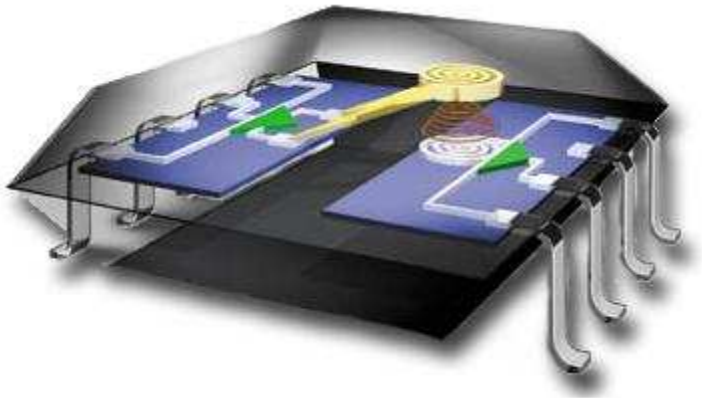
CURSO FPGA (PROGRAMACION DE ARREGLOS DE COMPUERSTAS)



RUBEN DARIO CARDENAS ESPINOSA
DSc. MSc Enginnering Electronic (AIU- USA)
Especialista Gerencia en Finanzas (UAM – Colombia)
Ingeniero Electrónico (UAM – Colombia)
Tecnólogo Profesional en Electrónica y Automatización Industrial (UAM – Colombia)

AGOSTO DE 2009
MANIZALES – CALDAS
REPUBLICA DE COLOMBIA

Tabla de contenido

INTRODUCCIÓN *4*

RESUMEN *5*

MARCO CONCEPTUAL *5*

CIRCUITOS DIGITALES INTEGRADOS **7**

TECNOLOGIAS DE CIRCUITOS INTEGRADOS **7**

DISPOSITIVOS LOGICOS PROGRAMABLES.(PLD) **8**

Programación De Los Dispositivos Lógicos Programables (PLD) **12**

Software De Los Dispositivos PLD: **12**

FPGAs (Field Programmable Gate Array): **12**

Metodología de diseño con FPGA's **13**

CONFIGURACION Y USO SOFTWARE Y HARDWARE FPGA *14*

1 SOFTWARE UTILIZADO **14**

2. HARDWARE EMPLEADO. **20**

3 PROCEDIMIENTO DE CONFIGURACIÓN **22**

Use esta página para seleccionar el dispositivo y otra información del diseño de su proyecto, Seleccionar la herramienta de síntesis y lenguaje, y el simulador **25**

4 PROYECTO # 2 : Compuerta XOR 2 entradas, componentes discretos **54**

5 PROYECTO # 4 CONTADOR BINARIO ASCENDENTE-DECENDENTE **78**

UNIDAD ARITMETICO LOGICA (ALU) CON FPGA *86*

IMPLEMENTACION DE LA ALU EN LA FPGA **89**

SIMULACIÓN DEL DISEÑO. **100**

Creación de tiempos de sincronización **105**

Implementación diseño y verificar sincronismo **107**

Reimplementar diseño y verificación localización de pines **111**

Verificar diseño usando tiempos de simulación **112**

Proceso de diseño descargado en la tarjeta **114**

MONITORES CON FPGA *121*

Clasificación según estándares de monitores **121**

Monitores MDA: “Monochrome Display Adapter 121

Monitor CGA: “Color Graphics Adapter” 122

Monitor EGA: “Enhanced Graphics Adapter”, ... 122

Monitor VGA: “Video Graphics Array” 123

Monitor SVGA: 124

Clasificación según tecnología de monitores 124

Monitores CRT: 125

Pantallas LCD: 126

Pantallas Plasma: 127

¿Qué es la resolución de pantalla? 128

Descripción de los bloques 129

ESTRATEGIA METODOLOGICA 139

Etapa de Análisis 139

Etapa de diseño 139

Etapa de Implementación 139

CONCLUSIONES 141

RECOMENDACIONES 141

BIBLIOGRAFÍA 141

INTRODUCCIÓN

Durante el estudio de la ingeniería electrónica es necesario el diseño y montaje de circuitos digitales para entender su funcionamiento, para lo cual en la universidad se cuenta con un laboratorio que realiza préstamo de elementos electrónicos según los requerimientos del circuito a montar y probar, presentando dificultad en el proceso de montaje por que no hay los suficientes elementos o si se requiere modificar algo en el circuito inicial, demanda mas tiempo y realizar modificaciones en el circuito ubicado sobre el protoboard es difícil , luego surge la necesidad de utilizar un sistema que permita realizar diseño de circuitos digitales y simulación para facilitar el proceso de aprendizaje, esta característica la presenta los FPGA que es un dispositivo lógico programable.

En este trabajo se propone la utilización de una plataforma de hardware reconfigurable a través de un Kit de desarrollo de FIELD PROGRAMMABLE GATE ARRAY (FPGAs) para realizar diseños digitales en lógica secuencial y lógica combinacional.

Como introducción al uso del lenguaje VHDL y el Kit de desarrollo se tiene un curso básico, el cual a través de cuatro proyectos, explica los diferentes pasos para utilizar el software integrado ISE de Xilinx y programar un FPGA incluido en el Kit de desarrollo.

Los diseños digitales con lógica secuencial y combinacional se ve integrada en el desarrollo de un dispensador de productos y en la implementación de una ALU sencilla, igualmente la posibilidad que tiene el FPGA integrado en el kit de desarrollo de usar componentes periféricos se muestran en dos aplicaciones que permiten interactuar con un monitor vga o lcd y un teclado.

Este curso básico de aprendizaje es una guia de aprendizaje para nuevos usuarios del software de programación *Integrated Software Enviroment (ISE)* 8.2i.de XILINX, aquí se demuestran los pasos basicos para el diseño e implementación de circuitos digitales FPGA en el software ISE. 8.2i.y su implementacion en hardware sobre el kit de desarrollo HW-SPAR3E-SK-US de la familia SPARTAN 3E de XILINX.Al terminar el estudio del curso usted podra alcanzar la destreza para crear, simular e implementar sus propios diseños usando el ISE 8.2i y el kit de desarrollo.

El software ISE 8.2i realiza simulación y permite incluir banco de puebas que representan el comportamiento esperado del circuito, ademas el ISE 8.2i genera un archivo el cual se descarga al kit de desarrollo para configurar la FPGA para ver su funcionamiento.

El lenguaje a utilizar es lenguaje para descripción y modelado de circuitos (VHDL) que permite descomponer la estructura principal de diseño en subdiseños e interconectarlos; también permite la especificación de la función de diseño usando formas de lenguaje de programación familiar, igualmente como un resultado facilita hacer pruebas de simulación para hacer correcciones sin costo de hardware prototipo.

RESUMEN

En este trabajo se desarrolla un curso básico y 4 aplicaciones en circuitos digitales, implementando lógica combinacional y secuencial sobre un kit de desarrollo de *Field Programmable Gate Array* (FPGA) para el laboratorio de Electrónica de la Universidad Autónoma de Manizales

El curso básico describe de manera detallada proyectos simples que permite a los usuarios hacer uso del kit de desarrollo, guiándolos para que realicen la introducción y estudio del hardware y software integrado.

Las cuatro aplicaciones están orientadas a mostrar diseños con lógica combinacional y secuencial, además las posibilidades que tiene el kit de desarrollo para manejar dispositivos periféricos.
La primera aplicación esta orientada a la lógica combinacional diseñando una ALU como prototipo de prueba, la segunda aplicación es en lógica secuencial simulando un dispensador. de productos, la tercera aplicación muestra la formación de una cuadricula de colores sobre la pantalla del monitor VGA y la cuarta aplicación integra dispositivos periféricos los cuales son posibles conectar al kit de desarrollo como teclado y monitor VGA.

El Kit de desarrollo elegido es el HW-SPAR3E-SK-US de la familia SPARTAN 3E de XILINX, y el sistema de desarrollo Xilinx ISE & EDK de la misma compañía.

MARCO CONCEPTUAL

En microelectrónica es necesario trabajar con circuitos integrados los cuales usan diferentes tecnologías y escalas de integración entre las cuales están SSI, MSI, LSI, VLSI, Y ULSI., algunos permiten ser programados por el usuario, entre estos están los dispositivos lógicos programables (PLD), los mismos se utilizan en muchas aplicaciones para reemplazar a los circuitos SSI y MSI. Existen cuatro tipos de dispositivos que se clasifican como PLD: estos son PAL, PLA, GAL Y PROM, estos dispositivos tienen diferente grado de complejidad, siendo unos simples como los SPLD y otros complejos como los CPLD, para su programación se requieren un software, un hardware representado en un prototipo , este se programa luego de realizar la síntesis del circuito usando lenguaje VHDL., en este trabajo se utiliza el lenguaje VHDL para programar un prototipo de FPGAs, , luego los siguientes tópicos teóricos aportan para la realización de este proyecto:

CIRCUITOS DIGITALES INTEGRADOS

Un circuito integrado (CI) monolítico es un circuito electrónico construido enteramente sobre un pequeño chip de silicio. Todos los componentes que conforman el circuito como transistores, diodos, resistencias y condensadores son parte integrante de un único Chip, cuyos terminales se conectan a los pines del encapsulado para permitir las conexiones de entrada y salida al mundo exterior. Otra técnica de encapsulado de CI es la tecnología de montaje superficial (SMT) el cual es un método más moderno que permite ahorrar espacio, alternativo al montaje de inserción. Los pines de los encapsulados de montaje superficial se sueldan directamente a las pistas de una cara de la tarjeta, dejando la otra cara libre para añadir circuitos adicionales obteniendo un circuito más pequeño

TECNOLOGIAS DE CIRCUITOS INTEGRADOS

Los tipos de transistores con los que se implementan los circuitos integrados pueden ser transistores bipolares TTL (Transistor-transistor Logic) o MOSFET(metal-oxide semiconductor Field-Effect Transitor). La mayor parte de las tecnologías de circuitos que utilizan MOSFET son de tipo CMOST (complementary MOST) o NMOST(n-channel MOST). Por ejemplo los microprocesadores utilizan la tecnología CMOST.

Generalmente los circuitos de escala de integración SSI y MSI están disponibles en TTL y CMOST, y los circuitos de escalas de integración LSI, VLSI y ULSI se implementan normalmente con CMOST o NMOST, ya que estas tecnologías requieren una menor superficie de chip y consume menos potencia. Acontinuación se aclaran las escalas de integración:

- **SSI** (small-scale integration): Se refiere a circuitos que integran aproximadamente 10 compuertas por chip, surgieron en los años 60´s
- **MSI** (medium-scale integration): Se refiere a circuitos que integran aproximadamente entre100-1000 compuertas por chip, surgieron en los años 70´s
- **LSI**(large-scale integration): Se refiere a circuitos que integran aproximadamente entre1000-10000 compuertas por chip, surgieron en los años 80´s
- **VLSI** (very large-scale integration): Se refiere a circuitos que integran aproximadamente entre10000-100000 compuertas por chip, surgieron en los años 90´s
- **ULSI** (ultralarge scale integration):Se refiere a circuitos que integran aproximadamente entre1M-10M compuertas por chip,

DISPOSITIVOS LOGICOS PROGRAMABLES.(PLD)

Los dispositivos lógicos programables (PLD) se utilizan en aplicaciones para reemplazar a los circuitos SSI y MSI, ya que ahorran espacio y reducen el número y el costo de los dispositivos en un determinado diseño. Un **PLD** puede usarse como una caja negra que contiene compuertas lógicas y llaves programables, tal como se observa en la *figura 3*

Existen cuatro tipos de dispositivos que se clasifican como PLD:

PLA: (Programmable Logic Array): Consiste una matriz de compuertas AND, OR e inversores conectados por medio de una matriz de interruptores programables. cuya estructura se observa en la *figura 4*

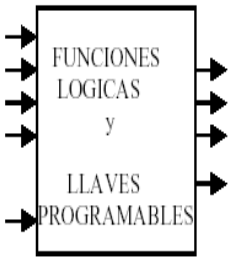


figura 3

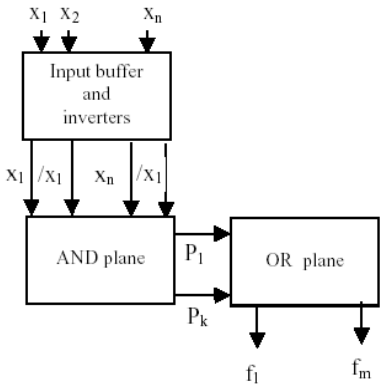


figura 4

Basada en la idea de que una función lógica puede ser realizada como una Suma De Productos**,** una **PLA**, comprende un arreglo (array) de compuertas **AND** que alimentan un arreglo de compuertas **OR** – Un esquema del circuito se muestra en la *figura 5.*

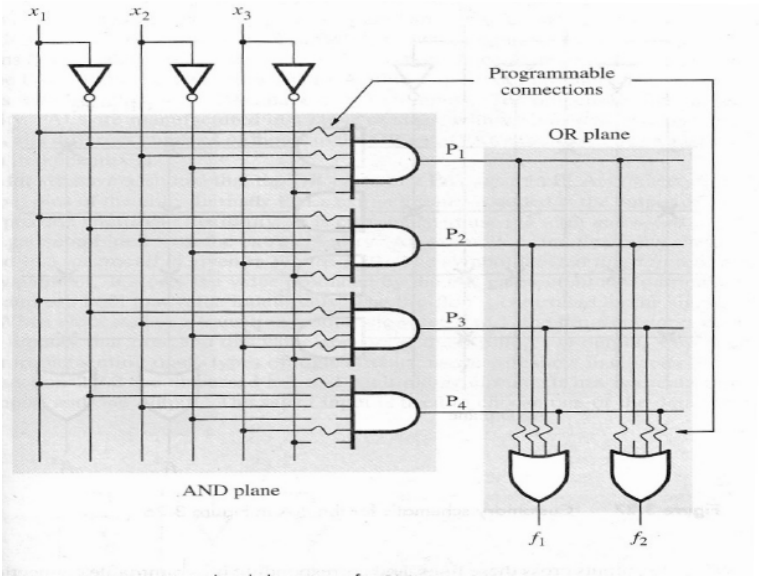


figura 5

En esta figura se observa que cada entrada a la compuerta **AND** se puede programar (conectar o nó). El circuito está diseñado para que una entrada no conectada no afecte el funcionamiento de las compuertas. Una **PLA** típica tiene 16 entradas y 8 salidas. El esquema de la **PLA** dado en la *figura 5* no es aconsejable cuando se trabaja con mayor número de entradas/salidas. En su lugar utilizamos el esquema dado en la *figura 6*.

PAL (Programmable Array Logic): en estos dispositivos la matriz de compuertas OR es fija, además, las salidas sé realimentan a la entrada de la matriz de compuertas AND permitiendo funciones lógicas multinivel. La **PAL** ofrece menos flexibilidad que la **PLA** debido a que tiene un arreglo fijo (**OR**). La **PLA** permite hasta 4 terminos producto por compuerta **OR**, mientras que la **PAL** solo 2. Ello se observa en *la figura 6*

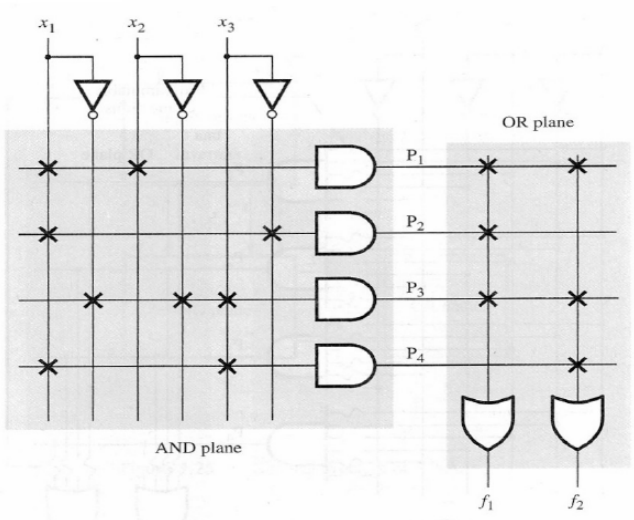


figura 6

Como se ha visto en las **PLAs** ambos arreglos (planos **AND y OR**) son programables. Esto tiene dos dificultades para el fabricante. En primer lugar es de fabricación compleja por la cantidad de llaves, por otro lado son poco veloces. Este problema los llevo a mejorar el diseño y desarrollaron un dispositivo de arreglo **AND** programable y **OR** fijo. Esta estructura se denomina Programmable Array Logic. Ello se observa en la *figura 6.*
En muchas **PAL** se agregan circuitos extras a cada salida **OR**, para proveer una mayor flexibilidad. Este circuito adicional a la salida **OR** se la denomina **MACROCELL**, se muestra en la *figura 7.*

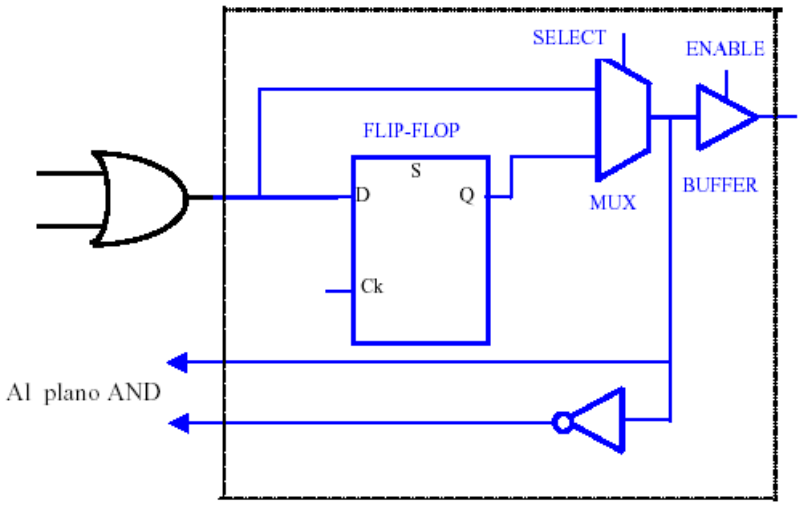


figura 7

SPLD (Simple Programmable Logic Device): en estos dispositivos, se les anexan multiplexores a cada una de las salidas combinacionales normales de los PAL y las salidas de los multiplexores se convierten en las salidas del dispositivo.

CPLD (Complex Programmable Logic Device):

Las **PAL y PLA** son útiles para implementar una gran variedad de pequeños/medianos circuitos digitales (Entradas más salidas que no superen el número de 32).
Cuando se requiere mayor cantidad de entradas/salidas se recurre a un **CPLD** (Complex Programmable Logic Devices).

Un **CPLD** comprende múltiples bloques en un único chip con recursos de cableado interno para conectar los bloques. Cada bloque es similar a una **PLA o PAL**, por lo que en el esquema de la *figura 8* se refiere a cada block como **PAL – like block.**

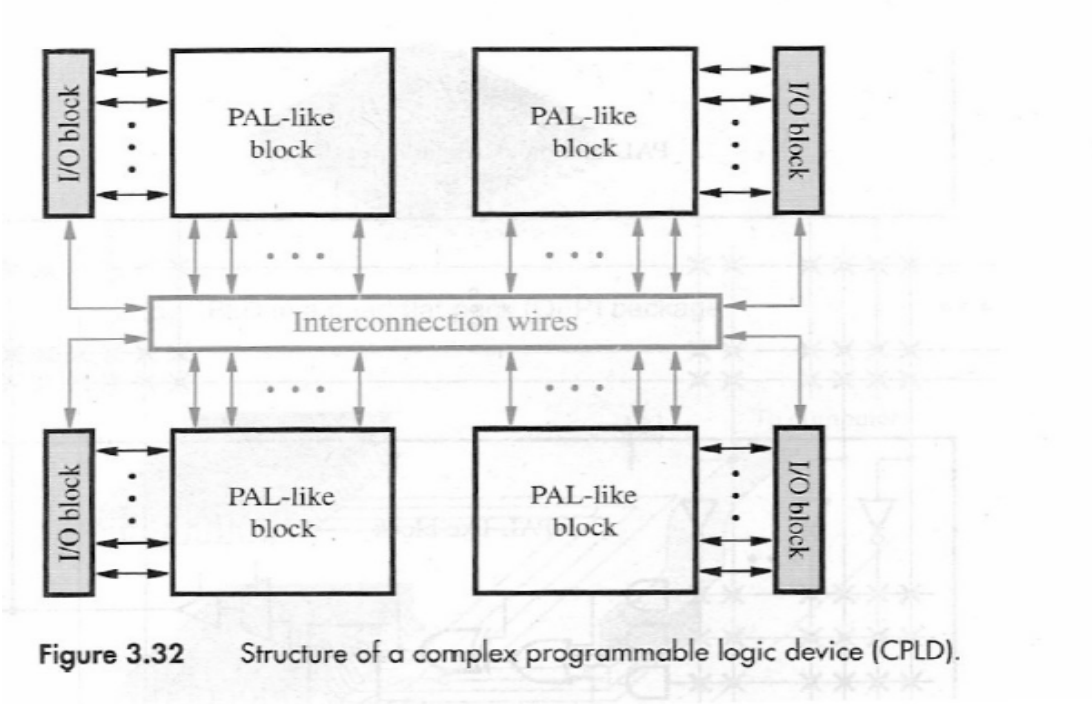


Figure 3.32 Structure of a complex programmable logic device (CPLD).

figura 8

estos dispositivos combinan una gran cantidad de SPLD, como el XC95108 de Xilinx que contiene seis bloques de función configurable, cada uno equivale a 18 SPLD con 36 entradas y 18 salidas.En la *figura 9* se observa un ejemplo de interconexionado

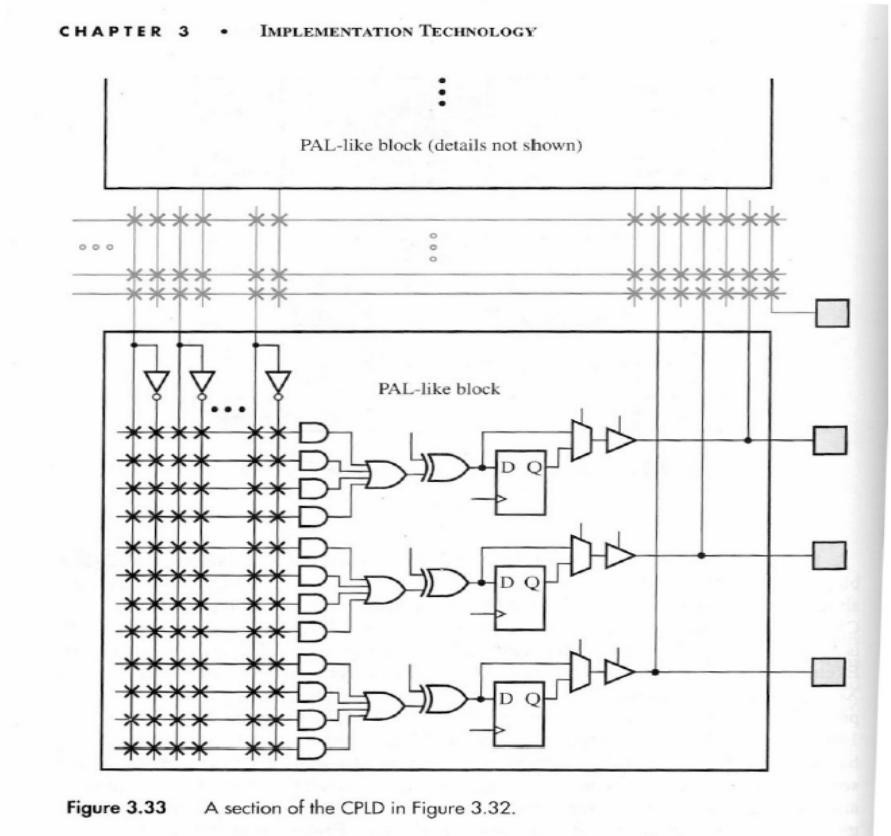


Figure 3.33 A section of the CPLD in Figure 3.32.

figura 9

El **PAL**-like block incluye 3 **MACROCELLS** (**CPLD** reales tienen cerca de 16 **MACROCELLS** en una **PAL**-like block), consistente en una compuerta **OR** de 4 entradas (real de 5 a 20 entradas) . Cada salida de la compuerta **OR** se conecta a una entrada **XOR**. La **MACROCELL** incluye **FF, Mux y Buffer** de salida con tercer estado. Los cables de interconexión (interconexión wires) contienen llaves programables que se utilizan para la conexión de los **PAL**-like blocks. El tamaño de las **CPLD** va de 2 a 100 **PAL**-like block.

otros tipos .de PLDs son las memorias

- ROM (read-only memory)
- PROM (programable ROM)
- EPROM (eléctricamente programables)
- EEPROM (eléctricamente programables o borrables)

Programación De Los Dispositivos Lógicos Programables (PLD)

Para realizar la programación de un PLD se requiere

1. Software de programación (compilador lógico).
2. Un computador personal que cumpla los requisitos de software.
3. Un programador controlable por software, el cual es un dispositivo hardware que acepta datos de programación procedentes de la computadora e implementa un diseño lógico específico en el PLD, esté se inserta en un zócalo en el programador.

Software De Los Dispositivos PLD:

Existen diversos paquetes de software para implementar los diseños lógicos basados en los PLD, es así como VHDL, ABEL, y CUPL, son lenguajes de descripción de hardware (HDL, Hardware Descripción Language) más comúnmente utilizados. Puesto que estos lenguajes son similares y producen el mismo resultado en términos de programación de un PLD, a menudo, su utilización es una cuestión de preferencias y disponibilidad.

FPGAs (Field Programmable Gate Array):

Introducidas por Xilinx en 1985. Son los dispositivos programables por el usuario de aplicación más general. Estos chips tienen unos componentes básicos que se pueden unir según las necesidades de diseño, Esta configuración se encuentra almacenada en una memoria ram interna, y se carga desde el exterior del chip. De igual forma que en los microcontroladores se carga software, en las FPGA's se carga la configuración que determina en qué circuito se va a convertir.

Una FPGA's es un chip que según cómo se configure, puede realizar cualquier circuito digital. una FPGA más grande, con más recursos internos, alcanza a implementar diseños más complejos. Pero al final se tiene una manera de poder crear diseños digitales sin tener que utilizar componentes externos. Y lo interesante es que una vez configurada la FPGA, lo que tenemos en su interior es hardware.

Metodología de diseño con FPGA's

La metodología de diseño es similar a la cualquier sistema digital, salvo que al final se obtiene un archivo ejecutable que se descarga a la FPGA para que se reconfigure, implementando así el diseño esperado.
Primero hay que tener una *descripción del circuito* a realizar. Tradicionalmente en las ingenierías se realizan *planos* o *esquemas* para esta descripción, de forma similar a como un arquitecto diseña un edificio. Sin embargo es posible realizar una descripción del hardware utilizando algún *lenguaje de descripción de hardware*, como VHDL o Verilog Con esta descripción se pueden realizar simulaciones del circuito, para comprobar que lo diseñado trabaja correctamente de lo contrario se volverá a modificar la descripción (esquemas o programa) hasta que la simulación sea satisfactoria. Hasta aquí sólo se a utilizado el computador y no se tocado hardware. Sin embargo en el caso del software, la propia simulación es la ejecución del programa. Se observa directamente el resultado del programa y se modifican el código fuente hasta que se eliminen los errores.

En el caso del hardware hay que construir el circuito. Y aquí es donde vienen las FPGA's para hacerlo. A partir de la especificación hardware y utilizando un compilador especial, obtenemos un archivo binario, llamado *bitstream* que contiene toda la información necesaria para configurar la FPGA. Este archivo, que es el equivalente a un programa ejecutable en el caso del software, es el que hay que cargar en la FPGA. Se carga este archivo en la FPGA y listo. Ya se tiene el hardware que queríamos situado en el interior de un chip. No se ha tenido que soldar, ni comprar componentes, ni perder tiempo haciendo un prototipo. Ahora los cambios en el diseño se pueden hacer igual de rápidos que en el caso de software. Sólo hay que cambiar la especificación del diseño, volver a compilar y reconfigurar la FPGA con el nuevo *bitstream* generado.

VHDL(Hardware Description Lenguaje): lenguaje para descripción y modelado de circuitos que permite descomponer la estructura principal de diseño en subdiseños e interconectarlos; también permite la especificación de la función de diseño usando formas de lenguaje de programación familiar, igualmente como un resultado facilita hacer pruebas de simulación para hacer correcciones sin costo de hardware prototipo.

La especificación de un circuito hasta ahora sólo se hacía de una manera: utilizando esquemas gráficos, en los que cada símbolo representa un componente o elemento lógico: multiplexores, puertas lógicas,n etc. Existe otra manera de describir un circuito: utilizando los llamados *lenguajes de descripción hardware*. Existen varios: VHDL, Verilog, Handle C, JBits. La

ventaja de estos lenguajes es que además de permitir describir el circuito, permiten definir bancos de pruebas (testbench), que son muy útiles para la simulación y la depuración.

Prototipo los prototipos son una técnica de uso común en algunas ingenierías cuando los desarrollos son complejos, laboriosos y caros, o no están completamente especificados; en los dominios de aplicación de los sistemas digitales, un prototipo permite evaluar la viabilidad de diferentes soluciones propuestas para un problema. Pueden plantearse subsistemas formados por módulos hardware específicos, o por módulos programables que se adapten mediante software a la funcionalidad de las especificaciones.

Matriz Programable: es una red de conductores distribuidos en filas y columnas con un fusible en cada punto de intersección, las matrices pueden ser fijas o programables

Sintesis: Proceso software por el que se convierte una descripción de circuito en un archivo JEDEC el cual es un archivo software estándar generado a partir de un software de compilación que se emplea en un dispositivo de programación para implementar un diseño lógico en un PLD, también se denomina mapa de fusibles o mapa de celdas.

CONFIGURACION Y USO SOFTWARE Y HARDWARE FPGA

1 SOFTWARE UTILIZADO

El software de diseño programación y simulación ***INTEGRATED SOFTWARE ENVIROMENT (ISE)*** **8.2I** es una herramienta útil para estudiantes y profesionales que desean acelerar y mejorar sus habilidades para el desarrollo de aplicaciones digitales empleando un entorno de programación gráfico o usando lenguaje VHDL y realizar la simulación de su funcionamiento sin el riesgo de ocasionar daños a los circuitos.

1.1 Breve historia del lenguaje VHDL.

VHDL es el Idioma de Descripción de Hardware, que se desarrolló en los años de 1980s como un proyecto de investigación de circuito integrado de gran velocidad el cual fue consolidado por el Departamento Americano de Defensa. Un equipo de ingenieros de tres compañías - IBM, Texas Instruments e Intermetrics - fue contratado por el Departamento de Defensa para completar la especificación y aplicación de un nuevo idioma basado en método de descripción de diseños digitales ideado por ellos. La primera versión públicamente disponible de VHDL, versión 7.2, se libero en 1985. En 1986, el Instituto de Eléctricidad y Electrónica Diseñada, Inc. (IEEE) se presentó con una propuesta para estandarizar el idioma que hizo en 1987, después de que las mejoras sustanciales y modificaciones fueron hechas por un equipo de gobierno y representantes académicos. La norma resultante, IEEE 1076-1987, es la base virtual

para cada simulación y el producto de la síntesis vendido hoy. Una nueva versión mejoro el lenguaje, la IEEE 1076-1993, se lanzo en 1994.

Aunque IEEE Standard 1076 define el lenguaje de VHDL completo, hay aspectos del lenguaje que hacen difícil la descripción completamente portátil del diseño (descripciones que pueden simularse usando las herramientas de diferentes vendedores). El problema proviene del hecho que VHDL soporta muchos tipos de datos abstractos, pero esto no se dirige al problema simple de caracterizar señales diferentes o las condiciones de la simulación normalmente usadas como los desconocidos y alta-impedancia.

Poco después IEEE 1076-1987 fue adoptada, las compañías de simuladores empezaron reforzando VHDL con los nuevos tipos no estandarizados y permitirles a sus clientes simular los circuitos electrónicos complejos con precisión. Esto causó problemas porque las descripciones de un diseño entraron en un simulador donde a menudo eran incompatibles con otros ambientes de simulación. VHDL estaba volviéndose no estandarizable rápidamente creando un problema.Otra norma se desarrolló por un comité de IEEE. Esta norma, numerada 1164, define un paquete estandar (un lenguaje VHDL que permite coleccionar las declaraciones normalmente usadas en una biblioteca externa) conteniendo las definiciones para un tipo de los datos normalizados. Este tipo de datos normalizado se llama el *std_logic*, y el paquete 1164 IEEE es frecuentemente llamado el paquete de la Lógica estandar.
Las normas IEEE 1076-1987 e IEEE 1164 completan el estándar VHDL siendo el más usado hoy.

1.2 REQUERIMIENTOS DEL SISTEMA

La instalación completa del software ISE 8.2i requiere:

- aproximadamente 1 GB de espacio de almacenamiento en disco duro.
- Memoria RAM 512 M minimo , ideal 1 GB de memoria Ram
- Windous 2000,.Windous XP/PRO, Linux, o solaris
- Procesador 1.7 Ghz.

1.3 INSTALACIÓN SOFTWARE ISE 8.2i

Inserte el DVD marcado como Xilinx 8.2i Design Tools Evaluation en la unidad de DVD. Este DVD contiene un comando autoejecutable que abrira un menu usando un navegador WEB.como muestra la Figura 1.1. Si el menu del DVD no habre automáticamente, use un navegador WEB para abrir el archivo HTML: index.htm.

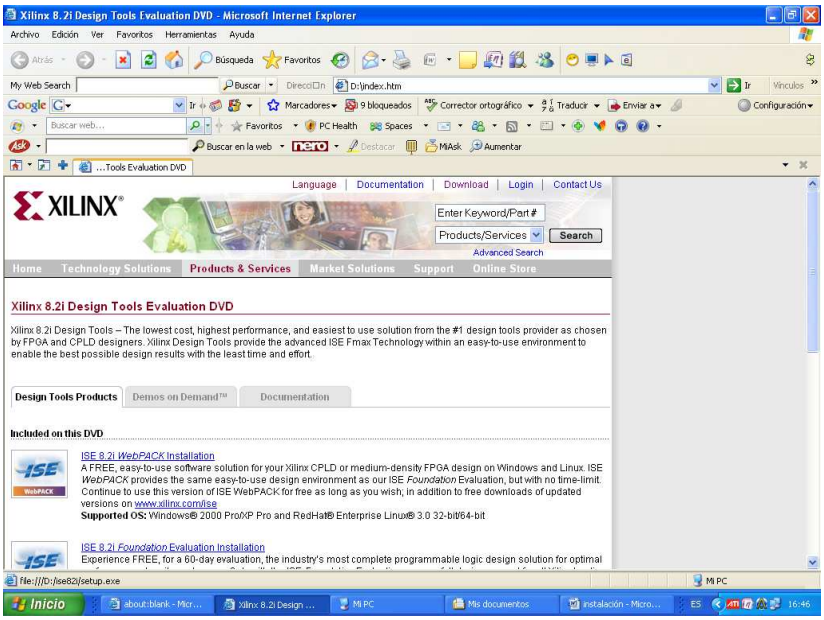


Figura 1.1 Pantallazo inicial en proceso de instalación

Haga click sobre el link de **Desing tools products** llamado **ISE 8.2i WebPACK** y de click en el boton ejecutar como muestra la Figura 1.2

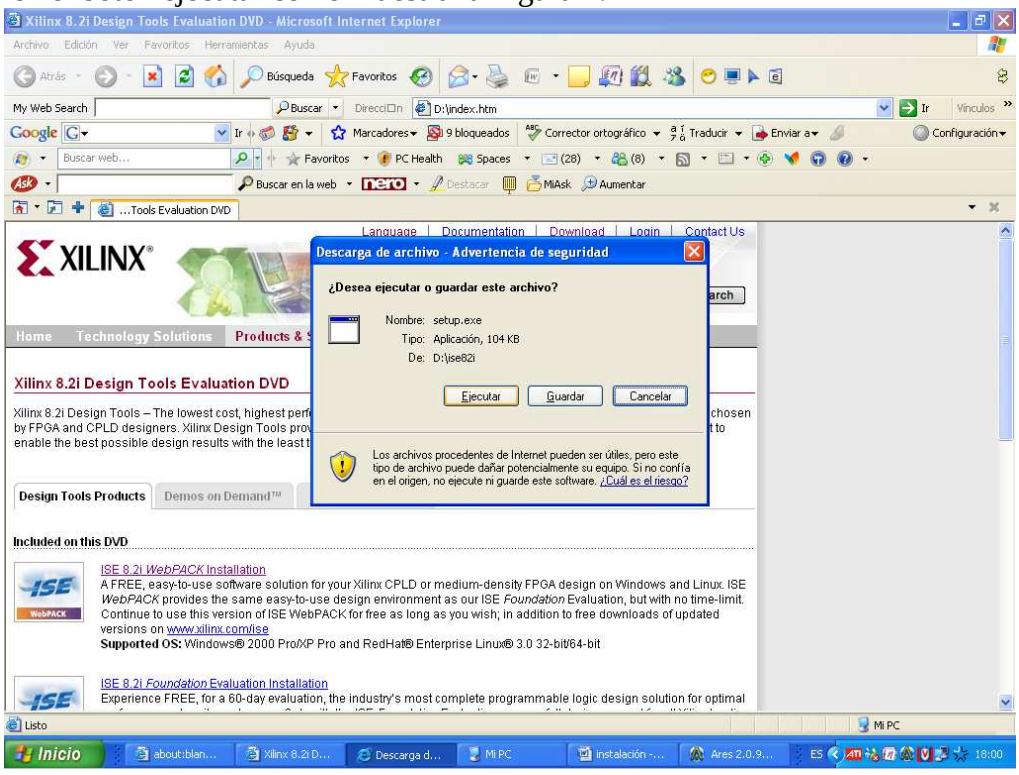


Figura 1.2 Inicio instalacion de **ISE 8.2i WebPACK**

Se habre una ventana para registrar el producto, haga click en next. Ver Figura 1.3

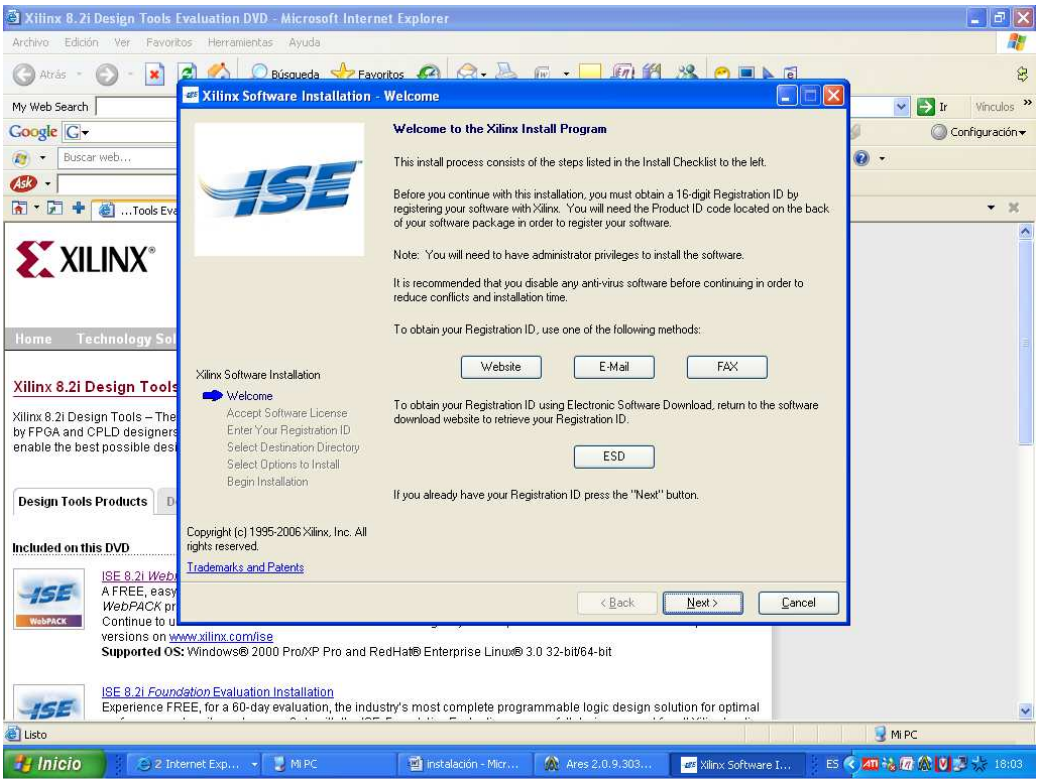


Figura 1.3 Obtener serial de registro

En la siguiente pantalla pide el serial de registro como indica la Figura 1.4, para lo cual digitamos: XXXX XXXX XXXX XXXX

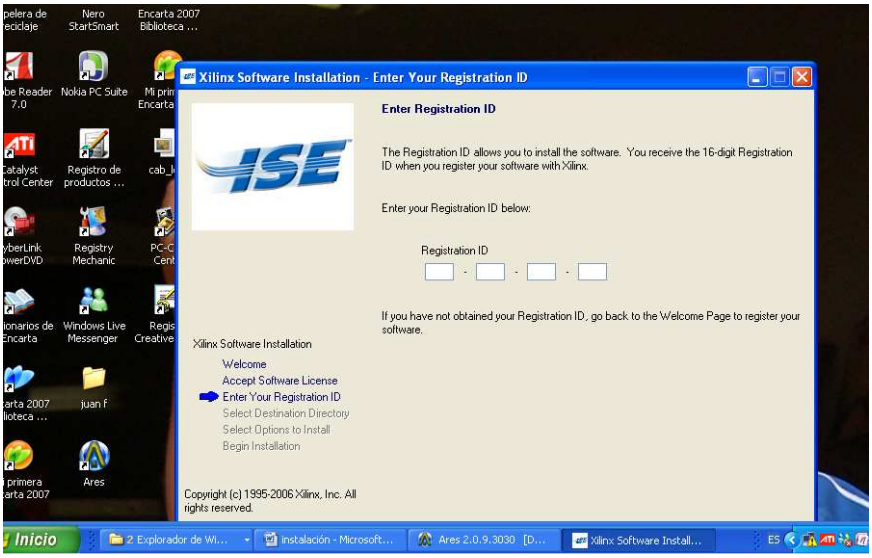


Figura 1.4 Digitar serial de registro

Acontinuación el instalador solicita establecer la ruta de la carpeta en la cual se instalara el Ise 8.2i WebPACK, por defecto lo realizar en el directorio raiz.Ver Figura 1.5

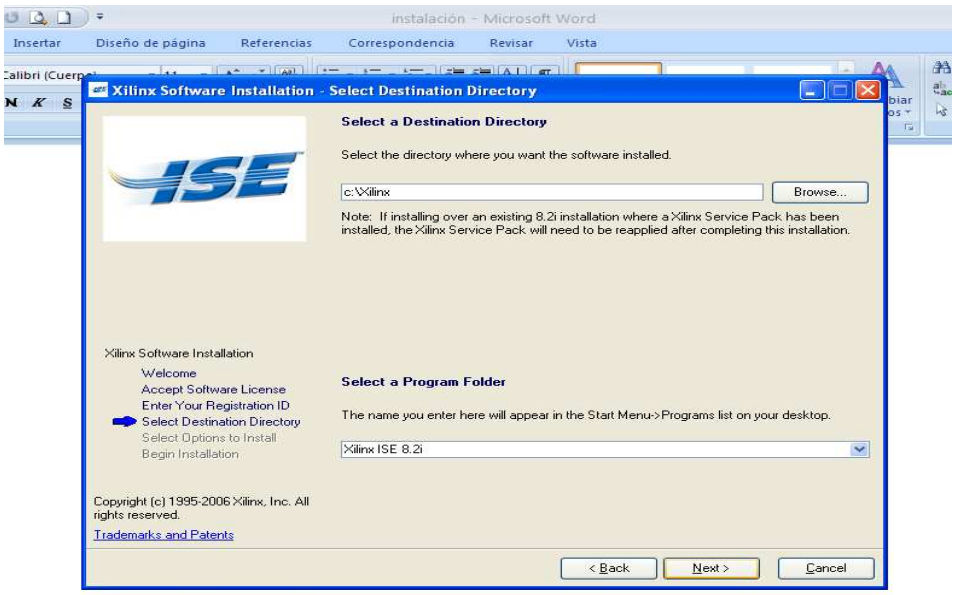


Figura 1.5 Determinar ruta de instalación programa

La ventana de selección opciones a instalar, muestra los modulos disponibles para instalar, por defecto todos aparecen seleccionados, haga click en boton **Next.** ver figura 1.6

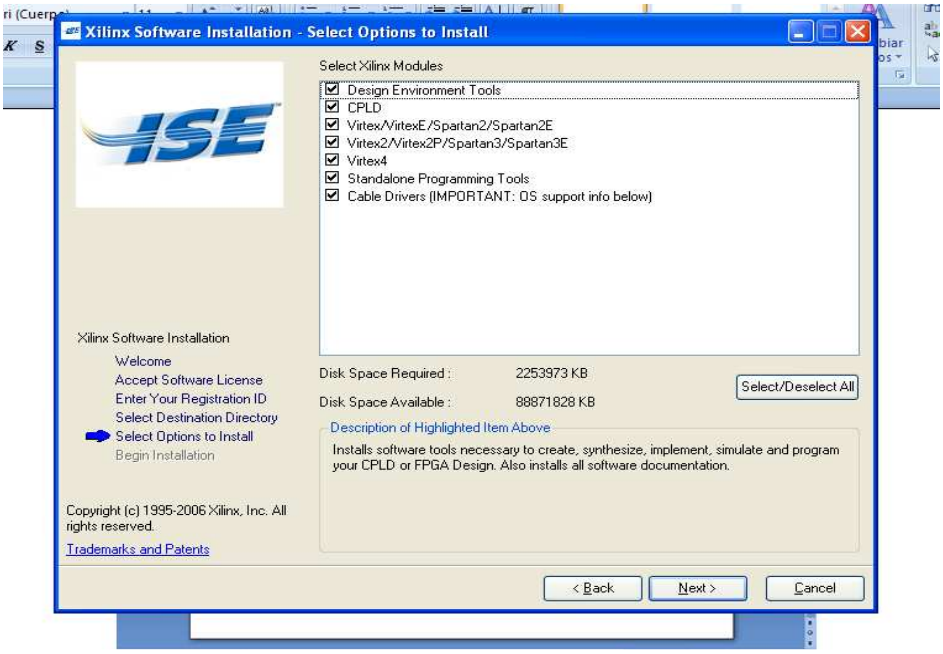


Figura 1.6 Selección modulos a instalar

En la ventana Update Enviroment las opciones aparecen preseleccionadas haga click en Next. Ver Figura 1.7.

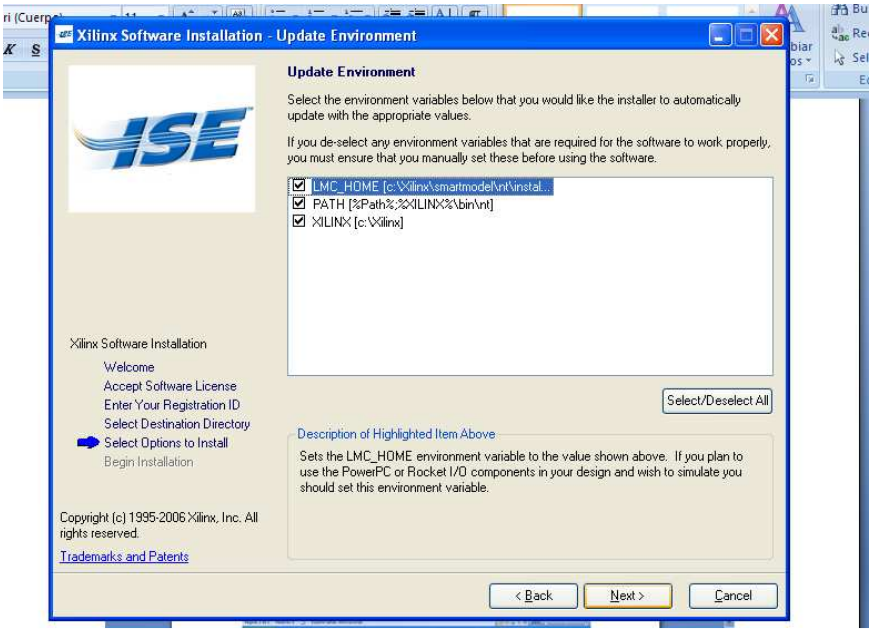


Figura 1.7 opciones de actualización ISE 8.2i

El resumen de la configuración de instalacion se muestra como en la Figura 1.8, haga click en **Install**.Este proceso toma 10 minutos aproximadamente.

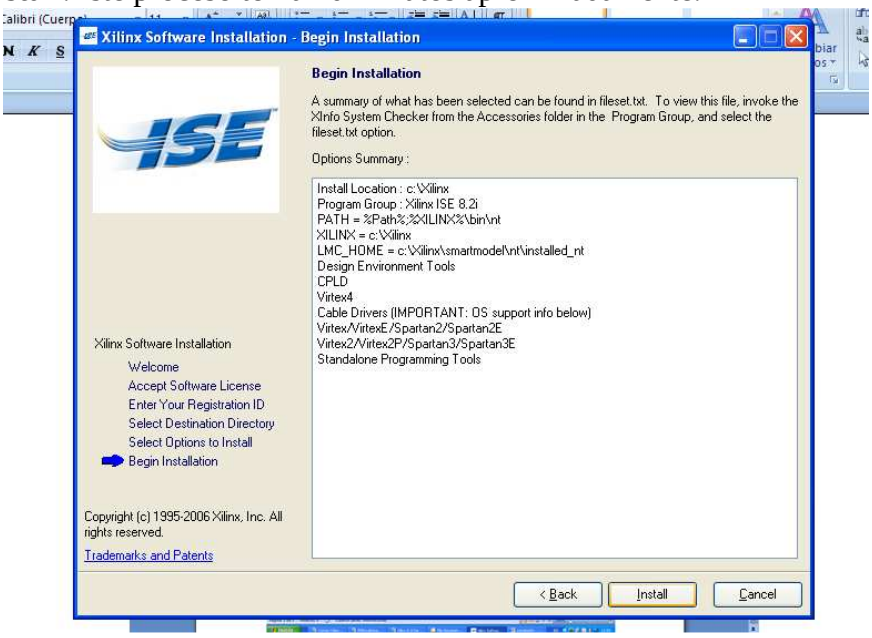


Figura 1.8 Iniciar proceso de Instalación

El proceso de instalación esta en curso como indica la figura 1.9

Figura 1.9 Instalación en curso

Diferentes pantallazos se muestran en pantalla hasta que aparece el mensaje de instalación completa Ver figura 1.10

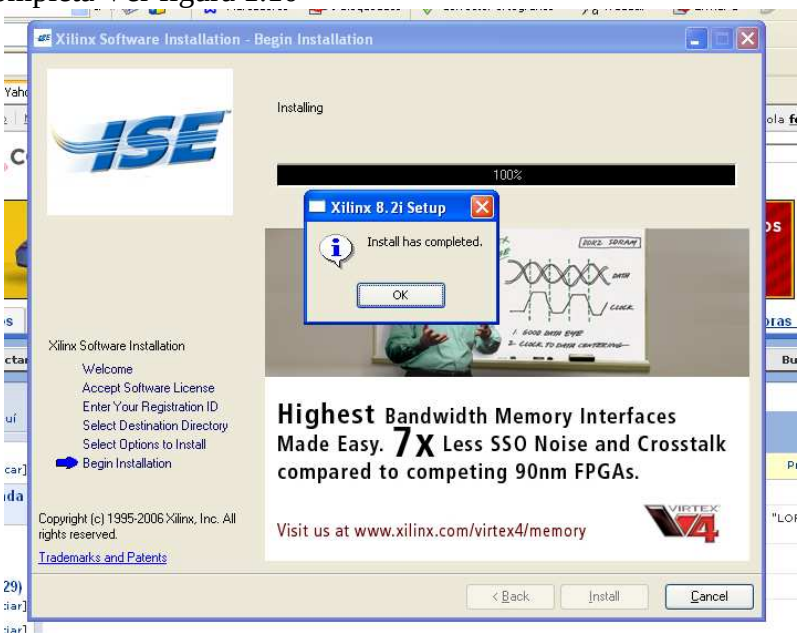


Figura 1.10 Instalación finalizada

Reinicie el computador y retire el DVD, en el escritorio aparecera el icono de ISE 8.2i. Figura 1.11

Figura 1.11 Icono de ISE 8.2i

2. HARDWARE EMPLEADO.

Se emplea como elemento base un kit de desarrollo FPGA comercial referencia HW-SPAR3E-SK-US de la familia SPARTAN 3E de XILINX. Mostrado en la Figura 2.1

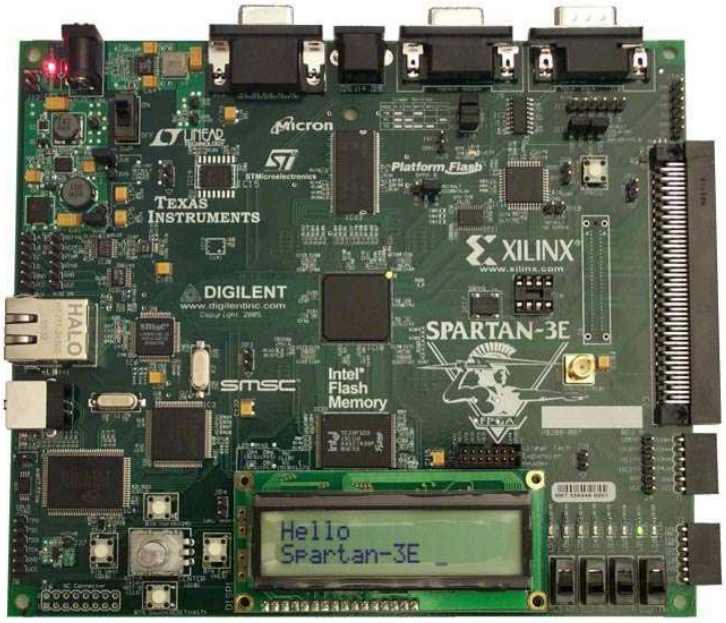


Figura 2.1 HW-SPAR3E-SK-US de la familia SPARTAN 3E de XILINX

Este kit de desarrollo posee

1 FPGA XC3S500E de la familia Spartan 3E, con mas de 232 pines de uso como I/O.Mas de 10000 celdas logicas.
1 Memoria PROM plataforma flash configurable 4 Mbit.
1 CoolRunner CPLD Ref. XC2C64A con 64 macro celdas
1 DDR SDRAM DE 64 Mbyte, 100 MHz
1 memoria flash 16 MByte (128 Mbit) para almacenar configuración del FPGA o el codigo del microprocesador
1 SPI serial flash de16 Mbits (STMicro), permite almacenar configuración del FPGA o el codigo del microprocesador
1 display LCD de 2-lineas, 16-characteres
1 puerto PS/2para mouse o teclado
1 puerto para monitor VGA
1 puerto 10/100 Ethernet PHY (equire Ethernet MAC in FPGA)
2 puertos de 9 pines RS-232 (estilo DTE- y DCE)
1 puerto USB para FPGA/CPLD download/debug interface
1 oscilador de 50 MHz.
1 memoria EEPROM para almacenar archivo de configuración
1 conector de expansión FX2 · Hirose
3 conectores de expansión 6 pines diligent
4 salidas para conversor digital-analogo DAC base SPI-
2 entradas para conversor analogo a digital ADC base SPI-con preamplificador de ganancia programable.
1 encoder rotacional con boton pulsador central
8 salidas discretas en LED´s
4 · swiches deslizables
4 swiches pulsadores
1 entrada para reloj · SMA
1 socket DIP · 8-pin para reloj oscilador auxiliary

3 PROCEDIMIENTO DE CONFIGURACIÓN

El aprendizaje para el manejo del kit de desarrollo en la Ingeniería Electrónica necesita de una constante realización práctica de ejercicios, tanto de simulación como de implementación. Por ello, este curso describe con ejemplos sencillos como utilizar el kit de desarrollo, empleando el dispositivo FPGA de Xilinx y el simulador ISE 8.2i.

3.1 Arranque del programa:

Para iniciar el Integrated Software Enviroment (ISE) dar:

Inicio → **Programs** → **Xilinx ISE 8.2i** → **Project Navigator,** se mostratrara un pantallazo como la Figura 3.1

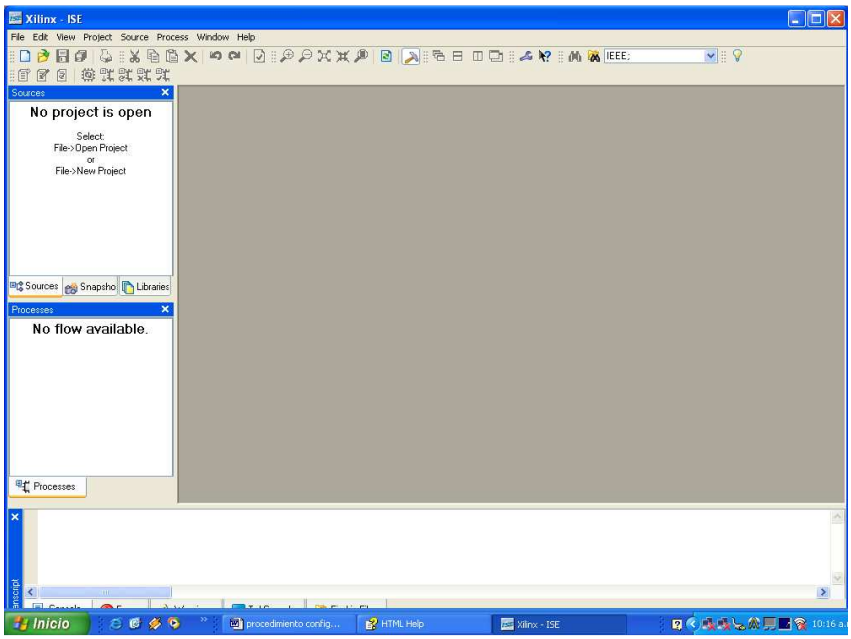


Figura 3.1. Pantallazo inicial software ISE 8.2i

En el pantallazo inicial del programa se observan cuatro (4) ventanas como muestra la Figura 3.2

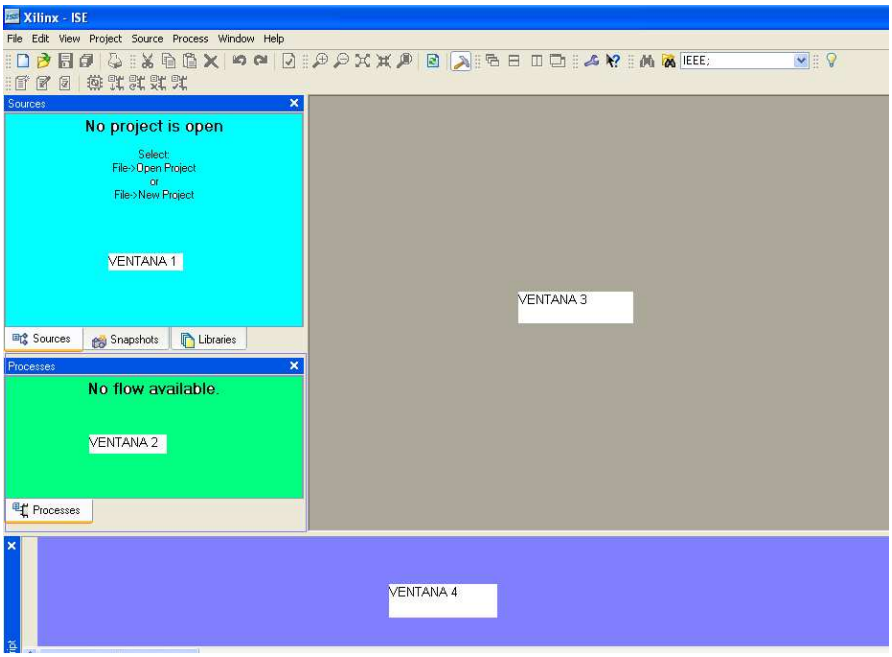


Figura 3.2. Descripción pantalla inicial software ISE 8.2i

La ventana 1 llamada **Sources** muestra los diferentes archivos fuente adicionados al proyecto
La ventana 2 llamada **Processes** muestra los diferentes procesos que se pueden realizar al archivo fuente seleccionado en la ventana 1.
La ventana 3 es el area de trabajo para mostrar y/o editar los diferentes archivos fuente seleccionados en la ventana 1.
La ventana 4 llamada **Transcript** muestra mensajes de estado de los diferentes procesos ejecutados durante el diseño, es asi como aquí aparecen las advertencias, los errores detectados en la revisión de sintaxis, compilación y simulación de un diseño electronico.

La forma corta para iniciar el Integrated Software Enviroment (ISE) es dar doble click en el icono del programa ISE mostrado en la Figura 3.3, el cual esta ubicado en el escritorio.

Figura 3.3 Icono de ISE 8.2i

3.2 Crear proyecto # 1 : Compuerta AND 4 entradas

Para crear el nuevo proyecto el cual llamaremos ***compuertand4.*** procedemos así:

A. Seleccione **File > New Project**. Como lo indica la Figura 3.4.

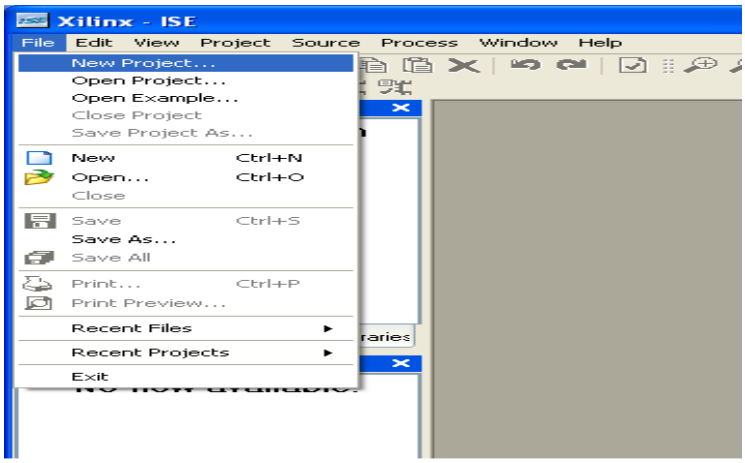


Figura 3.4 Creación de proyecto ***compuertand4.***

La página de creación de nuevo proyecto aparece, aquí damos la información general del proyecto así:

Project Name: Especifique el nombre del proyecto *compuertand4.*

Project Location: Especifique la localización del proyecto.Por defecto el nombre de la carpeta es el mismo dado en el campo de nombre del proyecto

Para cambiar la ruta del directorio haga click en el boton ubicado al lado del campo Project Location y de la ruta deseada.

NOTA: Cada proyecto debe tener su propia carpeta. Si multiples proyectos quedan en la misma carpeta pueden ocurrir conflictos.

Top-Level Source Type

Especifique el top-level source type. Selecione HDL de la lista, mas adelante se explicaran las otras opciones Schematic, EDIF or NGC/NGO.

La Figura 3.5 muestra la pagina configurada

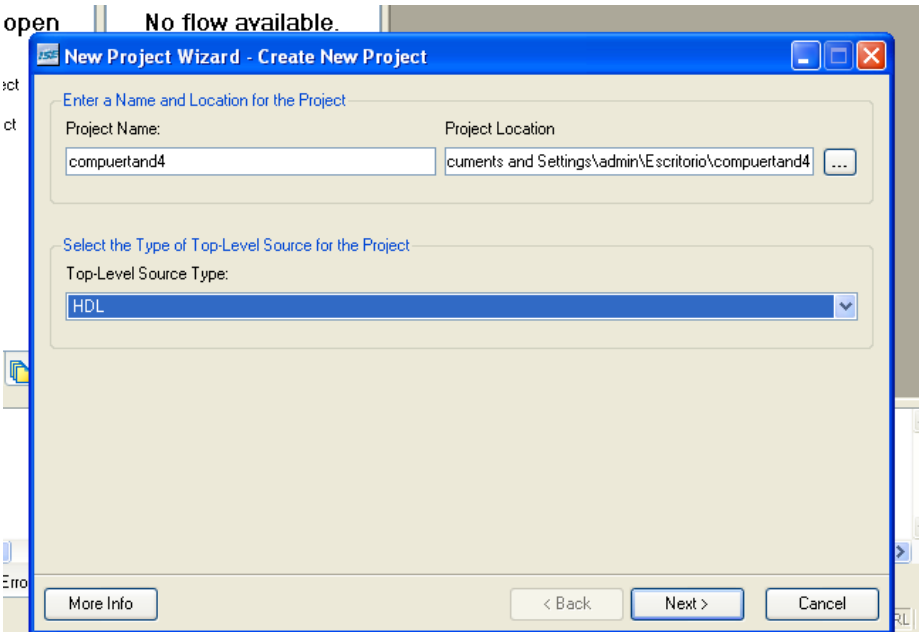


Figura 3.5 Pagina configuración nuevo proyecto

En la configuración de los parámetros en un nuevo proyecto puede obtener ayuda para las diferentes ventanas dando ckick en el boton de **More Info**

B. Click Next para moverse a la página de propiedades del dispositivo.

Use esta página para seleccionar el dispositivo y otra información del diseño de su proyecto, Seleccionar la herramienta de síntesis y lenguaje, y el simulador

Product Category Específica la categoria del producto según su aplicación. Esta selección filtra las familias de dispositivos y dispositivos disponibles en los dos siguientes campos

Family Específica la familia de dispositivos, o arquitectura Xilinx, dentro del cual se implementara su diseño

Device Especifique el dispositivo en el cual implementara su poryecto

Package Específica la presentación del dispositivo con respecto al número de entradas y salidas.

Speed Específica la velocidad del dispositivo seleccionado.

Top-Level Source Type Especifica el tipo de fuente para el nivel de diseño.

Synthesis Tool Especifica la herramienta de sintesis y lenguaje de sintesis usado para su diseño en este caso Xilinx Synthesis Technology (XST) es suministrado con ISE.Hay otras herramientas de sintesis como synplify® Synplify®,LeonardoSpectrum.Precision® que no estan disponible en este programa

Simulator Especifica la herramienta usada para simulacion y el lenguaje usado para generar el archivo netlists de la simulación ISE Simulator esta integrado en ISE 8.2i.

Enable Enhanced Design Summary Selecionando esta opción usted puede ver información adicional en el resumen de diseño.El número acumulado de errores y notas de advertencia para cada paso de implementación deben ser mostrados así como los mensajes de errores, notas de advertencia y mensajes de información.

Cuando la tabla esta completa, las propiedades se mostraran así como la Figura 3.6:

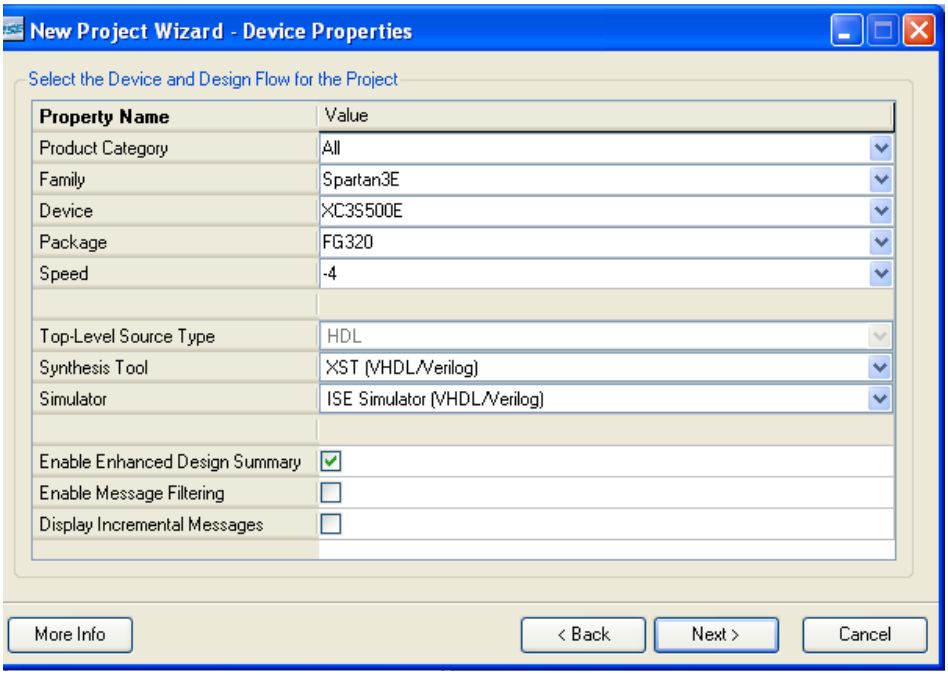


Figura 3.6: Propiedades del proyecto

Verifique que **Enable Enhanced Design Summary** es seleccionado.
Deje los valores por defecto en los otros campos.

C. Click Next para crear la ventana New Source en el New Project, como indica la Figura 3.7

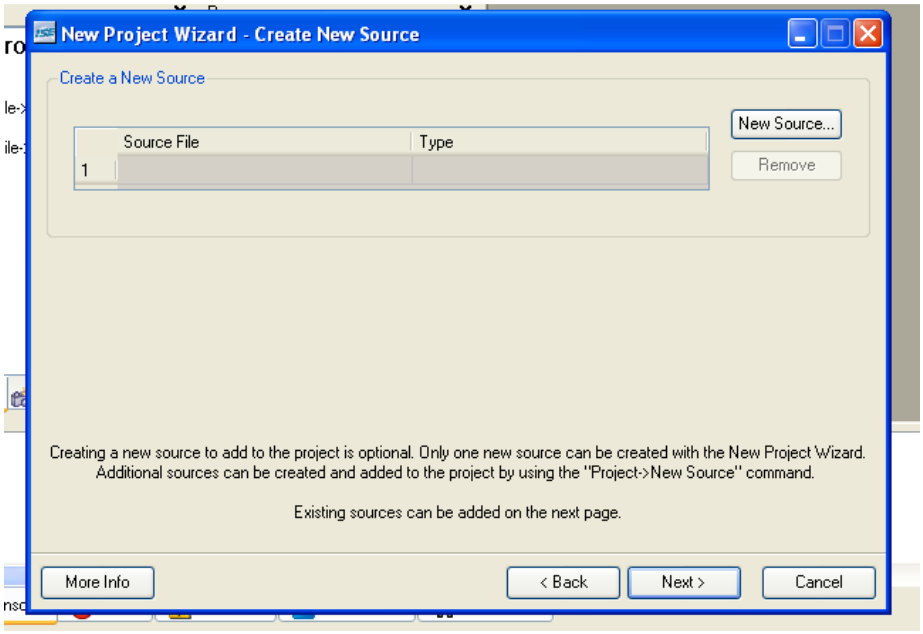


Figura 3.7 Crear nueva fuente para el proyecto

No se requiere crear nueva fuente, haga click en next se mostrara la ventana de la figura 3.8

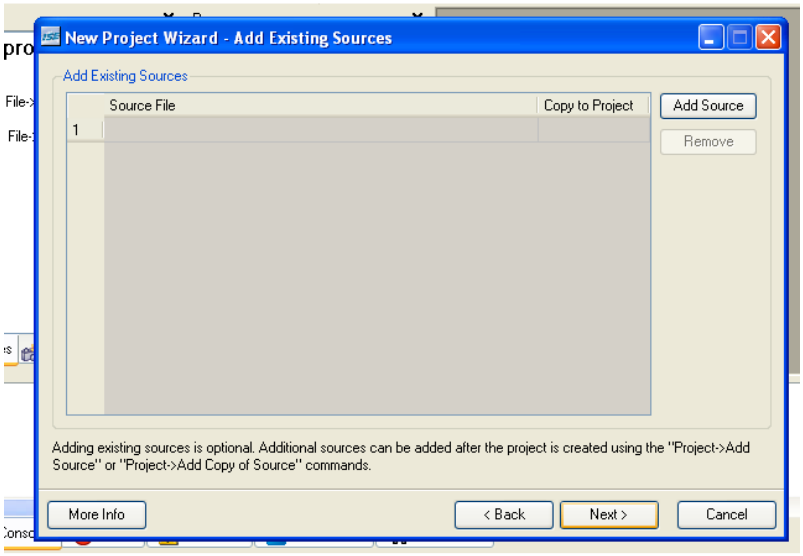


Figura 3.8 Adicionar nueva fuente para el proyecto

aquí da la opción de anexar nuevos archivos fuente , no se requiere, haga click en next nuevamente, así se muestra en pantalla el resumen de las propiedades del nuevo proyecto creado como indica la figura 3.9

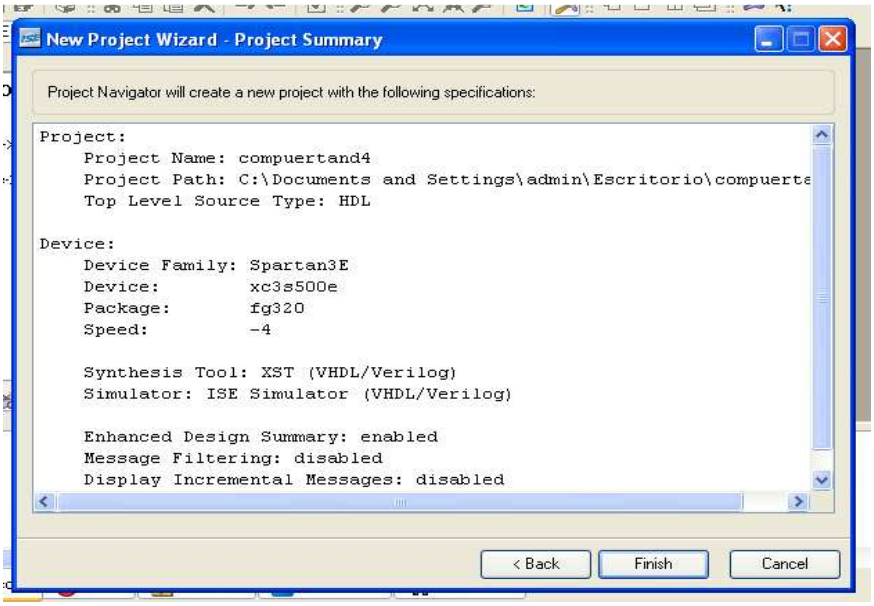


Figura 3.9 Ventana resumen de las propiedades del proyecto

haga click en **finish,** su nuevo proyecto esta creado, En la ventana sources se observa el icono del nuevo proyecto a desarrollar llamado *compuertand4.* como se muestra en la figura 3.10.

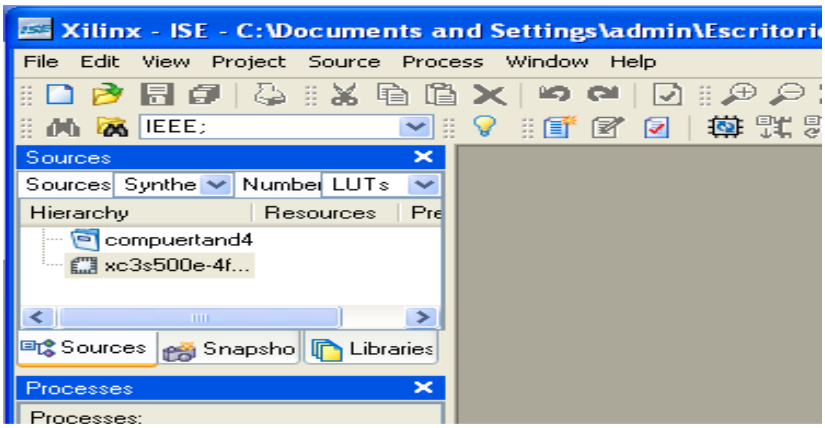


Figura 3.10 Nuevo icono del proyecto compuertand4. en la ventana sources

3.3 Crear un archivo fuente HDL

En esta sección, usted puede crear el .archivo fuente HDL para su diseño como sigue:.

A. Click en **New Source** dentro de la ventana New Project Wizard como muestra la Figura 3.11.

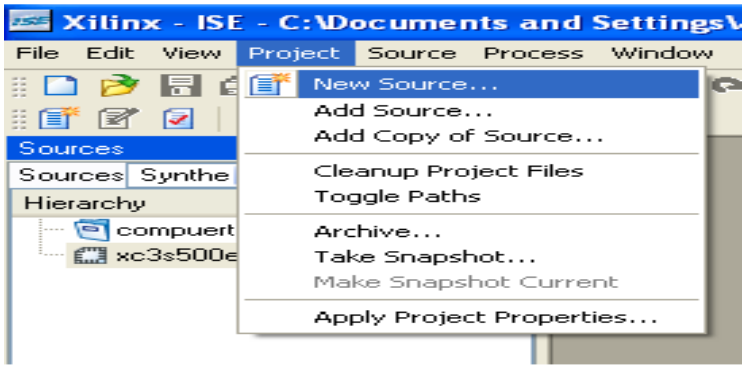


Figura 3.11 Crear nuevo archivo fuente del proyecto

B Seleccione un tipo de archivo fuente de la lista dependiendo del dispositivo y diseño especificado para su proyecto, para este caso seleccione **VHDL Module** como tipo de fuente

C. Digite en el campo **file name** el nombre iniciando por una letra (A-Z, a-z) y debe contener solamente carácteres alfanumericos (A-Z, a-z, 0-9) y underscores (_). Para este caso digite *compuertand4..el campo* **Location** contiene la ruta del proyecto actual por defecto.Verifique que la casilla **Add to project** este seleccionada, para adicionar el archivo fuente al proyecto como muestra la figura 3.12 , para información y ayuda detallada dar click en boton **More Info**

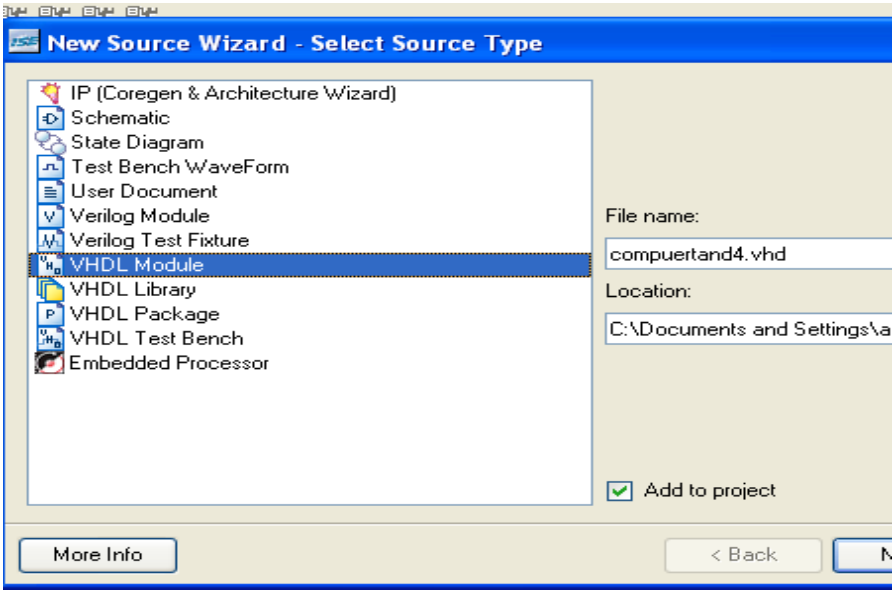


Figura 3.12 Selección tipo de fuente

D. Click en Next.
E. Declare los puertos para el diseño de la compuerta con cuatro entradas y una salida, esta se define seleccionando out, llenando la información como se muestra en la Figura 3.13

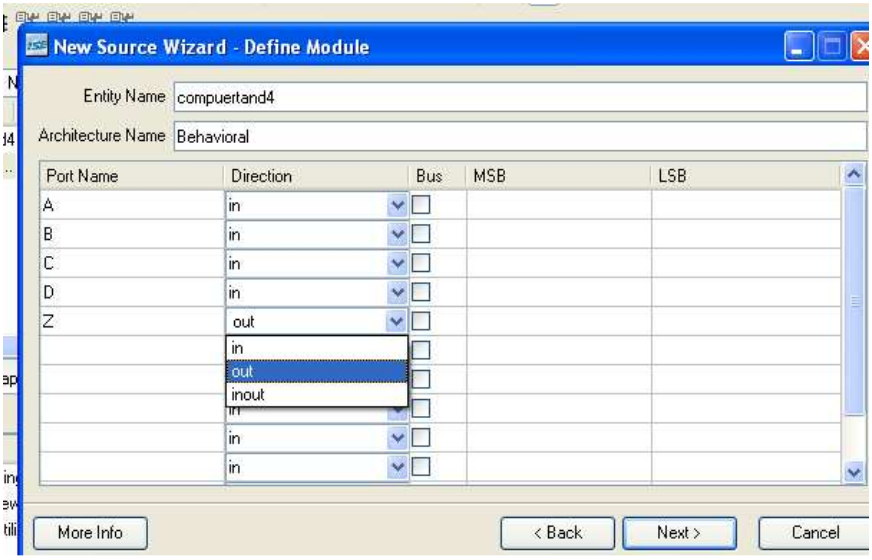


Figura 3.13 Declarar entradas y salidas del proyecto

F. Click Next,

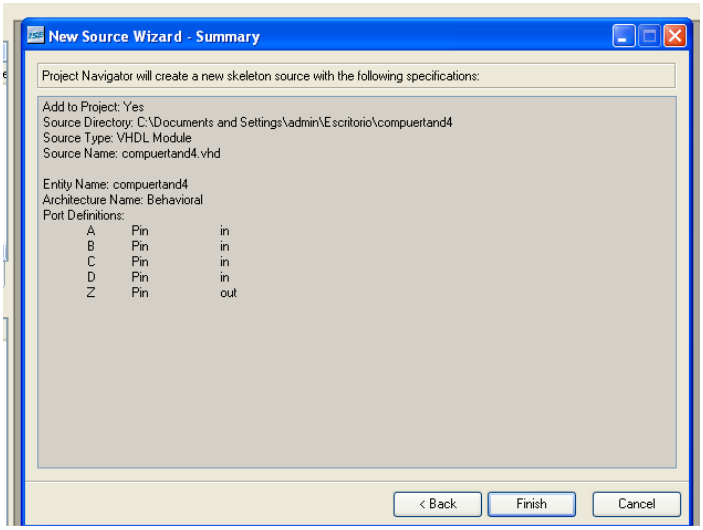


Figura 3.14 Resumen del nuevo archivo fuente

En la figura 3.14 se observa el cuadro resumen del nuevo archivo fuente del proyecto que se esta realizando.Ahora damos click en **Finish**
El archivo fuente contiene el par entidad / arquitectura visualizados en el espacio de trabajo, y el display de *compuertand4.* es como se muestra en la figura 3.15

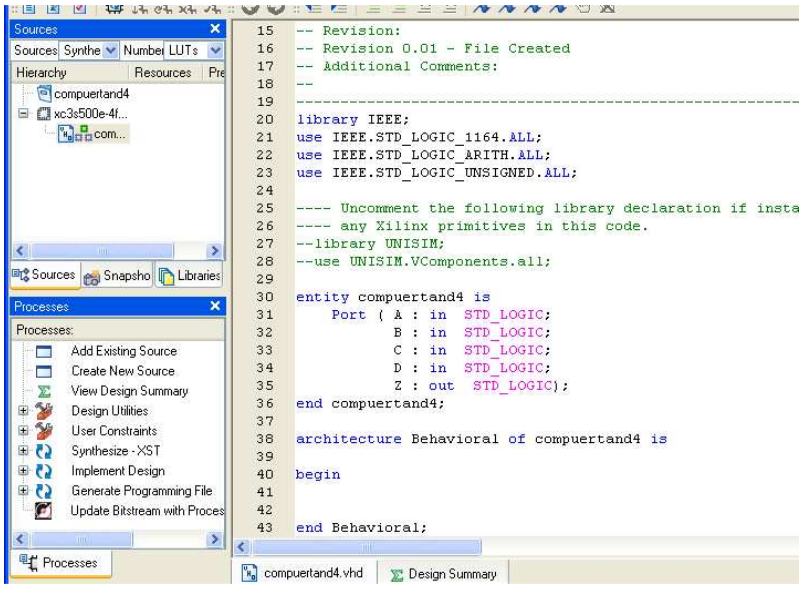


Figura 3.15 Espacio de trabajo para editar archivo fuente

3.4 Usando plantillas de lenguaje (VHDL)

El proximo paso en la creación del archivo fuente es adicionar la descripcion de la *compuertand4..* Para hacer esto usted necesita usar codigo de compuerta *and* desde plantillas del lenguaje ISE y utilizarlas en el diseño del circuito *compuertand4.*.

A. Ubique el cursor justo debajo donde comienza la declaracion dentro de la arquitectura de la compuertand4. en el espacio de trabajo como indica la figura 3.16

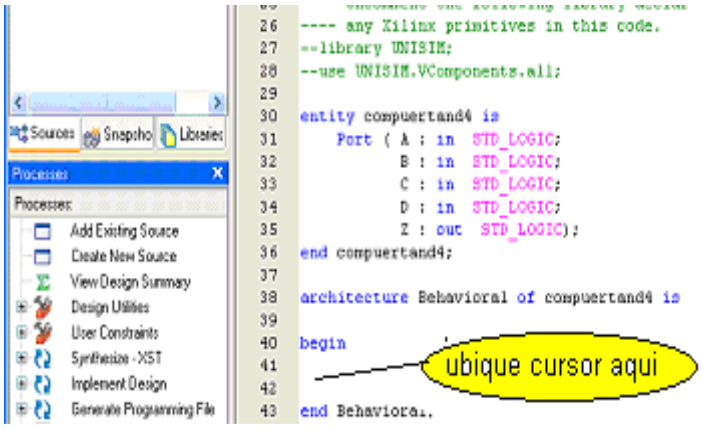


Figura 3.16 Ubicación cursor para edición archivo

B. Abra las plantillas de lenguaje seleccionando **Edit** →**Language Templates** *como muestra la Figura 3.17.*

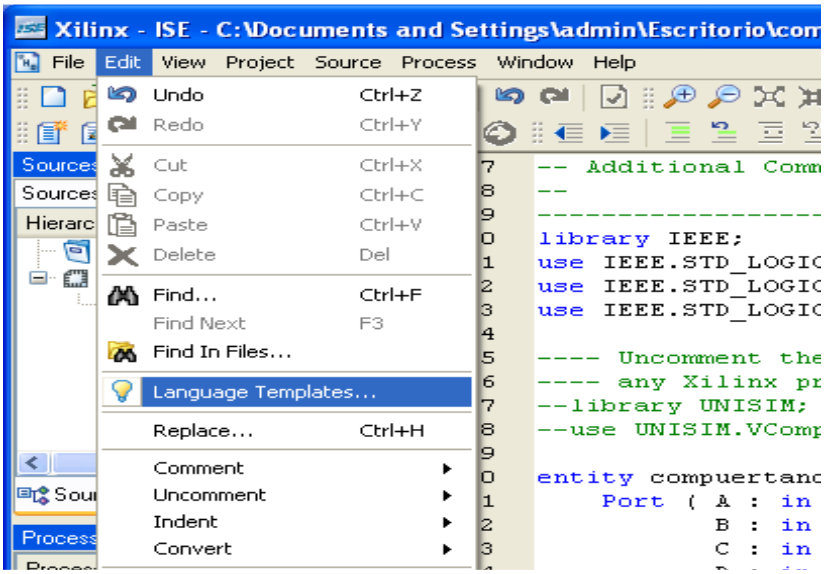


Figura 3.17 Ubicación plantillas de lenguaje

C. Usando el simbolo “+”, busque el siguiente codigo de ejemplo :
VHDL →**Synthesis Constructs** →**Coding Examples** →**Basic Gates** →**AND** →**4-Input AND Gate** como indica la Figura 3.18.

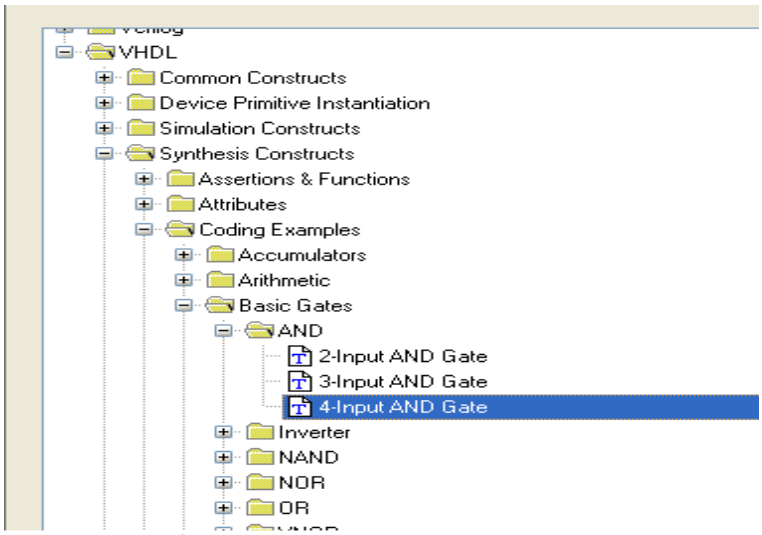


Figura 3.18 Ubicación plantilla VHDL para compuerta and 4 entradas

D. Con **4-Input AND Gate**, seleccione **Edit** →**Use in File** para copiar plantilla como muestra la Figura 3.19

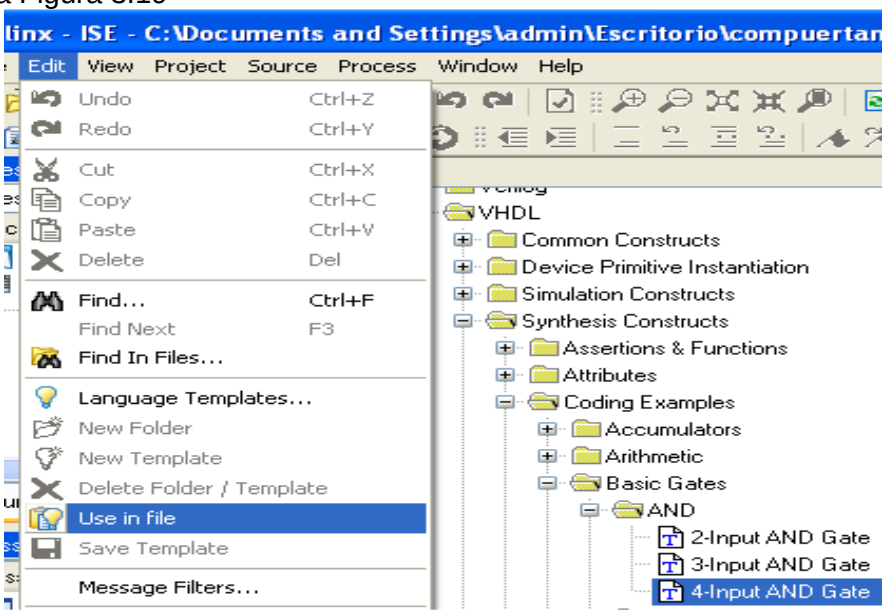


Figura 3.19 Usar plantilla seleccionada en archivo fuente

La plantilla copiada dentro del archivo fuente de *compuertand4.* se indica en la figura 3.20

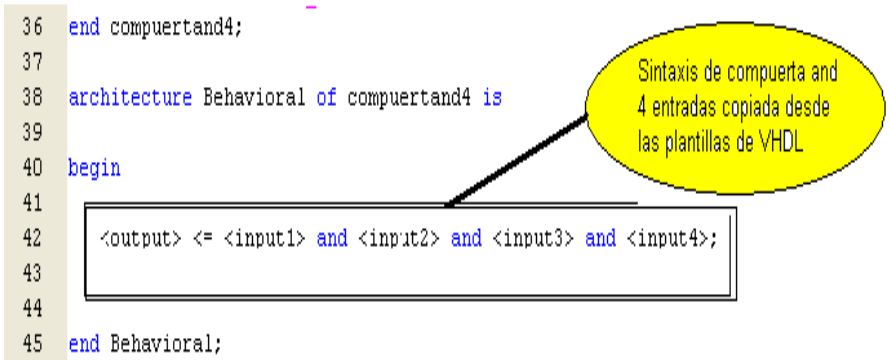


Figura 3.20 Sintaxis copiada de compuerta and

E. Cierre las plantillas de lenguaje seleccionando la pestana **Languaje Templates** y dar click en icono de cierre arriba ala derecha, ver Figura 3.21.

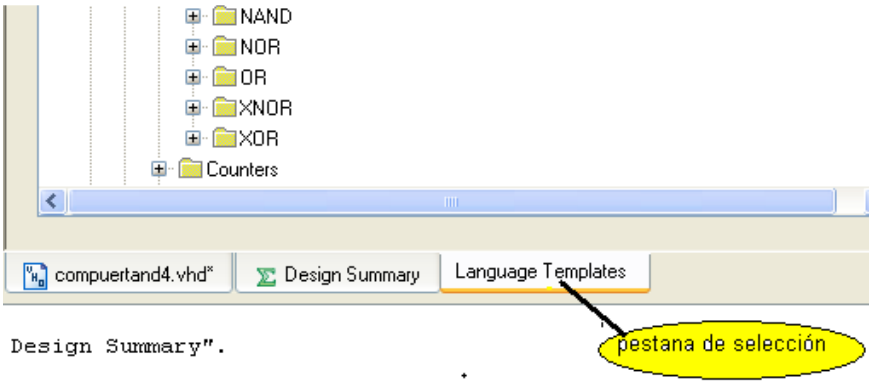


Figura 3.21 Cierre plantillas de lenguageVHDL

3.5 edición final del archivo fuente VHDL

A. Edite el archivo fuente para la compuerta AND diseñada reemplazando el nombre del puerto entre corchetes con uno real como sigue

- reemplace <output> con **Z**
  <input1> con **A**
  <input2> con **B**
  <input3> con **C**
  <input4> con **D**

B. Salve el archivo seleccionando **File** → **Save**.

Cuando usted finaliza, el archivo fuente de la compuerta AND se observa como indica la Figura 3.22

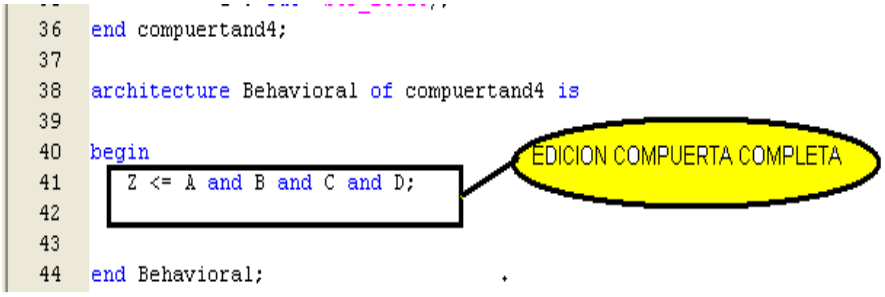


Figura 3.22 Proyecto final editado

Usted tiene creado ahora el archivo fuente VHDL luego procedemos a verificar la sintaxsis del nuevo modulo.

3.6 Chequeando la sintaxis del nuevo modulo de la *compuertand4.*

A. Verifique que **Synthesis/Implementation** es seleccionado en la lista de la ventana **Sources** como se indica en la figura 3.23

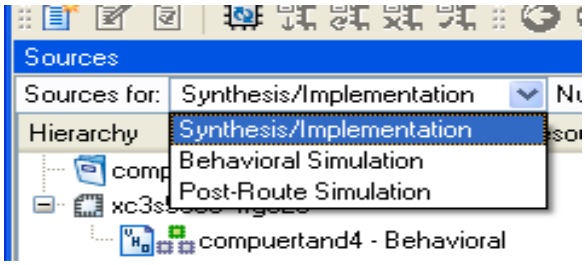


Figura 3.23 Modo Synthesis/Implementation

B. Seleccione el archivo fuente de diseño *compuertand4.* en la ventana **Sources** para visualizar el proceso relacionado en la ventana de **processes** , ver Figura 3.24

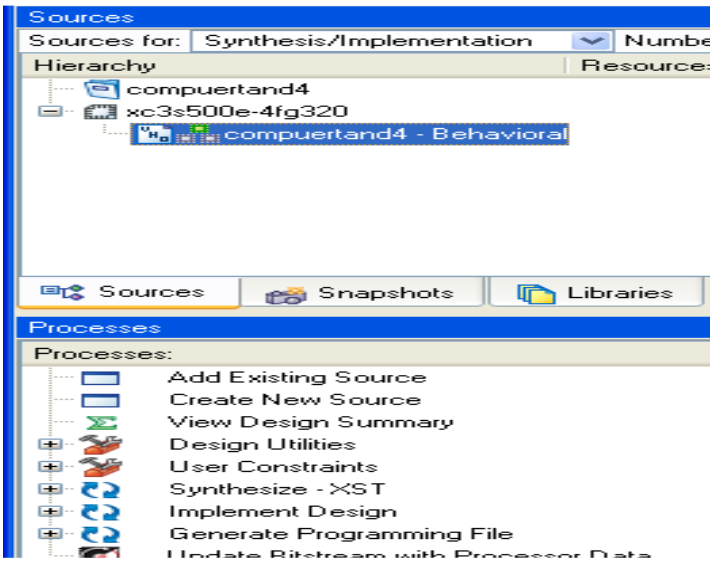


Figura 3.24 Vista procesos del archivo fuente

C. Click en "+" cerca al proceso Synthesize-XST para expandir el grupo de procesos.Ver Figura 3.25

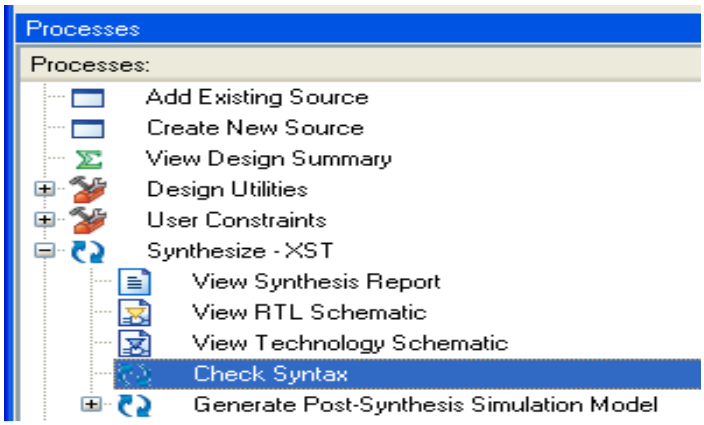


Figura 3.25 Expansión proceso Synthesize-XST

D. Hacer Doble-click en **Check Syntax** para que el programa realice una revisión de la sintaxis del módulo HDL que estaba editando.Al finalizar el chequeo de sintaxis en la ventana Transcript. Ver Figura 3.26

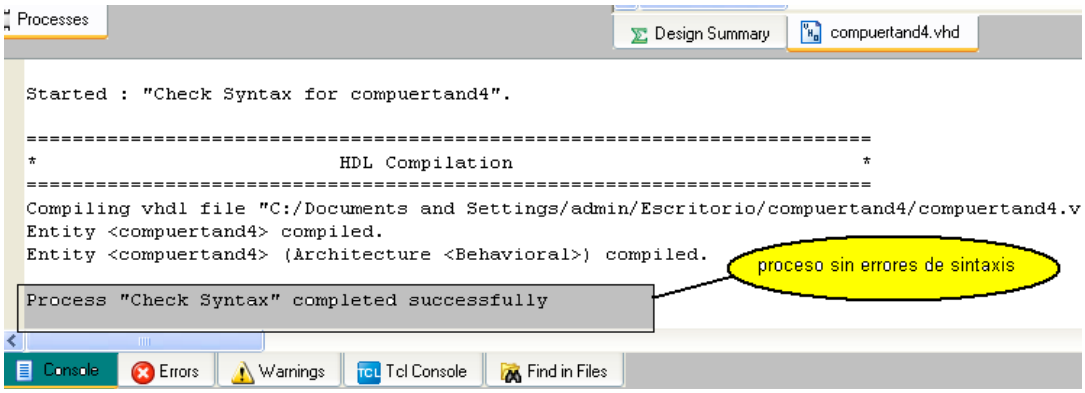


Figura 3.26 Proceso sintaxis finalizado sin errores

Nota: El resultado del chequeo se observa en la parte inferior de la pantalla, si fue satisfactorio aparece el mensaje "*Process "chec syntax" completed successfully"* pero si hay errores o advertencias indica la línea donde se presenta Si usted continua sin validar la sintaxis, usted puede no tener disponible la simulación o sintetizado de su diseño.

E. cierre el archivo HDL

.

3.7 Simulación del diseño

Verificar funcionalidad usando Behavioral Simulation

Cree un banco de prueba que contenga unas entradas determinadas para verificar la funcionalidad del modulo de la compuerta AND. El test bench waveform es un modo grafico de un banco de pruebas.

Cree el banco de pruebas como sigue:
A. Seleccione el archivo HDL compuertand4. en la ventana sources, ver Figura 3.27

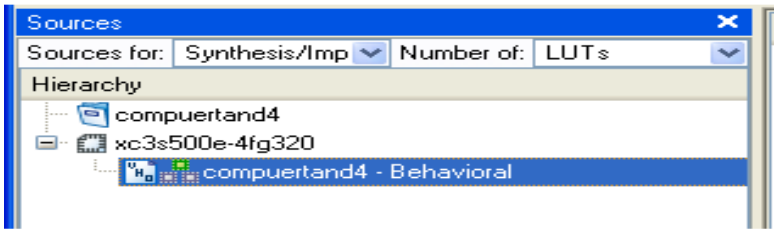


Figura 3.27 Selección archivo HDL

B. Cree un nuevo banco de pruebas seleccionando **Project →□New Source**.
En el **New source**, seleccione **Test Bench WaveForm** como el tipo de fuente, y digite **compuertand4._tbw** en en el campo de nombre de archivo como se indica en la Figura 3.28

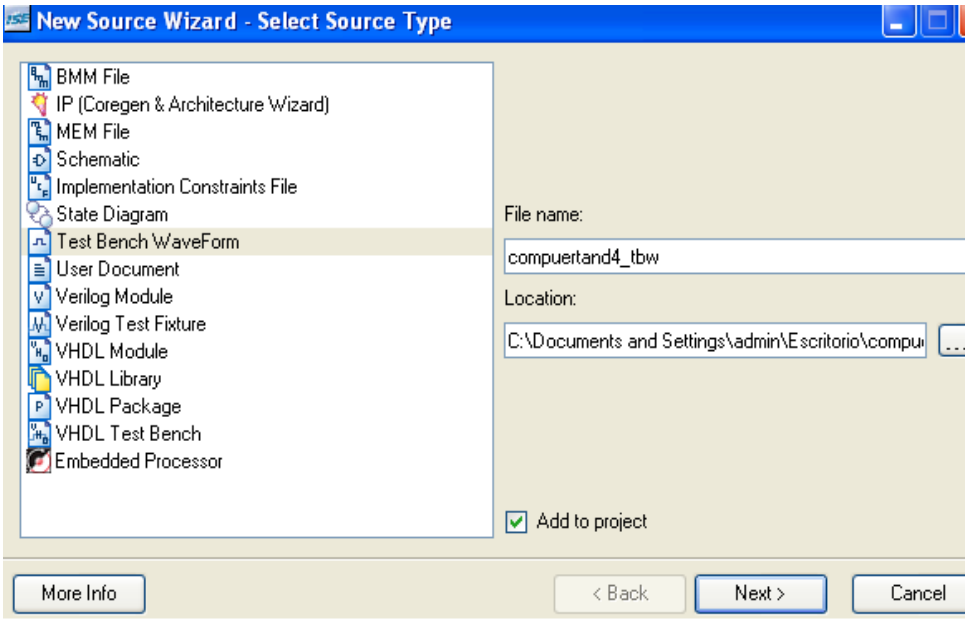


Figura 3.28 Creación módulo banco de pruebas

Nota . Verifique que el nombre quede con (_tbw).
C. Click Next. La ventana de fuente asociada muestra que usted esta asociando el banco de pruebas con el archivo fuente de la compuerta AND.. Click Next, ver figura 3.29.

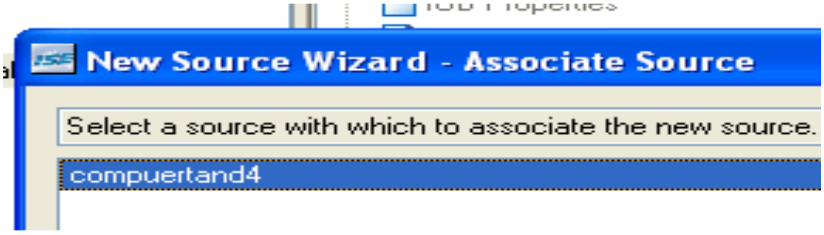


Figura 3.29 Asociación nuevo archivo fuente con proyecto

D. La pagina de resumen muestra que la fuente podria ser adicionada al proyecto,y este muestra el directorio raiz,tipo y nombre. Click **Finish**.Ver figura 3.30

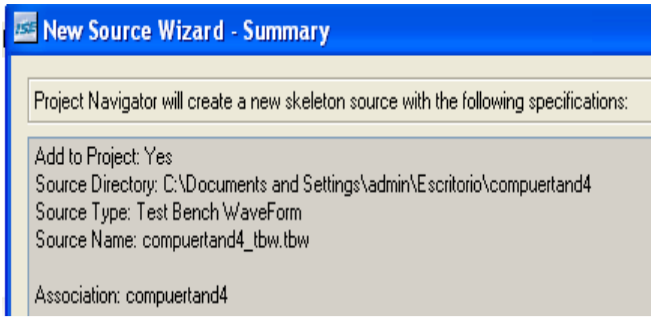


Figura 3.30 Resumen del módulo creado

E. Usted debe definir si necesita señal de reloj o es un circuito combinacional antes de abrir la ventana de edición, para este caso es combinacional luego seleccione Combinatorial (or internal clock)., configure 10 y 90 nanosegundos en check outputs y assign inputs respectivamente como se muestra en la Figura 3.31

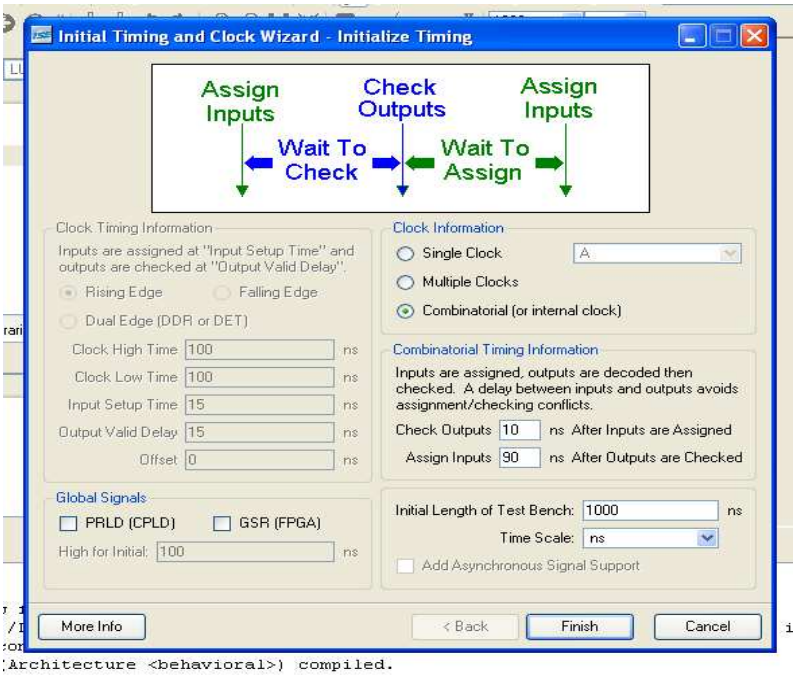


Figura 3.31 Configuración señal de reloj

F. Haga click en **Finish** para finalizar la configuración del tiempo de iniciación.
En la ventana de trabajo aparece las entradas con zonas azules y salidas zonas amarillas las cuales usted puede modificar de 1 a 0 o viceversa dando click en las zonas azules y amarillas para configurar los valores esperados de salida según los valores de entrada, para este caso lo ideal es tener 1 en la salida cuando todas las entradas son 1 y 0 en los demas casos donde se tenga 0 en una de sus entradas, ver figura 3.32

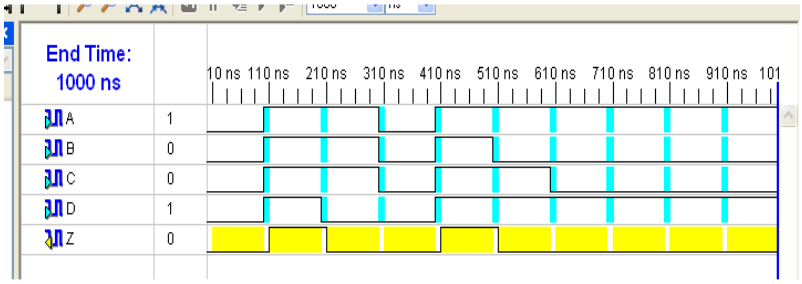


Figura 3.32 Banco de pruebas

G. Guarde el banco de pruebas. Dando click arriba a la derecha y salvar los cambios realizados dando **yes** en el mensaje como se muestra en la figura 3.33

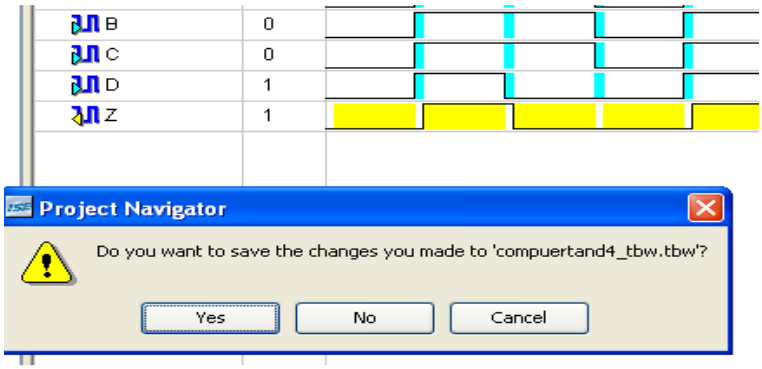


Figura 3.33 Guardando banco de pruebas

H. En la ventana de Sources, seleccione el modo **Behavioral Simulation** para ver que el archivo de banco de pruebas a sido adicionado automáticamente a su proyecto.Ver Figura 3.34

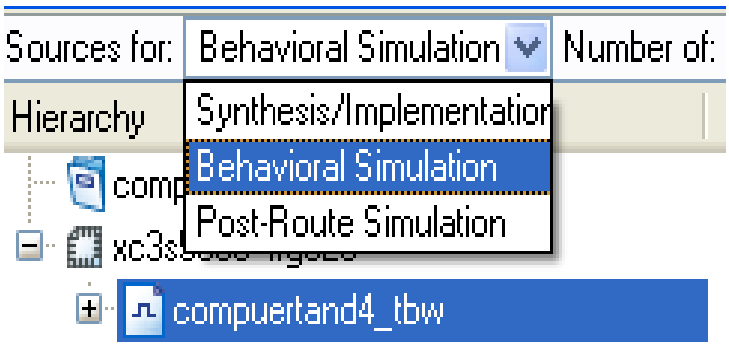


Figura 3.34 Banco de pruebas adicionado al proyecto

3.8 Crear un auto chequeo para el banco de prueba

Como ya adiciono los valores de salida esperados para finalizar la creación del test bench waveform,esto transforma el test bench waveform en una auto chequeo del test bench waveform

EL Beneficio de un auto-control de test bench waveform es que compara los valores de salida deseados y los reales y genera banderas de los errores en su diseño, ya que esté pasa por los diferentes transformaciones, desde HDL hasta el dispositivo específico de representación FPGA.

Para crear una auto-verificación de banco de pruebas, con los valores modificados de salida manualmente y compararlos, ejecutar el proceso **Generate Expected Results** para crearlo de forma automática. Si ejecuta el proceso **Generate Expected Results**, inspeccionar visualmente los valores de salida para ver si ellos son los que esperaba para el conjunto dado de valores de entrada.y la salida llamada DIFF

Para crear la auto-verificación de test bench waveform automáticamente, haga lo siguiente:

A. Verifique que **Behavioral Simulation** se ha seleccionado en la lista desplegable de la ventana **sourses**.

B. Seleccione el archivo **compuertand_tbw** en la ventana sources.

C. En la pestaña Processes, haga click en el signo "+" para ampliar el proceso de simulación de Xilinx ISE Haga doble click en el proceso. **Generate Expected Simulation Results** Este proceso simula el diseño en un proceso.Ver Figura 3.35

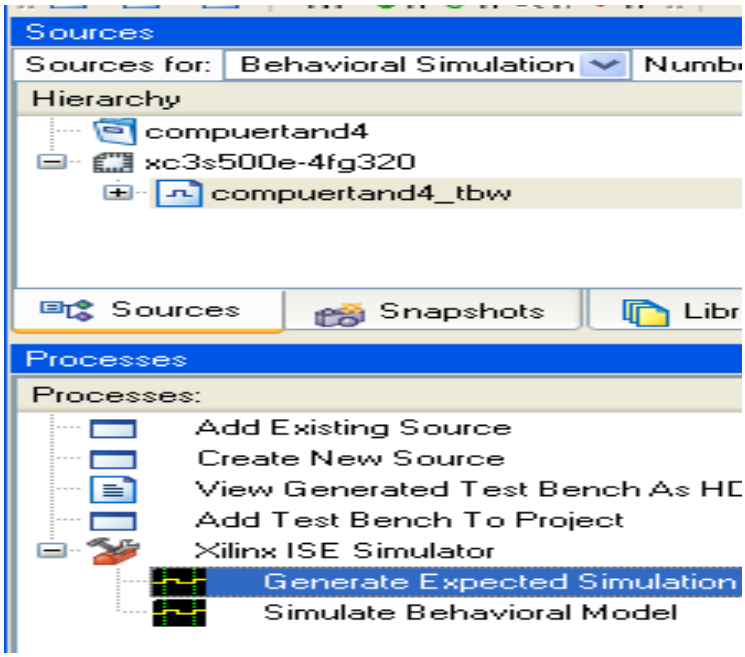


Figura 3.35 Generar auto chequeo con banco de pruebas

D. El cuadro de diálogo **Expected Results** se abre, hay dos opciones. Seleccione **No** para adicionar una señal de comparación en la visualizacion del resultado o **yes** para generar las salidas automáticamente.Si selecciona **No,** ver figura 3.36

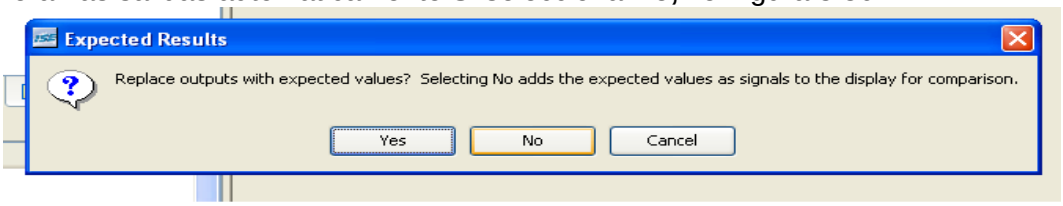


Figura 3.36 Selección para tener señal de comparación

en el area de trabajo se despliega los resultados , mostrando dos señales de salida llamadas **Z** y **Z _DIFF** donde **Z** es la salida previamente configura por el usuario y **Z_DIFF** es la calculada por el programa,luego aquí evalua el usuario si era lo esperado o si requiere modificar el programa del diseño, como se observa en la Figura 3.37

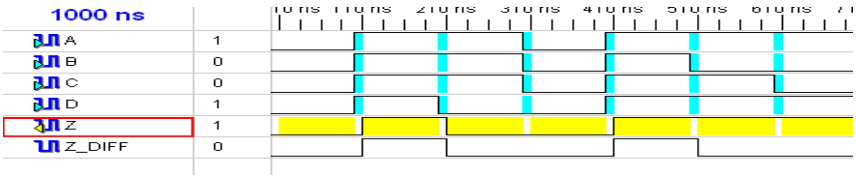


Figura 3.37 Senales de salida permite comparación de señales de salida

Ahora cuando el cuadro de diálogo **Expected Results** se abre y selecciona **yes,** como indica Figura 3.38 los resultados en el Banco de pruebas se actualizan y muestra la salida generada según las entradas, pero no hay comparación.

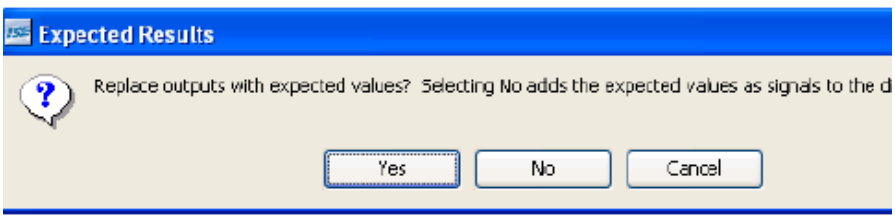


Figura 3.38 Selección para tener señal generada automaticamente

E. Haga click en el signo "+" para ampliar el bus **compuertand_OUT (si aplica)** y ver la lista de las transiciones corresponden al valor de salida de retardo (celdas amarillas) especificado en el cuadro de diálogo inicial de tiempos.Este caso no aplica porque hay una sola salida y no es un vector el cual tiene varias salidas como indica la figura 3.39.

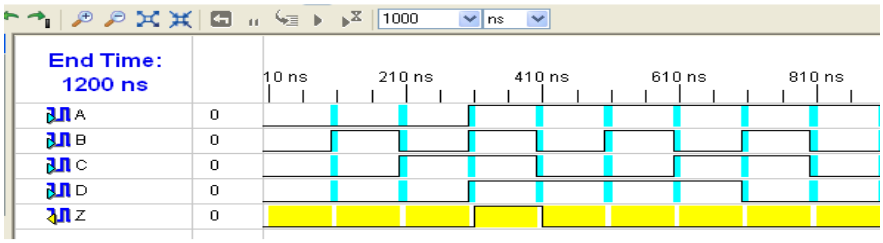


Figura 3.39 Banco de pruebas con resultado generado automaticamente

F. Salvar el test bench waveform y cerrarla, así queda creado un auto-chequeo del **test bench waveform.**

3.9 Simulando la funcionalidad del diseño

Verifique que la compuerta diseñada funciona como se espera realizando la simulación de la siguiente manera:
A. Verifique que **Behavioral Simulation** and **compuertand_tbw** son seleccionados en la ventana sources
B. En la pestaña **Processes**, haga click en el signo "+" para ampliar el proceso de simulación de Xilinx ISE y haga doble click en el proceso **Simulate Behavioral Model**.
El ISE 8.2i simulador abre y ejecuta la simulación del banco de pruebas.
C. Para ver los resultados de la simulación, seleccione la pestaña de **simulación** y haga un zoom en las transiciones.Los resultados de la simulación se verán como se muestra la Figura 3.40

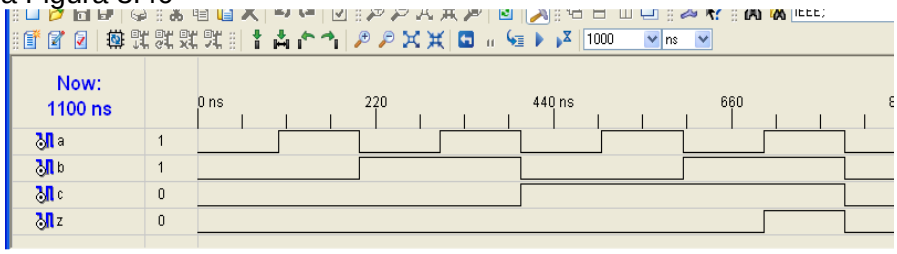


Figura 3.40 Resultados de la simulación

D. Verifique que la compuerta AND opera como se esperaba.
E. Cierre la simulación. Si aparece el siguiente mensaje: ""You have an active simulation open. Are you sure you want to close it?, Haga click en **yes** para continuar.como se indica en la figura 3.41

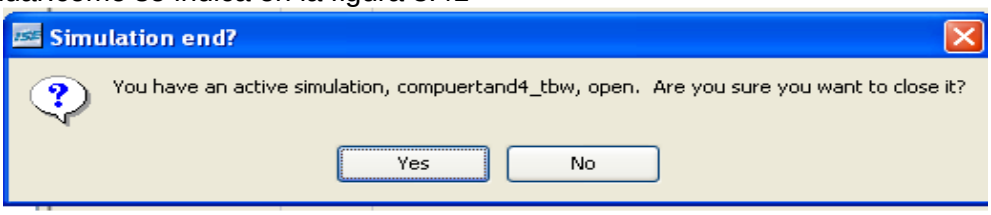


Figura 3.41 Finalizar simulación

Usted ya completo su simulación utilizando el simulador de ISE.

3.10 Creación de tiempos de sincronización

Sincronizando el tiempo entre la FPGA y la lógica de su entorno, así como la frecuencia del diseño es como deberá actuar en el ámbito interno de la FPGA. El tiempo es especificado para sincronizar el desempeño del diseño. Se recomienda que use sincronización a nivel global. El período de reloj sincroniza la frecuencia de reloj a la que su diseño debe operar dentro de la FPGA. El offset de sincronismo especifica cuándo esperar datos válidos en las entradas del FPGA y cuando datos validos estarán disponibles en las salidas del FPGA
Para sincronizar el diseño haga lo siguiente:

A. Seleccione **Synthesis/Implementation** de la lista desplegable en la ventana **sources**
B. Seleccione archivo fuente.HDL de *compuertand4.* como muestra la Figura 3.42

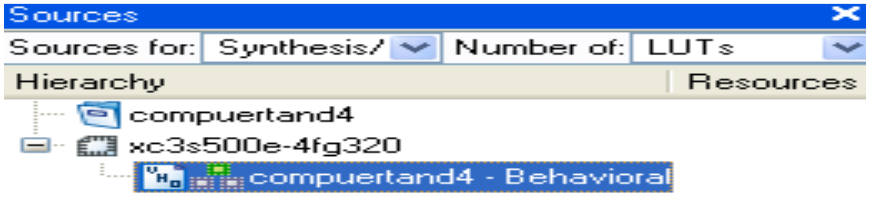


Figura 3.42 Selección archivo fuente VHD

C. Haga click en el signo "+" junto al grupo de procesos **user constraints**, y haga doble click en el proceso **Create Timing Constraints**.Ver Figura 3.43.

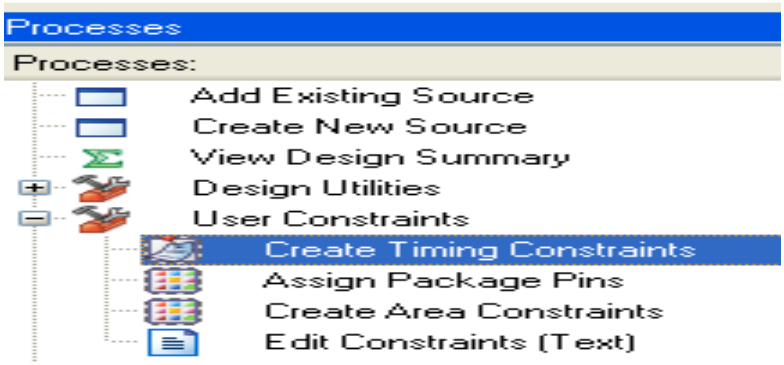


Figura 3.43 crear archivo de sincronismo

ISE corre los pasos de Synthesis y Translate creando automáticamente un **User Constraints File (UCF);** aparece un mensaje como lo muestra la Figura 3.44:

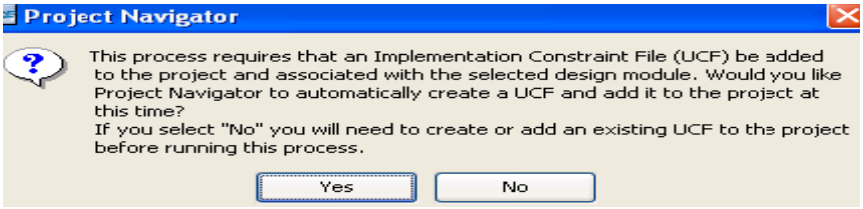


Figura 3.44 Anexar archivo UCF al proyecto

D. Haga click en **Yes** para agregar el archivo UCF a su proyecto. El archivo **Compuertand4.ucf** se añade a su proyecto y es visible en la ventana **sources** Ver figura 3.45.

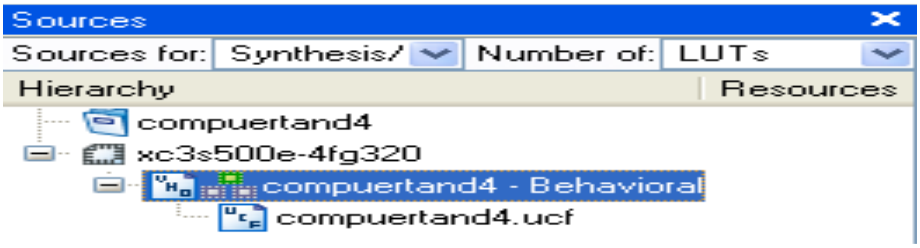


Figura 3.45 Verificar archivo UCF anexado al proyecto

El Xilinx Constraints Editor se abrirá automáticamente. Nota: También puede crear un archivo UCF para su proyecto seleccionando **Project Create New Source**.

Para este caso no se requiere reloj por ser un circuito combinacional, luego cierre la ventana Xilinx Constraints Editor arriba a la derecha.Ver figura 3.46

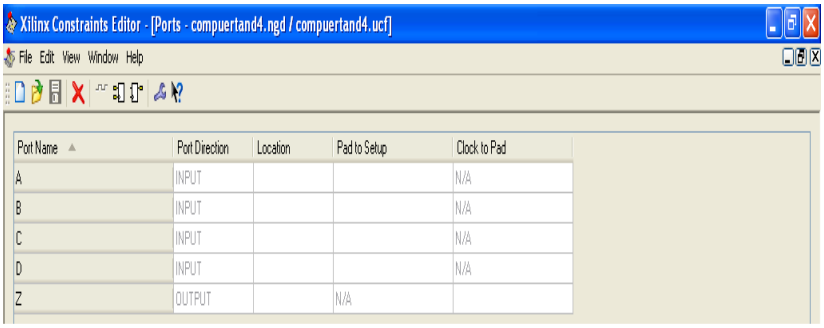


Figura 3.46 Ventana edición tiempos sincronismo

3.11 Implementación diseño y verificar sincronismo

Implementar el diseño y verificar que este cumple con los tiempos especificados en la sección anterior.

A. Seleccione el archivo fuente *compuertand4.* en la ventana sources
B. Abra el **Design Summary** haciendo doble click en el proceso **View Design Summary** en la pestaña Processes. Ver figura 3.47.

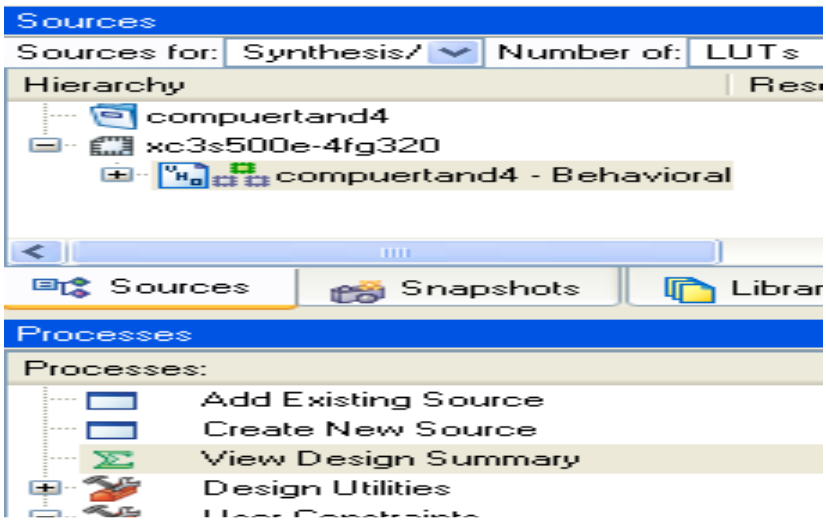


Figura 3.47 Activar resumen del diseño

C. Haga doble click en el proceso **Implement Design** en la pestaña Processes. Ver Figura 3.48.

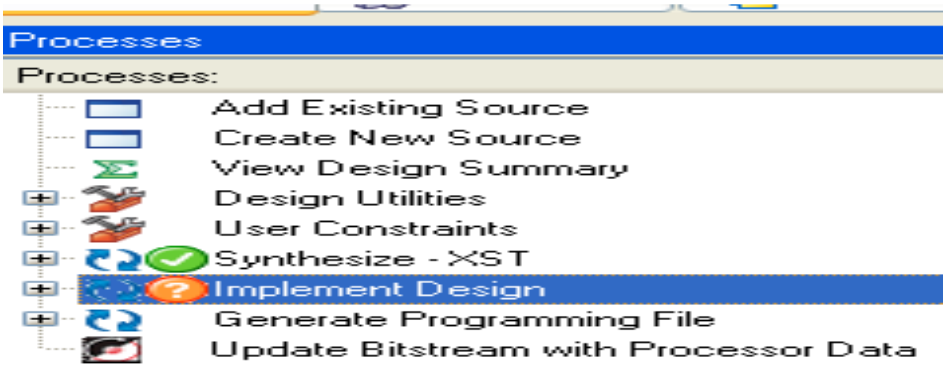


Figura 3.48 Implementar diseño

D. Observe que después de completar la implementación del diseño (implement Design) aparece al lado del proceso implementado un círculo verde con una flecha blanca, tal como se aprecia en el anterior grafico al lado de Synthessize – XST, la cual indica que terminó con éxito sin errores o advetencias.
G. Cierre el Design Summary.

3.12 Asignacion de pines del diseño al kit de desarrollo

Concretar la ubicación de los pines para los puertos del diseño a fin de que estén conectados correctamente en el Kit de desarrollo
Para asignar los puertos del diseño al paquete de pines haga lo siguiente:

ACompruebe que está seleccionado compuertand en la ventana sources

B.Haga doble click en el proceso **Assign Package Pins** encontrandolo en el grupo **User Constraints process**.ver figura 3.49

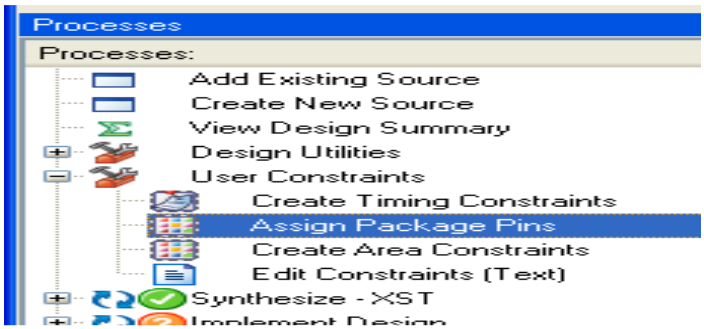


Figura 3.49 Ubicacion ventana asignación pines

C The Xilinx Pinout and Area Constraints Editor (PACE) abre(notese que al realizar este procedimiento se abre ventana con dos pestañas) seleccione la pestaña **Package View** ubicada en el area de trabajo. Ver figura 3.50

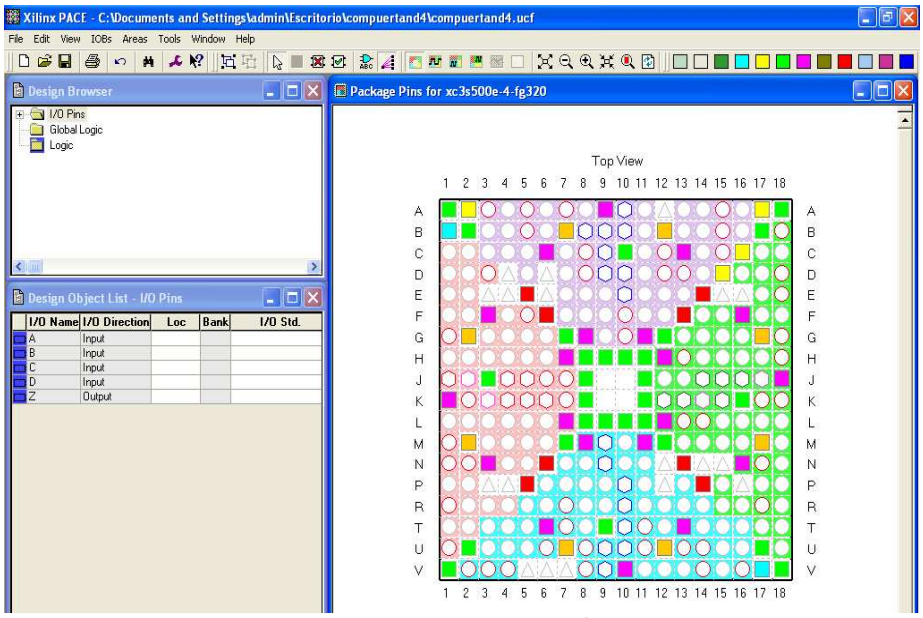


Figura 3.50 Ventana asignación de pines

D. En la ventana Design Object List, asignar una localización de un pin para cada pin en la columna **Loc**. (para este caso empezamos a utilizar simultáneamente la tarjeta electronica del kit de desarrollo, ver proceso de coneccion e instalacion de la misma)

Acontinuacion realizaremos la configuración de pines del diseño realizado con base a las entradas y/o salidas de la tarjeta electronica del kid de desarrollo, para este caso se utilizaran las configuraciones dadas en la Figura 3.51, (para mayor información ver tarjeta y manual de la misma), se debe tener en cuenta que para la tarjeta en el Caso de la entrada A se configura el pin como L13 debido a que así lo reconoce el software, aunque para la tarjeta electronica aparezca como SW0(l13).

Design Object List - I/O Pins

I/O Name	I/O Direction	Loc	Bank	I/O Std.
a	Input	L13	BANK	
b	Input	L14	BANK	
c	Input	H18	BANK	
d	Input	N17	BANK	
s	Output	F12	BANK	

Figura 3.51 Configuracion pines para el proyecto

Para el caso de la salida de la compuerta se configura el pin F12 (senalizado en la board LD0)

Note que el lugar del pin asignado se muestra en azul: tal como se aprecia en la Figura 3.52

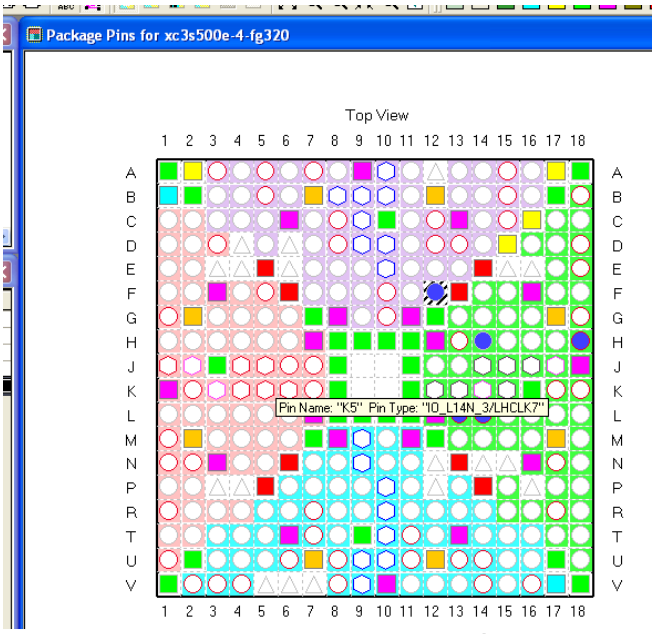


Figura 3.52 Planilla de asignación pines

E. Seleccione **File** →**Save** Aquí graba el archivo de asignación de pines
F. Cerrar PACE.
Note que **Implement Design** en la ventana processes tienen una marca color naranja junto a el, Indicando que están fuera de sincronismo con uno o varios de los archivos del diseño. Esto se debe a que la archivo UCF ha sido modificado haga doble click en **Implement Design**.al final del proceso la marca naranja se vuelve verde indicando proceso terminado.Por cada cambio que modifique los pines del diseño habra que volver a compilar la implemenrtacion del mismo. Ver figura 3.53

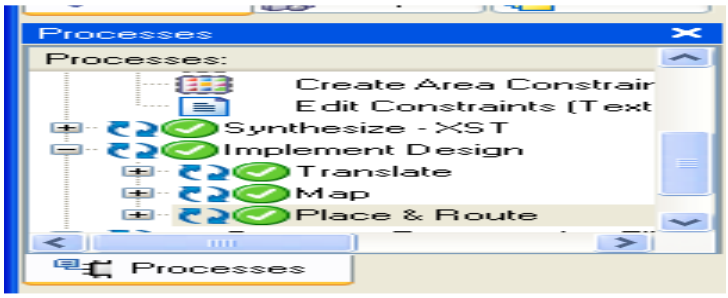


Figura 3.53 Implementación diseño terminado

3.13 Reimplementar diseño y verificación localización de pines

Reimplementar el diseño y verificar que los puertos de la compuerta diseñada estan enrutados al paquete de pines especificado en la sección anterior sera lo que realice así:.
A Revise el reporte de pines asignados desde la implementación previa haciendo lo siguiente: Habra el **Design Summary** haciendo doble click en el proceso **View Design Summary** en la ventana de procesos.Seleccione el **Pinout Report** y de click a la ventana **signal name**, aquí se muestra el resumen de la asignación de pines. Ver figura 3.54

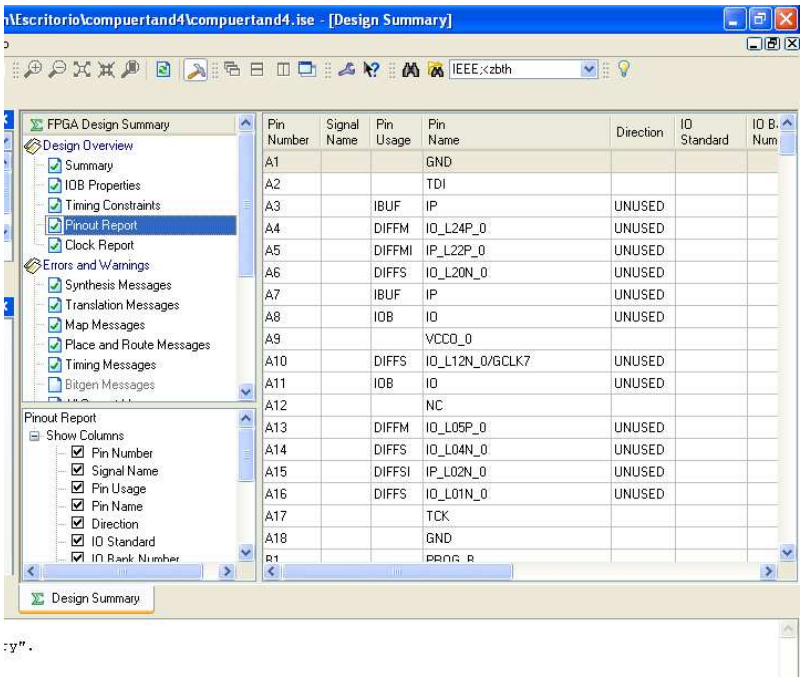


Figura 3.54 Cuadro resumen asignación de pines

B Reimplemente el diseño haciendo doble click en el proceso **Implement Design** (cuando realice cambios en la asignación de pines y le aparezca un círculo naranja con un signo de interrogación en blanco). Los pasos siguientes se hacen unicamente cuando se registro cambios en la asignación de pines, en caso contrario vaya directamente al paso **E**

C Seleccione el **Pinout Report** nuevamente y seleccione la columna **Signal Name** para ver el resumen de asignación de pines. Ver figura 3.55

Pin Number	Signal Name	Pin Usage	Pin Name	Direction	IO Stan
L13	A	IBUF	IP	INPUT	LVCM
L14	B	IBUF	IP	INPUT	LVCM
H18	C	IBUF	IP/VREF_1	INPUT	LVCM
H14	D	IBUF	IO_L17P_1	INPUT	LVCM
F12	Z	IOB	IO_L06P_0	OUTPUT	LVCM
A6		DIFFS	IO_L20N_0	UNUSED	
A7		IBUF	IP	UNUSED	
A8		IOB	IO	UNUSED	
A9			VCCO_0		

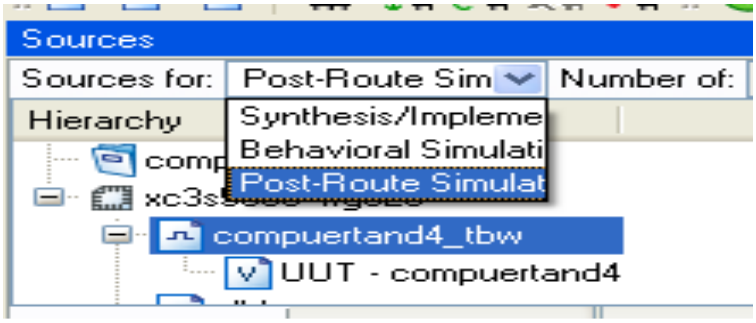
Figura 3.55 Asignación pines, luego de modificación diseño

D. Verifique que las señales ahora estan correctamente enrutadas.
E Cierre el **Design Summary**.

3.14 Verificar diseño usando tiempos de simulación

Use el mismo auto chequeo test bench waveform que usted creo en la sección anterior para verificar que la compuerta and diseñada después de esto a sido completamente implementada. Verifique que los tiempos de simulación operan dentro del sincronismo especificado después que retardos lógicos y enrutamientos son acomodados.
Realice la simulación como sigue:

A. Seleccione el **Post-Route Simulation** desde la lista desplegada en ventana sources
B. Seleccione compuertand en la ventana Sources. Ver figura 3.56



C Realice la simulación haciendo doble click en el proceso **Simulate Post-Place & Route Model** establecido en el grupo de procesos del simulador Xilinxs ISE.Ver Figura 3.57

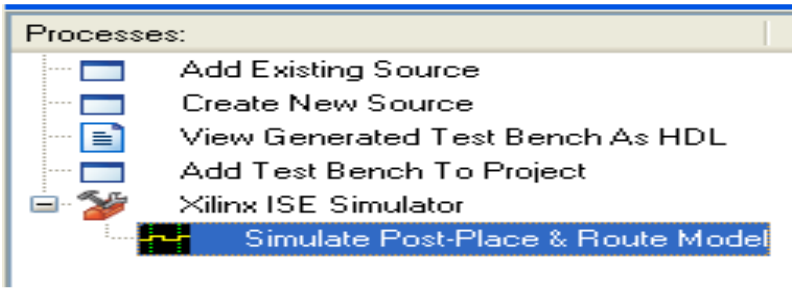


Figura 3.57 Comando Simulacion final

La simulación final del proyecto se observa en la figura 3.58

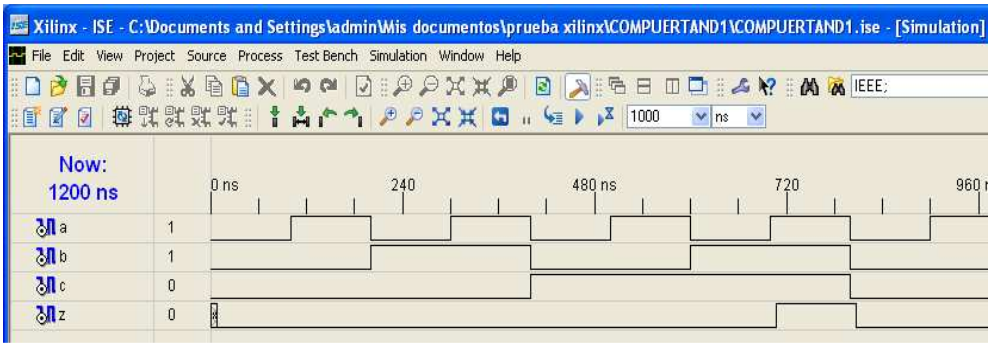


Figura 3.58 Simulación final del proyecto

D. Verifique que la compuertad4 esta funcionando directamente en el Puerto direccionado
E. Verifique que allí no hay errores reportados en la ventana de transcripción del simulador
F. **Zoom in** para ver el retardo actual desde el flanco del reloj para validar un cambio en la salida.
G. Cierre la simulación
Usted tiene completa la simulación de su diseño usando el ISE simulator
.

3.15 Proceso de diseño descargado en la tarjeta

Este es el último paso en el proceso de verificación del diseño. Esta sección provee simples instrucciones para descargar la compuerta diseñada al kit de desarrollo.

A Conecte el cable de potencia de 5Vdc a la entrada sobre el kit de desarrollo (J20).
B Conecte el cable de descarga entre el computador y el Kit de desarrollo.(J18) como se indica en la figura 3.59 .

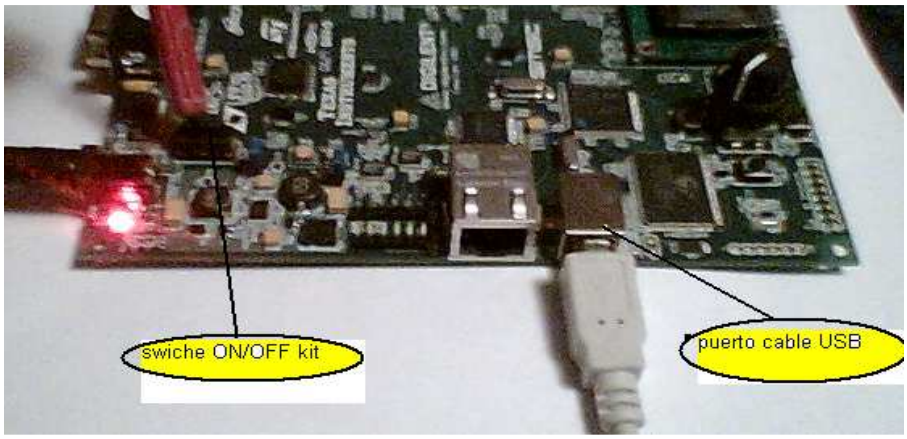


Figura 3.59 Conexión tarjeta kit de desarrollo

cuando el kit tiene comunicación activa con el computador un led cerca al puerto USB se ilumina como indica la Figura 3.60

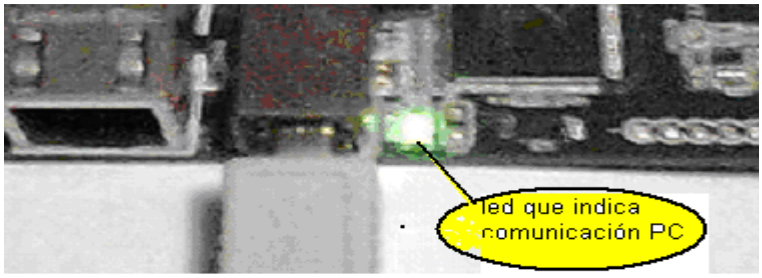


Figura 3.60 Indicacion comunicación activa con PC

C Seleccione **Synthesis/Implementation** desde la lista desplegada en la ventana de sources. Ver Figura 3.61

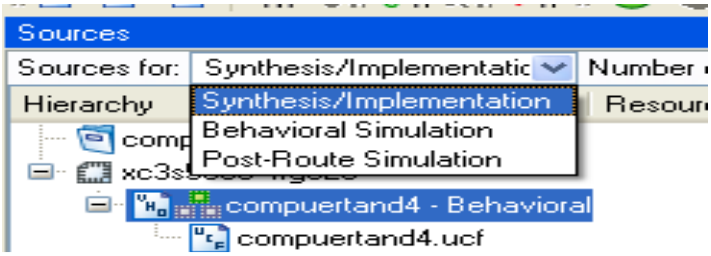


Figura 3.61

D Selecione *compuertand4.* en la ventana sources.
E En la ventana de procesos, haga click en signo “+” para desplegar el proceso **Generate Programming File** ver Figura 3.62

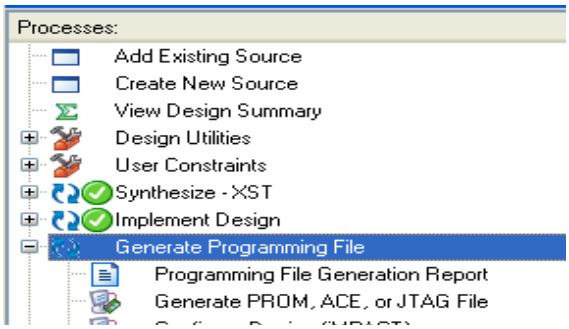


Figura 3.62 Abrir menu generacion archivo de programa

F Haga doble click en proceso **Configure Device (iMPACT)** para generar el archivo que se descargara en el kit de desarrollo. Ver figura 3.63

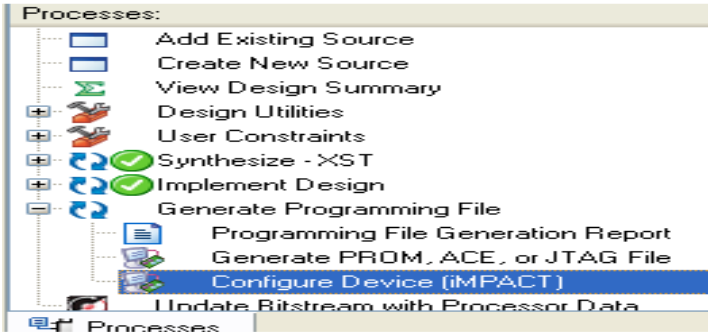


Figura 3.63 Generación archivo para descarga al kit

G La caja de dialogo de Xilinxs Web Talk puede abrirse durante este proceso. Haga click en **Decline.** Ver figura 3.64.

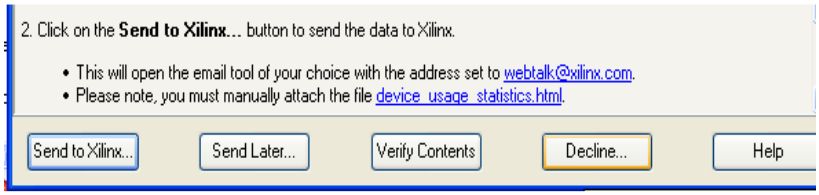


Figura 3.64

H Seleccione **"Disable the collection of device usage statistics fo this project only"** haga click en **OK.**Ver Figura 3.65

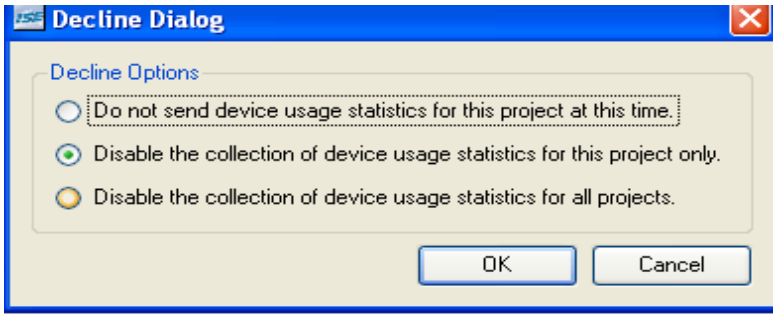


Figura 3.65

iMPACT .abre y la caja de dialogo de configurar dispositivo es visualizada
I En la caja de dialogo Welcome, seleccione **Configure devices using Boundary-Scan (JTAG)**.

J. Verifique que **Automatically connect to a cable and identify Boundary-Scan chain** es seleccionado.como se observa en la figura 3.66

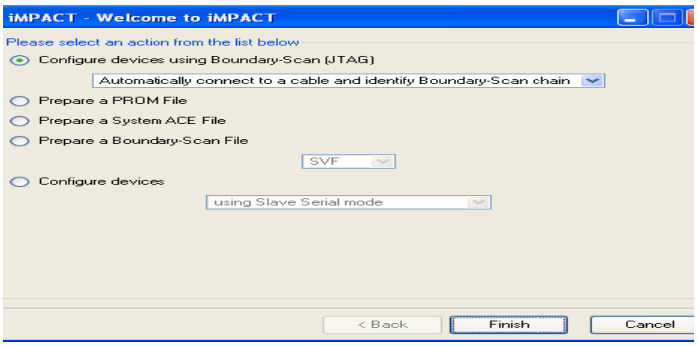
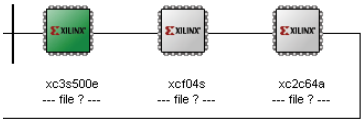


Figura 3.66 Conectar varios dispositivos por puerto JTAG

K. Haga click en **Finish**.

L. Si usted tiene un mensaje diciendo que allí hay dos dispositivos encontrados, haga click para continuar

Los dispositivos conectados a la cadena JTAG sobre la tarjeta pueden ser detectados y visualizados en la ventana IMPACT, para este caso el FPGA esta en primer lugar en color verde. Ver figura 3.67



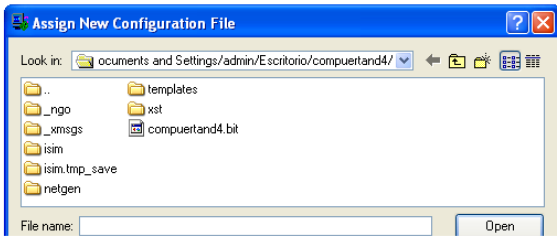


Figura 3.67 ubicación dispositivo FPGA en kit de desarrollo

M. La caja de dialogo **Assign New Configuration File** aparece. Para asignar un archivo de configuración al dispositivo XC3S500 en la cadena JTAG, Seleccione el archivo *compuertand4.*.bit y haga click en **Open** como se observa en la Figura 3.68.

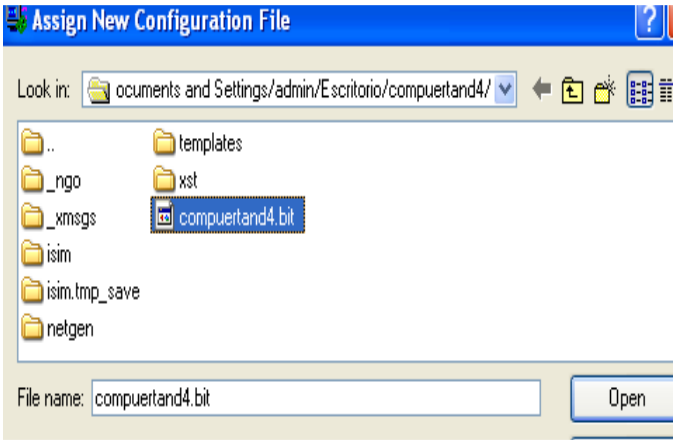


Figura 3.68 Asignar archivo .bit al FPGA

N. Si usted obtiene un mensaje de atención como el mostrado en la Figura 3.69, haga click en **OK**

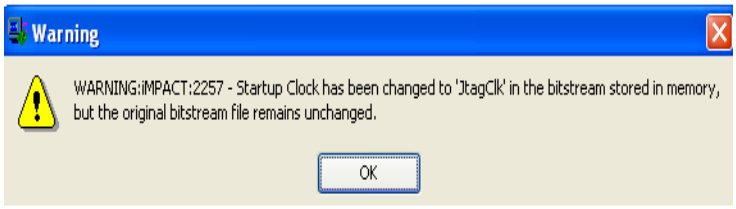


Figura 3.69

Nota: Como para este ejemplo no se van utilizar otros dispositivos detectados en el kit de desarrollo se le da click al boton bypass como se sugiere a continuación.
O. Seleccione **Bypass** para pasar por alto el segundo dispositivo detectado en la cadena el cual aparece ahora en verde como se muestra en la figura 3.70

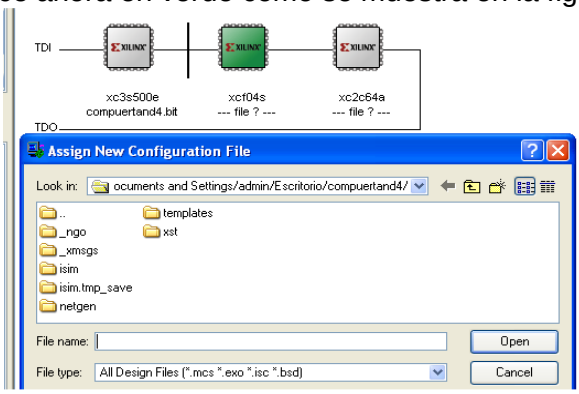


Figura 3.70 Asignacion archivo.bit a dispositivo 2

Ahora el tercer dispositivo en la cadena es resaltado en verde, seleccione nuevamente **Bypass** para pasarlo por alto como se indica en la Figura 3.71

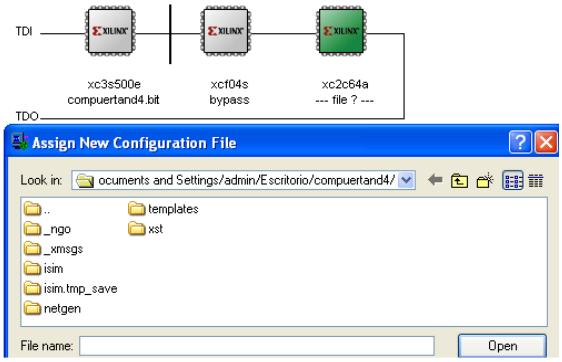


Figura 3.71 Asignacion archivo.bit a dispositivo 3

P.Haga click con el boton derecho del mouse sobre el icono del dispositivo XC3S500 y seleccione la opcion, **Program.** Ver figura 3.72

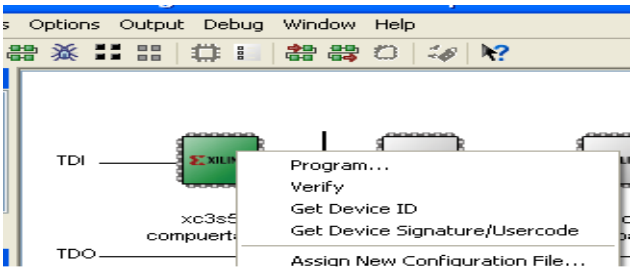


Figura 3.72 Programar FPGA ubicada sobre el kit

. La caja de dialogo **Programming Properties** se abre como muestra la Figura 3.73

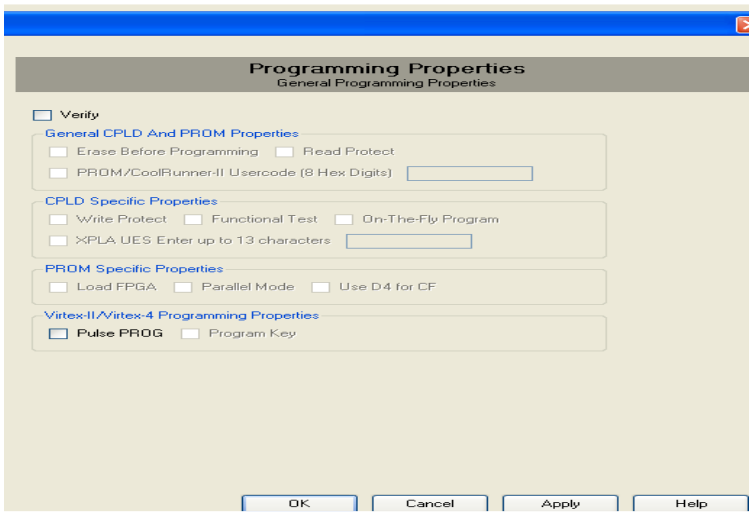


Figura 3.73 Propiedades de programación FPGA

Q. Haga click en **OK** para programar el dispositivo XC3S500.Cuando la programación es completada, el mensaje **Program Succeeded** es visualizado **Ver Figura 3.74**

Figura 3.74Programa descargado en FPGA XC3S500

Sobre el kit de desarrollo la salida LD-D (LED amarillo) se ilumina indicando que la FPGA esta programada.Ver Figura 3.75

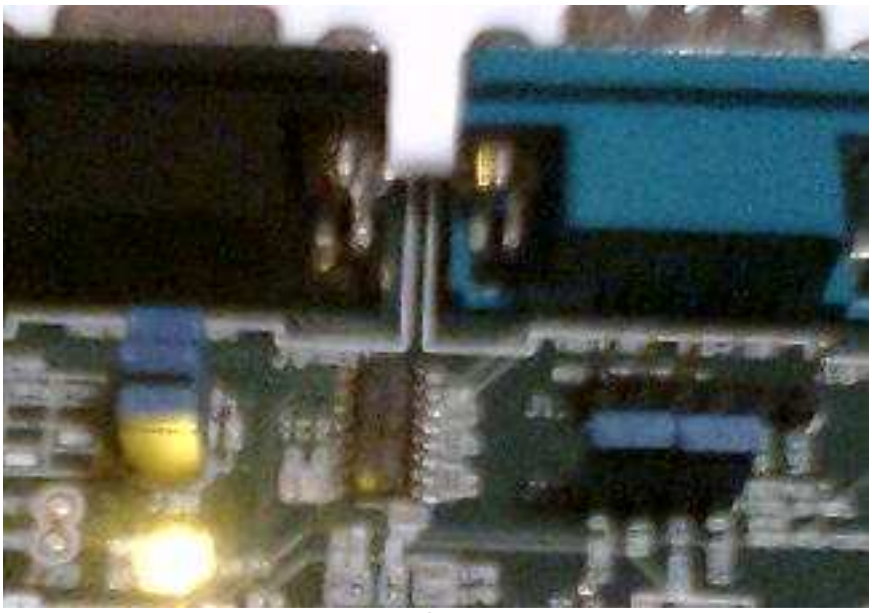


Figura 3.75 Indicación FPGA programada

En este momento puede jugar con las entradas de la tarjeta y ver su efecto sobre la salida. (teniendo en cuenta que el programa se descargo).ahora si las cuatro entradas estan en uno el LED 0 se ilumina y cambia según se modifiquen los swiches SW0,SW1,SW2, SW3.En la Figura 3.76 se indica cuando se activa o desactiva un swiche.

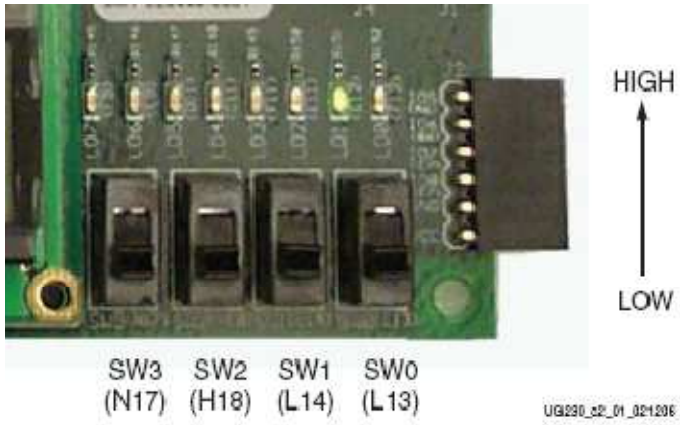


Figura 3.76 Activación swiches

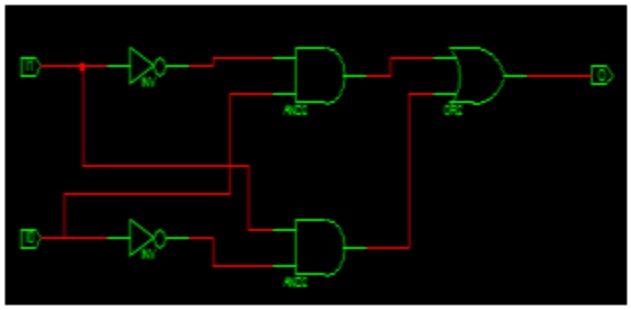
R Cierre iMPACT sin salvar.

Aquí termino el proyecto # 1.de la compuerta AND de 4 entradas, si desacopla el cable USB el kit de desarrollo mantendra la configuración mientras no desenergize el mismo.

4 PROYECTO # 2 : Compuerta XOR 2 entradas, componentes discretos

Realizar una compuerta XOR a partir de componentes discretos como indica la Figura 4.1

Esquema de compuerta XOR discreta



B	A	salida
0	0	0
0	1	1
1	0	1
1	1	0

tabla de verdad

Figura 4.1 Compuerta XOR 2 entradas

Para crear esta compuerta XOR se necesitan dos compuertas AND, dos inversores y una compuerta OR, por esta razón se debe crear un componente por cada clase, es decir una compuerta AND, una compuerta OR y un inversor para luego ser integrados en un solo proyecto y así crear la compuerta XOR.

4.1 Crear compuerta OR 2 entradas.

Para crear este componente realice el mismo procedimiento utilizado en el proyecto #1 de la compuerta AND 4 entradas teniendo encuenta lo siguiente:

Nombre del proyecto : *gateor*
Propiedades del proyecto: Son las mismas mostradas en la figura 3.6 del proyecto *compuertand4*.
Cree el archivo fuente HDL como indica la sección 3.3 de este curso básico, en la sección 3.3.E defina las entradas y salidas según se indica en la Figura 4.2 :

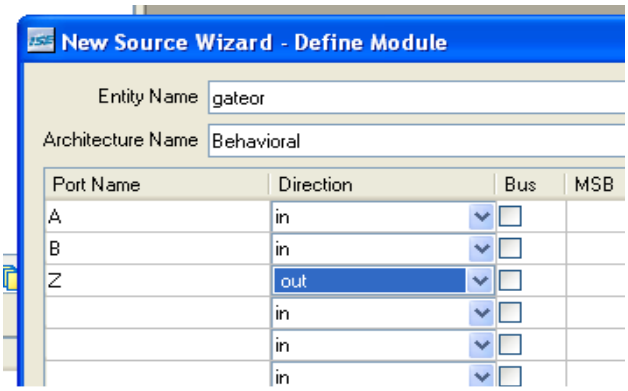


Figura 4.2 Declarar entradas y salidas del proyecto

Utilice las plantillas VHDL para realizar edición de la compuerta OR, como indica la sección 3.4, realice la edición final como muestra la figura 4.3

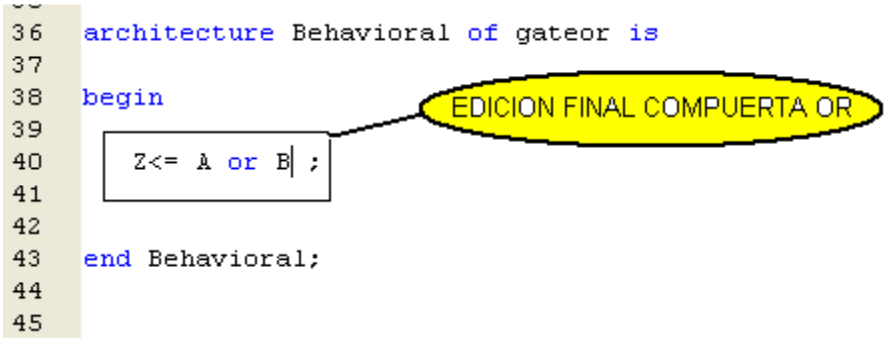


Figura 4.3 Edición final compuerta OR

Ahora realice verificación de la sintaxis según procedimiento descrito en la sección 3.6 de este curso.Como es necesario crear un banco de prueba siga la instrucción de la sección 3.7 , modifique las entradas y la salida como muestra la figura 4.4.

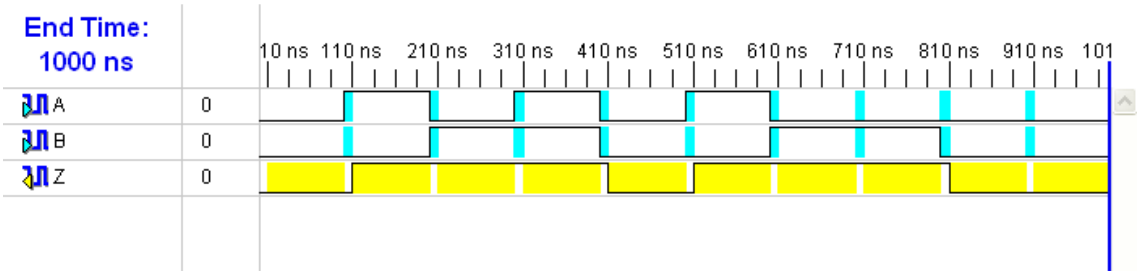


Figura 4.4 banco de prueba para compuerta OR

como se creo el banco de prueba con la salida esperada, realice un auto chequeo según procedimiento sección 3.8 D, así obtendra una salida como Indica la figura 4.5

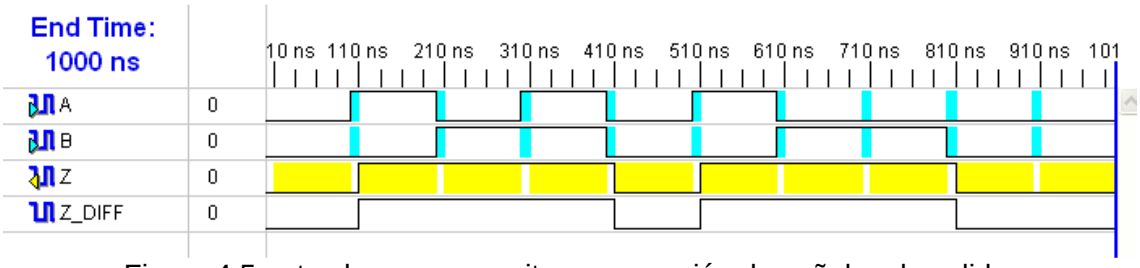


Figura 4.5 auto chequeo, permite comparación de señales de salida

En la figura 4.5 observe que la salida **Z** coincide con la salida **Z_DIFF** que era lo esperado en el comportamiento del circuito,.finalmente podremos realizar una simulación del diseño siguiendo las indicaciones de la sección 3.9 de este curso, debe generar una salida como la mostrada en la figura 4.6

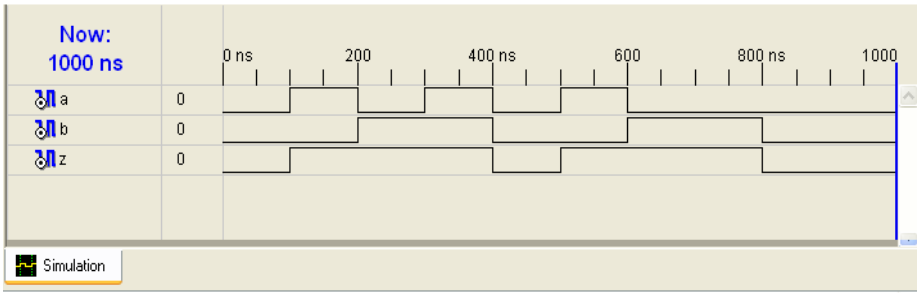
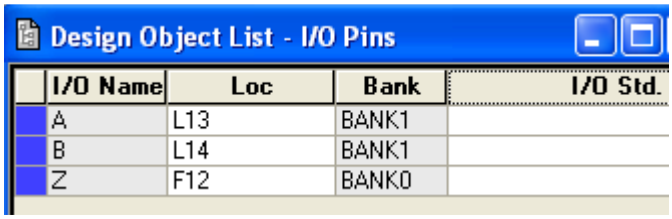


Figura 4.6 simulación compuerta OR

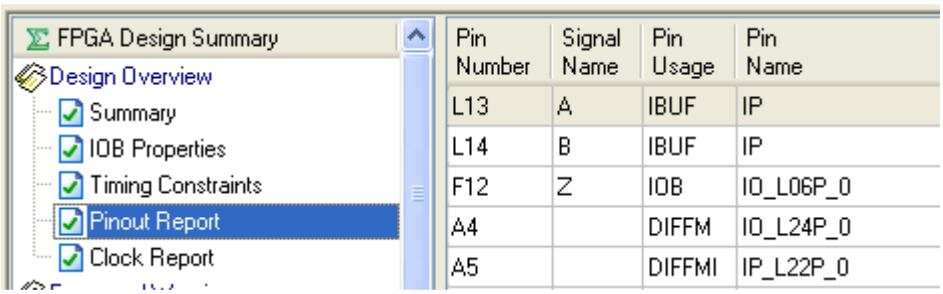
Ahora cree el archivo UCF de tiempos de sincronizacion según indica la sección 3.10 del curso. Teniendo en cuenta que este proyecto es de lógica combinacional, no requiere configurar señales de reloj.
Para continuar con este proyecto debe implementar el diseño como se indica en la sección 3.11, luego se realiza la asignación de pines del diseño con respecto al kit de desarrollo siguiendo la sección 3.12, asignando los pines como muestra la figura 4.7



	I/O Name	Loc	Bank	I/O Std.
	A	L13	BANK1	
	B	L14	BANK1	
	Z	F12	BANK0	

Figura 4.7 Asignación entradas y salidas en kit de desarrollo

Cuando se realiza la asignación de pines o modifica alguna asignación es necesario reimplementar el diseño como indica la sección 3.13 verificando en el reporte de asignación como indica en la figura 4.8



Pin Number	Signal Name	Pin Usage	Pin Name
L13	A	IBUF	IP
L14	B	IBUF	IP
F12	Z	IOB	IO_L06P_0
A4		DIFFM	IO_L24P_0
A5		DIFFMI	IP_L22P_0

Figura 4.8 Resumen asignación pines

Ahora use el mismo auto chequeo test bench waveform que usted creo para verificar que la compuerta OR diseñada después de esto a sido completamente implementada. Verifique que los tiempos de simulación operan dentro del sincronismo especificado después que retardos lógicos y enrutamientos son acomodados.
Realice la simulación como indica la sección 3.14, el resultado esperado debe ser como muestra la figura 4.9. Note que en el diagrama de tiempos aparecen los retardos lógicos luego de ser implementado el circuito, esto permite revisar si cumple con los requerimientos reales del diseño

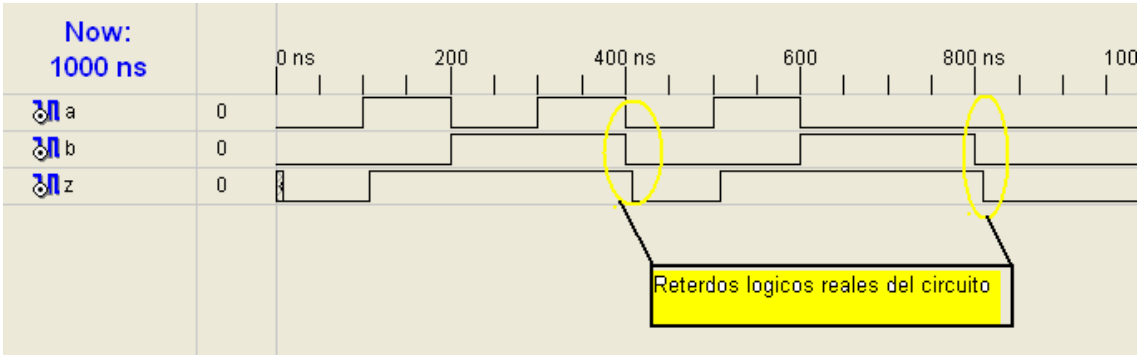


Figura 4.9 Simulación final del diseño

Finalmente debemos descargar el diseño de la compuerta OR al kit de desarrollo y comprobar su funcionamiento, siga los pasos descritos en la sección 3.15, cuando el led amarillo LD-0 se active indicara que el kit de desarrollo ya tiene cargado el diseño creado por usted, en este caso la compuerta OR, por lo tanto usted puede manipular y verificar su operación según la asignación de entradas y salidas previamente configuradas.
Cierre el proyecto en el software de desarrollo ISE 8.2i ,en el kit de desarrollo borre el proyecto descargado presionando el pulsador **PROG** ubicado cerca de los puertos seriales DCE y DTE;ahora se tiene un proyecto llamado *gateor* el cual lo reutilizaremos mas adelante en el desarrollo del proyecto de la compuerta XOR.

4.2 Crear compuerta AND 2 entradas

Para crear este componente realice el mismo procedimiento utilizado en el proyecto #1 de la compuerta AND 4 entradas teniendo encuenta lo siguiente:

Nombre del proyecto: *gateand*
Propiedades del proyecto: Son las mismas mostradas en la figura 3.6 del proyecto compuertand4..
Cree el archivo fuente HDL como indica la sección 3.3 de este curso básico, en la sección 3.3.E defina las entradas y salidas según se indica en la Figura 4.10:

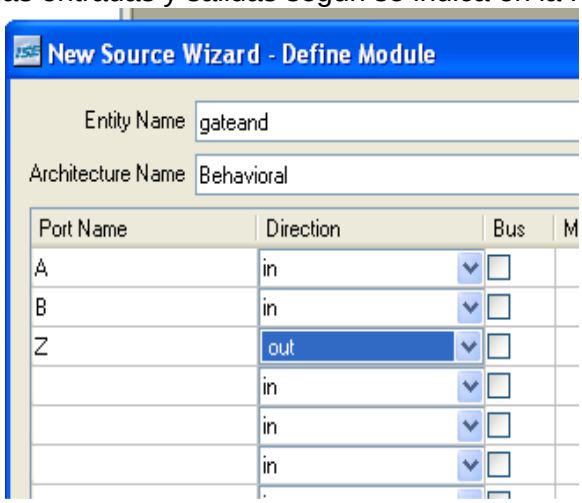


Figura 4.10 Declarar entradas y salidas del proyecto

Utilice las plantillas VHDL para realizar edición de la compuerta AND, como indica la sección 3.4, realice la edición final como muestra la figura 4.11

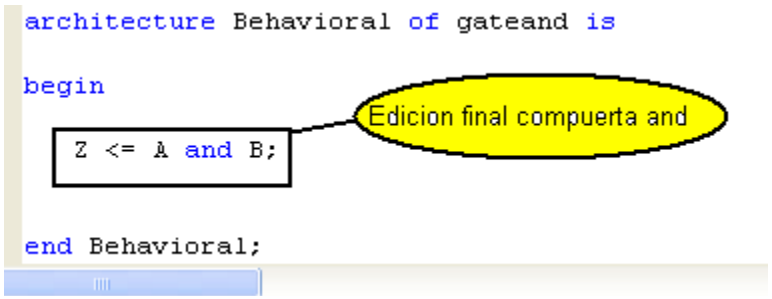


Figura 4.11 Edición final compuerta AND

Ahora realice verificación de la sintaxis según procedimiento descrito en la sección 3.6 de este curso.Como es necesario crear un banco de prueba siga la instrucción de la sección 3.7 , modifique las entradas y la salida como muestra la figura 4.12.

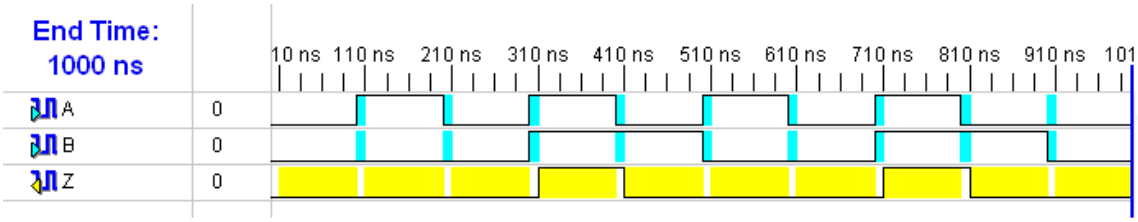


Figura 4.12 banco de prueba para compuerta AND

como se creo el banco de prueba con la salida esperada, realice un auto chequeo según procedimiento sección 3.8 D, así obtendra una salida como Indica la figura 4.13

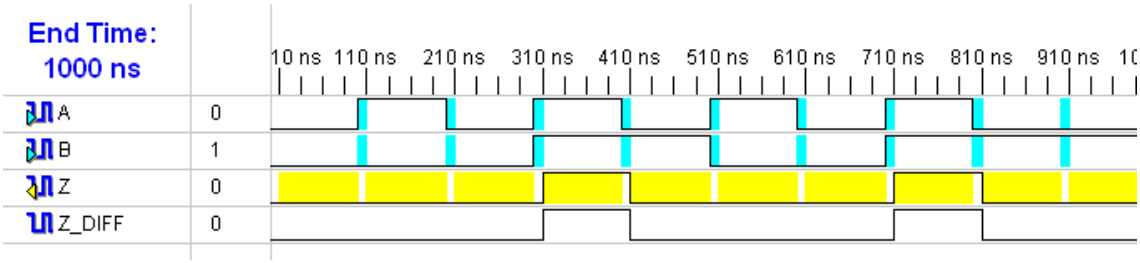


Figura 4.13 auto chequeo, permite comparación de señales de salida

En la figura 4.13 observe que la salida **Z** coincide con la salida **Z_DIFF** que era lo esperado en el comportamiento del circuito,.finalmente podremos realizar una simulación del diseño siguiendo las indicaciones de la sección 3.9 de este curso, debe generar una salida como la mostrada en la figura 4.14

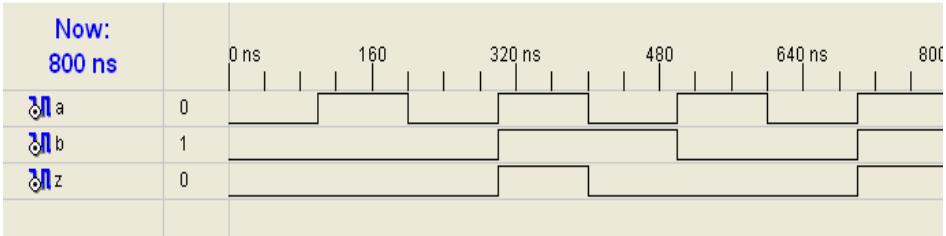


Figura 4.14 simulación compuerta AND

Ahora cree el archivo UCF de tiempos de sincronizacion según indica la sección 3.10 del curso. Teniendo en cuenta que este proyecto es de lógica combinacional, no requiere configurar señales de reloj.

Para continuar con este proyecto debe implementar el diseño como se indica en la sección 3.11,luego se realiza la asignación de pines del diseño con respecto al kit de desarrollo siguiendo la sección 3.12, asignando los pines como muestra la figura 4.15

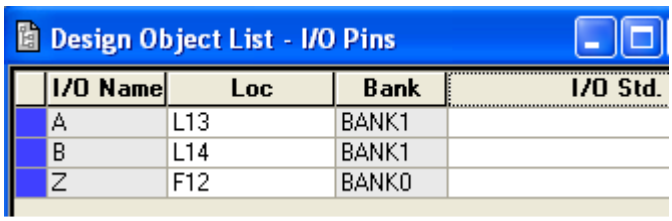


Figura 4.15 Asignación entradas y salidas en kit de desarrollo

Cuando se realiza la asignación de pines o modifica alguna asignación es necesario reinplementar el diseño como indica la sección 3.13 verificando en el reporte de asignación como indica en la figura 4.16

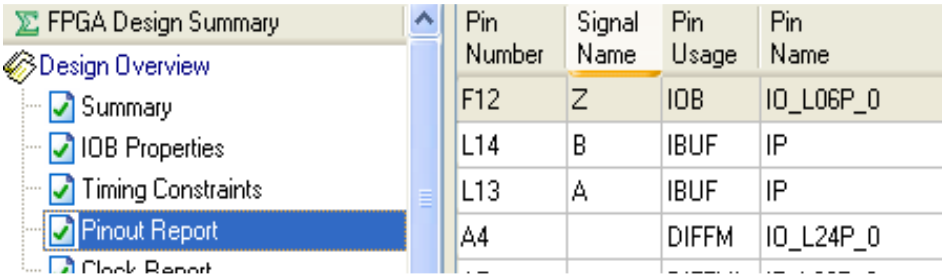


Figura 4.16 Resumen asignación pines

Ahora use el mismo auto chequeo test bench waveform que usted creo para verificar que la compuerta AND diseñada después de esto a sido completamente implementada. Verifique que los tiempos de simulación operan dentro del sincronismo especificado después que retardos lógicos y enrutamientos son acomodados.
Realice la simulación como indica la sección 3.14, el resultado esperado debe ser como muestra la figura 4.17. Note que en el diagrama de tiempos aparecen los retardos lógicos luego de ser implementado el circuito, esto permite revisar si cumple con los requerimientos reales del diseño

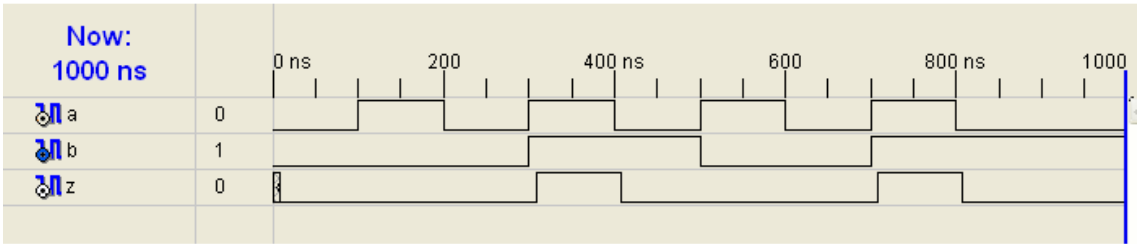


Figura 4.17 Simulación final del diseño

Finalmente debemos descargar el diseño de la compuerta AND al kit de desarrollo y comprobar su funcionamiento, siga los pasos descritos en la sección 3.15, cuando el led amarillo LD-0 se active indicara que el kit de desarrollo ya tiene cargado el diseño creado por usted, en este caso la compuerta AND, por lo tanto usted puede manipular y verificar su operación según la asignación de entradas y salidas previamente configuradas.
Cierre el proyecto en el software de desarrollo ISE 8.2i, en el kit de desarrollo borre el proyecto descargado presionando el pulsador **PROG** ubicado cerca de los puertos

seriales DCE y DTE; ahora se tiene un proyecto llamado *gateand* el cual lo reutilizaremos mas adelante en el desarrollo del proyecto de la compuerta XOR.

4.3 Crear inversor

Para crear este componente realice el mismo procedimiento utilizado en el proyecto #1 de la compuerta AND 4 entradas teniendo encuenta lo siguiente:

Nombre del proyecto: *inversor*
Propiedades del proyecto: Son las mismas mostradas en la figura 3.6 del proyecto compuertand4..
Cree el archivo fuente HDL como indica la sección 3.3 de este curso básico, en la sección 3.3.E defina la entradas y salida según se indica en la Figura 4.18:

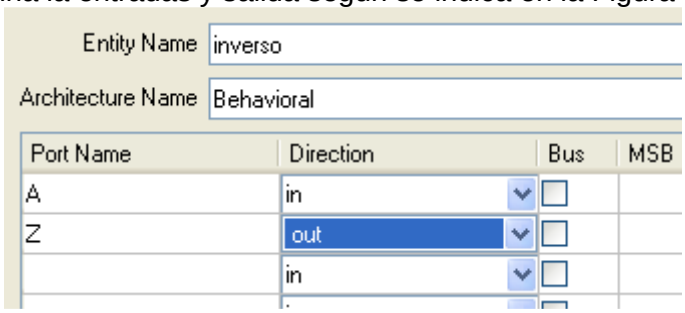


Figura 4.18 Declarar entrada y salida del proyecto

Utilice las plantillas VHDL para realizar edición del inversor, como indica la sección 3.4, realice la edición final como muestra la figura 4.19

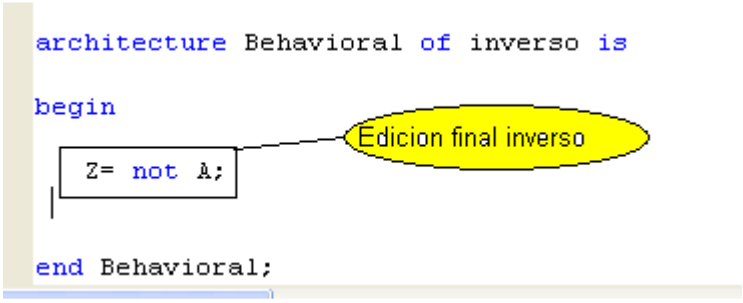


Figura 4.19 Edición final inversor

Ahora realice verificación de la sintaxis según procedimiento descrito en la sección 3.6 de este curso.Como es necesario crear un banco de prueba siga la instrucción de la sección 3.7 , modifique las entrada y la salida como muestra la figura 4.20.

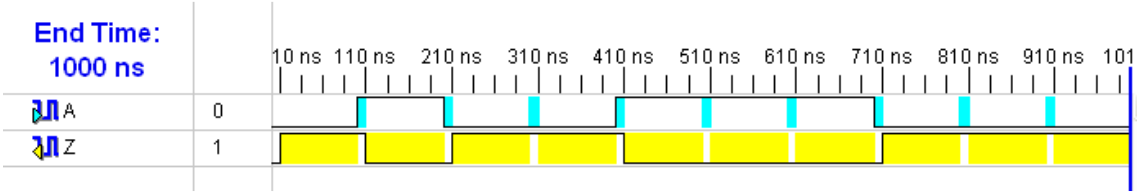


Figura 4.20 banco de prueba para inversor

Como se creo el banco de prueba con la salida esperada, realice un auto chequeo según procedimiento sección 3.8 D, así obtendra una salida como Indica la figura 4.21

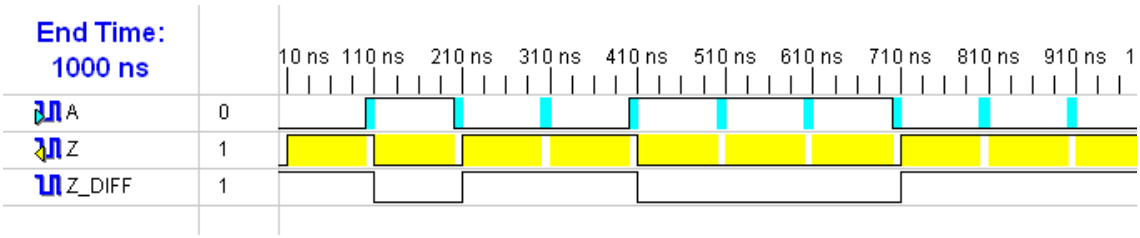


Figura 4.21 auto chequeo, permite comparación de señales de salida

En la figura 4.21 observe que la salida **Z** coincide con la salida **Z_DIFF** que era lo esperado en el comportamiento del circuito,finalmente podremos realizar una simulación del diseño siguiendo las indicaciones de la sección 3.9 de este curso, debe generar una salida como la mostrada en la figura 4.22

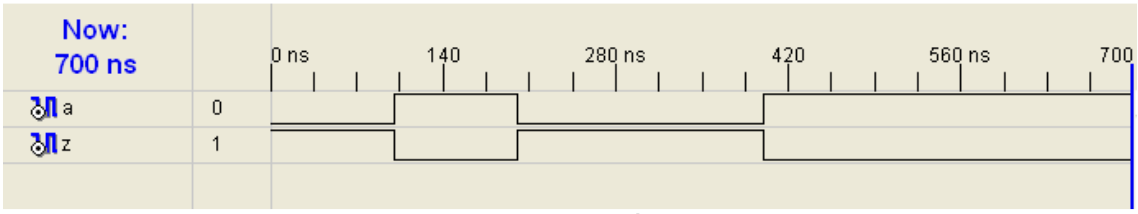


Figura 4.22 Simulación inversor

Ahora cree el archivo UCF de tiempos de sincronizacion según indica la sección 3.10 del curso. Teniendo en cuenta que este proyecto es de lógica combinacional no requiere configurar señales de reloj.
Para continuar con este proyecto debe implementar el diseño como se indica en la sección 3.11, luego se realiza la asignación de pines del diseño con respecto al kit de desarrollo siguiendo la sección 3.12, asignando los pines como muestra la figura 4.23

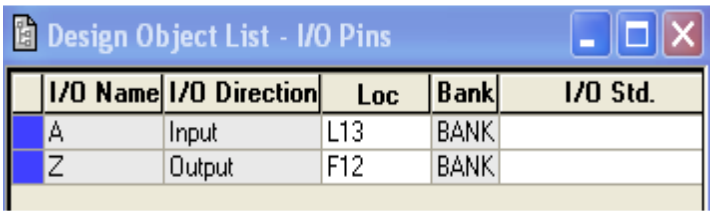


Figura 4.23 Asignación entrada y salida en kit de desarrollo

Cuando se realiza la asignación de pines o modifica alguna asignación es necesario reimplementar el diseño como indica la sección 3.13 verificando en el reporte de asignación como indica en la figura 4.24

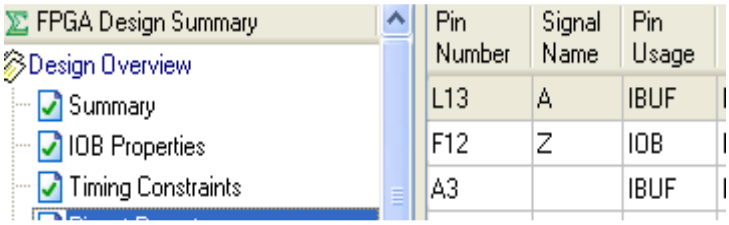


Figura 4.24 Resumen asignación pines

Ahora use el mismo auto chequeo test bench waveform que usted creo para verificar que el inversor diseñado después de esto a sido completamente implementado. Verifique que los tiempos de simulación operan dentro del sincronismo especificado después que retardos lógicos y enrutamientos son acomodados.

Realice la simulación como indica la sección 3.14, el resultado esperado debe ser como muestra la figura 4.25. Note que en el diagrama de tiempos aparecen los retardos lógicos luego de ser implementado el circuito, esto permite revisar si cumple con los requerimientos reales del diseño

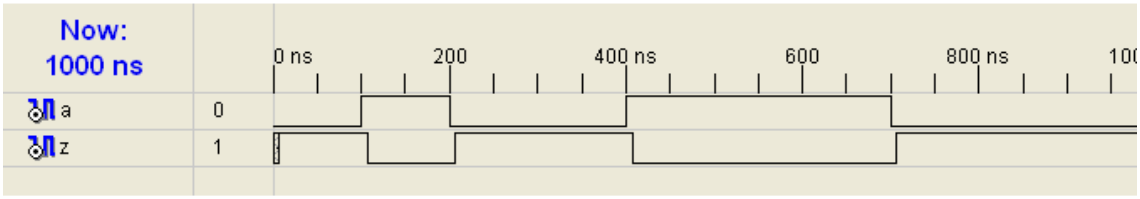


Figura 4.25 Simulación final del diseño

Finalmente debemos descargar el diseño del la inversor al kit de desarrollo y comprobar su funcionamiento, siga los pasos descritos en la sección 3.15, cuando el led amarillo LD-0 se active indicara que el kit de desarrollo ya tiene cargado el diseño creado por usted, en este caso el inversor, por lo tanto usted puede manipular y verificar su operación según la asignación de entradas y salidas previamente configuradas.
Cierre el proyecto en el software de desarrollo ISE 8.2i, en el kit de desarrollo borre el proyecto descargado presionando el pulsador **PROG** ubicado cerca de los puertos seriales DCE y DTE; ahora se tiene un proyecto llamado *inverso* el cual lo reutilizaremos acontinuación en el desarrollo del proyecto de la compuerta XOR.

4.4 Crear compuerta XOR discreta de dos entradas

El objetivo de este proyecto es crear una compuerta XOR discreta, por esta razón es necesario reutilizar los proyectos *gateor, gateand e inverso* previamente creados, como componentes del nuevo proyecto. Acontinuacion se describe el procedimiento el cual varía con respecto a los proyectos hasta ahora realizados en algunos pasos.

Para crear el proyecto realice el mismo procedimiento utilizado en el proyecto #1 de la compuerta AND de 4 entradas teniendo encuenta lo siguiente:

Nombre del proyecto: *gatexor2*
Propiedades del proyecto: Son las mismas mostradas en la figura 3.6 del proyecto *compuertand4.*
Cree el archivo fuente HDL como indica la sección 3.3 de este curso básico, en la sección 3.3.E defina las entradas y salidas según se indica en la Figura 4.26:

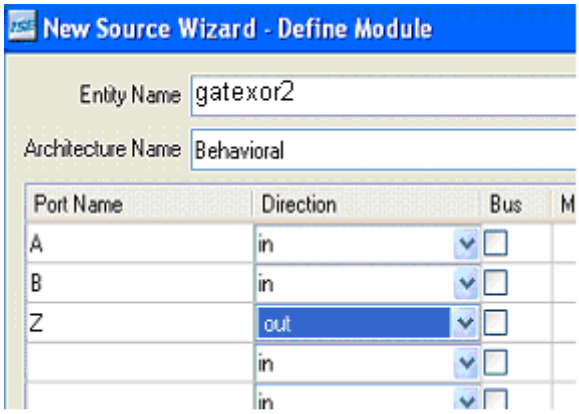


Figura 4.26 Declarar entradas y salida del proyecto

Ahora debemos adicionar los componentes *gateor, gateand e inverso* para realizar la edición de la compuerta XOR discreta. Para adicionar componentes en la ventana **sources** seleccione el módulo VHD creado como indica la figura 4.27

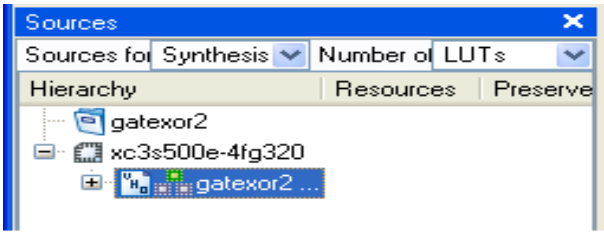


Figura 4.27 proyecto seleccionado para adicionar componentes

Ahora para adicionar componentes en la pestaña **Project** seleccione **Add Source** como muestra la Figura 4.28

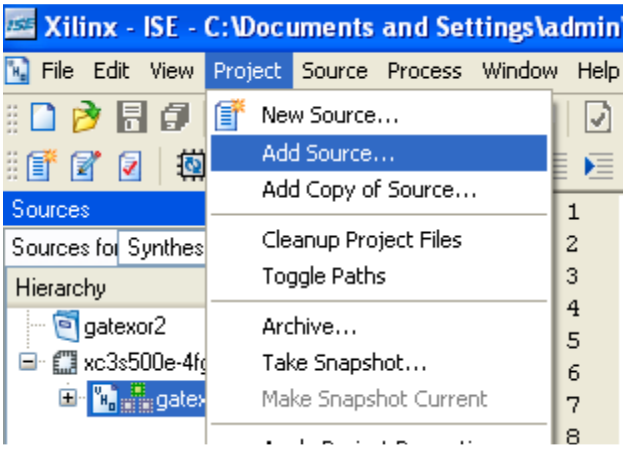
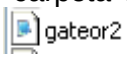


Figura 4.28 Modo para adicionar componentes

En la ventana que se abre ubicar carpeta con el proyecto *gateor*, busque el archivo *gateor* tipo VHD, cuyo icono es 

, como indica la Figura 4.29, haga click para abrirlo

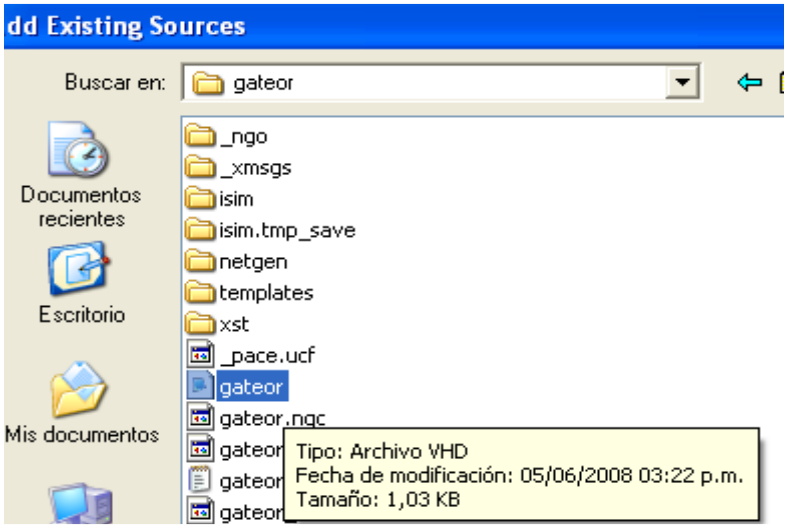


Figura 4.29 Selección de componentes para adicionar al proyecto

En la ventana que aparece como indica la Figura 4.30, haga click en **OK** para adicionar este proyecto como un componente de la compuerta XOR

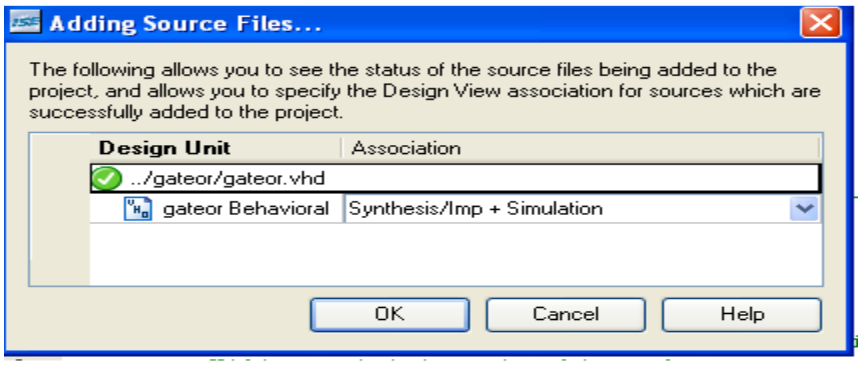


Figura 4.30 Adición de archivo como componente del proyecto

Haga el mismo ejercicio para adicionar el archivo *gateand e inverso,* al finalizar esta adición en la ventana **Sources** se mostrara el proyecto como indica la figura 4.31.

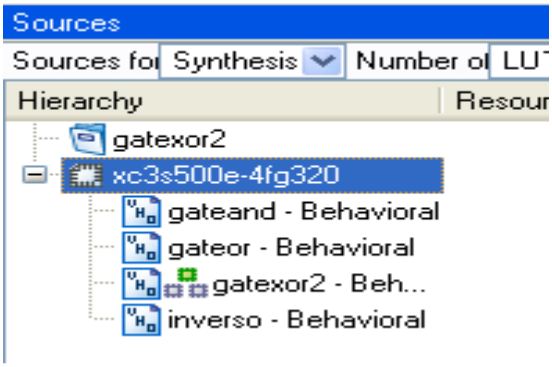


Figura 4.31 Lista de componentes adicionados al proyecto

Ahora seleccione **gatexor2-Behavioral** de la lista, realice la edición final como muestra la figura 4.32

```
36  architecture Behavioral of gatexor2 is
37
38      component gateor
39              Port ( A : in  STD_LOGIC;
40                     B : in  STD_LOGIC;
41                     Z : out  STD_LOGIC);
42      end component;
43
44      component gateand
45              Port ( A : in  STD_LOGIC;
46                     B : in  STD_LOGIC;
47                     Z : out  STD_LOGIC);
48      end component;
49
50      component inverso
51              Port ( A : in  STD_LOGIC;
52                     Z : out  STD_LOGIC);
53      end component;
54
55      signal m,n,x,v:STD_LOGIC;
56
57      begin
58          etiq1:inverso port map(B,x);
59          etiq2:inverso port map(A,v);
60          etiq3:gateand port map(A,x,m);
61          etiq4:gateand port map(v,B,n);
62          etiq5:gateor port map(m,n,Z);
63
64  end Behavioral;
65
```

Figura 4.32 Edición final compuerta XOR discreta

Esta edición final se realizo teniendo como referencia el esquema mostrado en la figura 4.33, donde se observa que se dan nombres a las salidas de los inversores y compuertas and con las letras **x,v,m,n** respectivamente para facilitar la misma,

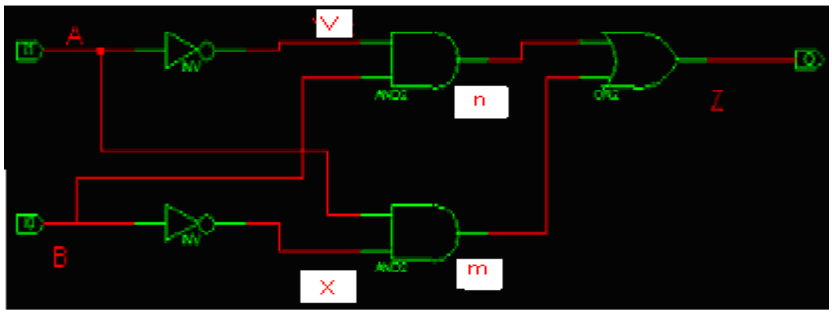


Figura 4.33 Diseño discreto de la compuerta XOR

Observe en la edición de la compuerta XOR, que se inicia por relacionar las compuertas OR, AND e INVERSOR como componentes, aquí el nombre de cada componente es el mismo dado al proyecto inicialmente creado, ejemplo la compuerta OR fue llamada *gateor* y sus entradas marcadas como A,B y su salida Z, así se deben editar para evitar errores de sintaxis como indica la figura 4.34

```
component gateor
      Port ( A : in  STD_LOGIC;
             B : in  STD_LOGIC;
             Z : out  STD_LOGIC);
end component;
```

Figura 4.34 Edición de un componente

En la edición tambien se generan señales intermedias llamadas ***signal m,n,x,v:STD_LOGIC***; las cuales permiten describir el circuito, la edición de cada conexión de describe como

```
etiq3:gateand port map(A,x,m);
```

Aquí **etiq3** es el numero de conexión (tramo de cable), **gateand** es el nombre del componente el cual estamos interconectando, **port map (A,x,m)** indica que sus entradas son **A,x** su salida es llamada **m**, como se observa en la figura 4.32,. igualmente esta figura muestra que existen 5 conexciones(tramos de cable), por esta razón aparecen 5 etiquetas para describir la compuerta XOR exclusiva.

Ahora realice verificación de la sintaxis según procedimiento descrito en la sección 3.6 de este curso. Al finalizar la ventana **sources** mostrara el proyecto como se indica en la figura 4.35 donde se observa que el bloque principal es el archivo fuente gatexor2 y los componentes adicionados son subdirectorios.

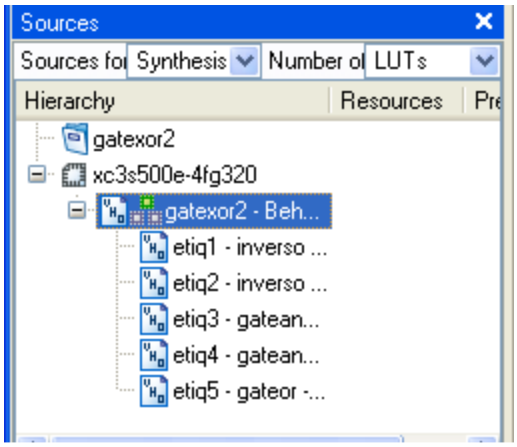


Figura 4.35 jerarquia de los componentes

Como es necesario crear un banco de prueba siga la instrucción de la sección 3.7, teniendo encuenta de seleccionar el archivo principal llamado *gatexor2* mostrado en la figura 4.35 modifique las entradas y la salida como muestra la figura 4.36.

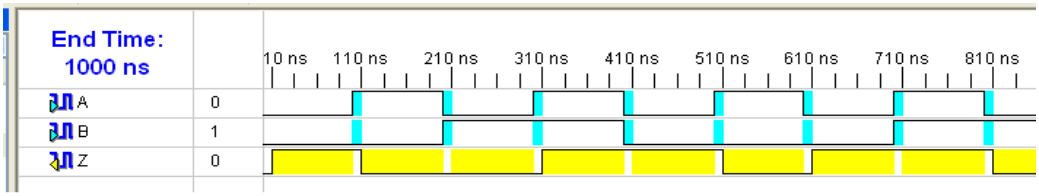


Figura 4.36 banco de prueba compuerta XOR

como se creo el banco de prueba con la salida esperada, realice un auto chequeo según procedimiento sección 3.8 D, así obtendra una salida como Indica la figura 4.37

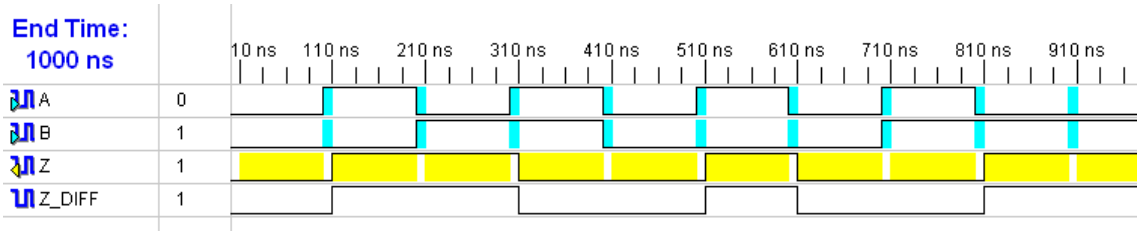


Figura 4.37 auto chequeo, permite comparación de señales de salida

En la figura 4.37 observe que la salida **Z** coincide con la salida **Z_DIFF** que era lo esperado en el comportamiento del circuito,.finalmente podremos realizar una simulación del diseño siguiendo las indicaciones de la sección 3.9 de este curso, debe generar una salida como la mostrada en la figura 4.38

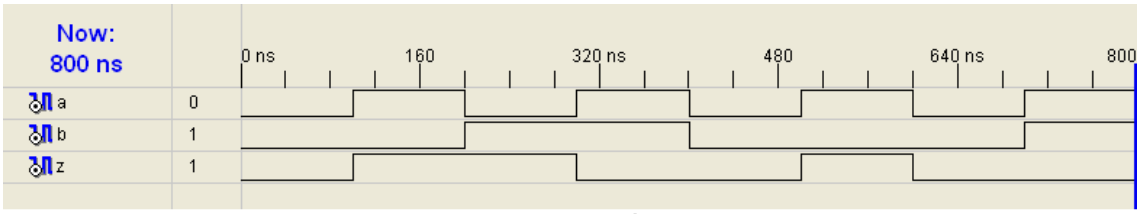
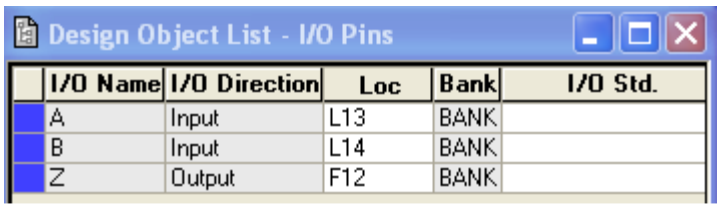


Figura 4.38 simulación inversor

Ahora cree el archivo UCF de tiempos de sincronizacion según indica la sección 3.10 del curso. Teniendo en cuenta que este proyecto es de lógica combinacional no requiere configurar señales de reloj.

Para continuar con este proyecto debe implementar el diseño como se indica en la sección 3.11, luego se realiza la asignación de pines del diseño con respecto al kit de desarrollo siguiendo la sección 3.12, asignando los pines como muestra la figura 4.39



	I/O Name	I/O Direction	Loc	Bank	I/O Std.
	A	Input	L13	BANK	
	B	Input	L14	BANK	
	Z	Output	F12	BANK	

Figura 4.39 Asignación entradas y salidas en kit de desarrollo

Cuando se realiza la asignación de pines o modifica alguna asignación es necesario reimplementar el diseño como indica la sección 3.13 verificando en el reporte de asignación como indica en la figura 4.40

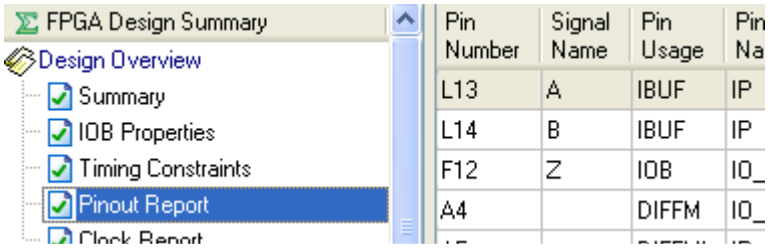


Figura 4.40 Resumen asignación pines

Ahora use el mismo auto chequeo test bench waveform que usted creo para verificar que la compuerta XOR diseñada después de esto a sido completamente implementado. Verifique que los tiempos de simulación operan dentro del sincronismo especificado después que retardos lógicos y enrutamientos son acomodados.
Realice la simulación como indica la sección 3.14, el resultado esperado debe ser como muestra la figura 4.41. Note que en el diagrama de tiempos aparecen los retardos lógicos luego de ser implementado el circuito, esto permite revisar si cumple con los requerimientos reales del diseño

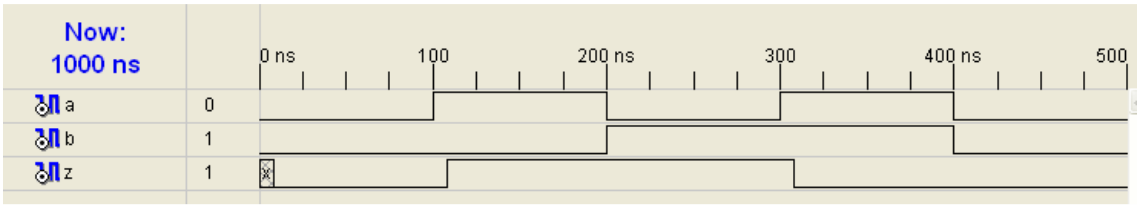


Figura 4.41 Simulación final del diseño

Finalmente debemos descargar el diseño del la compuerta XOR al kit de desarrollo y comprobar su funcionamiento, siga los pasos descritos en la sección 3.15, cuando el led amarillo LD-0 se active indicara que el kit de desarrollo ya tiene cargado el diseño creado por usted, en este caso la compuerta XOR, por lo tanto usted puede manipular y verificar su operación según la asignación de entradas y salidas previamente configuradas y la tabla de verdad mostrada en la figura 4.1.
Cierre el proyecto en el software de desarrollo.

5 PROYECTO # 3 COMPARADOR DE 1 BIT

Realizar comparador de un bit como muestra la figura 5.1, su funcionamiento es el siguiente:

Si la entrada A es mayor que B se activa la salida R
Si la entrada A es igual que B se activa la salida S
Si la entrada A es menor que B se activa la salida T

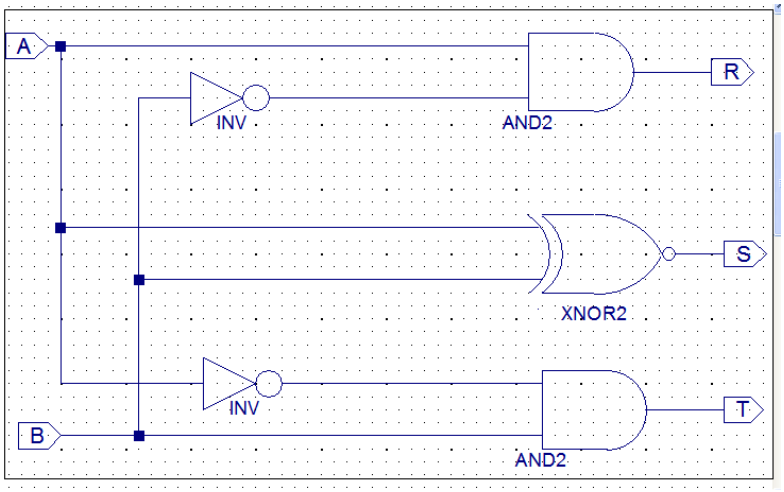


Figura 5.1 Comparador de un bit

Para crear este comparador se necesitan dos compuertas AND, dos inversores y una compuerta XNOR, cuando se realizo el proyecto de la compuerta XOR se utilizaron como componentes la compuerta AND y el Inversor, pero no existe el componente compuerta XNOR, por esta razón se debe crear para luego ser integrados en el proyecto del comparador de 1 bit

5.1 Crear compuerta XNOR 2 entradas.

Para crear este componente realice el mismo procedimiento utilizado en el proyecto #1 de la compuerta AND 4 entradas teniendo encuenta lo siguiente:

Nombre del proyecto : *gatexnor2*
Propiedades del proyecto: Son las mismas mostradas en la figura 3.6 del proyecto *compuertand4*.
Cree el archivo fuente HDL como indica la sección 3.3 de este curso básico, en la sección 3.3.E defina las entradas y salidas según se indica en la Figura 5.2:

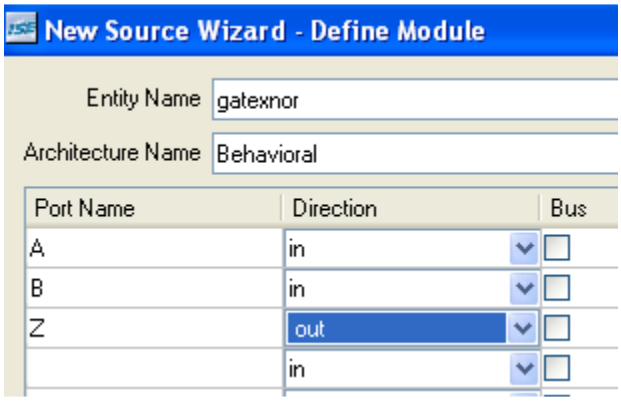


Figura 5.2 Declarar entradas y salidas del proyecto

Utilice las plantillas VHDL para realizar edición de la compuerta OR, como indica la sección 3.4, realice la edición final como muestra la figura 5.3

```
36  architecture Behavioral of gatexnor is
37
38  begin
39
40      Z <= A xnor B ;
41
42  end Behavioral;
43
44
```

Edicion final compuerta XNOR

Figura 5.3 Edición final compuerta XNOR

Ahora realice verificación de la sintaxis según procedimiento descrito en la sección 3.6 de este curso.Como es necesario crear un banco de prueba siga la instrucción de la sección 3.7 , modifique las entradas y la salida como muestra la figura 5.4.

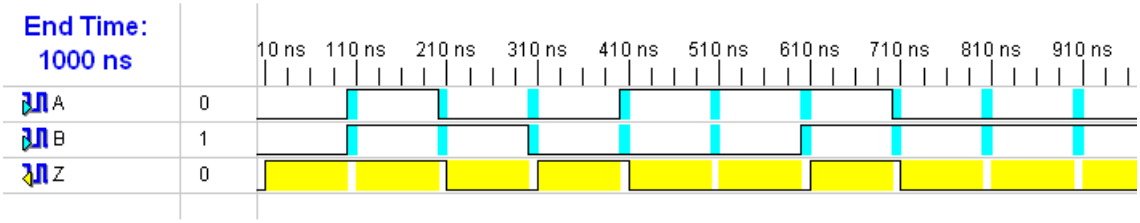


Figura 5.4 banco de prueba para compuerta XNOR

como se creo el banco de prueba con la salida esperada, realice un auto chequeo según procedimiento sección 3.8 D, así obtendra una salida como Indica la figura 5.5

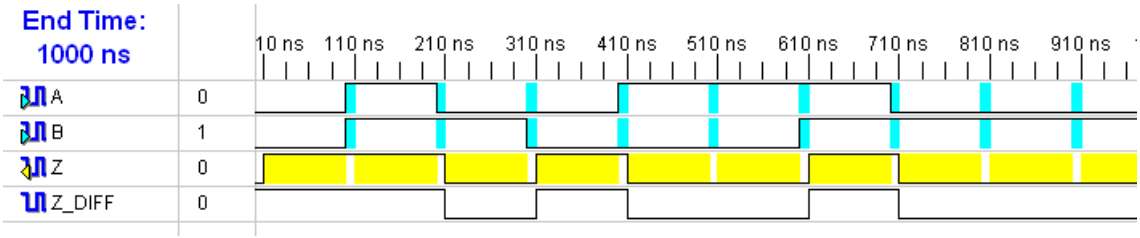


Figura 5.5 Auto chequeo, permite comparación de señales de salida

En la figura 5.5 observe que la salida **Z** coincide con la salida **Z_DIFF** que era lo esperado en el comportamiento del circuito,.finalmente podremos realizar una simulación del diseño siguiendo las indicaciones de la sección 3.9 de este curso, debe generar una salida como la mostrada en la figura 5.6

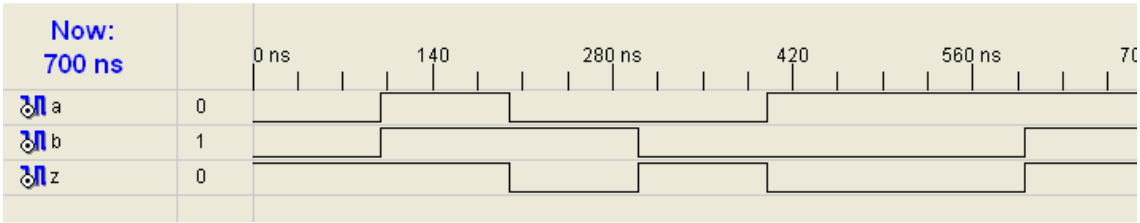


Figura 5.6 simulación compuerta XNOR

Ahora cree el archivo UCF de tiempos de sincronizacion según indica la sección 3.10 del curso. Teniendo en cuenta que este proyecto es de lógica combinacional, no requiere configurar señales de reloj.

Para continuar con este proyecto debe implementar el diseño como se indica en la sección 3.1, luego se realiza la asignación de pines del diseño con respecto al kit de desarrollo siguiendo la sección 3.12, asignando los pines como muestra la figura 5.7

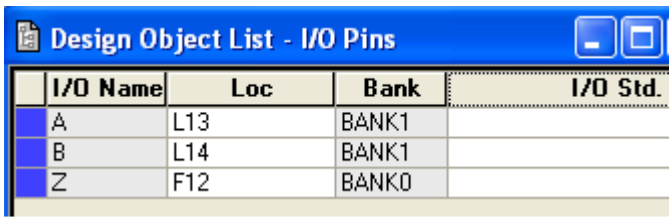


Figura 5.7 Asignación entradas y salidas en kit de desarrollo

Cuando se realiza la asignación de pines o modifica alguna asignación es necesario reimplementar el diseño como indica la sección 3.13 verificando en el reporte de asignación como indica en la figura 5.8

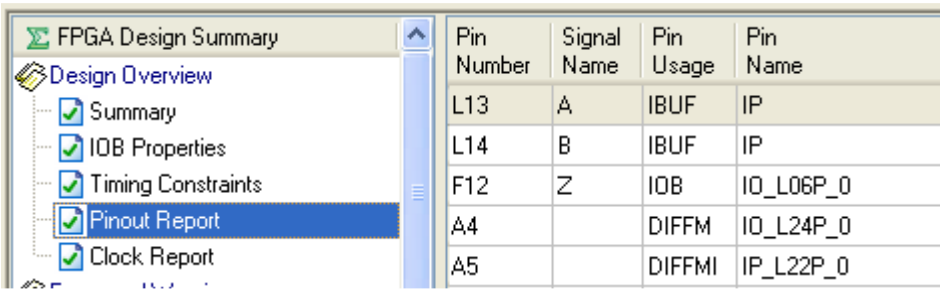


Figura 5.8 Resumen asignación pines

Ahora use el mismo auto chequeo test bench waveform que usted creo para verificar que la compuerta XNOR diseñada después de esto a sido completamente implementada. Verifique que los tiempos de simulación operan dentro del sincronismo especificado después que retardos lógicos y enrutamientos son acomodados.
Realice la simulación como indica la sección 3.14, el resultado esperado debe ser como muestra la figura 5.9. Note que en el diagrama de tiempos aparecen los retardos lógicos luego de ser implementado el circuito, esto permite revisar si cumple con los requerimientos reales del diseño

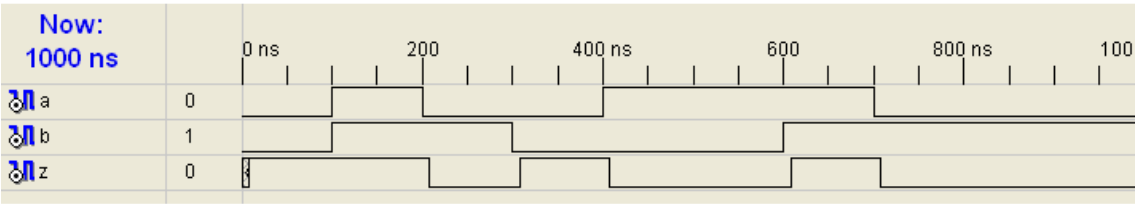


Figura 5.9 Simulación final del diseño

Finalmente debemos descargar el diseño de la compuerta XNOR al kit de desarrollo y comprobar su funcionamiento, siga los pasos descritos en la sección 3.15, cuando el led amarillo LD-0 se active indicara que el kit de desarrollo ya tiene cargado el diseño creado por usted, en este caso la compuerta XNOR, por lo tanto usted puede manipular y verificar su operación según la asignación de entradas y salidas previamente configuradas.
Cierre el proyecto en el software de desarrollo ISE 8.2i, en el kit de desarrollo borre el proyecto descargado presionando el pulsador **PROG** ubicado cerca de los puertos seriales DCE y DTE; ahora se tiene un proyecto llamado *gatexnor2* el cual se utiliza a continuación en el desarrollo del proyecto comparador de un bit.

5.2 Crear comparador de un BIT

El objetivo de este proyecto es crear comparador de un bit, por esta razón es necesario reutilizar los proyectos *gatexnor2, gateand e inverso* previamente creados, como componentes del nuevo proyecto. Acontinuacion se describe el procedimiento el cual varía con respecto a los proyectos hasta ahora realizados en algunos pasos.

Para crear el proyecto realice el mismo procedimiento utilizado en el proyecto #1 de la compuerta AND de 4 entradas teniendo encuenta lo siguiente:

Nombre del proyecto: *comparador1*
Propiedades del proyecto: Son las mismas mostradas en la figura 3.6 del proyecto *compuertand4.*
Cree el archivo fuente HDL como indica la sección 3.3 de este curso básico, en la sección 3.3.E defina las entradas y salidas según se indica en la Figura 5.10:

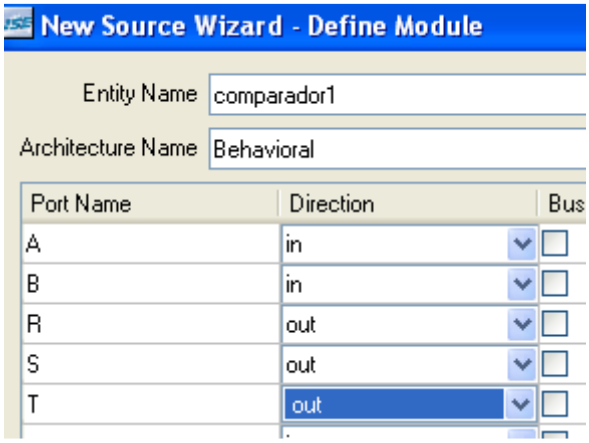


Figura 5.10 Declarar entradas y salida del proyecto

Ahora debemos adicionar los componentes *gatexnor2, gateand e inverso* para realizar la edición del comparador. Para adicionar componentes en la ventana **sources** seleccione el módulo VHD creado como indica la figura 5.11

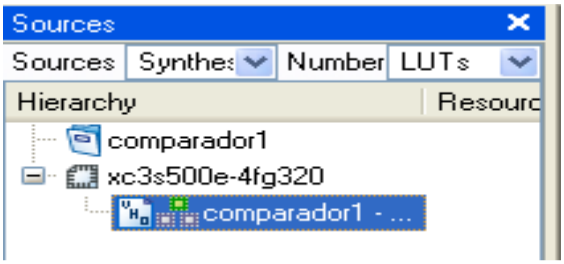


Figura 5.11 proyecto seleccionado para adicionar componentes

Ahora para adicionar componentes en la pestaña **Project** seleccione **Add Source** como muestra la Figura 5.12

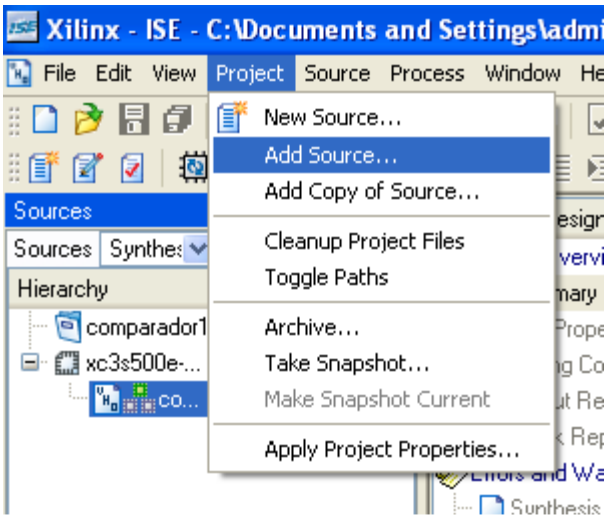
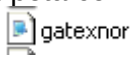


Figura 5.12 Modo para adicionar componentes

En la ventana que se abre ubicar carpeta con el proyecto *gatexnor2*, busque el archivo *gatexnor2* tipo VHD cuyo icono es 

, como indica la Figura 5.13, haga click para abrirlo

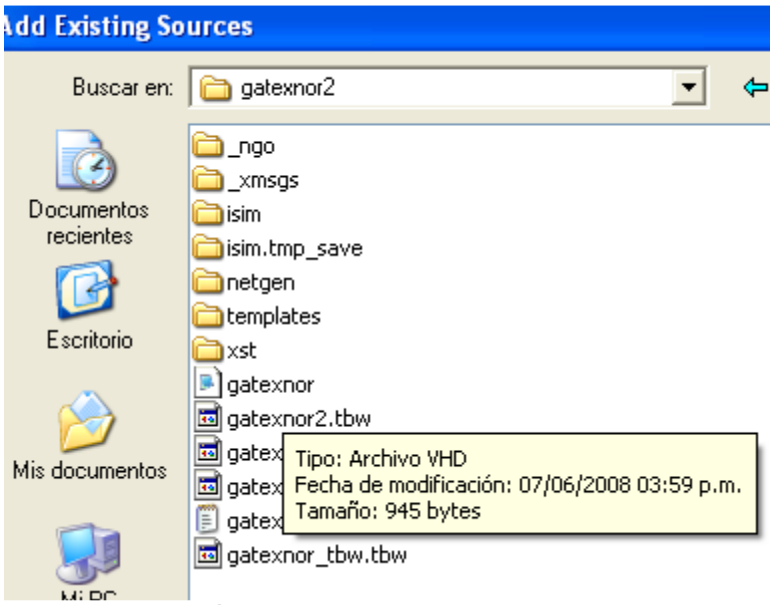


Figura 5.13 Selección de componentes para adicionar al proyecto

En la ventana que aparece como indica la Figura 5.14, haga click en **OK** para adicionar este proyecto como un componente del comparador.

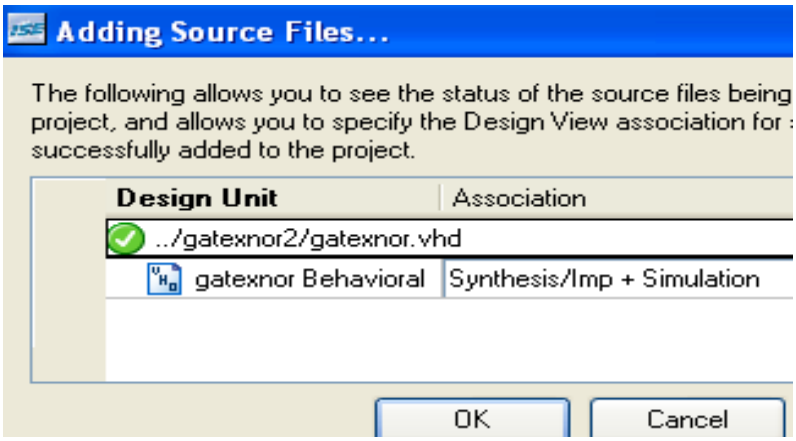


Figura 5.14 Adición de archivo como componente del proyecto

Haga el mismo ejercicio para adicionar el archivo *gateand e inverso,* al finalizar esta adición en la ventana **Sources** se mostrara el proyecto como indica la figura 5.15.

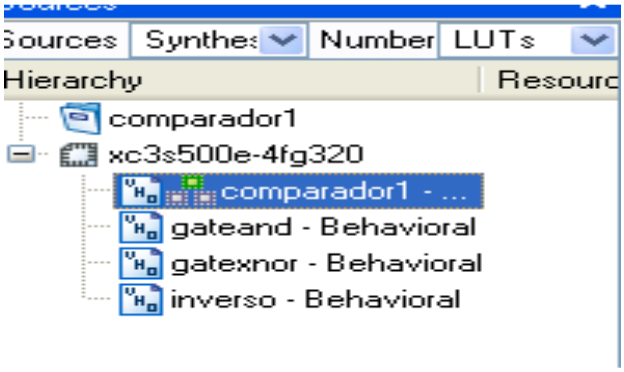


Figura 5.15 Lista de componentes adicionados al proyecto

Ahora seleccione **comparador1** de la lista, realice la edición final como muestra la figura 5.16

```
37
38  architecture Behavioral of comparador1 is
39
40  component gateand
41      Port ( A : in  STD_LOGIC;
42             B : in  STD_LOGIC;
43             Z : out  STD_LOGIC);
44  end component;
45
46  component inverso
47      Port ( A : in  STD_LOGIC;
48             Z : out  STD_LOGIC);
49  end component;
50
51  component gatexnor2
52      Port ( A : in  STD_LOGIC;
53             B : in  STD_LOGIC;
54             Z : out  STD_LOGIC);
55  end component;
56
57  signal m,n :STD_LOGIC;
58  begin
59  etiq1 : inverso port map(A,n);
60  etiq2 : inverso port map(B,m);
61  etiq3 : gateand port map(A,m,R);
62  etiq4 : gateand port map(B,n,T);
63  etiq5 : gatexnor2 port map(A,B,S);
64
65  end Behavioral;
```

Figura 5.16 Edición final comparador de un bit

Esta edición final se realizo teniendo como referencia el esquema mostrado en la figura 5.17, donde se observa que se dan nombres a las salidas de los inversores con las letras **m,n** para facilitar la misma,

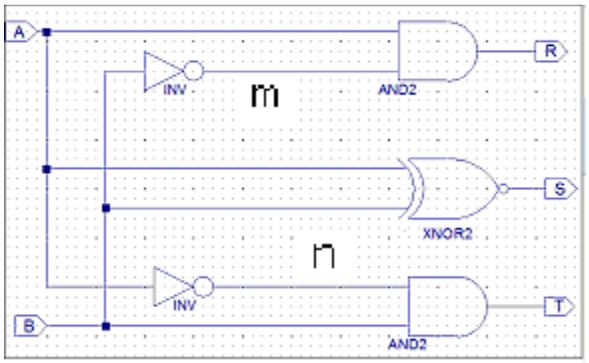


Figura 5.17 Diseño discreto del comparador

Observe en la edición del comparador, que se inicia por relacionar las compuertas AND, INVERSOR y XNOR como componentes, aquí el nombre de cada componente es el mismo dado al proyecto inicialmente creado, ejemplo la compuerta AND fue llamada *gateand* y sus entradas marcadas como A, B y su salida Z, así se deben editar para evitar errores de sintaxis como indica la figura 5.18

```
component gateand
    Port ( A : in  STD_LOGIC;
           B : in  STD_LOGIC;
           Z : out  STD_LOGIC);
end component;
```

Figura 5.18 Edición de un componente

En la edición tambien se generan señales intermedias llamadas ***signal m,n:STD_LOGIC***; las cuales permiten describir el circuito, la edición de cada conexión de describe como

```
etiq4 : gateand port map(B,n,T);
```

Aquí **etiq4** es el numero de conexión (tramo de cable), **gateand** es el nombre del componente el cual estamos interconectando, **port map (B,n,T)** indica que sus entradas son **B,n** su salida es llamada **T**, como se observa en la figura 5.17, igualmente esta figura muestra que existen 5 conexciones(tramos de cable), por esta razón aparecen 5 etiquetas para describir el comparador de un bit.

Ahora realice verificación de la sintaxis según procedimiento descrito en la sección 3.6 de este curso. Al finalizar la ventana **sources** mostrara el proyecto como se indica en la figura 5.19 donde se observa que el bloque principal es el archivo fuente comparador1 y los componentes adicionados son subdirectorios.

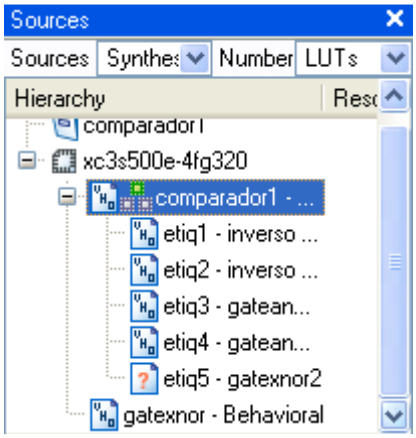


Figura 5.19 Jerarquia de los componentes

Como es necesario crear un banco de prueba siga la instrucción de la sección 3.7, teniendo encuenta de seleccionar el archivo principal llamado *comparador1* mostrado en la figura 5.19 modifique las entradas y la salida como muestra la figura 5.20.

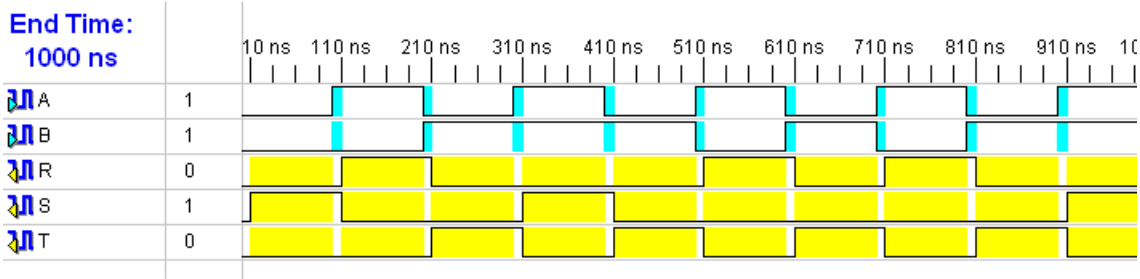


Figura 5.20 banco de prueba comparador1

Como se creo el banco de prueba con la salida esperada, realice un auto chequeo según procedimiento sección 3.8 D, así obtendra una salida como Indica la figura 5.21

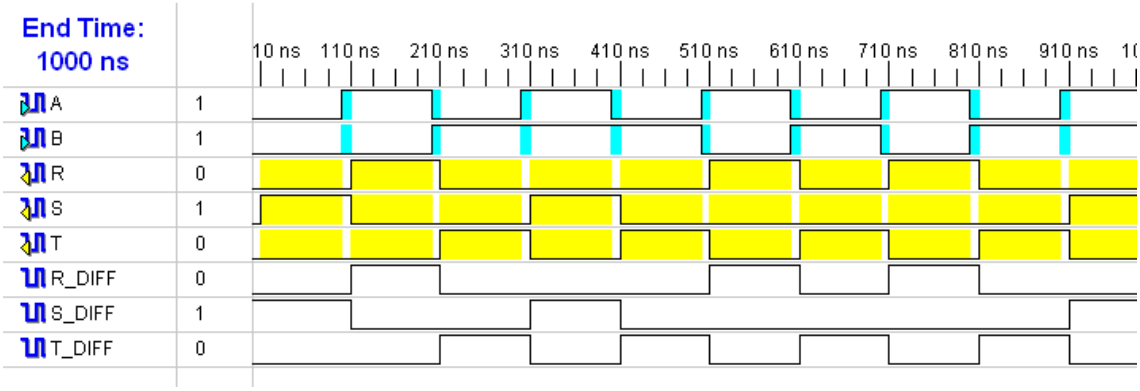


Figura 5.21 auto chequeo, permite comparación de señales de salida

En la figura 5.21 observe que las salida **R, S, T** coincide con la salida **R_DIFF, S_DIFF, T_DIFF** respectivamente que era lo esperado en el comportamiento del circuito,finalmente podremos realizar una simulación del diseño siguiendo las indicaciones de la sección 3.9 de este curso, debe generar una salida como la mostrada en la figura 5.22

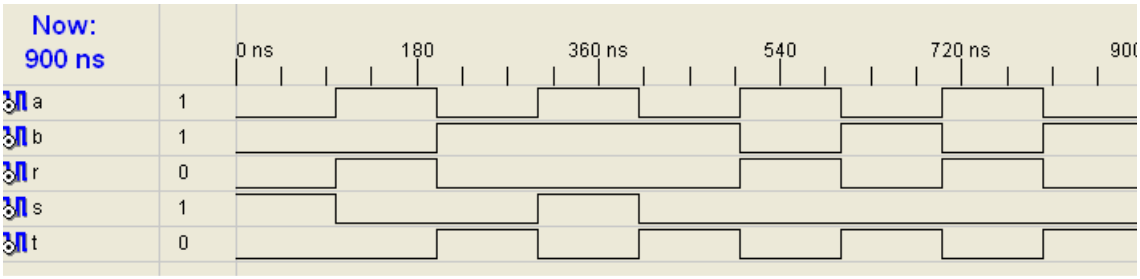
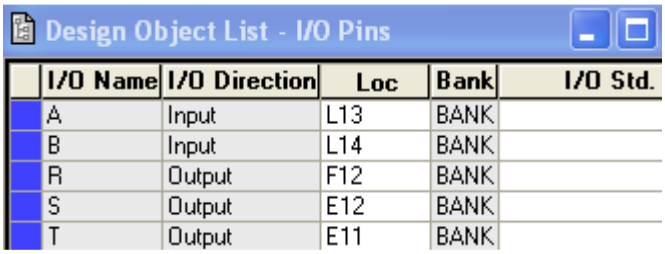


Figura 5.22 simulación comprador de un bit

Ahora cree el archivo UCF de tiempos de sincronizacion según indica la sección 3.10 del curso. Teniendo en cuenta que este proyecto es de lógica combinacional no requiere configurar señales de reloj.
Para continuar con este proyecto debe implementar el diseño como se indica en la sección 3.11, luego se realiza la asignación de pines del diseño con respecto al kit de desarrollo siguiendo la sección 3.12, asignando los pines como muestra la figura 5.23



I/O Name	I/O Direction	Loc	Bank	I/O Std.
A	Input	L13	BANK	
B	Input	L14	BANK	
R	Output	F12	BANK	
S	Output	E12	BANK	
T	Output	E11	BANK	

Figura 5.23 Asignación entradas y salidas en kit de desarrollo

Cuando se realiza la asignación de pines o modifica alguna asignación es necesario reimplementar el diseño como indica la sección 3.13 verificando en el reporte de asignación como indica en la figura 5.24

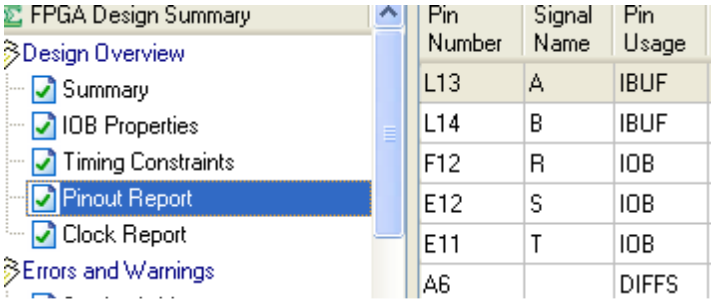


Figura 5.24 Resumen asignación pines

Ahora use el mismo auto chequeo test bench waveform que usted creo para verificar que la compuerta XOR diseñada después de esto a sido completamente implementado. Verifique que los tiempos de simulación operan dentro del sincronismo especificado después que retardos lógicos y enrutamientos son acomodados.
Realice la simulación como indica la sección 3.14, el resultado esperado debe ser como muestra la figura 5.25. Note que en el diagrama de tiempos aparecen los retardos lógicos luego de ser implementado el circuito, esto permite revisar si cumple con los requerimientos reales del diseño

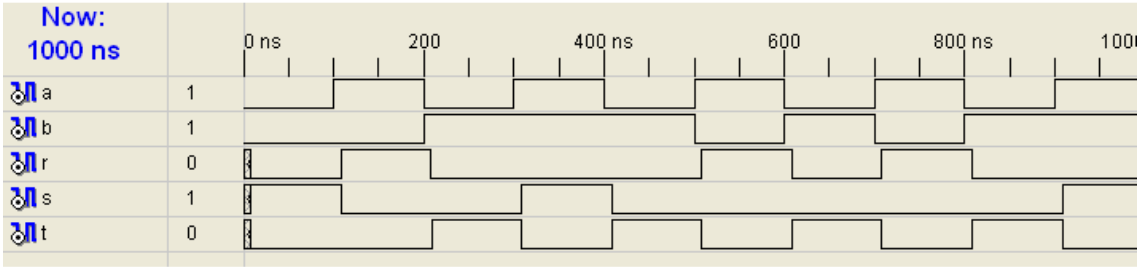


Figura 5.25 Simulación final del diseño

Finalmente debemos descargar el diseño del comparador de un bit al kit de desarrollo y comprobar su funcionamiento, siga los pasos descritos en la sección 3.15, cuando el led amarillo LD-0 se active indicara que el kit de desarrollo ya tiene cargado el diseño creado por usted, en este caso el comparador, por lo tanto puede manipular y verificar su operación según la asignación de entradas y salidas previamente configuradas.
Cierre el proyecto en el software de desarrollo.

5 PROYECTO # 4 CONTADOR BINARIO ASCENDENTE-DECENDENTE

El Contador ascendente –descendente tiene una entrada selectora que determina la dirección del conteo, cuenta hasta 32 binario cada vez que damos un pulso simulando la entrada de reloj.

6.1 Crear contador binario.

Para crear el contador realice el mismo procedimiento utilizado en el proyecto #1 de la compuerta AND 4 entradas teniendo encuenta lo siguiente:

Nombre del proyecto: *contador32*
Propiedades del proyecto: Son las mismas mostradas en la figura 3.6 del proyecto *compuertand4*.
Cree el archivo fuente HDL como indica la sección 3.3 de este curso básico, en la sección 3.3.E defina las entradas y salidas según se indica en la Figura 6.1:

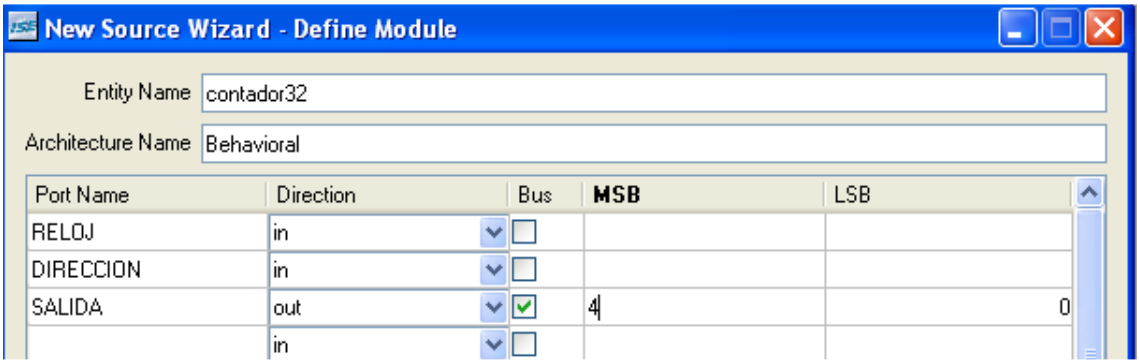


Figura 6.1 Declarar entradas y salidas del proyecto

Observe que SALIDA se configuro como vector, definiendo 5 posiciones; ahora utilice las plantillas VHDL para realizar edición del contador, ubique la plantilla como indica la figura 6.2, siguiendo las instrucciones de la sección 3.4

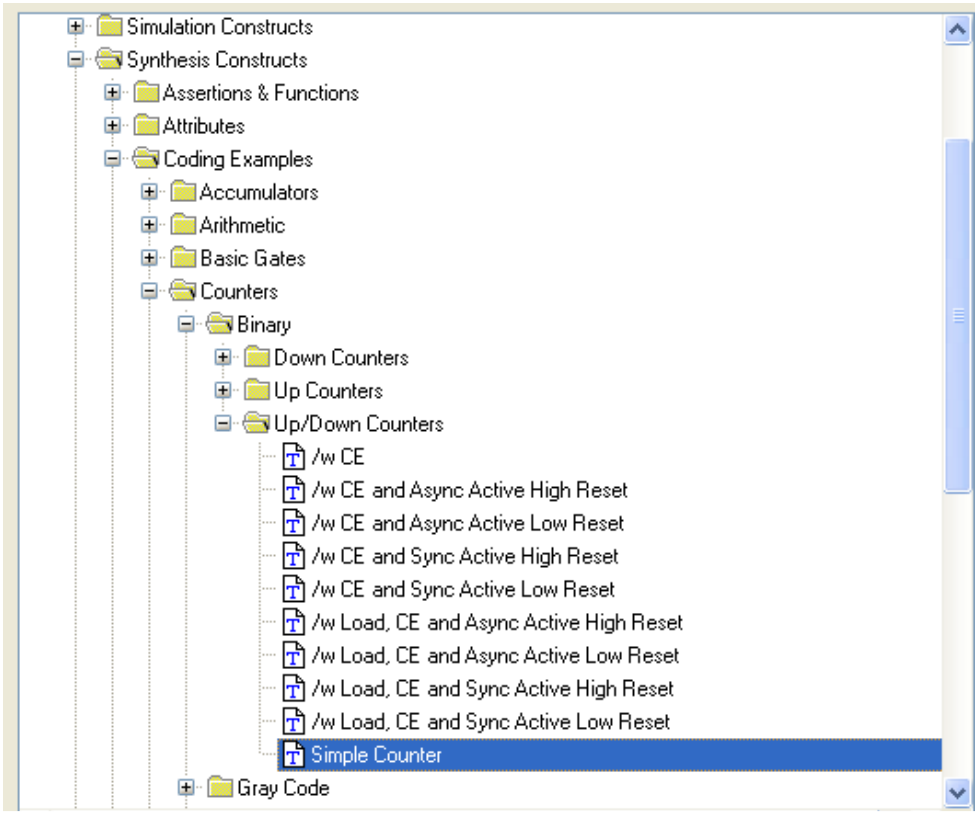


Figura 6.2 Plantilla para contador

Realice la edición final como muestra la figura 6.3

```
29
30  entity contador32 is
31      Port ( RELOJ : in  STD_LOGIC;
32             DIRECCION : in  STD_LOGIC;
33             SALIDA : out  STD_LOGIC_VECTOR (4 downto 0));
34  end contador32;
35
36  architecture Behavioral of contador32 is
37
38  SIGNAL CONTAR32: STD_LOGIC_VECTOR(4 DOWNTO 0):= "00000";
39
40  begin
41
42  process (RELOJ)
43  begin
44     if RELOJ='1' and RELOJ'event then
45        if DIRECCION='1' then
46           CONTAR32 <= CONTAR32 + 1;
47        else
48           CONTAR32 <= CONTAR32 - 1;
49        end if;
50     end if;
51  end process;
52   SALIDA<=CONTAR32;
53
54  end Behavioral;
55
```

Figura 6.3 Edición final contador

Ahora ejecute verificación de la sintaxis según procedimiento descrito en la sección 3.6 de este curso.Como es necesario crear un banco de prueba siga la instrucción de la sección 3.7.

Configure la ventana de tiempos de inicialización como indica la figura 6.4

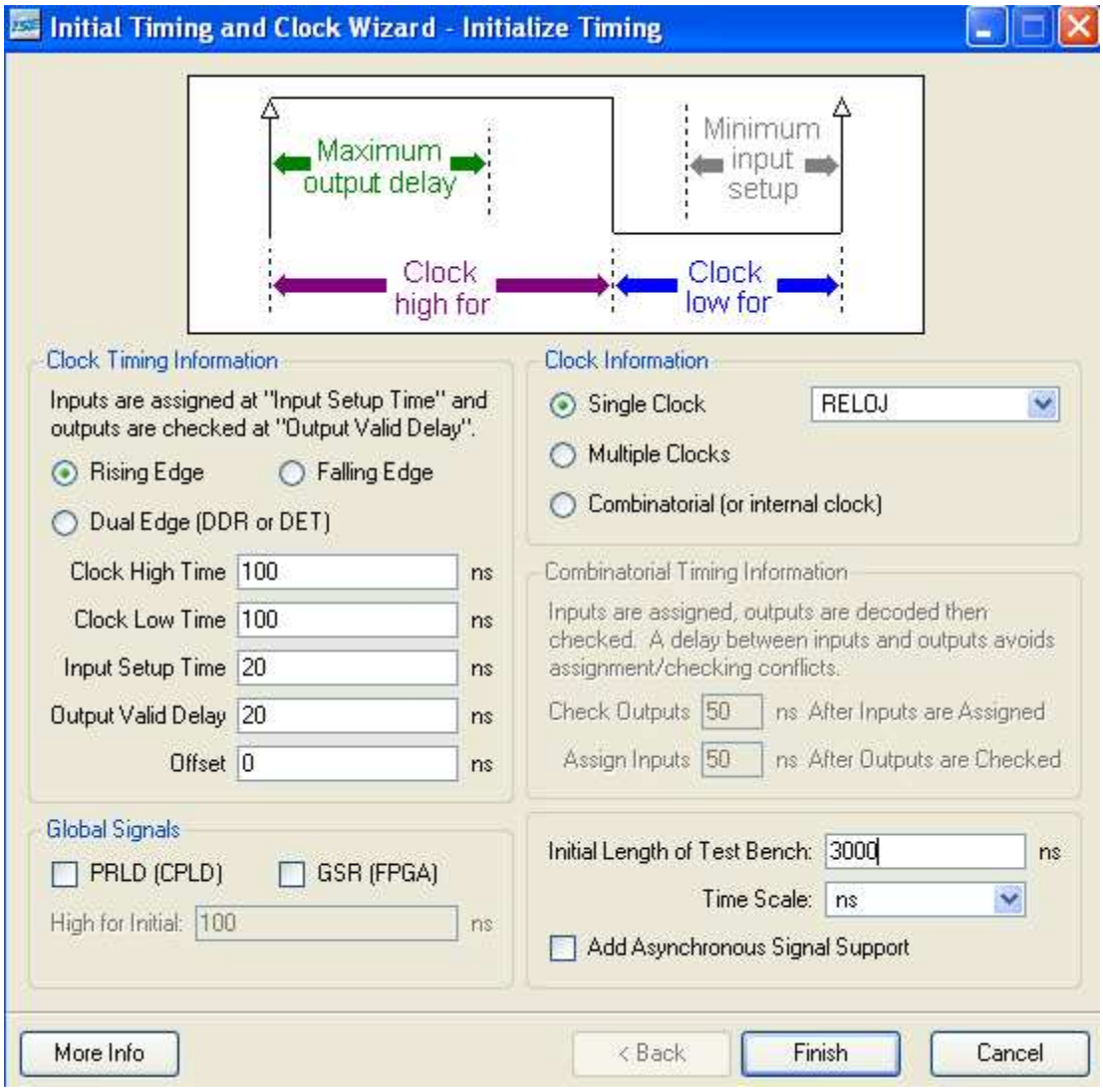


Figura 6.4 Configuracion señal de reloj

Cuando ha configurado la señal de reloj modifique la señal DIRECCION como muestra la figura 6.5., las salidas no requiere modificarlas.

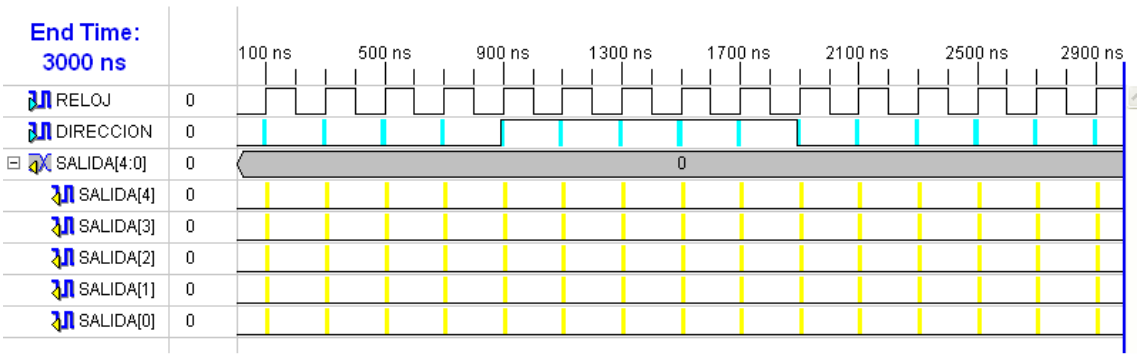


Figura 6.5 banco de prueba para contador

Como se creo el banco de prueba, realice un auto chequeo según procedimiento sección 3.8 D, dando **YES** en la caja de dialogo según figura 3.38, para obtener automáticamente la salida como indica la figura 6.6.

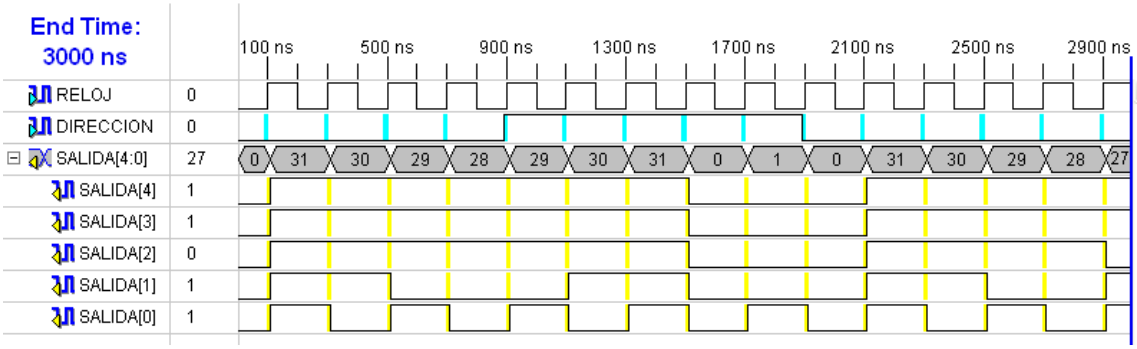


Figura 6.6 Auto chequeo del banco de prueba

En la figura 6.6 observe la señal SALIDA, incrementa el conteo cuando la señal DIRECCION esta en uno y disminuye cuando esta en cero que era lo esperado en el comportamiento del circuito, para ver la salida binaria haga click en el signo **+** ubicado al lado izquierdo de la palabra SALIDA**,** finalmente podremos realizar una simulación del diseño siguiendo las indicaciones de la sección 3.9 de este curso, debe generar una salida como la mostrada en la figura 6.7, no olvide dar click en el signo **+** para desplegar todas las salidas

.

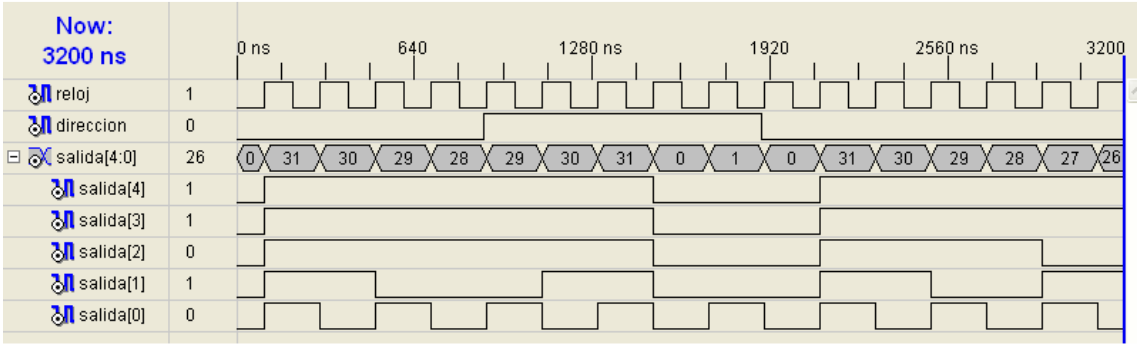


Figura 6.7 simulación contador

Ahora cree el archivo UCF de tiempos de sincronización según indica la sección 3.10 del curso. Teniendo claro que este proyecto es de lógica secuencial, requiere configurar señal de reloj así:
Seleccione RELOJ en el campo **Clock Net Name,** luego haga doble click en el campo vacío de periodo o en el icono para visualizar la caja de dialogo de periodo de reloj y configure 100 ns como indica la figura 6.8, haga click en **OK** para cerrarla misma.

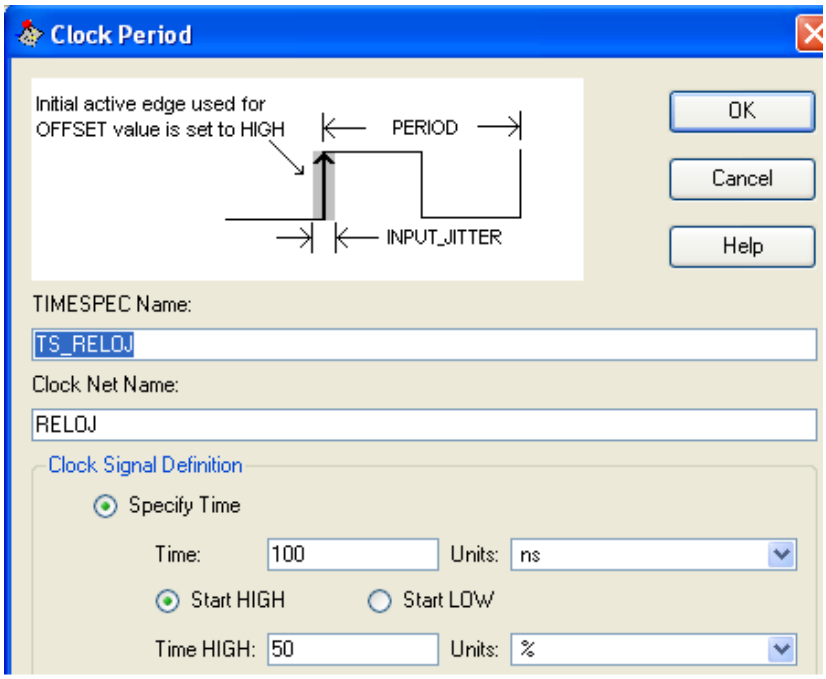


Figura 6.8 Configurar periodo del reloj

Ahora haga doble click en el campo **Pad to setup** o click en el icono  para abrir caja de dialogo, configure 20 ns de OFFSET para validar la entrada como muestra la figura 6.9, haga click en **OK** para cerrar la misma.

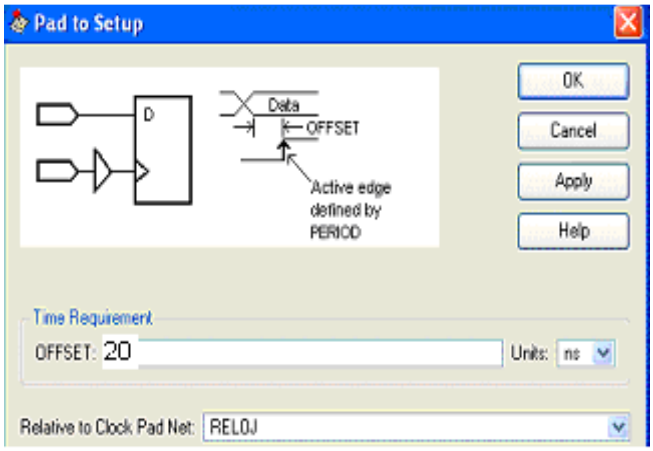


Figura 6.9 Configurar offset entrada datos

Ahora haga doble click en el campo **Clock to Pad** o click en el icono  para abrir caja de dialogo, configure 20 ns de OFFSET para validar la salida de datos como muestra la figura 6.10, haga click en OK para cerrar la caja de dialogo.

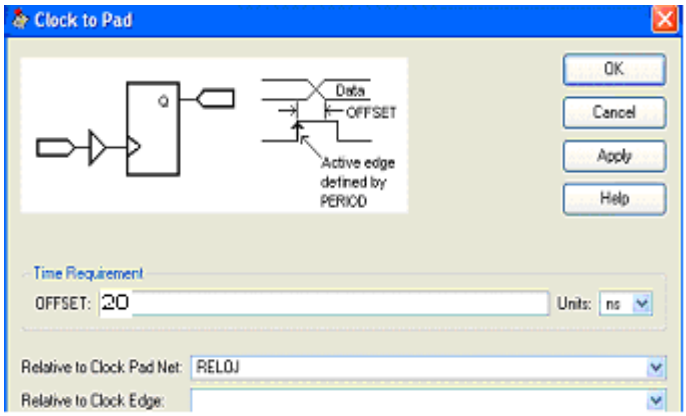


Figura 6.10 configurar offset salida de datos

Finalmente se mostrara un resumen en el editor como muestra la figura 6.11, salve los cambios realizados y cierre el editor.

Figura 6.11 Resumen edición tiempos de sincronización

Para continuar con este proyecto debe implementar el diseño como se indica en la sección 3.11, como es un proyecto secuencial debe verificar los tiempos localizando el **Performance Summary** en la tabla resumen como muestra la figura 6.12,

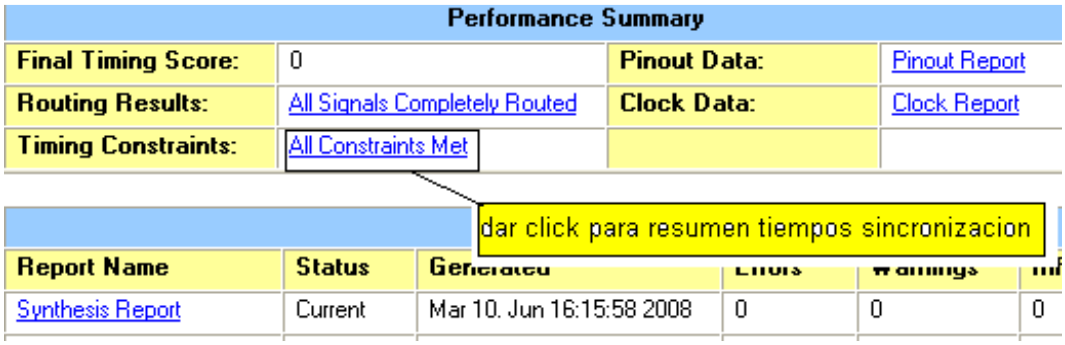


Figura 6.12 Ubicación enlace para reporte resumen tiempos

Haga click en **All Constraints Met** para verificar que cumple con los tiempos configurados en la sección anterior o realizar los ajustes necesarios reeditando el archivo UCF, el reporte es como el mostrado en la figura 6.13, aquí se observa que los tiempos de la columna **Actual** son menores a los tiempos de la columna **Requested,** lo cual indica que si cumple con los tiempos configurados.

Met	Constraint	Requested	Actual	Logic Levels	Absolute Slack	Number of errors
Yes	OFFSET = OUT 20 ns AFTER COMP "RELOJ"	20.000ns	8.547ns	1	11.453ns	0
Yes	OFFSET = IN 20 ns BEFORE COMP "RELOJ"	20.000ns	2.169ns	3	17.831ns	0
Yes	TS_RELOJ = PERIOD TIMEGRP "RELOJ" 100 ns HIGH	100.000ns	4.302ns	2	95.698ns	0

Figura 6.13 Tabla resumen de tiempos

Luego se realiza la asignación de pines del diseño con respecto al kit de desarrollo siguiendo la sección 3.12, asignando los pines como muestra la figura 6.14

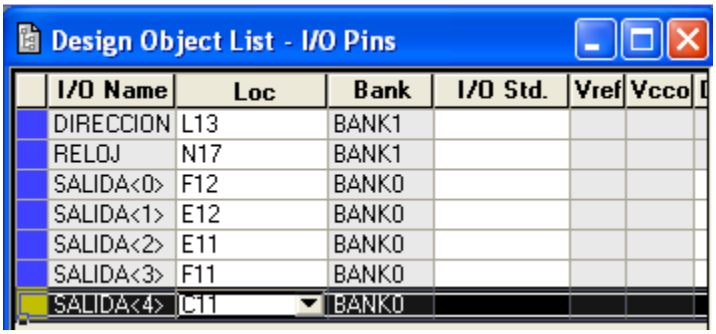


Figura 6.14 Asignación entradas y salidas en kit de desarrollo

Cuando se realiza la asignación de pines o modifica alguna asignación es necesario reimplementar el diseño como indica la sección 3.13 verificando en el reporte de asignación como indica en la figura 6.15

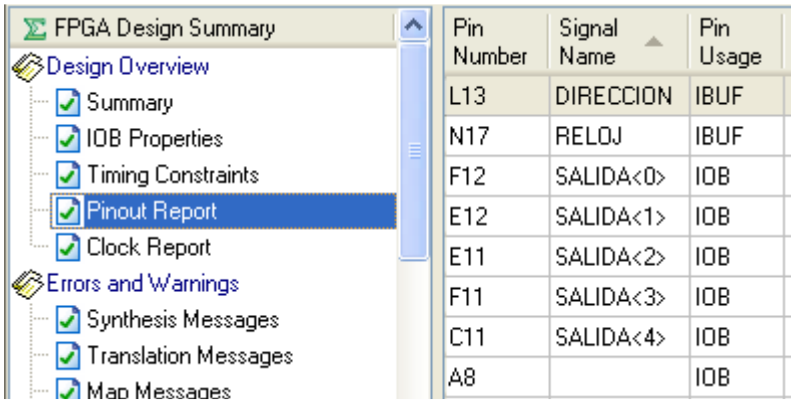


Figura 6.15 Resumen asignación pines

Ahora use el mismo auto chequeo test bench waveform que usted creo para verificar que contador diseñado después de esto a sido completamente implementado. Verifique que los tiempos de simulación operan dentro del sincronismo especificado después que retardos lógicos y enrutamientos son acomodados.
Realice la simulación como indica la sección 3.14, el resultado esperado debe ser como muestra la figura 6.16. Note que en el diagrama de tiempos aparecen los retardos lógicos luego de ser implementado el circuito, esto permite revisar si cumple con los requerimientos reales del diseño.

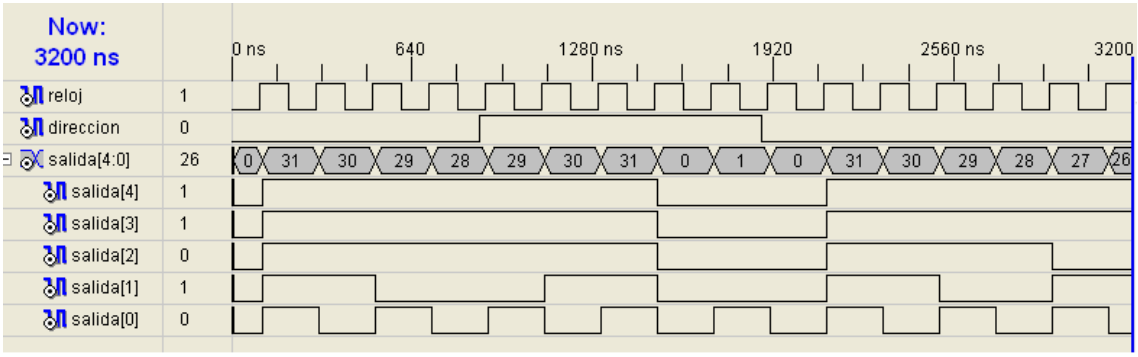


Figura 6.16 Simulación final del diseño

Finalmente debemos descargar el diseño del contador al kit de desarrollo y comprobar su funcionamiento, siga los pasos descritos en la sección 3.15, cuando el led amarillo LD-0 se active indicara que el kit de desarrollo ya tiene cargado el diseño creado por usted, en este caso el contador binario, por lo tanto usted puede manipular y verificar su operación según la asignación de entradas y salidas previamente configuradas.
Cierre el proyecto en el software de desarrollo ISE 8.2i, en el kit de desarrollo borre el proyecto descargado presionando el pulsador **PROG** ubicado cerca de los puertos seriales DCE y DTE; ahora se tiene un proyecto llamado *contador32* el cual esta disponible para usarse en otro proyecto si lo requiere.

UNIDAD ARITMETICO LOGICA (ALU) CON FPGA

Se diseñara una ALU, con las operaciones aritméticas y lógicas mostradas en las tablas siguientes. Para el diseño solo se utilizaran circuitos sumadores completos y las compuertas básicas (AND, OR, XOR, NOT).

Las operaciones aritméticas dependerán de un bit S_2 puesto en un cero lógico **(S_2=0)** y se harán las siguientes operaciones:

S_1	S_0	C_{in} = 0	C_{in} = 1
0	0	F = 0000	F = 1111
0	1	F = A menos B menos 1 más C_{in}	F = B menos A menos 1 más C_{in}
1	0	F = A más B más C_{in}	F = B más A más C_{in}
1	1	F = A menos 1 más C_{in}	F = B menos 1 más C_{in}

Ahora las operaciones lógicas dependen de que el bit S_2 este en uno (S_2=1) y tales acciones para las siguientes opciones son:

S_1	S_0	Salida
0	0	F = A XNOR B
0	1	F = A NAND B
1	0	F = A NOR B
1	1	F = NOT A

La aplicación a implementar en el software Xilinx ISE 8.2I empleara una combinación de dos de los 4 métodos mas empleados en VHDL, el primero será el método por flujo de datos con el cual se crearan los 2 bloques del diseño, este consiste en realizar la descripción del dispositivo o mejor de su funcionamiento por medio de sus ecuaciones booleanas, las cuales se consiguen después de tener la tabla de verdad del elemento y de realizar la correspondiente minimización de tal tabla, lo habitual es por medio de los mapas de karnough. Y el segundo método a emplear se conoce como diseño estructural, con este lo que se hará es reunir los dos bloques que conforman la ALU para realizar la adecuada interconexión entre ellos. Este método lo que hace es valerse de elementos ya creados para utilizarlos en un nuevo producto, es bastante útil en la mayoría de los casos pues la idea de todo programador y desarrollador de hardware es realizar las implementaciones por partes para ir probando y depurando los posibles problemas que se puedan presentar durante el proceso.

SOLUCION

El diseño de la ALU consistirá entonces en un par de bloques así, el primero de ellos es el circuito combinacional encargado de llevar a cabo todas las operaciones que ha de realizar la unidad lógico aritmética, cuyos resultados corresponden a 3 bits (X, Y, Z), cuyas salidas pasaran al siguiente componente de la ALU que es un sumador completo, que entregara el resultado y el carry si lo hay.

Operaciones de la ALU

S_2	S_1	S_0	C_{in}	F	X	Y	Z	
0	0	0	0	'0'	'0'	'0'	0	
0	0	0	1	'1'	'1'	'1'	1	
0	0	1	0	A+B'	A	B'	0	
0	0	1	1	B+A'+1	A'	B	1	**PARTE ARITMÉTICA**
0	1	0	0	A+B	A	B	0	
0	1	0	1	A+B+1	A	B	1	
0	1	1	0	A'-1	A	'1'	0	
0	1	1	1	B	'1'	B	1	
1	0	0	X	A ⊙ B	A ⊙ B	'0'	'0'	
1	0	1	X	(AB)'	(AB)'	'0'	'0'	**PARTE LÓGICA**
1	1	0	X	(A+B)'	(A+B)'	'0'	'0'	
1	1	1	X	A'	A'	'0'	'0'	

Diseño del circuito combinacional

Para este primer bloque del diseño de la ALU se realizan las minimizaciones para cada una de las salidas, dividas en dos grupos, primero se realizara la solución para la parte aritmética de la ALU y la segunda parte corresponde a la minimización de la parte lógica.

Operaciones Aritméticas

Para las operaciones aritméticas el comportamiento de cada una de las salidas debe ser como se nuestra a continuación. Para la salida X se tiene lo siguiente:

C_{in} \ S_1 S_0	00	01	11	10
0	0	A	A	A
1	1	A'	1	A

$$X_A = S_1A + S_0C_{in}'A + S0'C_{in}A + S_1'C_{in}A' + S_0C_{in}A'$$

Ahora en la segunda salida (Y) se tiene que:

C_{in} \ S_1 S_0	00	01	11	10
0	0	B'	1	B
1	1	B	B	B

$$Y_A = S_1B + C_{in}B + S_0C_{in}'B' + S_1'S_0'C_{in}$$

Es importante tener en cuenta que para las dos salidas anteriores no se tuvo en cuenta el bit S_2, el cual es necesario ya que este determina que tipo de operación se desea realizar. Para las dos variables anteriores toda su ecuación ha de ir multiplicada por S_2', ya que las operaciones aritméticas se realizan con este bit puesto en '0'.

Y finalmente el bit de salida Z se comporta de la siguiente manera:

C_{in} \ S_1 S_0	00	01	11	10
0	0	0	0	0
1	1	1	1	1

$$Z_A = C_{in}$$

Operaciones Lógicas

En esta parte no se realiza el empleo de la minimización por mapas de karnough, ya que las funciones a realizar se pueden obtener simplemente de las tareas a realizar y de las condiciones que se deben cumplir para que determinada tarea se lleve a cabo.
Ahora las ecuaciones booleanas para cada uno de los 3 bits de salida son:

$$X_L = S_1'S_0'(A \oplus B)' + S_1'S_0(AB)' + S_1S_0'(A+B)' + S_1S_0A'$$
$$Y_L = 0$$
$$Z_L = 0$$

Al igual que como para la parte aritmética se ha de tener en cuenta el valor del bit S_2 que como ya se menciono determina el tipo de operación a realizar.

Una vez se tienen estas dos partes completas se deben unir para obtener el circuito combinacional encargado de todas las posibles tareas de la ALU, tal como lo muestra el siguiente par de ecuaciones.

$$X = S_2'X_A + S_2X_L$$
$$Y = S_2'Y_A + S_2Y_L = S_2'Y_A$$
$$Z = S_2'Z_A + S_2Z_L = S_2'Z_A$$

Y cuyas ecuaciones expandidas son:

$$X = S_2'S_1A + S_2'S_0C_{in}'A + S_2'S0'C_{in}A + S_2'S_1'C_{in}A' + S_2'S_0C_{in}A' + S_2S_1'S_0'(A \oplus B)' + S_2S_1'S_0(AB)' + S_2S_1S_0'(A+B)' + S_2S_1S_0A'$$

$$Y = S_2'S_1B + S_2'C_{in}B + S_2'S_0C_{in}'B' + S_2'S_1'S_0'C_{in}$$

$$Z = S_2'C_{in}$$

Diseño del sumador completo

Este simplemente se compone de 3 entradas que son los bits de salida del circuito combinacional y consta de 2 bits de salida uno es el bit F que no es mas que el resultado de la ALU y el otro es el bit de acarreo.

El bit de resultado depende de la acción de una XOR entre los 3 bits de entrada, así:

$$F = X \oplus Y \oplus Z$$

Y el bit de acarreo esta determinado por la siguiente ecuación.

$$C = XY + XZ + YZ$$

IMPLEMENTACION DE LA ALU EN LA FPGA

Para crear el nuevo proyecto el cual llamaremos ***ALU_flujo.*** *P*rocedemos así:

A. Seleccione **File >New Project**. Como lo indica la Figura 6.1.

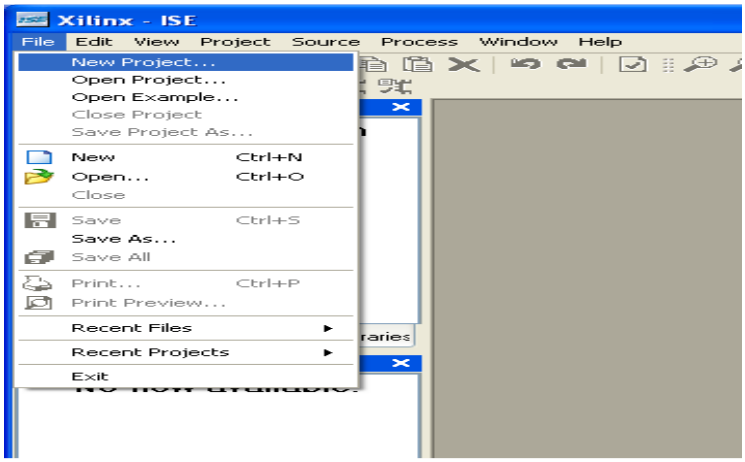


Figura 6.1 Creación de proyecto ***ALU_flujo.***

B. La pagina de creación de nuevo proyecto pide la información general del proyecto, así:

Project Name: Corresponde al nombre del proyecto *ALU_flujo.*
Project Location: Pertenece a la ubicación que se le darán a los archivos. Por defecto el nombre de la carpeta es el mismo nombre del proyecto.

NOTA: Cada proyecto debe tener su propia carpeta. Si multiples proyectos quedan en la misma carpeta pueden ocurrir conflictos.

Top-Level Source Type: En esta opción y para este proyecto en específico seleccione HDL, esta se emplea cuando el archivo principal va a ser realizado con código, para saber mas acerca de las otras opciones diríjase al “Curso Básico FPGA”.

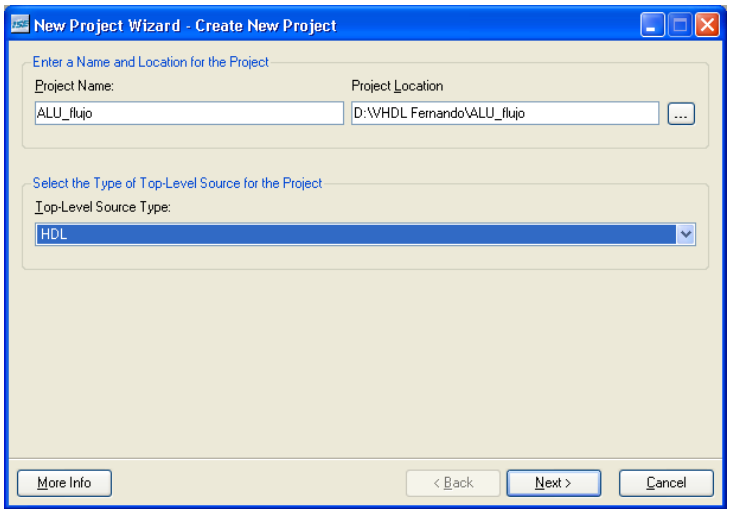
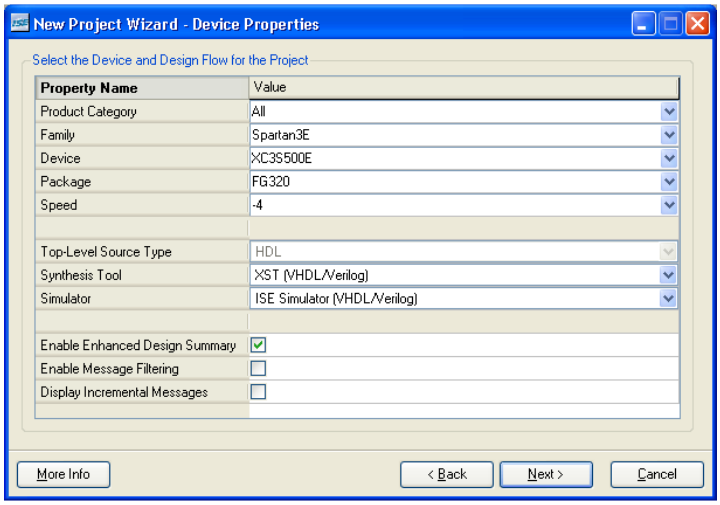


Figura 6.2 Configuración nuevo proyecto

C. Una vez haya configurado las opciones de la figura 6.2 presione el botón Next, cuya acción le mostrara la ventana de la figura 6.3, en la cual se han de configurar las propiedades de la herramienta a emplear. Tales propiedades deben quedar de la siguiente manera:

Product Category: All
Family: Spartan3E
Device: XC3S500E
Package: FG320
Speed: -4
Top-Level Source Type: HDL
Synthesis Tool: XST (VHDL/Verilog)
Simulator: ISE Simulator (VHDL/Verilog)

Y de las ultimas tres opciones solo se debe seleccionar la primera, “Enable Enhanced Design Summary”.



6.3 Propiedades de la herramienta

D. Se presiona nuevamente Next y sale una tercera ventana , por orden se debería crear el archivo o uno de los archivos sobre los que se va a trabajar,

aunque este paso no es obligatorio. Presionando otra vez el botón Next sale otra ventana muy similar a la anterior (figura 6.4) pero en esta lo que se hace es adicionar archivos ya creados que se puedan requerir en el proyecto.

Se recomienda que todo archivo que se vaya a adjuntar a un proyecto primero se almacene en la carpeta del proyecto, simplemente basta con pasar el archivo ".vhd" si se realizo por código o el archivo ".sch" si es una aplicación esquemática.

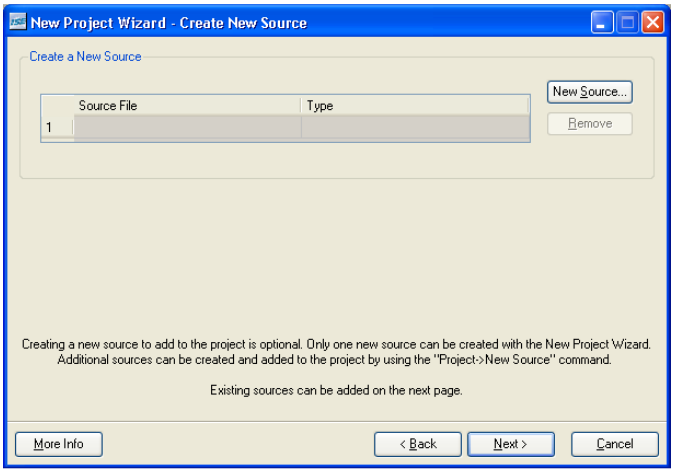


Figura 6.4 Adición de nuevas fuentes al proyecto

Para continuar con la actividad se va a crear en la ventana mostrada en la figura 6.4 el archivo correspondiente al circuito combinacional descrito en la sección 5.2 este se llamara *combinacional.vhd*, para ello se selecciona New Source que abre la ventana de la figura 6.5. En ella se ha de seleccionar el tipo de archivo a emplear, para este caso se usa el VHDL Module que es el adecuado para los diseños realizados mediante código, en el lado derecho se encuentra el campo para el nombre del archivo y la ubicación en que quedara, lo correcto es que sea la misma carpeta de todo el proyecto y finalmente que la opción adicionar al proyecto debe estar activa.

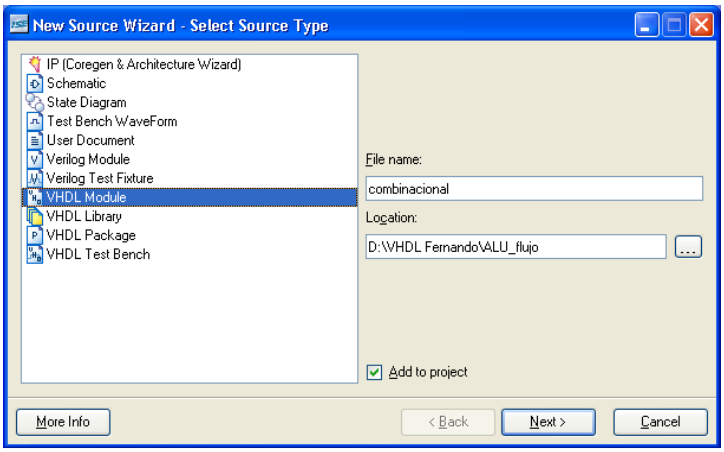


Figura 6.5 Creación de un nuevo archivo

Una vez se haya seleccionado el tipo de archivo en el cual se va a trabajar y definido su nombre debe aparecer una ventana en la que se deben indicar cuales serán los puertos de entrada y salida del diseño, al igual que uno de los

pasos anteriores no es obligatorio realizarlo, ya que también el programador lo puede hacer digitando esta información directamente en la entidad (este concepto se verá mas adelante). En la figura 6.6 aparece la ventana que se acabo de mencionar junto con los puertos requeridos, a los cuales se les debe asignar un nombre, indicar de que tipo son, si de entrada, salida o entrada/salida y para el caso de la entrada S se indica además que es un bus o vector de un tamaño de 3 bits con la posición 2 como la MSB y la 0 como la LSB.

Los nombres de la entidad y la arquitectura se pueden modificar al gusto del programador pero su cambio no representa ningún cambio relevante en el desempeño del dispositivo.
NOTA: Una regla importante para la asignación de los nombres del archivo y la entidad es que estos dos siempre deben ser los mismos, además deben cumplir las siguientes reglas:

Regla	Correcto	Incorrecto
El primer carácter siempre es una letra	Suma4	4suma
El segundo carácter no puede ser un guión bajo	S4_bits	S_4bits
No se permiten dos guiones seguidos	Resta_4_	Resta__4
No se puede emplear símbolos especiales	Clear_8	Clear#8

Por cierto estas reglas aplican a todos los identificadores que se empleen en VHDL, no solo a los nombres de las entidades.

Una vez completada esta información y al realizar el clic sobre el Next aparece un resumen acerca de el archivo creado tal como lo muestra la figura 6.7.

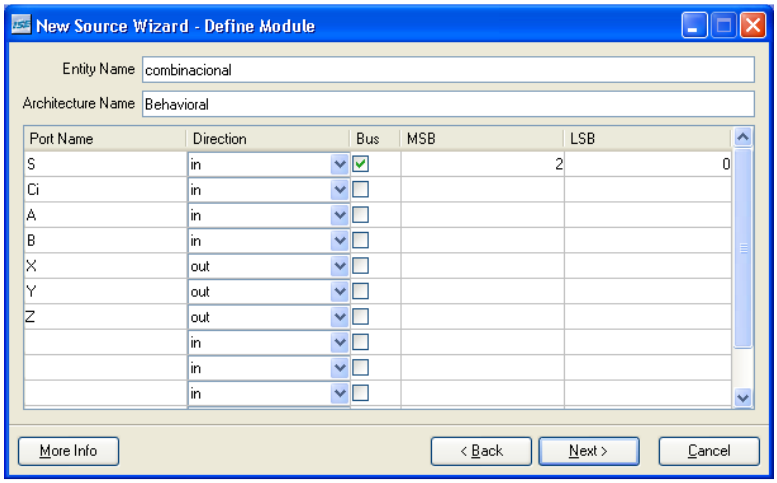


Figura 6.6 Adición puertos de entrada y salida

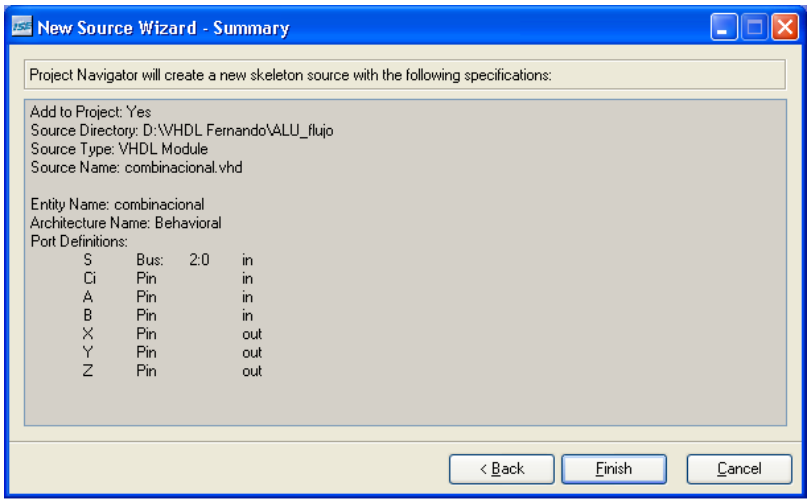


Figura 6.7 Resumen del componente creado

Por ultimo se da clic en Finish y se regresa a la ventana de la figura 6.4 con la única diferencia que en aquella ya debe aparecer el archivo creado y el tipo al que pertenece.

E. Se pasa al siguiente paso que es la adición de elementos ya creados, pero como para el caso no es necesario, se pasa por alto esta acción.

F. Y la última ventana antes de completar la creación del proyecto contiene un resumen de todo lo configurado para este. Para dar el proceso por terminado solo basta con presionar el botón Finish.

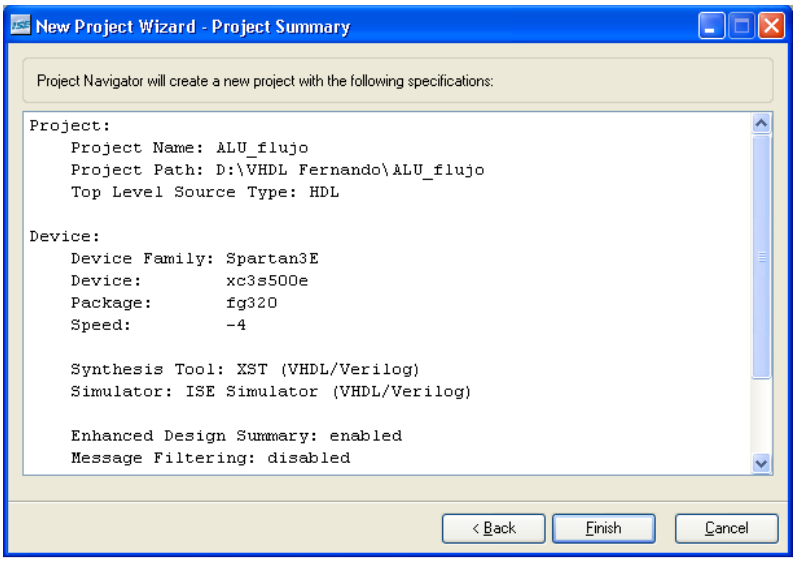


Figura 6.8 Resumen del proyecto creado

G. A partir de este punto comienza la implementación de la ALU requerida. Como primer paso es ubicarse en el archivo creado hace un momento el *combinacional.vhd* que se debe haber abierto al terminar de crear el proyecto. Como lo muestra la figura 6.9 al lado derecho esta el código básico para la realización de esta parte de la ALU, el cual comprende las siguientes partes: la inclusión de librerías mediante la palabra reservada “library” y de paquetes

mediante la palabra “use”, los paquetes adicionados por el programa son los básicos pero a la vez los más empleados.

El paquete STD_LOGIC_1164 incluye todos los métodos esenciales para los desarrollos de las aplicaciones, el STD_LOGIC_ARITH contiene las aplicaciones matemáticas y el STD_LOGIC_UNSIGNED toda implementación relacionada con las cantidades sin signo.

La segunda parte es la declaración de la entidad, en esta lo que se hace es definir en si al componente ante el exterior mediante la asignación de un nombre y los puertos de entrada/salida necesarios para su correcto funcionamiento, tales puertos como recordaran se definieron al adicionar al proyecto un nuevo archivo.

Y como tercera y última parte del código esta la arquitectura, que así como la entidad distingue al componente de los demás, esta se encarga de definir como se han de comportar las salidas de acuerdo a los valores que puedan tomar los puertos de entrada.

El lenguaje que muestra la figura 6.9 es simplemente la parte básica de todo dispositivo que se desee crear mediante código, en lo único que todos los archivos deben variar es en sus identificadores de la entidad.

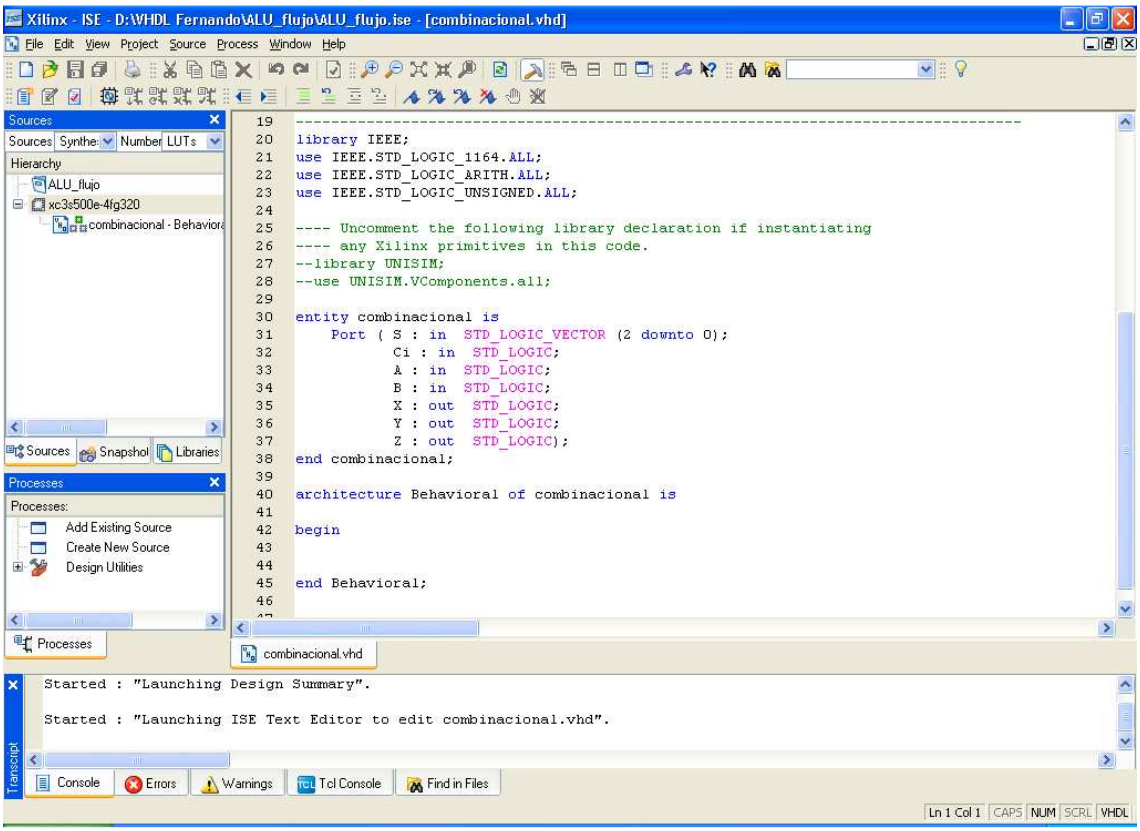


Figura 6.9 Archivo combinacional.vhd – básico

La parte de la que se ha de encargar el programador debe ir en la arquitectura la cual siempre debe estar entre el “begin” y el “end behavioral”, tal como lo muestra la figura 6.10.

```
architecture Behavioral of combinacional is

signal Xa,Xl,Ya,Za, ns0,nS1,nS2,nCi: std_logic;

begin

nS0 <= not(S(0));
nS1 <= not(S(1));
nS2 <= not(S(2));
nCi <= not(Ci);
Xa <= (s(1) and A) or (S(0) and nCi and A) or (nS0 and Ci and A) or (nS1 and Ci and not(A)) or (S(0) and Ci and not(A));
Ya <= (S(1) and B) or (Ci and B) or (S(0) and nCi and not(B)) or (nS1 and nS0 and Ci);
Za <= Ci;
Xl <= (nS1 and nS0 and not(A xor B)) or (nS1 and S(0) and not(A and B)) or (s(1) and nS0 and not(A or B)) or (S(1) and S(0) and not(A));

X <= (nS2 and Xa) or (S(2) and Xl);
Y <= nS2 and Ya;
Z <= nS2 and Za;

end Behavioral;
```

6.10 Arquitectura de combinacional.vhd

Como ya se había mencionado anteriormente, la técnica a emplear para la implementación de cada parte de la ALU es mediante el flujo de datos (Ecuaciones booleanas), cuyas ecuaciones son las mostradas en la imagen para cada variable.

Si se observa cuidadosamente se ve que entre la declaración de la arquitectura y su inicio hay una instrucción adicional, la cual consiste en la asignación de unas señales internas que resultan bastante útiles para realizar el desarrollo del elemento de una manera mas ordenada. La selección del tipo de estas variables debe ser el mismo de las variables con las que se va a trabajar, para este caso como todos los puertos son de tipo std_logic este es el mas adecuado para las señales.
Regresando al campo de diseño se tiene la siguiente configuración: en las primeras cuatro líneas se realiza simplemente una operación “not” para las variables que requieren ser negadas y cuyo valor ha de ser almacenado también en 4 de las señales declaradas (nS_0, nS_1, nS_2, nC_i), luego se declaran las ecuaciones halladas en la secciones 5.2.1 operaciones aritméticas (X_A, Y_A, Z_A) y 5.2.3 operaciones lógicas (X_L). Es en esta parte en donde se refleja realmente la importancia de realizar la aplicación por partes pues como se ve estas ecuaciones resultan bastante largas, por lo general complejas y difíciles de comprender para la depuración de errores por la cantidad de signos de agrupación y operaciones que contienen.

Finalmente y a partir de las ecuaciones de la sección 5.3 resultan las ecuaciones finales correspondientes a los tres puertos de salida del componente, que a su vez dependen de las señales y ecuaciones empleadas en las 8 instrucciones anteriores.

Como un ultimo apunte en esta parte es de gran importancia destacar el símbolo empleado para la asignación de los valores (<=) que como se observa de la figura 6.10 se emplea en todas las líneas, pero esto tiene su razón de ser ya que este se utiliza siempre que se estén manipulando los puertos de salida declarados en la entidad o las señales creadas antes de empezar la arquitectura.

H. Ahora guarde lo realizado y si se desea se puede sintetizar el proyecto pero por el momento no es muy necesario. El paso a seguir es la adición de un nuevo archivo de texto al proyecto para lo cual se va a hacer lo siguiente: se dirige el cursor a la pestaña *Project / New Source*, en este punto debe salir la misma ventana de la figura 6.5 en la cual se selecciona el tipo de archivo a crear y el nombre del mismo, luego se adicionan los puertos y se configuran sus características, por ultimo aparece el resumen del archivo que se esta creando y una vez se le de Finish, de inmediato debe abrirse en la ventana Principal del Xilinx-ISE. Además la inclusión del elemento se puede verificar en la ventana Sources del mismo programa (Figura 6.11)

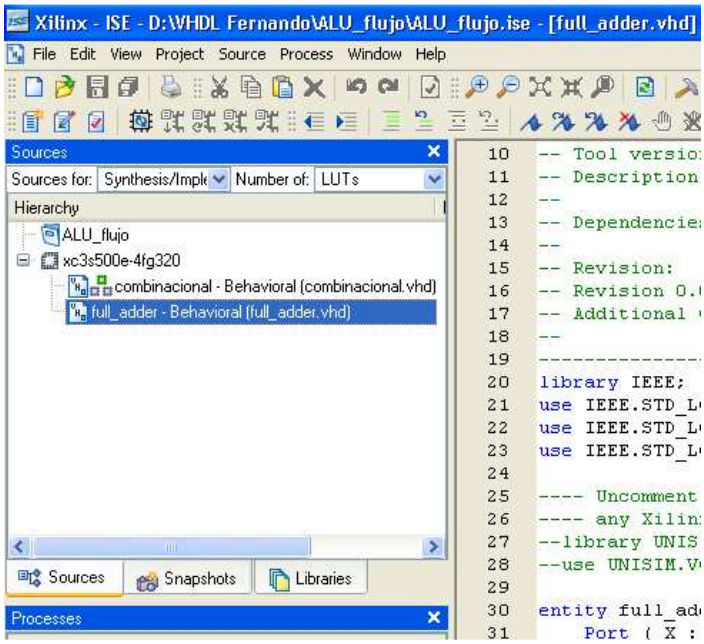


Figura 6.11 Ventana Sources

El elemento que se debe crear es el full-adder, es igual que el combinacional un modulo VHDL y sus puertos son los siguientes: X, Y, Z son puertos de entrada de un solo bit y F y C son los puertos de salida también de un solo bit.

I. Ahora lo que sigue es adicionar a la arquitectura el comportamiento que debe tener el componente *full_adder.vhd* que no es más que adicionar el par de ecuaciones determinadas en la sección 5.3.

Nuevamente se guarda lo realizado, se comprueba que la sintaxis esta correcta seleccionando la opción “Check Syntax” en la ventana Processes y si después de realizar el respectivo análisis muestra un círculo verde como en la figura 6.12 la prueba se ha pasado exitosamente.

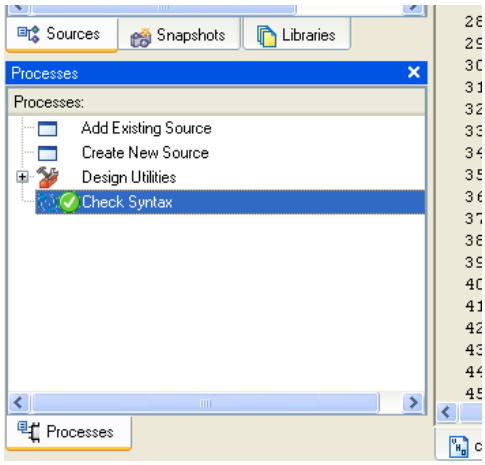


Figura 6.12 Ventana Processes

J. Teniendo ya creados los dos componentes de la ALU solo basta adicionar un nuevo archivo .vhd llamado ALU que será el contenedor de aquellos dos, es decir, es la ALU en si, pero lo único que se hace en este es realizar la interconexión de los bloques, las entradas y las salidas, mediante la técnica denominada diseño estructural, que como se comento al principio consiste en realizar los diseños por bloques y en un archivo global llamar a los diferentes componentes para formar el dispositivo que se requiere.

La figura 6.13 muestra el archivo global (ALU) en el cual se percibe como se hace el llamado a los componentes y desde luego como se han de utilizar.

Para empezar es importante aclarar que en este programa no es necesario realizar el llamado de los componentes debido a que todos estos han sido incluidos en el proyecto, aunque si se realiza el llamado no genera errores, además se va a ser simplemente con el fin dar a conocer la manera de hacerlo ya que muchas otras compañías que trabajan en el desarrollo del VHDL si necesitan que se realicen tales llamados como sucede con Altera.

Entonces mirando las líneas de la 42 a la 50 y de la 51 a la 57 en la figura 6.13 esta el llamado para los dos elementos creados anteriormente, cuya declaración si se compara con la de la entidad se ve que son similares, realmente lo único que cambia es la palabra reservada “entity” por “component”, se elimina el “is” y se finaliza con “end component”.

```
29
30  entity ALU is
31      Port ( Ci : in  STD_LOGIC;
32             S : in  STD_LOGIC_VECTOR (2 downto 0);
33             A : in  STD_LOGIC;
34             B : in  STD_LOGIC;
35             F : out  STD_LOGIC;
36             C : out  STD_LOGIC);
37  end ALU;
38
39  architecture Behavioral of ALU is
40  -- en XILINX no es necesario llamar los elementos que se van a emplear, debido a que se
41  -- guarda en proyectos.
42  component combinacional
43     port(S : in  STD_LOGIC_VECTOR (2 downto 0);
44             Ci : in  STD_LOGIC;
45             A : in  STD_LOGIC;
46             B : in  STD_LOGIC;
47             X : out  STD_LOGIC;
48             Y : out  STD_LOGIC;
49             Z : out  STD_LOGIC);
50  end component;
51  component full_adder
52      Port ( X : in  STD_LOGIC;
53             Y : in  STD_LOGIC;
54             Z : in  STD_LOGIC;
55             F : out  STD_LOGIC;
56             C : out  STD_LOGIC);
57  end component;
58
59  signal Xi,Yi,Zi: std_logic;
60
61  begin
62
63     etiq1: combinacional port map(S,Ci,A,B,Xi,Yi,Zi);
64     etiq2: full_adder port map(Xi,Yi,Zi,F,C);
65
66  end Behavioral;
```

6.13 ALU.vhd

Los puertos declarados de cada elemento deben ser iguales a los nombrados en el archivo de cada uno. Algo que hasta el momento no se ha explicado es la ubicación de los punto y coma, su labor es la misma que se usa en C simplemente indican la finalización de una instrucción, para este signo hay un caso especial que se presenta en la declaración de las entidades o los componentes, si se observa en las diferentes imágenes que contiene código incluida la entidad se puede ver que siempre en el ultimo puerto declarado el punto y coma se pone después de cerrar la instrucción port( ). Esta característica es de especial importancia ya que es un error común durante la digitación del código.

Luego de llamar a los componentes se crean las señales, para este caso solo se crean 3 que se emplean para realizar la conexión entre las salidas del circuito combinacional y las entradas del sumador completo. Después de esto se inicia la arquitectura y es en esta parte donde se utilizan los componentes necesarios, tal como muestra la figura 6.13 el modo de uso de estos es de la siguiente manera, lo primero que se hace es etiquetar la instrucción, luego se invoca el componente llamado previamente y mediante la instrucción “port map” se realiza el mapeo de puertos, entonces lo que se hace en este punto es que mediante el “port map” las entradas se pasan al componente empleado y este se encarga de devolver los resultados por los respectivos puertos de salida. El orden en que se han de ubicar de los diferentes puertos en esta instrucción debe ser el mismo que tiene en la entidad. Por ejemplo con estos dos componentes primero esta declaradas las entradas y por ultimo las salidas, para el caso del combinacional la primera entrada es el vector de 3 bits, luego

el carry de entrada, y después los dos bits a operar y luego las 3 salidas de un bit.

Como las 2 salidas dependen del resultado final entregado por el sumador completo, las salidas del bloque combinacional se han de guardar en las señales creadas antes de iniciar la arquitectura, las cuales a su vez van a ser las entradas al sumador.

Los tres elementos desarrollados en este proyecto fueron creados a partir de instrucciones concurrentes, esto quiere decir que todas las instrucciones que se encuentran dentro de la arquitectura se realizan de manera simultánea, siempre que una instrucción posterior no dependa del resultado de una instrucción anterior.

Una manera que el programador puede utilizar para verificar que los componentes se están añadiendo al archivo que se desea, es ver que en la ventana Sources en tal archivo se tengan los elementos adicionados tal como lo muestra la figura 6.14

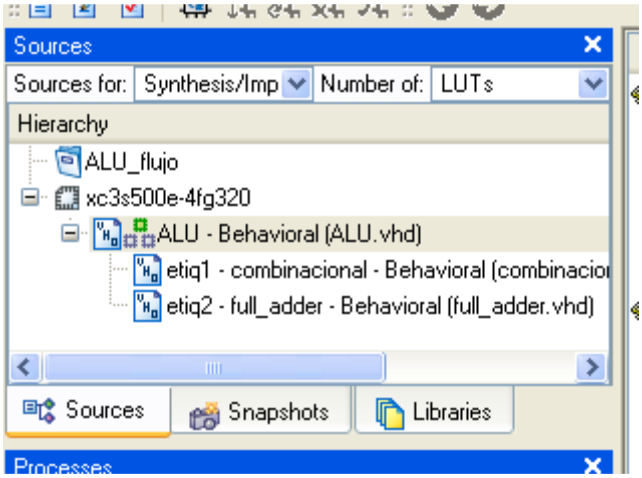


Figura 6.14 Adición de elementos

K. Con esto se ha finalizado la creación de la ALU, solo hace falta guardar el proyecto y compilarlo para verificar que ninguno de los tres elementos presente algún tipo de error.

Para compilarlo vaya a la ventana sources seleccione el componente ALU-Behavioral y en la ventana processes de doble clic sobre la opción Synthesize-XST, la prueba ha pasado con éxito cuando tal opción se pone en verde (figura 6.15).

Antes de continuar con las demás opciones de configuración del proyecto se recomienda hacer la cantidad de pruebas mediante simulación necesarias para la verificación del adecuado funcionamiento del diseño. Esto porque después de la creación del Timing Constrains (archivo UCF) la modificación de algunos parámetros se hace demasiado difícil y porque se supone que cuando ya se realiza este paso la Unidad Aritmético Lógica esta funcionando correctamente.

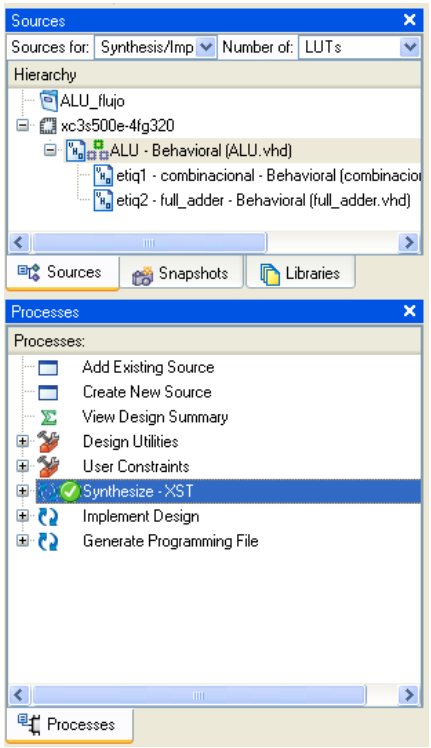


Figura 6.15 Compilación del proyecto

SIMULACIÓN DEL DISEÑO.

A. Al igual que como si fuera para la creación de los elementos anteriores se debe adicionar un archivo en este caso del tipo simulación (Test Bench Waveform) para lo que se debe ir a la pestaña Project / New Source y nuevamente se debe ver la ventana de la figura 6.5 y se siguen las instrucciones, primero seleccionar el tipo de archivo para este es el Test Bench Wavefom, darle un nombre y presionar Next. El siguiente paso es indicar con que elemento se va a asociar la simulación, para este proyecto lo agrupamos con la ALU, luego aparece el resumen de creación y solo hace falta finalizar para que se vea en la pantalla principal.

Después aparece una ventana de configuración de temporización de la simulación. La primera opción que debe cuadrar es la información del reloj que cuenta con 3 opciones las dos primeras (un solo reloj y múltiples relojes) son para diseño con lógica secuencial y la tercera como su nombre lo indica es para diseño de lógica combinacional o diseños que tienen su propio reloj. Como este proyecto es una aplicación de lógica combinacional la opción que se requiere es la tercera Combinatorial (or internal clock).

Con la selección de información del reloj en combinacional se desactivan todas las opciones del lado izquierdo de la pantalla quedando solo 3 opciones por configurar en el lado derecho. Las 2 primeras son un par de tiempos, el primero pregunta por cuanto tiempo esperar para validar las salidas después de haberse asignado las entradas y el segundo por cuanto tiempo esperar después de chequear las salidas para asignar los siguientes valores y la

tercera opción permite configurar el tiempo que durara la simulación (figura 7.1).

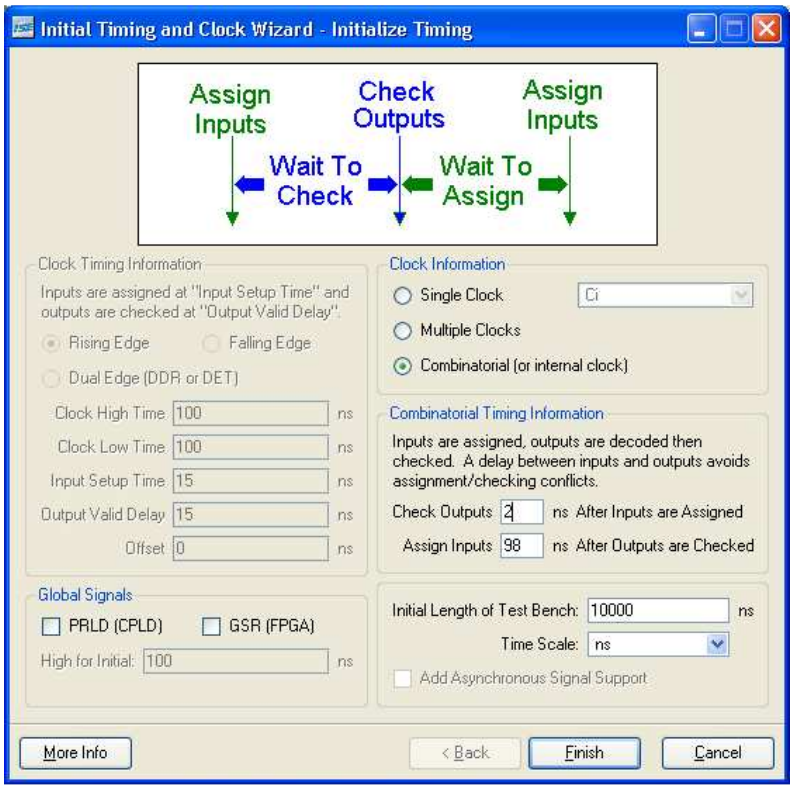


Figura 7.1 Configuración del temporizador de la Simulación

B. Una vez finalizada la configuración de la simulación el Xilinx-ISE retorna a su pantalla principal en donde esta abierto el archivo de simulación. En el cual se procede a asignar los valores para cada una de las variables de entrada, de la siguiente manera: una forma puede ser modificando cada posible valor de 1 a 0 o viceversa dando clic en las zonas azules para las entradas y en las zonas amarillas para las salidas esperadas. La otra es ubicar el cursor en algún punto de la simulación dar clic derecho y seleccionar la opción “Set value…” (Figura 7.2) que mostrara una pequeña ventana que cuenta con un espacio para ingresar el valor de ese punto o presionando el botón “Pattern Wizard” muestra una siguiente ventana que permite configurar todos los valores o una parte de ellos de determinada variable (figura 7.3).

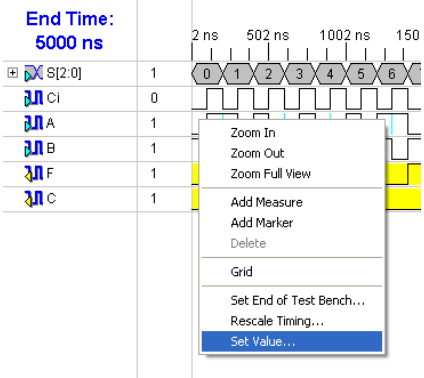


Figura 7.2 Asignación de valores a las variables

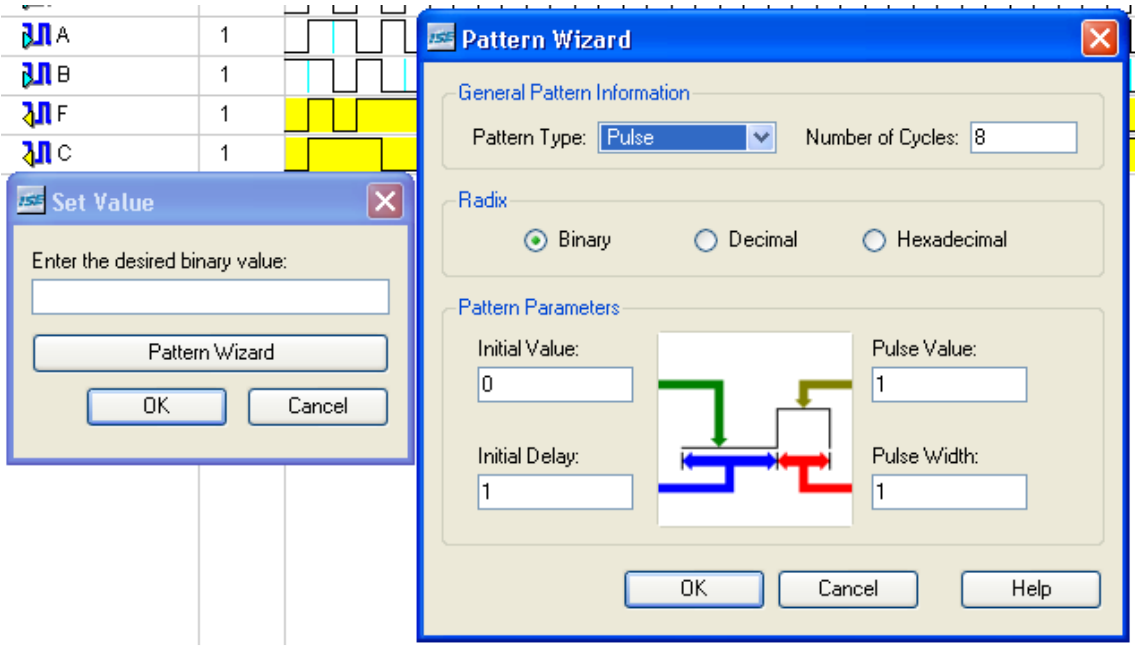


Figura 7.3 Configuración de los valores para la simulación

Finalmente se guarda la simulación con la configuración de los valores y se procede a compilar la simulación que se acabo de guardar.

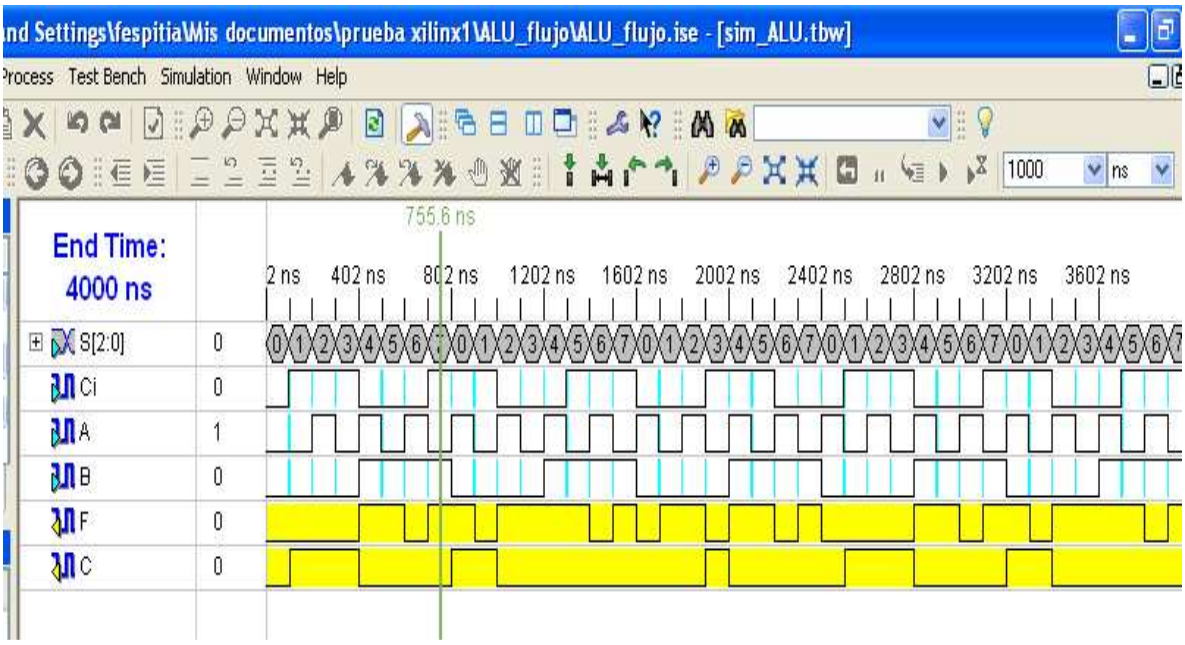


Figura 7.4 BANCO DE PRUEBA

C. En la ventana de Sources, seleccione el modo **Behavioral Simulation** para ver que el archivo de banco de pruebas a sido adicionado automáticamente a su proyecto.Ver Figura 7.5

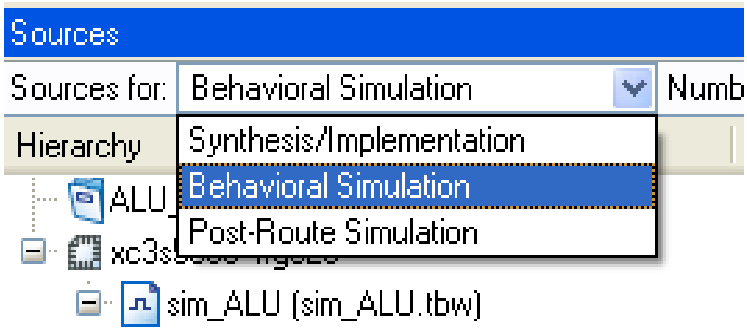


Figura 7.5 Banco de pruebas adicionado al proyecto

3.8 Crear un auto chequeo para el banco de prueba

Para crear una auto-verificación de banco de pruebas, con los valores modificados de salida manualmente y compararlos, ejecutar el proceso **Generate Expected Results** para crearlo de forma automática. Si ejecuta el proceso **Generate Expected Results**, inspeccionar visualmente los valores de salida para ver si ellos son los que esperaba para el conjunto dado de valores de entrada.y la salida llamada DIFF

Para crear la auto-verificación de test bench waveform automáticamente, haga lo siguiente:

A. Verifique que **Behavioral Simulation** se ha seleccionado en la lista desplegable de la ventana **sourses**.

B. Seleccione el archivo **sim_ALU.tbw** en la ventana sources.

C. En la pestaña Processes, haga click en el signo "+" para ampliar el proceso de simulación de Xilinx ISE Haga doble click en el proceso. **Generate Expected Simulation Results** Este proceso simula el diseño en un proceso.Ver Figura 7.6

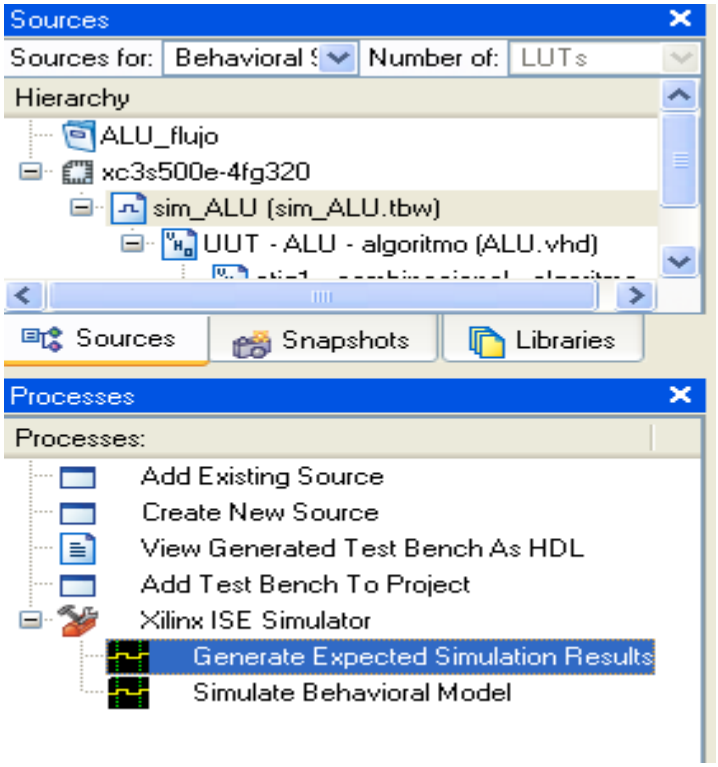


Figura 7.6 Generar auto chequeo con banco de pruebas

D. El cuadro de diálogo **Expected Results** se abre, hay dos opciones. Seleccione **No** para adicionar una señal de comparación en la visualizacion del resultado o **yes** para generar las salidas automáticamente. seleccione **No,** ver figura 7.7

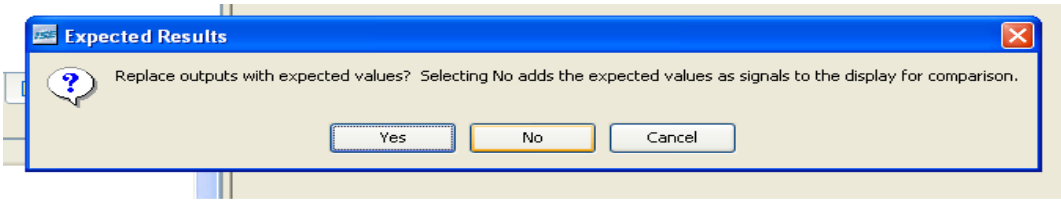


Figura 7.7 Selección para tener señal de comparación

en el area de trabajo se desplega los resultados , mostrando dos señales de salida llamadas **Z** y **Z _DIFF** donde **Z** es la salida previamente configura por el usuario y **Z_DIFF** es la calculada por el programa, luego aquí evalúa el usuario si era lo esperado o si requiere modificar el programa del diseño, como se observa en la Figura 3.37

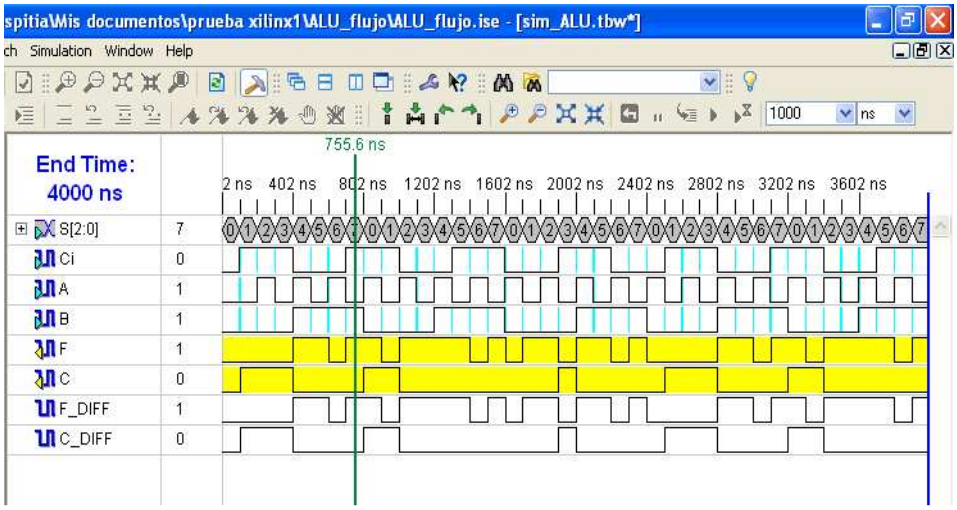


Figura 3.37 Señales de salida permite comparación con señales esperadas

F. Salvar el test bench waveform y cerrarla, así queda creado un auto-chequeo del **test bench waveform.**

3.9 Simulando la funcionalidad del diseño

Verifique que la compuerta diseñada funciona como se espera realizando la simulación de la siguiente manera:
A. Verifique que **Behavioral Simulation** and **sim_alu.tbw** son seleccionados en la ventana sources
B. En la pestaña **Processes**, haga click en el signo "+" para ampliar el proceso de simulación de Xilinx ISE y haga doble click en el proceso **Simulate Behavioral Model**.

El ISE 8.2i simulador abre y ejecuta la simulación del banco de pruebas.
C. Para ver los resultados de la simulación, seleccione la pestaña de **simulación** y haga un zoom en las transiciones. Los resultados de la simulación se verán como se muestra la Figura 7.8

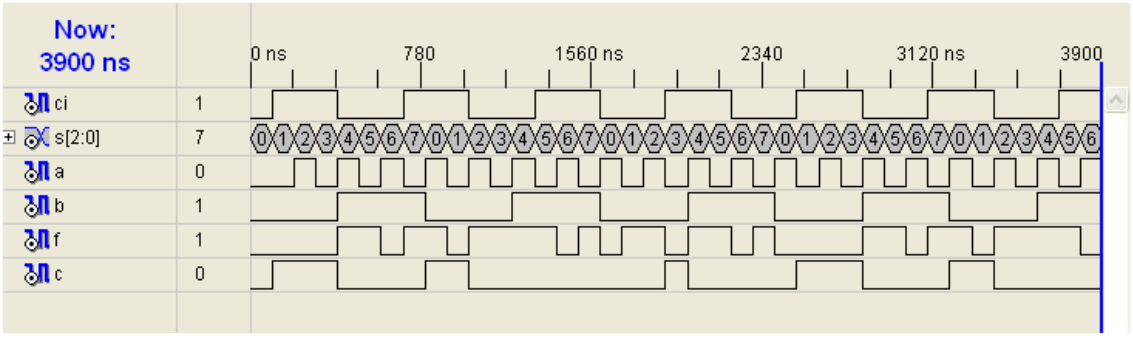


Figura 7.8 Resultados de la simulación

D. Verifique que la ALU opera como se esperaba.
E. Cierre la simulación. Si aparece el siguiente mensaje: ""You have an active simulation open. Are you sure you want to close it?, Haga click en **yes** para continuar.como se indica en la figura 3.41

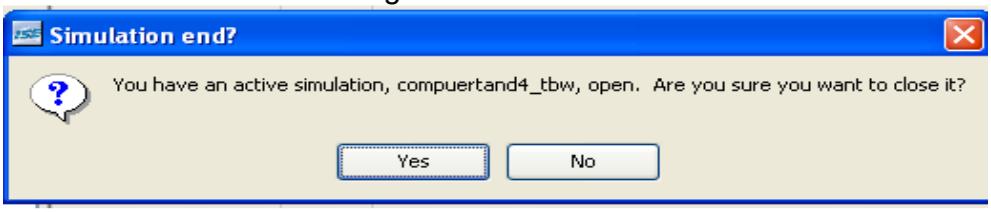


Figura 3.41 Finalizar simulación

Usted ya completo su simulación utilizando el simulador de ISE.

Creación de tiempos de sincronización

Sincronizando el tiempo entre la FPGA y la lógica de su entorno, así como la frecuencia del diseño es como deberá actuar en el ámbito interno de la FPGA. El tiempo es especificado para sincronizar el desempeño del diseño. Se recomienda que use sincronización a nivel global. El período de reloj sincroniza

la frecuencia de reloj a la que su diseño debe operar dentro de la FPGA. El offset de sincronismo especifica cuándo esperar datos válidos en las entradas del FPGA y cuando datos validos estarán disponibles en las salidas del FPGA Para sincronizar el diseño haga lo siguiente:

A. Seleccione **Synthesis/Implementation** de la lista desplegable en la ventana **sources**
B. Seleccione archivo fuente.HDL de *alu-algoritmo.* como muestra la Figura 8

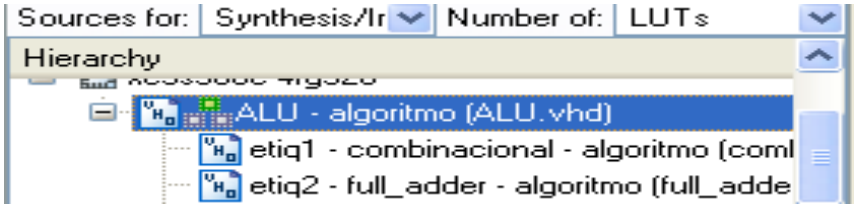


Figura 3.42 Selección archivo fuente VHD

C. Haga click en el signo "+" junto al grupo de procesos **user constraints**, y haga doble click en el proceso **Create Timing Constraints**.Ver Figura 8.2.

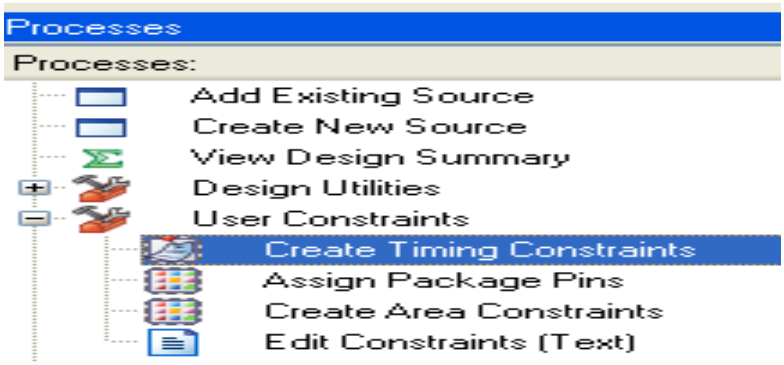


Figura 8.2 crear archivo de sincronismo

ISE corre los pasos de Synthesis y Translate creando automáticamente un **User Constraints File (UCF);** aparece un mensaje como lo muestra la Figura 3.44:

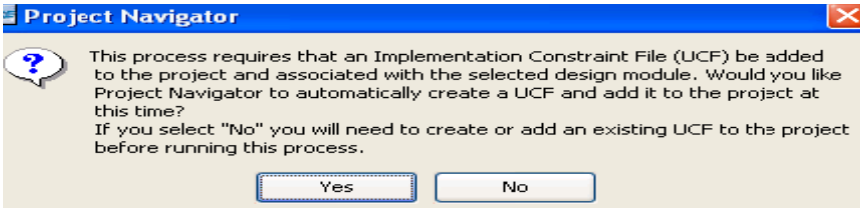


Figura 3.44 Anexar archivo UCF al proyecto

D. Haga click en **Yes** para agregar el archivo UCF a su proyecto. El archivo **ALU.ucf** se añade a su proyecto y es visible en la ventana **sources** Ver figura 8.3.

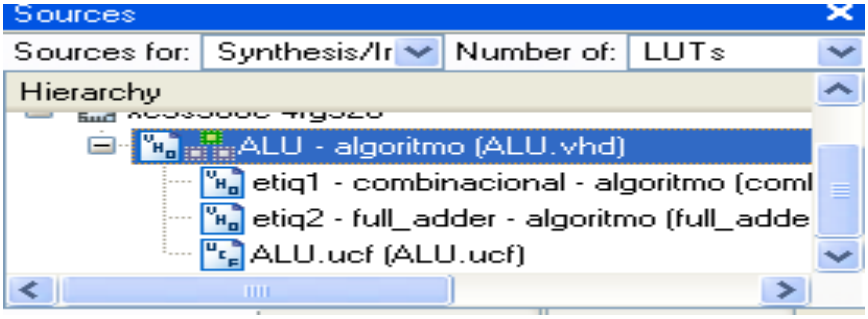


Figura 3.45 Verificar archivo UCF anexado al proyecto

El Xilinx Constraints Editor se abrirá automáticamente. Nota: También puede crear un archivo UCF para su proyecto seleccionando **Project** **Create New Source**.

Para este caso no se requiere reloj por ser un circuito combinacional, luego cierre la ventana Xilinx Constraints Editor arriba a la derecha.Ver figura 3.46

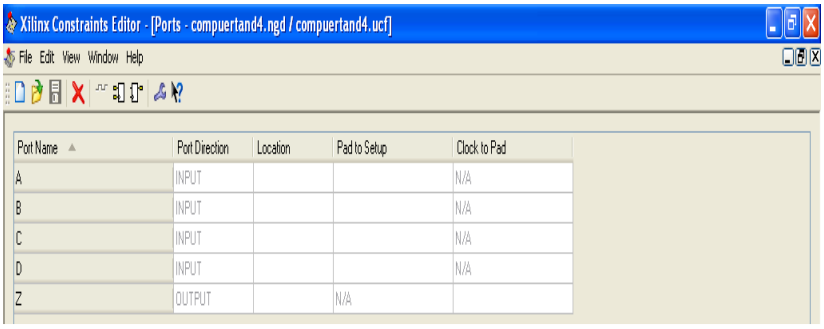


Figura 3.46 Ventana edición tiempos sincronismo

Implementación diseño y verificar sincronismo

Implementar el diseño y verificar que este cumple con los tiempos especificados en la sección anterior.

A. Seleccione el archivo fuente *ALU.* en la ventana sources
B. Abra el **Design Summary** haciendo doble click en el proceso **View Design Summary** en la pestaña Processes. Ver figura 3.47.

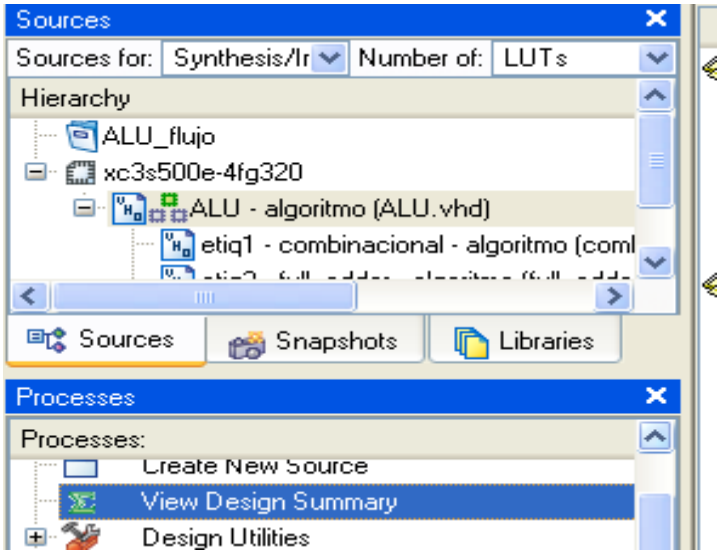


Figura 3.47 Activar resumen del diseño

C. Haga doble click en el proceso **Implement Design** en la pestaña Processes. Ver Figura 3.48.

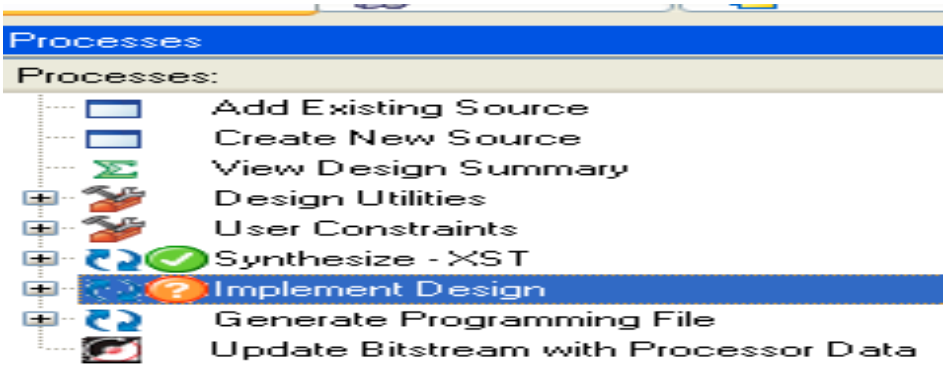


Figura 3.48 Implementar diseño

D. Observe que después de completar la implementación del diseño (implement Design) aparece al lado del proceso implementado un círculo verde con una flecha blanca, tal como se aprecia en el anterior grafico al lado de Synthessize – XST, la cual indica que terminó con éxito sin errores o advetencias.
G. Cierre el Design Summary.

3.12 Asignacion de pines del diseño al kit de desarrollo

Concretar la ubicación de los pines para los puertos del diseño a fin de que estén conectados correctamente en el Kit de desarrollo
Para asignar los puertos del diseño al paquete de pines haga lo siguiente:

ACompruebe que está seleccionado *alu-algoritmo* en la ventana sources
B.Haga doble click en el proceso **Assign Package Pins** encontrandolo en el grupo **User Constraints process**.ver figura 3.49

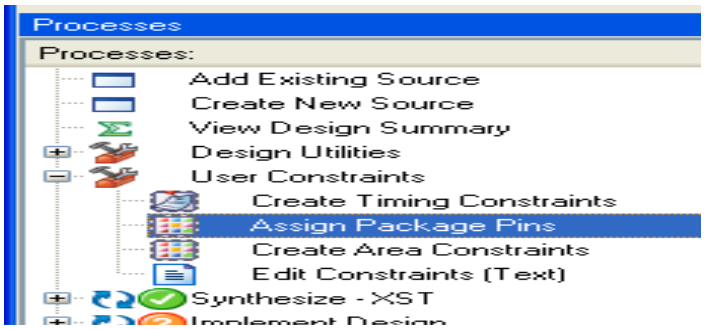


Figura 3.49 Ubicacion ventana asignación pines

C The Xilinx Pinout and Area Constraints Editor (PACE) abre(notese que al realizar este procedimiento se abre ventana con dos pestañas) seleccione la pestaña **Package View** ubicada en el area de trabajo. Ver figura 3.50

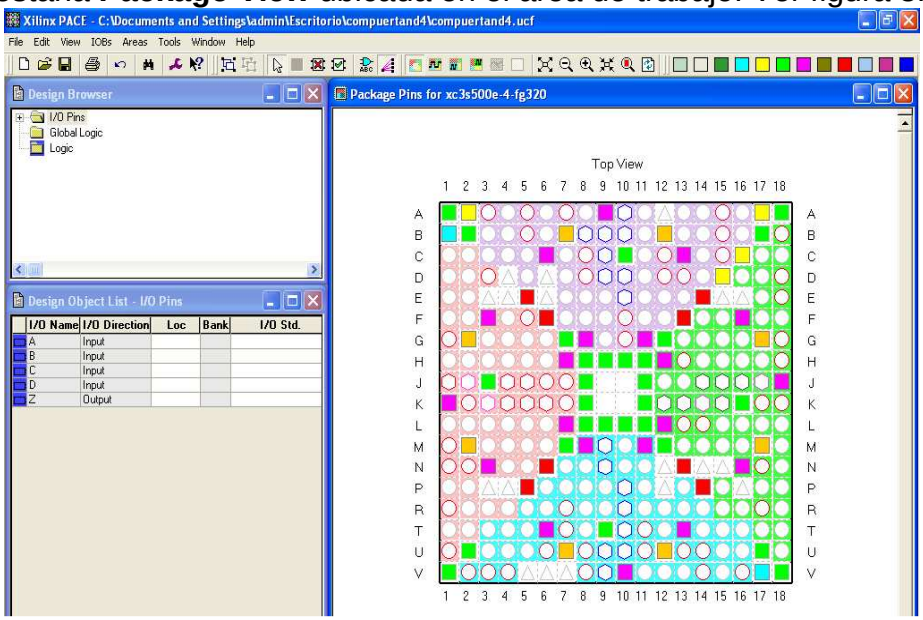


Figura 3.50 Ventana asignación de pines

D. En la ventana Design Object List, asignar una localización de un pin para cada pin en la columna **Loc**. (para este caso empezamos a utilizar simultáneamente la tarjeta electronica del kit de desarrollo, ver proceso de coneccion e instalacion de la misma)

Acontinuacion realizaremos la configuración de pines del diseño realizado con base a las entradas y/o salidas de la tarjeta electronica del kid de desarrollo, para este caso se utilizaran las configuraciones dadas en la Figura 3.51, (para mayor información ver tarjeta y manual de la misma), se debe tener en cuenta que para la tarjeta en el Caso de la entrada A se configura el pin como L13 debido a que así lo reconoce el software, aunque para la tarjeta electronica aparezca como SW0(l13).

Design Object List - I/O Pins

	I/O Name	I/O Direction	Loc	Bank	I/O Std.
	A	Input	L13	BANK1	
	B	Input	L14	BANK1	
	C	Output	F12	BANK0	
	Ci	Input	N17	BANK1	
	F	Output	F9	BANK0	
	S<0>	Input	D18	BANK1	
	S<1>	Input	K17	BANK1	
	S<2>	Input	H13	BANK1	

Figura 3.51 Configuracion pines para el proyecto

Para el caso de la salida de la ALU se configura el pin F12 (senalizado en la board LD0) , pin F9 (senalizado en la board LD0)

Note que el lugar del pin asignado se muestra en azul: tal como se aprecia en la Figura 3.52

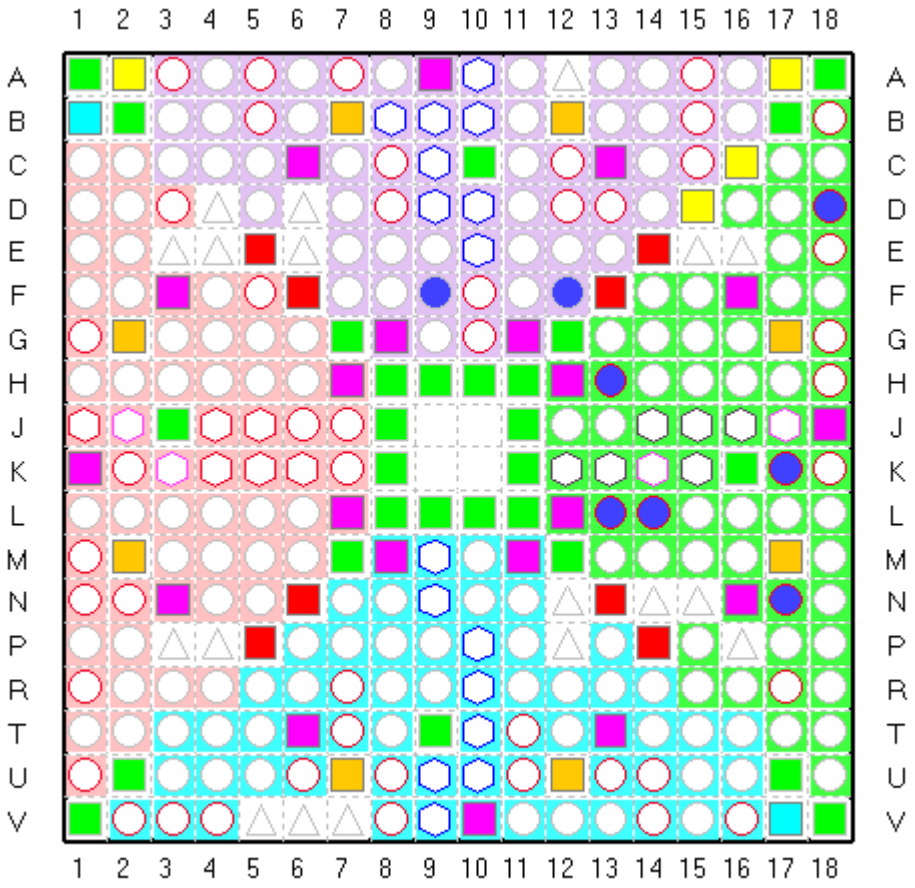


Figura 3.52 Planilla de asignación pines

E. Seleccione **File** →**Save** Aquí graba el archivo de asignación de pines
F. Cerrar PACE.
Note que **Implement Design** en la ventana processes tienen una marca color naranja junto a el, Indicando que están fuera de sincronismo con uno o varios de los archivos del diseño. Esto se debe a que la archivo UCF ha sido modificado haga doble click en **Implement Design**.al final del proceso la marca naranja se vuelve verde indicando proceso terminado.Por cada cambio que modifique los pines del diseño habra que volver a compilar la implemenrtacion del mismo. Ver figura 3.53

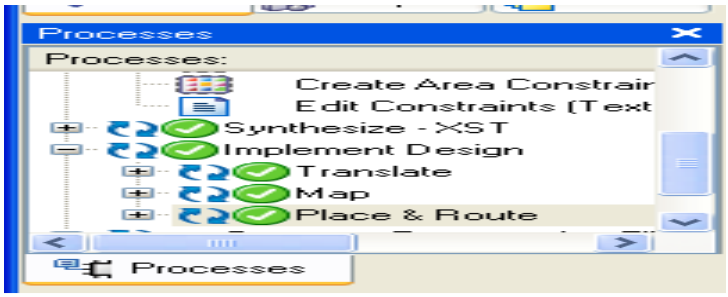


Figura 3.53 Implementación diseño terminado

Reimplementar diseño y verificación localización de pines

Reimplementar el diseño y verificar que los puertos de la compuerta diseñada estan enrutados al paquete de pines especificado en la sección anterior sera lo que realice así:.
A Revise el reporte de pines asignados desde la implementación previa haciendo lo siguiente: Habra el **Design Summary** haciendo doble click en el proceso **View Design Summary** en la ventana de procesos.Seleccione el **Pinout Report** y de click a la ventana **signal name**, aquí se muestra el resumen de la asignación de pines. Ver figura 3.54

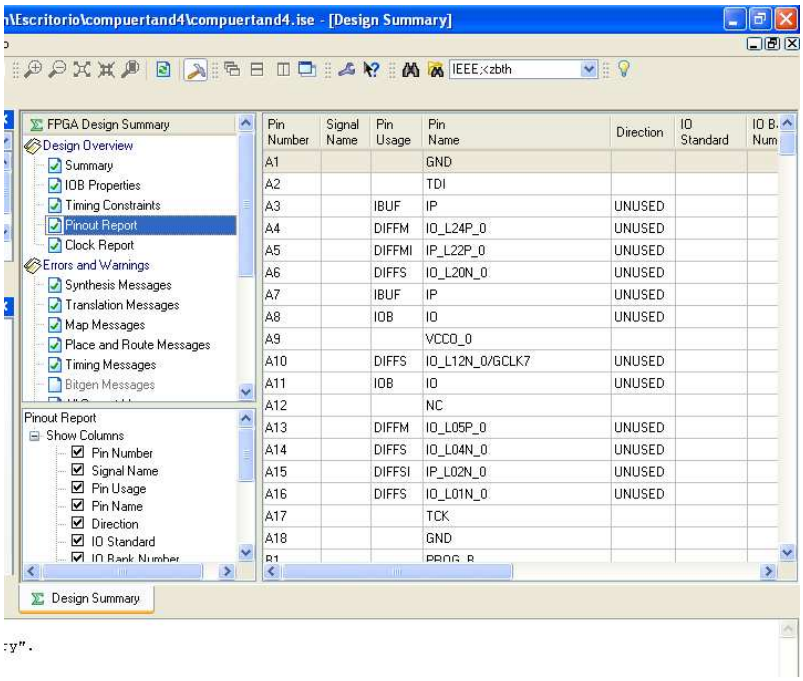


Figura 3.54 Cuadro resumen asignación de pines

B Reimplemente el diseño haciendo doble click en el proceso **Implement Design** (cuando realice cambios en la asignación de pines y le aparezca un círculo naranja con un signo de interrogación en blanco). Los pasos siguientes se hacen unicamente cuando se registro cambios en la asignación de pines, en caso contrario vaya directamente al paso **E**

C Seleccione el **Pinout Report** nuevamente y seleccione la columna **Signal Name** para ver el resumen de asignación de pines. Ver figura 3.55

Pin Number	Signal Name	Pin Usage	Pin Name	Direction
L13	A	IBUF	IP	INPUT
L14	B	IBUF	IP	INPUT
F12	C	IOB	IO_L06P_0	OUTPUT
N17	Ci	IBUF	IP	INPUT
F9	F	IOB	IO_L15N_0	OUTPUT
D18	S<0>	IBUF	IP/VREF_1	INPUT
K17	S<1>	IBUF	IP	INPUT
H13	S<2>	IBUF	IP	INPUT

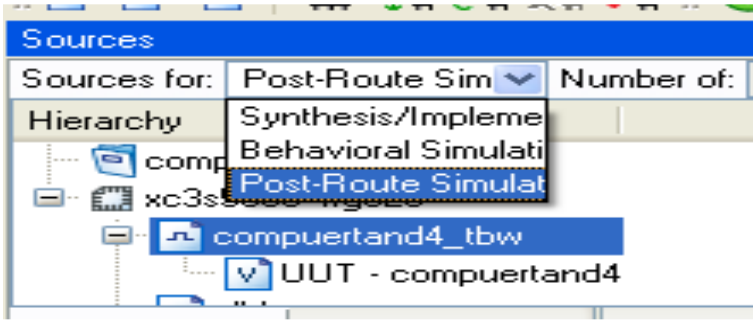
Figura 3.55 Asignación pines, luego de modificación diseño

D. Verifique que las señales ahora estan correctamente enrutadas.
E Cierre el **Design Summary**.

Verificar diseño usando tiempos de simulación

Use el mismo auto chequeo test bench waveform que usted creo en la sección anterior para verificar que la compuerta and diseñada después de esto a sido

completamente implementada. Verifique que los tiempos de simulación operan dentro del sincronismo especificado después que retardos lógicos y enrutamientos son acomodados.
Realice la simulación como sigue:
A. Seleccione el **Post-Route Simulation** desde la lista desplegada en ventana sources
B. Seleccione compuertand en la ventana Sources. Ver figura 3.56



C Realice la simulación haciendo doble click en el proceso **Simulate Post-Place & Route Model** establecido en el grupo de procesos del simulador Xilinxs ISE.Ver Figura 3.57

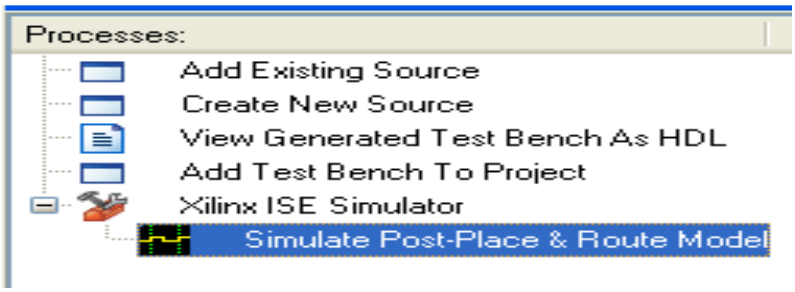


Figura 3.57 Comando Simulacion final

La simulación final del proyecto se observa en la figura 3.58

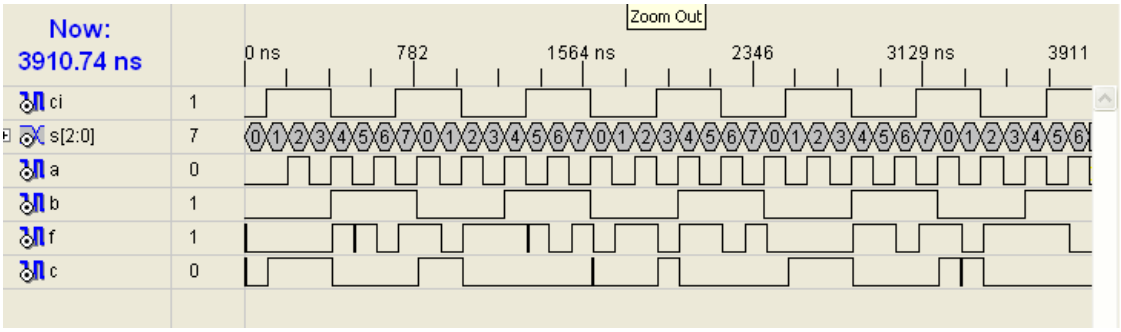


Figura 3.58 Simulación final del proyecto

D. Verifique que la compuertad4 esta funcionando directamente en el Puerto direccionado
E. Verifique que allí no hay errores reportados en la ventana de transcripción del simulador

F. **Zoom in** para ver el retardo actual desde el flanco del reloj para validar un cambio en la salida.
G. Cierre la simulación
Usted tiene completa la simulación de su diseño usando el ISE simulator
.

Proceso de diseño descargado en la tarjeta

Este es el último paso en el proceso de verificación del diseño. Esta sección provee simples instrucciones para descargar la compuerta diseñada al kit de desarrollo.

A Conecte el cable de potencia de 5Vdc a la entrada sobre el kit de desarrollo (J20).
B Conecte el cable de descarga entre el computador y el Kit de desarrollo.(J18) como se indica en la figura 3.59 .

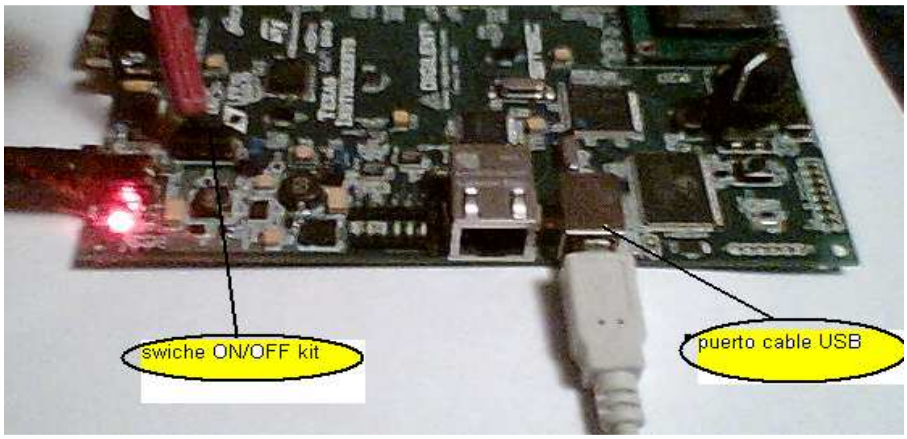


Figura 3.59 Conexión tarjeta kit de desarrollo

cuando el kit tiene comunicación activa con el computador un led cerca al puerto USB se ilumina como indica la Figura 3.60

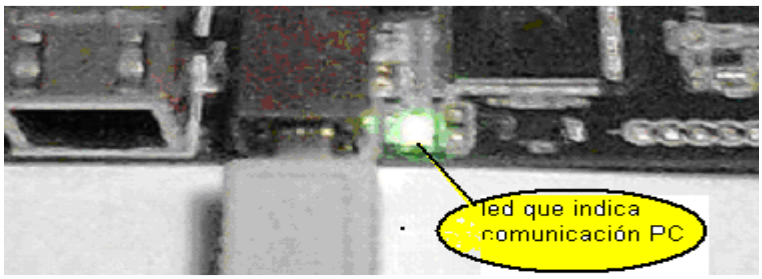


Figura 3.60 Indicacion comunicación activa con PC

C Seleccione **Synthesis/Implementation** desde la lista desplegada en la ventana de sources. Ver Figura 3.61

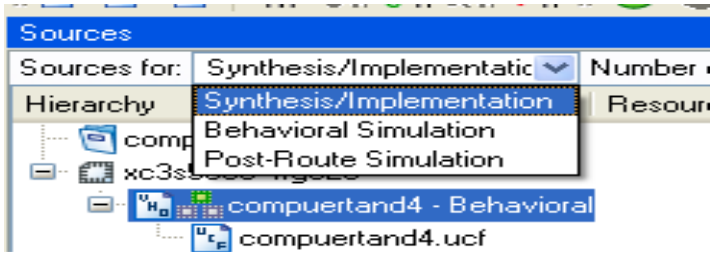


Figura 3.61

D Selecione *compuertand4.* en la ventana sources.

E En la ventana de procesos, haga click en signo "+" para desplegar el proceso **Generate Programming File** ver Figura 3.62

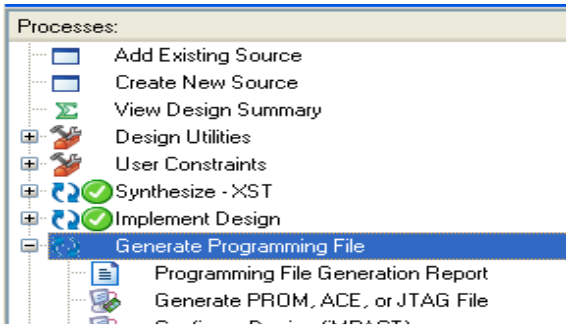


Figura 3.62 Abrir menu generacion archivo de programa

F Haga doble click en proceso **Configure Device (iMPACT)** para generar el archivo que se descargara en el kit de desarrollo. Ver figura 3.63

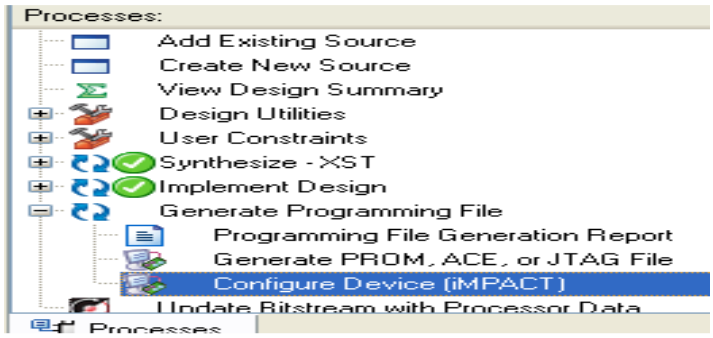


Figura 3.63 Generación archivo para descarga al kit

G La caja de dialogo de Xilinxs Web Talk puede abrirse durante este proceso. Haga click en **Decline.** Ver figura 3.64.

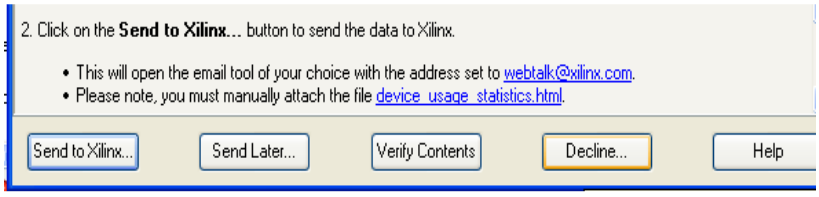


Figura 3.64

H Seleccione **"Disable the collection of device usage statistics fo this project only"** haga click en **OK.**Ver Figura 3.65

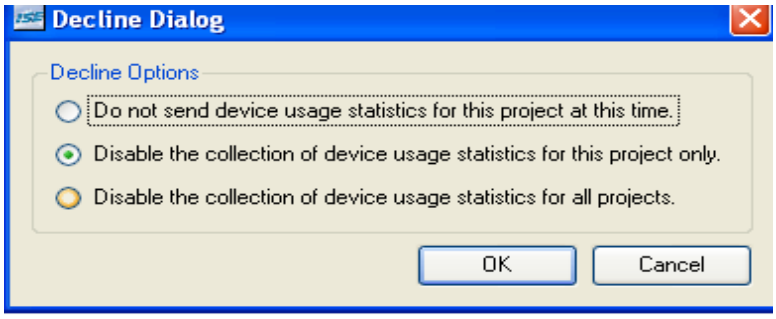


Figura 3.65

iMPACT .abre y la caja de dialogo de configurar dispositivo es visualizada

I En la caja de dialogo Welcome, seleccione **Configure devices using Boundary-Scan (JTAG)**.

J. Verifique que **Automatically connect to a cable and identify Boundary-Scan chain** es seleccionado.como se observa en la figura 3.66

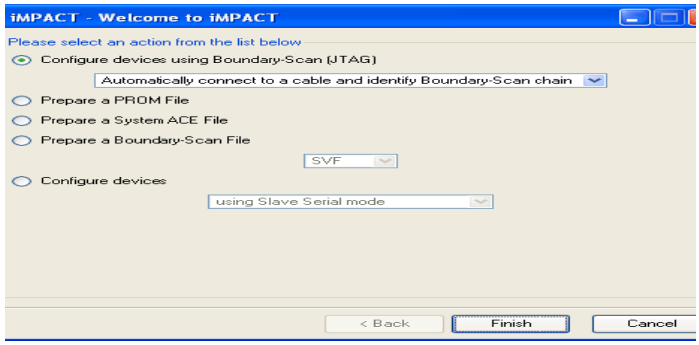


Figura 3.66 Conectar varios dispositivos por puerto JTAG

K. Haga click en **Finish**.

L. Si usted tiene un mensaje diciendo que allí hay dos dispositivos encontrados, haga click para continuar

Los dispositivos conectados a la cadena JTAG sobre la tarjeta pueden ser detectados y visualizados en la ventana IMPACT, para este caso el FPGA esta en primer lugar en color verde. Ver figura 3.67

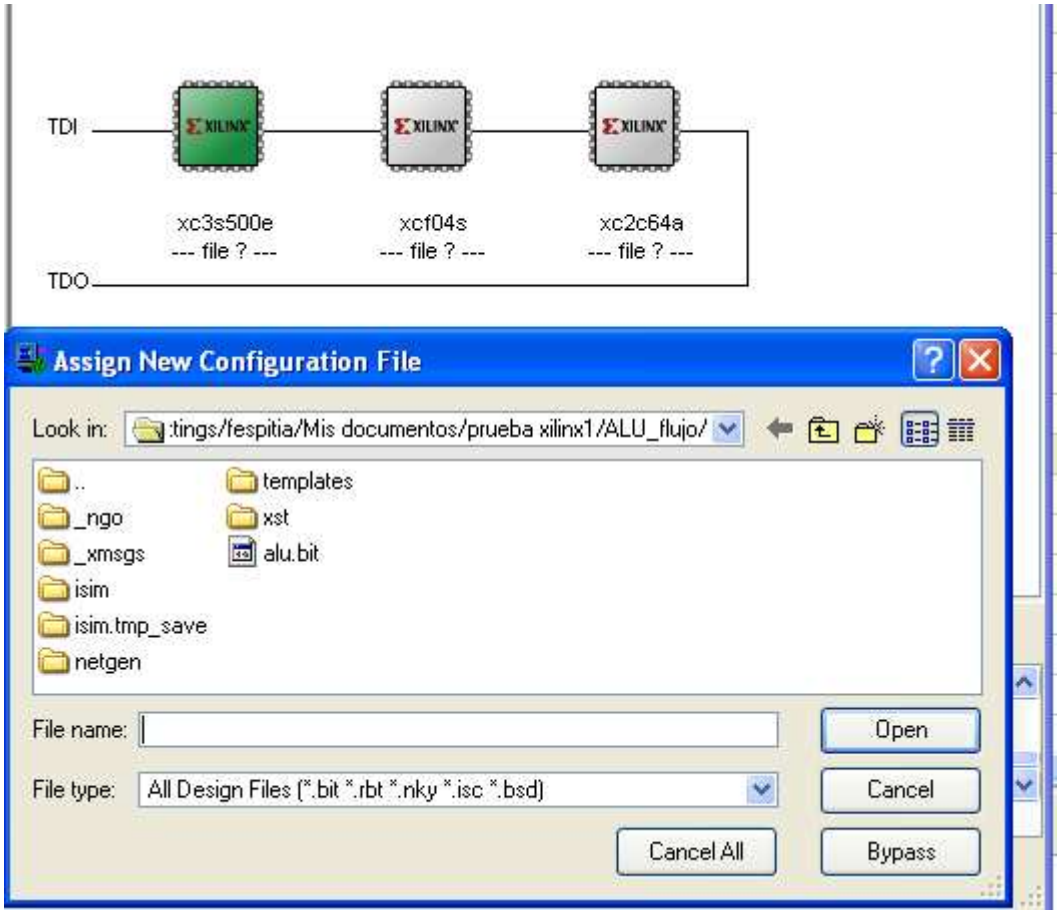


Figura 3.67 ubicación dispositivo FPGA en kit de desarrollo

M. La caja de dialogo **Assign New Configuration File** aparece. Para asignar un archivo de configuración al dispositivo XC3S500 en la cadena JTAG, Seleccione el archivo *compuertand4.*.bit y haga click en **Open** como se observa en la Figura 3.68.

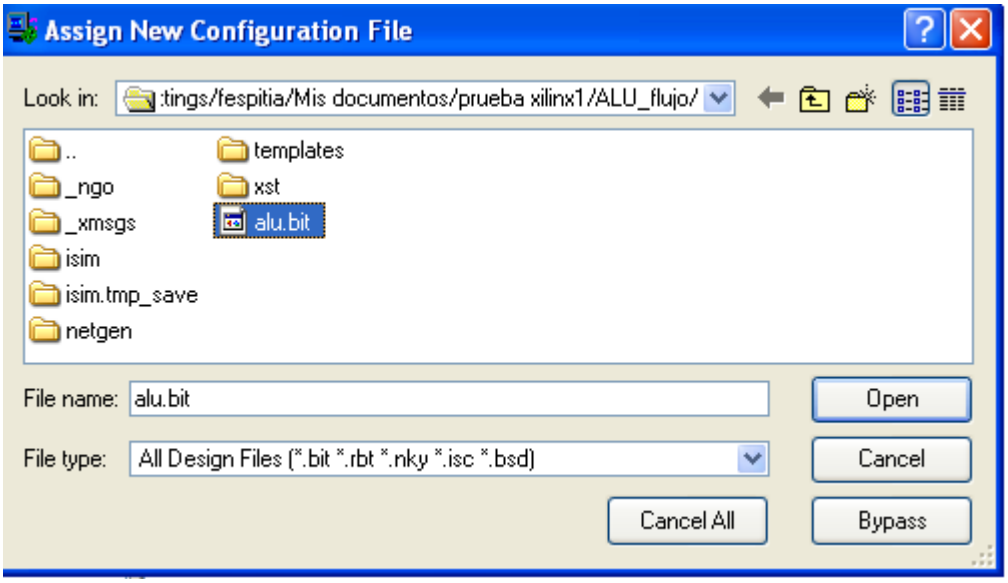


Figura 3.68 Asignar archivo .bit al FPGA

N. Si usted obtiene un mensaje de atención como el mostrado en la Figura 3.69, haga click en **OK**

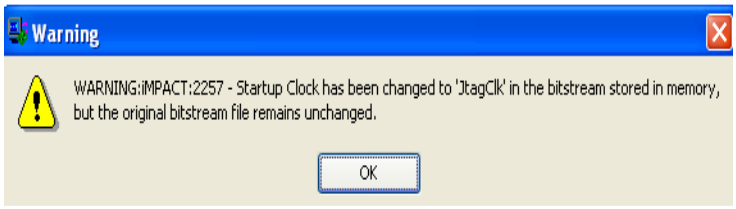


Figura 3.69

Nota: Como para este ejemplo no se van utilizar otros dispositivos detectados en el kit de desarrollo se le da click al boton bypass como se sugiere a continuación.
O. Seleccione **Bypass** para pasar por alto el segundo dispositivo detectado en la cadena el cual aparece ahora en verde como se muestra en la figura 3.70

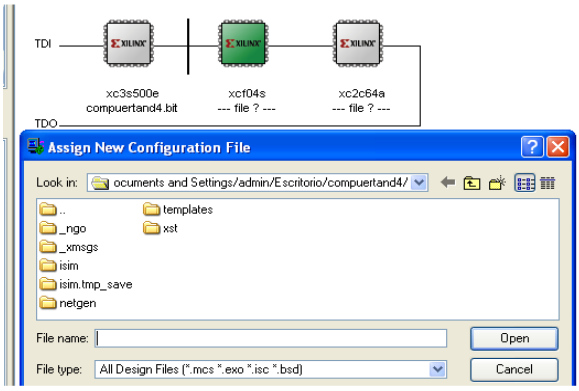


Figura 3.70 Asignacion archivo.bit a dispositivo 2

Ahora el tercer dispositivo en la cadena es resaltado en verde, seleccione nuevamente **Bypass** para pasarlo por alto como se indica en la Figura 3.71

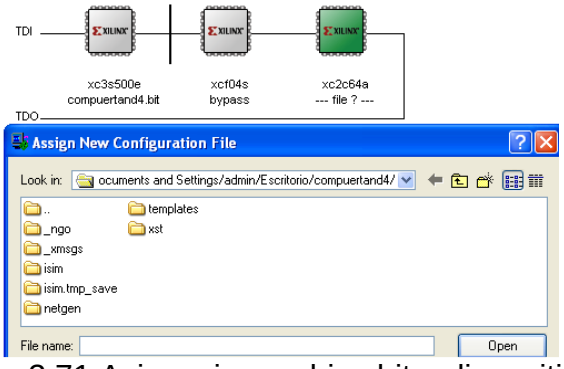


Figura 3.71 Asignacion archivo.bit a dispositivo 3

P.Haga click con el boton derecho del mouse sobre el icono del dispositivo XC3S500 y seleccione la opcion, **Program.** Ver figura 3.72

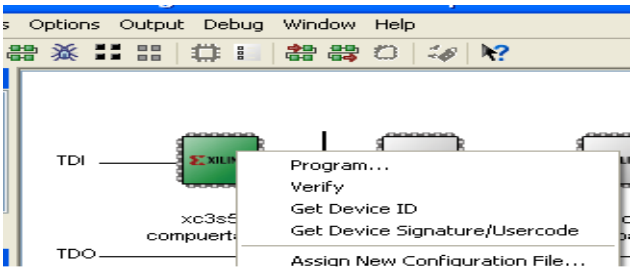


Figura 3.72 Programar FPGA ubicada sobre el kit

. La caja de dialogo **Programming Properties** se abre como muestra la Figura 3.73

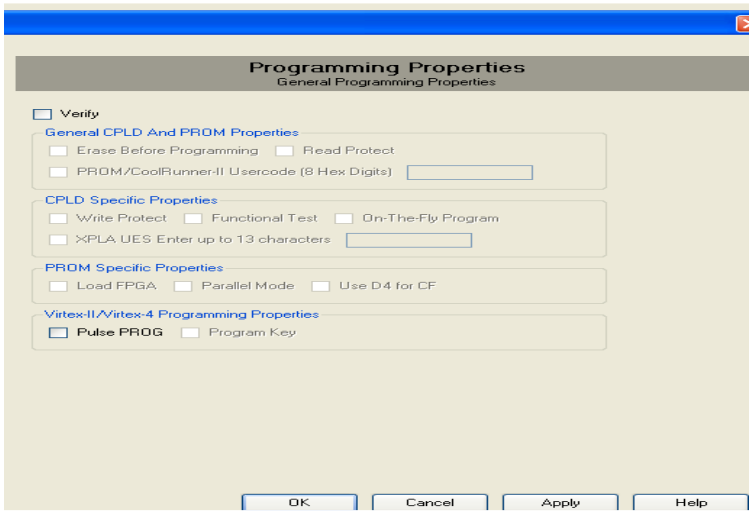


Figura 3.73 Propiedades de programación FPGA

Q. Haga click en **OK** para programar el dispositivo XC3S500.Cuando la programación es completada, el mensaje **Program Succeeded** es visualizado **Ver Figura 3.74**

Figura 3.74Programa descargado en FPGA XC3S500

Sobre el kit de desarrollo la salida LD-D (LED amarillo) se ilumina indicando que la FPGA esta programada.Ver Figura 3.75

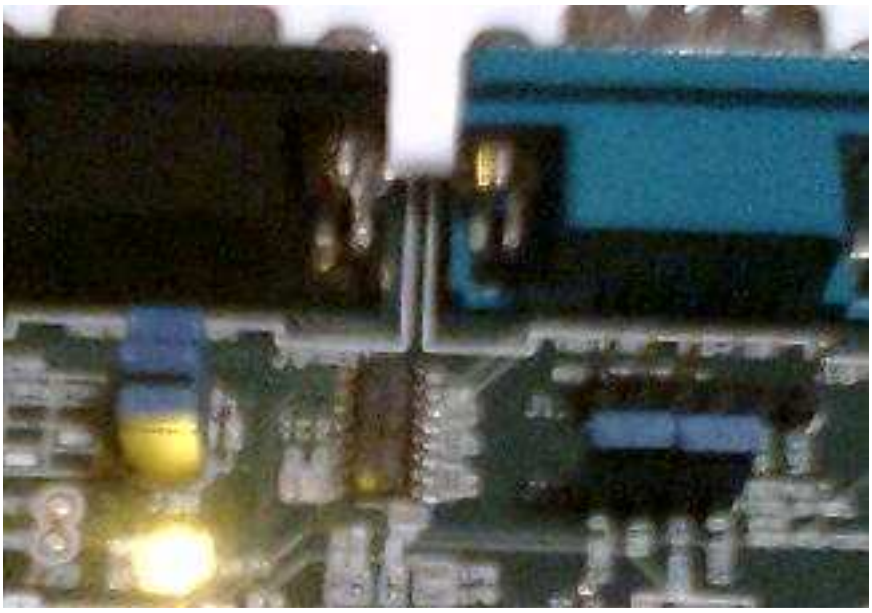


Figura 3.75 Indicación FPGA programada

En este momento puede jugar con las entradas de la tarjeta y ver su efecto sobre la salida. (teniendo en cuenta que el programa se descargo).ahora si las cuatro entradas estan en uno el LED 0 se ilumina y cambia según se modifiquen los swiches SW0,SW1,SW2, SW3.En la Figura 3.76 se indica cuando se activa o desactiva un swiche.

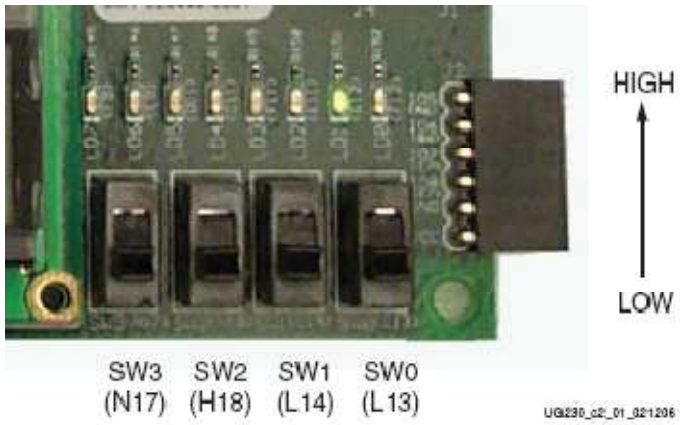


Figura 3.76 Activación swiches

R Cierre iMPACT sin salvar.

Aquí termino el proyecto # 1.de la compuerta AND de 4 entradas, si desacopla el cable USB el kit de desarrollo mantendra la configuración mientras no desenergize el mismo.

MONITORES CON FPGA

El monitor es el principal periférico de salida de una computadora. Estos se conectan a través de una tarjeta gráfica conocida con el nombre de adaptador o tarjeta de vídeo.

La imagen que podemos observar en los monitores está formada por una matriz de puntos de luz. Cada punto de luz reflejado en la pantalla es denominado como un píxel.

Clasificación según estándares de monitores

Según los estándares de monitores se pueden clasificar en varias categorías. Todos han ido evolucionando con el objetivo de ofrecer mayores prestaciones, definiciones y mejorar la calidad de las imágenes.

Monitores MDA: "Monochrome Display Adapter

Los monitores MDA por sus siglas en inglés "Monochrome Display Adapter" surgieron en el año 1981. Junto con la tarjeta CGA de IBM. Los **MDA** conocidos popularmente por los monitores monocromáticos solo ofrecían textos, no incorporaban modos gráficos.

Este tipo de monitores se caracterizaban por tener un único color principalmente verde. El mismo creaba irritación en los ojos de sus usuarios.

Características:

- Sin modo gráfico.
- Resolución 720_350 píxeles.
- Soporte de texto monocromático.
- No soporta gráfico ni colores.
- La tarjeta gráfica cuenta con una memoria de vídeo de 4 KB.
- Soporta subrayado, negrita, cursiva, normal, invisibilidad para textos.

Monitor CGA: "Color Graphics Adapter"

 Los monitores **CGA** por sus siglas en inglés "Color Graphics Adapter" o "Adaptador de Gráficos en Color" en español. Este tipo de monitores fueron comercializados a partir del año 1981, cuando se desarrollo la primera tarjeta gráfica conjuntamente con un estándar de IBM.

A pesar del lanzamiento de este nuevo monitor los compradores de PC seguían optando por los monitores MDA, ambos fueron lanzados al mercado en el mismo año existiendo competencia entre ellos. CGA fue el primero en contener sistema gráfico a color.

Características:

- Resoluciones 160x200, 320×200, 640×200 píxeles.
- Soporte de gráfico a color.
- Diseñado principalmente para juegos de computadoras.
- La tarjeta gráfica contenía 16 KB de memoria de vídeo.

Monitor EGA: "Enhanced Graphics Adapter",

 Por sus siglas en inglés "Enhanced Graphics Adapter", es un estándar desarrollado IBM para la visualización de gráficos, creado en 1984. Este nuevo monitor incorporaba una mayor amplitud de colores y resolución.

EGA incorporaba mejoras con respecto al anterior **CGA**. Años después también sería sustituido por un monitor de mayores características.

Características:

- Resolución de 640_350 píxeles.

- Soporte para 16 colores.
- La tarjeta gráfica EGA estándar traían 64 KB de memoria de vídeo.

Monitor VGA: "Video Graphics Array"

Los monitores VGA por sus siglas en inglés "Video Graphics Array", fue lanzado en 1987 por IBM. A partir del lanzamiento de los monitores VGA, los monitores anteriores empezaban a quedar obsoletos. El VGA incorporaba modo 256 con altas resoluciones.

Por el desarrollo alcanzado hasta la fecha, incluidas en las tarjetas gráficas, los monitores anteriores no son compatibles a los VGA, estos incorporan señales analógicas.

Características:

- Soporte de 720×400 píxeles en modo texto.
- Soporte de 640×480 píxeles en modo gráfico con 16 colores.
- Soporte de 320×200 píxeles en modo gráfico con 256 colores.
- Las tarjetas gráficas VGA estándares incorporaban 256 KB de memoria de vídeo.

Monitor SVGA:

SVGA denominado por sus siglas en inglés “Super Video Graphics Array”, también conocidos por “Súper VGA”. Estos tipos de monitores y estándares fueron desarrollados para eliminar incompatibilidades y crear nuevas mejoras de su antecesor VGA.

SVGA fue lanzado en 1989, diseñado para brindar mayores resoluciones que el VGA. Este estándar cuenta con varias versiones, los cuales soportan diferentes resoluciones.

Características:

- Resolución de 800×600, 1024_768 píxeles y superiores.
- Para este nuevo monitor se desarrollaron diferentes modelos de tarjetas gráficas como: ATI, GeForce, NVIDIA, entre otros.

Clasificación según tecnología de monitores

En cuanto al tipo de tecnología los monitores se pueden clasificar en varios aspectos. Estas evoluciones de la tecnología han sido llevadas a cabo en parte por el ahorro de energía, tamaño y por brindar un nuevo producto en el mercado.

Monitores CRT:

Está basado en un Tubo de Rayos Catódicos, en inglés “Cathode Ray Tube”. Es el más conocido, fue desarrollado en 1987 por Karl Ferdinand Braun.

Utilizado principalmente en televisores, ordenadores, entre otros. Para lograr la calidad que hoy cuentan, estos pasaron por diferentes modificaciones y que en la actualidad también se realizan.

Funcionamiento:

Dibuja una imagen barriendo una señal eléctrica horizontalmente a lo largo de la pantalla, una línea por vez. La amplitud de dicha señal en el tiempo representa el brillo instantáneo en ese punto de la pantalla.

Una amplitud nula, indica que el punto de la pantalla que se marca en ese instante no tendrá representando un píxel negro. Una amplitud máxima determina que ese punto tendrá el máximo brillo.

Ventajas:

- Excelente calidad de imagen (definición, contraste, luminosidad).
- Económico.
- Tecnología robusta.
- Resolución de alta calidad.

Desventajas:

- Presenta parpadeo por el refrescado de imagen.
- Consumo de energía.
- Generación de calor.
- Generación de radiaciones eléctricas y magnéticas.
- Alto peso y tamaño.

Pantallas LCD:

A este tipo de tecnología se le conoce por el nombre de pantalla o display LCD, sus siglas en inglés significan “Liquid Crystal Display” o “Pantalla de Cristal Líquido” en español. Este dispositivo fue inventado por Jack Janning.

Estas pantallas son incluidas en los ordenadores portátiles, cámaras fotográficas, entre otros.

Funcionamiento:

El funcionamiento de estas pantallas se fundamenta en sustancias que comparten las propiedades de sólidos y líquidos a la vez.

Cuando un rayo de luz atraviesa una partícula de estas sustancias tiene necesariamente que seguir el espacio vacío que hay entre sus moléculas como lo haría atravesar un cristal sólido pero a cada una de estas partículas se le puede aplicar una corriente eléctrica que cambie su polarización dejando pasar la luz o no.

Una pantalla LCD esta formada por 2 filtros polarizados colocados perpendicularmente de manera que al aplicar una corriente eléctrica deja pasar o no la luz. Para conseguir el color es necesario aplicar tres filtros más para cada uno de los colores básicos rojo, verde y azul.

Para la reproducción de varias tonalidades de color se deben aplicar diferentes niveles de brillo intermedios entre luz y no luz lo cual se consigue con variaciones en el voltaje que se aplica a los filtros.

Ventajas:

- Poco peso y tamaño.
- Buena calidad de colores.
- No contiene parpadeo.
- Poco consume de energía.
- Poca generación de calor.
- No genera radiaciones eléctricas y magnéticas.

Desventajas:

- Alto costo.
- Angulo limitado de visibilidad.
- Brillo limitado.
- Bajo tiempo de respuesta de píxeles.
- Contiene mercurio.

Pantallas Plasma:

La pantalla de plasma fue desarrollada en la Universidad de Illinois por Donald L. Bitzer y H. Gene Slottow.

Originalmente los paneles eran monocromáticos. En 1995 Larry Weber logró crear la pantalla de plasma de color. Este tipo de pantalla entre sus principales ventajas se encuentran una la mayor resolución y ángulo de visibilidad.

Funcionamiento:

El principio de funcionamiento de una pantalla de plasma consiste en iluminar pequeñas luces fluorescentes de colores para conformar una imagen. Las pantallas de plasma funcionan como las lámparas fluorescentes, en que cada píxel es semejante a un pequeño foco coloreado.

Cada uno de los píxeles que integran la pantalla está formado por una pequeña celda estanca que contiene un gas inerte (generalmente neón o xenón). Al aplicar una diferencia de potencial entre los electrodos de la celda, dicho gas pasa al estado de plasma.

El gas así cargado emite radiación ultravioleta (UV) que golpea y excita el material fosforescente que recubre el interior de la celda. Cuando el material fosforescente regresa a su estado energético natural, emite luz visible.

Ventajas:

- Excelente brillo.
- Alta resolución.
- Amplio ángulo de visión.
- No contiene mercurio.

- Tamaño de pantalla elevado.

Desventajas:

- Vida útil corta.
- Coste de fabricación elevado, superior a los LCD.
- Consumo de electricidad elevado.
- Poca pureza del color.
- Consumo energético y emisión de calor elevada.

¿Qué es la resolución de pantalla?

Se denomina al número de píxeles (o máxima resolución de imagen) que puede ser mostrada en la pantalla. Viene dada por el producto de las columnas (”X”), el cual se coloca al principio y el número de filas (”Y”) con el que se obtiene una razón. Por ejemplo podemos encontrar:

Estándar	Resolución (píxeles)
MDA	720_350
CGA	160_200 320x200 640x200
EGA	640_350
VGA	720x400 640x480 320x200
SVGA	800x600 1024_768

Los monitores han evolucionado conjuntamente con las tarjetas de vídeos. La necesidad de mostrar resoluciones mayores, con alta calidad de colores, ha llevado día a día a su desarrollo

1 Descripción del sistema

Esta aplicación implementa un simple generador de imagen en monitor VGA o LCD con una resolución de 640x 480

La imagen actualmente generada es simplemente una cuadricula de diferentes colores, pero el sistema ha sido generado por bloques para sustituir fácilmente el modulo generador de imagen con un modulo generado por alguna imagen diseñada por el usuario.

2) Descripción dela aplicación

Se tienen diferentes bloques , cada uno de ellos cumple una función .Los circuitos incluidos en esta aplicación son los siguientes:

- horizontal_ctr: Este cuenta los puntos de cada línea horizontal.
- vertical_ctr: Este cuenta las líneas de cada imagen.
- hsync_generator: Este genera la señal de sincronismo horizontal (hsync).
- vsync_generator : Este genera la señal de sincronismo vertical (vsync).
-blank_generator: Este genera el intervalo de blanqueamiento vertical y horizontal

- image_generator: Este genera la información de color para cada punto de la imagen, esto es , generar la imagen en si. El color depende del valor de la señal de color rojo, verde y azul. En este ejemplo solamente un bit es usado para cada señal, por esto son posibles 8 colores.
- top_test_vga_interface.vhd: Archivo principal, el cual contiene los demás bloques

3) Protocolo de prueba.
Una vez que el FPGA es configurado, usted puede conectar un monitor VGA o LCD al puerto J15 del kit de desarrollo, después de un corto tiempo este se sincroniza con el monitor VGA o LCD, mostrando la cuadricula de diferentes colores.

Descripción de los bloques

horizontal_ctr.vhd
Este bloque cuenta los puntos de cada línea horizontal para generar la imagen de 640x 480 pixels Por lo tanto para una imagen la frecuencia horizontal estandar es 31,5 Khz , lo cual corresponde a 31,746 *us* de tiempo del ciclo horizontal. La señal principal de reloj del FPGA tiene una frecuencia de 50 MHz (periodo 20 ns),entonces el contador de ciclos debe ser 1587 pulsos de reloj (31,746 us / 20 ns),-- Esto implica que el contador debe contar hasta 1586 y retornar a cero, luego debe tener 11 bits.

vertical_ctr.vhd –

Este bloque cuenta las líneas de cada imagen para generar una imagen de 640x 480 pixels. La frecuencia vertical estándar para una imagen es de 60 hz, lo cual corresponde a un ciclo vertical de 16,666 ms, este contador debe incrementarse con cada flanco de sincronismo horizontal, el cual es activado cada 31,746 *us*. Entonces el contador debe contar 525 ciclos de reloj (16,666 ms/31,746 *us)*, esto implica que el contador debe llegar hasta 524 y retornar a 0, luego este registro debe tener 10 bits

hsync_generator.vhd –

Generador de sincronismo horizontal para imagen VGA
El tamaño de la imagen es 640 x 480 píxeles. La frecuencia horizontal estandar para cada imagen es 31.5 Khz lo cual corresponde a 31,746 *us* para un ciclo horizontal. Un máximo de 25,17 *us* corresponden para trazar una línea de píxeles que componen la imagen, un mínimo de al menos 6.6 us corresponde a un intervalo interno horizontal. En este intervalo blanco un pulso de sincronismo horizontal debe ser generado, el cual tiene un mínimo de duración de 3.77 us. Como señal principal del reloj tiene una frecuencia de 50 MHz (periodo 20 ns) se tiene un máximo de 1.258 ciclos del reloj (25,17us/20ns) para enviar el píxel de color para la imagen. Esto permite enviar 1,258 píxel si la resolución del monitor soporta este.
El pulso de sincronismo horizontal(o nivel cero) debe ser enviado centrado en el intervalo blanco horizontal y debe tener una duración de 188 ciclos de reloj (3.77 us / 20 ns).
La secuencia relativa al estado del contador horizontal debe ser la siguiente:
– 0 a 1,257 pixels de imagen a color
-- Intervalo de 1,258 to 1,586: blanco horizontal.
-- intervalo de 1,327 to 1,515: nivel zero de sincronismo horizontal.

vsync_generator.vhd

Generador de sincronismo vertical para imagen VGA
El tamaño de la imagen es 640 x 480 píxeles. La frecuencia vertical estandar para una imagen es 60 HZ, el cual corresponde a 16,666 ms de ciclo vertical, un máximo de 15.25 ms corresponde para una línea de pixels que componen la imagen. Un mínimo de al menos 1,534 ms corresponden al intervalo de retorno vertical, en este intervalo un pulso de sincronismo debe ser generado, el cual tiene una duración mínima de 64 us. El contador vertical debe incrementar este estado con cada flanco positivo de sincronismo horizontal, el cual es activado cada 31,746 us
Entonces se tiene un máximo de 480 ciclos de reloj(15.25 ms / 31.746 us) para enviar una línea de color de la imagen. Esto permite enviar exactamente las 480 líneas de imagen de una imagen de 640 x480.

El pulso de sincronismo vertical (o nivel cero) debería enviarse centrado en el intervalo barrido vertical y debería tener una duración de 2 ciclos de

sincronismo horizontal(64 us / 31.746 us).Se tiene decidido enviar un pulso de 3 ciclos de sincronismo horizontal, para asegurar una apropiada operación.

.Luego, la secuencia de señal relativa al estado del contador vertical debe ser la siguiente
estado 0 a 479 líneas de imagen de color
estado 480 a 524 intervalo de barrido vertical
estado 500 a 502 pulso sincronismo de nivel cero

blank_generator.vhd –

Generador de intervalo blanco para generación de imagen VGA .
La imagen es de tamaño 640x480,La frecuencia estandar horizontal para una imagen es 31.5 KHz, el cual corresponde a 31,746 us de ciclo horizontal
Un máximo de 25,17 us corresponden a los puntos de color de cada línea que compone la imagen. Un mínimo de al menos 6.6 us corresponde a un intervalo de blanco horizontal. Una frecuencia vertical estandar para cada imagen es 60 Hz, el cual corresponde a 16,666 ms de tiempo de ciclo vertical.
Un máximo de 15.25 milisegundos corresponden a las líneas que componen la imagen. Un mínimo de al menos 1,534 ms corresponden a un intervalo de barrido vertical. Entonces, la secuencia del intervalo de blanco depende de los contadores horizontal y vertical en la siguiente forma:

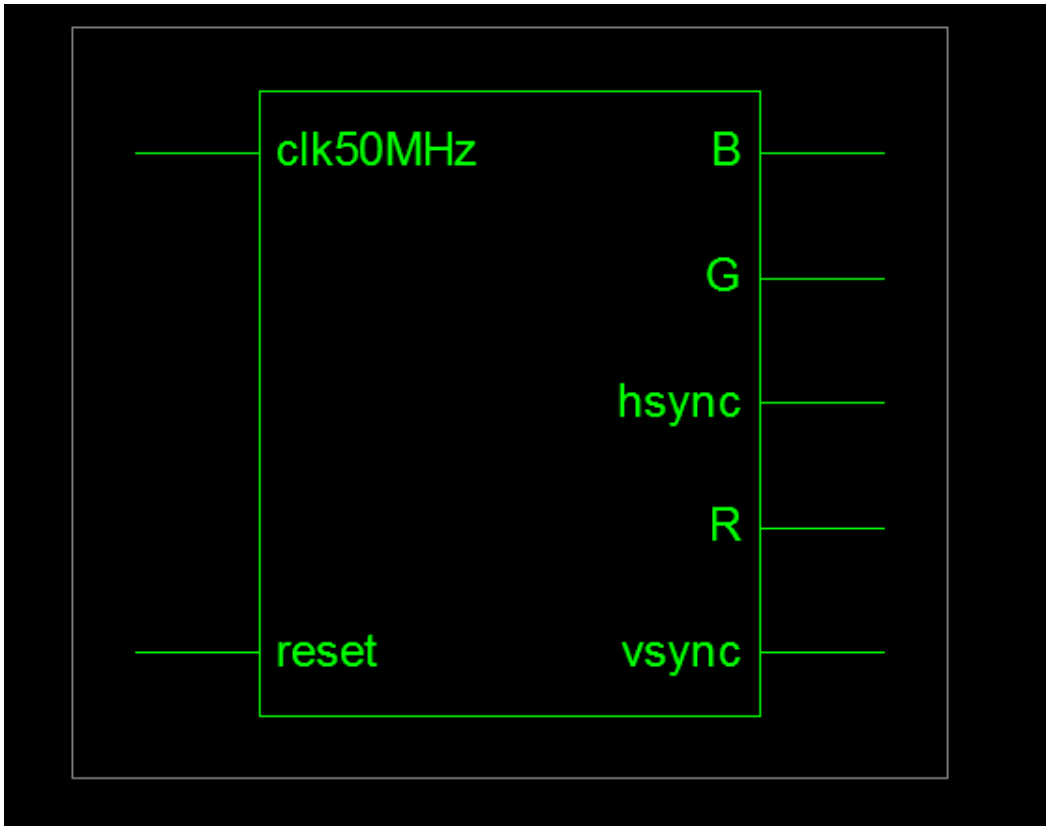
Estado 1258 a 1586 del contador horizontal : intervalo de blanco horizontal.
Estado 480 to 524 del contador vertical: intervalo de blanco vertical.

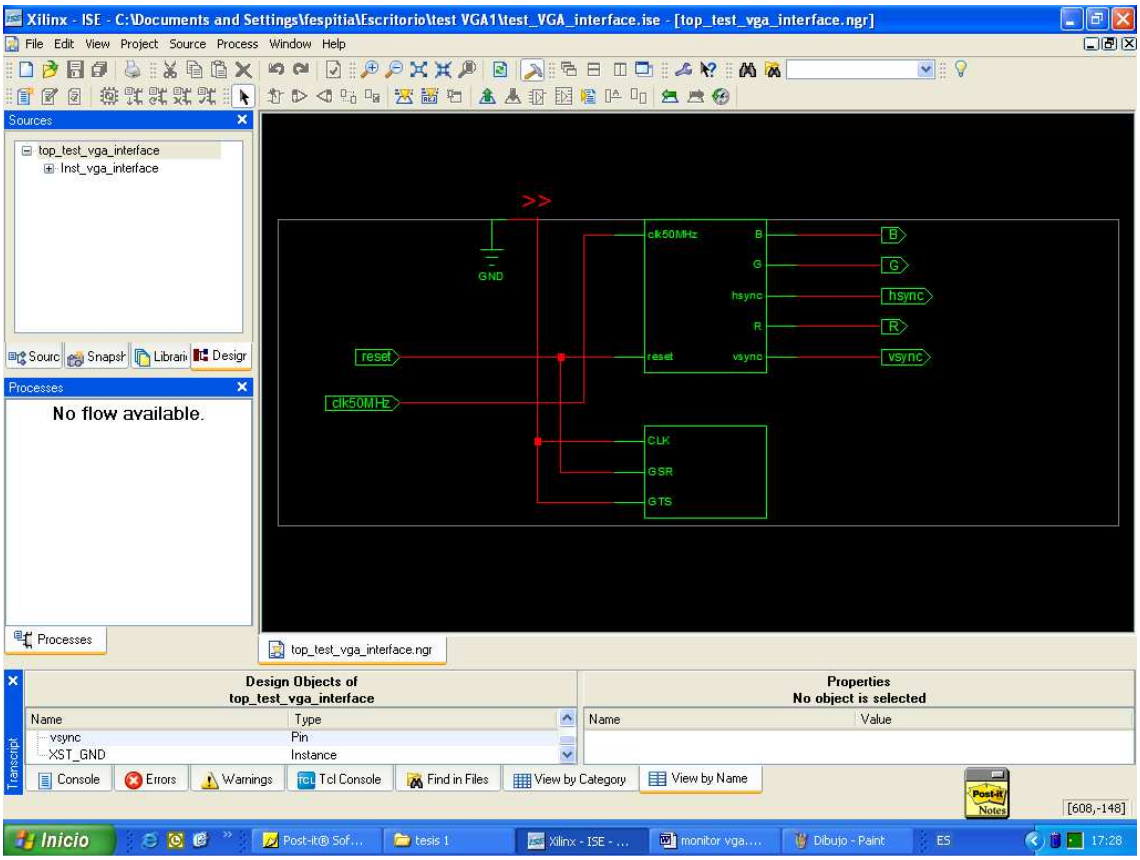
RGB_generator.vhd

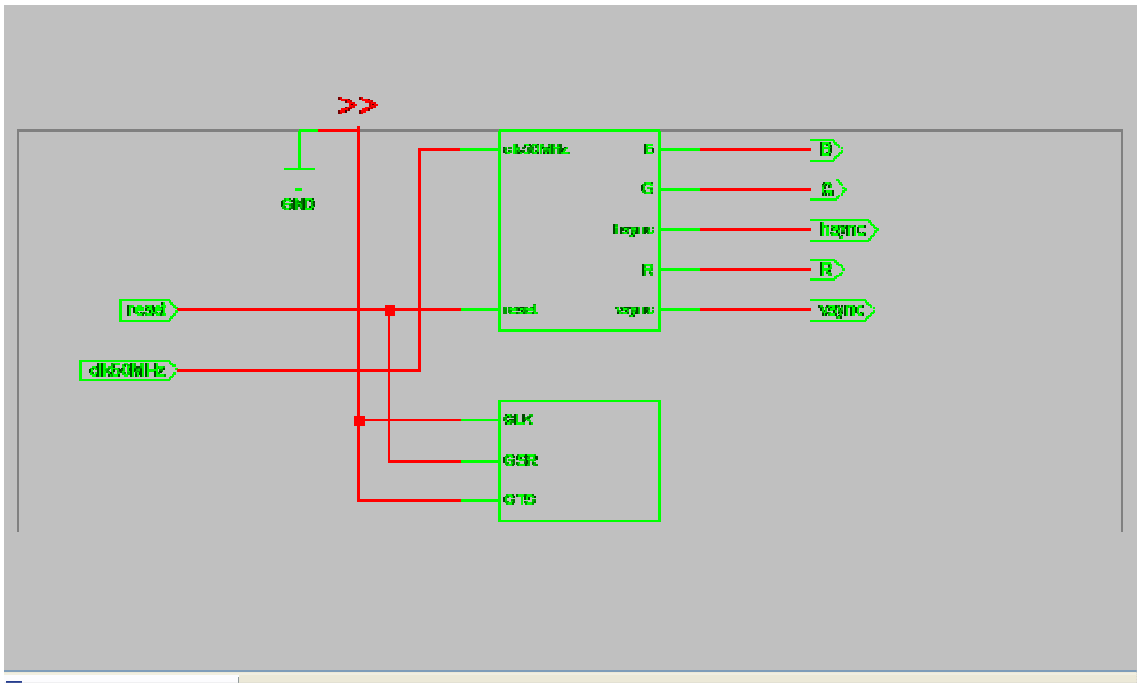
Generador RGB para generación de imagen VGA.
Este circuito genera la señal roja (R), verde (R) y azul(B) a partir de la información del color del píxel y la señal de blanco
En esta aplicación solamente un bit es usado para cada señal, luego solamente 8 colores son posibles

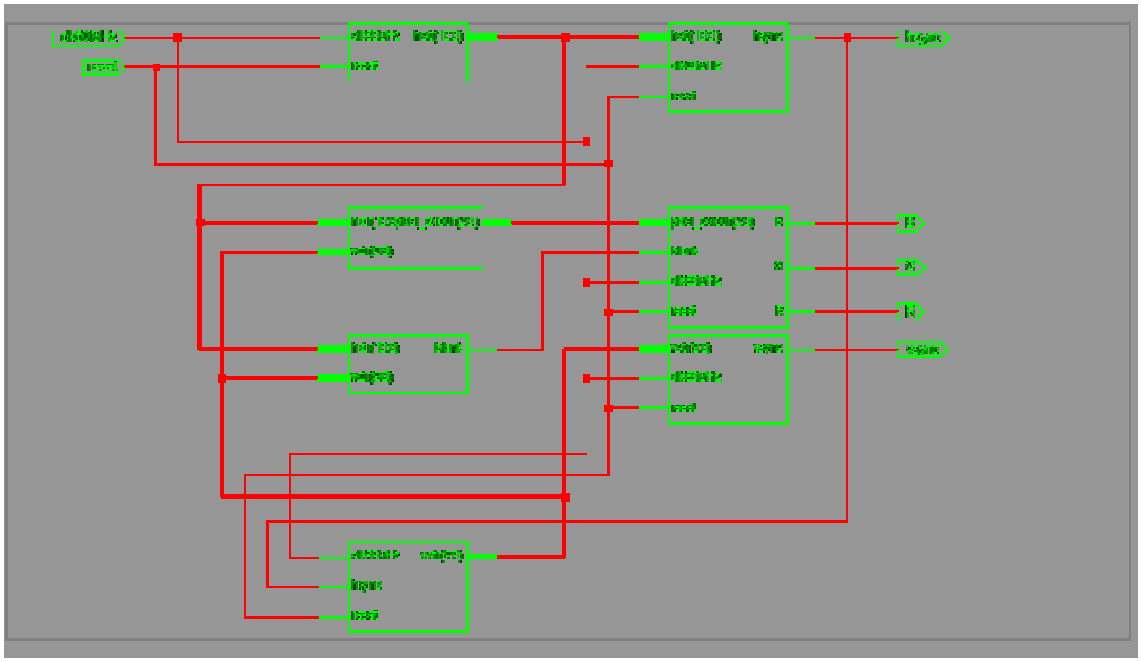
Los colores obtenidos son una función de los valores R,G y B

-- 000: negro
-- 001: azul
-- 010: verde
-- 011: azul claro
-- 100: rojo
-- 101: púrpura
-- 110: amarillo
-- 111: blanco


clk50MHz
B
G
hsync
R
reset
vsync


Xilinx - ISE - C:\Documents and Settings\fespitia\Escritorio\test VGA1\test_VGA_interface.ise - [top_test_vga_interface.ngr]
File Edit View Project Source Process Window Help
Sources
top_test_vga_interface
Inst_vga_interface
Processes
No flow available.
GND
clk50MHz
B
G
hsync
R
vsync
reset
CLK
GSR
GTS
top_test_vga_interface.ngr
Design Objects of top_test_vga_interface
Name
Type
vsync
Pin
XST_GND
Instance
Properties
No object is selected
Value
Console
Errors
Warnings
Tcl Console
Find in Files
View by Category
View by Name
[608,-148]
Inicio
tesis 1
monitor vga....
Dibujo - Paint
ES
17:28





54 www.xilinx.com **Spartan-3E Starter Kit Board User Guide**
UG230 (v1.0) March 9, 2006
Signal Timing for a 60 Hz, 640x480 VGA Display

Senales de tiempo para un monitor VGA de 640x480

Un monitor VGA basado en tubo de rayos catodicos usa amplitud modulada, moviendo un rayo de electrones para visualizar la información sobre una pantalla a phosphor-coated screen.Un monitor LCD usa un arreglo de swiches que pueden permitir un voltaje atraves de una pequeña cantidad de cristal liquido, permitiendo cambios de luz , thereby costantemente a traves de un cristal sobre una base through the crystal on a pixel-by-pixel basis. Although por lo tanto la siguiente descripción se limita a monitores ctr y lcd's tienen evolved usar la misma señal de tiempos como monitor CTR.Consecuentemente la siguiente discusión aplica para ambos CRTs y LCDs.

Dentro de un monitor CTR, senales de corriente pasan atraves de bobinas para producir campos magneticos que deflextan el rayo de electrones para convertir la superficie de la pantalla en un patron de rastreo horizontal de izquierda a derecha y verticalmente de arriba abajo. Como indica la figura 1

o transverse., esta información solamente se visualiza cuando el movimiento hacia adelante del rayo de izquierda a derecha y de arriba a abajo y no durante el tiempo que retorna el rayo al lado izquierdo y esquina superior de la pantalla. Much of the potential display time is therefore lost in *blanking* periods when the beam is reset and stabilized to begin a new horizontal or vertical display pass.

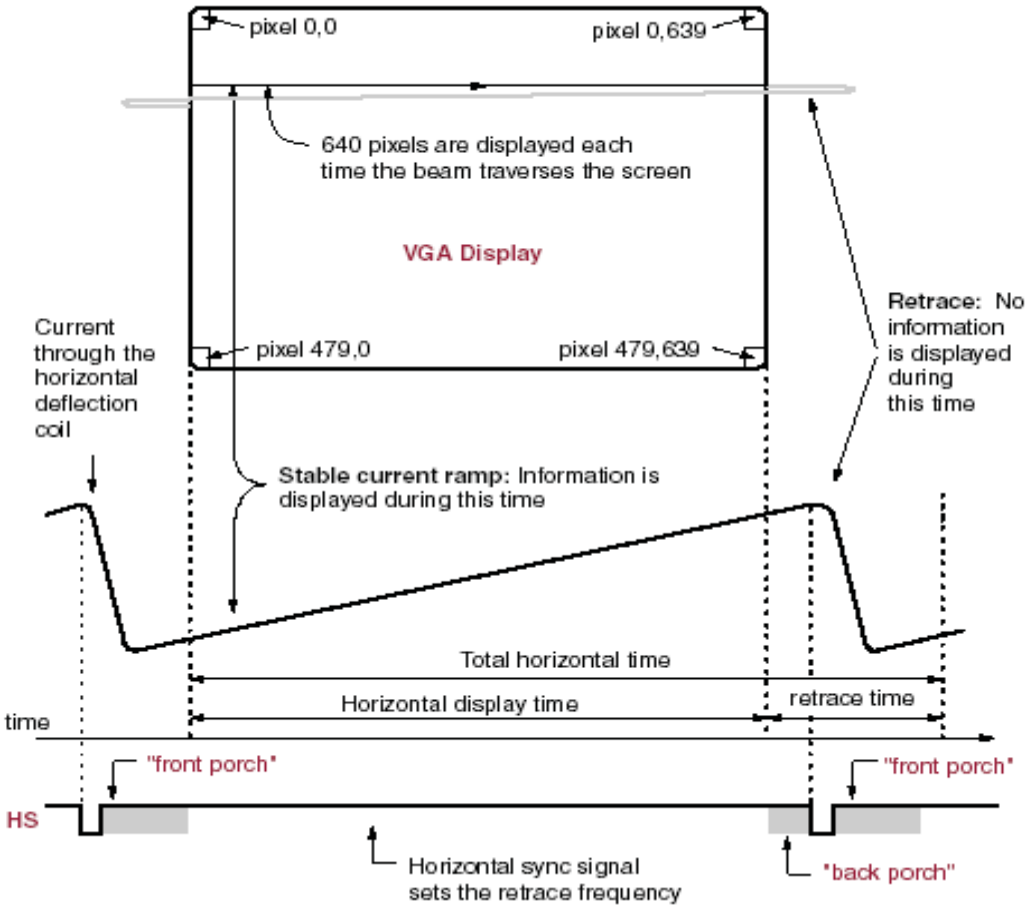


Figura 1 Tiempos para la pantalla CTR

Signal Timing for a 60 Hz, 640x480 VGA Display R

The display resolution defines the size of the beams, the frequency at which the beam traces across the display, and the frequency at which the electron beam is modulated Modern VGA displays support multiple display resolutions, and the VGA controllerdictates the resolution by producing timing signals to control the raster patterns. Thecontroller produces TTL-level synchronizing pulses that set the frequency at which currentflows through the deflection coils, and it ensures that pixel or video data is applied to theelectron guns at the correct time.

Video data typically comes from a video refresh memory with one or more bytes assigned to each pixel location. The Spartan-3E Starter Kit board uses three bits per pixel, producing one of the eight possible colors shown in Table 6-1. The controller indexes into the video data buffer as the beams move across the display. The controller then retrieves and applies video data to the display at precisely the time the electron beam is moving across a given pixel.

640 pixels are displayed each
time the beam traverses the screen
pixel 0,639 pixel 0,0
pixel 479,0 pixel 479,639

As shown in Figure 6-2, the VGA controller generates the horizontal sync (HS) and vertical sync (VS) timings signals and coordinates the delivery of video data on each pixel clock. The pixel clock defines the time available to display one pixel of information. The VS signal defines the *refresh* frequency of the display, or the frequency at which all information on the display is redrawn. The minimum refresh frequency is a function of the display's phosphor and electron beam intensity, with practical refresh frequencies in the 60 Hz to 120 Hz
range. The number of horizontal lines displayed at a given refresh frequency defines the horizontal *retrace* frequency.

VGA Signal Timing

The signal timings in Table 6-2 are derived for a 640-pixel by 480-row display using a
25 MHz pixel clock and 60 Hz ± 1 refresh. Figure 6-3 shows the relation between each of
the timing symbols. The timing for the sync pulse width (TPW) and front and back porch
intervals (TFP and TBP) are based on observations from various VGA displays. The front
and back porch intervals are the pre- and post-sync pulse times. Information cannot be
displayed during these times.

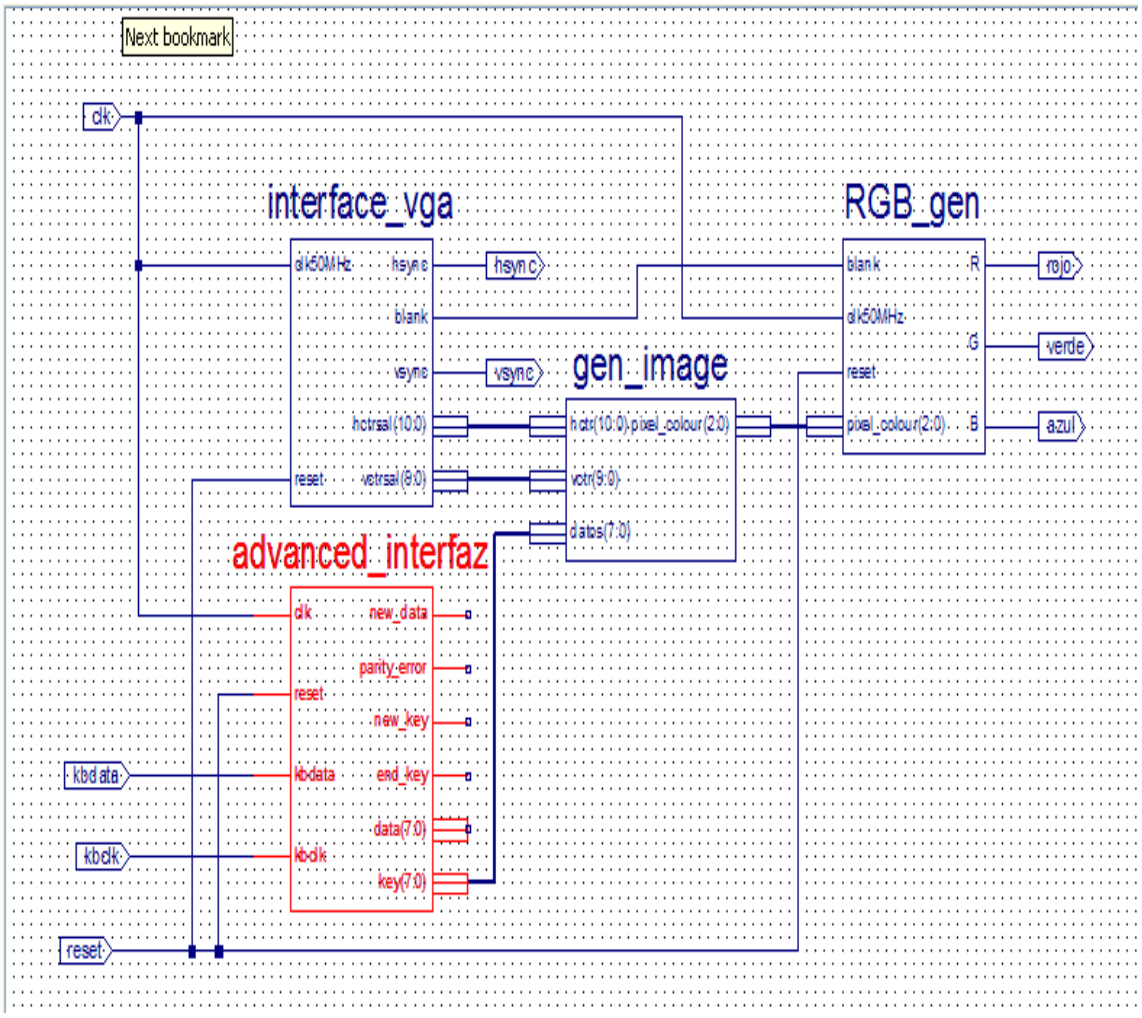
Generally, a counter clocked by the pixel clock controls the horizontal timing. Decoded
counter values generate the HS signal. This counter tracks the current pixel display
location on a given row.

A separate counter tracks the vertical timing. The vertical-sync counter increments with
each HS pulse and decoded values generate the VS signal. This counter tracks the current

display row. These two continuously running counters form the address into a video
display buffer. For example, the on-board DDR SDRAM provides an ideal display buffer.
No time relationship is specified between the onset of the HS pulse and the onset of the VS
pulse. Consequently, the counters can be arranged to easily form video RAM addresses, or
to minimize decoding logic for sync pulse generation.
Table 6-2: **640x480 Mode VGA Timing**
Symbol Parameter
Vertical Sync Horizontal Sync
Time Clocks Lines Time Clocks
TS Sync pulse time 16.7 ms 416,800 521 32 µs 800
TDISP Display time 15.36 ms 384,000 480 25.6 µs 640
TPW Pulse width 64 µs 1,600 2 3.84 µs 96
TFP Front porch 320 µs 8,000 10 640 ns 16
TBP Back porch 928 µs 23,200 29 1.92 µs 48
Figure 6-3: **VGA Control Timing**
Tfp Tdisp
TS
Tpw
Tbp
UG230_c6_03_021706

UCF Location Constraints
Figure 6-4 provides the UCF constraints for the VGA display port, including the I/O pin
assignment, the I/O standard used, the output slew rate, and the output drive current.
Related Resources
□□VESA
http://www.vesa.org
□□VGA timing information
http://www.epanorama.net/documents/pc/vga_timing.html
Figure 6-4: **UCF Constraints for VGA Display Port**
```
NET "VGA_RED" LOC = "H14" | IOSTANDARD = LVTTL | DRIVE = 8 | SLEW = FAST ;
NET "VGA_GREEN" LOC = "H15" | IOSTANDARD = LVTTL | DRIVE = 8 | SLEW = FAST ;
NET "VGA_BLUE" LOC = "G15" | IOSTANDARD = LVTTL | DRIVE = 8 | SLEW = FAST ;
NET "VGA_HSYNC" LOC = "F15" | IOSTANDARD = LVTTL | DRIVE = 8 | SLEW = FAST ;
NET "VGA_VSYNC" LOC = "F14" | IOSTANDARD = LVTTL | DRIVE = 8 | SLEW = FAST ;
```

Pin Number	Signal Name	Pin Usage	Pin Name	Direction	IO Standard	I
G15	B	IOB	IO_L18P_1	OUTPUT	LVCMOS25	
C9	clk50MHz	IBUF	IO_L14P_0/GCLK10	INPUT	LVCMOS25	
H15	G	IOB	IO_L17N_1	OUTPUT	LVCMOS25	
F15	hsync	IOB	IO_L21P_1	OUTPUT	LVCMOS25	
H14	R	IOB	IO_L17P_1	OUTPUT	LVCMOS25	
D18	reset	IBUF	IP/VREF_1	INPUT	LVCMOS25	
F14	vsync	IOB	IO_L21N_1	OUTPUT	LVCMOS25	
A8		IOB	IO	UNUSED		
A9			VCCO_0			


Next bookmark
clk
interface_vga
clk50MHz
hsync
blank
vsync
hctrsal(10:0)
vctrsal(9:0)
reset
hsync
vsync
gen_image
hctr(10:0)
vctr(9:0)
datos(7:0)
pixel_colour(2:0)
RGB_gen
blank
clk50MHz
reset
pixel_colour(2:0)
R
G
B
rojo
verde
azul
advanced_interfaz
clk
reset
kbdata
kbclk
new_data
parity_error
new_key
end_key
data(7:0)
key(7:0)
kbdata
kbclk
reset

ESTRATEGIA METODOLOGICA

Etapa de Análisis

Se espera seleccionar y comprar el kit de desarrollo, para esto se realizaran las siguientes actividades:

- Recopilación y revisión de información relacionada con temas y tópicos requeridos para el desarrollo del proyecto.
- Identificación de los diferentes prototipos disponibles en el mercado de dispositivos lógicos programables.
- Selección del prototipo a usar para el desarrollo de las aplicaciones sobre FPGAs.
- Cotización del Kit de desarrollo.
- Compra e importación del kit de desarrollo.

Etapa de diseño

Se espera hacer el curso basico y diseñar las cuatro aplicaciones para lo cual se realizaran las siguientes actividades:

- Estudiar sobre programación VHDL.
- Estudiar el manual de características del prototipo seleccionado.
- Realizar pruebas de programación FPGAs.
- Diseñar aplicación para lógica combinacional.
- Diseñar aplicación para lógica secuencial.
- Diseñar 2 aplicaciones que manejen dispositivos periféricos del kit de desarrollo.

Etapa de Implementación

En está se realizaran las siguientes actividades:

- Implementar la aplicaciones para lógica combinacional, lógica secuencial, en Kit de desarrollo y realizar pruebas funcionales sobre FPGAs..
- .Redacción los documentos didácticos de las cuatro aplicaciones.
- Redacción los documentos del curso básico y de las cuatro aplicaciones.
- Entrega de kit de desarrollo , documentos del curso básico y aplicaciones en FPGA

Instrumentos para el desarrollo del proyecto se cuenta con el laboratorio de la Universidad dotado de instrumentos de medición tales como multímetros digitales, osciloscopios, generadores de señales, frecuencímetros.

- Software
  - Ambiente de software integrado(ISE) incluido con el kit de inicio, Xilinx ISE & EDK

Medios se utilizaran los siguientes:
Investigador Principal Rubén Darío Cárdenas Espinosa. Docente de la Universidad Autónoma de Manizales
Coinvestigador Héctor Fernando Espitia.

- Equipos
  - Kit de inicio HW-SPAR3E-SK-US de la familia SPARTAN 3E de XILINX.
  - Computador de proceso para desarrollo de la aplicación. Y con acceso a internet
- Bibliografía Para documentación del presente trabajo se contó con el material bibliográfico relacionado en el capitulo bibliografía y consultas en internet.

CONCLUSIONES

Con la aparición de los lenguajes de descripción hardware y sobre todo, la posibilidad de sintetizar este hardware en una FPGA,se eliminan muchas de las diferencias entre hardware y software, convirtiendo los diseños digitales en programas fuente que se pueden copiar y probar muy fácilmente.

una FPGA's en particular tienen mucho potencial a nivel docente. Las prácticas de electrónica digital de cualquier universidad, que ahora se están realizando usando tecnología TTL, se podrían desarrollar como programas en un lenguaje de descripción hardware y luego probarlo en una placa entrenadora.

RECOMENDACIONES

Se recomienda la vinculación de la Universidad a Surlabs:"Laboratorios coordinados latinoamericanos de tecnología FPGA" los contactos son gustavo.sutter@uam.es y eduardo.boemo@uam.es de la Escuela Politécnica Superior Universidad Autónoma de Madrid. El objetivo central es la creación y coordinación de grupos de investigadores en las distintas universidades a fin de compartir líneas de investigación, métodos de diseño y material de laboratorio. Actualmente formado por más de 15 universidades y patrocinado por el banco Santander Central Hispano a través de proyectos UAM-SCH

BIBLIOGRAFÍA

Ashenden, Peter J. The VHDL Cookbook. Dept. Computer Science. University of Adelaide South Australia. Julio, 1990

C. Torres, R. Nieto, A. Bernal; .Diseño e Implementación de una Tarjeta PCI para Adquisición de Datos Basada en una FPGA. PCI SYSTEM ARCHITECTURE ,Fourth Editions,Ed: Addison Wesley Mayo de 1999.

Empresa Xilinx. Fabrica y comercializadora de FPGA's. http://www.xilinx.com/ ← Ojo con la referencia, revisar la norma.

FLOYD, Thomas L. Fundamentos de Sistemas Digitales,7 ed. España.Prentice.Hall.c2002 .ISBN 84-205-2994-X

Instructor: Bob Zeidman. An Introduction to FPGA Design. Class #448,468 Embedded Systems Conference 1999.

Pardo C Fernando, VHDL Lenguaje para descripción y modelado de circuitos 14 de octubre de 1997

Parnell, Karen. Mehta, Nick. Programmable Logic Design Quick Start Hand Book. 4ta Edición. Junio 2003.

PROAKIS, John. y MANOLAKIS, Dimitris. Tratamiento digital de señales. 3 ed. España.Prentice.Hall.c2001.ISBN 84-8322-000-8

Ronald J. Tocci (1993).Sistemas Digitales-Principios y Aplicaciones. 5ª Edición. Prentice Hall**.** México

Surin Kittitornkun and Charles R. Kime. ECE 554 – Digital Engineering Laboratory. FPGA Design Tutorial Version 3.1 - Fall 2001

Xilinx Corporation. Notas de aplicación: *DS003(v1.9), Xapp138(v2.3), Xapp151(v1.5. Xapp153(v1.0)* www.xilinx.com

Titulo: Fundamentals of Digital Logic with VHDL Design
Autores: Stephen Brown, Zvonko Vranesic
Ed. Mc Graw Hill