S	F
0	I0
1	I1

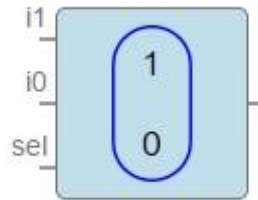
La ecuación nos queda de la siguiente manera

$$F = \bar{S} \cdot I0 + S \cdot I1$$

Teniendo en cuenta la función, la programación de la FPGA mediante la colección con puertas lógicas nos queda de la siguiente manera:



El multiplexor con 2 entradas y 1 entrada de selección también se puede hacer con un elemento de la colección. Este elemento está en el apartado de combinacionales en el apartado de multiplex.



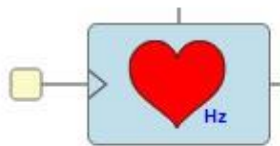
Con los multiplexores, podemos separar datos, resolver funciones....

### Ejercicio 13 (lcestudio)

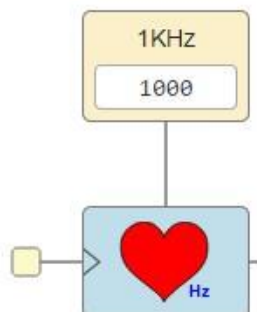
Encender un led con dos frecuencias (1KHz y 2KHz) que alterne a una frecuencia de 2Hz.

En este caso utilizaremos un bloque que está en nuestra colección que nos permite introducir la frecuencia que queremos que salga.

Este bloque está en el apartado secuenciales/multivib/astables/ y se denomina corazón\_Hz. Este bloque es el que a continuación se muestra.



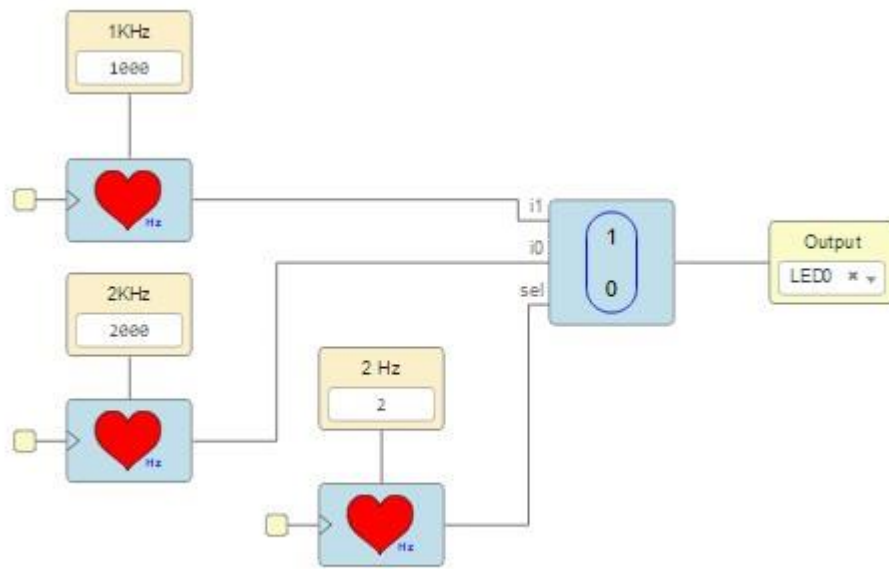
A este bloque hay que añadir una constante. Esta constante se añade desde nuestra colección y el apartado básico/constante. Conectaremos la constante al bloque corazón\_hz e introduciremos el número de hz que queramos que nos saque, quedando el bloque de la siguiente manera:



En este ejemplo sacaremos pulsos de una frecuencia de 1Khz.

En el ejercicio que nos plantean, utilizaremos un multiplexor que nos permitirá sacar dos frecuencias distintas a una frecuencia de 2 Hz. Para ello introduciremos en cada una de las entradas del multiplexor las distintas frecuencias y en la entrada de selección introduciremos una señal de 2Hz que nos permitirá sacar la señal de 1Khz mientras que la señal sea "0" y sacaremos la señal de 2KHz mientras la señal este a "1".

El circuito nos quedara de la siguiente manera:

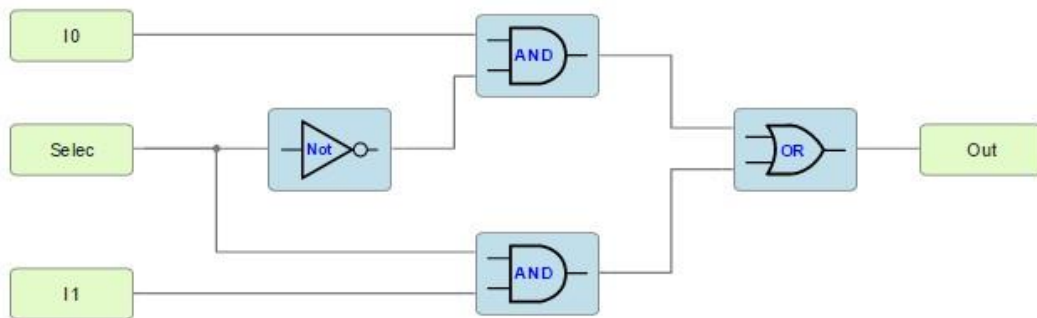


## Creación de componentes

Una de las posibilidades que nos ofrece ICStudio es que nos permitirá crear nuevos bloques a partir de un circuito que nosotros creamos.

Vamos a crear el bloque del multiplexor de dos entradas y una salida como ejemplo.

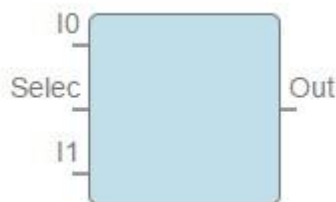
Para ello crearemos el circuito del multiplexor con puertas lógicas. Las entradas y las salidas las seleccionaremos como pines genéricos, es decir, no seleccionaremos como pines de la FPGA. El circuito nos queda de la siguiente manera:



Editaremos la información del bloque (Editar/Información del proyecto) donde le daremos el nombre, la descripción y podremos añadir la imagen del bloque (esta imagen tiene que tener formato SVG).

Guardaremos este circuito como cualquier otro circuito.

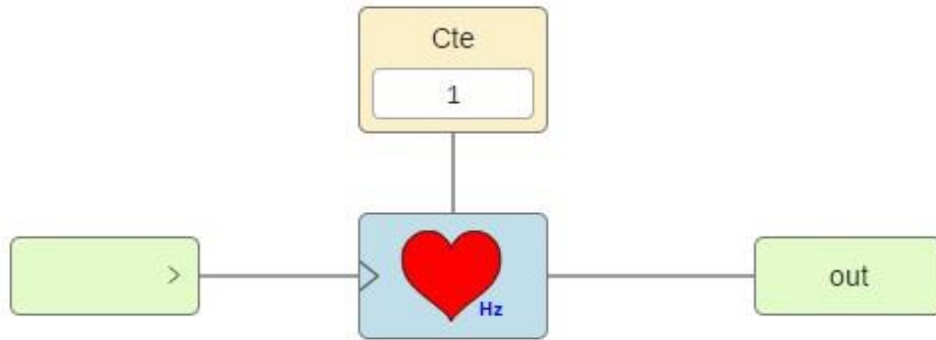
Una vez guardado si queremos utilizarlo tendremos que ir a archivo y seleccionar añadir como bloque, donde seleccionaremos el circuito guardado. A partir de aquí podemos ya usar nuestro bloque que en nuestro caso es el de multiplexor de 2 a 1, quedando el circuito de la siguiente manera:



En el caso de que uno de los pines sea de reloj seleccionaremos la entrada y le diremos que es un pin de reloj. En el caso de poner una entrada parametrizada pondremos una constante, si esta entrada parametrizada la seleccionamos como parámetro local esa constante quedara como dentro del bloque y no será accesible. Para que sea accesible no tendrá que estar seleccionada la opción parámetro local.

Como ejemplo realizaremos un bloque que nos dé una frecuencia de un número determinado de hz.

El circuito quedara de la siguiente manera:



Este circuito como bloque quedara de la siguiente manera:

