

**INSTITUTO POLITÉCNICO NACIONAL**

Escuela Superior de Cómputo

Práctica 2

LOPEZ PEREZ ALBERTO ANDREI

**USO DE GALAXY**

**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).

1. **OBJETIVOS ESPECÍFICOS:**
2. Distinguir los beneficios proporcionados por las herramientas CAD.
3. Confirmar el conocimiento adquirido en clase y realizar un programa en un HDL que se ejecute sobre un PLD.
4. Comprobar físicamente el correcto funcionamiento del sistema diseñado, el cual será implementando sobre un protoboard.
5. **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

1. **INTRODUCCIÓN TEÓRICA:**
   1. **LAS HERRAMIENTAS CAD-EDA.**

La automatización de diseño electrónico (EDA) se refiere a una categoría de herramientas de software enfocadas en el proyecto, concepción, y producción de sistemas electrónicos, abarcando desde el proyecto de circuitos integrados hasta el desarrollo de placas de circuito impreso.

# Lenguajes de descripción de hardware (HDL)

Un lenguaje de descripción de hardware (HDL) es un lenguaje de programación especializado que se utiliza para definir la estructura, diseño y operación de circuitos electrónicos, y más comúnmente, de circuitos electrónicos digitales, como el convertidor analógico-digital o cualquier antena satelital. Así, los lenguajes de descripción de hardware hacen posible una descripción formal de un circuito electrónico, y posibilitan su análisis automático y su simulación.

A partir de la descripción del circuito es posible simular su comportamiento mediante herramientas informáticas (simuladores) para comprobar su correcto funcionamiento antes de construir el circuito real.

Algunos tipos de descripción son:

* Funcional (asignación continua).
* Procedimental (bloques always).
* Estructural

Algunos lenguajes HDL son:

* VHDL
* Verilog
* ABEL HDL

# Los dispositivos lógicos programables (PLD’s)

# Un Dispositivo Lógico Programable (PLD) es un componente electrónico usado para construir circuitos digitales reconfigurables. A diferencia de una compuerta lógica que tiene una función fija, los PLDs salen de fábrica sin una función en específico, por lo tanto, necesitan ser programados o reconfigurados antes de poder ser usados. Los PLDs tienen varias ventajas. La primera es la habilidad de integración, que permite integrar una gran cantidad de funcionalidad en un solo chip. Los PLDs eliminan el uso de múltiples chips, así como la inconveniencia y desconfianza de usar cableado externo. La segunda ventaja es el hecho de poder cambiar el diseño. Muchos PLDs permiten ser reprogramados o reconfigurados. Existen dos ramas principales dentro de los dispositivos lógicos programables, la lógica programable de campo y la de fábrica. El término campo en este contexto implica que los dispositivos puedan ser programados en el “campo” del usuario, mientras que la lógica de fábrica puede ser programada en la misma fábrica donde se construyen, de acuerdo con los requerimientos del cliente. En este caso, la tecnología de programación usa procesos irreversibles, por lo que solo es posible hacerlo una vez.

# La GAL 22V10

# El GAL22V10 es una serie de dispositivos lógicos programables, implementados como circuitos integrados de lógica de matriz genéricos basados en CMOS y disponibles en paquetes en línea duales o portadores de chips con plomo de plástico. Es un ejemplo de un dispositivo GAL de producción estándar que se utiliza a menudo en entornos educativos como un PLD básico. Este es un arreglo lógico programable y borrable, en el tiempo de retardo máximo de propagación 4ns, combina un proceso CMOS de alto rendimiento con borrable eléctricamente (E2) flotando tecnología de compuerta para proporcionar el máximo rendimiento disponible de cualquier dispositivo 22V10 en el mercado. Circuitos CMOS permite el GAL22V10 a consumir mucha menos energía en comparación con bipolares 22V10 dispositivos. E2 tecnología ofrece alta velocidad (<100 ms) Borrar tiempos, proporcionando la capacidad de reprogramar o reconfigurar el dispositivo de forma rápida y eficiente. La arquitectura genérica proporciona máxima flexibilidad de diseño por permitiendo la salida lógica macro célula (OLMC) para ser configurado por el usuario. El GAL22V10 es totalmente función del fusible / mapa / paramétrico compatible con estándar bipolar y dispositivos CMOS 22V10. Circuitos de prueba único y células reprogramables permite completa AC, DC, y funcionales pruebas durante la fabricación. Como resultado, Lattice Semiconductor proporciona 100% de programación de campo y la funcionalidad de todos los productos GAL. Además, 100 de borrado / escritura de ciclos y retención de los datos de más de 20 años se especifican.

# Herramientas de desarrollo

Las herramientas de desarrollo utilizadas y existentes en la literatura y en la industria hoy en día son incontables, ya que durante el poco tiempo que la humanidad ha tenido con la tecnología a desarrollado una infinidad de herramientas para innovar el desarrollo de la literatura como la industria, alguna de estas herramientas es:

* ISE de Xilinx:  
  es una herramienta de software descontinuada de Xilinx para síntesis y análisis de diseños HDL, que se enfoca principalmente en el desarrollo de firmware integrado para familias de productos Xilinx FPGA y CPLD de circuito integrado (IC). El uso de la última edición publicada en octubre de 2013 continúa para la programación en el sistema de diseños de hardware heredados 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.

* Quartus II software de Altera:  
  Es una herramienta de software producida por Altera para el análisis y la síntesis de diseños realizados en HDL.

permite al desarrollador compilar sus diseños, realizar análisis temporales, examinar diagramas RTL y configurar el dispositivo de destino con el programador.

Con el programa de diseño los diseñadores pueden usar los dispositivos HardCopy Stratix de manera que puede prever y verificar su rendimiento, el cual resulta en promedio un 1050 por ciento más rápido que su FPGA equivalente. Además, en el flujo de diseño del HardCopy Stratix, Quartus II incluye una serie de utilidades que reducen el tiempo de diseño. Como contraste adicional el bajo precio del Quartus II en comparación con otras herramientas de diseño de ASIC.

* Galaxy de Cypress Semiconductor Corporation:  
  Es un Entorno de Desarrollo para Lenguaje VHDL, Utilizado en la programación de SPLD'S (como la GAL22V10) y CPLD'S.

Unas herramientas de hardware para programas son:

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, hace para el uso sin esfuerzo.

1. **DESARROLLO**

# Ambiente de desarrollo

# El ambiente de desarrollo aquí utilizado es el Galaxy emulado en Windows XP, de Cypress Semiconductor Corporation, este ambiente de desarrollo esta conformado por mi computadora desde la comodidad de mi casa debido a la cuarentena, no cuento con mesa de trabajo de Sistemas Digitales ni con un material, solo tengo mi ordenador, un Windows XP emulado por un VMware, donde tengo instalado el Galaxy.

# 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 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

Imagen que contiene reloj, medidor

Descripción generada automáticamente

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

Figura 5.1. Diagrama a bloque del circuito

# Solución del problema

**Diagrama

Descripción generada automáticamente**

## (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 |

# 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**

**DISPLAY**

**); END DECO;**

**: IN STD\_LOGIC\_VECTOR( 2 DOWNTO 0 );**

**: OUT STD\_LOGIC\_VECTOR( 6 DOWNTO 0 )**

**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.

* + - 1. Una vez salvado el archivo se crea un proyecto, incluyendo dicho archivo. Esto se hace de la siguiente manera:
         1. **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 (***Prac3\_deco***) y además se selecciona el lenguaje que se va a utilizar (VHDL o Verilog HDL), que en este caso es VHDL.
         2. 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>.

* + - * 1. Finalmente, aparece una nueva ventana que confirma que se salvó el proyecto y se da aceptar en

# <si>.

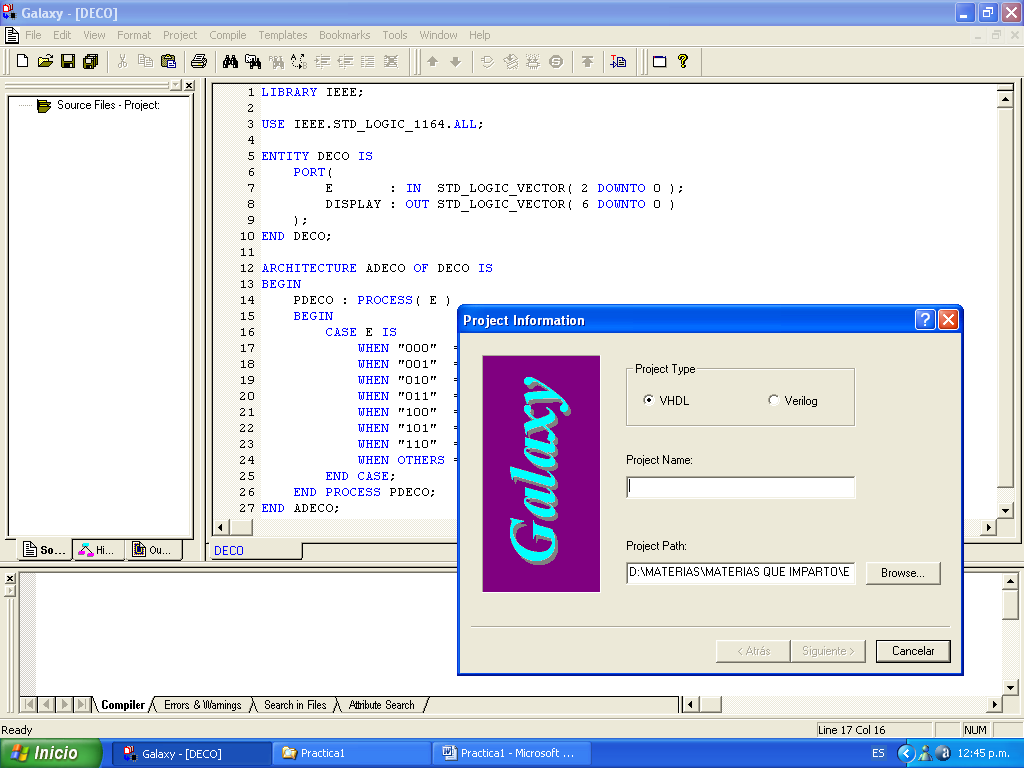


Figura 1. Nombre del proyecto (*prac3\_deco*) y path.

* + - 1. 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.
      2. 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.

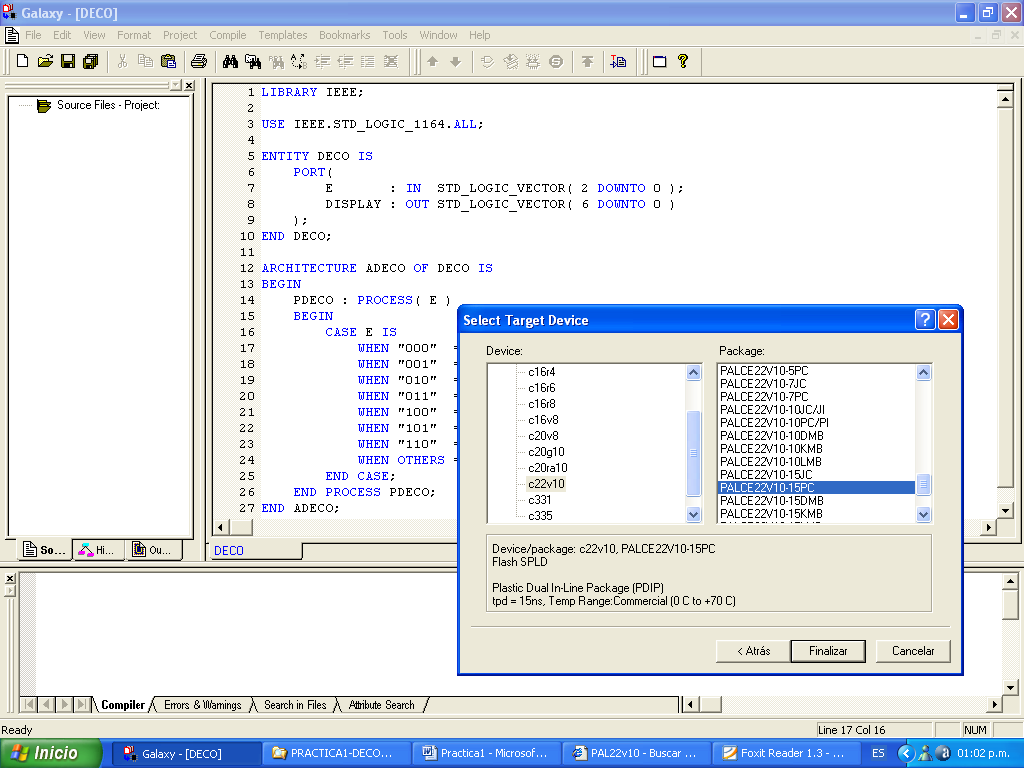
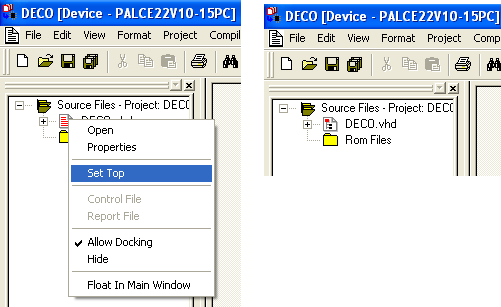


Figura 2. Selección del tipo de dispositivo.



* + - * 1. b)

Figura 3.

* + - 1. 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.

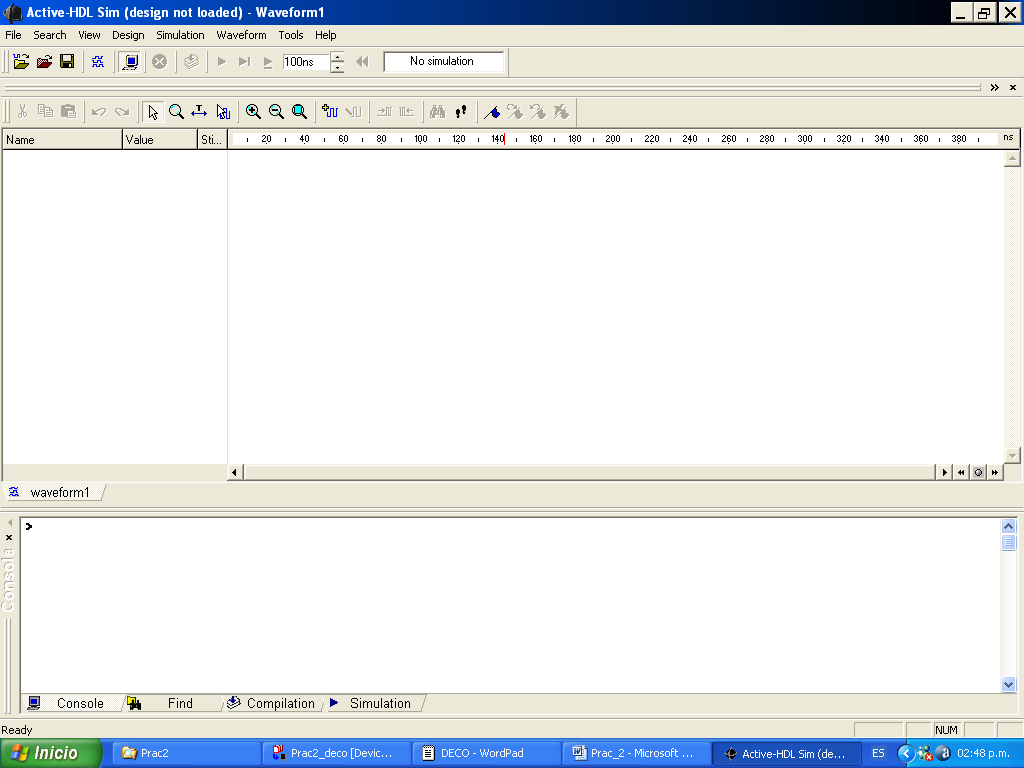


Figura 4. Ventana de simulación.

* + - 1. Se abre el archivo ***deco.vhd***, esto en: **File****Open VHDL file for simulation** y se selecciona elarchivo, 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.

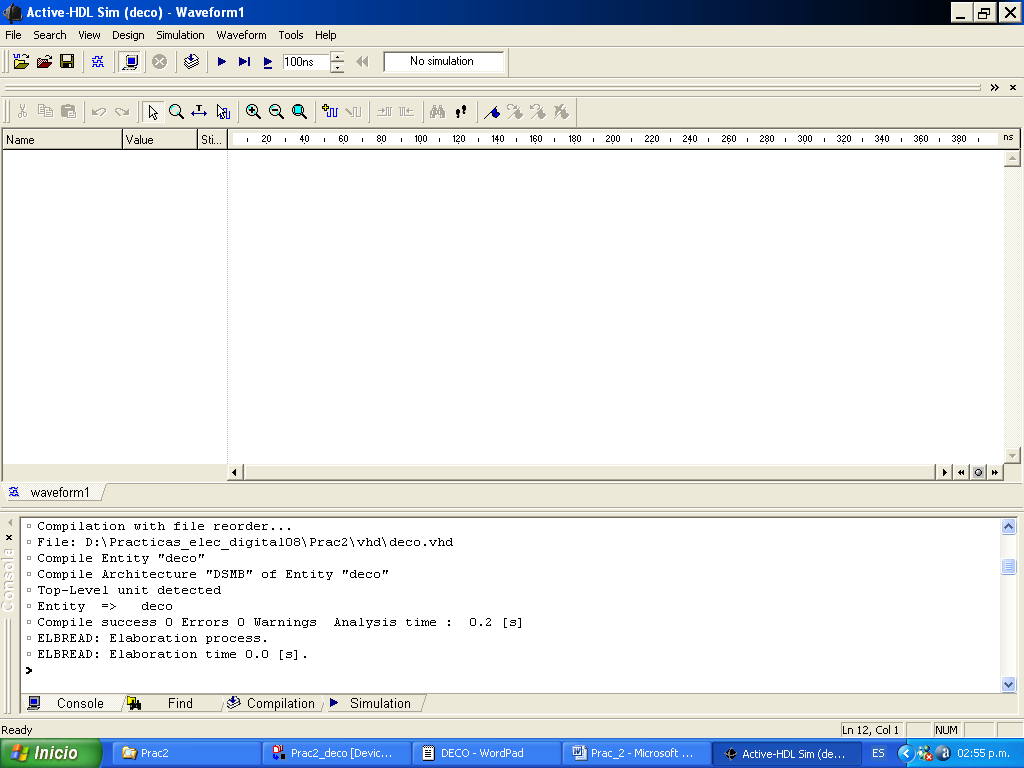


Figura 5. Ventana que muestra la compilación, en el software para la simulación.

* + - 1. 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.

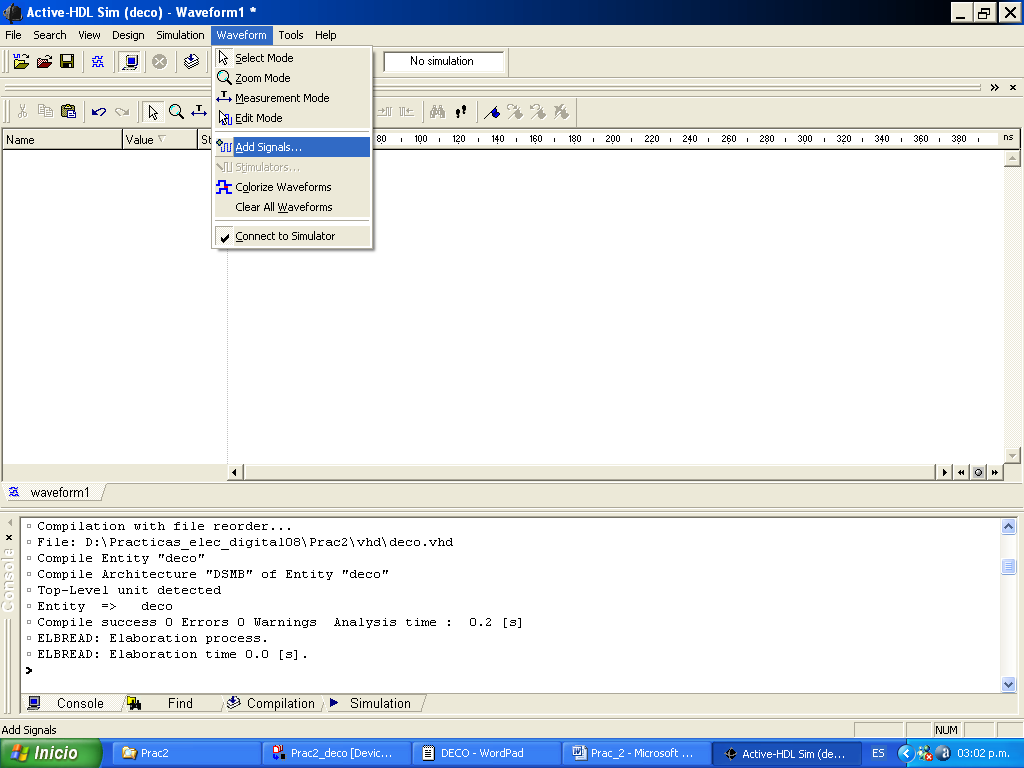


Figura 6. Ventana que muestra la compilación, en el software para la simulación.

* + - 1. Al hacer el paso anterior se abre una nueva ventana, como se muestra en la figura 7, y se seleccionan las señales a monitorear.

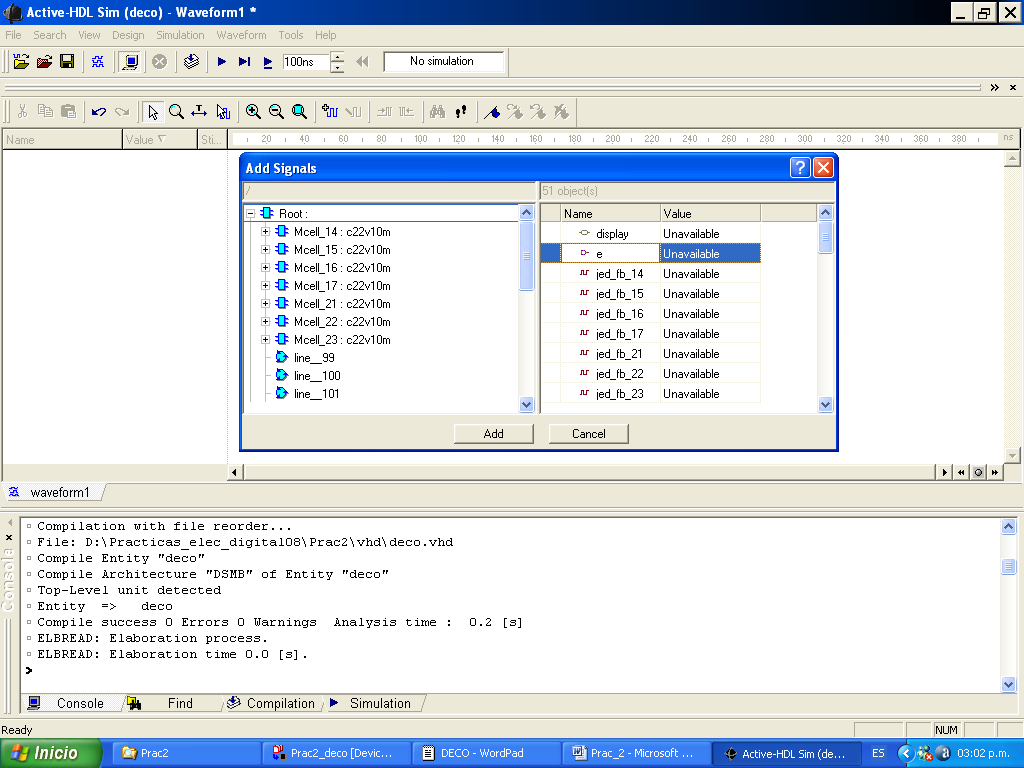


Figura 7. Ventana que muestra las señales a monitorear en la simulación.

* + - 1. 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.

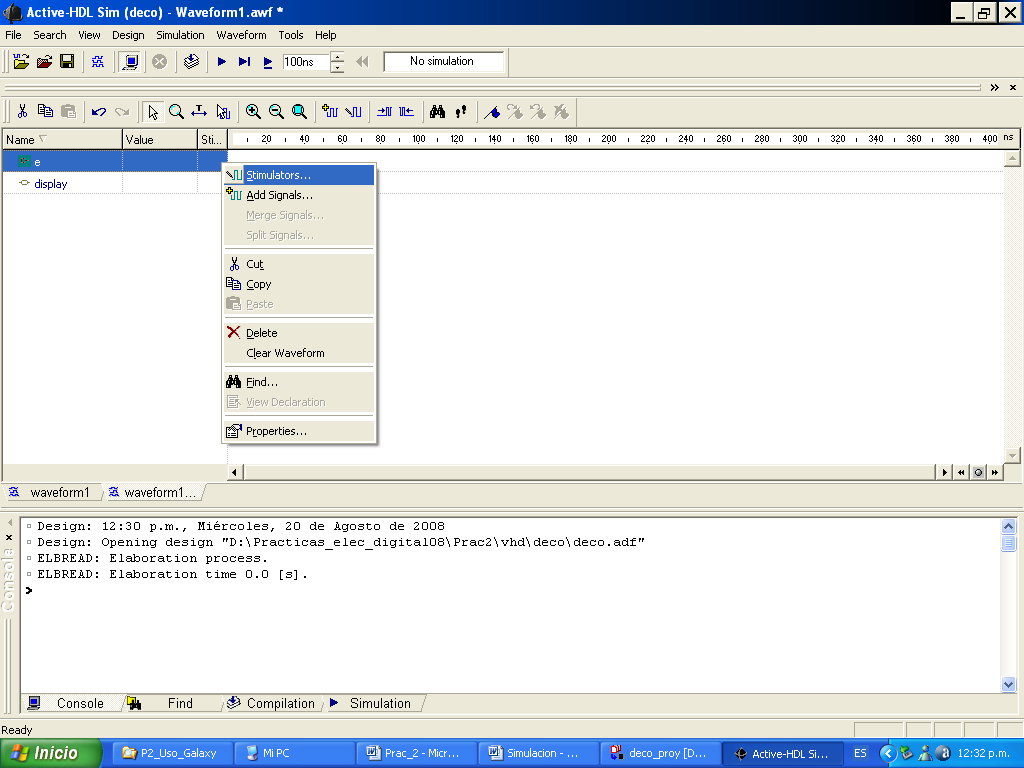


Figura 8.

* + - 1. En la nueva ventana, se indican el tipo de estimulo, 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>**.

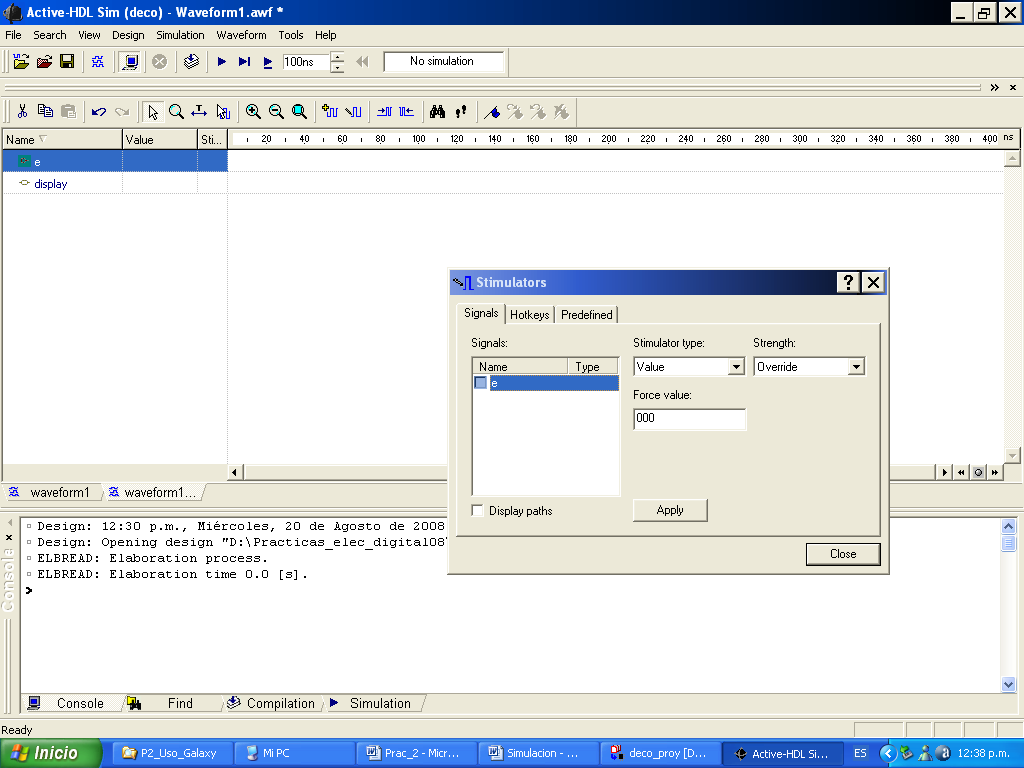


Figura 9.

* + - 1. Para observar la simulación con dicho estimulo es necesario correr dicho estimulo asignado. Estose hace en: **Simulation**  **Run for**. Al hacer esto se muestran los valores de entrada y se generanlos

Valores de la salida, como se puede ver en la figura 10. Se verifica si los valores obtenidos son los deseados.

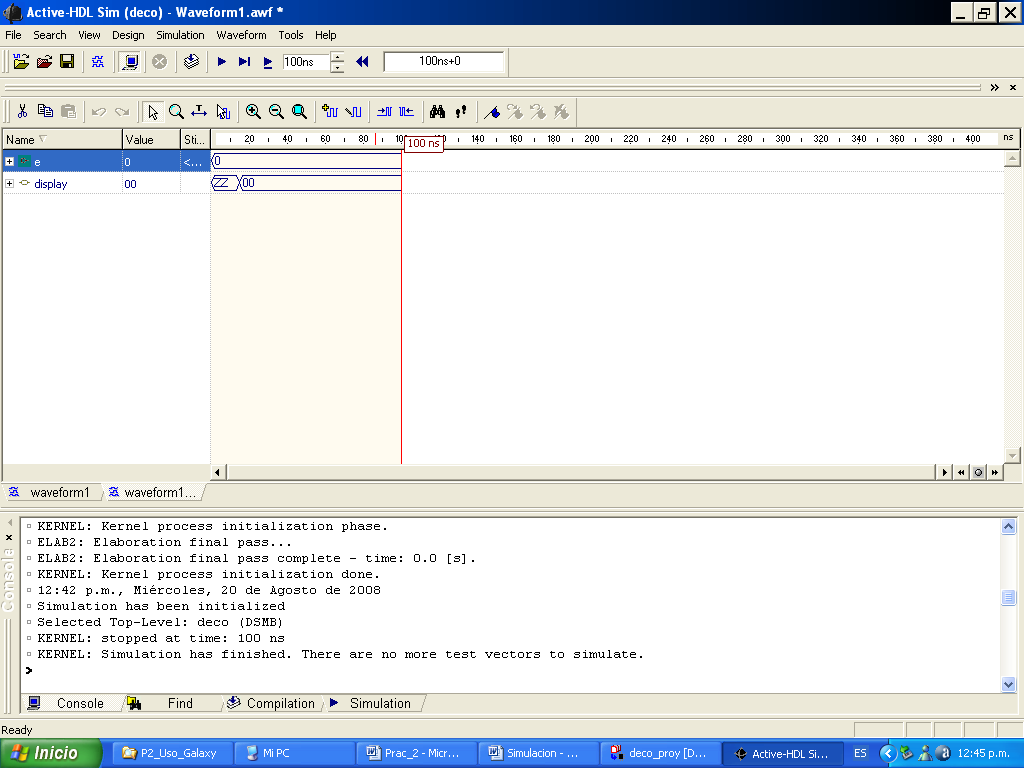


Figura 10.

* + - 1. 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 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.

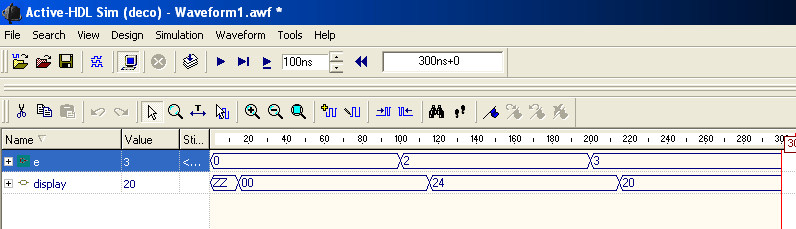


Figura 11. Simulación de tres estímulos de entrada: 000, 010 y 011.

* + - 1. 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 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.

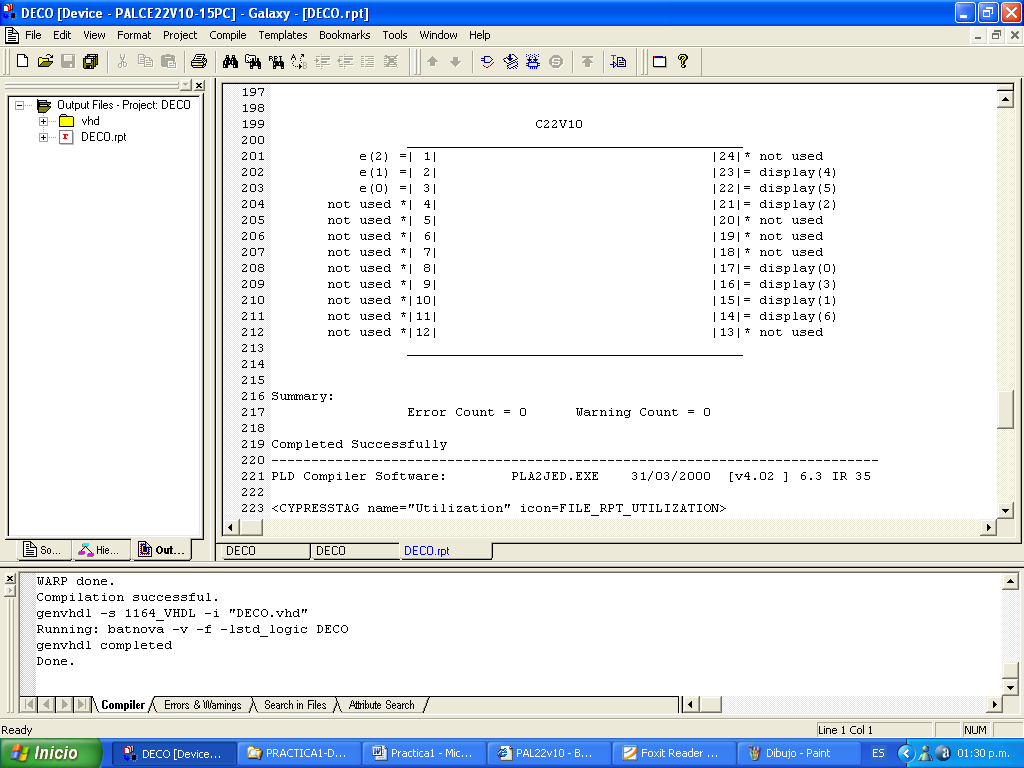


Figura 12. Reporte del diseño.

Information: Output Logic Product Term Utilization.

Node# Output Signal Name Used Max

10 / 121 = 8 %

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| | 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 | | |

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

* + - 1. 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).
      2. 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.

* + - 1. 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.

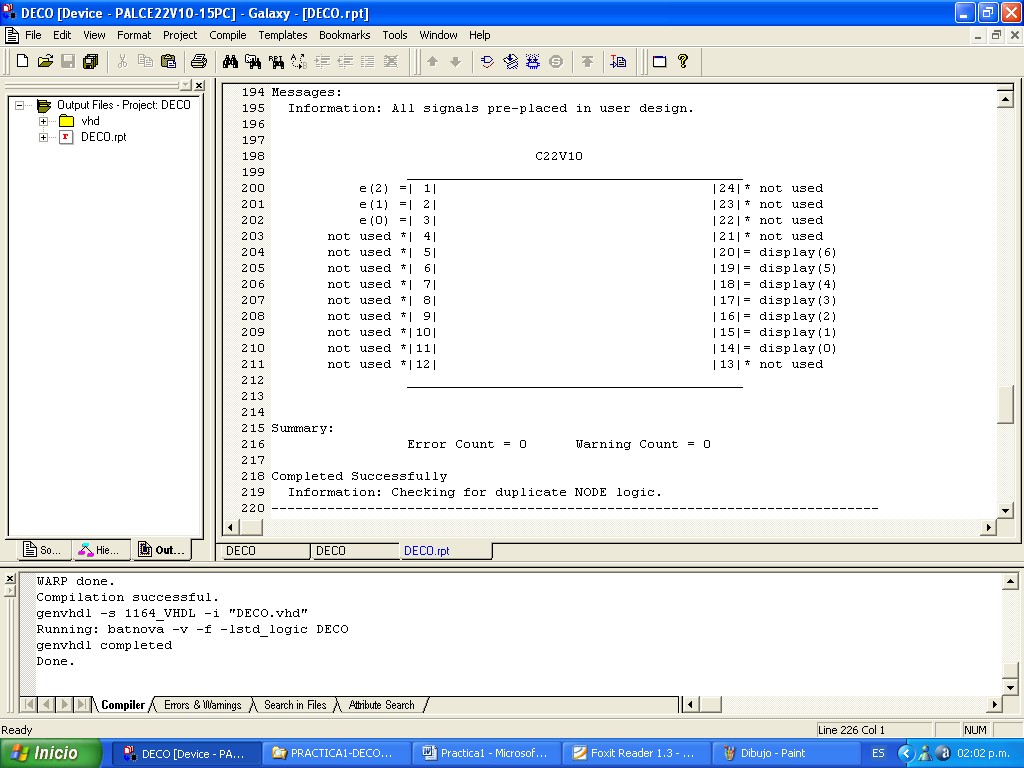


Figura 13. Se muestra claramente la modificación de las terminales de asignación.

* + - 1. **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.

# 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.

# 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.

## Video De Uso de Galaxy

<https://youtu.be/04j7_21WXIw>

# OBSERVACIONES Y CONCLUSIONES

Texto, Carta

Descripción generada automáticamente

# BIBLIOGRAFÍA

* <https://es.wikipedia.org/wiki/Lenguaje_de_descripci%C3%B3n_de_hardware>
* <https://www.dte.us.es/docencia/master/micr/dapa/temas/tema_02/tema2_ldh.pdf>
* <https://xdoc.mx/preview/lenguajes-de-descripcion-de-hardware-5c2d1aa29da5a>
* <https://es.wikipedia.org/wiki/Automatizaci%C3%B3n_de_dise%C3%B1o_electr%C3%B3nico>
* <https://www.digikey.com.mx/es/resources/design-tools/design-tools>
* <http://www.ptolomeo.unam.mx:8080/jspui/bitstream/132.248.52.100/658/8/A8.pdf>
* <https://en.wikipedia.org/wiki/GAL22V10>
* <https://www.ledsemiconductors.com/product-page/gal22v10>
* <https://en.wikipedia.org/wiki/Xilinx_ISE>
* <http://arantxa.ii.uam.es/~cedeps/Tutorial_ISE_Esqu.pdf>
* <https://es.wikipedia.org/wiki/Quartus_II>
* <http://ygelectronicaintegrada.blogspot.com/2008/02/sistema-de-control-utilizando-altera.html>
* <https://www.vesselfinder.com/es/vessels/CYPRESS-GALAXY-IMO-9379973-MMSI-356768000>
* <https://softwaredigitales.blogspot.com/2015/03/cypress-warp-galaxy-ide-63.html>
* <https://es.rs-online.com/web/p/depuradores-y-emuladores-en-circuito/6904093/>