Instituto Politécnico Nacional

Escuela Superior de Cómputo

Práctica 2

Uso de galaxy

Materia: Fundamentos de Diseño Digital

Alumnos: Ávila López Víctor Iván
Ruiz Gonzales Carlos Emilio
Cazares Cruz Jeremy Sajid

Grupo: 3CM5

I OBJETIVO GENERAL:

Al finalizar la práctica, el alumno aprenderá a programar un dispositivo lógico programable (PLD). Para esto estudiará y aprenderá a utilizar las herramientas adecuadas para lograr dicho objetivo. Así mismo, reafirmará el conocimiento adquirido en clase al realizar un programa que ejecute una función específica, sobre el dispositivo utilizado, y comprobar físicamente el correcto funcionamiento del programa desarrollado. Finalmente, sabrá lo que es un PLD y como programar una función específica sobre el dispositivo, utilizando un lenguaje de descripción de hardware (HDL).

II OBJETIVOS ESPECÍFICOS:

I. Distinguir los beneficios proporcionados por las herramientas CAD.
II. Confirmar el conocimiento adquirido en clase y realizar un programa en un HDL que se ejecute sobre un PLD. III. Comprobar físicamente el correcto funcionamiento del sistema diseñado, el cual será implementando sobre un protoboard.

III MATERIAL Y EQUIPO EMPLEADO

- Mesa de instrumentación, del laboratorio de sistemas digitales
- Programador universal
- Fuente de 5V
- 1 GAL22V10
- 1 DIP switch de 4
- 3 Resistencias de 1KΩ
- 7 Resistencias de 220Ω
- 1 Display de Ánodo común

IV INTRODUCCIÓN TEÓRICA:

4.1 LAS HERRAMIENTAS CAD-EDA.

El término CAD hace referencia a sus siglas en inglés (Computer Aided Design o en español Diseño asistido por computadora) lo cual hace referencia al uso de una herramienta computacional a manera de software dentro de una computadora permite la creación, modificación y optimización de varios recursos tales como planos, modelos en dos y tres dimensiones. La herramienta que se tiene más presente a nuestro nivel es Desk Auto-CAD el cual se tenía el uso para creación, modificación y visualización de planos, figuras geométricas y modelos siendo utilizado en dibujo técnico

Una herramienta EDA (Electronic Design Automation) son herramientas diseñadas específicamente a proyectos y producción de sistemas electrónicos tales como pueden ser circuitos integrados, compuertas y hasta el desarrollo de placas fenólicas para un circuito impreso (PCB) siendo que se utiliza mayormente como un software dedicado a la parte esquemática del proyecto, la integración de los componentes en previsualización para la parte funcional y física del mismo, y la integración final de manera física del proyecto

4.2 Lenguajes de descripción de hardware (HDL)

Se trata de lenguajes de programación que se especializa en mayor parte para definir la estructura, diseño y operación de los circuitos electrónicos analógicos, aunque es más común que se trate de circuitos electrónicos digitales siendo que se usa de mayor manera en sistemas digitales, siendo estos mismo de vital ayuda o bien de gran aportación para el análisis de automatización y la simulación de los circuitos a manera de software antes de ser probado de manera física

Los HDL utilizan expresiones estándar basadas en texto que reflejan la estructura de los circuitos electrónicos, si se viera dentro de una tarjeta, se podrían observar más de un millón de compuertas a disposición del programador. Al igual que los lenguajes de programación concurrentes, la sintaxis y semántica de los HDL incluyen notaciones específicas para la concurrencia. Sin embargo, al contrario de lo que ocurre con la mayoría de los lenguajes de programación, los HDL incluyen también una notación específica para el tiempo, debido a que este es una característica fundamental en los circuitos electrónicos reales.

4.3 Los dispositivos lógicos programables (PLD's)

Un PLD (Programmable Logic Device, Dispositivo lógico programable) es un componente electrónico empleado para la fabricación de circuitos digitales. A diferencia de las puertas lógicas un PLD tiene una función indefinida. Antes de que un PLD pueda ser usado en un circuito este puede ser programado.

Un PLD está formado por una matriz de compuertas AND y puertas OR, que se pueden programar para conseguir funciones lógicas específicas. Existen cuatro tipos de dispositivos que se clasifican como PLD.

4.3.1 La GAL 22V10

El microprocesador GAL22V10, con un tiempo de retardo de propagación máximo de 4ns, combina un proceso CMOS (complementary metal-oxide-semiconductor) de alto rendimiento con borrable eléctrico (E2).

Tecnología de puerta flotante para proporcionar el mayor rendimiento disponible de cualquier dispositivo 22V10 en el mercado. Los circuitos CMOS permiten el GAL22V10 (ver Figura 1) para consumir mucha menos energía en comparación con dispositivos bipolares 22V10. E2 la tecnología ofrece alta velocidad (< 100ms) borrar tiempos, proporcionando la capacidad de reprogramar o reconfigurar el dispositivo de forma rápida y eficiente.

La arquitectura genérica proporciona la máxima flexibilidad de diseño permitiendo que la macro célula lógica de salida (OLMC) sea configurada por el usuario. El GAL22V10 (ver figura 1) es totalmente funcional, compatible con dispositivos bipolares estándar y CMOS 22V10 (Quiles, Ortiz, Moreno, & Benavides Benítez).

4.4 Herramientas de desarrollo

Galaxy Cypress

El compilador de síntesis Warp es un compilador VHDL de última generación para diseñar con PLD y CPLD. Warp utiliza un subconjunto de VHDL como lenguaje de descripción de hardware (HDL) para el diseño. Warp acepta la entrada de texto VHDL y luego sintetiza y optimiza el diseño para el hardware de destino. Warp luego genera un mapa jedec para programar PLD y CPLD, como se muestra en la Figura 3. El mapa jedec que produce Warp al apuntar a PLD y CPLD se puede utilizar para programar partes con un programador de dispositivos. El mapa también se puede utilizar como entrada para Nova ™ simulador funcional. Nova es un simulador gráfico interactivo que permite al usuario examinar el comportamiento de los diseños sintetizados (ACM, 2019).

Xilinx ISE

Es una herramienta de software discontinuada de Xilinx para la síntesis y el análisis de diseños HDL, cuyo objetivo principal es el desarrollo de firmware integrado para las familias de productos de circuitos integrados (IC) FPGA y CPLD de Xilinx... Le sucedió Xilinx Vivado. El uso de la última edición lanzada de octubre de 2013 continúa para la programación en el sistema de hardware heredado diseños que contienen FPGA y CPLD más antiguos que de otro modo quedarían huérfanos por la herramienta de diseño de reemplazo, Vivado Design Suite.

ISE permite al desarrollador sintetizar ("compilar") sus diseños, realizar análisis de tiempo, examinar diagramas RTL, simular la reacción de un diseño a diferentes estímulos y configurar el dispositivo de destino con el programador. Otros componentes enviados con Xilinx ISE incluyen el kit de desarrollo integrado (EDK), un kit de desarrollo de software (SDK) y ChipScope Pro. [4] El ISE de Xilinx se utiliza principalmente para la síntesis y el diseño de circuitos, mientras que el simulador lógico ISIM o ModelSim se utiliza para las pruebas a nivel de sistema.

Altera Blaster USB

El altera USB Blaster es un gran dispositivo de programación FPGA. Este dispositivo conecta su PC a un cabezal de 10 pines conectado a su FPGA y le permitirá enviar datos de configuración desde su PC a la FPGA durante la creación de prototipos o programar datos en el sistema durante la producción. El blaster es compatible con máquinas Windows y Linux. Con el 10-pin integrado en el dispositivo, se hace para el uso sin esfuerzo.

El blaster USB contiene 3 tipos diferentes de modos de programación, entre los que se incluyen:
Grupo de acción de prueba conjunta (JTAG): Programa y configura todos los dispositivos altera
Serie pasiva: Configura todos los dispositivos altera compatibles con Quartus II.
Serie activa: Programa un único EPCs1, EPCs4, EPCs16 o EPCs128.

V DESARROLLO

5.1 Ambiente de desarrollo

Galaxy Cypress

El compilador de síntesis Warp es un compilador VHDL de última generación para diseñar con PLD y CPLD. Warp utiliza un subconjunto de VHDL como lenguaje de descripción de hardware (HDL) para el diseño. Warp acepta la entrada de texto VHDL y luego sintetiza y optimiza el diseño para el hardware de destino. Warp luego genera un mapa jedec para programar PLD y CPLD, como se muestra en la Figura 3. El mapa jedec que produce Warp al apuntar a PLD y CPLD se puede utilizar para programar partes con un programador de dispositivos. El mapa también se puede utilizar como entrada para Nova ™ simulador funcional. Nova es un simulador gráfico interactivo que permite al usuario examinar el comportamiento de los diseños sintetizados (ACM, 2019).

5.2 Problema

El problema propuesto es simple, pues la finalidad es relacionarse con las herramientas CAD, comprender la mecánica del proceso de diseño utilizando PLD's y realizar la implementación física de la función deseada.

De esta manera, el problema consiste en:
Programar e implementar físicamente, verificando el correcto funcionamiento, de un decodificador (DEC). El DEC tiene tres entradas y ocho salidas. Las salidas serán capaces de mostrar, en un display de 7 segmentos, el dato decodificado que recibe en la entrada.

Nota: El DEC deberá mostrar en su salida un número de boleta, por ejemplo: 89560171

Para el diseño se deberá programar utilizando uno de los siguientes métodos: método de Ecuaciones ó When else ó With select when ó If then ó Case when, del lenguaje VHDL a través del IDE Galaxy.

El DEC deberá mostrar en su salida un número de boleta, por ejemplo: 89560171

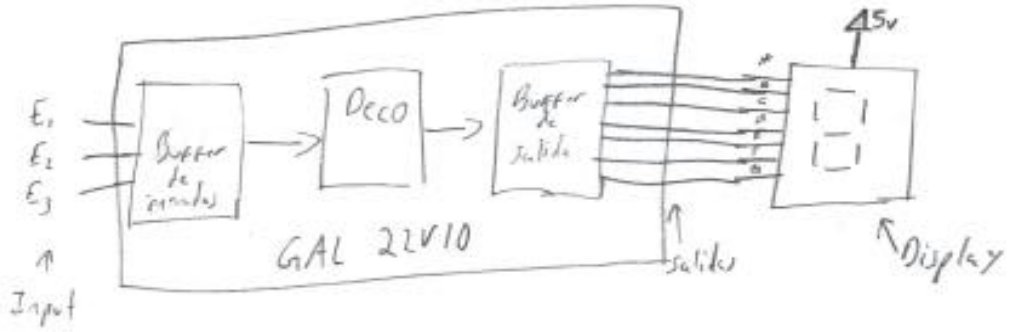

Figura 5.1. Diagrama a bloque del circuito.

5.3 Solución del problema

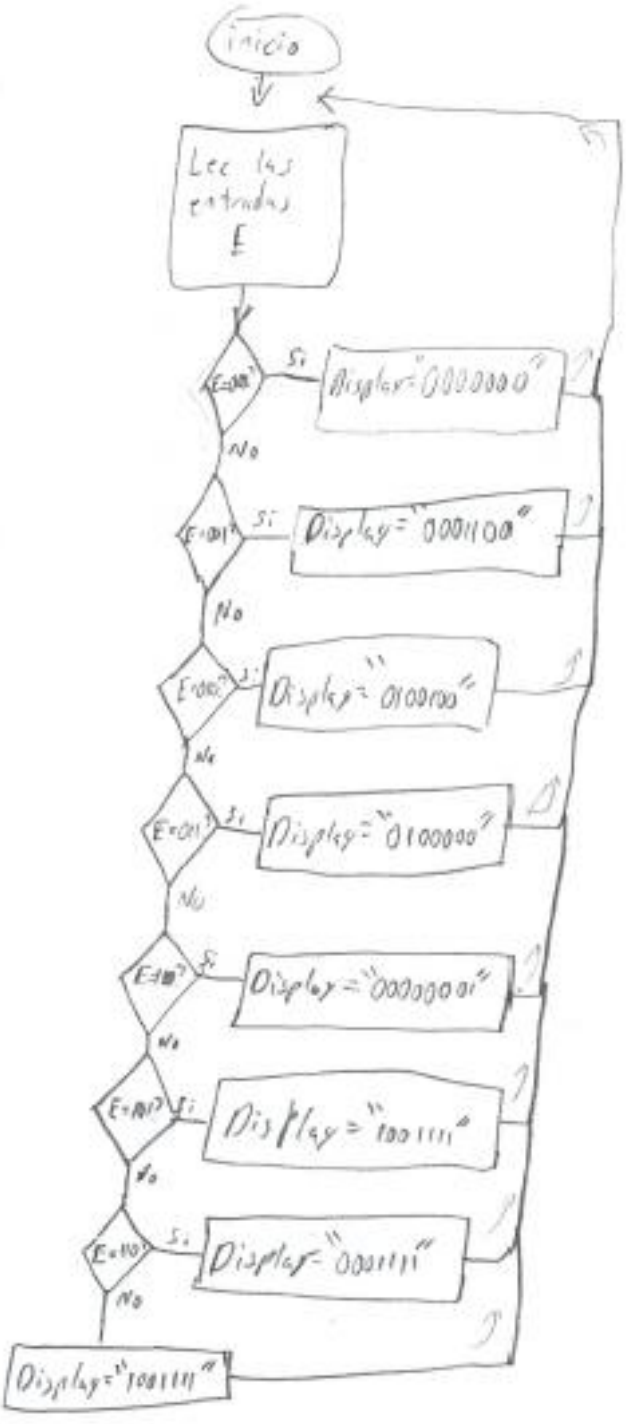

Figura 5.1. Diagrama a flujo de la metodología de diseño.

Una vez analizado el problema se procede a determinar la tabla de verdad para facilitar la escritura del código. (Nota: Este paso sería innecesario si se cuenta con la suficiente experiencia, sin embargo, siempre es recomendable hacer este ejercicio.)

Entradas			**Segmentos del Display (Salidas)**							**Número (Boleta)**
E3	E2	E1	A	B	C	D	E	F	G	
0	0	0	0	0	0	0	0	0	0	8
0	0	1	0	0	0	1	1	0	0	9
0	1	0	0	1	0	0	1	0	0	5
0	1	1	0	1	0	0	0	0	0	6
1	0	0	0	0	0	0	0	0	1	0
1	0	1	1	0	0	1	1	1	1	1
1	1	0	0	0	0	1	1	1	1	7
1	1	1	1	0	0	1	1	1	1	1

5.3.1 Edición del programa.

Una vez que el software está instalado en el computador y listo para ser utilizado, los pasos a seguir para trabajar con Galaxy son los descritos a continuación:

1. Ejecutar el software Galaxy.
2. Se crea un archivo de texto mediante: **File□New□Text File** y después **<OK>**.
3. Se escribe el código y después se salva el archivo, preferentemente con el nombre de la ENTIDAD, con terminación ***.vhd**. (En este caso se llama ***deco.vhd***)

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;

ENTITY DECO IS
   PORT (
   E: IN STD_LOGIC_VECTOR (2 DOWNTO 0);
   DISPLAY: OUT STD_LOGIC_VECTOR (6 DOWNTO 0)
   );
END DECO;


ARCHITECTURE ADECO OF DECO IS
```

```
BEGIN
   PDECO: PROCESS (E)
   BEGIN
            CASE E IS
WHEN "000" => DISPLAY <= "0000000"; -- 8
WHEN "001" => DISPLAY <= "0001100"; -- 9
WHEN "010" => DISPLAY <= "0100100"; -- 5
WHEN "011" => DISPLAY <= "0100000"; -- 6
WHEN "100" => DISPLAY <= "0000001"; -- 0
WHEN "101" => DISPLAY <= "1001111"; -- 1
WHEN "110" => DISPLAY <= "0001111"; -- 7
  WHEN OTHERS => DISPLAY <= "1001111"; -- 1
            END CASE;
   END PROCESS PDECO;
END ADECO;
```

Tabla 1.

4. Una vez salvado el archivo se crea un proyecto, incluyendo dicho archivo. Esto se hace de la siguiente manera:
   a) **File□New□ Project [Target-Device].** Esto abre una ventana, como se muestra en la figura 1, en la cual se introduce la ruta donde se salvará el proyecto, el nombre de este (***Prac3_deco***) y además se selecciona el lenguaje que se va a utilizar (VHDL o Verilog HDL), que en este caso esVHDL.
   b) Se da clic en **<siguiente>** y aparece una nueva ventana que da la opción para agregar el archivo deco.vhd, creado anteriormente, dando clic en **<add>.** Después en **<siguiente>** y se abre una nueva ventana que permite seleccionar el dispositivo a utilizar, ver figura 2. En esta práctica se utilizará un dispositivo del tipo GAL22V10, que es un **SPLD** localizado dentro de **C22V10** y se selecciona el PALCE22V10-25PC o -15PC, según sea su retardo de programación (**speed (ns)**) y se da
      <Finalizar>.
   c) Finalmente, aparece una nueva ventana que confirma que se salvó el proyecto y se da aceptar en **<si>.**

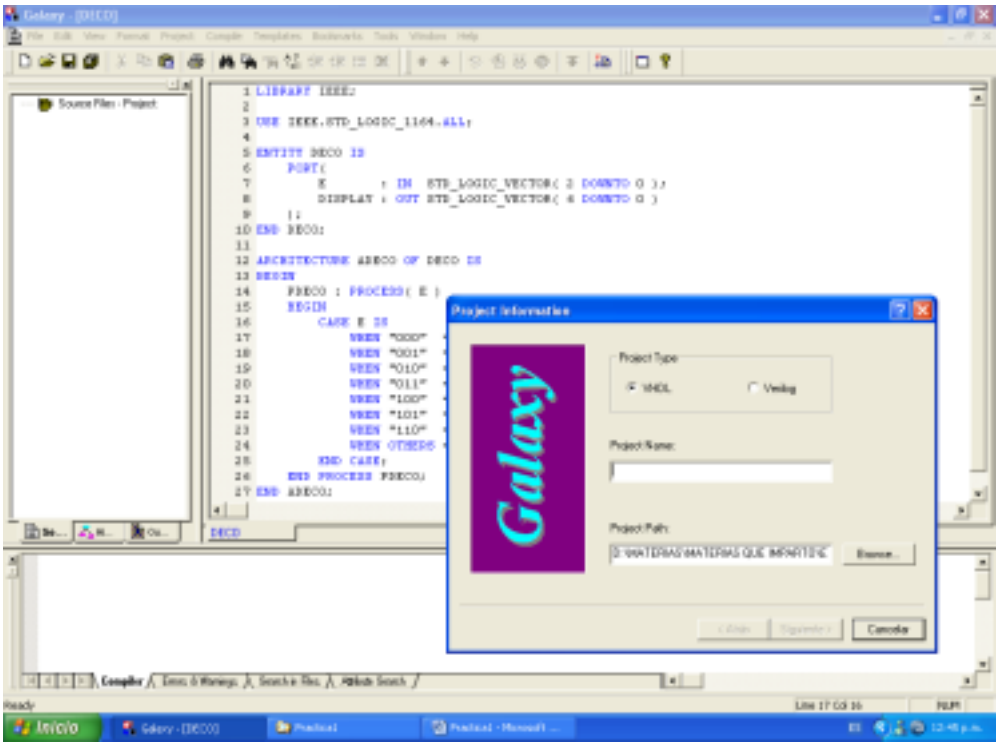

Figura 1. Nombre del proyecto (*prac3_deco*) y path.

5. Se puede observar en la figura 3, que la ventana izquierda cambia. Esto es debido al hecho de que ya es un proyecto. Se da clic con el botón derecho del ratón y se selecciona **Set Top**, todo esto es sobre el icono del archivo creado ***deco.vhd***. Además, se puede observar, en la figura 3b, que también cambia dicho icono.
6. Ya con esto se compila el archivo en: **Compile** **Selected File(s)** y se ejecuta dicho proceso. En la parte inferior de la ventana se despliega la información del resultado de la compilación. En caso de que no fuese satisfactoria la compilación es posible saber en dónde están los errores o advertencias. Para tener acceso a dicha información es necesario cambiar de pestaña. La pestaña en donde se despliegan los errores o advertencia dice: ***Errors & Warnings***. La pestaña está localizada en la parte inferior de la ventana de abajo.

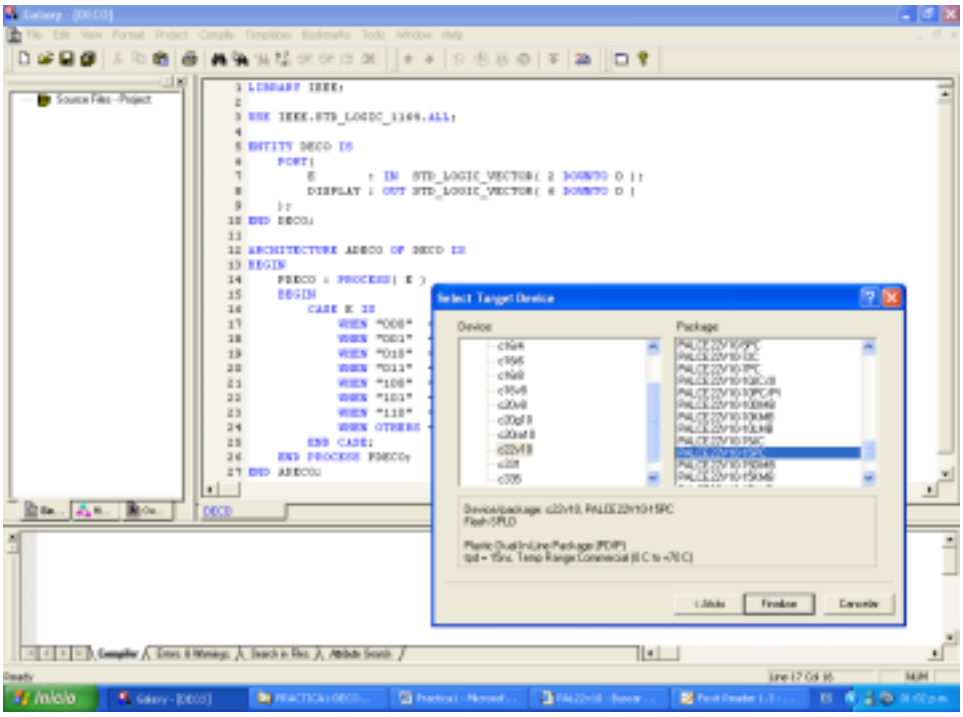

Figura 2. Selección del tipo de dispositivo.

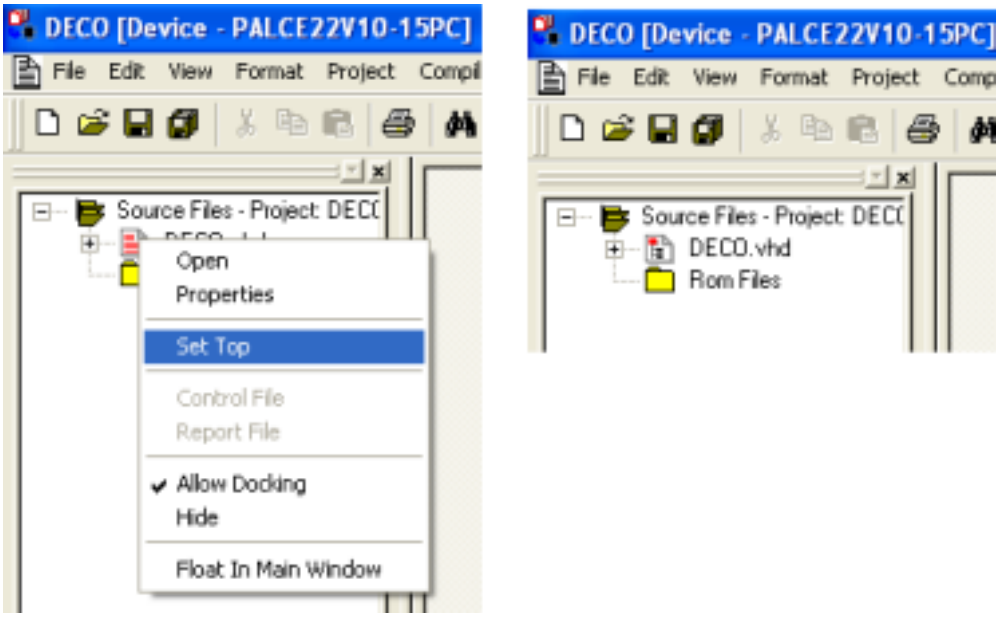

Figura 3.

7. Para corroborar el funcionamiento correcto del programa es posible realizar la **SIMULACION** del mismo. Para esto, se va a: **Tool □ Active-HDL Sim**, abriéndose una nueva ventana, como se muestra en la figura 4.

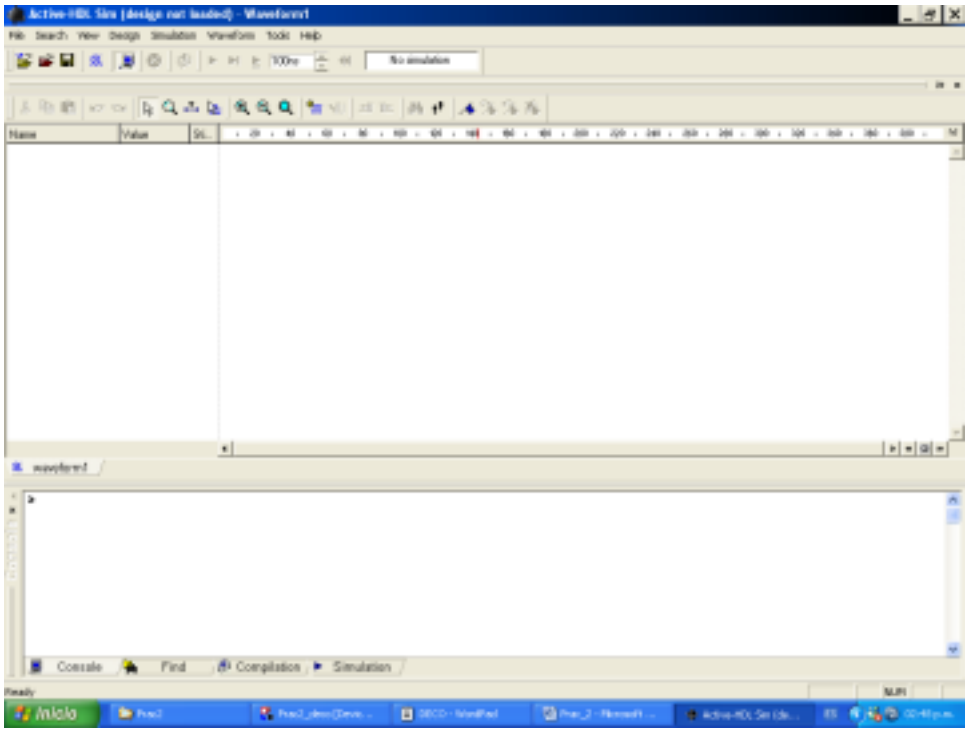

Figura 4. Ventana de simulación.

8. Se abre el archivo ***deco.vhd***, esto en: **File□Open VHDL file for simulation** y se selecciona el archivo, localizado en la carpeta ***vhd***, ***creada por el proyecto***. Se da **<Abrir>**. Al hacer eso se compila el programa apareciendo comentarios en la consola, de la ventana de simulación, como se muestra en la figura 5.

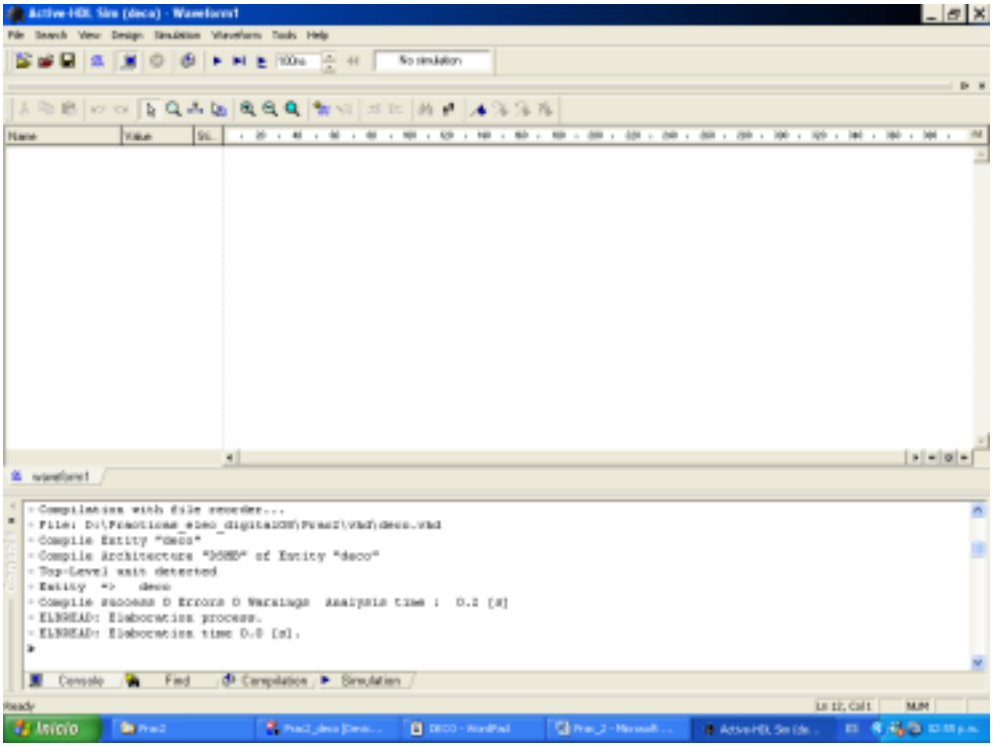

Figura 5. Ventana que muestra la compilación, en el software para la simulación.

9. Para visualizar las señales a simular es necesario agregarlas a la ventana de simulación. Para agregarlas es en: **Waveform** **Add signal**, como se muestra en la figura 6.

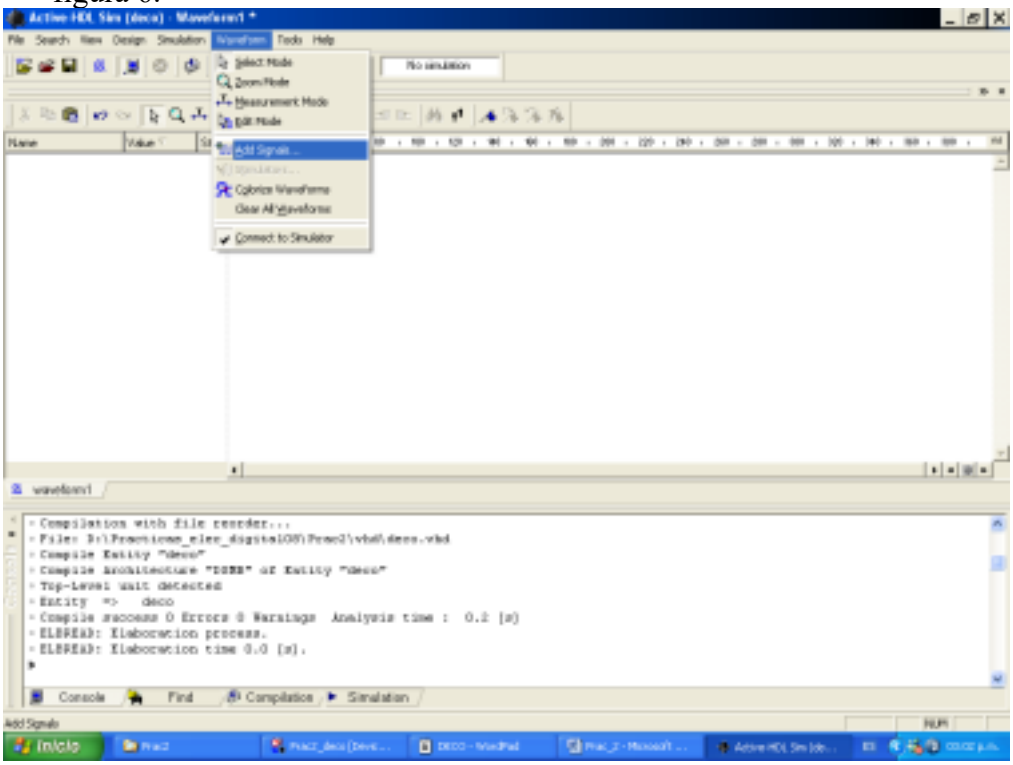

Figura 6. Ventana que muestra la compilación, en el software para la simulación.

10. Al hacer el paso anterior se abre una nueva ventana, como se muestra en la figura 7, y se seleccionan las señales a monitorear.

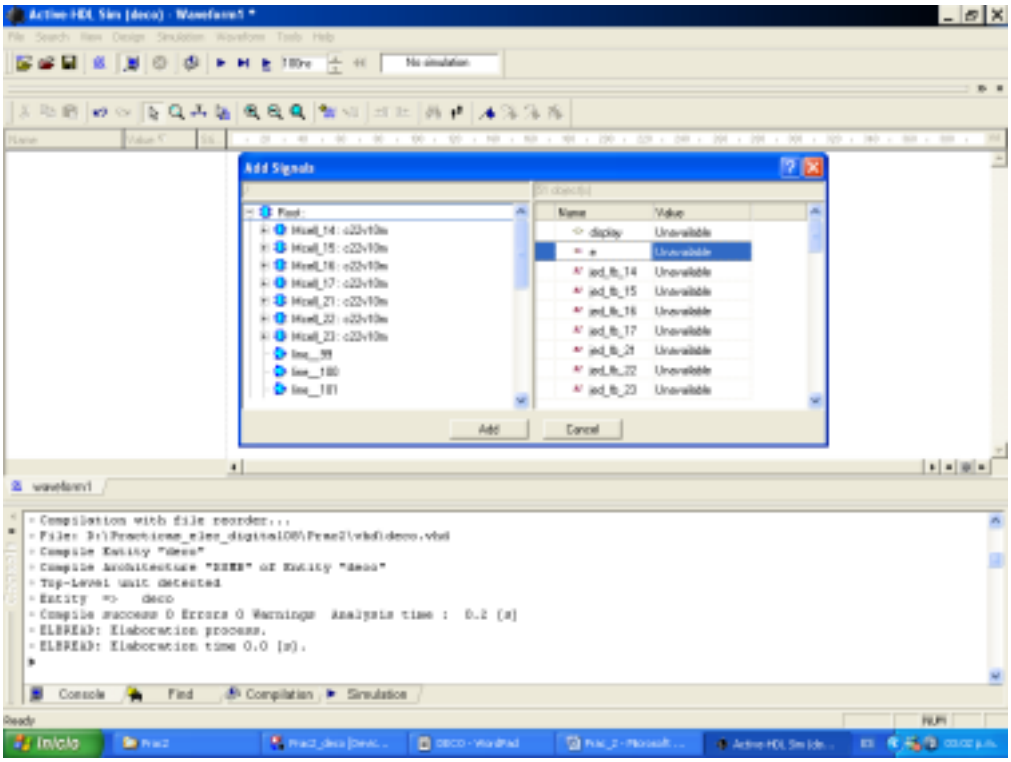

Figura 7. Ventana que muestra las señales a monitorear en la simulación.

11. En este caso se selecciona la entrada **e** y las salidas **display**. En el caso de la entrada es necesario asignar valores para su simulación. Esto se hace dando clic con el botón derecho, sobre la señal como se muestra en la figura 8. Al hacer eso se abre una pequeña "ventana" y se da clic en **Stimulators**…, abriéndose una nueva ventana, como se muestra en la figura 9.

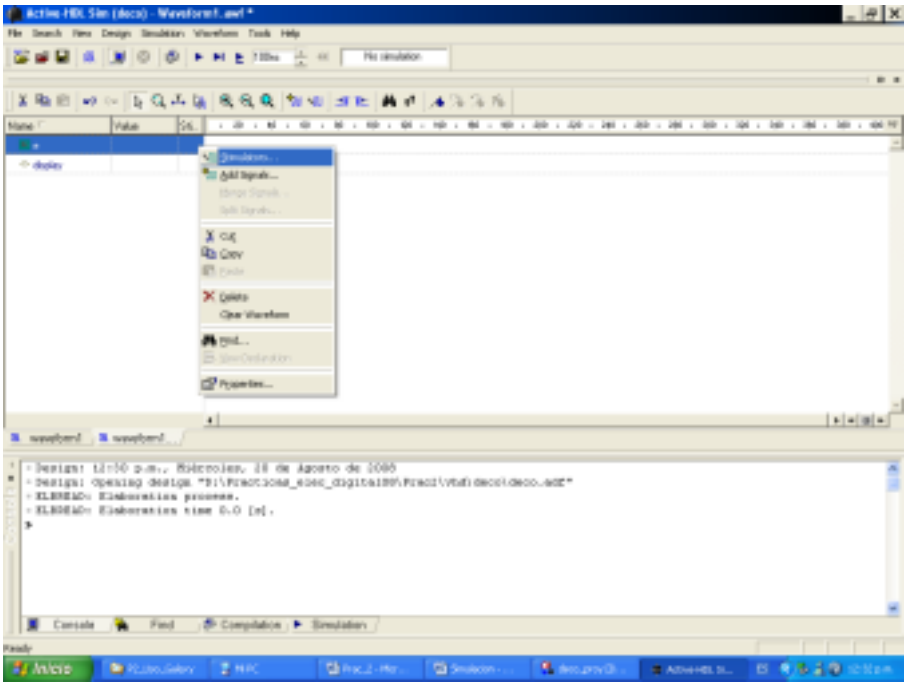

Figura 8.

12. En la nueva ventana, se indican el tipo de estímulo, en este caso daremos: **Stimulator type: value, Strength: Override, Force value:** (valor que deseamos asignar, en este caso será **000**). Para que tenga efecto se da clic en **<Apply>**.

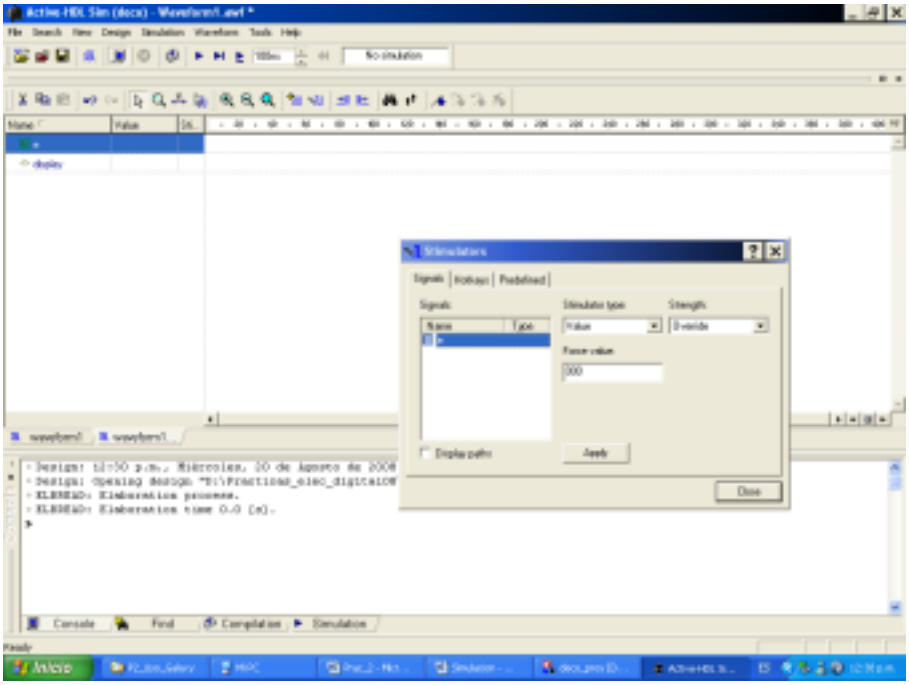

Figura 9.

13. Para observar la simulación con dicho estimulo es necesario correr dicho estimulo asignado. Esto se hace en: **Simulation Run for**. Al hacer esto se muestran los valores de entrada y se generan los valores de la salida, como se puede ver en la figura 10. Se verifica si los valores obtenidos son los deseados.

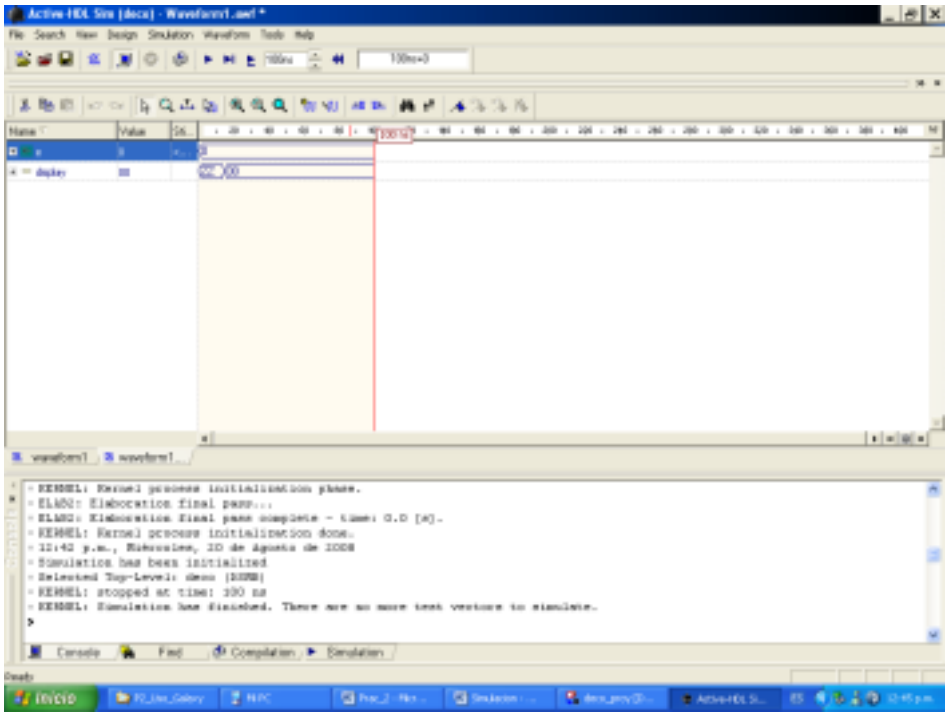

Figura 10.

14. Si se desea hacer una nueva simulación con otro valor u otro estimulo de entrada, es necesario repetir los dos pasos anteriores. Se asignar el nuevo estímulo y correr la simulación. En este caso se asignarán dos valores **010** y después el **011**. (Nota: Es posible agregar los dos estímulos y al final se ejecuta la simulación.) La simulación se muestra en la figura 11.

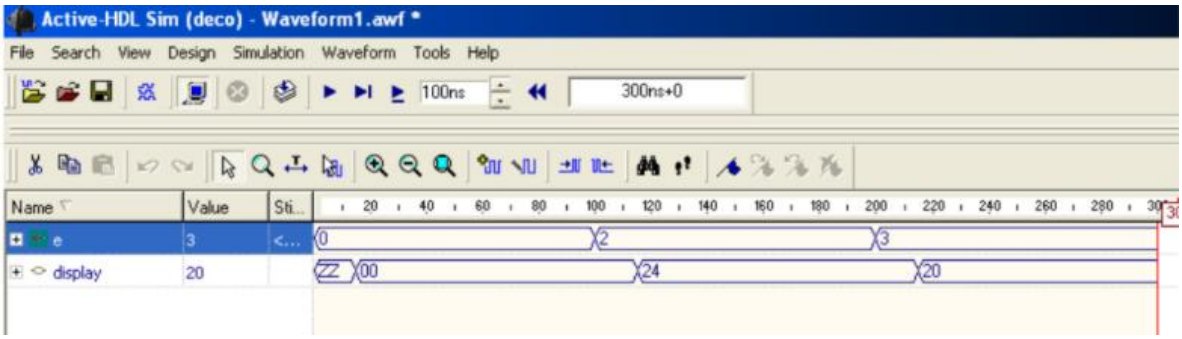

Figura 11. Simulación de tres estímulos de entrada: 000, 010 y 011.

15. Una vez que se haya revisado el correcto funcionamiento, en simulación, del sistema diseñado es necesario saber en qué terminales se encuentra cada señal de entrada/salida. El archivo para su conexionado es el reporte de salida ***Output Files***, que se localiza en la ventana izquierda del proyecto, en donde se tiene el archivo ***deco.vhd***. Para tener acceso al archivo de reporte se tiene que cambiar de pestaña, localizada en la parte inferior de la misma ventana, como se muestra en la figura 12.

Dando doble clic en el archivo ***deco.rpt*** se puede observar su contenido, el cual indica las terminalesasignadas a cada archivo y el uso del dispositivo, como muestra la tabla 2.

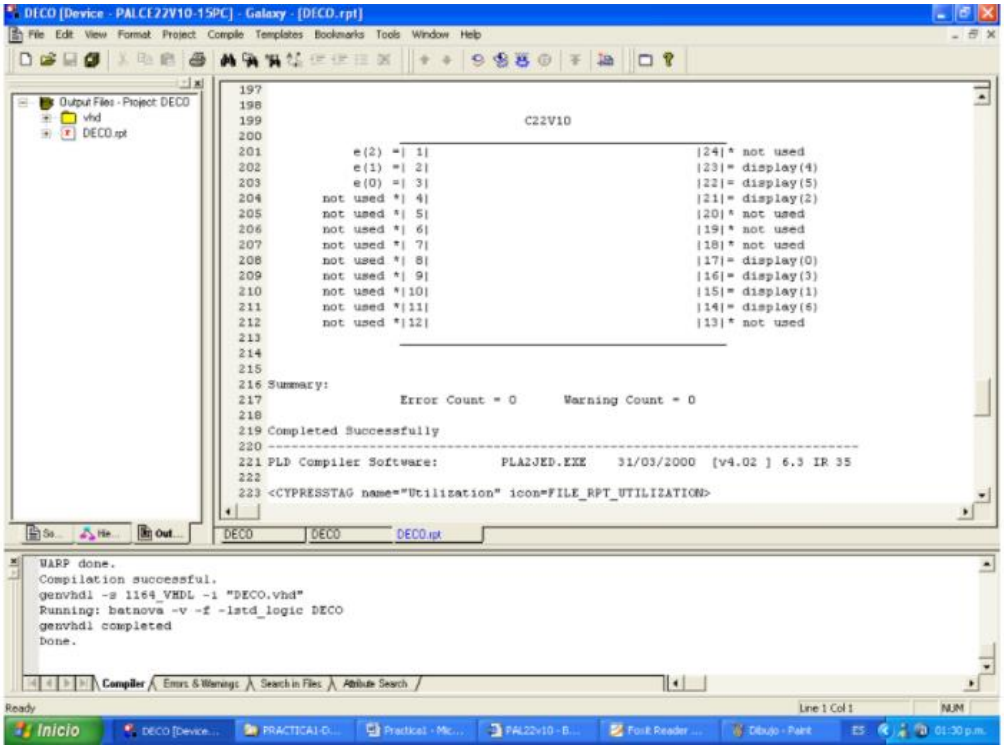

Figura 12. Reporte del diseño.

Information: Output Logic Product Term Utilization.

Node# Output Signal Name Used Max

14	display (6)	2	8
15	display (1)	2	10
16	display (3)	1	12
17	display (0)	1	14
18	Unused	0	16
19	Unused	0	16
20	Unused	0	14
21	display (2)	1	12
22	display (5)	1	10
23	display (4)	2	8
25	Unused	0	1

10 / 121 = 8 %

Tabla 2. Información de los términos productos máximos (Max) y utilizados (Used) por el dispositivo.

16. Si se detecta, como en este caso, que es posible hacer ajustes de asignación de terminales para facilitar el diseño y ensamble del circuito o bien por necesidades de una interconexión específica. Por ejemplo: tener terminales juntas es posible modificar la asignación automática por una asignación especificada por el usuario (por nosotros).

17. Es posible observar, de la tabla 2, que el número máximo, de términos productos utilizados, para cada salida es de 2. Por otro lado, casi todas las salidas (excepto la 25) tienen un máximo de 8, 10, 12, 14 ó 16
términos productos. Esto nos permite, si así lo deseamos, tener todas las salidas de manera consecutiva, esto es: display (0:6) =Terminal 14 hasta la 20. Para hacer esto es necesario modificar el programa deco.vhd y compilar nuevamente. La modificación consiste en indicar la salida que deseamos para una terminal física específica, del dispositivo. Ello se logra agregando dentro de la ENTIDAD el código que se muestra en la tabla 3.

```
ENTITY DECO IS
      PORT (.
            ..);

ATTRIBUTE PIN_NUMBERS OF DECO: ENTITY IS
"E (0):3 E (1):2 E (2):1"
& " display (0):14 display (1):15 display (2):16 display
(3):17"
& " display (4):18 display (5):19 display (6):20";

END DECO;
```

Tabla 3. Código necesario para indicar al compilador que deseamos unas terminales específicas para nuestras señales.

18. Una vez hechas las modificaciones correctamente, se realiza una nueva compilación y después se procede a revisar el archivo de salida (deco.rpt). Como se puede ver en la figura 13, el propósito se cumplió. De esta manera ya se puede programar el dispositivo.

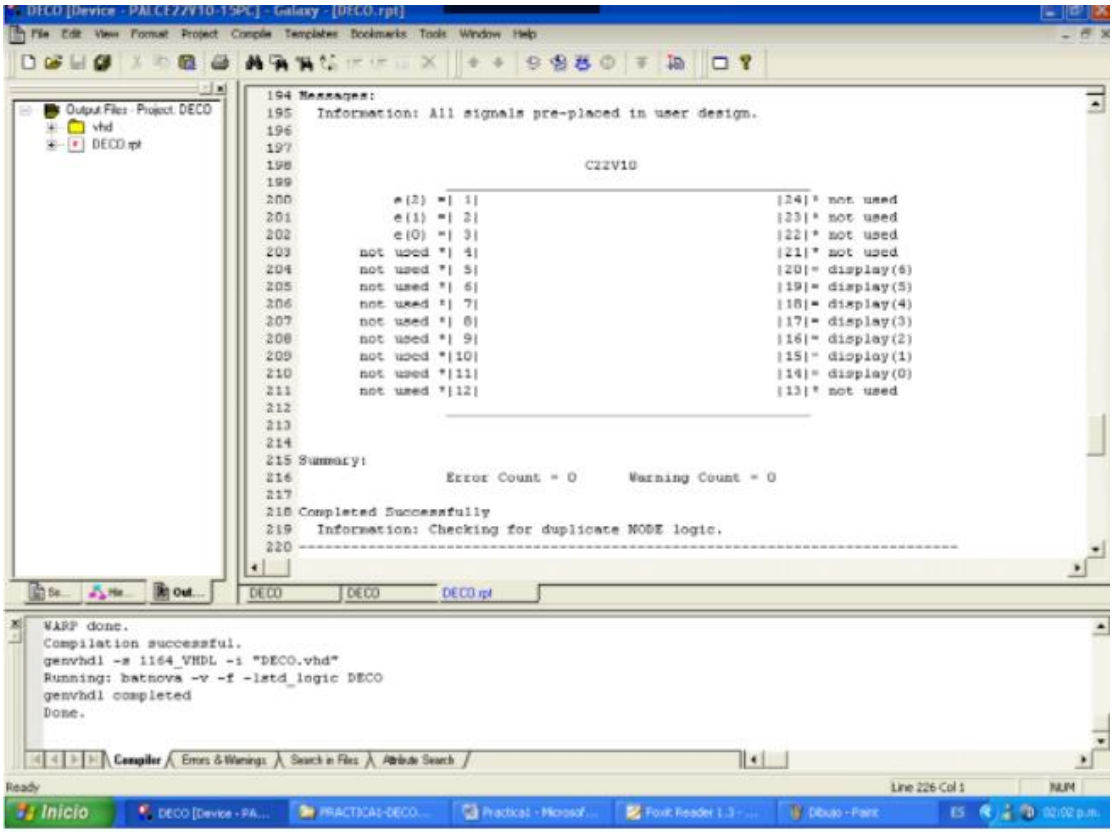

Figura 13.

Se muestra claramente la modificación de las terminales de asignación. 19.
PROGRAMACION DEL DISPOSITIVO: Para descargar el programa al dispositivo es necesario utilizar un programador. Para esto se requiere portar el archivo que se desea programar, el cual tiene terminación jedec (***deco.jed***), e ir al programador del laboratorio. En el laboratorio se tiene un programador con un software para realizar dicho propósito. En el software se selecciona el archivo e indica a que dispositivo se va a descargar, para esto se ha insertado el dispositivo en la base del programador y se procede a programarlo.

5.4 Programación

Para descargar el programa al dispositivo es necesario utilizar un programador. Para esto se requiere portar el archivo que se desea programar, el cual tiene terminación ***"jedec"*** (***deco.jed***), e ir al programador del laboratorio. En el laboratorio se tiene un programador con un software para realizar dicho propósito. En el software se selecciona el archivo e indica a que dispositivo se va a descargar, para esto se ha insertado el dispositivo en la base del programador y se procede a programarlo.

5.5 Comprobación física.

Esta última etapa consiste en realizar la prueba física al circuito y corroborar que efectivamente hace la función para la cual fue programado.

La interconexión se realiza como se muestra en la figura 14.

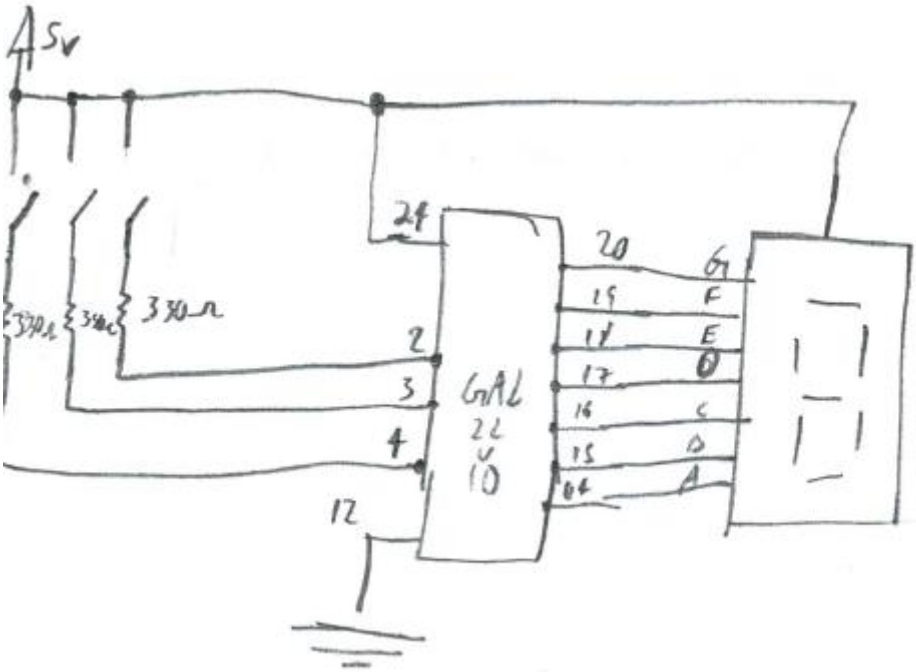

Figura 14. Circuito para comprobar el programa realizado en la sección 5.3.

VI OBSERVACIONES Y CONCLUSIONES

Con la práctica de hoy, pudimos observar cómo se programa en VHDL, cómo se modifican los pines de entrada y salida para la GAL22v10 y por último cómo cargar los programas en la GAL22v10, para qué se pueda ocupar en los circuitos electrónicos.

En esta práctica podemos analizar el cómo la teoría se comprueba gracias a lo práctico, si bien es un poco complejo el software, se logró realizar esta práctica.

Mediante la realización de la practica se pudo conocer la forma en la que se trabaja el software para VHDL así como el método de programación del GAL teniendo en cuenta la

estructura del propio código y los pines de entrada y salida, así como ver diferentes beneficios del uso de esta CAD al momento de programar simular y usar estos circuitos integrados programables

VII BIBLIOGRAFÍA

− El alumno puede consultar bibliografía y anotar la referencia. Si consulta una página de Internet ésta debe ser seria y deberá anotar dicha dirección.

Diosdado, R. (2014, 13 abril). *Resistencias de Pull-Up y Pull-Down*. Zona Maker.

https://www.zonamaker.com/electronica/intro-electronica/teoria/resistencias-de-pull-up-y-pull-down

Herramientas EDA & Design | DigiKey Electronics. (s. f.). digikey.

https://www.digikey.com.mx/es/resources/design-tools/design-tools

Wikipedia contributors. (2022, 10 marzo). *Xilinx ISE*. Wikipedia.

https://en.wikipedia.org/wiki/Xilinx_ISE