Practica 4

Uso galaxy

Profesor: Barron Vera Jose Emanuel
Materia: Fundamentos de diseño digital
Grupo: 3CV6
Alumno: Cazares Cruz Jeremy Sajid
Boleta: 2021630179

Para la realización de la práctica se utiliza el software de Galaxy el cual nos ayuda a reafirmar lo visto en clase acerca de los circuitos integrados PLD así como la forma de poder programarlos para una función específica, siendo así que esta primera practica con estos circuitos integrados es para conocer la programación de los mismos conforme a las compuertas básicas AND, NAND, OR, NOR, XOR y XNOR, siendo de esta manera que debe dar salidas iguales a las ya antes mencionadas compuertas, con la diferencia de que se utilizar 3 compuertas por el uso de 4 variables esto en un circuito integrado GAL22V100 mediante la programación en VHDL

La compuerta se define mediante “entity” seguido de su nombre se introduce lo que serán los pines de entrada y se sabe que son de entrada por el comando “in std_logic” de tal manera que los pines o variables que se escriban antes del comando serán tomados como entradas, se puede saber esto por el comando in de igual manera si se escribiera un comando “out std_logic” se tomara como salida los pines o variables que se encuentren.

Para definir la función lógica que manejará el circuito integrado en este caso la gal se usa “architecture x of” donde a continuación se escribe el nombre dado de la función, tal como se puede ver en el código de la figura siguiente para dar a conocer la función simplemente damos como valor a asignar a la salida antes programada y lo que será con las variables de entrada, como en esta primera asignación donde se hará un repaso de las compuertas lógicas de 4 entradas mediante el uso de la gal

Un detalle importante es que se debe terminar la función con el mismo nombre con el cual se inicia, como en este caso que inicia con el nombre “función” debe terminar con un end función

```
1  LIBRARY ieee;
2  USE ieee.std_logic_1164.all;
3
4  ENTITY compuerta_and IS
5      PORT(a, b, c, d : in std_logic;
6           salida : out std_logic
7           );
8  end compuerta_and;
9
10 architecture funcion of compuerta_and IS
11 begin
12
13     salida <= a and b and c and d;
14
15     end funcion;
```

figura 1 función AND

Al igual que C y C++ cada línea de código debe tener un punto y coma (;) para poder realizarse la compilación

Una vez terminado el código vhdl el archivo deberá unirse al proyecto principal para posteriormente compilar y ejecutar una simulación

Para la simulación del archivo se usa el programa active hdl el cual se abre mediante el menú de herramientas en el software Galaxy, este software de simulación que trabaja en conjunto con Galaxy simula la función de la gal

mediante trenes de pulsos los cuales se deben estar estimulando para poder ver una reacción en la salida, un ejemplo de esto es el de la compuerta and, al estar todas las variables unidas por una and (a and b and c and d) el resultado de la función será 1 cuando todas las entradas sean 1, esto en el simulador se vería como que todas las ondas o trenes de pulso estén en 1 lógico siendo como que todas las ondas estén “arriba”

Un caso particular dentro de este lenguaje de programación es al utilizar las compuertas NAND, en particular para esta primera práctica, ya que el programa para estas 4 entradas hace las operaciones lógicas por parejas de tal manera que al hacer una operación NAND a A y B tendrá como resultado la negación de A and B pero al hacer esto con negación de C and D al negarse una vez más y hacer uso de la operación lógica AND el resultado sería el mismo al de la operación AND, por lo tanto en esos casos donde la función es la negación de otra como es el caso de AND con NAND y OR con NOR se hará la función de la compuerta y al final se negara para tener el resultado teórico verdadero

A continuación, se mostrarán capturas de los códigos usados para las otras compuertas:

```
1  LIBRARY ieee;
2  USE ieee.std_logic_1164.all;
3
4  ENTITY compuerta_nand IS
5      PORT(a, b, c, d : in std_logic;
6          salida : out std_logic
7          );
8  end compuerta_nand;
9
10 architecture funcion of compuerta_nand IS
11 begin
12
13     salida <= not(a and b and c and d);
14
15
16     end funcion;
```

figura 2 función NAND

```
1  LIBRARY ieee;
2  USE ieee.std_logic_1164.all;
3
4  ENTITY compuerta_or IS
5      PORT(a, b, c, d : in std_logic;
6          salida : out std_logic
7          );
8  end compuerta_or;
9
10 architecture funcion of compuerta_or IS
11 begin
12
13     salida <= a or b or c or d;
14
15     end funcion;
```

figura 3 función OR

```
1  LIBRARY ieee;
2  USE ieee.std_logic_1164.all;
3
4  ENTITY compuerta_nor IS
5      PORT(a, b, c, d : in std_logic;
6          salida : out std_logic
7          );
8  end compuerta_nor;
9
10 architecture funcion of compuerta_nor IS
11 begin
12
13     salida <= not (a or b or c or d);
14
15     end funcion;
```

figura 4 función NOR

```
1  LIBRARY ieee;
2  USE ieee.std_logic_1164.all;
3
4  ENTITY compuerta_xor IS
5      PORT(a, b, c, d : in std_logic;
6          salida : out std_logic
7          );
8  end compuerta_xor;
9
10 architecture funcion of compuerta_xor IS
11 begin
12
13     salida <= a xor b xor c xor d;
14
15     end funcion;
```

figura 5 función XOR

```
1  LIBRARY ieee;
2  USE ieee.std_logic_1164.all;
3
4  ENTITY compuerta_xnor IS
5      PORT(a, b, c, d : in std_logic;
6          salida : out std_logic
7          );
8  end compuerta_xnor;
9
10 architecture funcion of compuerta_xnor IS
11 begin
12
13     salida <= not(a xor b xor c xor d);
14
15     end funcion;
```

figura 6 función XNOR

Para la simulación de las compuertas se tiene lo siguiente:

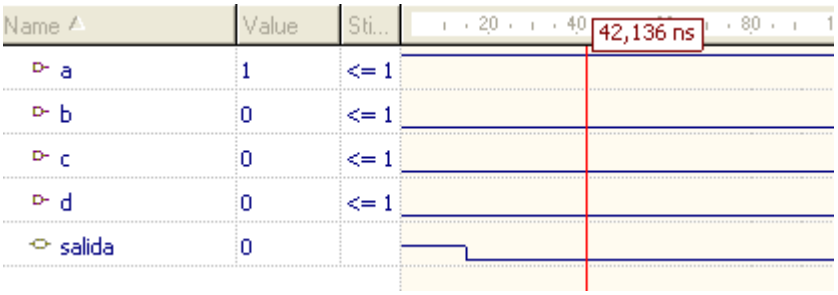

figura 7 AND 1

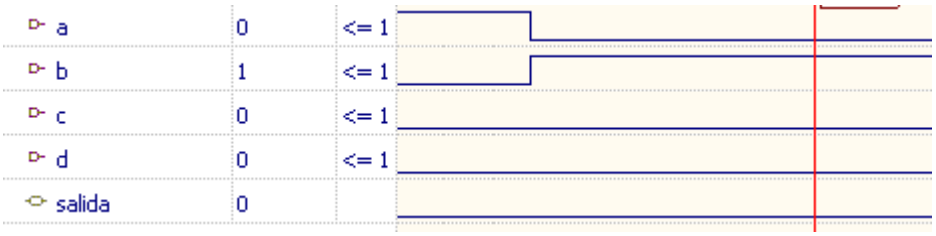

figura 8 AND 2

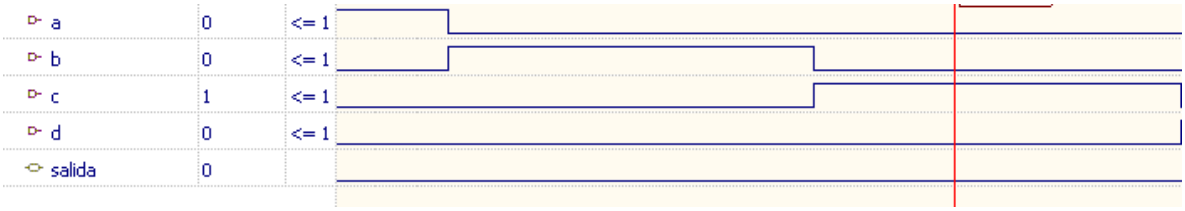

figura 9 AND 3

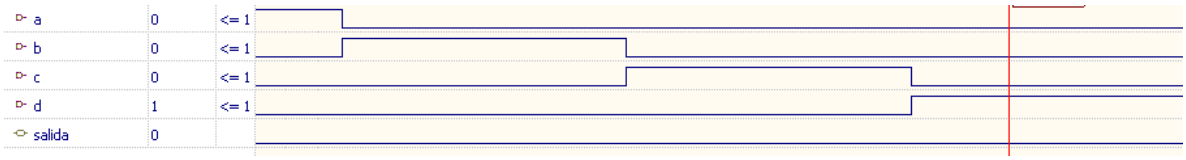

figura 11 AND 4

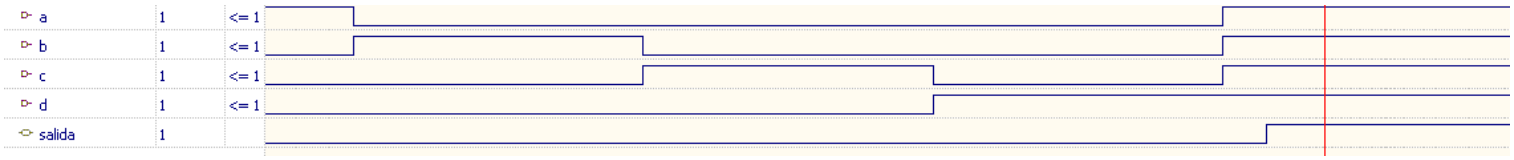

figura 10 AND 5

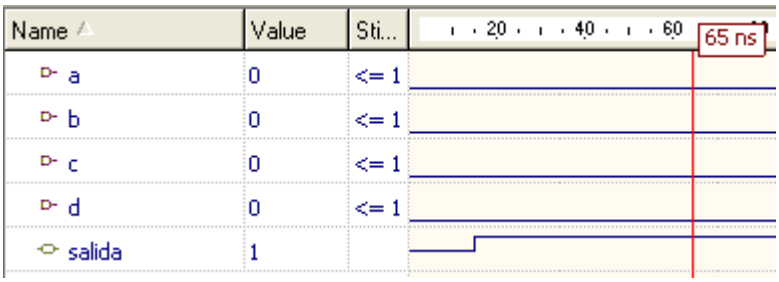

figura 12 NAND 1

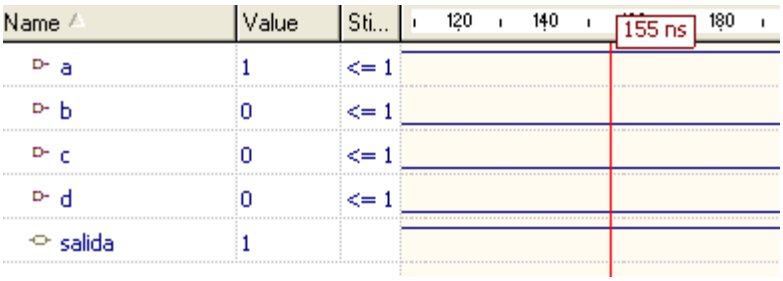

figura 14 NAND 2

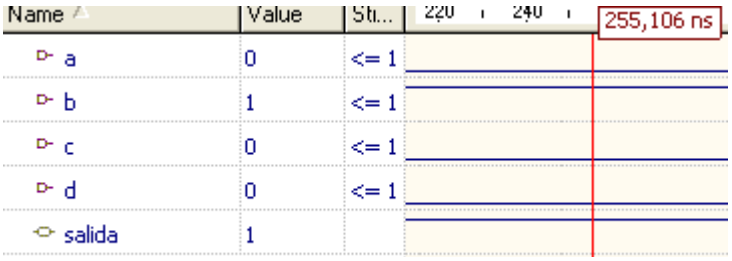

figura 13 NAND 3

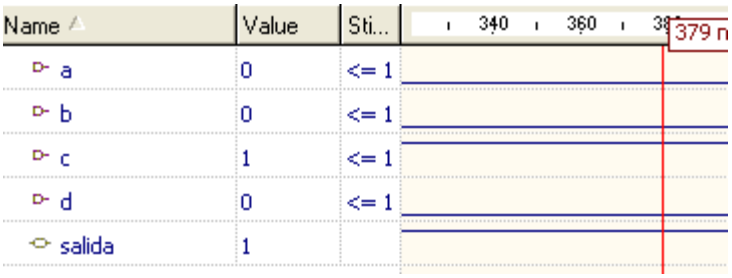

figura 15 NAND 4

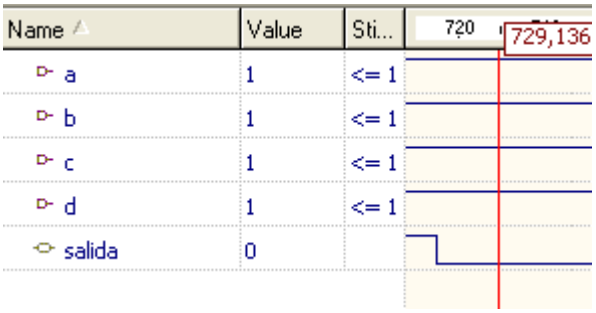

figura 16 NAND 5

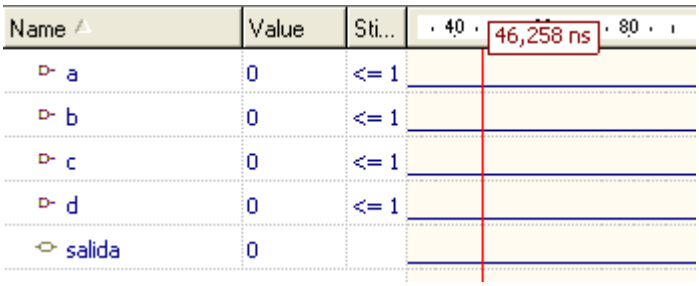

figura 17 OR 1

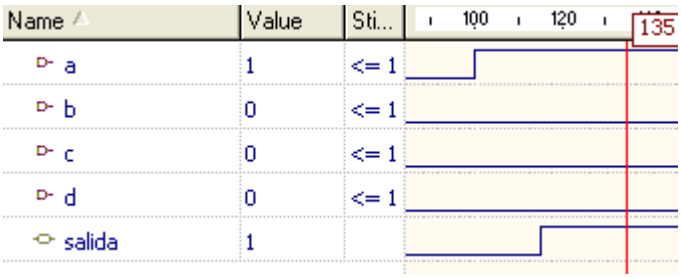

figura 18 OR 2

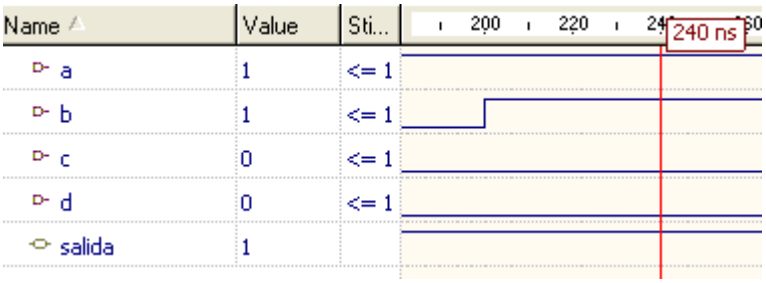

figura 19 OR 3

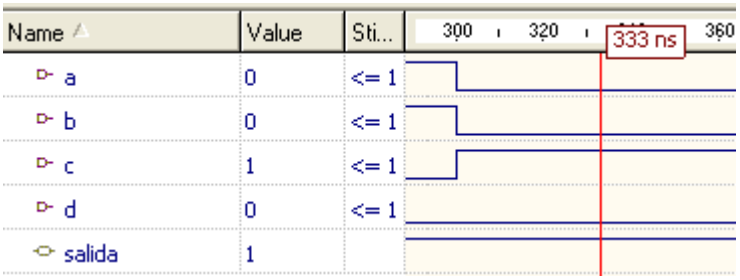

figura 20 OR 4

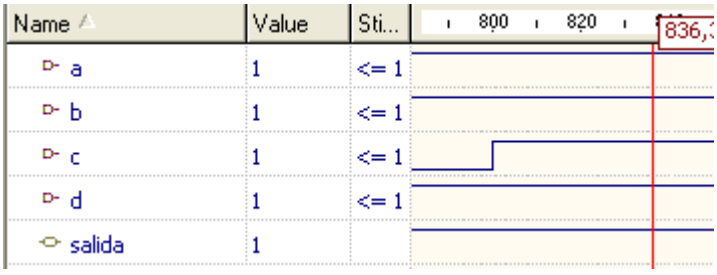

figura 21 OR 5

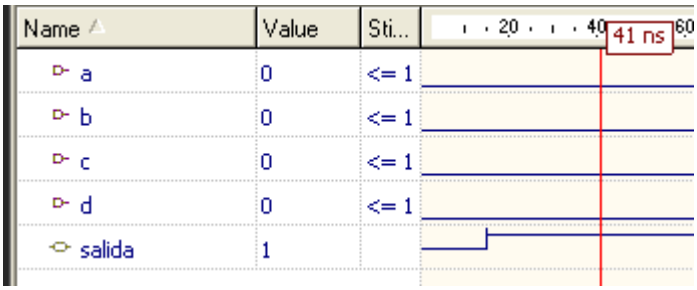

figura 22 NOR 1

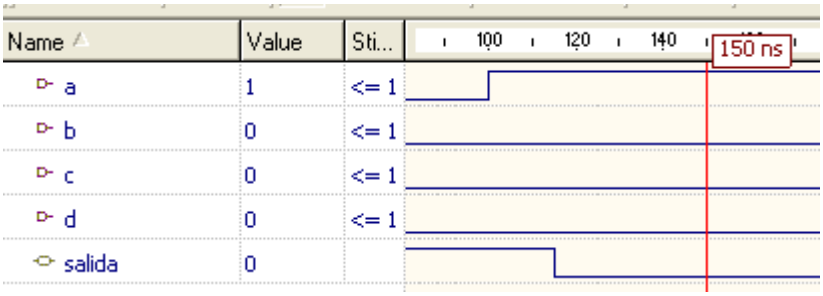

figura 23 NOR 2

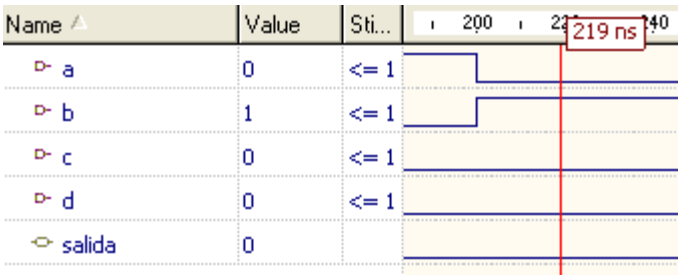

figura 24 NOR 3

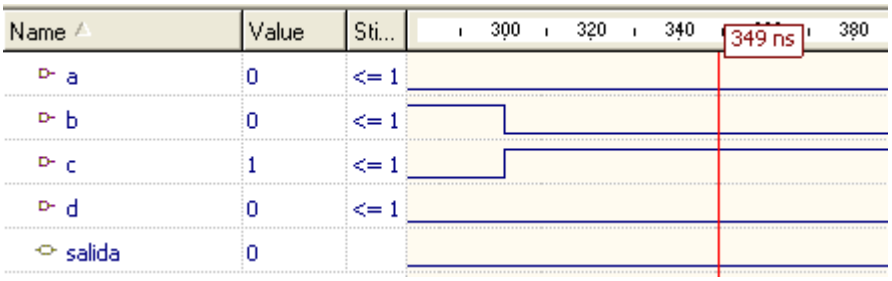

figura 25 NOR 4

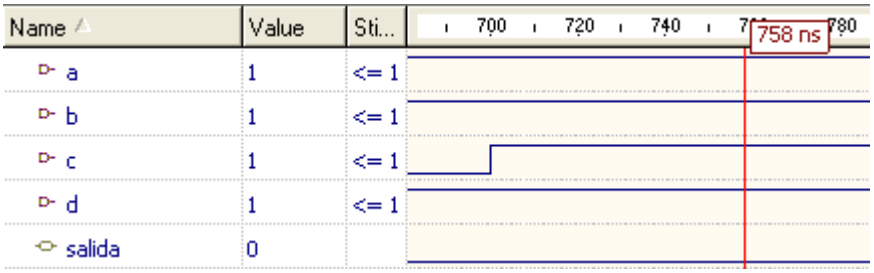

figura 26 NOR 5

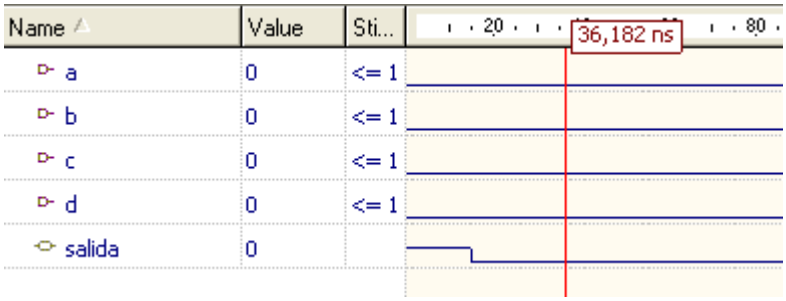

figura 27 XOR 1

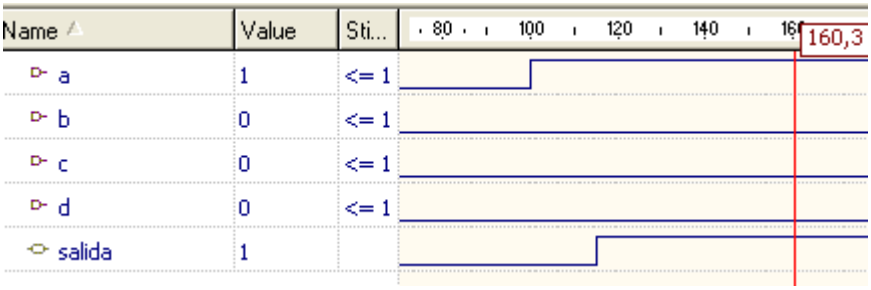

figura 28 XOR 2

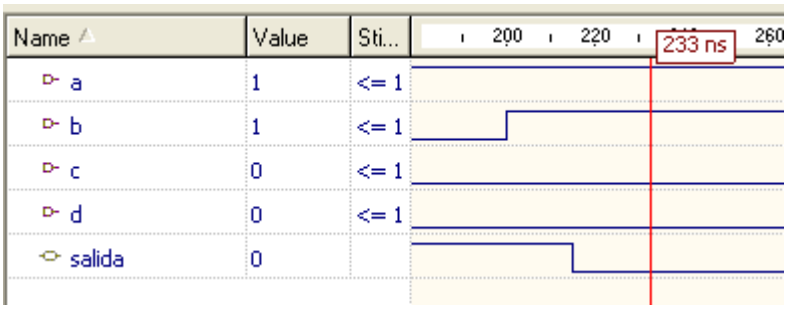

figura 29 XOR 3

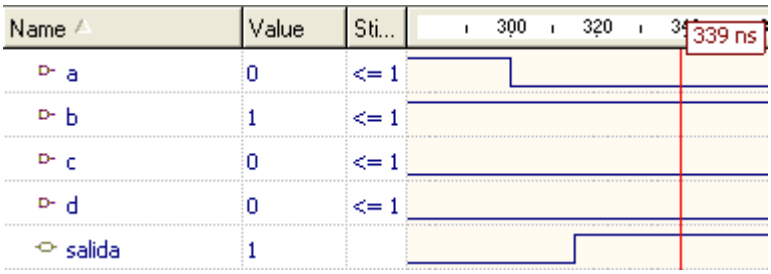

figura 30 XOR 4

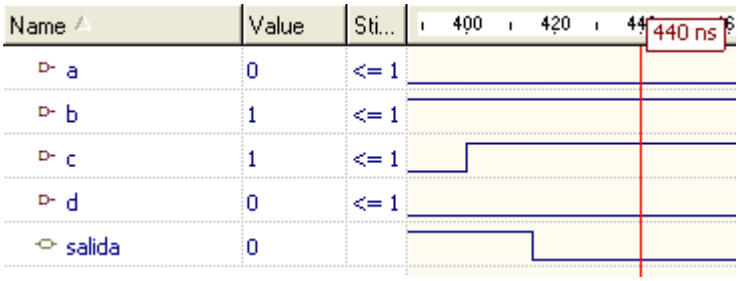

figura 31 XOR 5

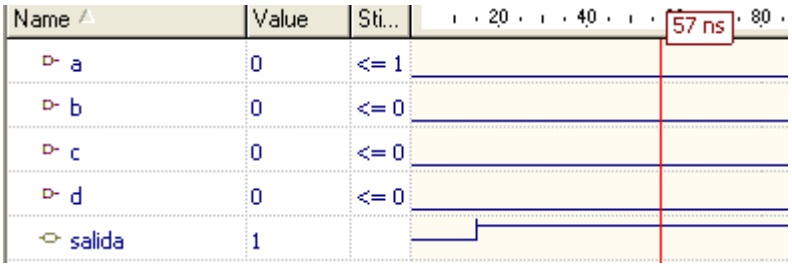

figura 32 XNOR 1

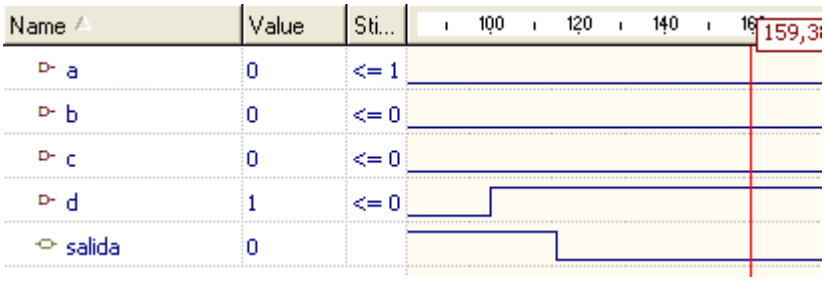

figura 33 XNOR 2

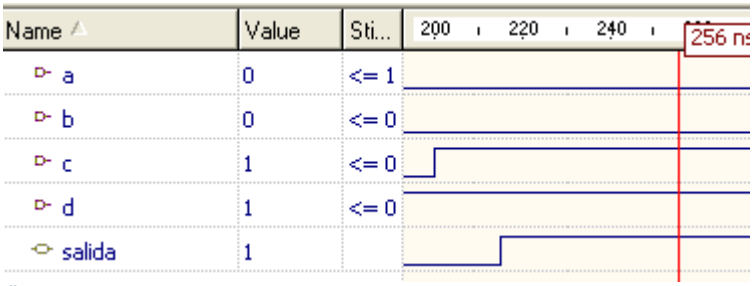

figura 34 XNOR 3

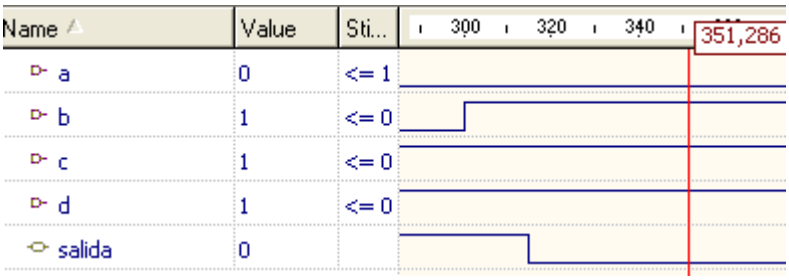

figura 35 XNOR 4

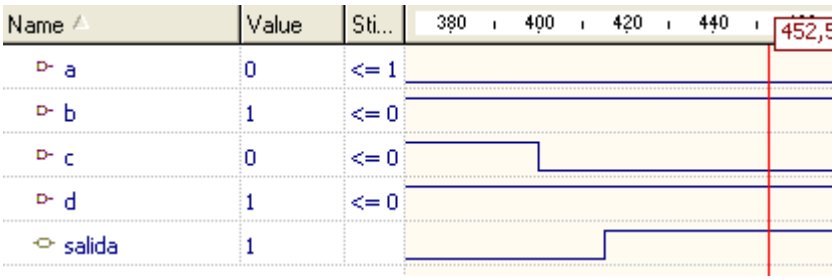

figura 36 XNOR 5

Tablas de verdad

USO DE GALAXY

Una vez realizado el programa, se procede a determinar la tabla de verdad para la compuerta.

Entradas				Salida compuerta
A	B	C	D	AND
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

figura 37 tabla de verdad AND

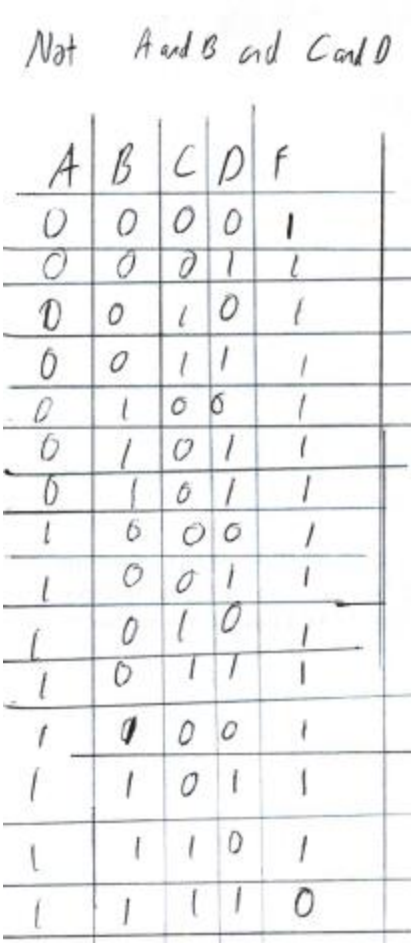

figura 38 tabla de verdad NAND

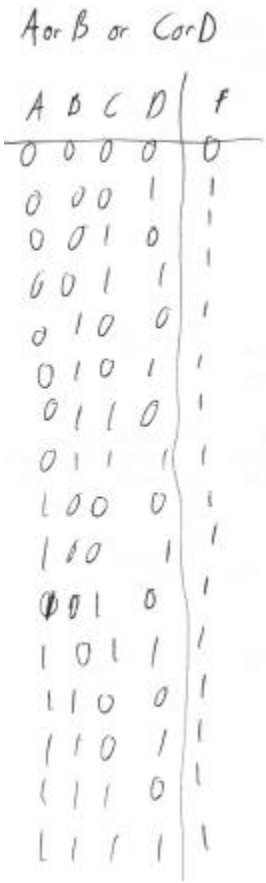

figura 39 tabla de verdad OR

figura 41 tabla de verdad NOR

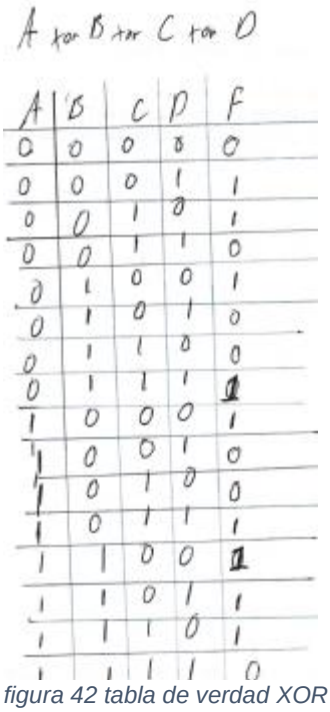

figura 42 tabla de verdad XOR

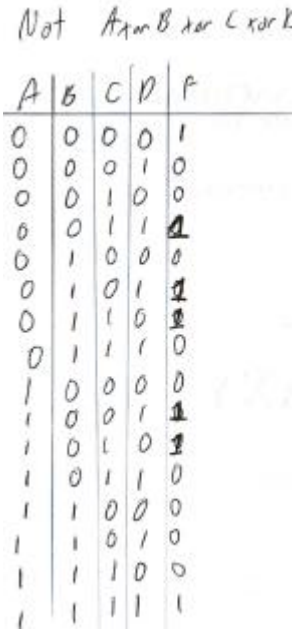

figura 40 tabla de verdad XNOR

Conclusiones

Por medio de la práctica se logró reafirmar lo antes visto en clase además de conocer la herramienta Galaxy para el uso, programación y simulación de plds siendo una herramienta útil al momento de diseñar circuitos en cuestión de expresiones booleanas