

**INSTITUTO POLITÉCNICO NACIONAL**

**ESCUELA SUPERIOR DE CÓMPUTO**

**LABORATORIO DE  
FUNDAMENTOS DE DISEÑO DIGITAL**

**PRÁCTICA No. 2**

**“USO AVANZADO DEL IDE GALAXY”**

GRUPO: \_\_\_\_\_

EQUIPO: \_\_\_\_\_

INTEGRANTES: \_\_\_\_\_  
\_\_\_\_\_  
\_\_\_\_\_

PROFESOR: \_\_\_\_\_

FECHA DE REALIZACIÓN: \_\_\_\_\_

FECHA DE ENTREGA: \_\_\_\_\_

COMENTARIOS DEL PROF: \_\_\_\_\_  
\_\_\_\_\_  
\_\_\_\_\_  
\_\_\_\_\_

## **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 1K $\Omega$
- 7 Resistencias de 220 $\Omega$
- 1 Display de Ánodo común
- 

## **IV INTRODUCCIÓN TEÓRICA:**

### **4.1 Las herramientas CAD-EDA.**

Agregar una breve descripción de dichos temas.

### **4.2 Lenguajes de descripción de hardware (HDL)**

Agregar una breve descripción de dicho tema.

### **4.3 Los dispositivos lógicos programables (PLD's)**

Agregar una breve descripción de dichos dispositivos.

#### **4.3.1 La GAL 22V10**

Agregar una descripción del dispositivo, tal como la arquitectura, registros, etc. (Ver manual de dicho dispositivo).

### **4.4 Herramientas de desarrollo**

Hablar de las herramientas de desarrollo utilizadas y existentes en la literatura y la industria. Tales como: ISE de Xilinx, Quartus II software de Altera, Galaxy de Cypress Semiconductor Corporation, etc.

**También:**

Hablar de herramientas hardware para programar, como el programador universal utilizado en la escuela, USB Blaster, etc.

## **V DESARROLLO**

### **5.1 Ambiente de desarrollo**

Hablar brevemente del ambiente de desarrollo utilizado en esta práctica.

Ejemplo:

*El ambiente de desarrollo aquí utilizado es el **Galaxy v. 6.3IR35**, de Cypress Semiconductor Corporation. El ambiente es un "Entorno de Desarrollo" conformado por "tales" herramientas.....*

### **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 capaz 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 boleto, 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 boleto, por ejemplo: 89560171**

**(Nota: Dibujar un diagrama a bloque del circuito deseado)**

Figura 5.1. Diagrama a bloque del circuito.

### **5.3 Solución del problema**

**(Nota: Dibujar el diagrama a flujo de la metodología de diseño)**

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 esta 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 del mismo (**Prac2\_deco**) y además se selecciona el lenguaje que se va a utilizar (VHDL o Verilog HDL), que en este caso es VHDL.
  - 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 ésta práctica se utilizará un dispositivo del tipo GAL22V10, que es un **SPLD** localizado dentro de **C22V10** y se selecciona el PALCE22V10-25PC ó -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>**.

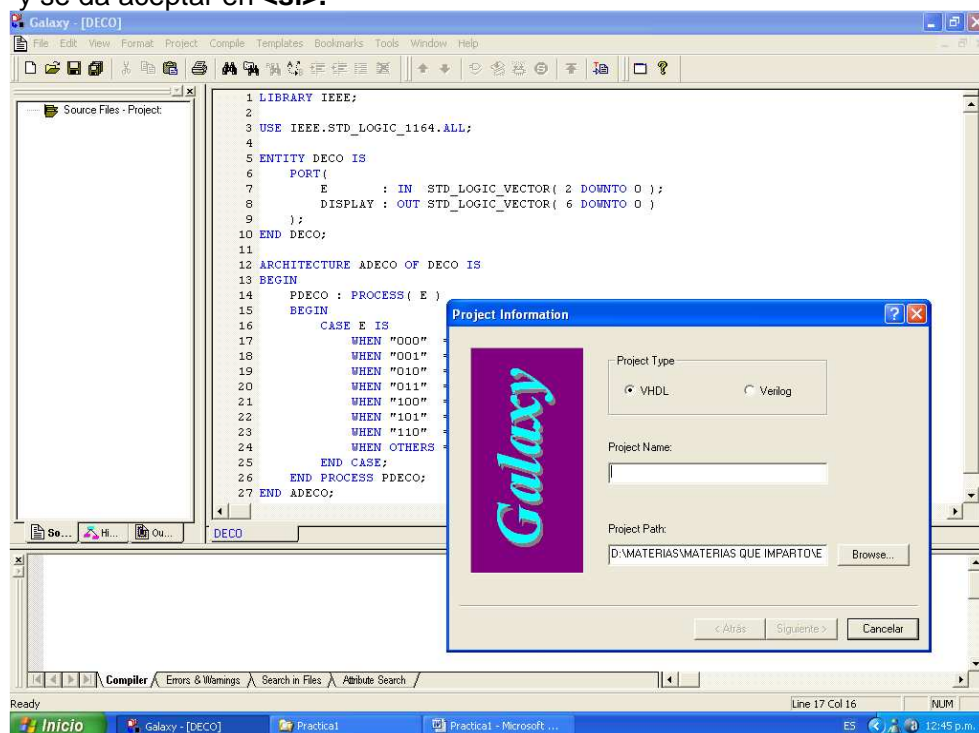


Figura 1. Nombre del proyecto (*prac2\_deco*) y path.

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 donde 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.

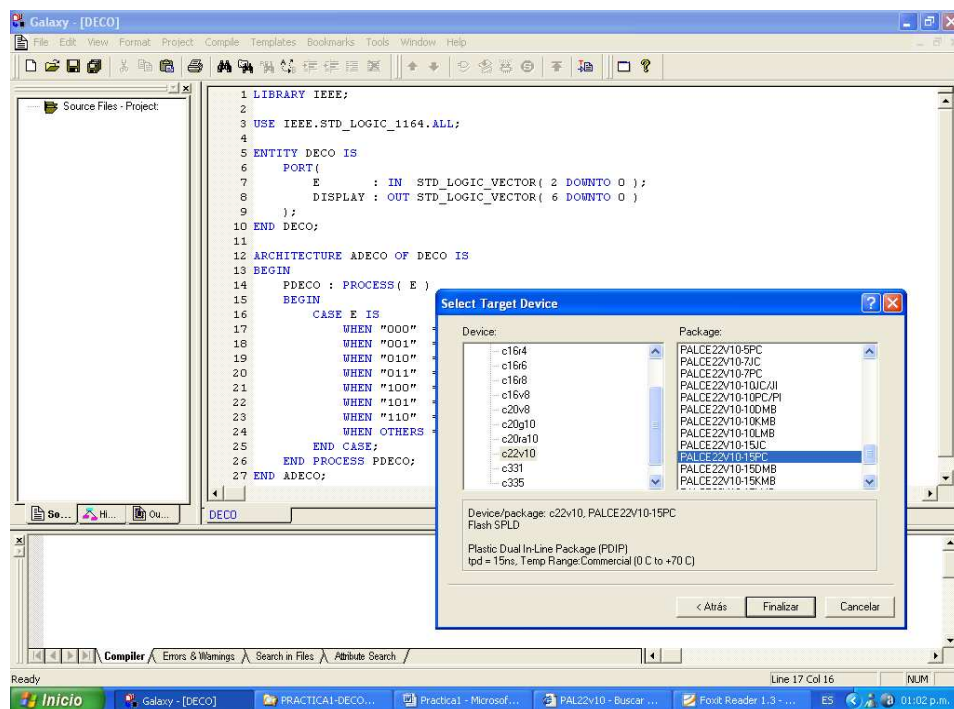


Figura 2. Selección del tipo de dispositivo.

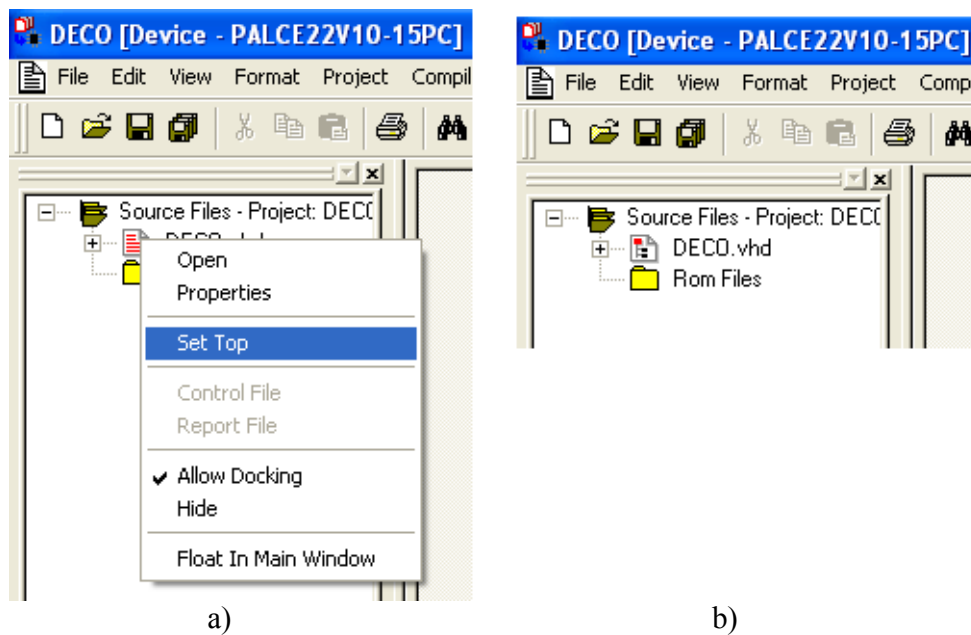


Figura 3.

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.

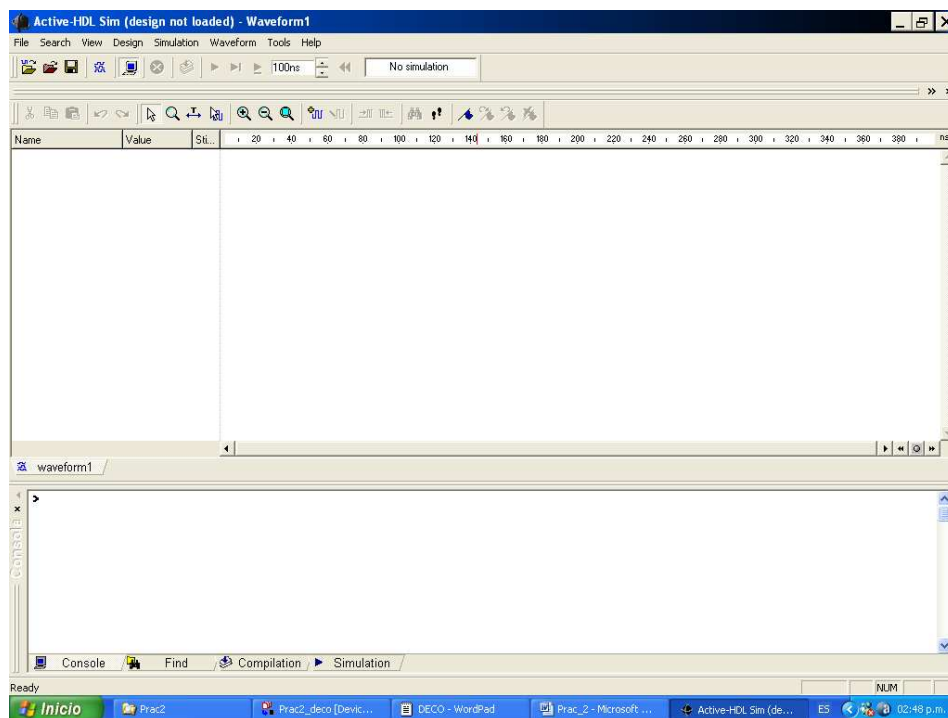


Figura 4. Ventana de simulación.

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.

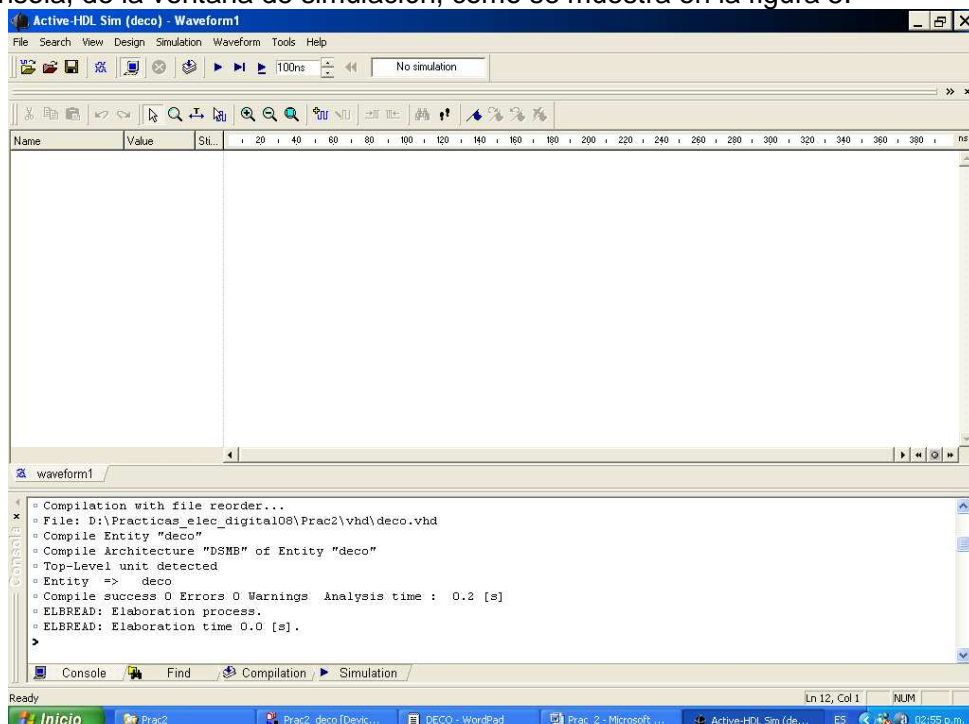


Figura 5. Ventana que muestra la compilación, en el software para la simulación.

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.

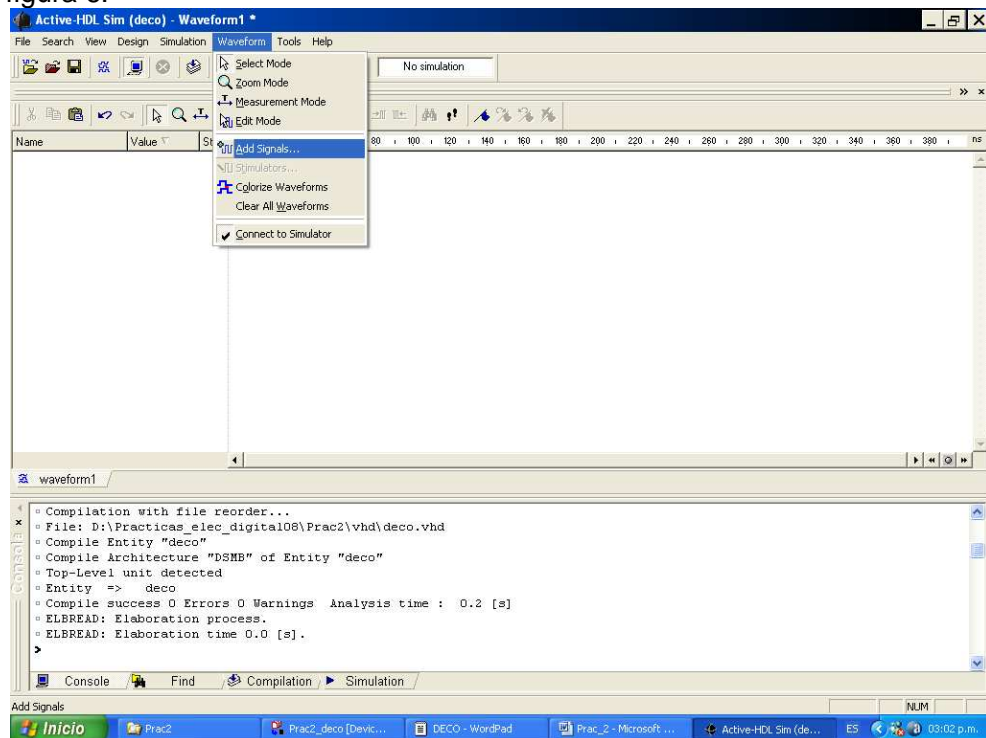


Figura 6. Ventana que muestra la compilación, en el software para la simulación.

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.

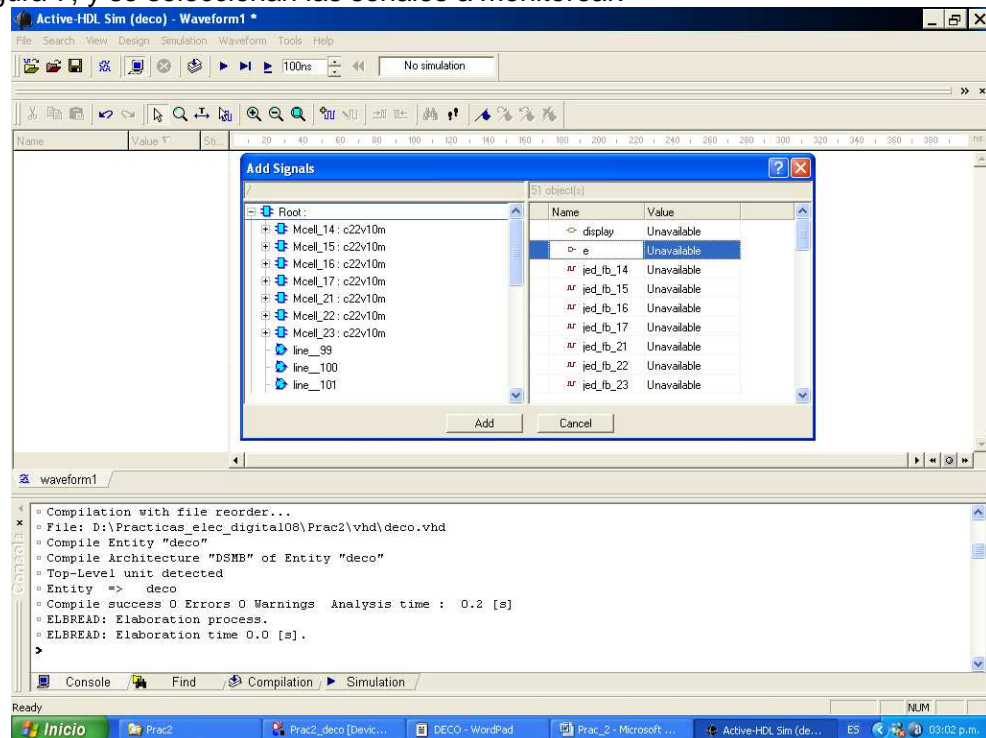


Figura 7. Ventana que muestra las señales a monitorear en la simulación.



11. En este caso se selecciona la entrada **e** y las salidas **display**. En el caso de la entrada e es necesario asignar valores para su simulación. Esto se hace dando clic con el botón derecho, sobre la señal **e** 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.

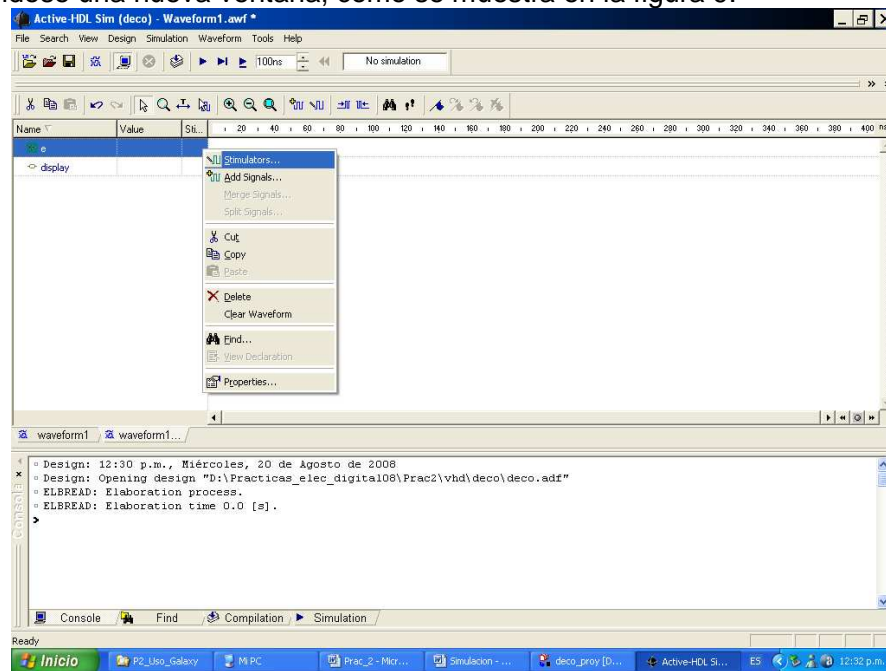


Figura 8.

12. En la nueva ventana, se indican el tipo de estímulo, en esta 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>**.

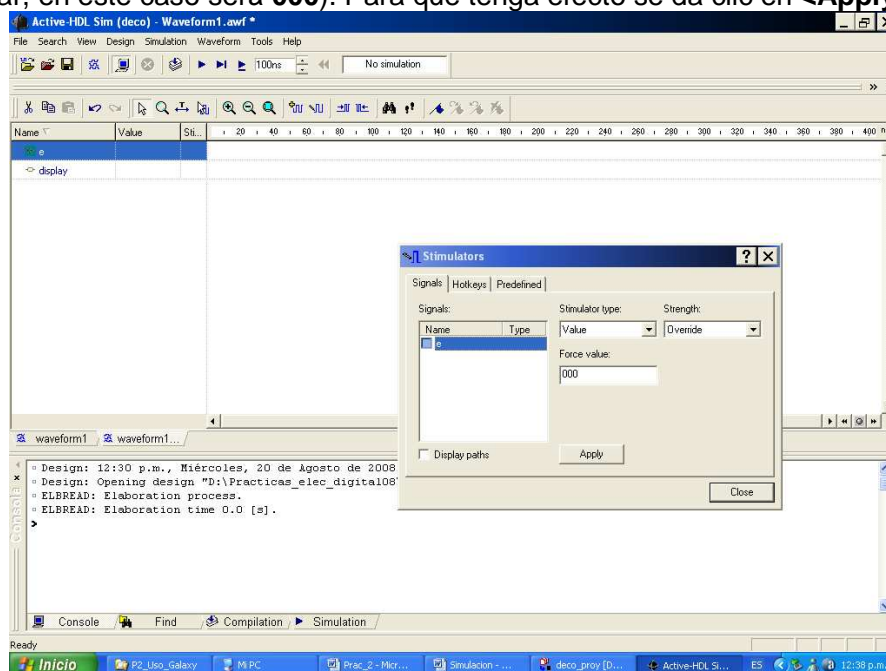


Figura 9.

13. Para observar la simulación con dicho estímulo es necesario correr dicho estímulo 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.

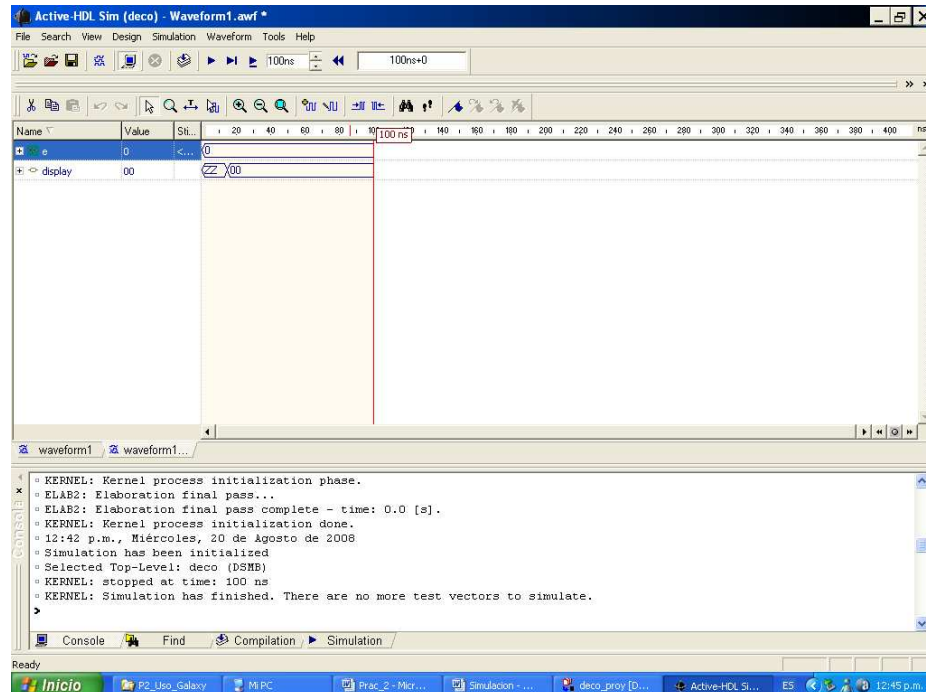


Figura 10.

14. Si se desea hacer una nueva simulación con otro valor u otro estímulo de entrada, es necesario repetir los dos pasos anteriores. Se asignar el nuevo estímulo y correr la simulación. En este caso se asignaran 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.

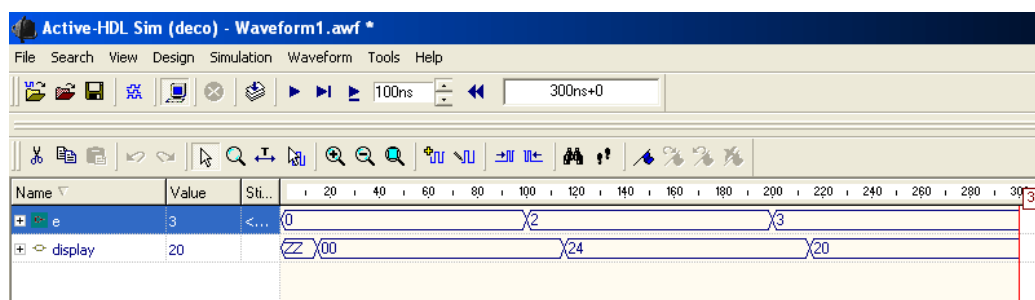


Figura 11. Simulación de tres estímulos de entrada: 000, 010 y 011.

15. Una vez que se haya revisado el correcto funcionamiento, en simulación, del sistema diseñado es necesario saber en que terminales se encuentra cada señal de entrada/salida. El archivo para su conexión, 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 terminales asignadas a cada archivo y el uso del dispositivo, como muestra la tabla 2.

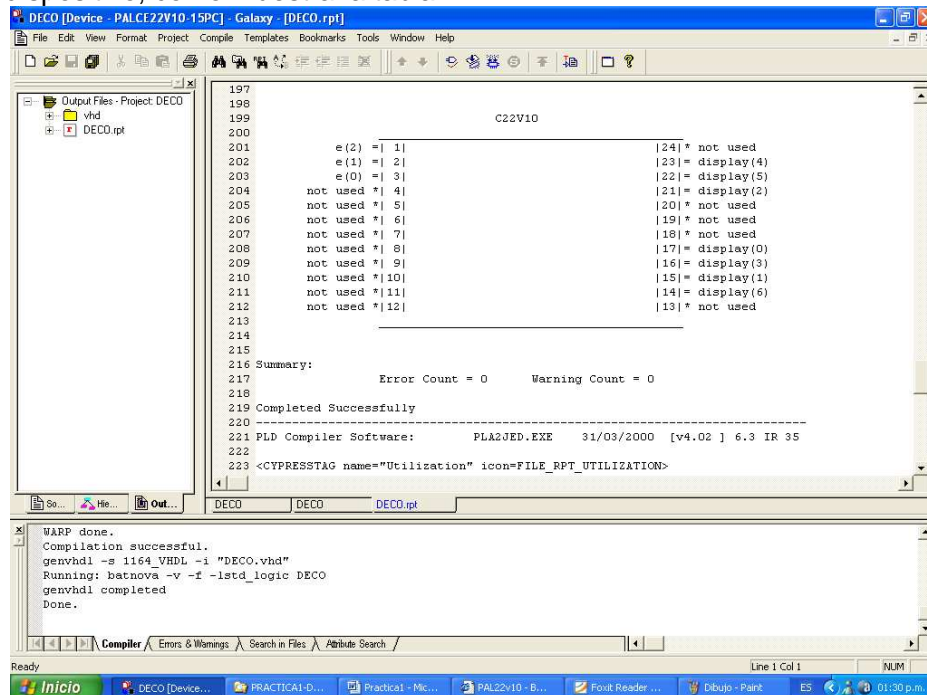


Figura 12. Reporte del diseño.

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.

- 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.

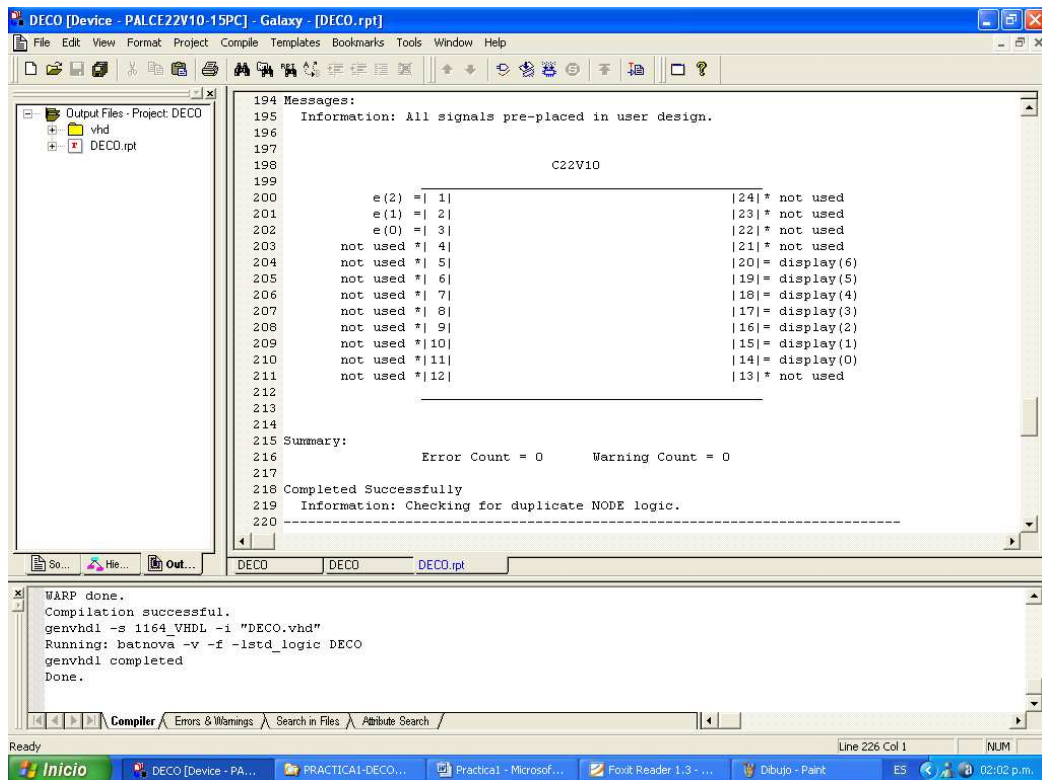


Figura 13. Se muestra claramente la modificación de las terminales de asignación.

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.

**(Nota: Dibujar como se interconectó el circuito realmente)**

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

### VI OBSERVACIONES Y CONCLUSIONES

(Son individuales)

### 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.