Práctica 06
Compuertas Lógicas Usando VHDL

ALUMNO: MEZA VARGAS BRANDON DAVID
GRUPO: 2CM5
BOLETA: 2020630288

1) Objetivo general

El alumno realizará las compuertas lógicas AND, OR, NAND, NOR, XOR y XNOR programando en el lenguaje VHDL y programará su GAL 22V10 para verificar el resultado.

2) Introducción Teórica

VHDL

VHDL es un lenguaje de alto nivel que describe todas las características de circuitos electrónicos digitales de variada complejidad. El significado de las siglas VHDL es V de VHSIC (Circuitos Integrados de muy alta Velocidad – Very High Speed Integrated Circuits) y HDL de (Lenguaje de Descripción del Hardware – Hardware Description Language).

- El VHDL nace por iniciativa del Departamento de Defensa de EE.UU. a comienzos de los 80.
- En julio de 1983 tres compañías (Texas Instruments, IBM e Internetics) reciben el encargo de desarrollarlo.
- La primera versión fue publicada en agosto de 1985.
- En 1986 se convierte en un estándar del IEEE (IEEE 1076-1987).
- Posteriormente, con el estándar IEEE 1164 se le añaden nuevas características.

Modelación del hardware

Los comentarios que se hagan a las instrucciones en el programa deben comenzar con 2 guiones así: --COMENTARIOS

Está compuesto por las LIBRERÍAS, la ENTIDAD y de la ARQUITECTURA.

LIBRERÍAS (LIBRARY): la librería estándar es siempre visible, si se requieren otras librerías se deben importar.

ENTIDAD (Entity): primero se define un bloque dándole un nombre y luego se específica la interfaz del circuito declarando las líneas de entrada y de salida (llamadas puertos).

ARQUITECTURA (Architecture): es la descripción interna del circuito, describe lo que el módulo hace, o sea, la forma en que las salidas se relacionan con las entradas. Puede describirse de diferentes maneras (deferentes formas de modelar).

Operadores Lógicos

Operadores Lógicos	not	Not A
	and	(A and B)
	or	(A or B)
	xor	(A xor B)
	nand	(A nand B) o not (A and B)
	nor	(A nor B) o not (A or B)
	xnor	(A xnor B) o not (A xor B)

Ejemplos:

X <= not A and B; -- A B

Y <= not (A and B); -- (A B)

Z <= A nand B; -- (A B)

Operadores de asignación

Operador	Descripción
<=	Asignar valores a una señal X <= '1'; -- Asignar "1" a la **señal** X
:=	Asignar valores a una variable y/o constante. Establece valores iniciales. Y := "0010"; -- Asignar "0010" a la **variable** Y
=>	Asignar valores a los elementos de un vector individual. Z <= "1000"; -- Asignar "1" al LSB y a los demás "0" si la salida Z fue declarada como to. Asignar "1" al MSB y a los demás "0" si la salida Z fue declarada como downto. W <= (3 => '1', others => '0'); -- Asignar "1" al LSB y a los otros "0" si la salida Z fue declarada como to. Asignar "1" al MSB y a los otros "0" si la salida Z fue declarada como downto.

Librerías – library

Al declarar una librería, son necesarias dos líneas de código: una que contenga el nombre de la librería y otra con la sentencia USE, así:

library *Nombre de la librería;*

use *Nombre de la librería. Nombre del paquete. Parte del paquete*

```
library IEEE;                    ①
use IEEE.STD_LOGIC_1164.ALL;     ②
use IEEE.STD_LOGIC_ARITH.ALL;    ③
use IEEE.STD_LOGIC_UNSIGNED.ALL; ④
```

1. Nombre de la librería
2. Librería IEEE. Paquete STD_LOGIC_1164. Todo el paquete;
3. Librería IEEE. Paquete STD_LOGIC_ARITH.ALL. Todo el paquete;
4. Librería IEEE. Paquete STD_LOGIC_UNSIGNED. Todo el paquete;

Entidad – entity

Se identifica al módulo como un bloque donde se definen los puertos de entrada y salida

La entidad define el NOMBRE de los puertos, el MODO (entradas o salidas) y el TIPO (integer, bit, vector,…).

entity Nombre_entidad is

Port (NOMBRE de la señal: MODO y TIPO de dato;

NOMBRE de la señal: MODO y TIPO de dato);

end Nombre_entidad;

Puertos

Son los puntos de conexión entre una entidad y el medio exterior, se representan por entradas y salidas y requiere de un sentido ya sea como entrada, salida o bidireccional.

- Los identificadores son los nombres válidos para referir variables, constantes, señales, procesos, etc.
- No tienen longitud máxima.
- Puede contener caracteres de la „A" a la „Z", de la „a" a la „z", caracteres numéricos de „0" al „9" y el caractér subrayado "_".
- No se diferencia entre mayúsculas y minúsculas (CONTADOR, contador y ConTadoR son el mismo identificador.)
- Debe empezar por un carácter alfabético, no puede terminar con un subrayado, ni puede tener dos subrayados seguidos.
- No puede usarse como identificador una palabra reservada.

Modo	Descripción
IN	Las señales solo entran en la entidad y no salen. La señal puede ser leída pero no escrita
OUT	Las señales salen de la entidad y no pueden ser leídas dentro de ella
BUFFER	Este modo se utiliza para las señales que además de salir de la entidad pueden usarse como entradas realimentadas
INOUT	Este modo se utiliza para señales bidireccionales. Se emplea en salida con tres estados (tri-state)

Arquitectura – architecture

Es la estructura que define el funcionamiento de una entidad. Indica el tipo de procesamiento que se realiza con las señales de entradas declaradas en la entidad.

Estilos de modelación:

- Estilo de Flujo de Datos = con ecuaciones boolenas y con las instrucciones WHEN (cuando) – ELSE (el resto).
- Estilo Estructural = en el modelado estructural se declaran previamente los dispositivos que componen el circuito y se realizan las interconexiones.
- Estilo Comportamental o Funcional = expone la forma en que trabaja el sistema, relación que hay entre las entradas y salidas del circuito, sin importar como esté organizado en su interior (caja negra), este modelado se basa en el uso de procesos y declaraciones secuenciales.

3) Materiales empleados

- ✓ 1 Circuito Integrado GAL22V10
- ✓ 05 LEDS de colores
- ✓ 05 Resistores de 330Ω
- ✓ 05 Resistores de 1KΩ
- ✓ 1 Dip switch de 8
- ✓ Alambre telefónico
- ✓ 1 Tablilla de Prueba (Protoboard)
- ✓ 1 Pinzas de punta
- ✓ 1 Pinzas de corte
- ✓ Cables Banana-Caimán (para alimentar el circuito)

4) Equipo empleado

- ✓ Multímetro
- ✓ Fuente de Alimentación de 5 Volts
- ✓ Manual de MOTOROLA, “FAST and LS TTL”
- ✓ Programador Universal

5) Desarrollo Experimental y Actividades

1.- Implementar las siguientes compuertas lógicas usando VHDL.

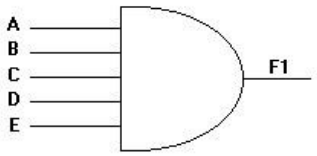
2.- Llene su tabla de verdad para todas las compuertas.

3.- Coloque el código final de su programa junto con el archivo de asignación de pines RPT.

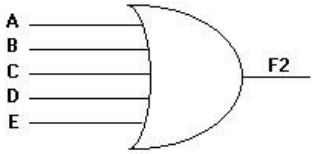
```
 library ieee;
use ieee.std_logic_1164.all;
entity compuertas is
      port(A,B,C,D,E: in std_logic;
            F1,F2,F3,F4,F5,F6: out std_logic);
end compuertas;
architecture booleana of compuertas is
begin
      F1 <= A and B and C and D and E;
      F2 <= A or B or C or D or E;
      F3 <= A xor B xor C xor D xor E;
```

```
    F4 <= not F1;
    F5 <= not F2;
    F6 <= not F2;
end booleana;
```

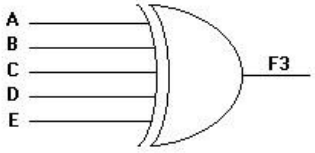
4.- Arme su circuito y programe la GAL para verificar sus tablas de verdad.



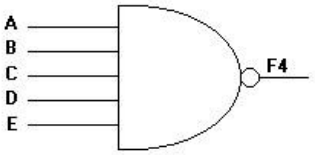
Compuerta AND



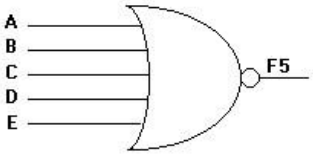
Compuerta OR



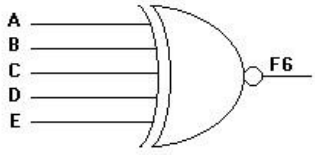
Compuerta XOR



Compuerta NAND



Compuerta NOR



Compuerta NEOR

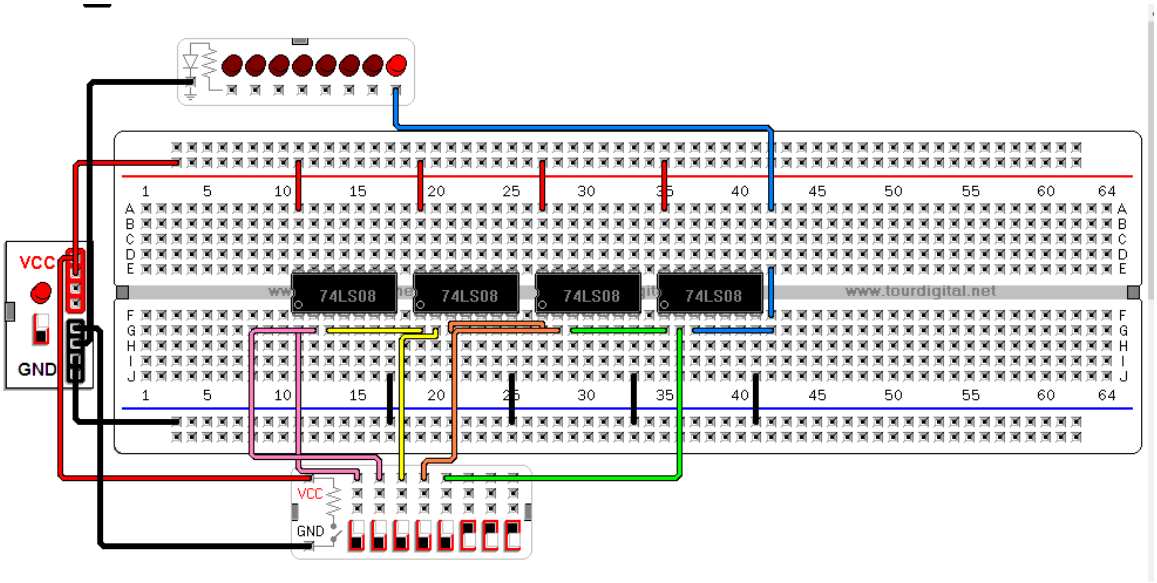
Tabla de Verdad

#	A	B	C	D	E	F1 AND	F2 OR	F3 XOR	F4 NAND	F5 NOR	F6 NEOR
0	0	0	0	0	0	0	0	0	1	1	1
1	0	0	0	0	1	0	1	1	1	0	0
2	0	0	0	1	0	0	1	1	1	0	0
3	0	0	0	1	1	0	1	0	1	0	1
4	0	0	1	0	0	0	1	1	1	0	0
5	0	0	1	0	1	0	1	0	1	0	1
6	0	0	1	1	0	0	1	0	1	0	1
7	0	0	1	1	1	0	1	1	1	0	0
8	0	1	0	0	0	0	1	1	1	0	0
9	0	1	0	0	1	0	1	0	1	0	1
10	0	1	0	1	0	0	1	0	1	0	1
11	0	1	0	1	1	0	1	1	1	0	0
12	0	1	1	0	0	0	1	0	1	0	1
13	0	1	1	0	1	0	1	1	1	0	0
14	0	1	1	1	0	0	1	1	1	0	0
15	0	1	1	1	1	0	1	0	1	0	1
16	1	0	0	0	0	0	1	1	1	0	0

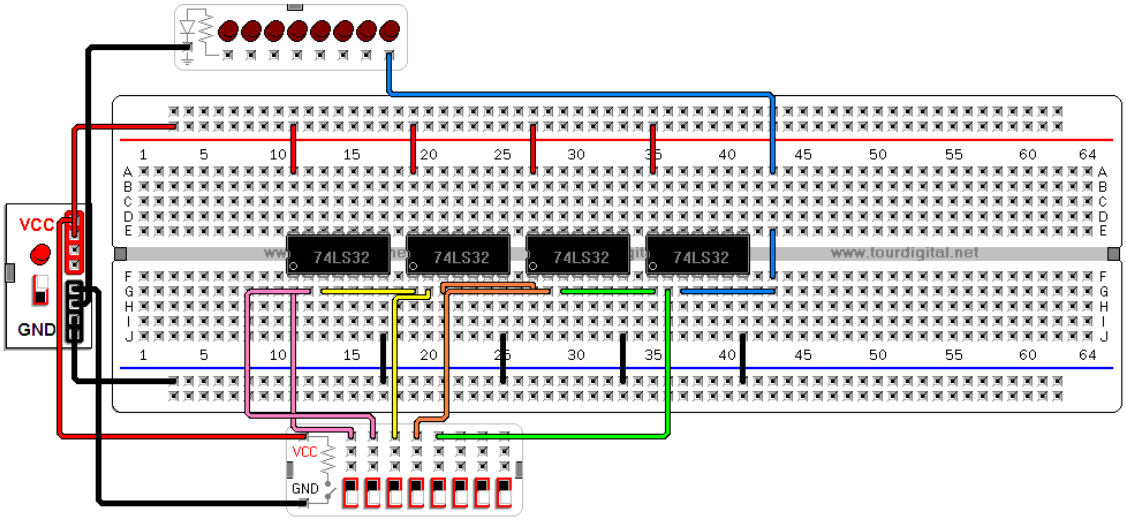
17	1	0	0	0	1	0	1	0	1	0	1
18	1	0	0	1	0	0	1	0	1	0	1
19	1	0	0	1	1	0	1	1	1	0	0
20	1	0	1	0	0	0	1	0	1	0	1
21	1	0	1	0	1	0	1	1	1	0	0
22	1	0	1	1	0	0	1	1	1	0	0
23	1	0	1	1	1	0	1	0	1	0	1
24	1	1	0	0	0	0	1	0	1	0	1
25	1	1	0	0	1	0	1	1	1	0	0
26	1	1	0	1	0	0	1	1	1	0	0
27	1	1	0	1	1	0	1	0	1	0	1
28	1	1	1	0	0	0	1	1	1	0	0
29	1	1	1	0	1	0	1	0	1	0	1
30	1	1	1	1	0	0	1	0	1	0	1
31	1	1	1	1	1	1	1	1	0	0	0

Protos armados:

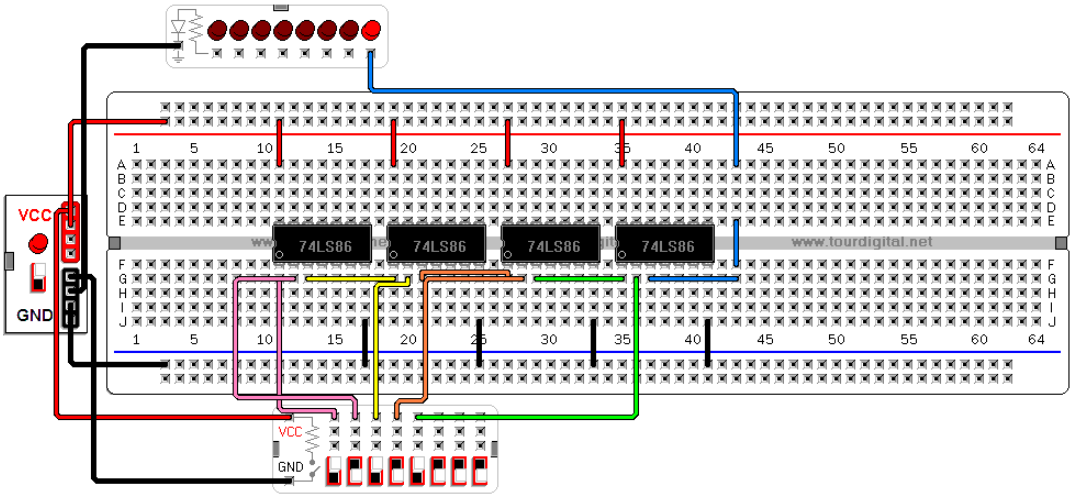
COMPUERTA AND:



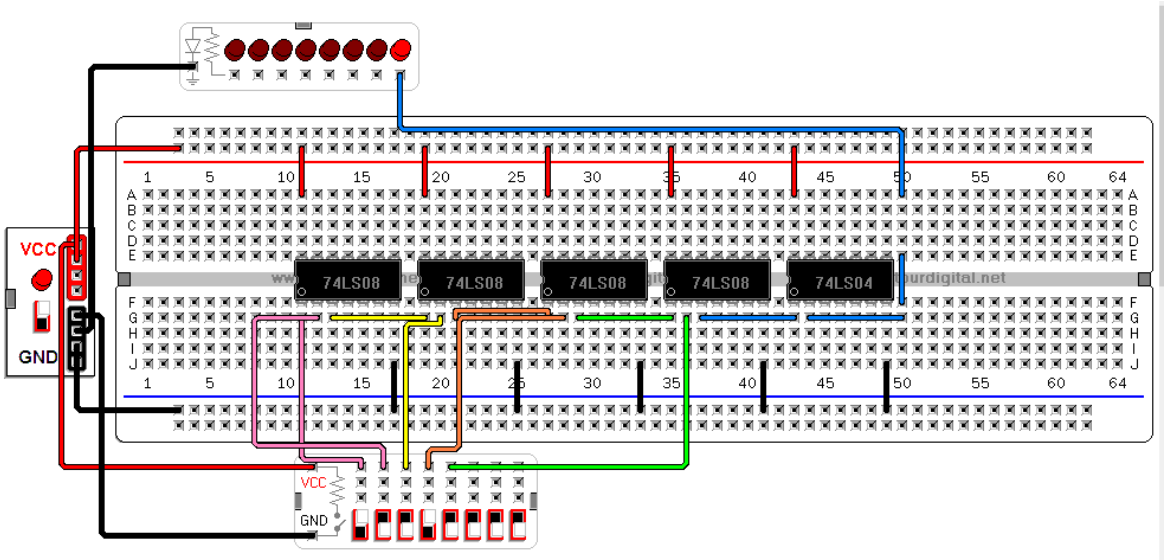
COMPUERTA OR:



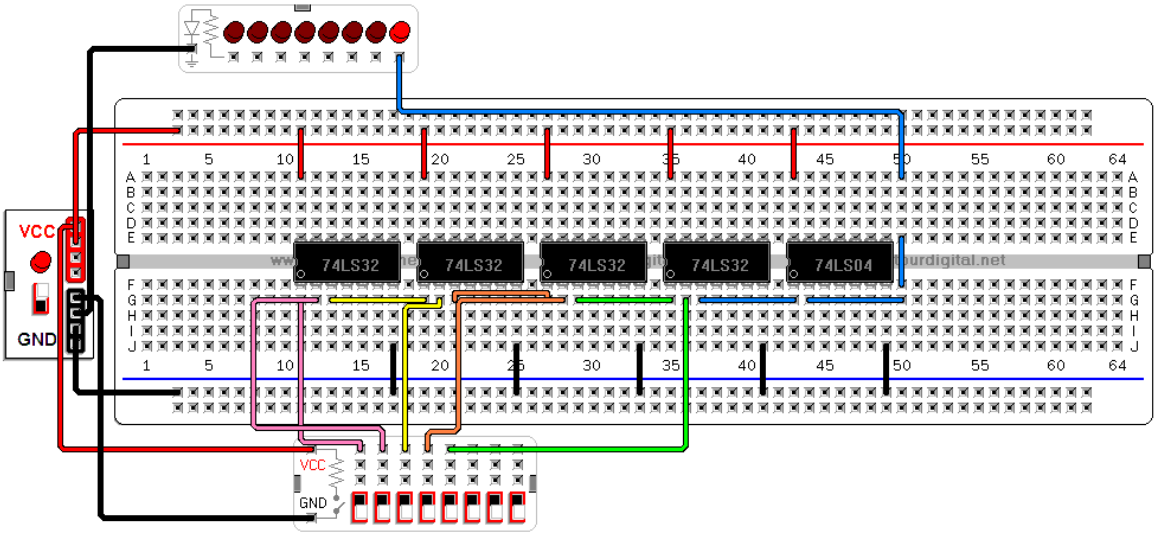
COMPUERTA XOR:



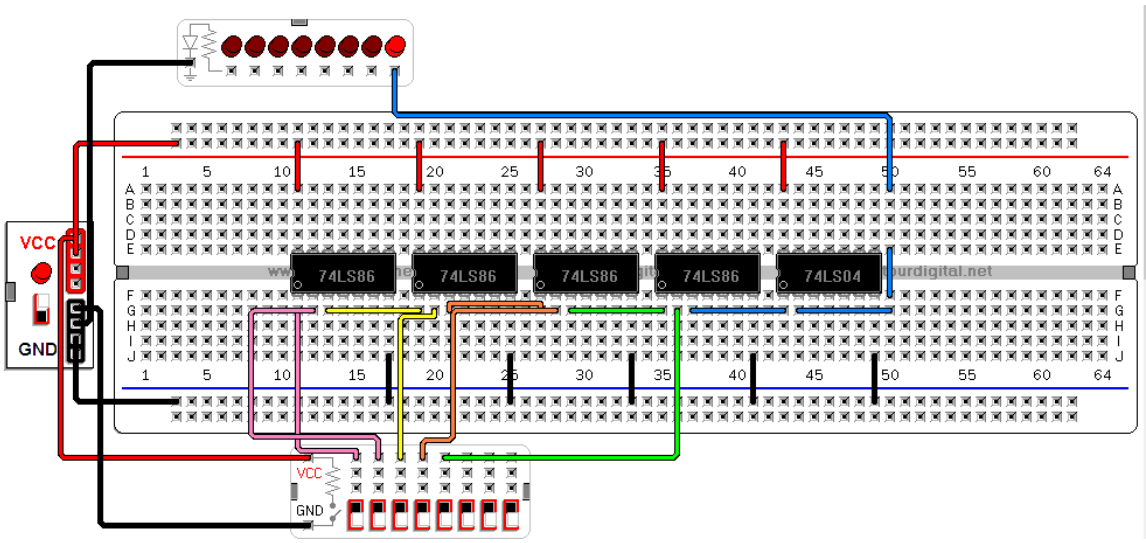
COMPUERTA NAND:



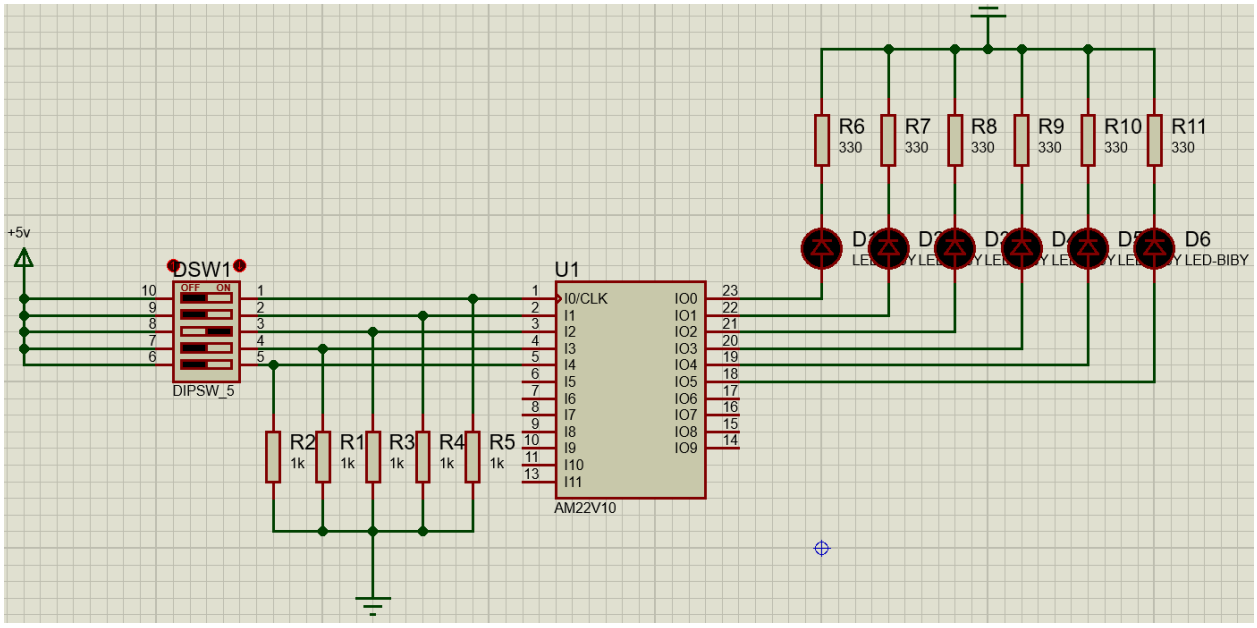
COMPUERTA NOR:



COMPUERTA NEOR:



El siguiente es el circuito lógico, pero al no tener el archivo *.jed no hará nada nuestro circuito



6) Conclusiones Individuales.

Esta práctica fue un poco más corta, pues nos limitamos a solo llenar el formato de esta, y a realizar los circuitos con compuertas lógicas de 5 entradas en la protoboard gracias al simulador de circuitos, a pesar de eso, gracias a los ejercicios que se resolvieron con ayuda del video que realizo el profesor, puedo concluir que gracias al lenguaje VHDL describimos circuitos electrónicos, en este caso fue con compuertas, para posteriormente poder implementarlo en un sistema real, es decir, en un PLD.

De igual forma

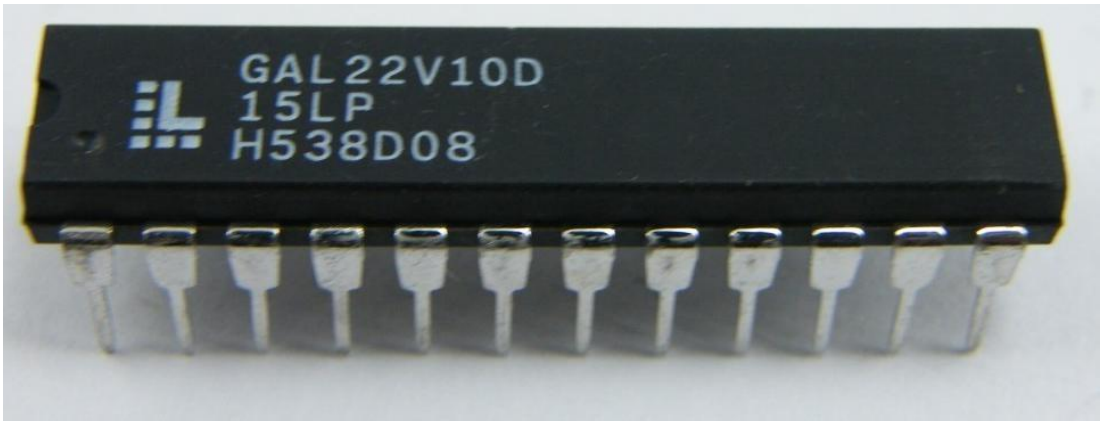
Definitivamente me hubiera gustado realizar los ejercicios de forma práctica, sin embargo, trabajar como lo hemos estado haciendo no evita que me lleve el conocimiento.

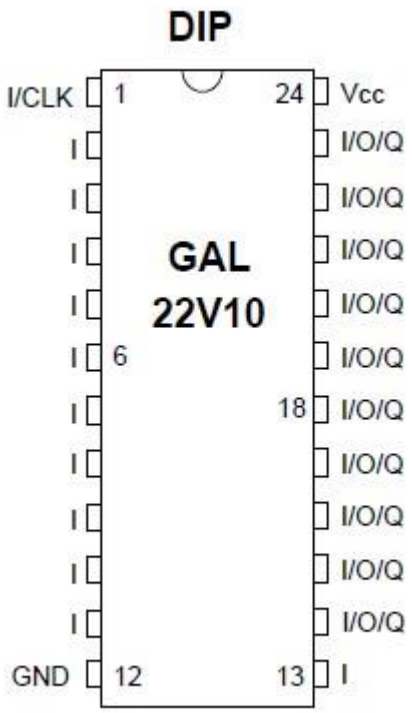
7) Bibliografía.

-Boltero, H. (2018). “VHDL y FPGA”. Obtenido de: http://files.oscarbotero.webnode.com.co/200000823-ccb6acdb09/VHDL_FPGA1.pdf

-Juarez, J. (2020). “VHDL: los ladrillos básicos; and, or, not, nand, nor, xor y xnor”. Obtenido de: https://jnjsite.com/vhdl-los-ladrillos-basicos-and-or-not-nand-nor-xor-y-xnor/

8) ANEXOS.


GAL22V10D
15LP
H538D08


DIP
I/CLK
1
24
Vcc
I
I/O/Q
I
I/O/Q
I
I/O/Q
I
I/O/Q
I
6
I/O/Q
I
18
I/O/Q
I
I/O/Q
I
I/O/Q
I
I/O/Q
I
I/O/Q
GND
12
13
I
GAL
22V10