



UNIVERSIDAD CATOLICA DE SANTIAGO DE GUAYAQUIL

FACULTAD TECNICA PARA EL DESARROLLO

Tesis De Grado

Previo a la obtención del título de

INGENIERO EN TELECOMUNICACIONES

**“PROPUESTA PARA LA IMPLEMENTACION DE LABORATORIO PARA
CAPACITACIÓN EN DISEÑO DIGITAL BASADO EN TECNOLOGÍA FPGA
DE XILINX PARA LA FACULTAD DE EDUCACION TECNICA DE LA UCSG”**

REALIZADO POR:

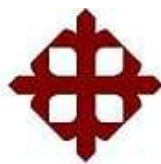
**ALVAREZ VÉLEZ ROXY VIVIANNE
LÓPEZ LLERENA BETTY MARICELA**

DIRECTOR DE TESIS:

ING. MARCOS ANDRADE

GUAYAQUIL – ECUADOR

2010



**FACULTAD TECNICA PARA EL DESARROLLO
TESIS DE GRADO**

TITULO:

**Propuesta Para La Implementación De Laboratorio Para Capacitación En Diseño
Digital Basado En Tecnología FPGA De Xilinx Para La FET De La UCSG**

**Presentada a la Facultad de Educación Técnica para el Desarrollo, Carrera de
Ingeniería en Telecomunicación de la Universidad de Guayaquil.**

POR:

Álvarez Vélez Roxy Vivianne

López Llerena Betty Maricela

**Para dar cumplimiento con uno de los requisitos para optar por el Título de:
Ingeniero en Telecomunicaciones
Mención en Gestión Empresarial**

.....
Decano

.....
Director de la Carrera

.....
Director de Tesis

.....
Vocal Principal

.....
Vocal Principal

.....
Secretario

DEDICATORIA

La presente tesis va dedicada para las personas de nuestro entorno que nos dieron las facultades para pensar en nuestro futuro y sobre todo a nuestras madres, fieles amigas, acompañantes y consejeras que si no fuera por sus sacrificios no estaríamos en estos momentos.

No tenemos palabras para seguir diciendo el gran regocijo que me da poder terminar esta carrera en donde profesores y compañeros dejan parte de su vida, para dar vida a las ilusiones de niño y que hoy en día se hacen realidad.

Solo sabemos que este camino es solo el comienzo de una gran historia de virtudes.

AGRADECIMIENTOS

En primera estancia a la fuente de nuestra vida y sabiduría, Dios, que nos ha permitido culminar con éxito un dichoso y afortunado proceso en este nuestro segundo hogar, nuestra tan querida Facultad Técnica para el Desarrollo.

A nuestros padres, pilares imprescindibles de nuestra formación, a los cuales dedicamos este triunfo, conscientes como el que más, de que sin ellos hubiera sido casi imposible alcanzar este logro, del cual todos nos sentimos agraciados al saber que ellos se sienten orgullosos de nosotros.

A todos aquellos que forman nuestra prestigiosa Universidad, cada uno de ustedes.... Maestros, que supieron ser guía y consejo en aquel momento de despejar nuestras mentes y llenarlas con la luz del conocimiento.

Un enfático agradecimiento al Ingeniero Marcos Andrade, nuestro director de tesis por la paciencia, apoyo, y la orientación brindada durante toda la elaboración del proyecto.

PRÓLOGO

Es indiscutible el progreso y evolución de la humanidad en todo aspecto y estamos inmersos en este proceso, es por eso que basamos nuestra investigación en el diseño digital basado en tecnología FPGA de Xilinx para la Facultad Técnica para el Desarrollo, tomando así una de las tantas alternativas que existen en el vasto universo de las telecomunicaciones.

El presente documento se ha desarrollado en 4 capítulos en los cuales nos hemos propuesto expresar de la manera más didáctica y explícita el argumento teórico que nos fue necesario para desarrollar este proyecto.

Esperamos que durante el desarrollo de esta tesis podamos lograr que nuestros compañeros desarrollen conocimientos y despejen dudas referentes al contenido en general, en el diseño digital basado en tecnología FPGA de Xilinx.

INDICE

DEDICATORIA	3
AGRADECIMIENTOS	4
PRÓLOGO	5
INDICE	6
INDICE DE FIGURAS	12
INTRODUCCION	18
CAPITULO I FPGAs	
1.1 Antecedentes	19
1.2 Planteamiento del problema	20
1.3 Justificación	22
1.4 Hipótesis	24
1.5 Objetivos	25
1.5.1 Objetivo General	25
1.5.2 Objetivos Específicos	25
1.6 Marco Teórico	26
1.7 Metodología	29
1.7.1 Fase I: Definición del Marco Teórico	29
1.7.2 Fase II: Análisis de la Situación Actual	29
1.7.3 Fase III: Análisis de Requerimiento	30
1.7.4 Fase IV: Diseño	31

CAPITULO II BENEFICIOS, ARQUITECTURA, APLICACIONES

2.1 Beneficios Principales de la Tecnología FPGA	33
2.1.1 Rendimiento	33
2.1.2 Tiempo en llegar al Mercado	33
2.1.3 Precio	34
2.1.4 Fiabilidad	34
2.1.5 Mantenimiento a Largo Plazo	35
2.1.6 Escoger un FPGA	35
2.2 Arquitectura de las FPGA de XILINX	37
2.2.1 Tecnología de Programación	37
2.2.2 Descripción de las principales familias	38
2.2.2.1 Arquitectura de Spartan-3E	40
2.2.3 Aplicaciones	42
2.3 Configuración de la Fpga	43

CAPITULO III PROGRAMACIÓN DE LA FPGA MEDIANTE UN SOFTWARE

3.1 Programando una FPGA con ISE 10.1 de Xilinx	44
3.2 Crear un Nuevo Proyecto	44
3.3 Categorías de Productos: All	44
3.4 Crear un nuevo Proyecto Source HDL	49

3.5 Creación del Test-Bench	54
3.6 Crear un Nuevo Proyecto	65
3.7 Crear una Fuente de HDL	68
3.8 Uso del Language Templates (VHDL)	69
3.9 Montaje de la Fuente VHDL	69
3.10 Simulación del Diseño	71
3.11 Simulación de la Funcionalidad del Diseño	74
3.12 Aplicación del Diseño	79
3.13 Asignación del Pin Ubicación Restricciones	80
3.14 Reimplementar diseño y Verificar Pin Localidades	81
3.15 Transferencia de Diseño a la Spartan 3 Demo Board	82

CAPITULO IV PRÁCTICAS DE LABORATORIO DE FPGAs EN XILINX

KIT DE ARRANQUE OPERACIÓN SPARTAN -3E

4.1 Herramienta de Flujo de Trabajo XILINX	87
4.1.1 Introducción	87
4.1.2 Objetivos	87
4.1.3 Procedimiento	87
4.1.4 Crear un Nuevo Proyecto	88
4.1.5 Añadir un diseño ya existente en el proyecto	92
4.1.6 Complete el diseño	93
4.1.7 Implementación del Diseño	100

4.1.8 Conclusión	103
ORIENTACIÓN XUP SPARTAN -3E	
4.2 Asistente de Arquitectura y Práctica PACE	105
4.2.1 Introducción	105
4.2.2 Objetivos	105
4.2.3 Procedimientos	105
4.2.4 Diseño General	106
4.2.5 Configurar un DCM	107
4.2.6 Una instancia de la DCM en un diseño Verilog	111
4.2.7 Una instancia de la DCM en un diseño VHDL	113
4.2.8 Asignación de Ubicación de Pins	114
4.2.9 Prueba de Diseño en Hardware	117
4.2.10 Funcionamiento de la UART-Real Time Clock	120
4.2.11 Conclusión	122
ORIENTACIÓN DE LA SPARTAN -3E STARTER KIT	
4.3 Calendario Global de Restricciones de Laboratorio	123
4.3.1 Introducción	124
4.3.2 Objetivos	124
4.3.3 Referencias	124
4.3.4 Diseño Descripción	125
4.3.5 Procedimiento	127
4.3.6 Prepare un programa de Plantilla	127

4.3.7 Introduzca el mundial de restricciones temporales	129
4.3.8 Introduzca el paso Localización Pin 3 Limitaciones	136
4.3.9 Implementar el diseño y Análisis de la Sincronización	138
4.3.10 Realice el paso de simulación HDL	139
4.3.11 Generar un Archivo de Programación	143
4.3.12 Probar el diseño de Hardware	146
4.3.13 Conclusión	147
4.3.14 Respuestas	148
ORIENTACIÓN DE LA SPARTAN -3E STARTER KIT	
4.4 La síntesis de Técnicas de Laboratorio	149
4.4.1 Introducción	150
4.4.2 Objetivos	150
4.4.3 Procedimiento	150
4.4.4 Complete el diseño	151
4.4.5 Sintetizar e Implementar uso de las opciones por defecto	152
4.4.6 Cambiar las opciones de Síntesis	154
4.4.7 Descarga y prueba de Sistema	159
4.4.8 Conclusión	161
4.4.9 Respuestas	161
ORIENTACIÓN DE LA SPARTAN -3E STARTER KIT	
4.5 CORE Sistema Generador de Laboratorio	164
4.5.1 Introducción	164

4.5.2	Objetivos	164
4.5.3	Procedimiento	164
4.5.4	Crear un Núcleo usando Core Generator	165
4.5.5	Una instancia de un núcleo de RAM Block en Fuente Verilog	168
4.5.6	Instanciación Bloque RAM Core en fuente VHDL	170
4.5.7	Realizar la simulación del Comportamiento	172
4.5.8	Pruebe la aplicación en Hardware	173
	ORIENTACIÓN XUP SPARTAN -3E	175
4.6	Chipscope Depuración de Laboratorio	177
4.6.1	Introducción	177
4.6.2	Objetivos	177
4.6.3	Procedimiento	177
4.6.4	Diseño General	
4.6.5	Crear un Chipscope nueva Pro	178
4.6.6	Configurar y Conectar un ILA CORE	180
4.6.7	Especifique Chipscope Analyser Opciones	187
4.6.8	Realizar un chip Verificacion	191
4.6.9	Conclusiones	193
	CONCLUSIONES	194
	PLAN DE TRABAJO	196
	BIBLIOGRAFIA	197

INDICE DE FIGURAS

Figura 1 Tarjeta FPGA de Xilinx	32
Figura 2 Tipos de conectores Utilizados	37
Figura 3 Familia de Fabricantes Xilinx	38
Figura 4 Arquitectura de CLB de la XC2000	39
Figura 5 Recursos de Interconexión familia XC2000	41
Figura 3.1 Como acceder a Project Navigator	44
Figura 3.2 Formato de Diseño	45
Figura 3.3 Nuevo Proyecto Wizard- De vice Propeties	46
Figura 3.4 Project Summary	47
Figura 3.5 Create New Project	47
Figura 3.6 Proyecto Iniciado	48
Figura 3.7 NEW Source Wizard	49
Figura 3.8 Ventana de New Source Wizard	49
Figura 3.9 New Source Define Module	50
Figura 3.10 Verificación de Terminales	50
Figura 3.11 New Source Wizard Summary	51
Figura 3.12 Estructurta VHDL	51
Figura 3.13Codigo VHDL	52
Figura 3.14 Nuevo componente creado	52

Figura 3.15 Synthesis	53
Figura 3.16 User Constrains para conexiones de los puertos	54
Figura 3.17 Asiganción de nombre	54
Figura 3.18 Intial Timing and Clock Wizrd	55
Figura 3.19 Initial timing and clock-Intialize Timing	55
Figura 3.20 Creado Test Bench	56
Figura 3.21 Modificación para combinaciones	56
Figura 3.22 Simulación Behavioral Model	57
Figura 3.23 Simulación de la Operación del Dispositivo	57
Figura 3.24 Pre Sintasis	58
Figura 3.25 User constrains File	58
Figura 3.26 Floor PLAN Nand3	59
Figura 3.27 Edición del Package Del Componente Descrito.	59
Figura 3.28 For view XC2000	60
Figura 3.29 Generate Program File	61
Figura 3.30 Crear archive para Programar	61
Figura 3.31 Ventana a Welccome	62
Figura 3.32 Assign New Configuration file	62
Figura 3.33Selección del arcviho	63
Figura 3.34 File name configuración	64
Figura 3.35 Programing Properties	64
Figura 3.36 Desarrollo de la FPGA	65

Figura 3.37 Select the device properties	67
Figura 3.38 Define Module	68
Figura 3.39 Simulación de Diseño	73
Figura 3.40 Counter_tbw	74
Figura 3.41 Behavioral Simulation	74
Figura 3.42 Smulación counter	75
Figura 3.43 Opciones de Proyect.	76
Figura 3.44 Clock Period	77
Figura 3.45 Xilinx Constraints Editor	78
Figura 3.46 Process design Summary	79
Figura 3.47 Pinout Report	81
Figura 3.48 Opciones Pounet	82
Figura 3.49 Welcome to impact	84
Figura 3.50 Boundary scan	85
Figura 4.1.1 Nuevo Proyecto de Wizard	89
Figura 4.1.2 Dispositivos y discos de Flujo de Dialogo	90
Figura 4.1.3Crear un nuevo origen de Dialogo	90
Figura 4.1.4 Cuadro de Fuentes Añadidas	91
Figura 4.1.5 Elija Tipo de Fuentes	92
Figura 4.1.6 Ensamblador de archivo de Pico Blaze	93
Figura 4.1.7 VHDL y Verilog ROM Definición de Archivos	94
Figura 4.1.8 Vista Jerárquica de Diseño de Pico Blaze	95

Figura 4.1.9 Vista Jerárquica Incluyendo el Test-Bench	96
Figura 4.1.10 ISTM Simulación de Propiedades	96
Figura 4.1.11 ISTM Resultado de la Simulación de Propiedades	97
Figura 4.1.12 Acceso a Señales Interna	98
Figura 4.1.13 Interruptor de Servicio de Rutina	99
Figura 4.1.14 Forma de Onda de Salida	100
Figura 4.1.16 Procedimiento en Ventana de Fuente	101
Figura 4.1.19 Design Summmary	102
Figura 4.2.1 Asistente de Arquitectura caja de Selección	108
Figura 4.2.2 Xilinx Clocking Wizard- General Window	109
Figura 4.2.3 Xilinx Clocking Wizard-Clock Buffers Window	110
Figura 4.2.4 Especificaciones de de la Salida de Frecuencia	110
Figura 4.2.5 Enumerados de Jerarquía del DCM	111
Figura 4.2.6 Introduzca ubicación de PIN Y Restricciones	112
Figura 4.2.7 Ventana de Dispositivo de Arquitectura	116
Figura 4.2.8 Ajustes para comunicaciones por puerto en serie	117
Figura 4.2.9 ASCII Settings for Serial Port Connection	118
Figura 4.2.10 Inicialice JTAG Chain	119
Figura 4.2.11 Cadena JTAG configuración asignada de archivos	119
Figura 4.2.12 Comunicación en Serie con Pico Blaze	120
Figura 4.2.14 Muestra de tiempo de Alarma y Estado	121
Figura 4.3.1 Pico Blaze Sistema	125

Figura 4.3.4 Cambiar la configuración no echo de nuevo a los Led	129
Figura 4.3.5 Procesos para las fuentes de la ventana	130
Figura 4.3.7 Dialogo periodo de reloj	132
Figura 4.3.8 Desplazamiento en Asistente- Reloj Borde de la Página	133
Figura 4.3.9 Desplazamiento en Asistente-Pagina de Datos	134
Figura 4.3.10 OFFSET limitaciones a cabo el dialogo	135
Figura 4.3.11 Limitaciones de tiempo	135
Figura 4.3.12 Abra el Archivo UCF	136
Figura 4.3.13 Entre las limitaciones en el Archivo de la UCF	137
Figura 4.3.15 Cambie la configuración de Eco de nuevo a los LEDS	142
Figura 4.3.16 Ver mensajes en la Consola de Simulación	142
Figura 4.3.17 Preparación de los Archivos RPM	144
Figura 4.3.18 Especifique la PROM xcF04S para la Spartan -3E	145
Figura 4.3.19 Bistream se asocia ahora con la RPM	145
Figura 4.3.20 Inicializar Boundary Scan Cadena	146
Figura 4.5.2 Seleccione el cuadro De Dialogo Tipo de Núcleo	167
Figura4.5.3 Bloquear el contenido de Inicialización de RAM	168
Figura 4.5.3 ^a Plantillas de Idiomas	169
Figura 4.5.4 Ver la Salida de la Ventana de Hyperterminal	174
Figura 4.6.3 chispeo pro	180
Figura 4.6.6 Parametros de activación	184
Figura 4.6.11 Puertos de salida	187

Figura 4.6.16 Instalacion de la unidad del partido	190
Figura 4.6.17 Condicion de ecuación	190
Figura 4.6.18 Ecuacion de Almacenamiento	191
Figura 4.6.21 Configuracion y activación	192

INTRODUCCION

La síntesis del proyecto se resume a la adquisición de Hardware y Software dedicado a la implementación de ésta tecnología mediante la descripción por software, de circuitos integrados particularizados con propiedad intelectual que a partir de la consecución de los objetivos serán de la titularidad de la Universidad Católica de Santiago de Guayaquil a través de la Facultad de Educación Técnica para el Desarrollo.

En el mercado mundial existen dos grandes y reconocidos fabricantes que se dedican al desarrollo de nuevas tecnologías los cuales diseñan y proveen de FPGAs, así como del software para su descripción y Kits de desarrollo aplicables sobre sus componentes en el mercado, estos fabricantes son Xilinx y Altera y como propósito inicial esta tesis dejará implementado un Kit de desarrollo para comprobar lo que mediante software se pudiere diseñar sea como clases regulares, cursos especiales y/o diseños aplicados a soluciones de mercado en tecnología Xilinx.

En lo concerniente al espacio físico éste proyecto puede iniciar al interior del laboratorio de electrónica de la FET donde pueda coexistir con la estructura informática a implementarse en el proceso de modernización al que nos vemos comprometidos; estudiantes, directivos y profesores.

CAPITULO I FPGAs

1.1 ANTECEDENTES

De acuerdo a lo propuesto en la malla curricular de las carreras de Ingeniería de la FET se dictan las materias de digitales hasta en un segundo nivel de complejidad (circuitos secuenciales) basados en tecnologías tradicionales y concluyendo esta capacitación con el conocimiento de microcontroladores y microprocesadores que permiten a los educandos desarrollar pequeñas aplicaciones; sin embargo, esto dista en mucho del requerimiento de la industria moderna en electrónica y comunicaciones quienes a la postre serán los beneficiarios del talento humano formado en las carreras de la Universidad Católica.

La tecnología al cabo de estos últimos 30 años no se ha detenido sino más bien vertiginosamente ha evolucionado en la creación de componentes cada vez más eficaces y complejos lo que hace de la tecnología con que se capacita en la FET una nueva amenaza frente a nuestra competencia en el mercado académico al ver disminuida la confiabilidad y velocidad de éstos componentes tradicionales con que actualmente se capacita.

Considerando los nuevos estándares que propone lo que hoy llamamos *tecnología verde*, los recursos económicos y académicos que se invierten en la capacitación para el uso de componentes ahora costosos y obsoletos y que además requieren un consumo elevado de energía, uso de espacio y peso inadecuados los nuevos equipos y sistemas de comunicaciones.

El advenimiento de la tecnología de *descripción de circuitos integrados de alta velocidad* con arquitectura abierta y que a su vez permite una altísima integración de lógica conocida en el mundo de la electrónica moderna como **VHDL** que es el acrónimo

que representa la combinación de **VHSIC** y **HDL** donde VHSIC es el acrónimo de *Very High Speed Integrated Circuit* que se refiere específicamente al componente y HDL es a su vez el acrónimo de Hardware Descripción Language que orienta a la existencia de uno o varios tipos de lenguajes de *programación* (descripción) de éstos nuevos circuitos integrados abre un campo nuevo en la capacitación para la implementación de soluciones integrales o parciales a través de la descripción de componentes electrónicos que permitan desarrollar aplicaciones basadas en el conocimiento técnico adquirido en las áreas de electrónica y comunicaciones.

Además; considerando, que la Universidad Católica dentro del plan de acreditación institucional ante el CONEA deberá optar en el camino por sustituir o modernizar parte de su malla curricular y/o implementar estrategias académicas que permitan a sus estudiantes adquirir conocimiento actualizado pero a su vez garantizando la idoneidad de éstos en el mercado profesional basados en las prácticas y desarrollos que pudieren hacer dentro de los laboratorios. Debemos rescatar que la tecnología actual converge siempre a la utilización de dispositivos de superficie montada (SMD) y como no podría ser de otra manera a la utilización de las aquí nombradas FPGA, de manera que es imprescindible que la FET implemente un laboratorio que esté a la altura de las circunstancias y le permita *entender* el comportamiento de éstos dispositivos para poder conocer y así mismo en el proceso permita aportar con soluciones confiables hasta la propia sustitución de áreas circuitales o tarjetas electrónicas completas mediante la utilización de los arreglos lógicos programables FPGA.

1.2 PLANTEAMIENTO DEL PROBLEMA

La implementación de laboratorio para capacitación en diseño digital basado en tecnología FPGA de xilinx para la Facultad de Educación Técnica de la Universidad Católica Santiago de Guayaquil, es una necesidad para los estudiantes y egresados de ingeniería en telecomunicaciones, pues implementar un laboratorio de diseño que permita investigar y desarrollar aplicaciones basándose en las nuevas tecnologías de diseño digital con la utilización de componentes electrónicos de última generación; específicamente orientado a la utilización de la tecnología FPGA (Field Programmable Gate Array) con lo cual se logrará insertar a la UCSG a través de la FET en el mundo de las Universidades modernas, proponiendo capacitación de calidad y basado en estándares mundiales actuales.

1.3 JUSTIFICACION

Nuestro trabajo de tesis consiste en introducir a los estudiantes de la carrera de Ingeniería en Telecomunicaciones a nuevas tecnologías, ya que nuestra facultad no cuenta con actualizaciones de tecnología, nosotras preocupadas por nuestro bienestar y el de todos nuestros compañeros de la Facultad de Educación Técnica para el desarrollo damos a conocer una herramienta de trabajo que en la actualidad existe en algunas universidades de nuestro país.

Desde que Xilinx los inventó en 1984, los FPGAs han pasado de ser sencillos chips de lógica de acoplamiento, a reemplazar a los circuitos integrados de aplicación específica (ASICs) y procesadores para procesamiento de señales y aplicaciones de control.

El silicio reprogramable tiene la misma capacidad de ajustarse que un software que se ejecuta en un sistema basado en procesadores, pero no está limitado por el número de núcleos de proceso disponibles. A diferencia de los procesadores, los FPGAs llevan a cabo diferentes operaciones de manera paralela, por lo que éstas no necesitan competir por los mismos recursos. Cada tarea de procesos independientes se asigna a una sección dedicada del chip, y puede ejecutarse de manera autónoma sin ser afectada por otros bloques de lógica. Como resultado, el rendimiento de una parte de la aplicación no se ve afectado cuando se agregan otros procesos.

La lógica programable puede reproducir desde funciones tan sencillas como las llevadas a cabo por una puerta lógica o un sistema combinacional hasta complejos sistemas en un chip

1.4 HIPÓTESIS

Para cumplir con los objetivos planteados utilizamos dos elementos que son recopilación de conceptos y teorías asimiladas durante toda nuestra carrera universitaria.

Utilizaremos el Webpack de Xilinx cuya licencia se obtiene gratuitamente desde la web de la marca; software que sin embargo no es completo pero alcanza para cumplir los objetivos trazados por esta tesis y las metas inmediatas de la facultad, las 20 computadoras que se encuentran en el laboratorio de electrónica tienen ya instalado el mismo.

Para lograr este propósito haremos uso del siguiente programa Xilinx ISE 10.1, el uso final se desarrollará sobre una tarjeta de entrenamiento Xilinx Spartan 3E, misma que es uno de los recursos que heredaremos a la Facultad Educación Técnica de la UCSG para el posterior desarrollo de prácticas en diseño digital.

1.5 OBJETIVOS

1.5.1 OBJETIVO GENERAL.

Presentar una propuesta para implementar un laboratorio de diseño que permita investigar y desarrollar aplicaciones basándose en las nuevas tecnologías de diseño digital con la utilización de componentes electrónicos de última generación. El objetivo principal de este proyecto es el de facilitar el intercambio de los elementos necesarios para el desarrollo con **FPGA**.

El intercambio de medios físicos (hardware) es complejo debido al costo de replicación y el intercambio de programas bajo licencias es un delito. Por estas razones es que el proyecto pone énfasis en software libre o gratuito y en *cores* que puedan ser redistribuidos sin restricciones.

1.5.2 OBJETIVO ESPECIFICO.

Los objetivos específicos del Laboratorio de Diseño Digital están orientados a la consecución de las siguientes estrategias:

1. Lograr desarrollar una línea de capacitación basada en tecnología FPGA que permita catapultar a la FET a la altura de las Universidades de Prestigio mundial en el sector de capacitación técnica.
2. Desarrollar e implementar aplicaciones basadas en FPGA como parte curricular y proponer a los estudiantes basados en el conocimiento adquirido propongan soluciones tecnológicas inspiradas en el diseño de sistemas electrónicos y de comunicaciones.
3. Fomentar la creación de una Academia de Desarrollo basado en FPGA que permita a los estudiantes interactuar con personal capacitado, proyectistas y estudiantes de las diferentes Universidades del Ecuador y el mundo que actualmente desarrollen o capaciten en tecnología FPGA, para lograr la solvencia académica y/o transferencia de tecnología

1.6 MARCO TEORICO

Las FPGA son el resultado de la convergencia de dos tecnologías diferentes, los dispositivos lógicos programables (PLDs [Programmable Logic Devices]) y los circuitos integrados de aplicación específica (ASIC [application-specific integrated circuit]). La historia de los PLDs comenzó con los primeros dispositivos PROM (Programmable Read-Only Memory) y se les añadió versatilidad con los PAL (Programmable Array Logic) que permitieron un mayor número de entradas y la inclusión de registros. Esos dispositivos han continuado creciendo en tamaño y potencia. Mientras, los ASIC siempre han sido potentes dispositivos, pero su uso ha requerido tradicionalmente una considerable inversión tanto de tiempo como de dinero. Intentos de reducir esta carga la modularización de los elementos de los circuitos, como en los ASIC basados en celdas, y de la estandarización de las máscaras, tal como Ferranti fue pionero con la ULA (Uncommitted Logic Array). El paso final era combinar las dos estrategias con un mecanismo de interconexión que pudiese programarse utilizando fusibles, antifusibles o celdas RAM, como los innovadores dispositivos Xilinx de mediados de los 80. Los circuitos resultantes son similares en capacidad y aplicaciones a los PLDs más grandes, aunque hay diferencias puntuales que delatan antepasados diferentes. Además de en computación reconfigurable, las FPGAs se utilizan en controladores, codificadores/decodificadores y en el prototipado de circuitos VLSI y microprocesadores a medida

El primer fabricante de estos dispositivos fue Xilinx y los dispositivos de Xilinx se mantienen como uno de los más populares en compañías y grupos de investigación. Otros vendedores en este mercado son Atmel, Altera, AMD y Motorola.

VHDL en la FPGAs en la división del diseño principal en módulos separados, la modularidad es uno de los conceptos principales de todo diseño. Normalmente se diferencia entre dos metodologías de diseño: top-down y botton-up. La metodología top-down consiste en que un diseño complejo se divide en diseños más sencillos que se puedan diseñar (o describir) más fácilmente. La metodología botton-up consiste en construir un diseño complejo a partir de módulos, ya diseñados, más simples. En la práctica, un diseño usa generalmente ambas metodologías.

Entrada de diseños, pueden usarse diversos métodos tal como **VHDL** como se vio anteriormente.

Simulación funcional, es decir, comprobaremos que lo escrito en el punto anterior realmente funciona como queremos, si no lo hace tendremos que modificarlo. En este tipo de simulación se comprueba que el código VHDL o Verilog (u otro tipo de lenguaje HDL) ejecuta correctamente lo que se pretende.

Síntesis. En este paso se adapta el diseño anterior (que sabemos que funciona) a un hardware en concreto, ya sea una FPGA o un ASIC. Hay sentencias del lenguaje que no

son sintetizables, como por ejemplo divisiones o exponenciaciones con números no constantes. El hecho de que no todas las expresiones en VHDL sean sintetizables es que el VHDL es un lenguaje genérico para modelado de sistemas (no sólo para diseño de circuitos digitales), por lo que hay expresiones que no pueden ser transformadas a circuitos digitales. Durante la síntesis se tiene en cuenta la estructura interna del dispositivo, y se definen restricciones, como la asignación de pines. El sintetizador optimiza las expresiones lógicas con objeto de que ocupen menor área, o bien son eliminadas las expresiones lógicas que no son usadas por el circuito.

Simulación post-síntesis. En este tipo de simulación se comprueba que el sintetizador ha realizado correctamente la síntesis del circuito, al transformar el código HDL en bloques lógicos conectados entre sí. Este paso es necesario ya que, a veces, los sintetizadores producen resultados de síntesis incorrectos, o bien realiza simplificaciones del circuito al optimizarlo.

Placement y routing. El proceso de placement consiste en situar los bloques digitales obtenidos en la síntesis de forma óptima, de forma que aquellos bloques que se encuentran muy interconectados entre si se sitúen próximos entre si. El proceso de routing consiste en rutar adecuadamente los bloques entre si, intentando minimizar retardos de propagación para maximizar la frecuencia máxima de funcionamiento del dispositivo.

Back-annotation. Una vez ha sido completado el placement & routing, se extraen los retardos de los bloques y sus interconexiones, con objeto de poder realizar una simulación temporal (también llamada simulación post-layout). Estos retardos son anotados en un fichero SDF (Standart Delay Format) que asocia a cada bloque o interconexión un retardo mínimo/típico/máximo.

Simulación temporal. A pesar de la simulación anterior puede que el diseño no funcione cuando se programa, una de las causas puede ser por los retardos internos del chip. Con esta simulación se puede comprobar, y si hay errores se tiene que volver a uno de los anteriores pasos.

1.7 METODOLOGIA

La metodología para el desarrollo del proyecto consta de cuatro fases: definición de marco teórico, análisis de la situación actual, análisis de requerimientos y diseño asistido por computador.

1.7.1 FASE I: DEFENICION DEL MARCO TEORICO

En la definición del marco teórico de nuestra tesis vamos a resaltar varios conceptos específicos como lo de FPGAs, VHDL, XILINXS, SPARTAN 3. ISE DE XILINXS, son los elementos principales de este proyecto

1.7.2 FASE II: ANALISIS DE LA SITUACION ACTUAL

En esta fase se efectuará el análisis de los requerimientos para la implementación de laboratorio para capacitación en diseño digital basado en tecnología FPGAs de

XILINXS para la FET de la UCSG. La tecnología al cabo de estos últimos 30 años no se ha detenido sino más bien vertiginosamente ha evolucionado en la creación de componentes cada vez más eficaces y complejos lo que hace de la tecnología con que se capacita en la FET una nueva amenaza frente a nuestra competencia en el mercado académico al ver disminuida la confiabilidad y velocidad de éstos componentes tradicionales con que actualmente se capacita. Considerando los nuevos estándares que propone lo que hoy llamamos *tecnología verde*, los recursos económicos y académicos que se invierten en la capacitación para el uso de componentes ahora costosos y obsoletos y que además requieren un consumo elevado de energía, uso de espacio y peso inadecuados los nuevos equipos y sistemas de comunicaciones

1.7.3 FASE 3: ANALISIS DE REQUERIMIENTO

El proyecto **FPGA** nació con la idea de poder compartir en la Facultad Técnica para el Desarrollo la capacidad de aprender observando el código fuente de las aplicaciones, la posibilidad de adaptar a gusto según las necesidades particulares de cada interesado dando oportunidad de mejorar el código y brindar esas mejoras al resto de la comunidad impulsando el desarrollo con dispositivos **FPGA** utilizando herramientas de software libre u open source. Fomentar el intercambio y desarrollo de cores IP con licencias que posean el mismo espíritu que las del software libre.

1.7.4 FASE 4: DISEÑO

Una jerarquía de interconexiones programables permite a los bloques lógicos de un FPGA ser interconectados según la necesidad del diseñador del sistema, algo parecido a un protoboard *tarjeta de prototipados (protoboard)* programable. Estos bloques lógicos e interconexiones pueden ser programados después del proceso de manufactura por el usuario/diseñador, así que el FPGA puede desempeñar cualquier función lógica necesaria.

Una tendencia reciente ha sido combinar los bloques lógicos e interconexiones de los FPGA con microprocesadores y periféricos relacionados para formar un «Sistema programable en un chip». Ejemplo de tales tecnologías híbridas pueden ser encontradas en los dispositivos Virtex-II PRO y Virtex-4 de Xilinx, los cuales incluyen uno o más procesadores PowerPC embebidos junto con la lógica del FPGA. El FPSLIC de Atmel es otro dispositivo similar, el cual usa un procesador AVR en combinación con la arquitectura lógica programable de Atmel. Otra alternativa es hacer uso de núcleos de procesadores implementados haciendo uso de la lógica del FPGA. Esos núcleos incluyen los procesadores MicroBlaze y PicoBlaze de Xilinx, Nios y Nios II de Altera, y los procesadores de código abierto LatticeMicro32 y LatticeMicro8.

Muchos FPGA modernos soportan la reconfiguración parcial del sistema, permitiendo que una parte del diseño sea reprogramada, mientras las demás partes siguen funcionando. Este es el principio de la idea de la «computación reconfigurable», o los «sistemas reconfigurables».



Figura1. Tarjeta FPGA de Xilinx

CAPITULO II BENEFICIOS, ARQUITECTURA, APLICACIONES.

2.1 BENEFICIOS PRINCIPALES DE LA TECNOLOGÍA FPGA

2.1.1 RENDIMIENTO

Los FPGAs tomando ventaja del paralelismo del hardware, exceden la potencia de cómputo de los procesadores digitales de señales (DSPs) rompiendo el paradigma de ejecución secuencial y logrando más en cada ciclo de reloj. BDTI, una destacada firma analista que realiza evaluaciones de referencia, lanzó evaluaciones mostrando cómo los FPGAs pueden entregar significativamente más potencia de procesamiento por dólar que una solución de DSP, en algunas aplicaciones.

El controlar entradas y salidas (E/S) a nivel de hardware ofrece tiempos de respuesta más veloces y funcionalidad especializada que coincide con los requerimientos de una aplicación.

2.1.2 TIEMPO EN LLEGAR AL MERCADO

La tecnología FPGA ofrece flexibilidad y capacidades de rápido desarrollo de prototipos, para enfrentar las preocupaciones de tiempo incrementado en que un producto tarde en llegar al mercado. Usted puede probar una idea o un concepto y verificarlo en hardware sin tener que pasar por el largo proceso de fabricación por el que pasa un diseño personalizado de ASIC.

Posteriormente podrá implementar cambios y realizar iteraciones de un diseño FPGA en cuestión de horas en vez de semanas. También está disponible el hardware comercial listo para ejecutarse (COTS), con diferentes tipos de E/S ya conectados a un chip FPGA programable por el usuario. El aumento en disponibilidad de herramientas de software

de alto nivel disminuye la curva de aprendizaje con niveles de abstracción. Estas herramientas frecuentemente incluyen importantes núcleos IP (funciones pre construidas) para control avanzado y procesamiento de señales.

2.1.3 PRECIO

El precio de la ingeniería no recurrente de un diseño personalizado ASIC excede considerablemente al de las soluciones de hardware basadas en FPGA. La fuerte inversión inicial de los ASICs es fácilmente justificable para los fabricantes de equipos originales que embarcan miles de chips por año, pero muchos usuarios finales necesitan la funcionalidad de un hardware personalizado para decenas o cientos de sistemas en desarrollo. La misma naturaleza programable del silicio implica que no hay precio de fabricación o largo plazos de ejecución de ensamblado. Los requerimientos de un sistema van cambiando con el tiempo, y el precio de cambiar incrementalmente los diseños FPGA es insignificante al compararlo con el precio de implementar cambios en un ASIC antes de su lanzamiento.

2.1.4 FIABILIDAD

Mientras que las herramientas de software ofrecen un entorno de programación, los circuitos de un FPGA son una implementación segura de la ejecución de un programa. Los sistemas basados en procesadores frecuentemente implican varios niveles de abstracción para auxiliar a programar las tareas y compartir los recursos entre procesos múltiples. El nivel controlador se encarga de los recursos de hardware y el sistema operativo administra la memoria y el ancho de banda del procesador. El núcleo de un procesador sólo puede ejecutar una instrucción a la vez, y los sistemas basados en

procesadores están siempre en riesgo de que sus tareas se obstruyan entre sí. Los FPGAs, que no necesitan sistemas operativos, minimizan los retos de fiabilidad con ejecución paralela y hardware preciso dedicado a cada tarea.

2.1.5 MANTENIMIENTO A LARGO PLAZO

Como se mencionó anteriormente, los chips FPGA son actualizables en campo y no requieren el tiempo y el precio que implica rediseñar un ASIC. Los protocolos de comunicación digital por ejemplo, tienen especificaciones que podrían cambiar con el tiempo, y las interfaces basadas en ASICs podrían causar retos de mantenimiento y habilidad de actualización. Los chips FPGA, al ser reconfigurables, son capaces de mantenerse al tanto con modificaciones a futuro que pudieran ser necesarias. Mientras el producto o sistema se va desarrollando, usted puede implementarle mejoras funcionales sin la necesidad de invertir tiempo rediseñando el hardware o modificando el diseño de la tarjeta.

2.1.6 ESCOGER UN FPGA

Al examinar las especificaciones de un chip FPGA, observe que generalmente están divididos en bloques de lógica configurables como segmentos o células de lógica, funciones fijas de lógica como multiplicadores, y recursos de memoria como RAM en bloque embebida. El chip FPGA tiene otros componentes, pero éstos son generalmente los más importantes cuando se seleccionan y comparan FPGAs para una aplicación en particular.

	Virtex- II 1000	Virtex- II 3000	Spartan- 3 1000	Spartan- 3 2000	Virtex- 5 LX30	Virtex- 5 LX50	Virtex- 5 LX85	Virtex- 5 LX110
Compuertas	1 millón	3 millones	1 millón	2 millones	-----	-----	-----	-----
Flip-Flops	10,240	28,672	15,360	40,960	19,200	28,800	51,840	69,120
Tablas LUT	10,240	28,672	15,360	40,960	19,200	28,800	51,840	69,120
Multiplicadores	40	96	24	40	32	48	48	64
RAM en Bloque (kb)	720	1,728	432	720	1,152	1,728	3,456	4,608

Tabla. Especificaciones de Recursos FPGA de Varias Familias

La tabla muestra especificaciones de recursos utilizados para comparar chips FPGA dentro de varias familias de Xilinx. El número de compuertas ha sido una forma típica de comparar el tamaño de los chips FPGA contra la tecnología ASIC, pero no describe realmente el número de componentes individuales dentro de un FPGA. Esta es una de las razones por las cuales Xilinx no especificó el número de compuertas de sistema equivalentes en la nueva familia Virtex

2.2 ARQUITECTURA DE LAS FPGA DE XILINX

2.2.1 TECNOLOGIA DE PROGRAMACION

Antes de continuar con conocimientos mas avanzados acerca de FPGAs (de XILINX en concreto), hay que aclarar como se realiza el proceso de programación (ie., las conexiones necesarias entre bloques y pistas). En primer lugar, si se piensa que el número de dispositivos de conexión que hay en una FPGA es muy grande (típicamente superior a 100.000), es necesario que cumplan las siguientes propiedades:

- ✓ Ser lo mas pequeños posible.
- ✓ Tener la resistencia ON lo mas baja posible, mientras la OFF ha de ser lo mas alta posible (para que funcione como conmutador).
- ✓ Se deben poder incorporar al proceso de fabricación de la FPGA.

El proceso de programación no es único, se puede realizar mediante diferente “Tecnologías”, como son células RAM estáticas, transistores EPROM y EEPROM, etc. En el caso de las FPGAs de XILINX los elementos de programación se basan en células de memoria RAM que controlan transistores de paso, puertas de transmisión o multiplexores. En la figura se puede ver esquemáticamente como son. Dependiendo del

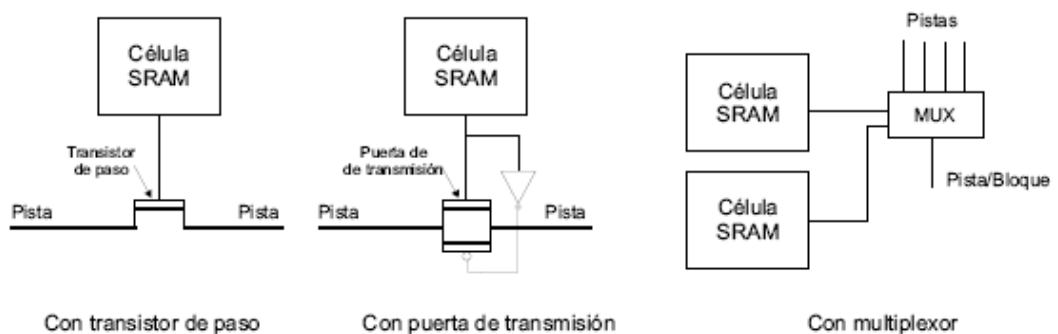


Figura 2. Tipos de conectores utilizados por XILINX

Es importante destacar que si se utilizan células SRAM la configuración de la FPGA será válida únicamente mientras esté conectada la alimentación, pues es memoria volátil. En los sistemas finales está claro que hace falta algún mecanismo de almacenamiento no volátil que cargue las células RAM. Esto se puede conseguir mediante EPROMs o disco.

Este elemento de programación es relativamente grande (necesita por lo menos 5 transistores), pero se puede implementar en el proceso normal de fabricación del circuito (CMOS). Además, permite reconfigurar la FPGA de una forma muy rápida.

2.2.2. DESCRIPCIÓN DE LAS PRINCIPALES FAMILIAS

Hay múltiples familias lógicas dentro de XILINX. Las primeras que surgieron son: XC2000 (descatalogada en el año 1999), XC3000 y XC4000, correspondientes respectivamente a la primera, segunda y tercera generación de dispositivos, que se distinguen por el tipo de bloque lógico configurable (CLB) que contienen. En la actualidad existen también las familias de FPGA Spartan II, Spartan III, Spartan VI, Virtex, Virtex VI y Virtex Pro. La figura 3 muestra la cantidad de CLBs que puede haber en cada FPGA de las familias base y ese mismo valor expresado en puertas equivalentes

SERIE	Tipo CLB	Nº de CLBs	Puertas Equivalentes
XC2000	1 LUT, 1 FF	64-100	1.200-1.800
XC3000	1 LUT, 2 FF	64-484	1.500-7.500
XC4000XL	3 LUT, 2 FF	64-3.136	1.600-180.000

Figura 3. Familias del fabricante XILINX

El bloque lógico ha de ser capaz de proporcionar una función lógica en general y reprogramable. La mejor forma de realizar esto es mediante una tabla de valores "preasignados" o "tablas de look-up". Básicamente, una tabla look-up (LUTs en adelante) es una memoria, con un circuito de control que se encarga de cargar los datos. Cuando se aplica en una dirección las entradas de la función booleana la memoria devuelve un dato, lo que se puede hacer corresponder con la salida requerida. Falta añadir los componentes necesarios para desempeñar funciones no implementables con una memoria, tales como una batería de registros, multiplexores, buffers etc. Estos componentes están en posiciones fijas del dispositivo.

El inconveniente es obvio: ocupan mucho espacio y no son muy aprovechables.

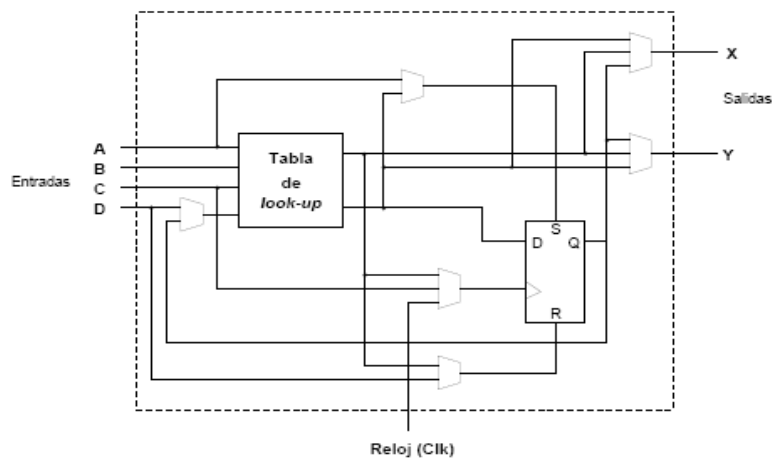


Figura 4. Arquitectura del CLB de la XC 2000

Los bloques lógicos configurables de la familia XC2000 se componen de una look up table con cuatro entradas y un biestable, con lo que puede generar cualquier función de

hasta 4 variables o dos funciones de 3 variables. El de la familia XC3000 es más complejo: permite implementar una función de 5 variables o dos funciones de 4 variables (limitadas a 5 diferentes entradas, claro). Además contiene dos biestables y cierta lógica. La familia XC4000 es ya mucho más sofisticada.

En general, los recursos de interconexión son de tres tipos:

- ✓ Conexiones directas, permiten la conexión de las salidas del CLB con sus vecinos más directos (N, S, E y O).
- ✓ Interconexiones de propósito general, para distancias superiores a un CLB (más allá de vecino). Son pistas horizontales y verticales del tamaño de un CLB, pero que se pueden empalmar para crear pistas más largas.
- ✓ Líneas de largo recorrido, suelen cubrir lo ancho o largo de la pastilla. Permiten conexiones con un retardo mucho menor que uniendo las anteriores.

El camino crítico de un circuito es el recorrido que, desde una entrada hasta una salida, presenta un retardo máximo.

2.2.2.1 ARQUITECTURA DE SPARTAN 3 E

Aunque hoy en día no se encuentran disponibles las FPGAs de esta familia, dado que contienen la arquitectura más sencilla, vamos a utilizarlas como base para comprender el funcionamiento de este tipo de dispositivos.

En la figura se puede ver como es el bloque configurable básico de las XC2000. Contiene como elementos principales una tabla de look-up de 4 entradas y un biestable D. La tabla de look-up puede reproducir cualquier función de cuatro variables o dos funciones de tres variables.

De las dos salidas del CLB una se puede registrar, o se pueden dejar las dos combinacionales.

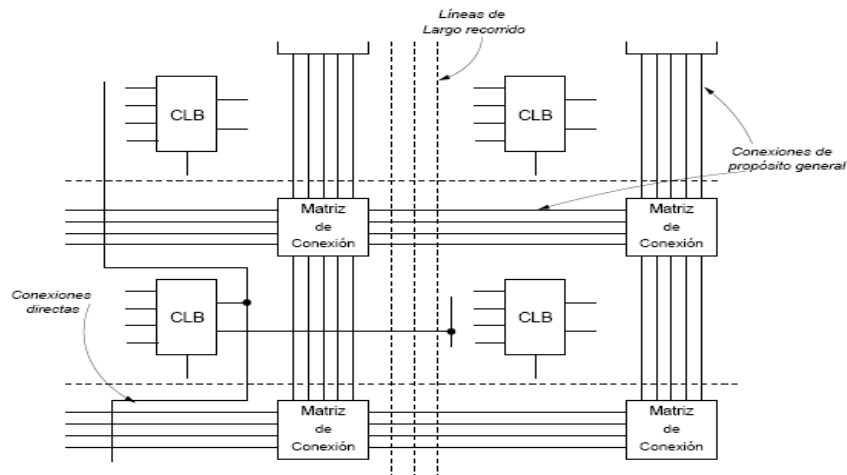


Figura 5. Recursos de Interconexión en la familia XC 2000

Adicionalmente, en el bloque hay 6 multiplexores que permitirían seleccionar las conexiones que se desea hacer dentro de cada CLB particular. Por ello, en sus terminales de selección necesitarían un elemento de memoria con el valor deseado. Nótese que la salida del biestable se puede llevar de vuelta a una de las entradas de la LUT, siempre y cuando se configuren adecuadamente los selectores oportunos. Esto es muy útil, pues permite implementar estructuras realimentadas como son contadores o máquinas de estados.

Por otro lado, la arquitectura de rutado de la familia XC2000 utiliza tres tipos de recursos de interconexión: conexiones directas, conexiones de propósito general y líneas de largo recorrido.

Todos estos recursos se pueden ver en la figura 5. Las conexiones directas (en la figura 5 aparecen solo para un CLB) proporcionan enlace desde la salida de un CLB hasta sus vecinos superior, inferior y a la derecha. Si hay que conectar una red a un bloque mas lejano hay que utilizar las conexiones de propósito general, que son segmentos de pista

dispuestas horizontal y verticalmente a lo largo de toda la FPGA. En particular, en esta familia hay cuatro segmentos horizontales y cinco verticales por canal. Su longitud esta limitada siempre a la distancia fija entre 2 CLBs, por lo que para realizar conexiones mas largas hay que utilizar las matrices de interconexión.

Es importante observar que la utilización de estos recursos repercutirá negativamente en las prestaciones del diseño, pues los conectores de la matriz introducen forzosamente un retardo.

Las líneas de largo recorrido se utilizan para conexiones que han de llegar a varios CLBs con bajo skew.

2.2.3. APLICACIONES

Cualquier circuito de aplicación específica puede ser implementado en un FPGA, siempre y cuando esta disponga de los recursos necesarios. Las aplicaciones donde más comúnmente se utilizan los FPGA incluyen a los DSP (procesamiento digital de señales), radio definido por software, sistemas aeroespaciales y de defensa, prototipos de ASICs, sistemas de imágenes para medicina, sistemas de visión para computadoras, reconocimiento de voz, bioinformática, emulación de hardware de computadora, entre otras. Cabe notar que su uso en otras áreas es cada vez mayor, sobre todo en aquellas aplicaciones que requieren un alto grado de paralelismo.

Existe código fuente disponible (bajo licencia GNU GPL) de sistemas como microprocesadores, microcontroladores, filtros, módulos de comunicaciones y memorias, entre otros. Estos códigos se llaman cores.

2.3 CONFIGURACIÓN DE LA FPGA

Finalmente, en la etapa de generación, otra herramienta genera un archivo de configuración, el que es descargado a la memoria de la FPGA y que contiene la trama de bits que produce la configuración adecuada.

CAPITULO III. PROGRAMACION DE LA FPGA MEDIANTE UN SOFTWARE

3.1 PROGRAMANDO UNA FPGA CON ISE 10.1 DE XILINX

Para iniciar el programa de desarrollo, haga doble click en el ícono ISE de la versión de software que tenga instalada en su PC

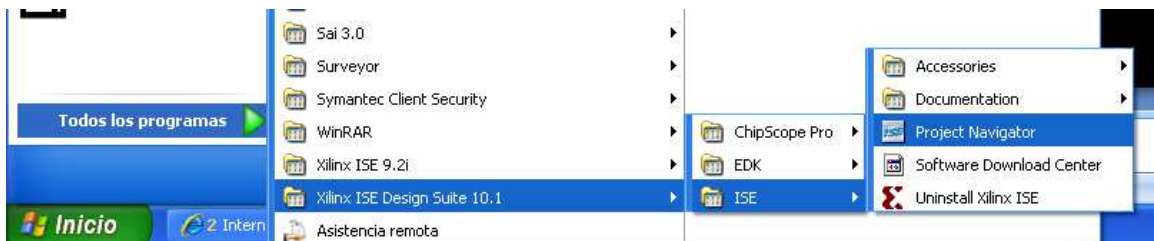


Figura 3.1 como acceder al Project Navigator

3.2 CREAR UN NUEVO PROYECTO

Crear un nuevo proyecto ISE que tendrá como objetivo el dispositivo FPGA del Kit Spartan-3 de la tarjeta demo:

1. Seleccione File > New Project. Project Wizard aparecerá.
2. Escriba N3_NAND3 en el campo Project Name.
3. Ingrese o busque una ruta (directory path) para el nuevo proyecto. Un a carpeta NAND3 es creado automáticamente.
4. Compruebe que HDL es seleccionado en Top-Level Source Type.
5. Click Next para ir a la página de propiedades del dispositivo.
6. Llenar las en c/u de los campos de propiedades del dispositivo como indica el listado siguiente:

3.3 CATEGORIAS DE PRODUCTOS: ALL

- ✓ Family: Spartan3A and Spartan 3AN
- ✓ Device: XC3S700AN
- ✓ Package: FGG484

- ✓ Speed Grade: -4
- ✓ Top-Level Source Type: HDL
- ✓ Synthesis Tool: XST (VHDL/Verilog)
- ✓ Simulator: ISE Simulator (VHDL/Verilog)
- ✓ Preferred Language: VHDL
- ✓ Verifique que Enable Enhanced Design Summary este seleccionado

Dejar los valores por default en los campos restantes.

7. Click Next para proceder a crear la ventana New Source en el New Project Wizard. Al finalizar esta sección su proyecto debe estar iniciado y deberá ser completado con el código correspondiente

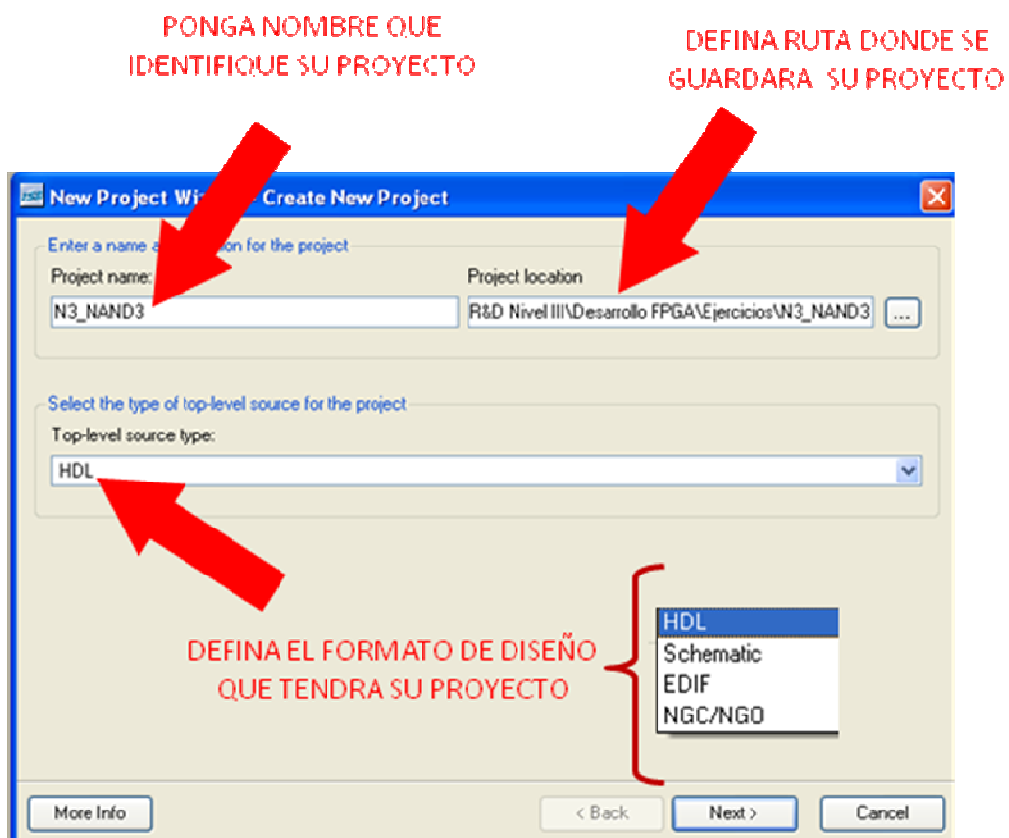


Figura 3.2 Formato de Diseño

Crear un nuevo proyecto ISE que tendrá como objetivo el dispositivo FPGA del Kit Spartan-3 de la tarjeta demo:

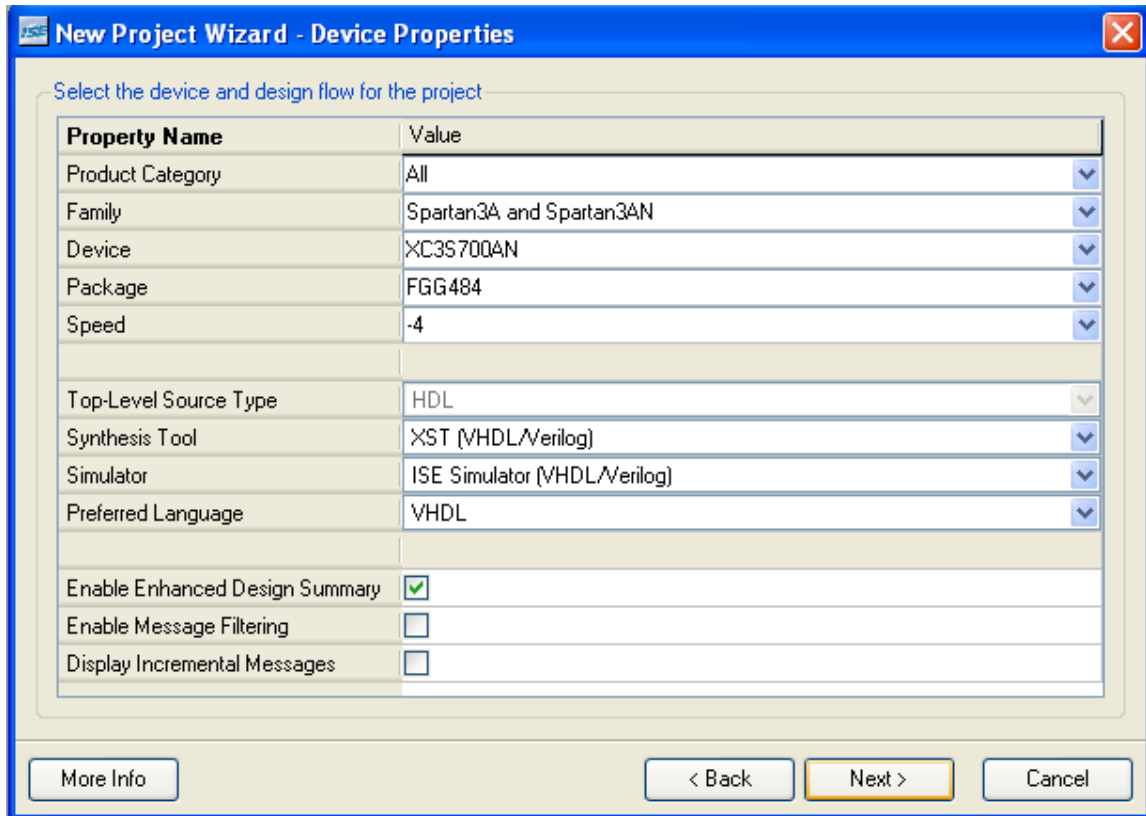
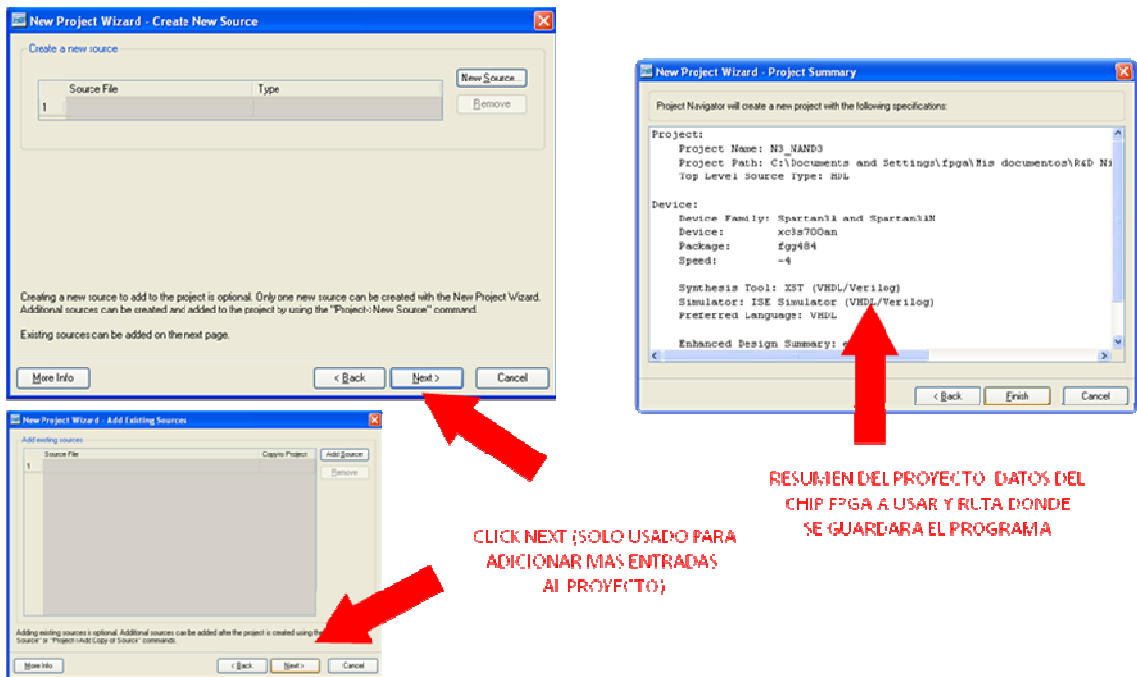


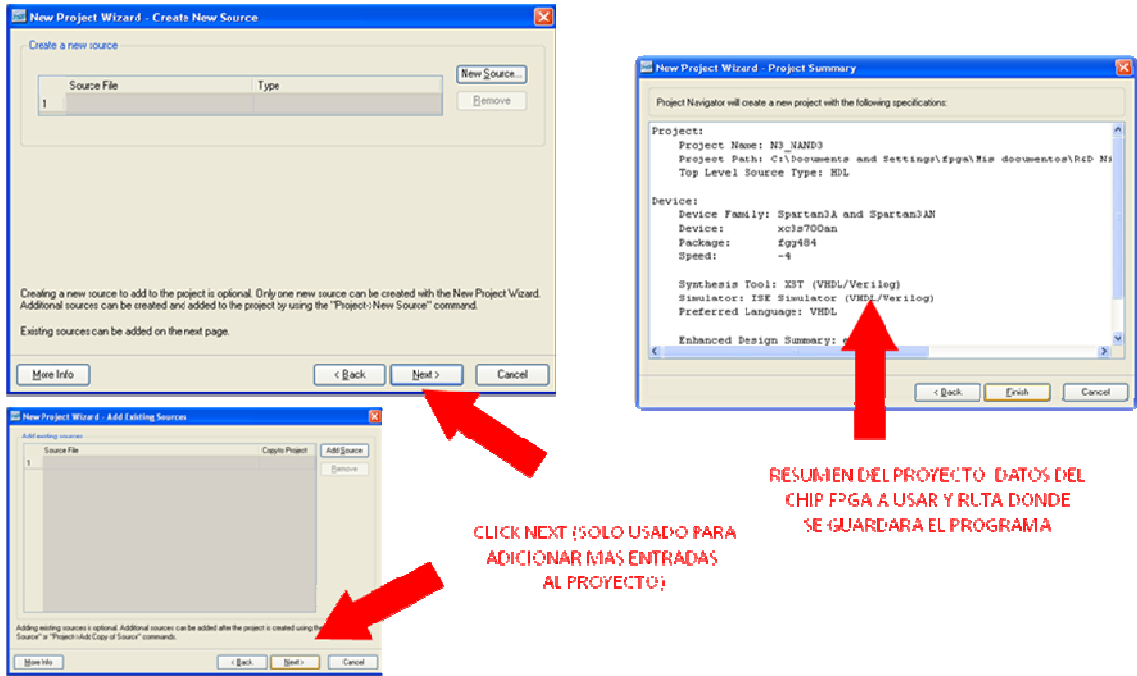
Figura 3.3 Nuevo Proyecto Wizard- De vice Propeties



CLICK NEXT (SOLO USADO PARA ADICIONAR MAS ENTRADAS AL PROYECTO)

RESUMEN DEL PROYECTO DATOS DEL CHIP FPGA A USAR Y RUTA DONDE SE GUARDARA EL PROGRAMA

Figura 3.4 Project Summary



CLICK NEXT (SOLO USADO PARA ADICIONAR MAS ENTRADAS AL PROYECTO)

RESUMEN DEL PROYECTO DATOS DEL CHIP FPGA A USAR Y RUTA DONDE SE GUARDARA EL PROGRAMA

Figura 3.5 Create New Project

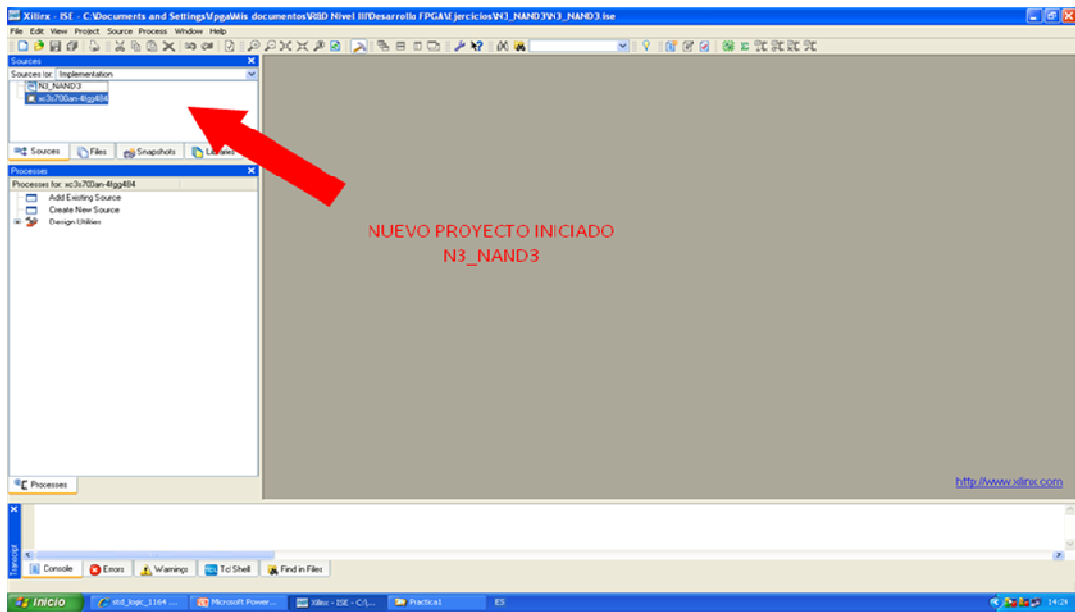


Figura 3.6 Proyecto Iniciado

3.4 CREAR UN NUEVO SOURCE HDL

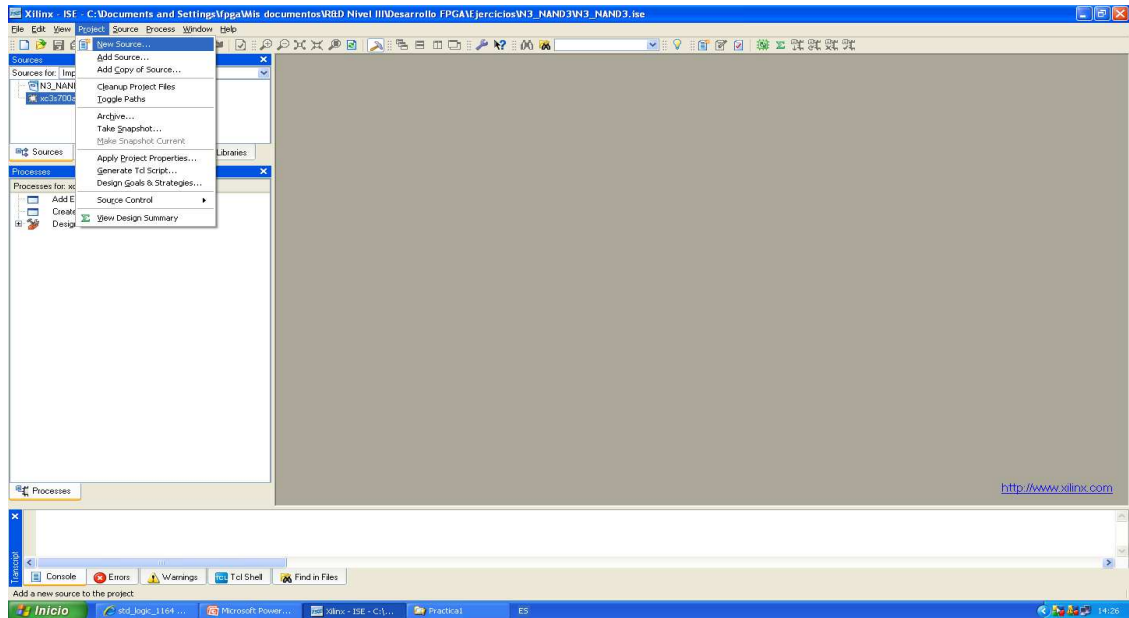


Figura 3.7 New Source wizard

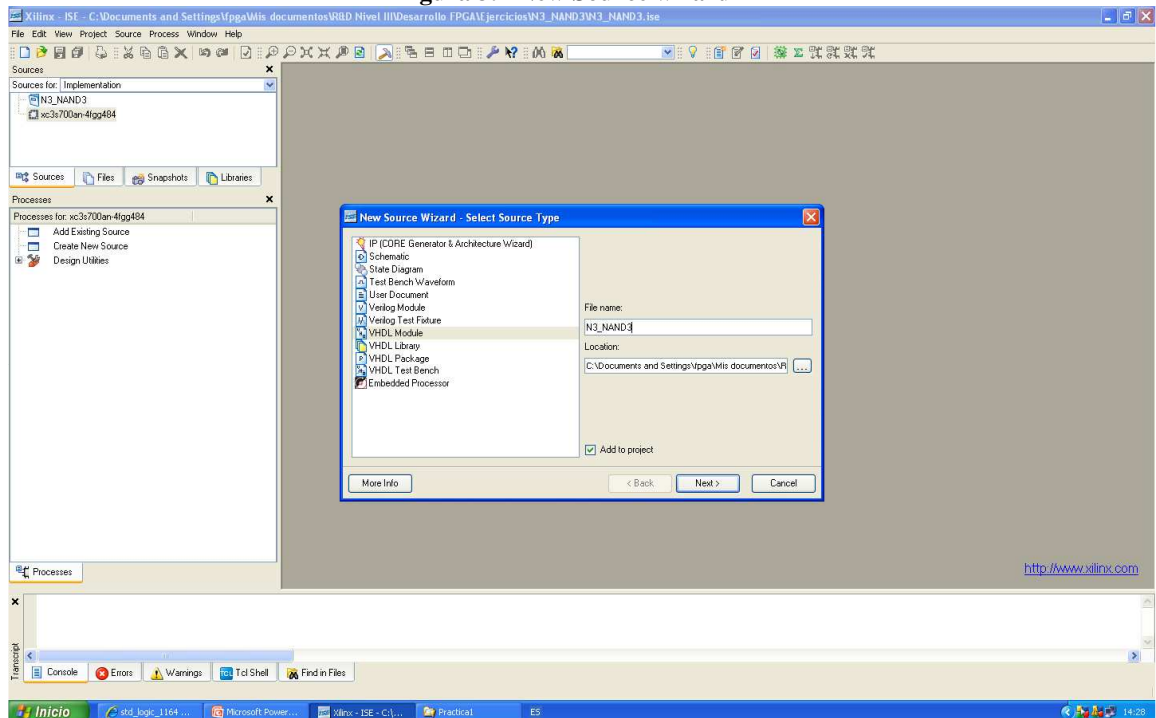
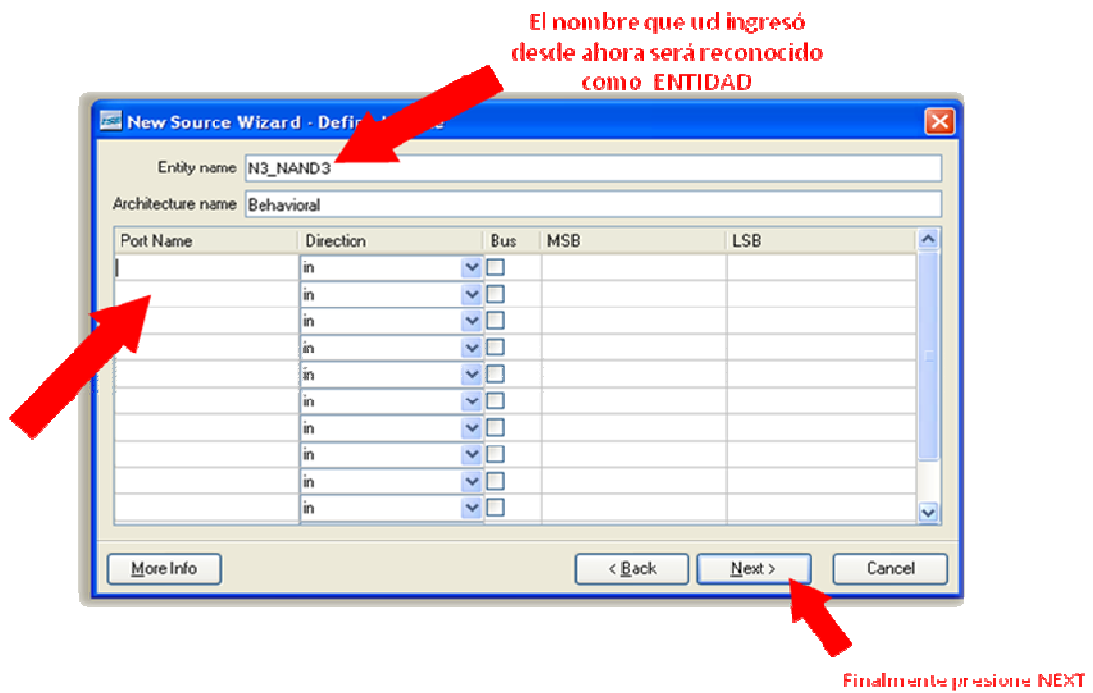


Figura 3.8 Ventana de New Wizard



3.9 New Source define Module

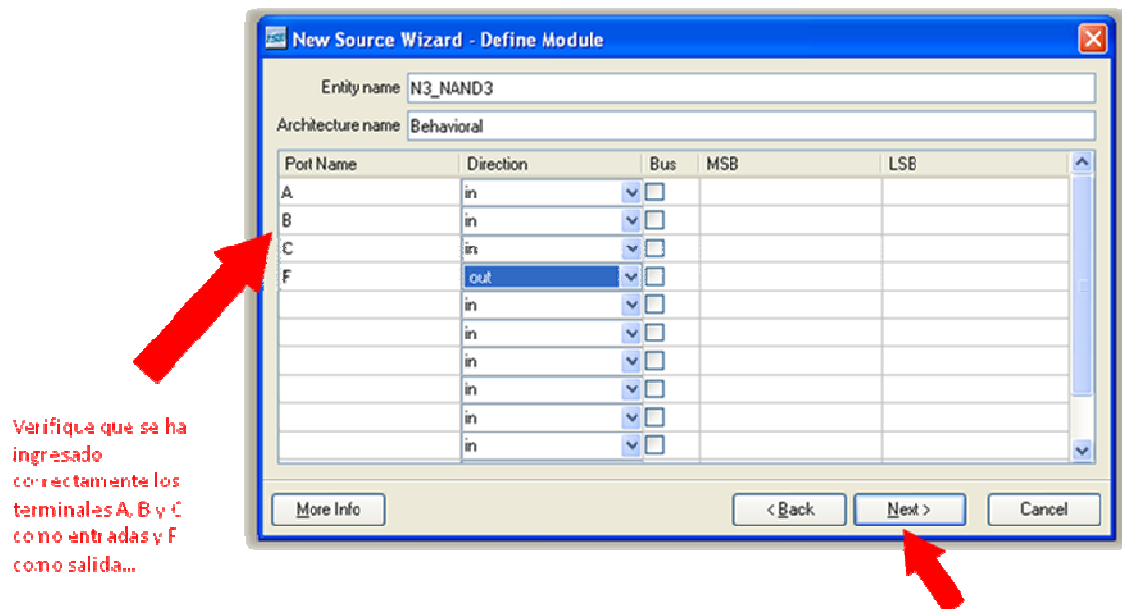
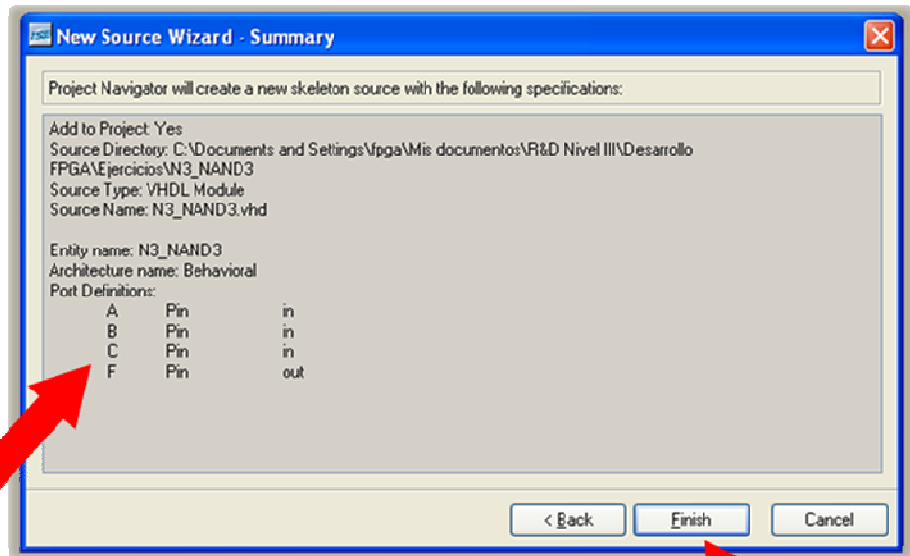
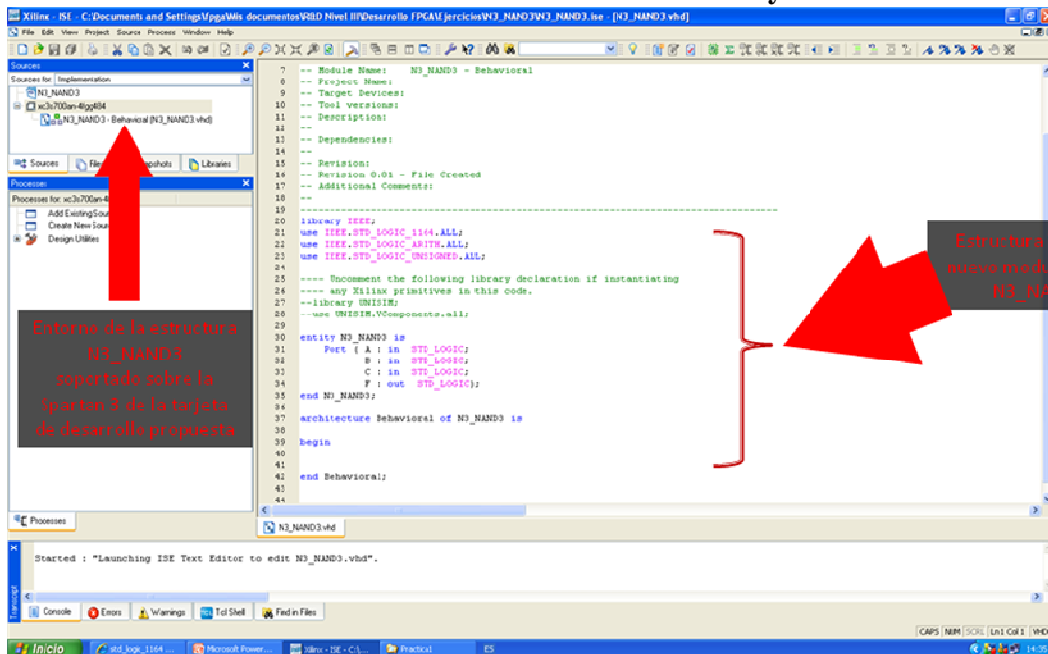


Figura 3.10 Verificación de terminales

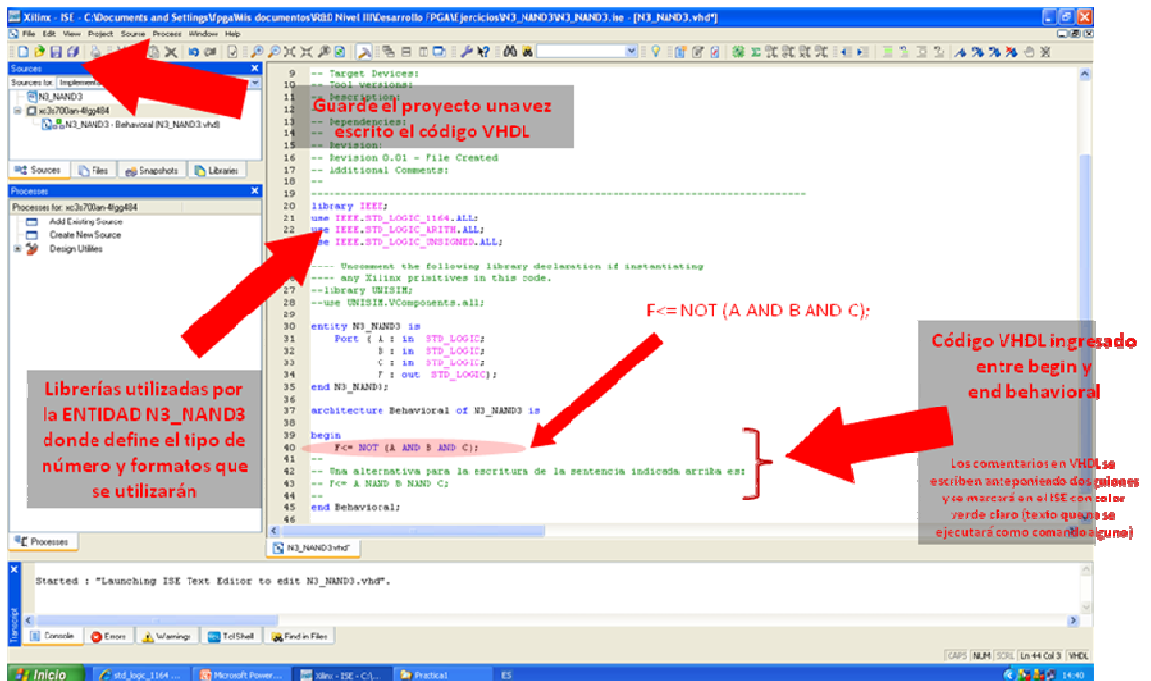


Sumario de la nueva entidad creada

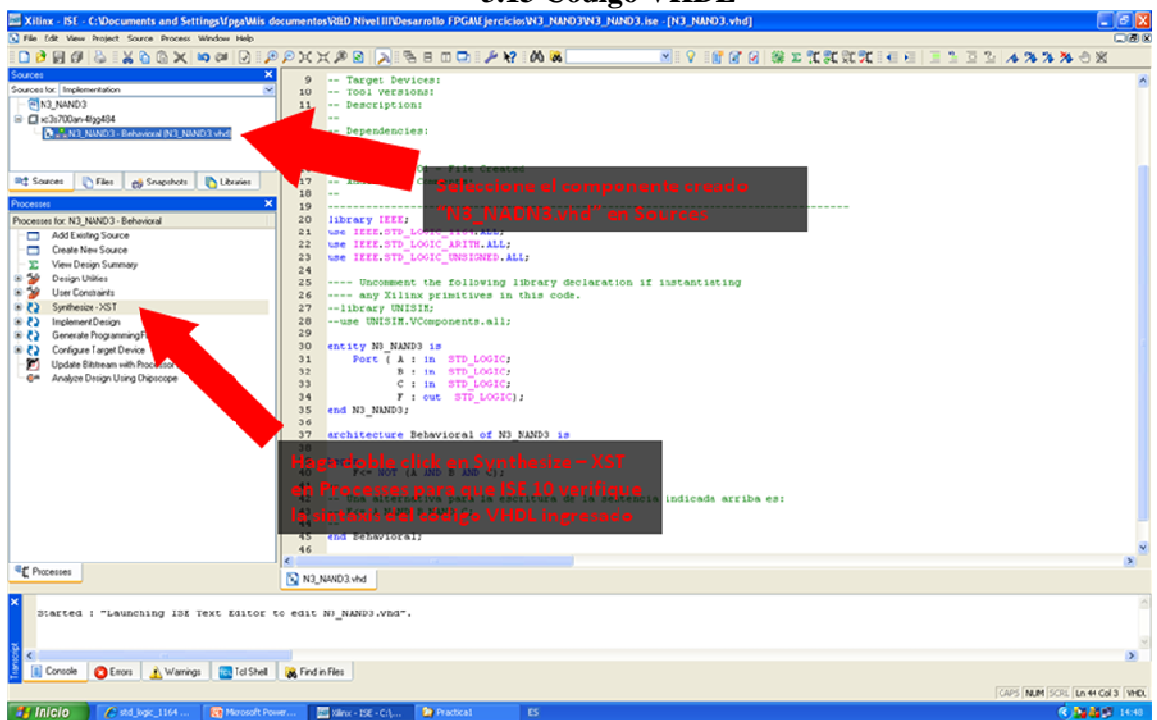
3.11 New source wizard summary



3.12 Estructura VHDL



3.13 Código VHDL



3.14 Nuevo componente creado

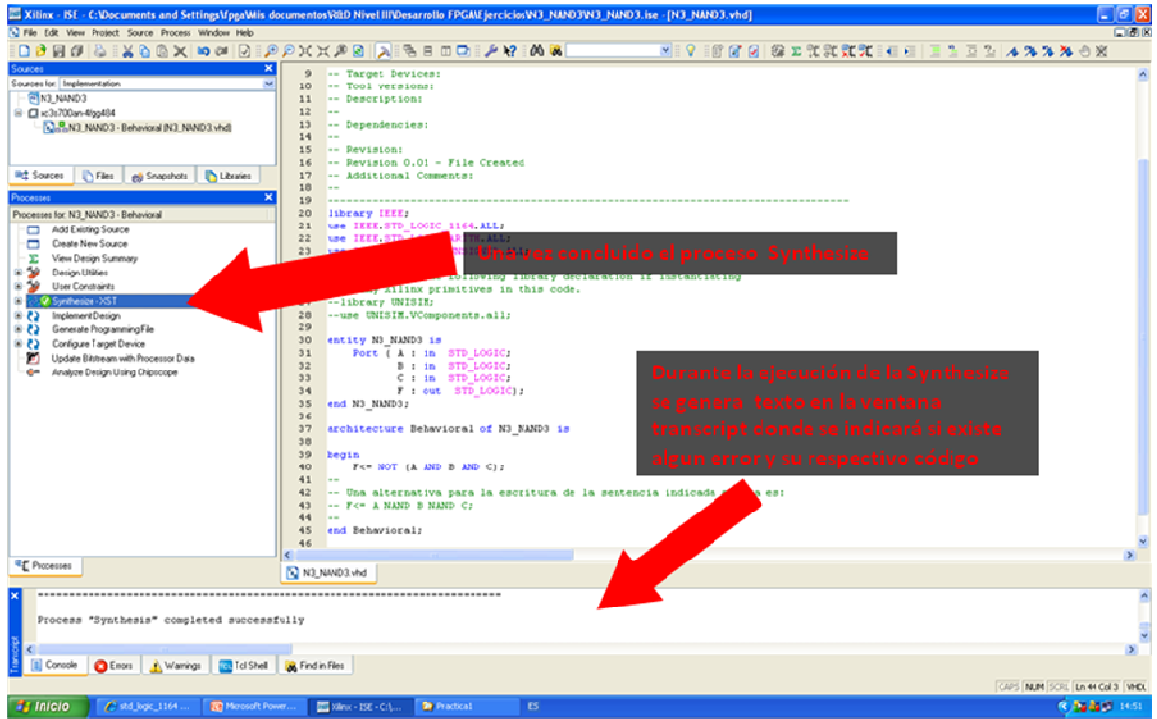


Figura 3.15 synthesis

3.5 CREACIÓN DEL TEST BENCH

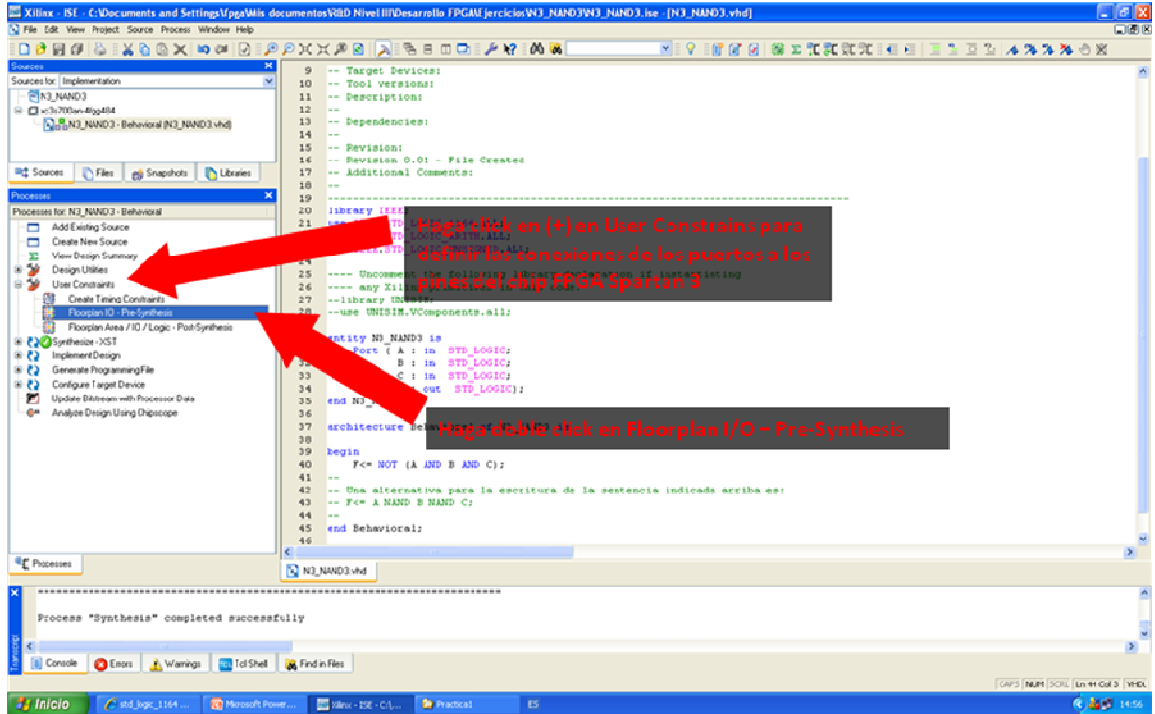


Figura 3.16 User constrains para conexiones de los puertos

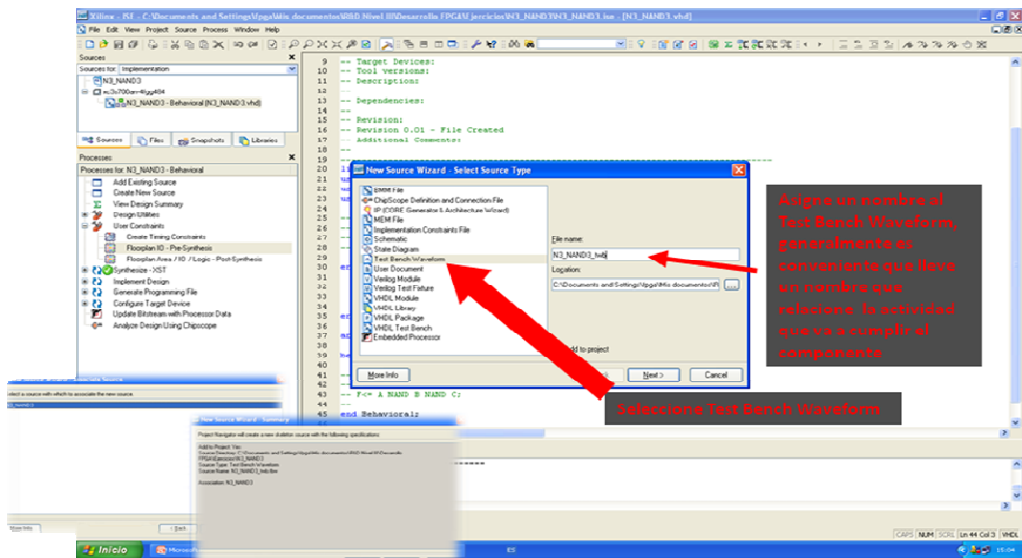
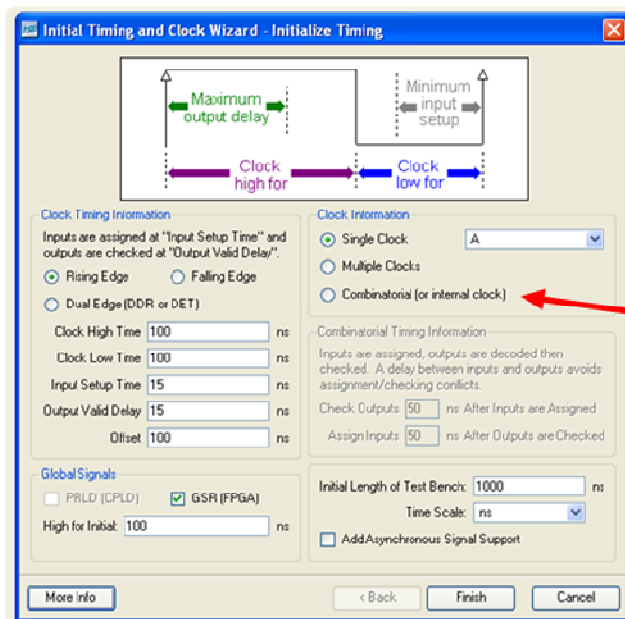
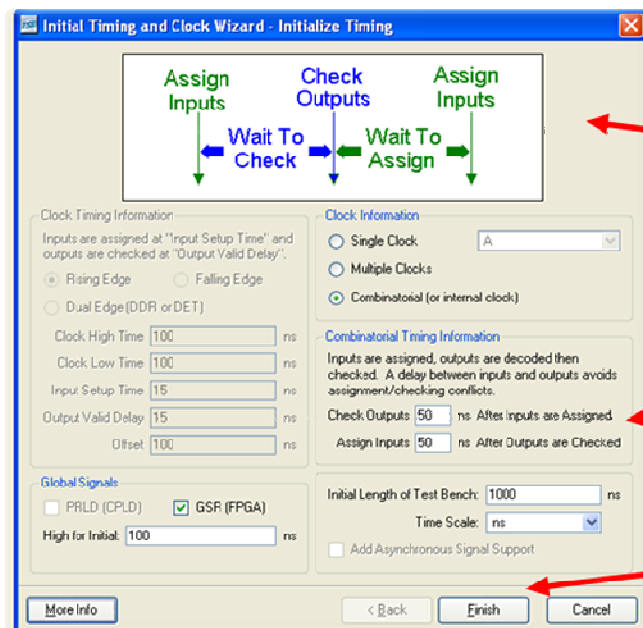


Figura 3.17 Asignación de Nombre



Selecione Combinatorial para indicar de el programa que se va a trabajar con un componente que no necesita definir un reloj para operar

Figura 3.18 Initial timing and clock wizard



Estructura de respuesta del componente combinatoria

Tiempo de exploración y retardo de propagación del componente

Haga click en Finish para continuar

Figura 3.19 Initialize Timing

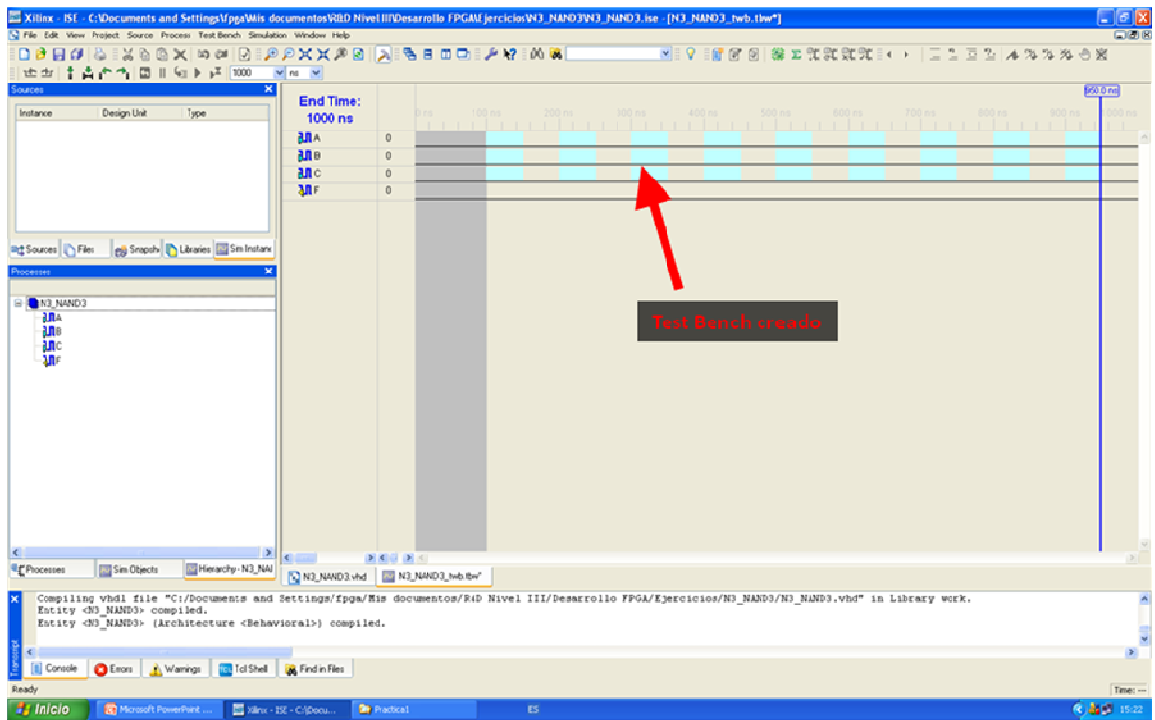


Figura 3.20 Creado test bench

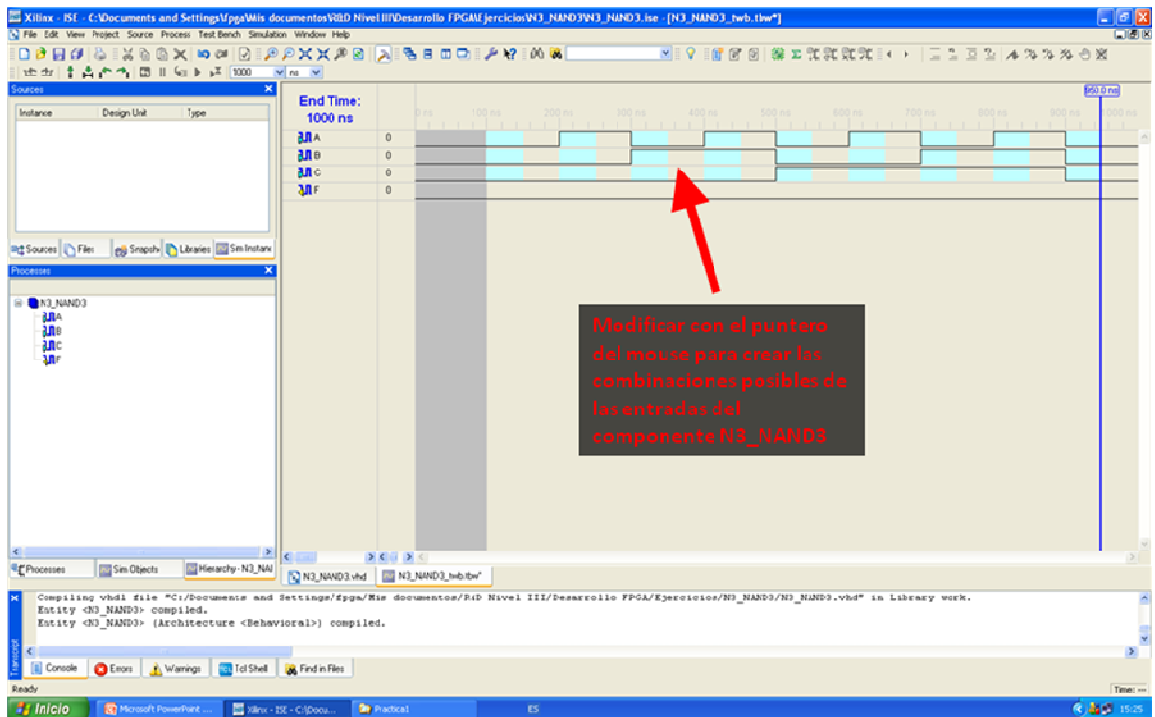


Figura 3.21 Modificación para combinaciones

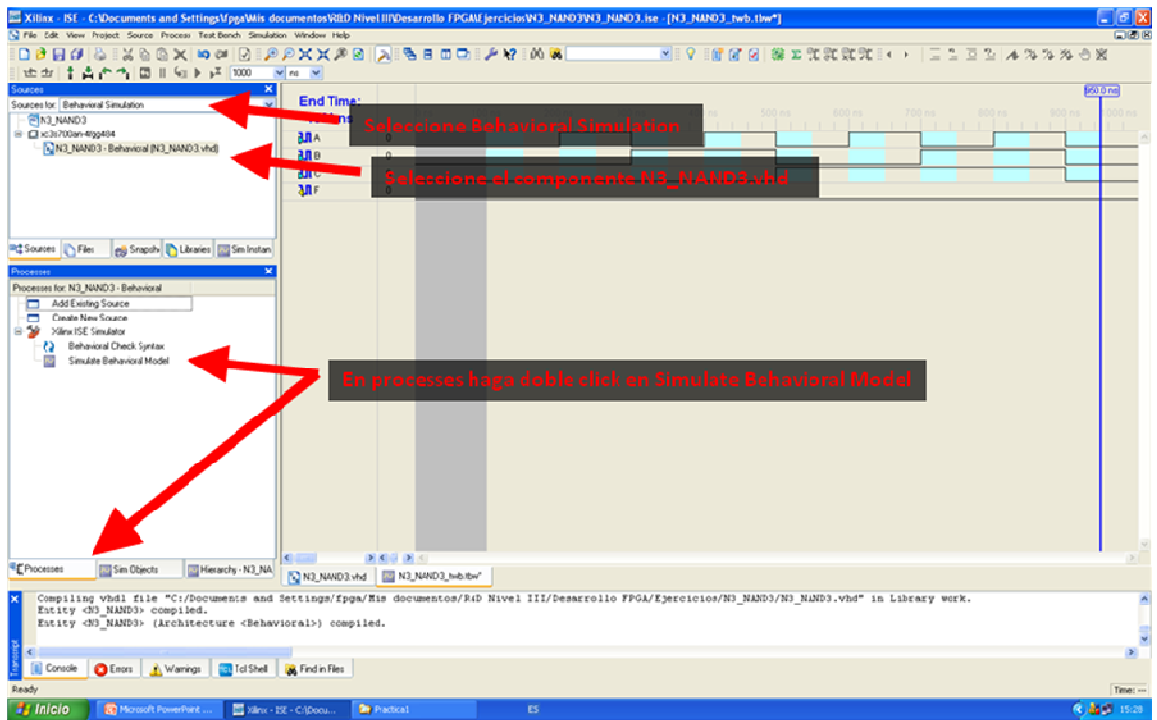


Figura 3.22 Simulación Behavioral Model

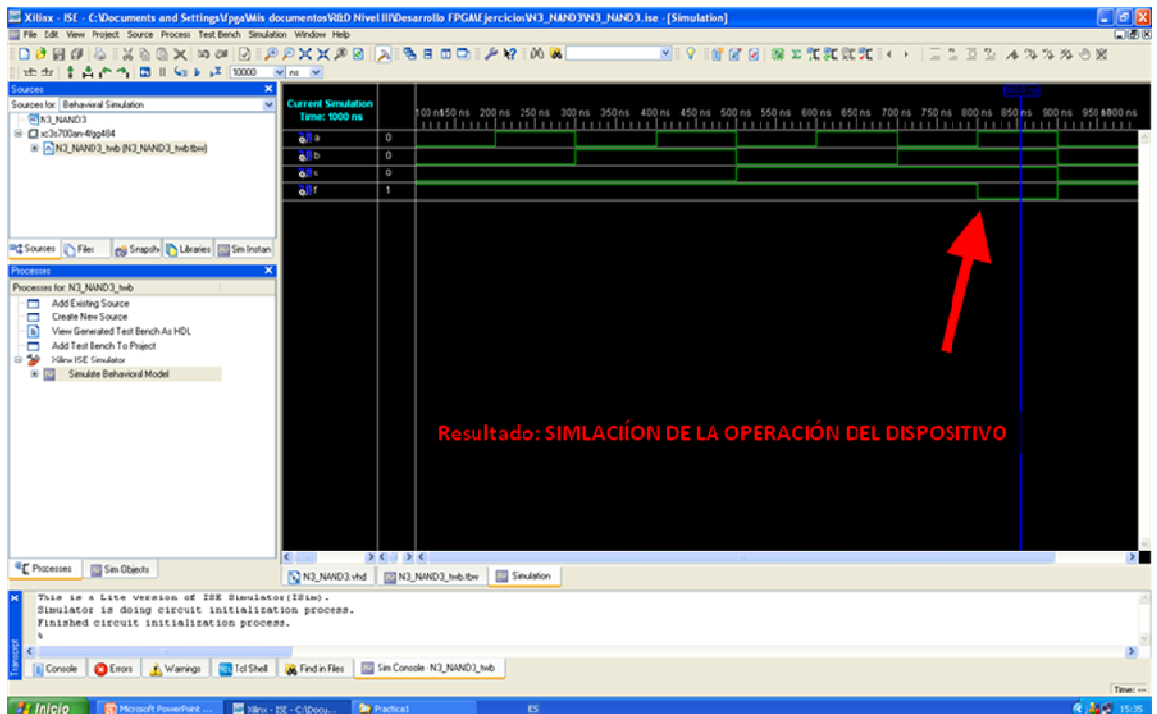


Figura 3.23 Simulación de la operación del dispositivo

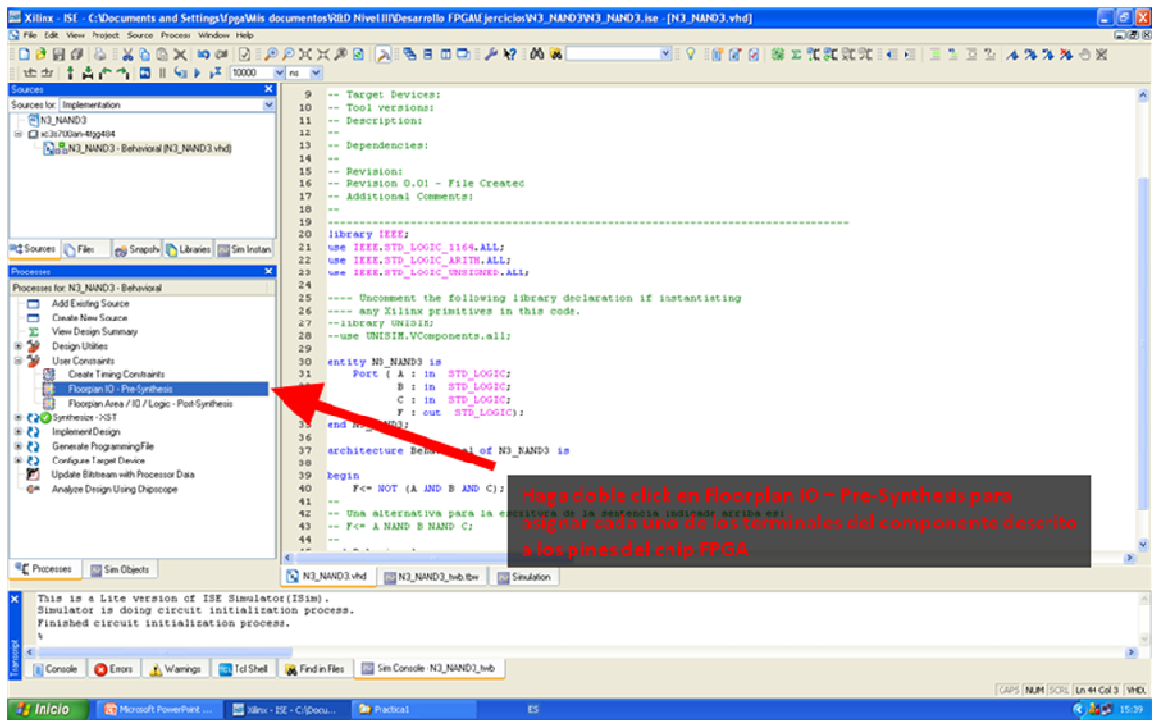
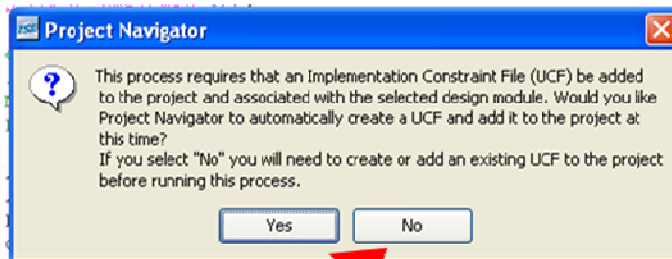


Figura 3.24 Pre- Synthesis



Esta acción le indicará que el programa deberá generar un UCF (User Constraints File) para continuar, presione YES y continue.

Figura 3.25 User constrains File

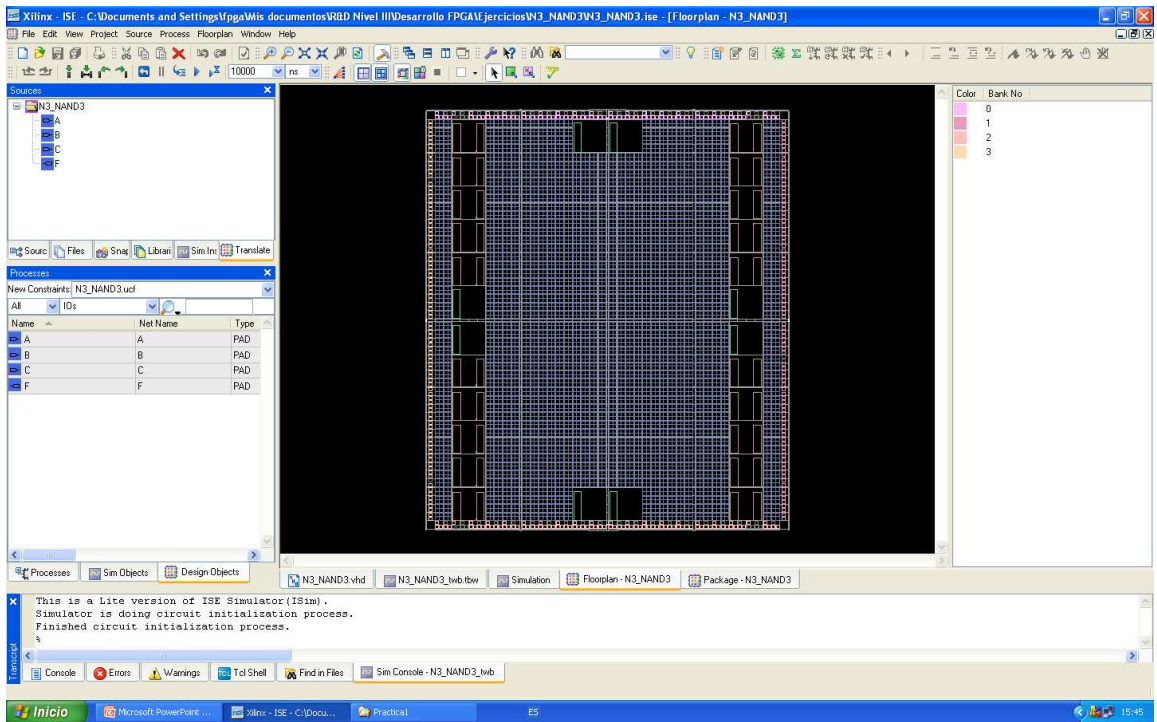


Figura 3.26 floor plan N3 Nand 3

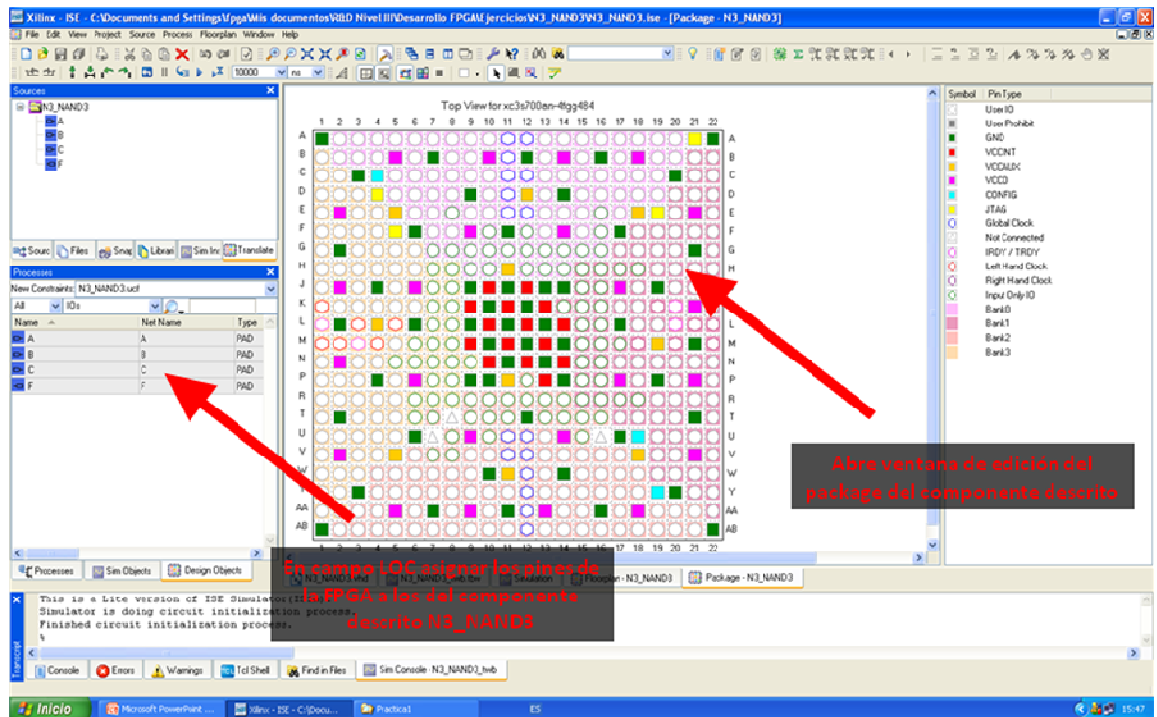


Figura 3.27 Edición de package del componente descrito

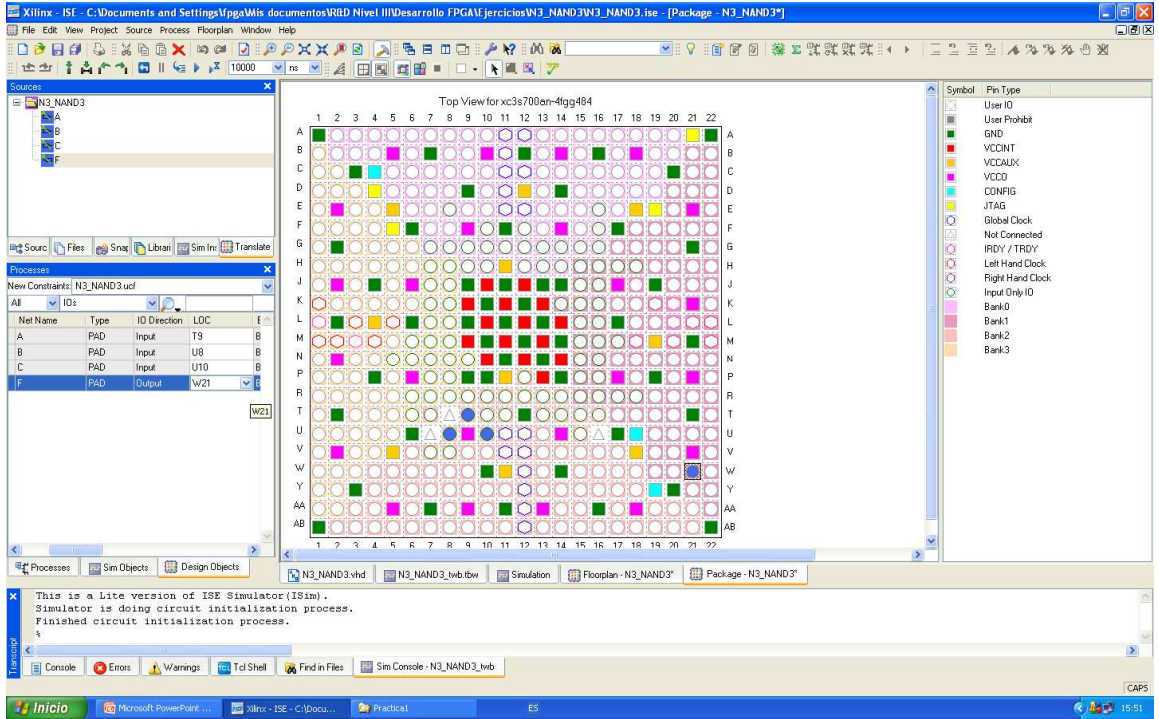


Figura3.28 for view xc 2000

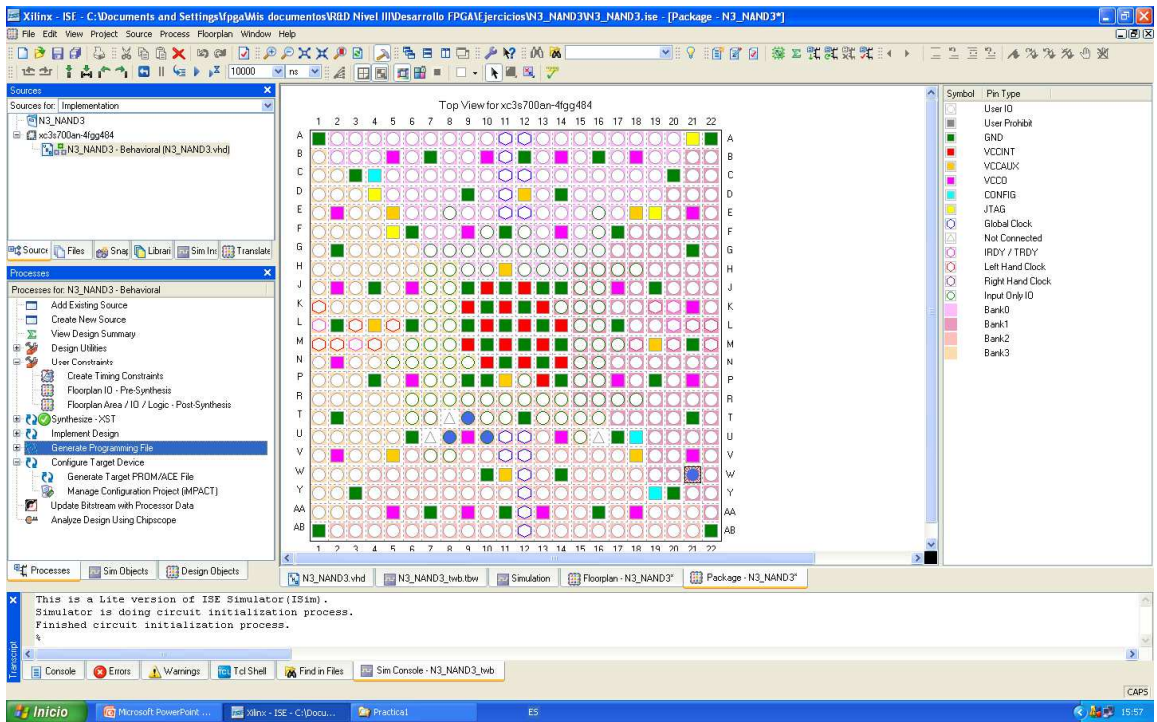


Figura 3.29 Generate program file

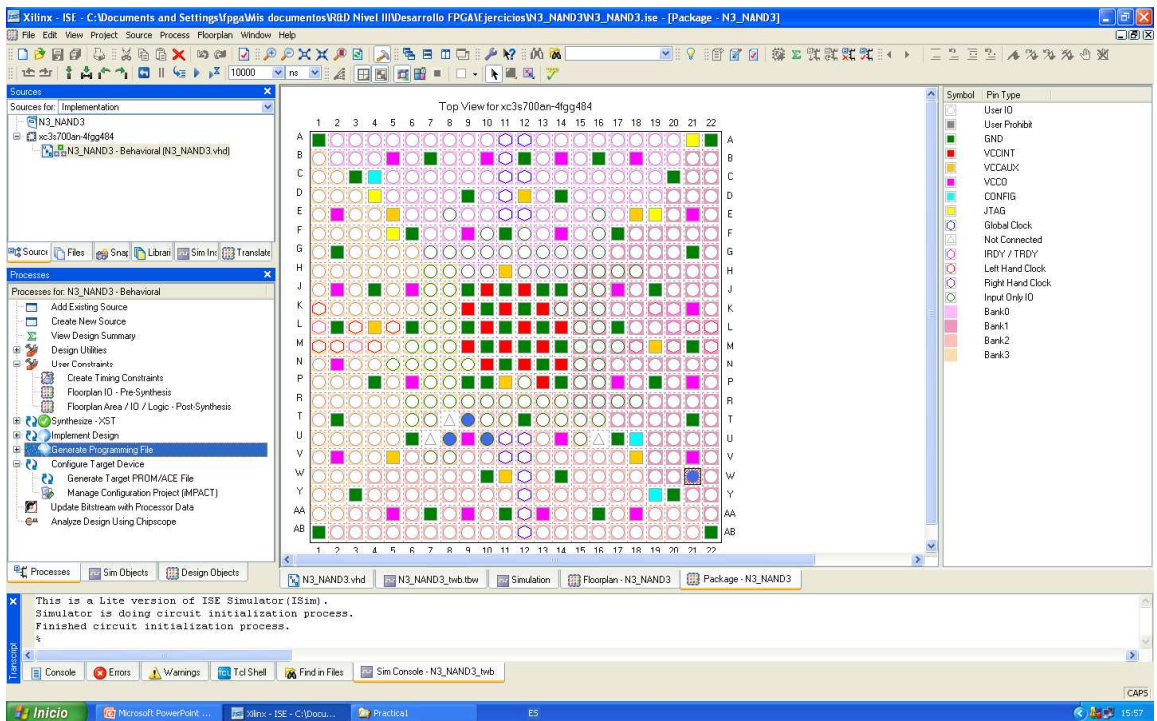


Fig 3.30 Creado archivo para programar

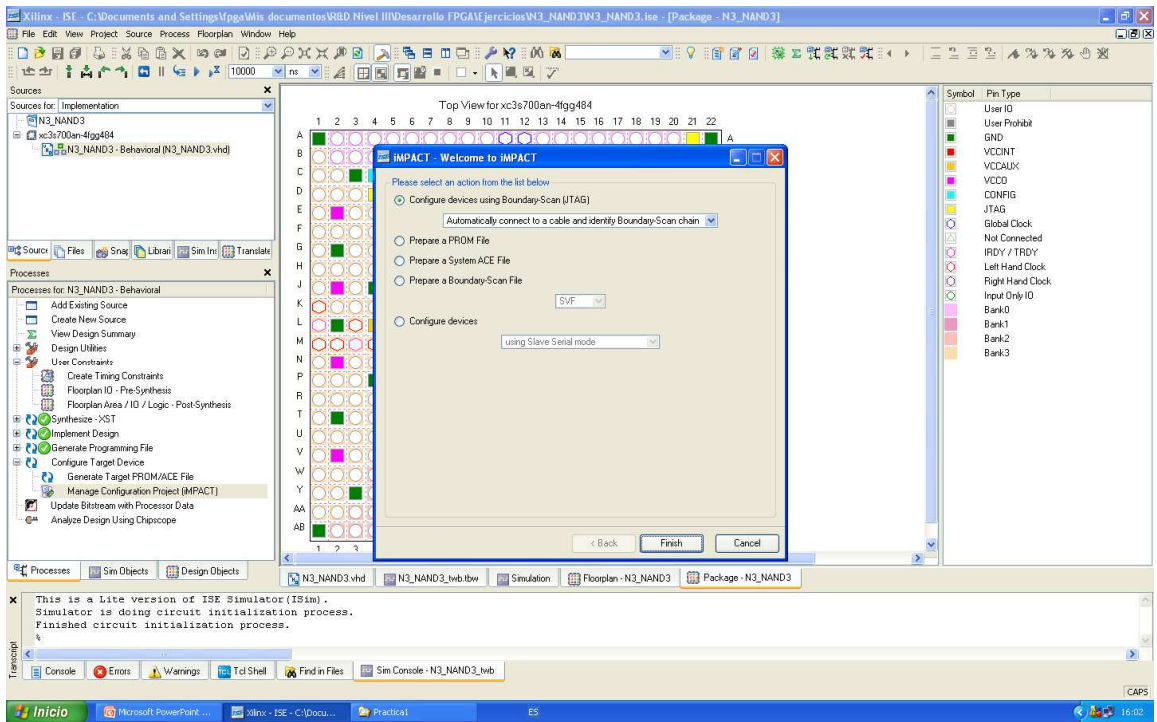


Figura 3.31 ventana a welcome Impact

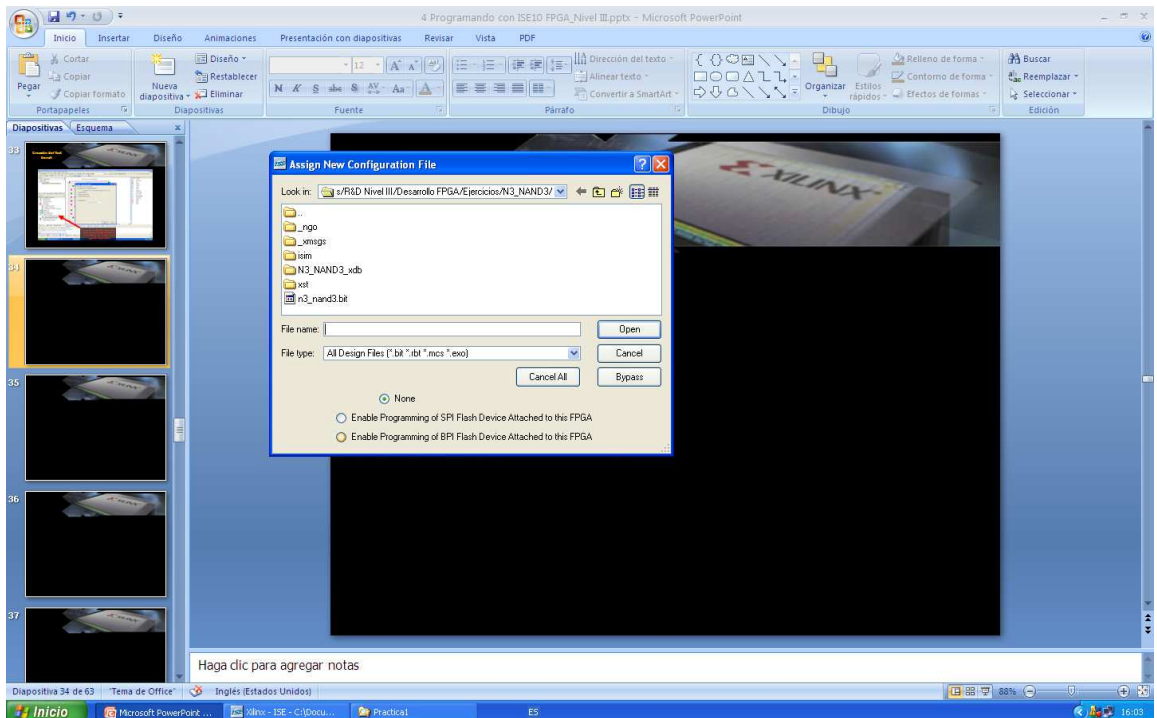


Figura 3.32 Assign new configuration file

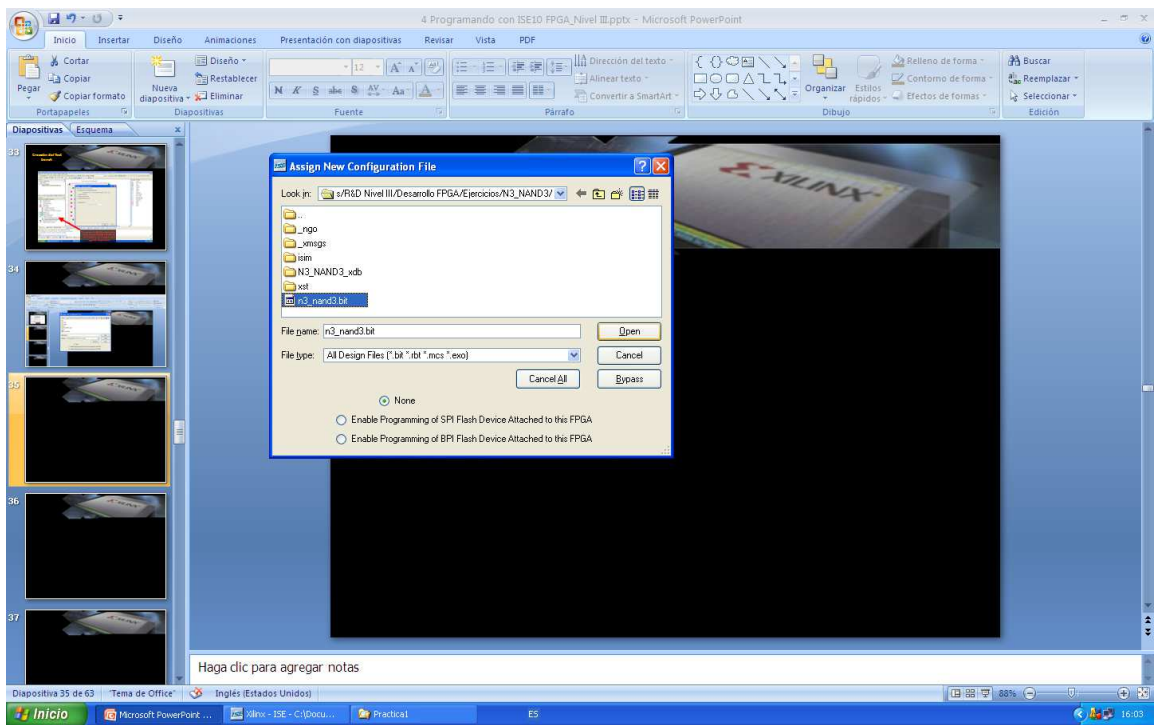


Figura 3.33 Selección del archivo

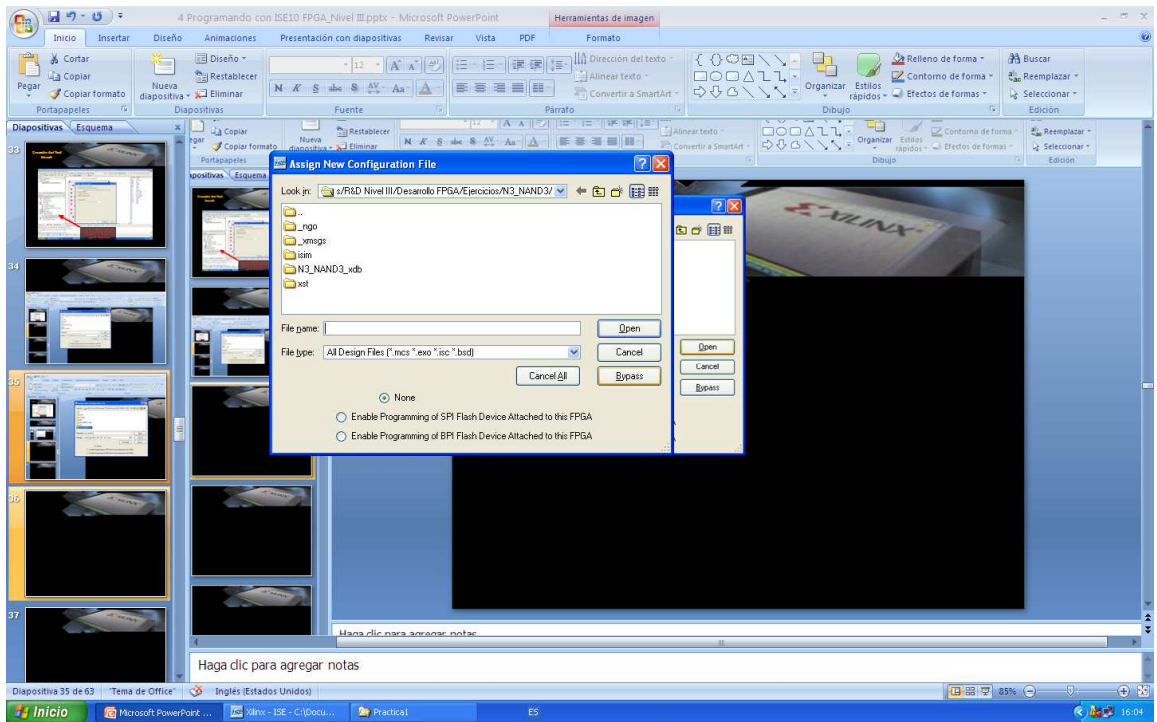


Figura 3.34 file name configuration file

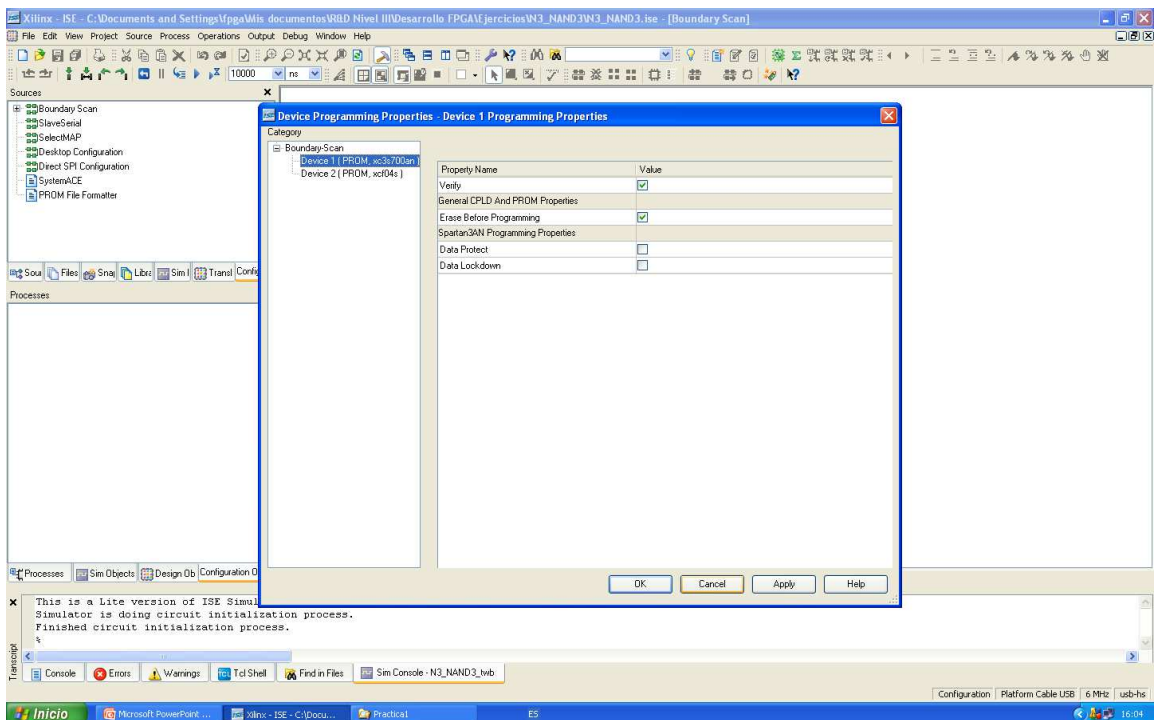


Figura 3.35 Programing propoties

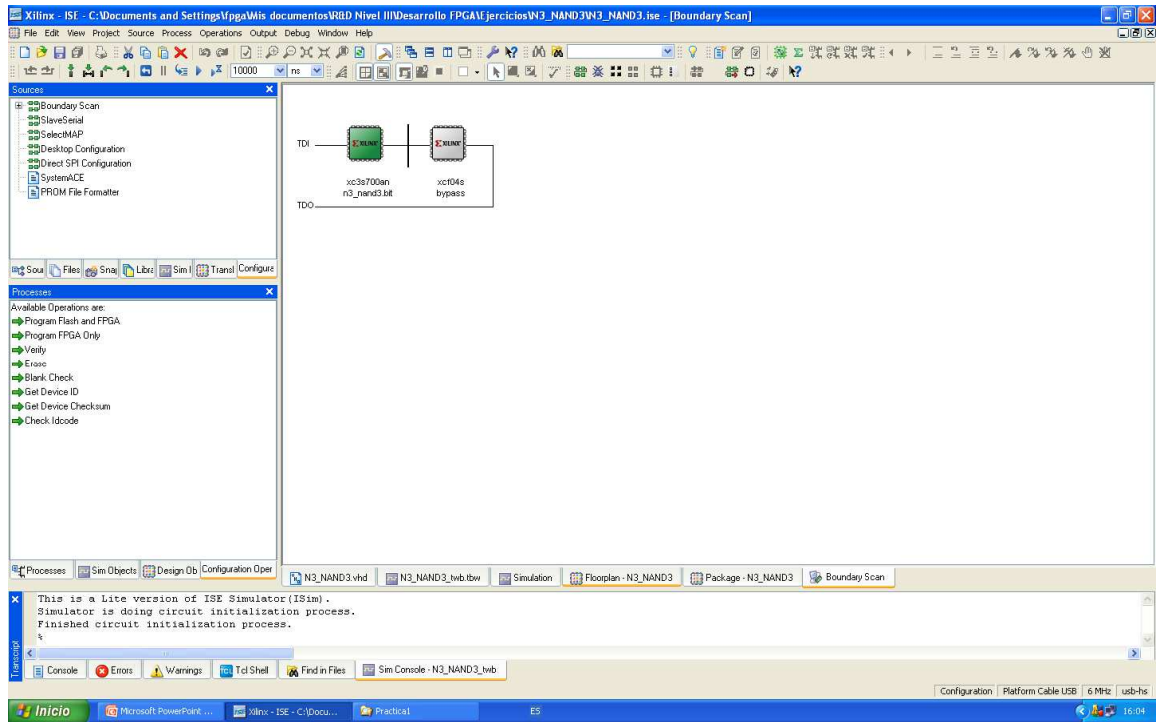


Figura 3.36 Desarrollo de la Fpga

3.6 CREAR UN NUEVO PROYECTO

Crear un nuevo proyecto de ISE, que se centrará en el dispositivo de FPGA en el Spartan-3 Kit de inicio de tarjeta de demostración.

1. Seleccione **File > New Project...** **The New Project Wizard appears.**
2. tipo **tutorial in the Project Name field.**
3. Introduzca o busque la localización (directory path) para un nuevo project. Un subdirectorio tutorial se crea automáticamente.
4. Compruebe que el HDL es seleccionado de la lista de los Top-Level Tipo de origen.
5. Haga clic en Siguiente para ir a la página de propiedades del dispositivo.
6. Rellene las propiedades de la tabla como se muestra a continuación:

PRODUCT CATEGORY: ALL

- ✓ Familia: **Spartan3**
- ✓ Dispositivo: **XC3S200**
- ✓ Paquete: **FT256**
- ✓ Velocidad de Grado: **-4**
- ✓ Nivel superior tipo de Fuente: **HDL**
- ✓ Síntesis de la Herramienta: **XST (VHDL/Verilog)**
- ✓ Simulador: **ISE Simulator (VHDL/Verilog)**
- ✓ Lenguaje Preferido: **Verilog (or VHDL)**
- ✓ Verificar que se ha seleccionado Activar la opción **Summary is selected.**

DEJA LOS VALORES POR DEFECTO EN LOS CAMPOS RESTANTES.

7. Click Next para proceder un nuevo origen en el asistente para un nuevo proyecto. Al final de la proxima seccion, el nuevo proyecto sera completo

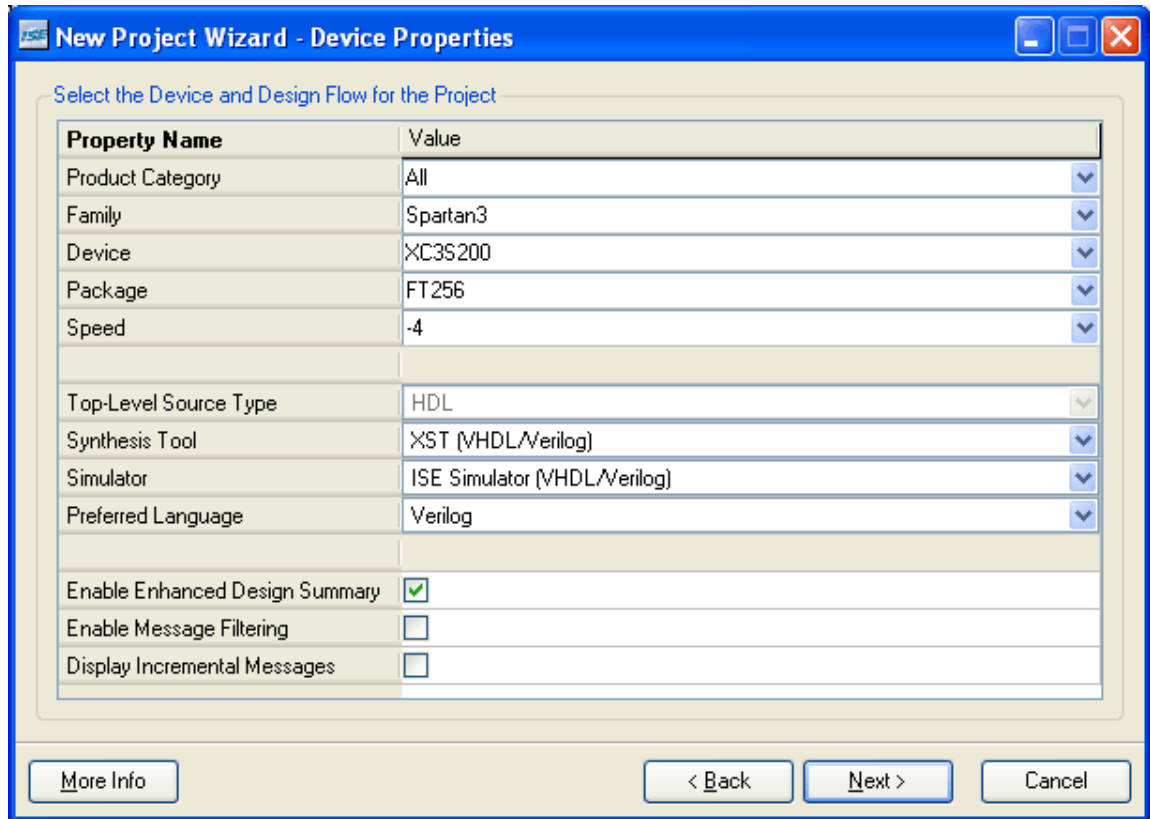


Figura 3.37 Select the device properties

Start → All Programs → Xilinx ISE 9.1i → Project Navigator

3.7 CREAR UNA FUENTE DE HDL

CREAR UN ARCHIVO FUENTE VHDL PARA EL PROYECTO DE LA SIGUIENTE MANERA:

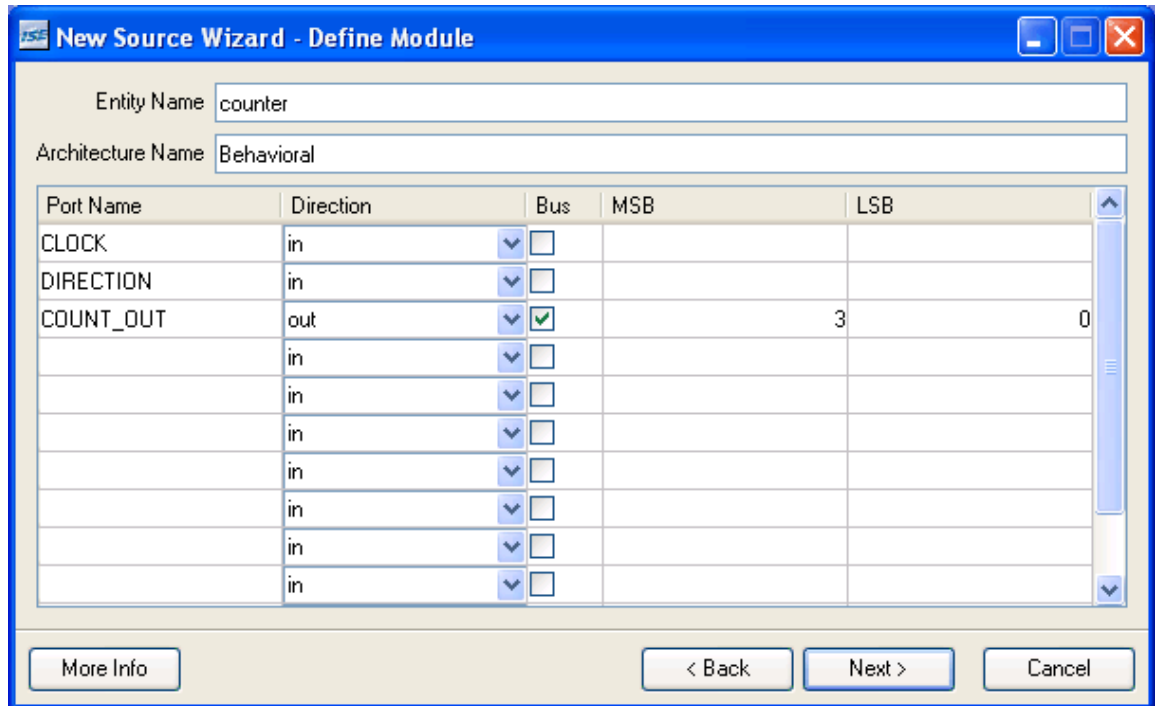


Figura 3.38 define Module

1. Haga clic en el botón **New Project Wizard** para nuevo proyecto.
2. Seleccione **VHDL Module** como el tipo de fuente
3. Escriba en el contador de nombre de archivo.
4. verifique que la casilla **Add to project checkbox** este seleccionado.
5. Click **Next**.
6. Declarar los puertos para el diseño del contador relleno de la información del puerto, como se muestra a continuación
7. Haga clic en Next y luego en Finish en **New Source Wizard** - Resumen cuadro de diálogo para completar la nueva fuente de archivo de plantilla.
8. Haga clic en Next y luego en Finish.

El archivo que contiene entity/architecture par de la arquitectura en el área de trabajo, y aparece el contador en la ficha Fuente, como se muestra a continuación:

3.8 USO DEL LENGUAJE TEMPLATES (VHDL).

El siguiente paso en la creación de la nueva fuente es añadir la descripción de la conducta para el contador. Para ello se utilizará un simple ejemplo de código de las plantillas de lucha contra el ISE Language Templates para el diseño de venta libre.

1. Sitúe el cursor justo debajo *begin* de la declaración para comenzar dentro de la arquitectura de venta libre.

2. Abrir las plantillas Language Templates Editar → Language Templates

Nota: Usted puede colocar las plantillas de idiomas y el archivo de contador seleccionando *Windows* → *Tile*

3. Utilizando el símbolo "+", busque el siguiente ejemplo de código siguiente: VHDL → Synthesis Constructs → Coding Examples → Counters → Binary → Up/Down Counters → Simple Counter

4. Simple contador seleccionado, seleccione, Edit → Use en File o seleccione la plantilla de Uso en el botón de barra de herramientas de archivo. Este paso copia de la plantilla en la fuente de la lucha contra el archivo.

5. cierre el Language Templates

3.9 MONTAJE FINAL DE LA FUENTE VHDL

Agregue la declaración de la siguiente señal para controlar la retroalimentación de la salida del contador por debajo de la declaración de la arquitectura y por encima de la primera declaración a comenzar:

```
signal count_int : std_logic_vector (3 downto 0) := "0000";
```

Personalizar el archivo de código fuente para el diseño mediante la sustitución de la lucha contra el puerto y el nombre de los marcadores de posición de la señal con los reales de la siguiente manera:

Reemplazar todas las apariciones of <clock> with CLOCK

Reemplazar todas las apariciones of <count_direction> with DIRECTION

Reemplazar todas las apariciones of <count> with count_int

3. Agregue a continuación la siguiente línea para el proceso final, de declaración:

```
COUNT_OUT <= count_int;
```

Guarde el archivo seleccionando **File** → **Save**.

Cuando haya terminado, la fuente de la lucha contra el archivo se parecerá a la imagen.

Ahora ha creado la fuente de VHDL para el proyecto del tutorial. Vaya a la sección “Checking the Syntax of the New Counter Module” section.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx primitive in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity counter is
  Port ( CLOCK : in  STD_LOGIC;
        DIRECTION : in  STD_LOGIC;
        COUNT_OUT : out  STD_LOGIC_VECTOR (3 downto 0));
end counter;

architecture Behavioral of counter is
  signal count_int : std_logic_vector(3 downto 0) := "0000";
begin
  process (CLOCK)
  begin
    if CLOCK='1' and CLOCK'event then
      if DIRECTION='1' then
        count_int <= count_int + 1;
      else
        count_int <= count_int - 1;
      end if;
    end if;
  end process;
  COUNT_OUT <= count_int;
end Behavioral;
```

CUANDO LOS ARCHIVOS DE ORIGEN ESTE COMPLETO, COMPRUEBE LA SINTAXIS DEL DISEÑO PARA ENCONTRAR ERRORES.

1. Verifique que la **Synthesis/Implementation** se seleccione de la lista desplegable en la ventana de Fuentes.
2. Seleccione el contador de Fuente de diseño en la ventana de Fuentes para mostrar la relación en la ventana de procesos.

3. Click en “+” **next to the Synthesize-XST** proceso para ampliar el grupo de procesos.
4. Doble-click en **Check Syntax process**.

Nota: Usted debe corregir los errores encontrados en los archivos de origen. Usted puede comprobar si hay errores en la pestaña de la consola de la ventana de transcripción. Si continúa sin una sintaxis válida, usted no será capaz de simular o sintetizar el diseño.

3.10 SIMULACION DE DISEÑO.

Crear un banco de pruebas de forma de onda con el estímulo de entrada que puede utilizar para comprobar la funcionalidad del módulo de venta libre. La forma de onda banco de pruebas, es una vista gráfica.

CREAR UN BANCO DE PRUEBAS DE LA FORMA DE ONDA DE LA SIGUIENTE MANERA:

1. Seleccione el archivo HDL en la ventana Fuentes.
2. Crear una nueva fuente de banco de pruebas de selección de Project → New Source.
3. En la nueva fuente asistente, de forma de onda Prueba de selección de banco como el tipo de fuente, y counter_tbw tipo en el campo Nombre de archivo
4. Haga click en Next.
5. La página de origen muestra que estamos asociando el banco de pruebas de forma de onda con el archivo de código fuente libre. Haga click en Next.
6. La página Resumen muestra que la fuente será añadido al proyecto, y que muestra el directorio de origen, tipo y nombre. Haga click en Finish.
7. Es necesario fijar la frecuencia de reloj, el tiempo de instalación y tiempo de retardo de salida en el cuadro de diálogo. Iniciar sincronización antes de que el banco de pruebas, abra la ventana de edición de forma de onda

Los requisitos para este diseño son las siguientes:

- ✓ El contador debe funcionar correctamente con una frecuencia de reloj de entrada = 25 MHz.
- ✓ La entrada de dirección será válida 10 ns antes del flanco de subida de reloj.
- ✓ La producción (COUNT_OUT) debe ser válido después de 10 ns el flanco de subida de reloj.

Los requerimientos de diseño corresponden a los valores de abajo. Rellene los campos en el cuadro de diálogo al Iniciar sincronización con la siguiente información:

Reloj High Time: 20 ns.
Reloj Time Low: 20 ns.
Ajuste de entradas: Más de 10 ns.
Salida válido Plazo: 10 ns.
Offset: 0 ns.
Mundial Señales: GSR (FPGA)

Nota: Cuando GSR (FPGA) está habilitada, 100 ns. se añade el valor de desplazamiento de forma automática.

- ✓ Longitud inicial del banco de ensayo: 1500 ns.

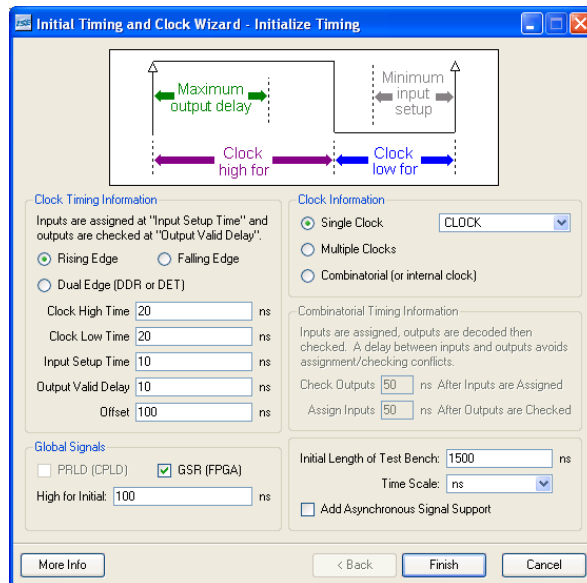


Figura 3.39 simulación de diseño

Deja los valores por defecto en los campos restantes.

8. Haga clic en Finalizar para completar la inicialización de tiempo.

9. Las áreas azules sombreadas que preceden el flanco de subida del reloj corresponden al tiempo de configuración de entrada en el cuadro de diálogo Iniciar sincronización. Activar el puerto DIRECCIÓN para definir el estímulo de entrada para el diseño del contador de la siguiente manera:

Haga clic en la celda azul en aproximadamente 300 ns para afirmar alta dirección para que el contador contará arriba.

Haga clic en la celda azul de aproximadamente de 900 ns para afirmar DIRECCIÓN baja para que el contador de cuenta atrás.

Nota: Para la alineación más precisa, puede utilizar el Zoom In y Zoom Out botones de barra de herramientas.

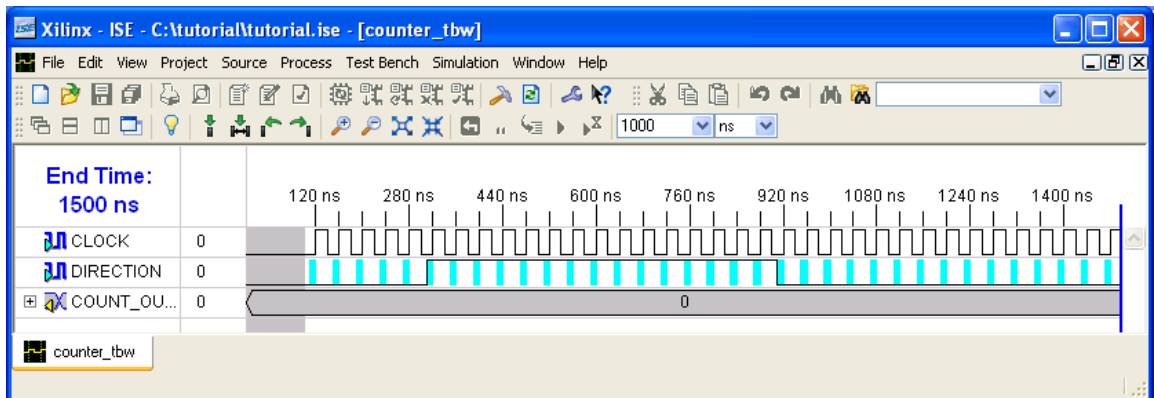


Figura 3.40 counter_tbw

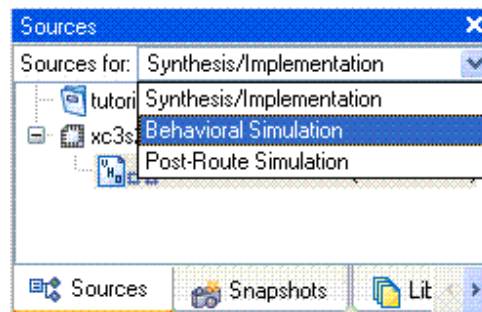


Figura 3.41 Behavioral Simulation

10. Guardar la forma de onda.

11. En la ventana Fuentes, seleccione la vista de simulación del comportamiento para ver que el banco de pruebas de archivos de forma de onda se añade automáticamente a su proyecto

12. Cierre la forma de onda de banco de pruebas.

3.11 SIMULACION DE LA FUNCIONALIDAD DEL DISEÑO

Los resultados de la simulación de forma de onda se verá de la siguiente manera:

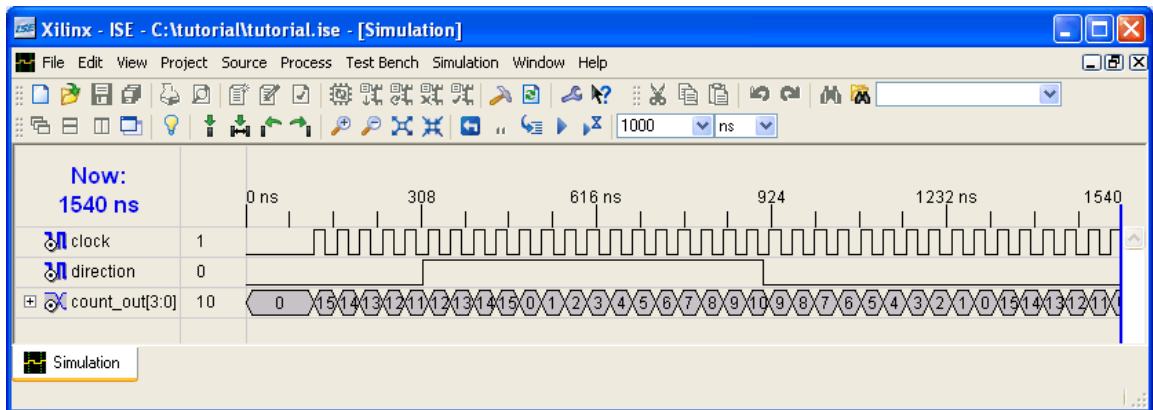


Figura 3.42 Modulación Counter

Compruebe que el contador de las funciones de diseño mediante la realización de simulación de comportamiento de la siguiente manera:

1. Compruebe que la simulación del comportamiento y counter_tbw se seleccionan en la ventana de Fuentes.

2. En la ficha Procesos, haga clic en el signo "+" para ampliar el proceso de Xilinx ISE Simulator y haga doble click **Simulate Behavioral Model process**.

El Simulador de ISE abre y ejecuta la simulación para el final del banco de pruebas.

3. Para ver los resultados de la simulación, seleccione **Simulation tab and zoom in on the transitions**.

Nota: Puede pasar por alto las filas que comienzan con TX.

4. Compruebe que el contador está contando de arriba abajo como se esperaba.

5. Cierre la vista de la simulación. Si se le pregunta con el siguiente mensaje: "Tienes una simulación activa y abierta. ¿Estás seguro de que desea cerrar? ", Haga clic en Sí para continuar.

Acaba de finalizar la simulación de su diseño utilizando el simulador de ISE.

CREAR RESTRICCIONES TEMPORALES

Especifique el tiempo entre la FPGA y su lógica alrededores, así como la frecuencia, el diseño debe funcionar a internos de la FPGA. El calendario se especifica introduciendo limitaciones que rigen la colocación y el enrutamiento del diseño.

Se recomienda que introduzca restricciones globales.

La restricción de periodo de reloj especifica la frecuencia de reloj en la que su diseño debe operar dentro de la FPGA.

Las restricciones de desplazamiento se deben especificar cuándo esperan los datos válidos en las entradas de FPGA y cuando los datos válidos estarán disponibles en las salidas de FPGA.

Para restringir el diseño realizar lo siguiente:

1. Seleccione Síntesis / Aplicación de la lista desplegable en la ventana de Fuentes.
2. Seleccione la lucha contra el archivo de código fuente de HDL.
3. Haga click en el signo "+" **sign next to the User Constraints processes group**, y haga doble click **Create Timing Constraints process**.

ISE ejecuta la síntesis y los pasos Traducir y crea automáticamente un archivo de restricciones de usuario (UCF). Se le pedirá con el siguiente mensa

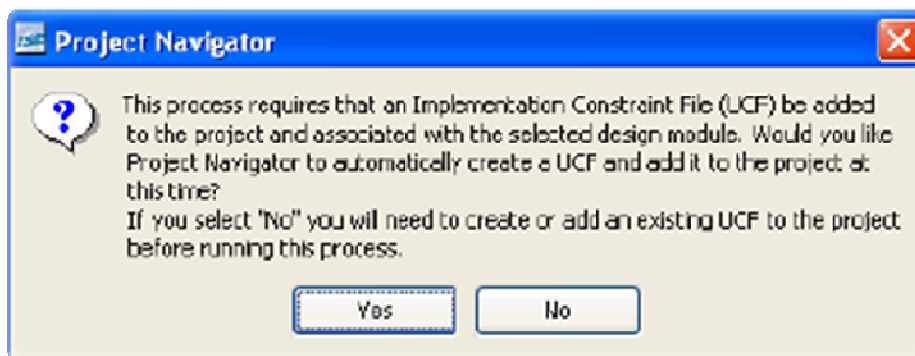


Figura 3.43 Opciones de Project

4. Haga click en Yes para agregar el archivo de la UCF a su proyecto.

El archivo counter.ucf se añade a su proyecto y es visible en la ventana Fuentes.

El editor de restricciones Xilinx se abre automáticamente.

Nota: También puede crear un archivo de la UCF para su proyecto mediante la selección **Project** → **Create New Source**.

En el siguiente paso, introducir valores en los campos asociados con el reloj en las restricciones Ficha Editor Global.

5. Seleccione Reloj en la esfera del reloj Net Nombre, seleccione el botón de la barra de herramientas Período o haga doble clic en el campo Periodo de vacío para mostrar el cuadro de reloj Periodo de diálogo.

6. Introduzca 40 ns en el campo Hora.

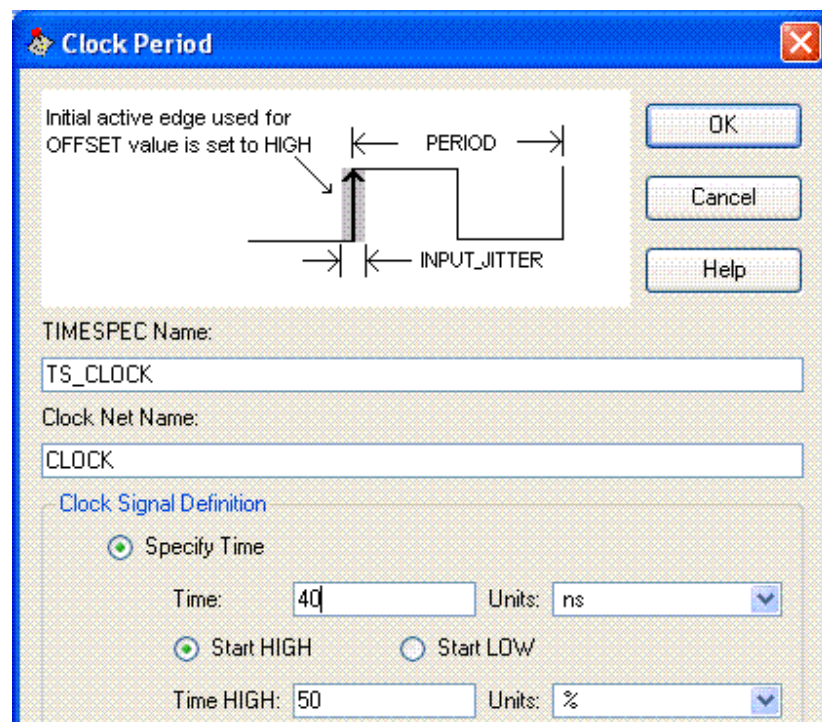


Figura 3.44 Clock Period

7. Haga click en OK.

8. Seleccione el botón de notas para el programa de instalación la barra de herramientas o haga doble clic en el Cuadro de vacío para la instalación de campo para mostrar las notas al programa de instalación cuadro de diálogo.

9. Introduzca 10 ns en el campo de desplazamiento para activar la función de compensar la restricción.

10. Haga click en ok.

11. Seleccione el **Clock to Pad toolbar button** or **double-click the empty Clock to Pad field** al reloj para Pad cuadro de diálogo.

12. Escriba 10 ns en el campo de desplazamiento para establecer la restricción de retardo de salida.

13. Haga clic en OK.

Las restricciones se muestran en las restricciones (read-write) ficha, como se muestra a continuación

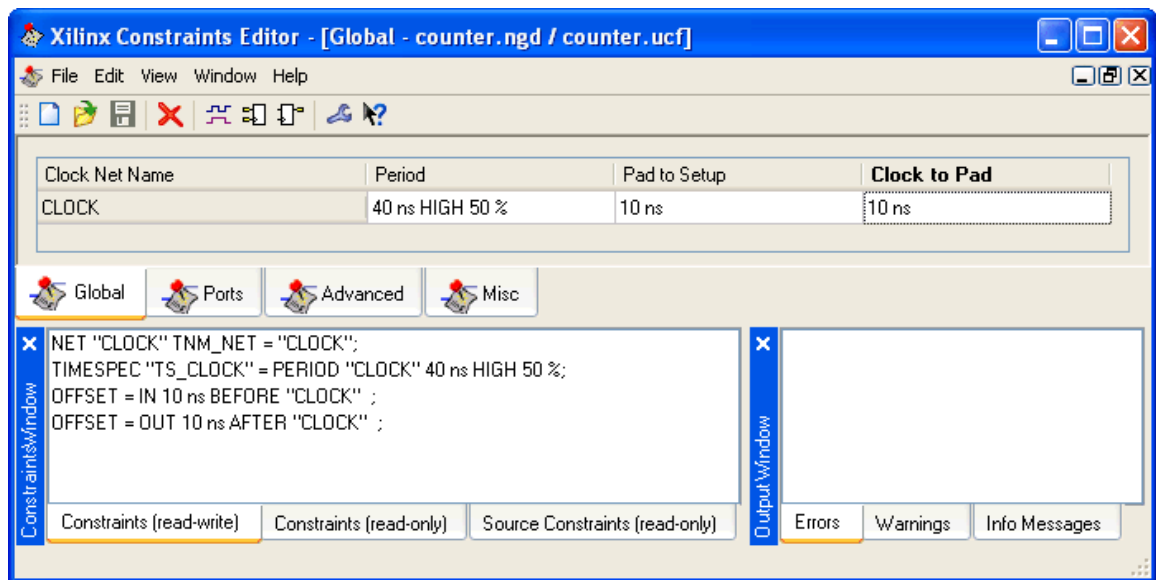


Figura 3.45 Xilinx constraints Editor

14. Guardar las restricciones temporales. Si se le pide que vuelva a TRANSLATE o XST paso, haga clic en OK para continuar.

15. Cierre el Editor de restricciones.

DISEÑO Y APLICACIÓN DE RESTRICCIONES VERIFICAR

Implementar el diseño y verificar que se ajusta a las limitaciones de tiempo especificado en la sección anterior.

3.12 APLICACIÓN DEL DISEÑO

1. Seleccione la lucha contra el archivo de código fuente en la ventana de Fuentes.
2. Abra el resumen de diseño haciendo doble clic en el proceso de diseño Ver resumen en la ficha Procesos.
3. Haga doble clic en Aplicar el proceso de diseño en la ficha Procesos.
4. Tenga en cuenta que después de la aplicación está completa, los procesos de aplicación con una marca verde junto a ellos indicando que se ha completado correctamente, sin errores o advertencias.

The screenshot shows the Xilinx ISE Design Summary window for a project named 'tutorial.isc'. The window is divided into several panes:

- Sources:** Shows the project hierarchy with 'xc3s200-4h256' and 'counter - Behavioral (counter.vhd)'.
- Processes:** Lists various design processes, with 'Design Summary' selected.
- FPGA Design Summary:** A tree view of report categories like Design Overview, Errors and Warnings, and Detailed Reports.
- TUTORIAL Project Status:** A table showing project details.
- TUTORIAL Partition Summary:** A table indicating no partition information was found.
- Device Utilization Summary:** A table showing logic utilization and distribution.
- Performance Summary:** A table showing additional JTAG gate count for IOBs.

Project	tutorial.isc	Current State:	Placed and Routed
File:	counter	• Errors:	
Module Name:		• Warnings:	
Target Device:	xc3s200-4h256	• Updated:	Tue Apr 25 15:37:47 2006
Product Version:	ISE 8.2i		

Logic Utilization	Used	Available	Utilization	Note(s)
Number of Slice Flip Flops	4	3,840	1%	
Number of 4 input LUTs	4	3,840	1%	
Logic Distribution				
Number of occupied Slices	3	1,920	1%	
Number of Slices containing only related logic	3	3	100%	
Number of Slices containing unrelated logic	0	3	0%	
Total Number of 4 input LUTs	4	3,840	1%	
Number of bonded IOBs	6	173	3%	
Number of GCLKs	1	8	12%	
Total equivalent gate count for design	62			
Additional JTAG gate count for IOBs	288			

Figura 3.46 Process Design summary

5. Busque la tabla Resumen de rendimiento en la parte inferior del Resumen de diseño.

6. Haga click en el enlace se conocieron las limitaciones en el ámbito de tiempo para ver el informe Timing Constraints. Compruebe que el diseño cumpla los requisitos de tiempo especificado.

7. Cierre el resumen del diseño.

3.13 ASIGNACIÓN DE PIN UBICACIÓN RESTRICCIONES

Especificar las ubicaciones de pines para los puertos del diseño a fin de que estén conectados correctamente en el Spartan-3 Kit de inicio de tarjeta de demostración.

Para restringir los puertos de diseño a los pines del paquete, haga lo siguiente:

1. Compruebe que está seleccionada en la lucha contra la ventana de Fuentes.
2. Haga doble click en el proceso de Paquete Asignar Pines en grupo el proceso de restricciones de usuario. El pinout Xilinx y Área Editor de restricciones (PACE) se abre.
3. Seleccione la pestaña de vista de paquetes.
4. En la ventana de la lista de objetos de diseño, introducir una ubicación de pines para cada pin en la columna Loc utilizando la siguiente información:

- ✓ CLOCK input port connects to FPGA pin **T9 (GCK0 signal on board)**
- ✓ COUNT_OUT<0> output port connects to FPGA pin **K12 (LD0 signal on board)**
- ✓ COUNT_OUT<1> output port connects to FPGA pin **P14 (LD1 signal on board)**
- ✓ COUNT_OUT<2> output port connects to FPGA pin **L12 (LD2 signal on board)**
- ✓ COUNT_OUT<3> output port connects to FPGA pin **N14 (LD3 signal on board)**
- ✓ DIRECTION input port connects to FPGA pin **K13 (SW7 signal on board)**

Tenga en cuenta que los lugares PIN asignado se muestran en azul:

5. Seleccione File → Save. Se le pedirá que seleccione el tipo de delimitador de autobús sobre la base de la herramienta de síntesis que está utilizando. Seleccione XST Default<> y haga click en ok.

6. Cerrar la PACE.

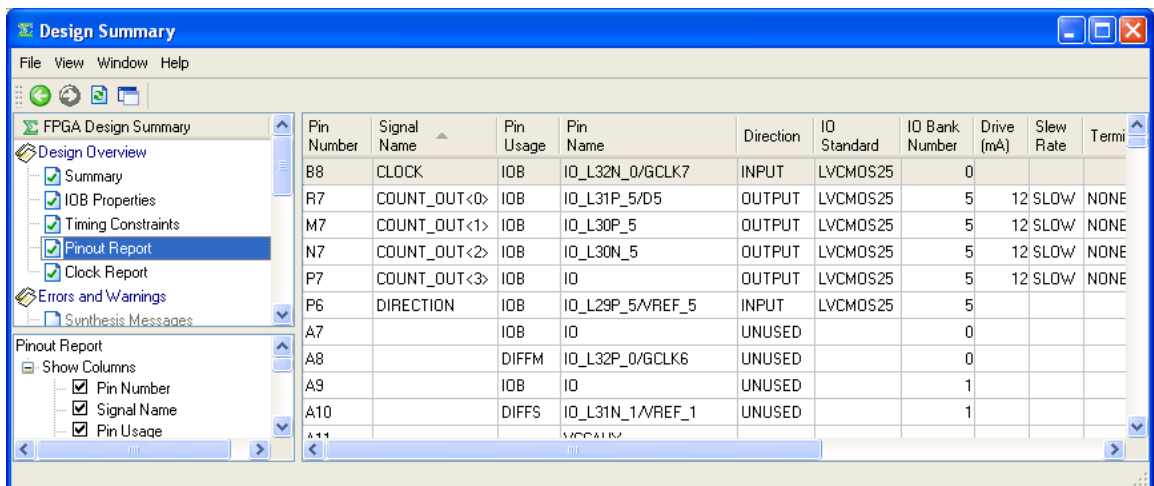
Tenga en cuenta que los procesos de diseño Aplicar tienen un signo de interrogación de color naranja al lado de ellos, indicando que están fuera de fecha con uno o más de los archivos de diseño. Esto es porque el archivo de la UCF ha sido modificado.

3.14 REIMPLEMENTAR DISEÑO Y VERIFICAR PIN LOCALIDADES

Reimplementar el diseño y verificar que los puertos del diseño se encaminan a la lucha contra el paquete de alfileres se especifica en la sección anterior.

En primer lugar, examinar el Informe Pinout de la aplicación anterior de la siguiente manera:

1. Abra el resumen de diseño haciendo doble clic en el proceso de diseño Ver resumen en la ventana de procesos.
2. Seleccione el informe de Pinout y seleccione la columna Nombre de la señal de cabecera para ordenar los nombres de señales. Notificación de los números de PIN asignado a los puertos de diseño en la ausencia de restricciones de localización.



Pin Number	Signal Name	Pin Usage	Pin Name	Direction	IO Standard	IO Bank Number	Drive (mA)	Slew Rate	Termi
B8	CLOCK	IOB	ID_L32N_0/GCLK7	INPUT	LVCMOS25		0		
R7	COUNT_OUT<0>	IOB	ID_L31P_5/D5	OUTPUT	LVCMOS25	5	12 SLOW	NONE	
M7	COUNT_OUT<1>	IOB	ID_L30P_5	OUTPUT	LVCMOS25	5	12 SLOW	NONE	
N7	COUNT_OUT<2>	IOB	ID_L30N_5	OUTPUT	LVCMOS25	5	12 SLOW	NONE	
P7	COUNT_OUT<3>	IOB	ID	OUTPUT	LVCMOS25	5	12 SLOW	NONE	
P6	DIRECTION	IOB	ID_L29P_5/REF_5	INPUT	LVCMOS25	5			
A7		IOB	ID	UNUSED			0		
A8		DIFFM	ID_L32P_0/GCLK6	UNUSED			0		
A9		IOB	ID	UNUSED			1		
A10		DIFFS	ID_L31N_1/REF_1	UNUSED			1		
A11									

Figura 3.47 Pinout report

3. Reimplementar el diseño haciendo doble clic en Aplicar el proceso de diseño.
4. Seleccione el informe Pinout de nuevo y seleccione la columna Nombre de la señal de cabecera para ordenar los nombres de señales.
5. Verificar que las señales están siendo enviados a los pines de paquete correcto.

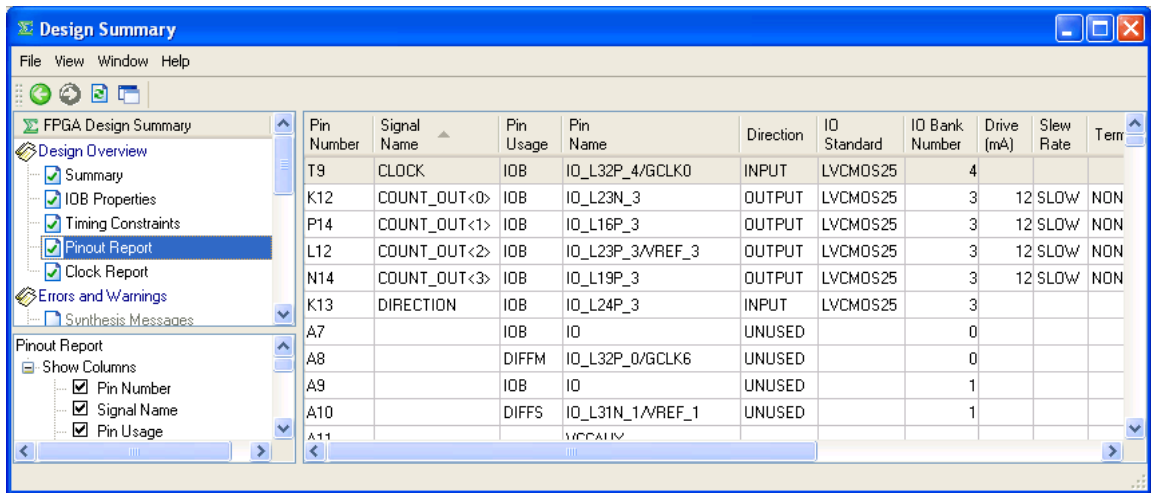


Figura 3.48 opciones pinout report

4. Cierre el resumen del diseño.

Transferencia de diseño a la Spartan™ -3 Demo Board

Este es el último paso en el proceso de verificación de diseño. Esta sección proporciona instrucciones sencillas para descargar el diseño en contra de los Spartan-3 Starter Kit de tarjeta de demostración.

1. Conecte el cable de alimentación de 5V DC a la entrada de energía en el tablero de demostración (J4).
2. Conecte el cable de descarga entre el PC y tarjeta de demostración (J7).
3. Seleccione Síntesis / Aplicación de la lista desplegable en la ventana de Fuentes.
4. La lucha contra Seleccione en la ventana de Fuentes.

3.15 Transferencia de diseño a la Spartan™ -3 Demo Board

Este es el último paso en el proceso de verificación de diseño. Esta sección proporciona instrucciones sencillas para descargar el diseño en contra de los Spartan-3 Starter Kit de tarjeta de demostración.

1. Conecte el cable de alimentación de 5V DC a la entrada de energía en el tablero de demostración (J4).
2. Conecte el cable de descarga entre el PC y tarjeta de demostración (J7).

3. Seleccione Síntesis / Aplicación de la lista desplegable en la ventana de Fuentes.
4. La lucha contra Seleccione en la ventana de Fuentes
5. En la ventana Procesos, haga clic en el signo "+" para expandir la programación de los procesos de generar el archivo
6. Haga doble click en proceso de configurar el dispositivo (IMPACT).
7. El cuadro de diálogo de WebTalk Xilinx pueden abrir durante este proceso. Haga clic en declive.
8. Seleccione Desactivar la recopilación de estadísticas de uso de dispositivo para este proyecto sólo y haga clic en OK.

Impact se abre y el cuadro de diálogo Configuración de Dispositivos de la pantalla.

9. En el cuadro de diálogo de bienvenida, seleccione Configurar dispositivos mediante Boundary-Scan (JTAG).
10. Compruebe que conecte automáticamente a un cable de Fronteras e identificar la cadena de exploración está seleccionado.
11. Haga click en Finalizar.
12. Si usted recibe un mensaje diciendo que hay dos dispositivos encontrados, haga clic en OK para continuar.

Los dispositivos conectados a la cadena de JTAG en el consejo será detectado y se muestran en la ventana de IMPACT

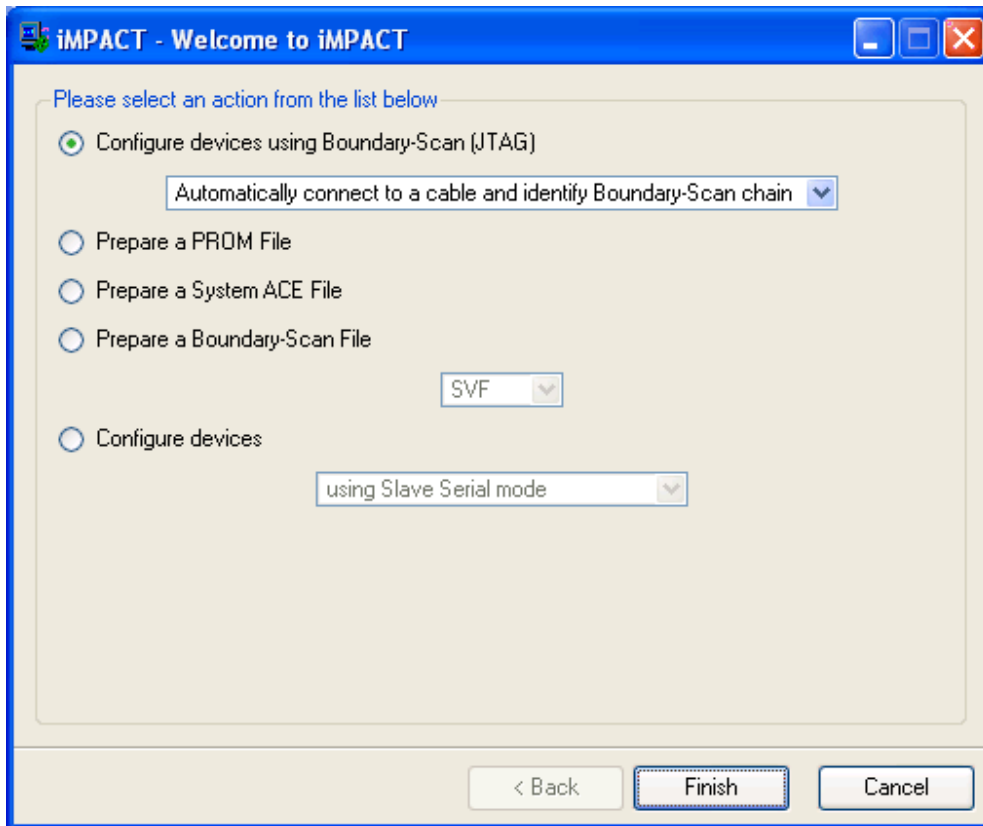


Figura 3.49 Welcome to Impact

13. Asignar la nueva configuración del archivo cuadro de diálogo. Para asignar un archivo de configuración para el dispositivo de XC3S200 en la cadena de JTAG, seleccione el archivo counter.bit y haga click en Open.

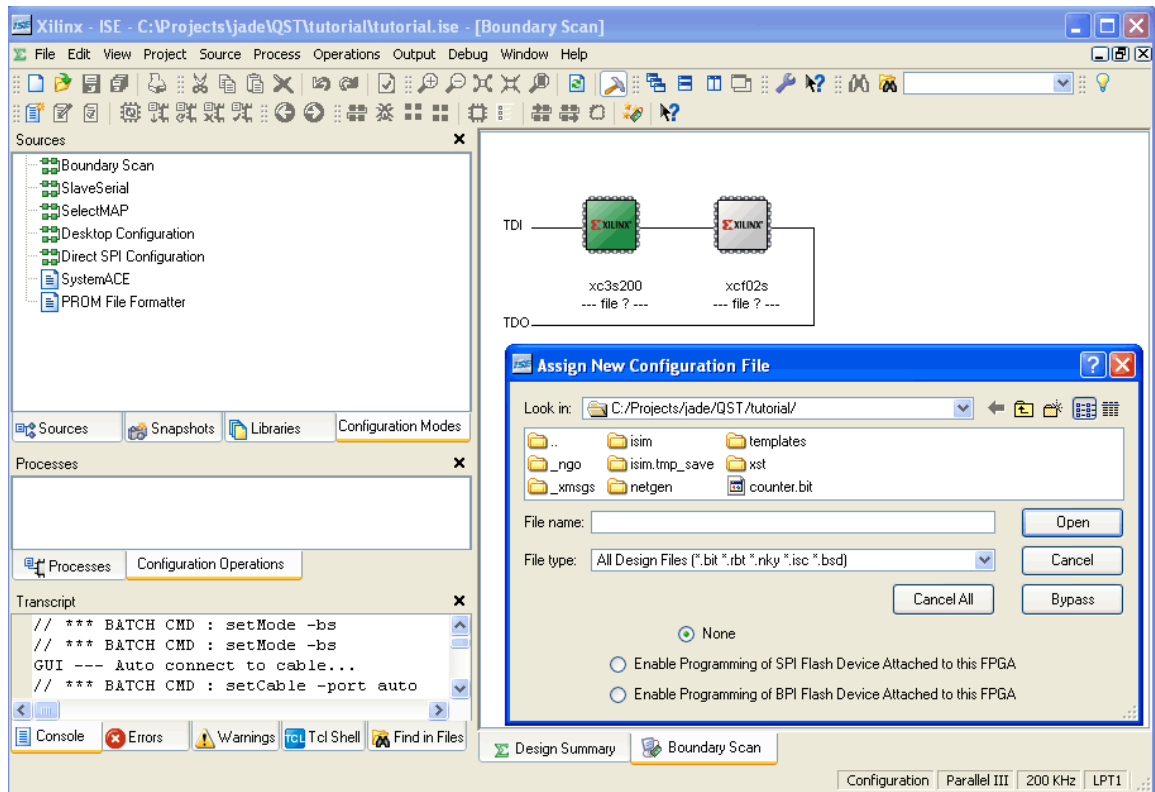


Figura 3.50 Boundary Scan

14. Si usted recibe un mensaje de advertencia, haga clic en Aceptar.

15. Seleccione Bypass para saltar todos los dispositivos restantes.

16. Haga clic en la imagen de la XC3S200 dispositivo, y seleccione Program ...

El cuadro de diálogo Propiedades de programación.

17. Haga clic en Aceptar para programar el dispositivo.

Cuando haya finalizado la programación, el mensaje del programa tuvo éxito

EL PROGRAMA TUVO EXITO

En la placa, los LEDs 0, 1, 2 y 3 están iluminados, lo que indica que el contador está en marcha.

18. Cerrar impacto sin guardar.

Ha completado el ISE Tutorial de inicio rápido. Para una explicación en profundidad de la ISE herramientas de diseño, consulte el ISE In-Depth Tutorial en el sitio web de Xilinx ® en: <http://www.xilinx.com/support/techsup/tutorials/>

**CAPITULO IV: PRÁCTICAS DE LABORATORIO DE FPGAs
EN XILINX**

***Practica 1: Flujo de Xilinx
Herramienta de Laboratorio***

KIT DE ARRANQUE OPERACION SPARTAN -3E

4.1 Herramienta de Flujo de Trabajo Xilinx

4.1.1 Introducción

Este laboratorio ofrece una introducción básica a las herramientas de software ISE. Tendrá que rellenar y poner en práctica un diseño existente. Serán utilizados en los laboratorios de todo este taller para ilustrar el flujo de diseño ISE y herramientas de punta diferente.

4.1.2 Objetivos

Después de participar en esta demostración, usted será capaz de:

- ✓ Crear un nuevo proyecto
- Simular un diseño
- Implementar un diseño

4.1.3 Procedimiento

Esta demostración se compone de cuatros pasos principales:

1. Crear un Nuevo Proyecto
2. Añadir un diseño existente
3. Completar el diseño
4. Simular el diseño
5. Implementar el diseño

Debajo de cada instrucción general para un procedimiento determinado, se encuentra el paso de acompañamiento por paso y cifras ilustran los que se detallan para la realización de la instrucción general. Si usted se siente confiado acerca de una instrucción específica, no dude de saltarse el paso por paso y pasar a la instrucción general siguiente en el procedimiento.

Nota: Si usted desea revisar la demostración en un momento posterior, usted puede descargar los archivos desde el sitio del programa en la Universidad de Xilinx

4.1.4 Crear un Nuevo Proyecto

Pasó 1

- ➡ Crear un nuevo proyecto destinado al uso del dispositivo Spartan -3e que se encuentre en la tarjeta del mismo nombre. Especifique su lenguaje de preferencia, VHDL o Verilog, para completar el laboratorio

Launch ISE: Select **Start** → **Programs** → **Xilinx ISE 10.1i** → **Project Navigator**



Algunas ventanas emergentes pueden aparecer con mensajes relacionados con la lectura de un directorio de red o ejecutar WebUpdate. Estos mensajes aparecen porque se están ejecutando las herramientas en los servidores de Toolwire, y pueden ser ignorados. Desestime el pop-ups a seguir.

- ② En el Project Navigator, seleccione **File** → **New Project**

El New Project Wizard se abre (**Figure 4-1.1**)

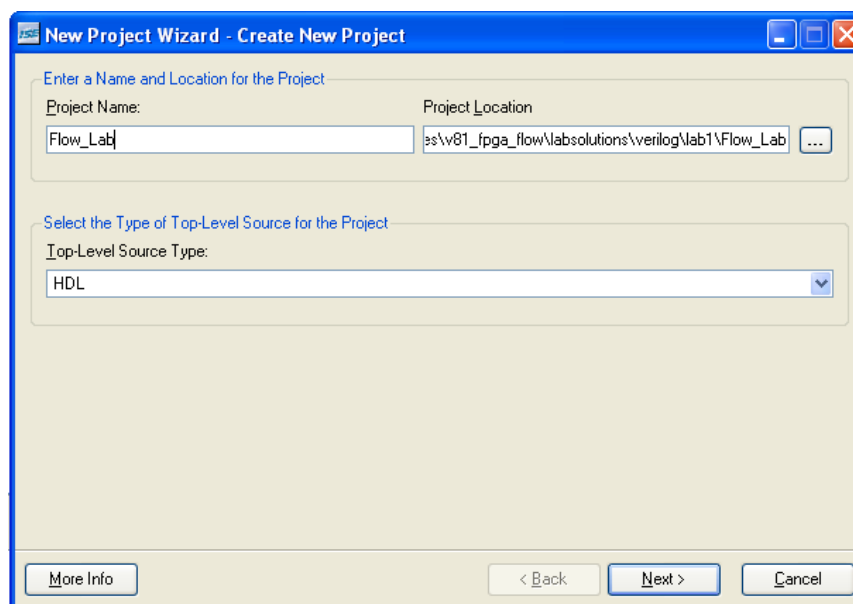


Figure 4.1.1 New Project Wizard

- ③ Para localización de proyecto, usa el botón “...” de examinar uno de las siguientes direcciones, y has click en <OK>
- Verilog users: *c:\xup\fpgaflow\labs\verilog\lab1*
- VHDL users: *c:\xup\fpgaflow\labs\vhdl\lab1*

- ④ Para nombre de proyecto, escribe *Flow_Lab*
- ⑤ Has click en **Next**

El dispositivo y el diseño de flujo de diálogo aparecerá (Figura 4.1.2)

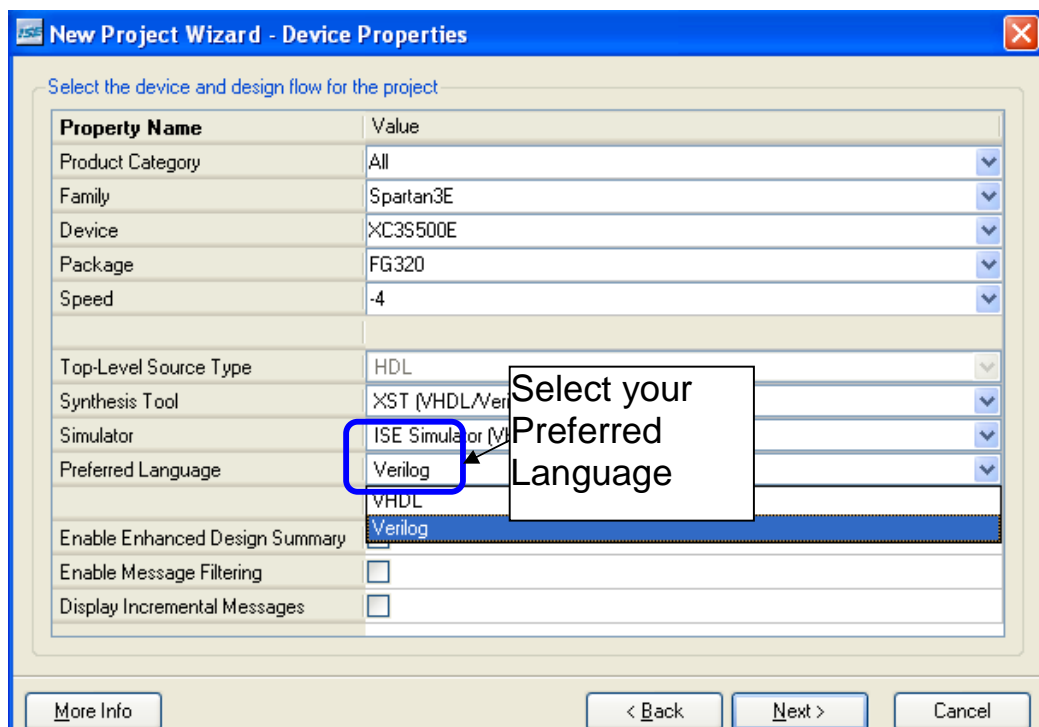


Figura 4.1.2 De dispositivos y de diseño del flujo de diálogo

- ⑥ Seleccione las opciones y haga click en siguiente:
 - ✓ Device Family: **Spartan3E**
 - ✓ Device: **xc3s500E**
 - ✓ Package: **fg320**
 - ✓ Speed Grade: **-4**
 - ✓ Synthesis Tool: **XST (VHDL/Verilog)**
 - ✓ Simulator: **ISE Simulator (VHDL/Verilog)**
 - ✓ Preferred Language: **Verilog** or **VHDL** (select your preference)

El cuadro de diálogo **Crear nueva fuente (Figura 4-1.3)**. Puede usar este diálogo para crear un nuevo archivo fuente de HDL por definir el nombre del módulo y los puertos. Todos los archivos de origen se han creado para usted en este proyecto, por lo que no creará un nuevo fichero fuente aquí.

La pantalla de Fuentes existente añadidas aparecerá (**Figura 4-1.4**).

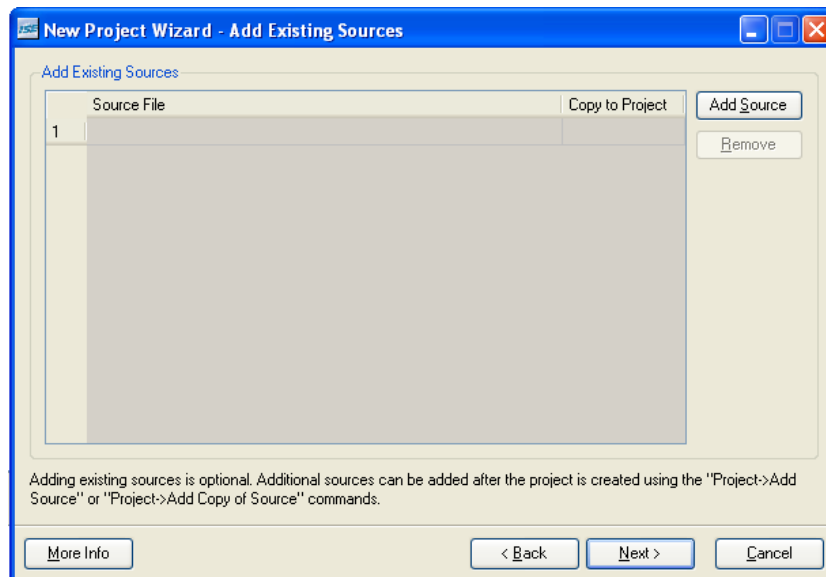


Figura 4-1.4. Cuadro de Fuentes añadidas

4.1.5 Añadir un diseño ya existente en el Proyecto

Paso 2



Añadir archivos de origen de HDL para un diseño PicoBlaze ejemplo. Usted puede revisar la documentación PicoBlaze para familiarizarse con la arquitectura de microcontrolador de 8 bits y assembler Consulte el KCPSM3_manual. Pdf.

Haga clic en

- ➊ Click **Add Source** y buscar en la carpeta `c:\xup\fpgaflow\KCPSM3\VHDL o Verilog` folder
- ➋ Seleccione los archivos VHDL/Verilog files `kcpsm3_int_test` y `kcpsm3` y haga clic en **Open**.
- ➌ Click **<Next>** leaving a check mark in each box for the copy to project option. Click **Finish**.

El dialogo debería aparecer debajo de lo que le permite seleccionar un flujo (none, implementation, simulation, o ambos) asociados con cada archivo origen.

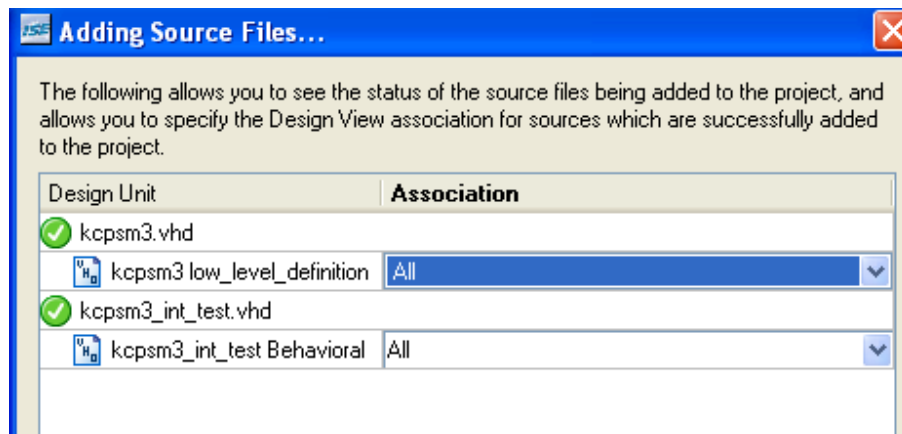


Figura 4-1.5. Elija Tipo de fuente

- ➍ Click **<OK>** aceptando la configuración por Default para todo (ALL) para ambos archivos de fuente.

Nota: Usted debe ver a un módulo llamado `int_test` enumerados en la vista de la jerarquía con un signo de interrogación rojo. Este módulo es un BlockRAM que

contendrá las instrucciones para el controlador de PicoBlaze, que se añadirá en un paso posterior.

4.1.6 Complete el Diseño

Paso 3



Un archivo ejemplo en Assembler (. PSM) archivo llamado `init_test.psm` se incluye con la distribución PicoBlaze. Seleccione este archivo para generar las instrucciones de ROM y agregarlo al diseño.

Abrir Windows Explorer y el Assembler esta ubicado en la carpeta KCPSM3

Nota: Los archivos en assembler `KCPSM3.exe` y `ROM_form *` archivos de plantilla, junto con dos archivos PSM ejemplo (véase la figura 4.1-6) debe residir en este directorio. Tenga en cuenta que los archivos de salida anexados se generarán en el directorio que contiene el ensamblador y los archivos de plantilla. Puede ser beneficioso para copiar el ensamblador y archivos de plantilla en el directorio del proyecto. Para el taller, vamos a mantener los archivos en la ubicación actual.

Name	Size	Type
KCPSM3.EXE	88 KB	Application
ROM_form.coe	1 KB	COE File
cleanup.bat	1 KB	MS-DOS Batch File
int_test.psm	2 KB	PSM File
unlock.psm	58 KB	PSM File
ROM_form.v	15 KB	V File
ROM_form.vhd	13 KB	VHD File

Figura 4-6. Ensamblador de archivos PicoBlaze

Figura 4-6 directorio del Assembler

② . Abra el archivo `Int_test psm` utilizando un editor de texto estándar como WordPad, y revise el código, refiriéndose a `PicoBlaze Embedded Microcontrolador 8-Guía del usuario` o `KCPSM3 manual de orientación técnica`. Estos documentos se proporcionan en el sub-directorio de documentos

③ Abra una ventana de comandos desde **Inicio** → **Programa** → **Accesorios** **Programs** → **Simbolo del Sistema (CMD)**

- ④ Busque en el directorio de **ASSEMBLER** usando el comando **CD**
 > cd c:\xup\fpagaflow\KCPSM3\Assembler
- ⑤ Genere los archivos de definición ROM de ensamble por medio del ejemplo de la aplicación. Ingrese el siguientes comando en el símbolo del sistema
 > kcpsm3 int_test.psm

Nota: Usted debe ahora ver varios archivos en el sub-directorio Assembler a partir de `int_test *`, incluyendo los VHDL (`int_test.vhd`) y Verilog (`int_test.v`) ROM

- ⑥ Verilog o VHDL definición ROM archivo Añadir la en el **proyecto**. Agregar **copia de Origen** y Ir al Proyecto seleccione el archivo `int_test.vhd` o `int_test.v` (Figura 4-1.7).

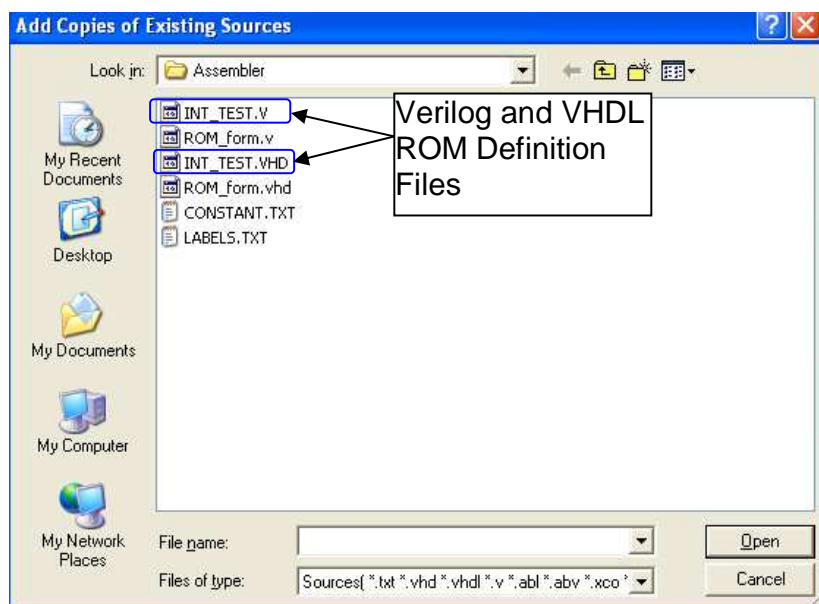


Figure 4-1.7. Add int_test HDL program ROM File to project
Figura 4-1.7. VHDL y Verilog ROM Definicion de Archivos

- 7 Click **Open** y luego **OK** para añadir **INIT_TEST** como un archivo VHDL/Verilog designado para el proyecto (figure 4-1. 8).

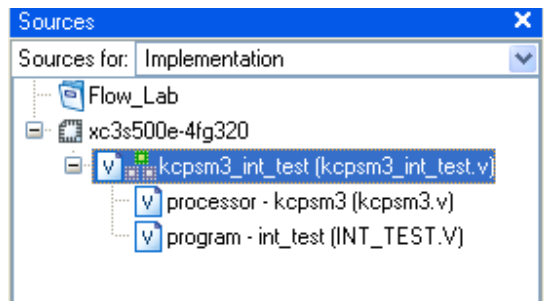


Figura 4-1.8. Vista jerárquica de diseño PicoBlaze

Nota: el nivel superior del archivo **kcp3m3_int_test** contiene un ejemplar del archivo de definición de la ROM **int_test**. Después de añadir este código fuente para **int_test**, el signo de interrogación rojo en la vista del módulo va a desaparecer, ya que ya no es visto como una caja negra

Simulación y Diseño

Step 4



Agregue el banco de pruebas y revisar el código. Ejecutar una simulación del comportamiento mediante el simulador de Xilinx ISIM y analizar los resultados.

- 1 Ir a **Proyecto** → **ADD COPY OF SOURCE** y buscar `c:\xup\fpgaflow\KCPSM3\vhdl` (o verilog)
- 2 Seleccionar el archivo testbench (**test_bench.vhd** o **testbench.v**) y hacer click en <Open>
- 3 Localizar la asociación para **Simular** y click <OK> para añadir el banco de prueba al proyecto

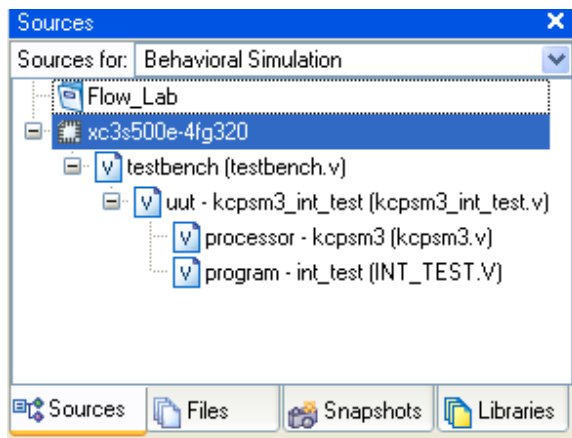


Figura 4-1.9. Ver jerárquica incluyendo la Test Bench

④ Abrir la ventana **ISE Simulator Properties** . Hacer Click en testbench si se encuentra destacado, expandir el Xilinx ISE Simulator toolbox en la ventana de procesos, hacienda click derecho en **Simulate Behavioral Model**, y seleccionar propiedades

⑤ Ingrese el valor de 25000 en **Simulation Run Time** y click <OK>

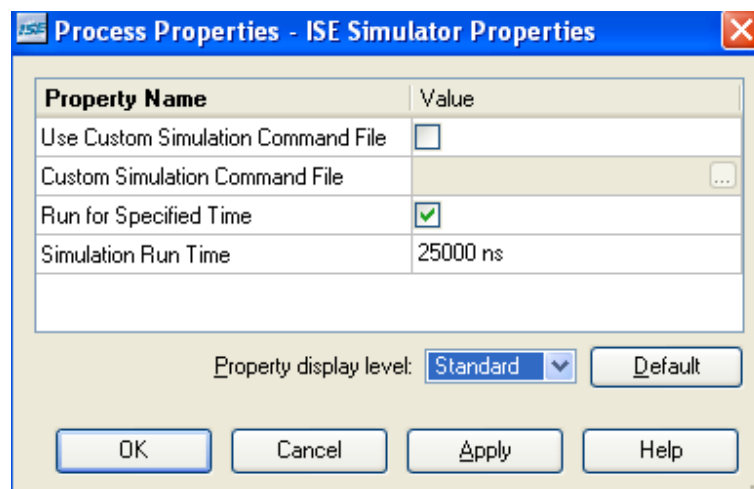


Figura 4-1.10. iSIM Simulación de Propiedades

⑥ Doble Click en **Simulate Behavioral Model** para simular comportamiento. (figure 4-1.11).

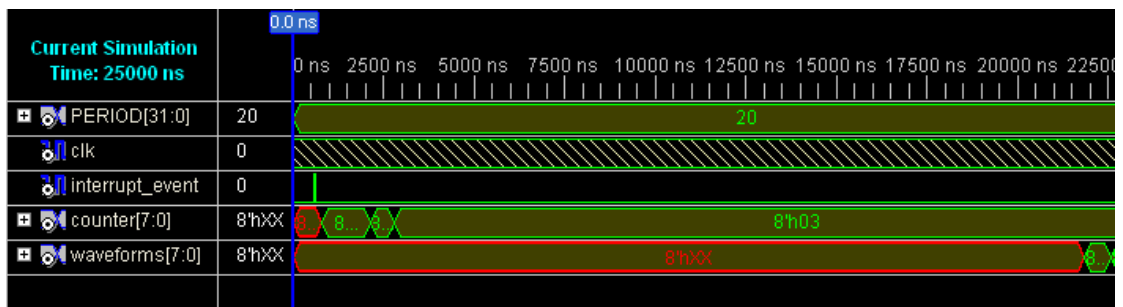


Figura 4-1.11. iSIM Resultado de la Simulación de Propiedades

Los pasos siguientes son sólo para fines ilustrativos, y mostrar cómo analizar las señales internas del diseño. El primer paso se muestra cómo agregar señales internas a la forma de onda. En el segundo paso se muestra como analizar el proceso de interrupción. El tercer paso se muestra cómo analizar el proceso de forma de onda de salida. Si lo desea, puede realizar estos pasos si tiene tiempo adicional al final del laboratorio.

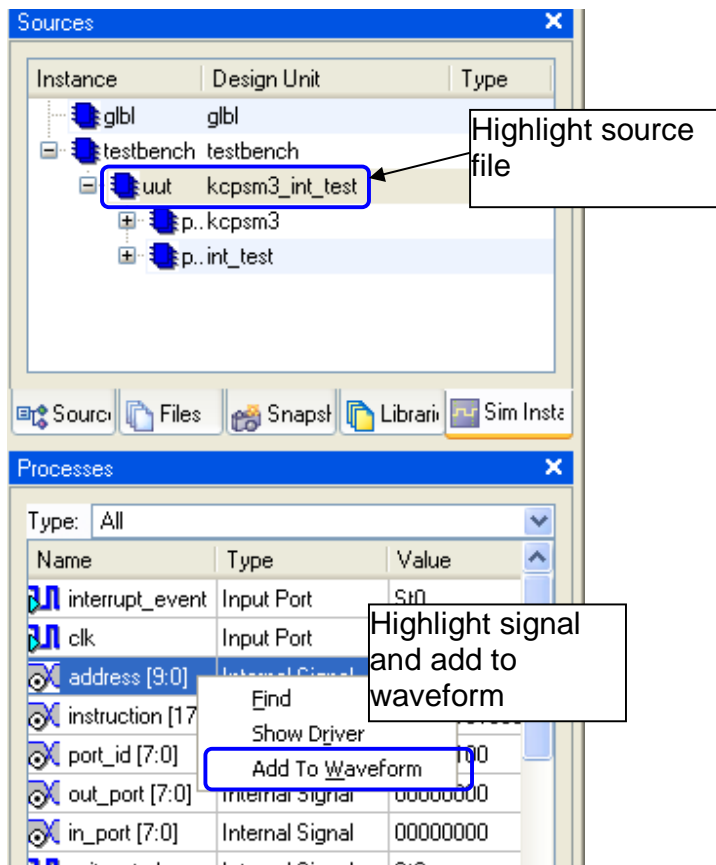
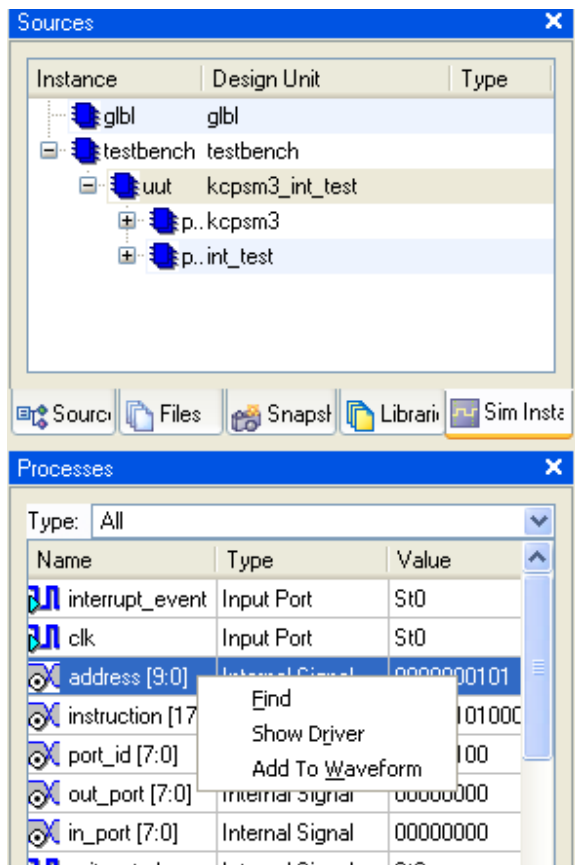


Figure 4-1.12. Accesando Señales Internas



② Re-simulado el diseño y analice la rutina de interrupción (figura 4-1.13).

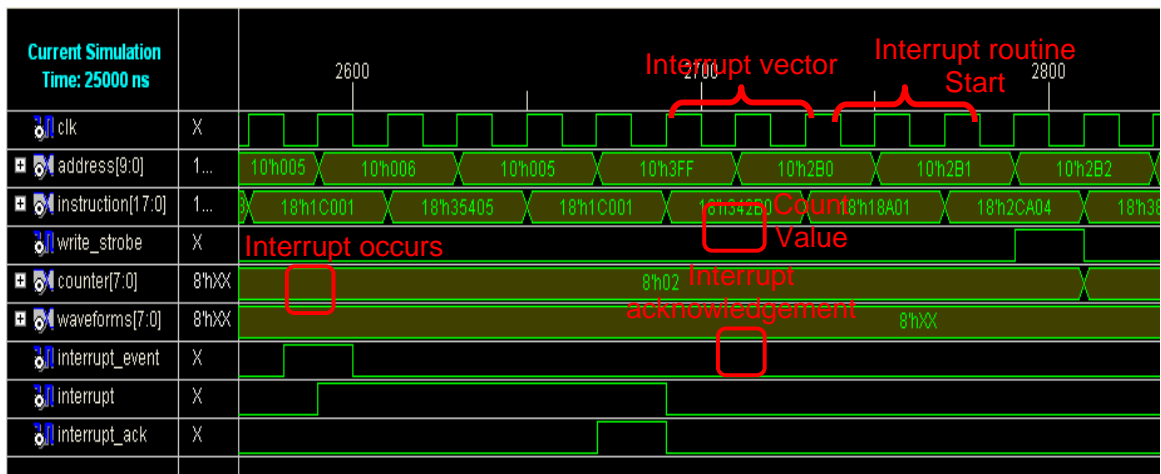


Figure 4-1.13 Interruptor de Servicio de Rutina

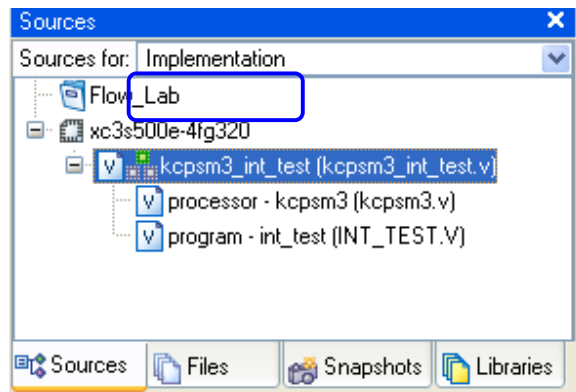


Figura 4-1.15. Ventana de Panel de Fuente

- ② En la ventana de proceso de fuente haga doble click en, **Implement Design** (Figura 4-1.16)

Tenga en cuenta que las herramientas de ejecutar todos los procesos necesarios para aplicar el diseño. En este caso, las herramientas de ejecución de síntesis antes de entrar en esta aplicación.

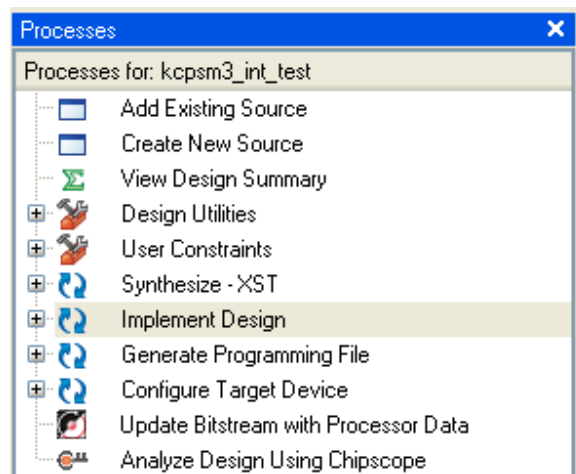


Figura 4-1.16. Procedimiento en Ventana de Fuente

- ③ Si bien la aplicación se esta ejecutando, haga click en el signo + next to **Implement Design** para aplicar la etapa de aplicación y ver el progreso. Nos referimos a esto Como la expansión de un proceso.

Después de cada etapa, aparecerá un símbolo a lado de cada etapa:

- ✓ Marca de Verificación para el éxito.
- ✓ Exclamación de puntos de advertencia

- 4 Lea algunos de los mensajes de la ventana de mensajes en la ventana de mensajes ubicados en la parte inferior de la ventana Project Navigator.
- 5 Cuando la aplicación este completa, la revisión de la utilización del diseño en la ventana de resumen de Diseño. (Figura 4-1.18).

FLOW_LAB Project Status			
Project File:	Flow_Lab.isc	Current State:	Placed and Routed
Module Name:	kcpsm3_int_test	Errors:	No Errors
Target Device:	xc3s500e-4fg320	Warnings:	345 Warnings
Product Version:	ISE, 8.1.03i	Updated:	Mon May 15 09:43:08 2006

Device Utilization Summary				
Logic Utilization	Used	Available	Utilization	Note(s)
Number of Slice Flip Flops	76	9,312	1%	
Number of 4 input LUTs	107	9,312	1%	
Logic Distribution				
Number of occupied Slices	99	4,656	2%	
Number of Slices containing only related logic	99	99	100%	
Number of Slices containing unrelated logic	0	99	0%	
Total Number 4 input LUTs	177	9,312	1%	
Number used as logic	107			
Number used as a route-thru	2			
Number used for Dual Port RAMs	16			
Number used for 32x1 RAMs	52			
Number of bonded IOBs	18	232	7%	
IOB Flip Flops	16			
Number of Block RAMs	1	20	5%	
Number of GCLKs	1	24	4%	
Total equivalent gate count for design	74,954			
Additional JTAG gate count for IOBs	864			

Performance Summary			
Final Timing Score:	0	Pinout Data:	Pinout Report
Routing Results:	All Signals Completely Routed	Clock Data:	Clock Report
Timing Constraints:	All Constraints Met		

Detailed Reports					
Report Name	Status	Generated	Errors	Warnings	Infos
Synthesis Report	Current	Mon May 15 09:42:22 2006	0	345 Warnings	0
Translation Report	Current	Mon May 15 09:42:28 2006	0	1 Warning	0
Map Report	Current	Mon May 15 09:42:38 2006	0	0	2 Infos
Place and Route Report	Current	Mon May 15 09:43:01 2006	0	0	2 Infos

Figure 4-1.19. Design Summary

4.1.8 Conclusión

En esta demostración, completó las grandes etapas del flujo de diseño ISE: la creación de un proyecto, añadió archivos de código fuente, simuló el diseño y la aplicación de diseño.

En el siguiente módulo, se examinarán algunos informes del software, se determinará cómo se llevó a cabo el diseño, y se determinará si se cumplieron sus objetivos de diseño para el área y el rendimiento.

*Práctica 2: Asistente de Arquitectura
y Práctica PACE*

Orientación XUP Spartan-3E

4.2 Asistente de Arquitectura y Práctica PACE

4.2.1 Introducción

La función de Asistente de Arquitectura permite a los diseñadores configurar y agregar recursos FPGA para el diseño. El PACE permite a los diseñadores agregar restricciones a un diseño.

4.2.2 Objetivos

Después de completar este laboratorio, usted será capaz de:

- ✓ Usar el Asistente de Arquitectura para configurar y crear una instancia de un componente de DCMU
- ✓ Utilice PACE para asignar lugares pin
- ✓ Implementar el diseño y confirmar que se han utilizado las asignaciones de pin
- ✓ Descargar y probar diseños en Hardware

4.2.3 Procedimientos

Esta demostración se compone de cuatro pasos primarios:

1. Configurar un DCM
2. Instanciar el DCM
3. Asignar localización de pines
4. Probar el diseño en hardware

Debajo de cada instrucción general para un procedimiento determinado, se encuentra el paso de acompañamiento-por paso y cifras ilustran los que se detallan para la realización de la instrucción general. Si usted se siente confiado acerca de una instrucción específica, no dude en saltarse el paso por paso y pasar a la instrucción general siguiente en el procedimiento.

Nota: Si usted desea revisar esta demostración de software en un momento posterior, puede descargar los archivos desde el sitio del Programa en la Universidad de Xilinx en <http://www.xilinx.com/univ>

4.2.4 Diseño General

Este laboratorio va a hacer uso de UART Real-Time clock y la información detallada se puede encontrar en la UART_real_time_clock.pdf suministrados con la distribución PicoBlaze. En esta sección se tratará de organizar y poner de relieve la función del diseño.

El diseño implementa un reloj de real-time -el mantenimiento de tiempo en horas, minutos y segundos, junto con la capacidad de establecer una alarma. La particularidad del diseño es que una comunicación en serie UART se utiliza para definir y observar la hora / alarma enviando comandos simples de texto y mensajes a través de una utilidad como HyperTerminal.

El diseño comprende algunos comandos simples ASCII e incluso soporta algunas modificaciones en su lista, usando la tecla de retroceso de su teclado. Un comando es sólo en la fecha que se introduce un retorno de carro. El diseño está preparado para aceptar un comando cuando el "KCPSM3>" del sistema se muestra.

El "uclock" programa proporcionado con la distribución es capaz de interpretar caracteres en mayúsculas y minúsculas mediante la conversión de comandos (consulte la documentación para más detalles) a mayúsculas antes de analizarlos. Comandos

incorrectos dará lugar a un "error de sintaxis" mensaje y los valores de hora incorrecta será indicado por un "mensaje no válido Time". Aunque es poco probable que ocurra cuando se utiliza HyperTerminal, un "error de desbordamiento de" mensaje será generado si los comandos se transmiten más rápido que el diseño puede procesarlas (es decir, de búfer receptor UART se llena).

El diseño requiere de un reloj de 55 MHz. Desde el Spartan-3E Junta incluye un oscilador de 50 MHz, se utiliza el Asistente de Arquitectura para generar un DCM con salida de 55 MHz y una instancia que en el diseño.

4.2.5 Configurar un DCM

Paso 1

Esta versión del diseño es para activar el componente DCM. Utilice el Asistente de Arquitectura para configurar un componente DCM para la producción de un reloj de 55 MHz.

- ❶ Seleccione **File** → **Open Project** y abrir el lab 2 project
 - Verilog usuarios: *c:\xup\fpgaflow\labs\veriloglab2\arwz_pace*
 - VHDL usuarios: *c:\xup\fpgaflow\labs\vhdl\lab2\arwz_pace*
- ❷ En la ventana de procesos de fuentes, haga doble click en **Create New Source**

Si usted no ve el proceso de Create New Source process, asegurese de que una fuente HDL será seleccionada en the Sources in Project window.
- ❸ En la nueva ventana de fuente, seleccione **IP (CoreGen & Architecture Wizard)** e ingrese *my_dcm* como nombre de archivo y haga click en *Next*
- ❹ En la selección de tipo de núcleo, expanda **FPGA Features and Design** → **Clocking** → **Spartan-3E, Spartan-3A** y seleccione **Single DCM SP v9.1i (Figura 2-1)**

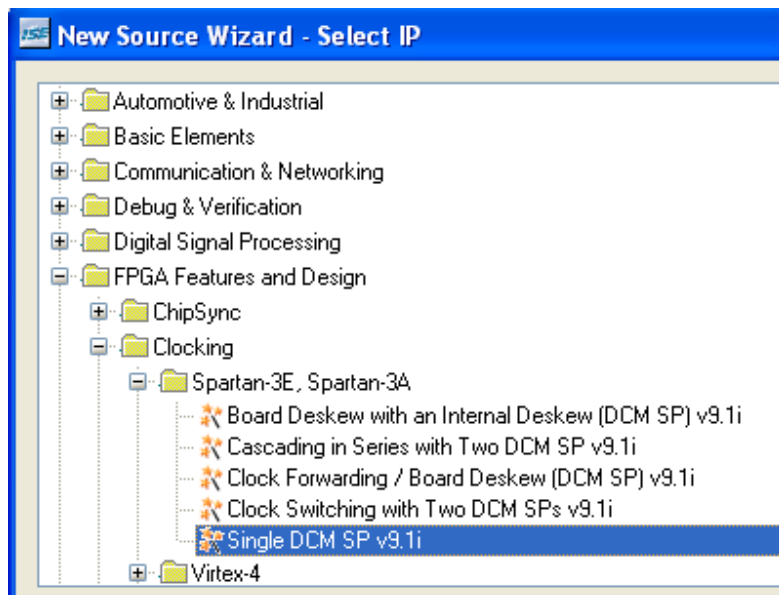


Figura 4.2-1. Asistente de Arquitectura caja de selección

- ④ Haga click en **Next**, y click en **Finish**
- ⑥ En el **Xilinx Clocking Wizard** – ventana **General Setup**, escriba las siguientes opciones (como se muestra en la **Figura 4.2-2**), y haga click en **<Next>** para continuar.
 - ✓ CLK0, CLKFX and LOCKED boxes: **revisado**
 - ✓ RST box: sin revisar
 - ✓ Entrada de Frecuencia de Reloj: 50 MHz

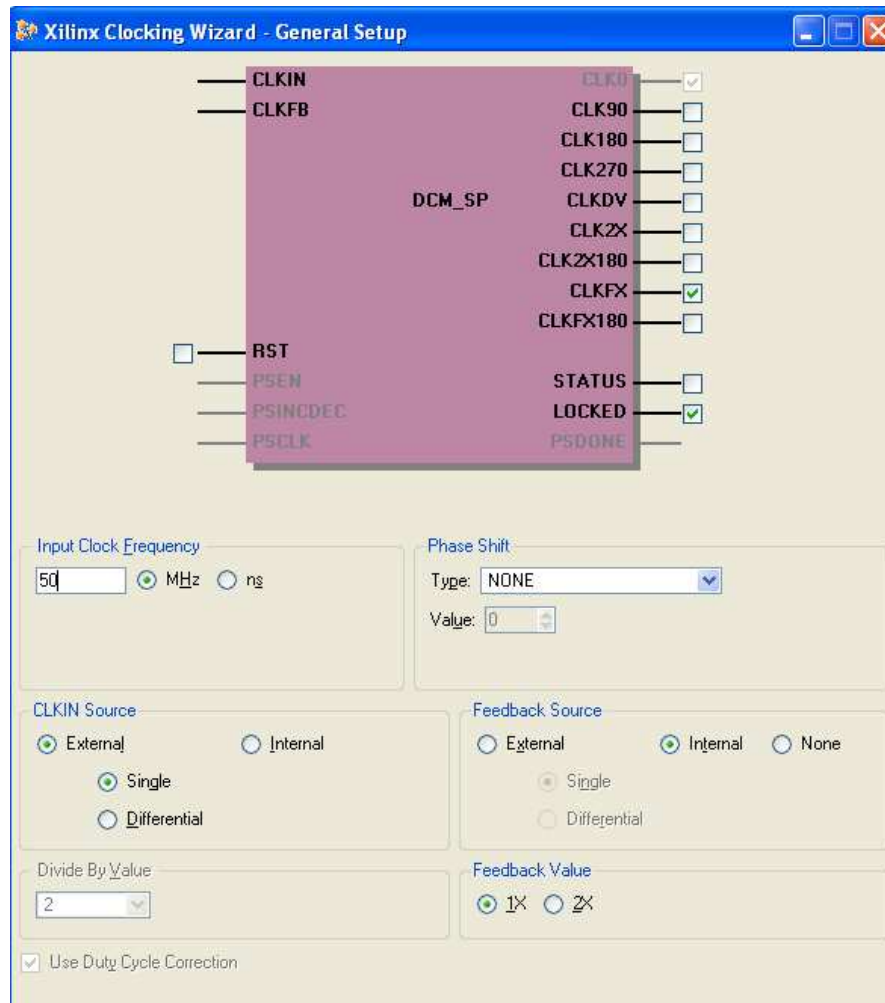


Figura 4.2-2. Xilinx Clocking Wizard – General Setup Window

- ➊ En el **Xilinx Clocking Wizard – ventana Clock Buffers** (Figura 4.2-3), mantenga sin cambio y haga click en **<Next>**

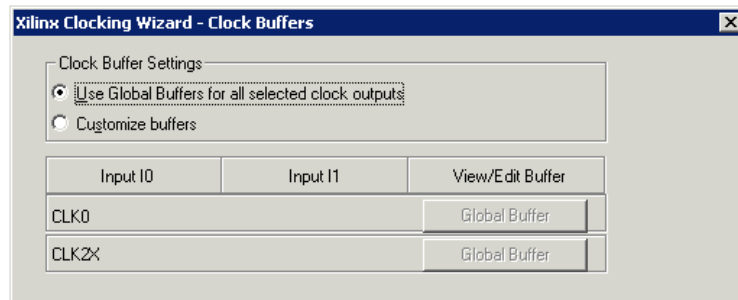


Figura 4. 2-3. Xilinx Clocking Wizard – Clock Buffers Window

⑧ En el **Xilinx Clocking Wizard – Clocking Frequency Synthesizer** de diálogo, introduzca 55 MHz como la frecuencia de salida, y haga clic en Calculate para determinar la M (multiplicar) y D (valores brecha) que se utilizan para calcular la frecuencia de salida.

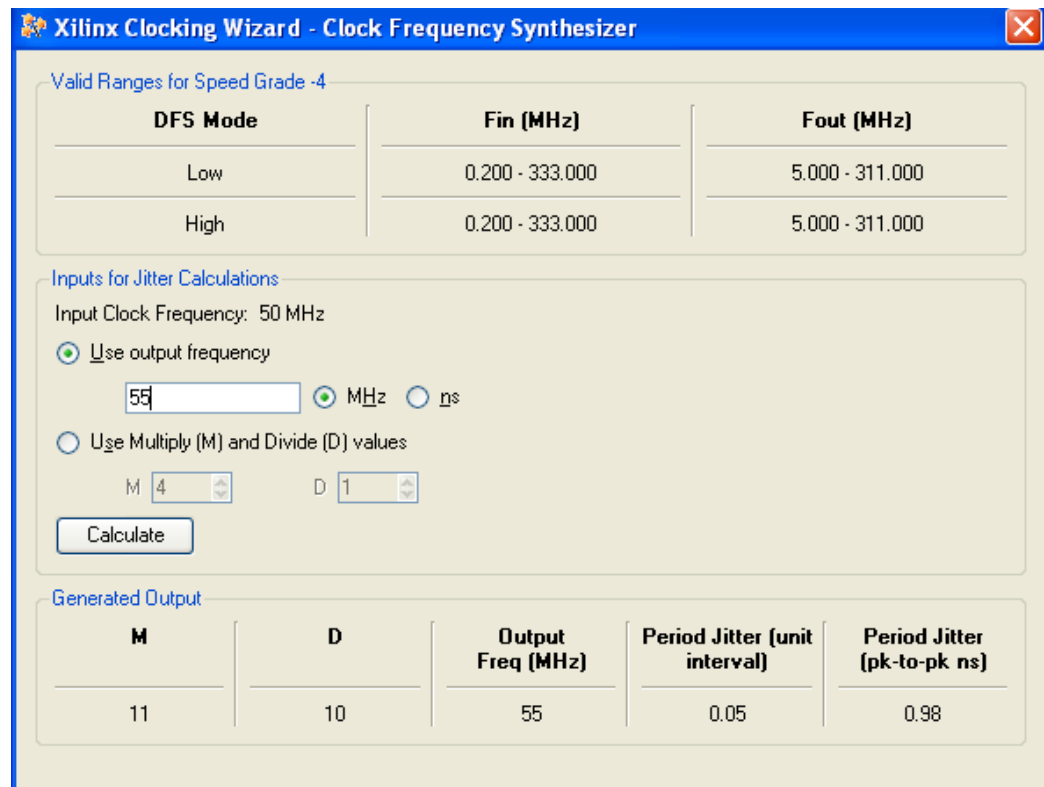


Figura 4.2-4. Especificaciones de la salida de frecuencia del DCM

- ➊ Haga Click en <Next> y luego en <Finish>.

Tenga en cuenta que un nuevo archivo (*my_dcm.xaw*) se añade como una fuente en el proyecto (Figura 4.2-5). Esta fuente de archivo no se incluirá en la jerarquía de diseño hasta que el componente ha sido una instancia en uno de los archivos fuente de HDL. Para ello, haga en el próximo paso.

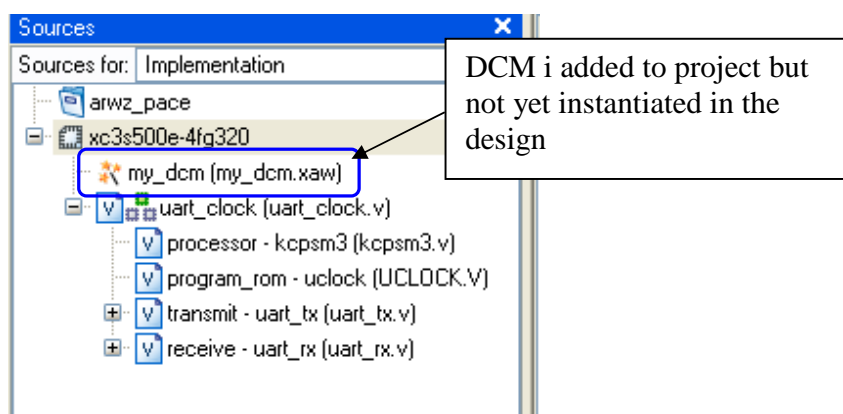


Figura 4.2-5. Enumerados de jerarquía del DCM

4.2.6 Una instancia de la DCM en un diseño Verilog Paso 2a

Usuarios VHDL pueden saltarse a la siguiente sección ...Step 2b



Ahora que ha creado los archivos necesarios, creará una instancia del componente de DCM en su diseño. Usted debe copiar y pegar el texto de la plantilla de instancias en *uart_clock.v* y conectar las señales.

- ➋ En la ventana de Sources de proyecto, haga doble-click en *uart_clock.v* para abrir esta ventana de fuentes de códigos en el editor de texto
- ➌ Seleccione *my_dcm.xaw* en la ventana de Sources del proyecto
- ➍ En los procedimientos de Fuentes, haga doble-click en **View HDL Instantiation Template** para abrir la creación de instancias de plantilla en el editor de texto

- ④ En la Instantiation Template *my_dcm.tfi*, copie la **instantiation module** y pegue en *uart_clock.v* bajo el comentario *// Insert DCM component here*
- ⑤ Complete la instancia llenando el Puerto de conexión a seguirse:

```
my_dcm inst_my_dcm (
```

- ✓ CLKIN_IN(clk),
- ✓ CLKFX_OUT(clk55MHz),
- ✓ CLKIN_IBUFG_OUT(),
- ✓ CLK0_OUT(),
- ✓ LOCKED_OUT(lock)

Nota: El Puerto *clkin_ibufg_out* es una salida que esta para soportar los diseños que usen RocketIO™ transceptores. El dispositivo Spartan-3E no contiene transceptores, este Puerto se conectara a una señal ficticia

- ✓ Añada una declaración de señal para los 55 MHz de salida del DCM bajo el comentario *// Signals for DCM*, a seguir:

```
Wire clk55MHz;
```

nota: Esta **marca** de pin de salida maneja el Led 1 en el tablero del Spartan-3, y será manejado por la señal marcada en el DCM. Esto indicara a los usuarios que el DCM ha sido exitoso

- ① Marcado en la señal de reloj de 50 MHz desde el oscilador.
- ② Haga Click en **File** → **File** → **Save** to save the file
- ③ Tómese en cuenta que la fuente del archivo *my_dcm.xaw* este insertado en la ubicación correcta del diseño de jerarquía.

4.2.7 Una instancia de la DCM en un diseño Vhdl Paso 2b

Ahora que ha creado los archivos necesarios, puede crear una instancia del componente de DCM en su diseño. Copiar y pegar el texto de la plantilla de instancias en `uart_clock.vhd` y conectar la señal.

- ❶ En la ventana Sources in Project, haga doble-click en `uart_clock.vhd` para abrir el código fuente en el editor de texto
- ❷ Seleccione `my_dcm.xaw` en la ventana de Sources in Project
- ❸ En la ventana de Processes for Source , haga doble-click en **View HDL Instantiation Template para abrir la** instantiation template en el editor de texto
- ❹ En la Instantiation Template `my_dcm.vhi`, copie el **component declaration** (**begin at COMPONENT my_dcm** y termina después del **END COMPONENT;**) y pegue en `uart_clock.vhd` debajo del comentario `-- Insert DCM component declaration here`
- ❺ En la HDL Instantiation Template `my_dcm.vhi`, copie la **component instantiation** (**begin at Inst_my_dcm: my_dcm hasta el final del archivo**) y pegue en `uart_clock.vhd` debajo del comentario `-- Insert DCM component instantiation here`
- ❻ Complete la instancia llenando el Puerto de conexiones como se muestra:

```
Inst_my_dcm: my_dcm PORT MAP(  
  CLKIN_IN      => clk,  
  CLKFX_OUT     => clk55MHz,  
  CLKIN_IBUFG_OUT => open,  
  CLK0_OUT      => open,  
  LOCKED_OUT    => lock  
);
```

Nota: El puerto `clkin_ibufg_out` es un Puerto de salida que sirve para soportar los diseños que usan señales de RocketIO™ transceptores. Porque el dispositivo Spartan-3E no contiene transceptores, este Puerto se conectara a una señal falsa.

- ❼ Añada una declaración de señal para los 55 MHz de salida del DCM debajo del comentario `-- Signals for DCM`, que se muestra:

```
signal clk55MHz : std_logic;
```

nota: El diseño `uart_clock.vhd` ha sido actualizado para que todo diseño se procese a tiempo usando la señal `clk 55MHz`.

- ③ Añada un pin de salida para marcar la entidad que se muestra:

```
entity uart_clock is
  Port
    tx : out std_logic;
    rx : in std_logic;
    alarm : out std_logic;
    clk : in std_logic;

    lock : out std_logic

end uart_clock;
```

nota: Esta **marca** de pin de salida guiara el Led 0 en le tablero de la Spartan-3E y que es guiada por la señal marcada en el DCM. Esta indicara a los usuarios de DCM que ha sido exitoso la marca en los 50 MHz de la seña de reloj desde el tablero del oscilador.

- ② Haga Click en **File** → **Save** to save the file
- ② Tómese en cuenta que la fuente del archive e `my_dcm.xaw` ha sido insertado en el lugar correcto por jerarquía.

4.2.8 Asignacion de Ubicación de Pins

Paso 3



En este paso, que va a utilizar PACE para asignar lugares a los pines en el diseño. A continuación, compruebe en el informe de notas que las patillas se han asignado después de ejecutar el lugar y ruta.

- ① En la ventana de Sources in Project seleccione el nivel alto del archive de diseño `uart_clock.vhd/v`
- ② En la ventana de Processes , expanda **User Constraints** y haga doble click en **Floorplan IO – Pre-Synthesis para abrir el PACE**

Haga Click en “yes” cuando pregunte para añadir un archivo UCF file en el proyecto.

- ③ Ingrese el pin que contiene en la ventana de **Design Object List** (mire la Figura 4.2-6).

I/O Name	I/O Direction	Loc	Bank	I/O Std.	Vref	Vcco	Drive Str.	Termina	Slew
alarm	Output	e12	BANK0	LVTTL	N/A	3.30	8		SLOW
clk	Input	c9	BANK0	LVCMS533	N/A	3.30			
lock	Output	f12	BANK0	LVTTL	N/A	3.30	8		SLOW
rx	Input	r7	BANK2	LVTTL	N/A	3.30			
tx	Output	m14	BANK1	LVTTL	N/A	3.30	8		SLOW

Figura 2-6. Introduzca el PIN Ubicación Restricciones

La lista de abajo describe operaciones para las señales I/O que acaba de asignar. Esto completa los pines de salida para el kit de Spartan-3E este puede ser encontrado en el manual de usuario

- ✓ clk : connected to 50 MHz oscillator
- ✓ lock : connected to led0
- ✓ alarm : connected to led1
- ✓ rx : connected to pin that receives serial data from Maxim MAX3232
- ✓ tx : connected to pin that transmits serial data to Maxim MAX3232

- ④ Haga Click en save y seleccione **XST Default: <>** como la I/O que limita el BUS. Haga Click en **OK**



Vea la asignación de pins en relación con la lógica interna.

- ① En el dispositivo de la ventana de Arquitectura, agrande hasta que pueda ver los números individuales de los pins (**Figura 4.2-7**)

La barra coloreada alrededor del I/O pins indican que pines están en la misma I/O .así que fácilmente puede intercambiar pines para los mismas entradas.

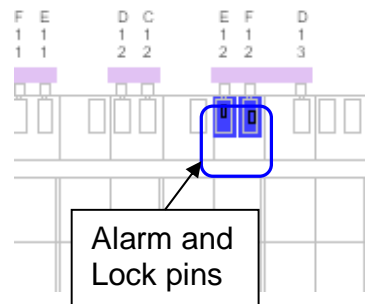


Figura 4.2-7. Ventana de Dispositivo de Arquitectura

- ➊ Haga Click en cada parte coloreada **I/O pin**. El pin correspondiente saldrá en la lista de diseño
- ➋ Haga Click en **File** → **Save** para save los pines asignados
- ➌ Haga Click en **File** → **Exit** para cerrar el PACE
- ➍ Sombree el archivo UCF en el navegador del proyector, expanda **User Constraints** y haga doble-click en **Edit Constraints (Texto)** para observar las restricciones creadas en *uart_clock.ucf* file through PACE. Vea la versión en texto del archivo UCF file para confirmar las restricciones que están escritas en el archivo
- ➎ Seleccione el nivel máximo del archivo de diseño *uart_clock.vhd/.v* en la ventana de Sources in Project
- ➏ En la ventana de Processes for Source expanda el **Implement Design** process, expanda **Place & Route**, y haga doble-click en **Pad Report**

El navegador del proyecto automáticamente determinara que procesos debe correr y abrirá el reporte una vez el lugar y la ruta estén completas.
- ➐ Baje el puntero del mouse en el reporte y confirme que el número de pin de la señal I/O sea igual a la asignada por usted.

4.2.9 Prueba del Diseño en Hardware

Paso 4

Configurar e iniciar una sesión de HyperTerminal. Conectar y alimentar la tarjeta. Generar el flujo de bits y configurar la FPGA. Verificar el funcionamiento del reloj de tiempo real en el hardware.

- ❶ Abra una sesión de hyperterminal session hacienda en **Start → All Programs → Accessories → Communications → HyperTerminal**
- ❷ De nombre a la sesión, de click en <OK>, y especifique un puerto COM (ie. COM1).
- ❸ Haga Click en botón de configuración y especifique los siguientes parámetros en el Puerto de sesión. haga Click en <OK> cuando termine
 - ✓ Velocidad de 38400, 8 bits de datos, n bits de paridad, 1 bit de parada, sin control de flujo

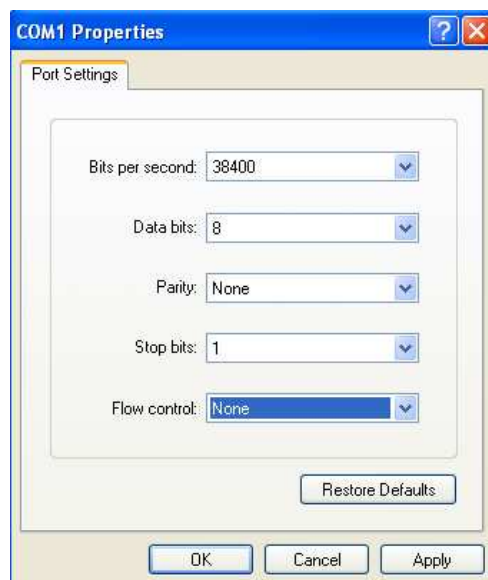


Figura 2-8. Ajustes para comunicaciones por puerto serie

- 4 Haga Click en **Settings**, luego en la ventana **ASCII Setup** y luego click hasta que una marca aparezca seguido por la opción **Append line feeds to incoming line ends**, y luego haga click en <OK>. Click en <OK> de Nuevo para salir de la ventana.

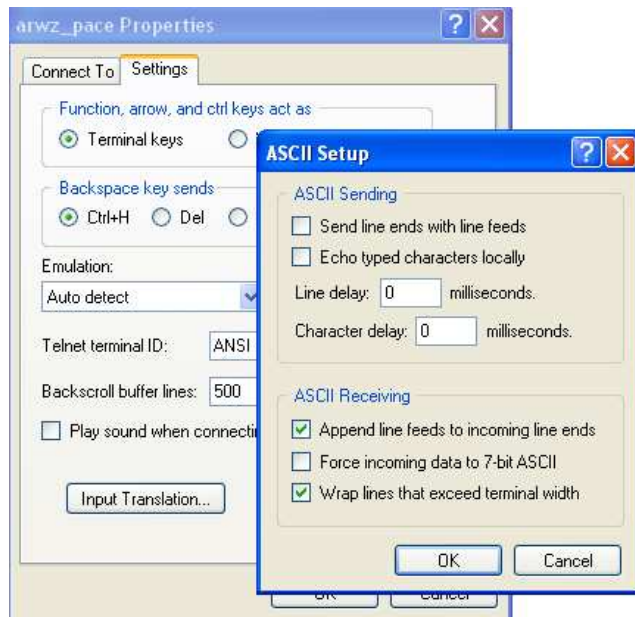


Figura 4.2-9. ASCII Settings for Serial Port Connection

- 5 Conecte los cables (poder, USB, y rs232) y encienda el tablero
- 6 generar el flujo de bits e invocar iMPACT: Highlight **uart_clock.vhd**, expand **Configure Target Device**, and double-click on **Manage Configuration Project (iMPACT)**.
- 7 Cuando el proyecto IMPACT se abre el diálogo, haga clic en cancelar para no crear un proyecto de impacto.
- 8 Haga doble -click en **Boundary Scan** y click en el icono **Initialize Chain** (figura 2-10)

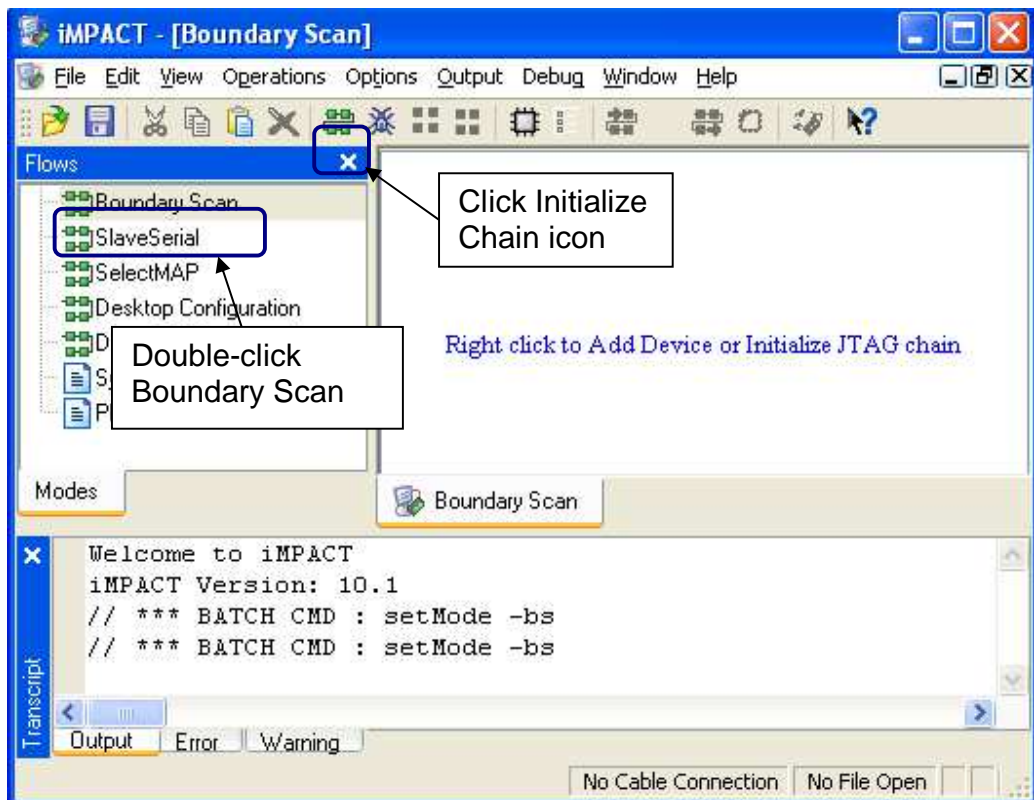


Figura 2-10. Initialize JTAG Chain

- ⑨ Cuando el cuadro de Asignación de nueva configuración del archivo de diálogo aparece, seleccione el archivo `uart_clock.bit` para el dispositivo `xc3s500e` (primero en la cadena JTAG) y haga clic en omitir el resto.
- ⑨ **Nota:** Un mensaje de advertencia que indica que el reloj de inicio se ha cambiado el reloj de JTAG. Haga clic en <OK>.

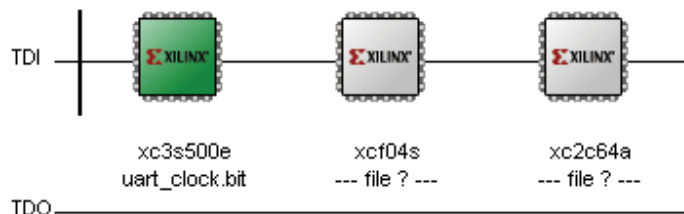


Figura 4.2-11. Cadena JTAG con configuración asignada de archivo

- ⑩ Haga click derecho en la ventana de IMPACTO donde sale xc3s500e , seleccione el programa, y haga click en <OK> en la ventana de **Programming Options** .

Nota: Usted deberá ver el icono de KCPSM3> en la ventana de la hyperterminal .

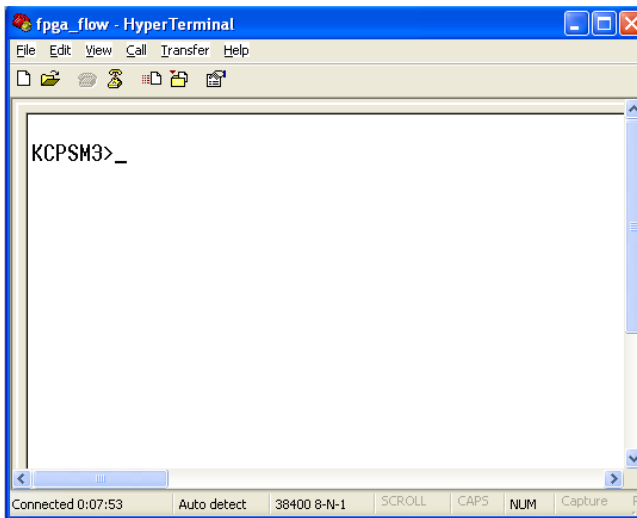


Figura 4.2-12. Comunicación en serie con PicoBlaze

4.2.10 Funcionamiento de la UART-Real Time Clock Paso 6



Usted emitir comandos para operar el real UART-reloj de tiempo, como se especifica en el archivo [UART_real_time_clock.pdf](#).

- ① Ingrese el comando “time” al comando mostrado para ver la hora actual en hh:mm:ss

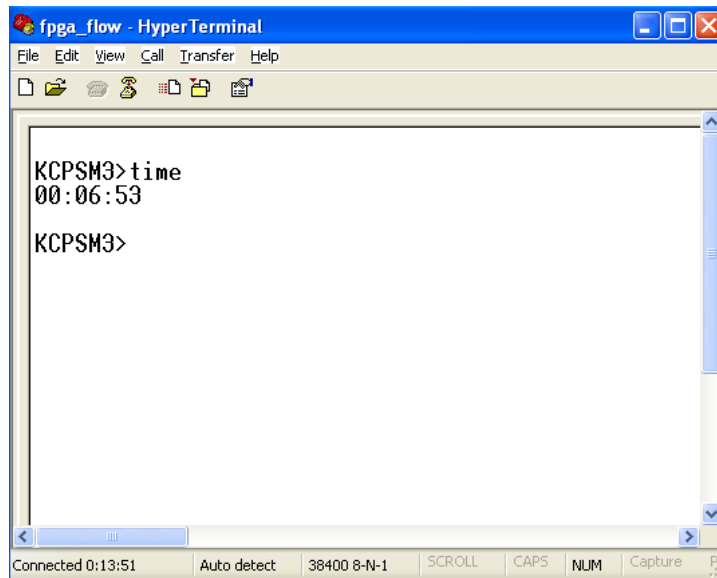


Figura 2-13. Muestra del tiempo actual

- ➊ Ingrese el comando “alarma” en la pantalla de comando para mostrar el tiempo de alarma real en la forma hh:mm:ss

Nota: La alarma esta inactiva

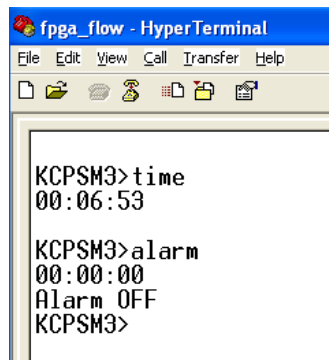


Figura 4.2-14. Muestra del tiempo de alarma y estado

- ➋ Ingrese el comando “alarm on” para que se active la alarma
- ➌ Ingrese el comando “alarm 00:00:30” para fijar la alarma en 30 segundos
- ➍ Ingrese el comando “time 00:00:00” para fijar la hora.

Nota: Usted deberá tener en cuenta que el Led 1 en la pantalla del tablero del Spartan-3E se encenderá una vez la alarma se desactive.

- ⑥ Ingrese el comando “alarm off” para apagar la alarma

Nota: Usted deberá tener en cuenta que si el Led 1 se enciende indica que la alarma ha sido desactivada.

4.2.11 Conclusión

En esta demostración, que utilizó el Asistente de Arquitectura (Wizard) para configurar un componente de DCM. Usted completó el diseño de una instancia del componente en el diseño y la asignación de lugares de PIN con PACE. Por último, se probó el diseño de hardware mediante la descarga del flujo de bits del sistema IMPACT y la introducción de órdenes de reloj de alarma a través de HyperTerminal

Práctica 3: Global Timing Constraints Lab

Orientación de la Spartan-3E Starter Kit

Calendario Global de Restricciones de Laboratorio

4.3.1 INTRODUCCIÓN

En esta práctica, deberá especificar sus requisitos momento introduciendo limitaciones de tiempo global, y luego analizar el rendimiento del diseño utilizando los informes de los tiempos de la. Usted completará un diseño PicoBlaze, simular, y lo prueba en hardware.

4.3.2 OBJETIVOS

Después de completar este laboratorio, usted será capaz de:

- ✓ Introduzca restricciones temporales globales usando el Editor de restricciones Xilinx
- ✓ Revisión, si las restricciones temporales son realistas utilizando el Post-Mapa de Regulación estática Informe
- ✓ Utilice el Post, si se cumplen las limitaciones de tiempo, lugar y ruta Regulación estática Informe

4.3.3 REFERENCIAS

Las siguientes piezas de la documentación pueden hacer referencia durante la realización de este laboratorio, que puede ser descargado desde el sitio Web de Xilinx en <http://www.xilinx.com>

- ✓ PicoBlaze Guía del usuario
- ✓ Spartan-3E Hoja de Datos
- ✓ Digilent Spartan-3E Junta Hoja de Datos
- ✓ Plataforma Flash In-System Programmable PROM hoja de datos de configuración

4.3.4 DISEÑO DESCRIPCIÓN

En este laboratorio, se pondrá en práctica un sistema de procesador embebido con varios periféricos. Desde un punto de vista del hardware, la mayor parte del sistema está prevista para usted. Le animamos a leer la descripción de hardware del sistema, sin embargo, de manera que usted lo entienda.

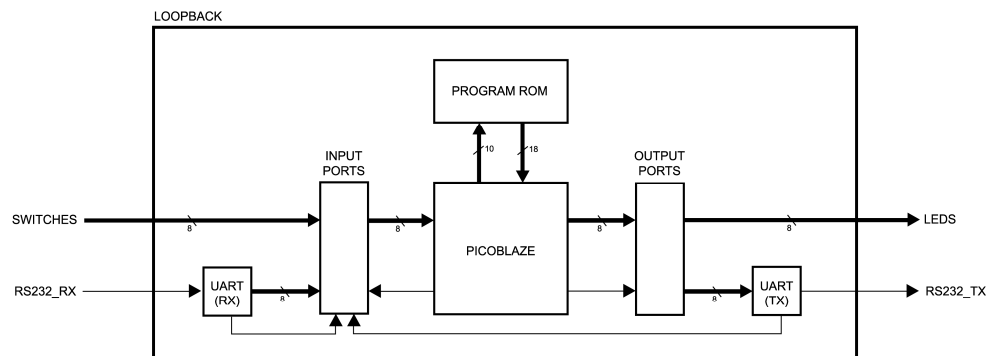


Figura 4.3-1. PicoBlaze Sistema

La principal tarea de este laboratorio es para escribir programas en PicoBlaze assembly para aplicar una prueba de bucle invertido. Una prueba de bucle invertido es una prueba en la que se envía una señal a un dispositivo y regresó desde el mismo dispositivo, como una forma de determinar si el dispositivo está funciona correctamente.

La primera prueba de bucle echo hay que cambiar la configuración en los LED. A continuación, se le transmite con sus dedos, y recibir con sus ojos lo que es devuelto de regreso por el sistema. La prueba de bucle segunda serie se hará eco de los datos recibidos en un puerto serie RS232. Aquí, un equipo de escritorio está transmitiendo con su puerto de serie, y recibir con su puerto serie lo que es devuelto de regreso por el sistema.

Como se muestra en la Figura 1, el sistema tiene un número de entradas. Hay un reloj y una entrada de reset, además de un interruptor de entrada de 4-BIT y una serie de recibir información. La serie de recibir información se origina en el conector RS232 en el

tablero y pasa a través de un traductor de nivel de voltaje antes de llegar al dispositivo FPGA.

De la señal de reloj CLK, 50 MHz del oscilador de

Señal de reinicio RST

De serie rs232_rx recibir aportaciones

Switches [7:0] 8-BIT de entrada de interruptor

También se muestra en la Figura 1 son las salidas. No es un 8-BIT LED de salida y una serie de transmitir la producción. La serie transmitirá la producción se origina en la FPGA y pasa a través de un traductor de nivel de voltaje antes de llegar al conector RS232 en el tablero.

rs232_tx transmitir la producción en serie

Leds [7:0] 8-BIT LED de salida

Usted debe implementar con éxito el sistema de los archivos de la fuente, y luego desarrollar un pequeño programa. El desarrollo de software se divide en tres partes. El final es la implementación del software necesario para transmitir un mensaje corto a cambiar y, a continuación, al mismo tiempo realizar dos funciones de bucle invertido:

- ✓ La configuración del switch eco en los LED
- ✓ Echo en serie los datos recibidos a través de una interfaz RS232

Al completar con éxito este laboratorio, se han adquirido una comprensión de cómo utilizar PicoBlaze para aplicar un sistema simple de procesador embebido

4.3.5 PROCEDIMIENTO

En este laboratorio, se creará un sistema integrado sencillo y especifique la ubicación y las limitaciones de tiempo global. Este laboratorio consta de tres pasos principales:

1. Prepare un programa de PicoBlaze
2. Introduzca limitaciones de tiempo mundial
3. Introduzca el PIN limitaciones de ubicación en la UCF
4. Implementar el diseño y análisis de tiempo real
5. Realizar una simulación de HDL
6. Generar un archivo de programación de
7. Probar el diseño en el hardware

Debajo de cada instrucción general para un procedimiento determinado, se encuentra el paso de acompañamiento-por paso y cifras ilustran los que se detallan para la realización de la instrucción general. Si usted se siente confiado acerca de una instrucción específica, no dude en saltarse el paso por paso y pasar a la instrucción general siguiente en el procedimiento.

Nota: Si usted no puede completar el laboratorio en este momento, usted puede descargar los archivos de laboratorio de este módulo desde el sitio del Programa en la Universidad de Xilinx <http://www.xilinx.com/univ>

4.3.6 PREPARE UN PROGRAMA DE PLANTILLA PASO 1

Montar el programa de plantilla, program. psm, que será utilizado para crear el bucle de recuperación posterior aplicación en el laboratorio. A continuación,

ampliar el diseño PicoBlaze mediante la adición de la ROM de la instrucción generada, programa.

- ✓ Seleccione **Inicio** □ **Programas** □ **Xilinx ISE 10.1i** □ **Project Navigator**
- ✓ Seleccione **Archivo** □ **Abrir proyecto** en el Navegador de proyectos
- ✓ Echar un vistazo a uno de los siguientes directorios:
- ✓ Los usuarios de VHDL: *c: \ xup \ fpgaflow \ labs \ VHDL \ LAB3*
- ✓ Usuarios Verilog: *c: \ xup \ fpgaflow \ labs \ Verilog \ LAB3*

- ✓ *Time_const.ise* Seleccionar, haga clic en **Abrir** y revisar el nivel superior de diseño
- ✓ Abra un símbolo del sistema va iniciar un **Program Accesorios -símbolo del sistema**
- ✓ CD en el sub.-directorio en ensamblador (que se encuentra en el directorio del proyecto), que contiene un programa de plantilla (program.psm) y un archivo por lotes para montarlo

Cd c: \ xup \ fpgaflow \ labs \ verilog \ LAB3 \ ensamblador (usuarios Verilog)

Cd c: \ xup \ fpgaflow \ labs \ VHDL \ LAB3 \ ensamblador (los usuarios de VHDL)

Escriba el siguiente comando en el indicador para montar el programa de plantilla para generar la ROM del programa: **Kcpsm3 programa**

Nota: Esta plantilla es sintácticamente correcta, pero funcionalmente inútil - se volverá a

Completar un significativo programa de tarde.

En ISE, agregar la ROM generado HDL archivo al proyecto.

Realizar una comprobación de sintaxis, poniendo de relieve el diseño de alto nivel de archivo y haga doble clic en **Comprobar sintaxis** en el marco del proceso de síntesis

Cambiar a modo de **simulación de comportamiento** y realizar una simulación usando el testbench.v / VHD, con un tiempo de parada de 35000.

Después de zoom, usted debe ver resultados similares a la figura siguiente, donde un valor hexadecimal de AA y seguido por 55 hexadecimal es la entrada en los interruptores, y no hay salida está presente en los LEDs. Más tarde, se le introduzca el código de hacerse eco de la cambiar a la configuración de los LEDs.

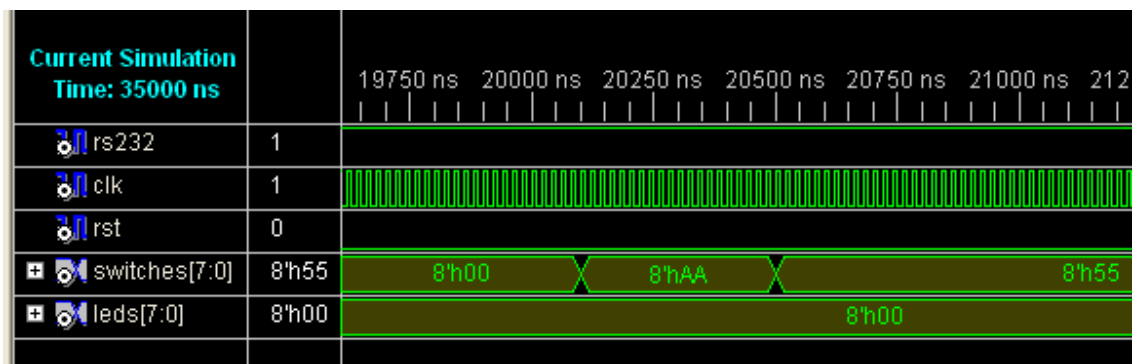


Figura 4.3-4. Cambiar la configuración no echo de nuevo a los LEDs

4.3.7 INTRODUCZA EL MUNDIAL DE RESTRICCIONES TEMPORALES PASO 2

En este paso, que va a utilizar una herramienta gráfica, llamada del editor de restricciones para entrar y PERIODO OFFSET IN / OUT limitaciones.

En las fuentes en la ventana de proyecto, seleccione el diseño de alto nivel de archivo *loopback.vhd / .v*

En los procesos para la ventana Fuente, expanda **Restricciones de usuario** y haga doble clic en **Crear limitaciones de tiempo (Figura 4.3-5)**

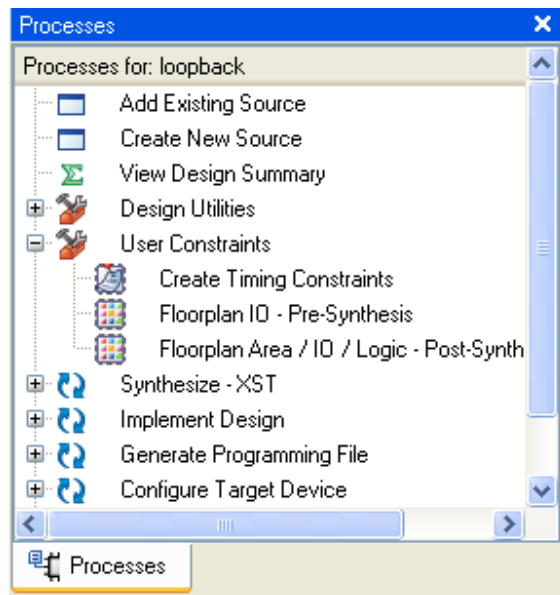


Figura 4.3-5. Procesos para la fuente de la ventana

El proyecto no tiene actualmente un UCF archivo asociado con él. El Navegador de proyecto ofrecerá a crear uno automáticamente.

- ✓ Haga clic en **Yes** para que un archivo de nuevo UCF (*loopback.ucf*) automáticamente crea y se agrega al proyecto
- ✓ En las restricciones, seleccione el **Mundial** (véase la Figura 4.3-6) a la lista de las señales disponibles a nivel global.

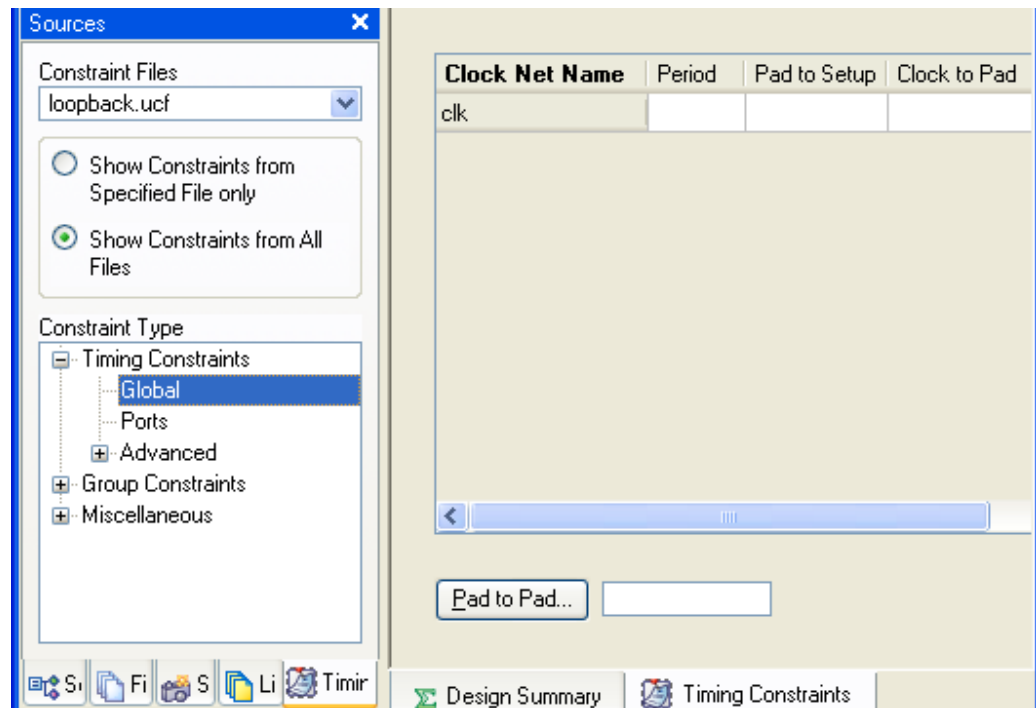


Figura 4.3-6. Editor de restricciones Global - Global ficha

Introduzca una restricción período de 20 ns para *CLK*.

- ✓ Haga doble clic white space under **Period** para abrir el diálogo Periodo de reloj (ver Figura 4.3-7)

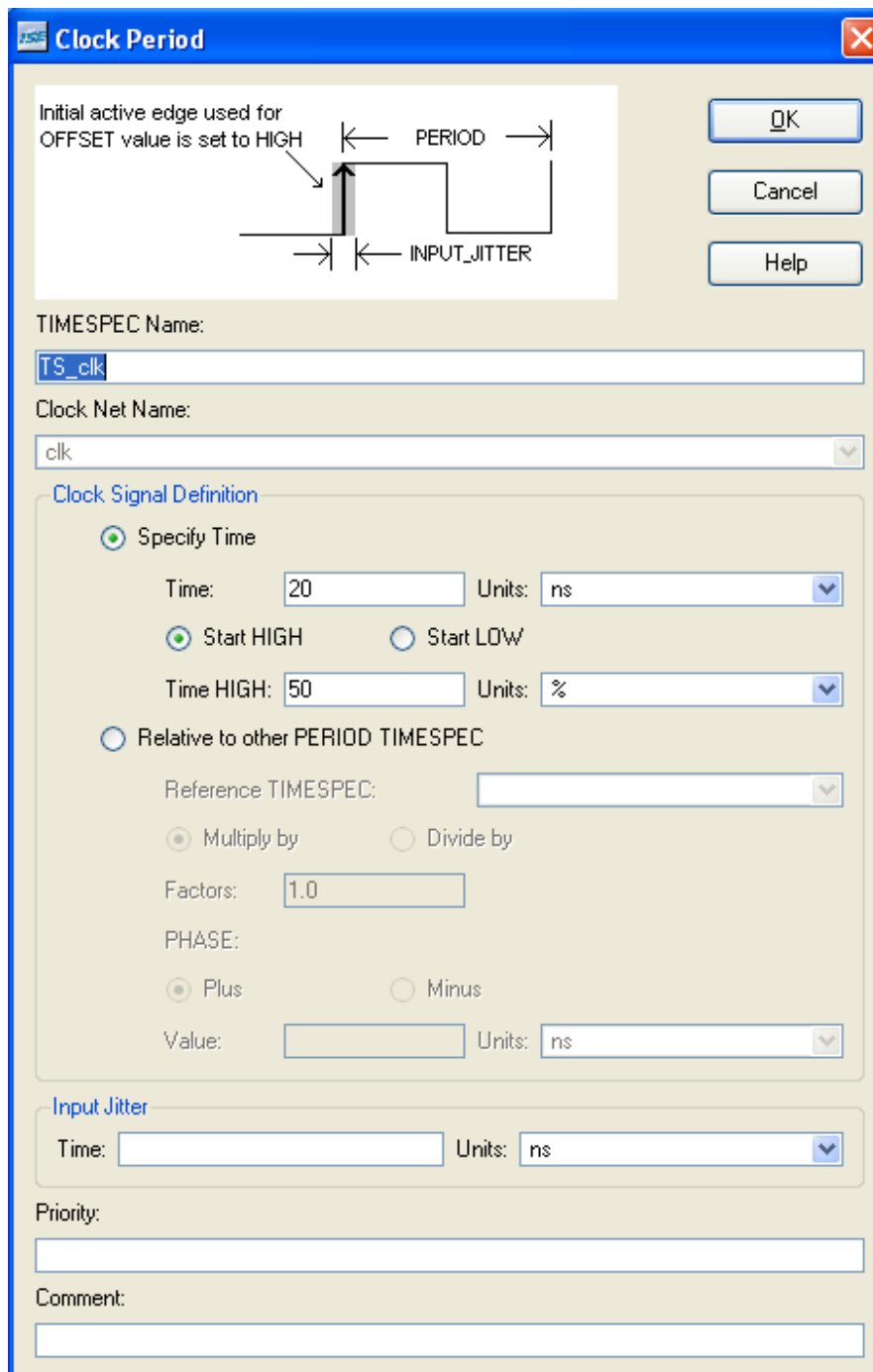


Figura 4.3-7. Diálogo período de reloj

Acepte la configuración predeterminada de 20ns y un 50% ciclo de trabajo, haga clic en Aceptar.

Invocar el desplazamiento en Asistente y entrar en un desplazamiento en valor de restricción de 7 ns para *CLK*.

Haga doble clic en el white space under Pad to Setup (OFFSET IN) para invocar el desplazamiento en Wizard. Salga de la configuración predeterminada (sistema síncrono, SDR, y aumento de borde) y haga clic en <Siguiete> después de revisar la descripción. Tenga en cuenta que el diseño PicoBlaze utiliza un reloj único para todo el diseño, donde todos los registros son en el flanco de subida.

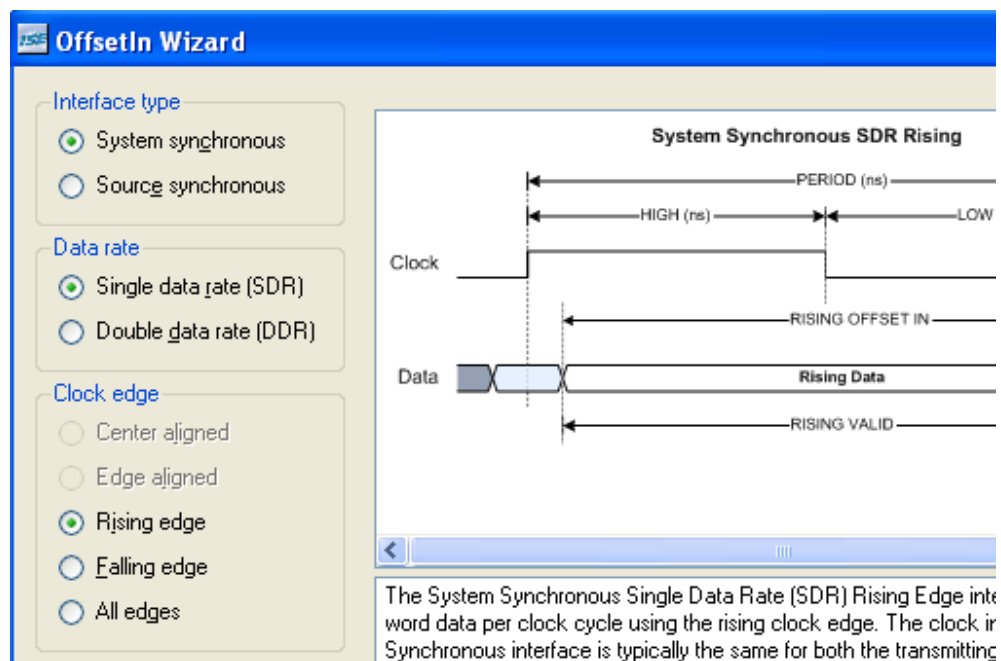


Figura 4.3-8. DESPLAZAMIENTO EN Asistente - Reloj borde de la página

Introduzca el valor de **7 ns** para OFFSET en (ver Figura 4.3-9) y haga clic en <Finish>. Tenga en cuenta que este diseño no tiene requisitos estrictos para el momento en la hora en datos externos, para un valor aleatorio, de 7 ns fue elegido.

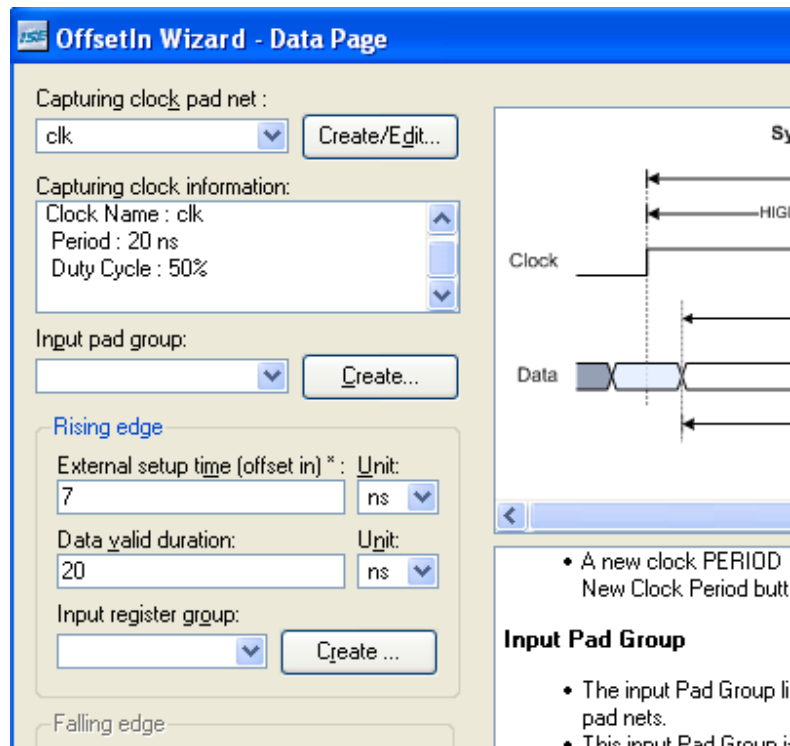


Figura 4.3-9. DESPLAZAMIENTO EN Asistente - Página de datos

Invocar el Asistente para fuera Offset y entrar en una salida de desplazamiento de $7,5\text{ ns}$ para *CLK*. A continuación, guarde los obstáculos y salir del Editor de restricciones.

Haga doble clic en el espacio en blanco en virtud de reloj para Pad (OFFSET OUT) para invocar el asistente de salida Offset y escriba un valor de **7,5 ns** (ver Figura 4.3-10) para la restricción de la salida Offset. <OK> Haga clic cuando haya terminado.

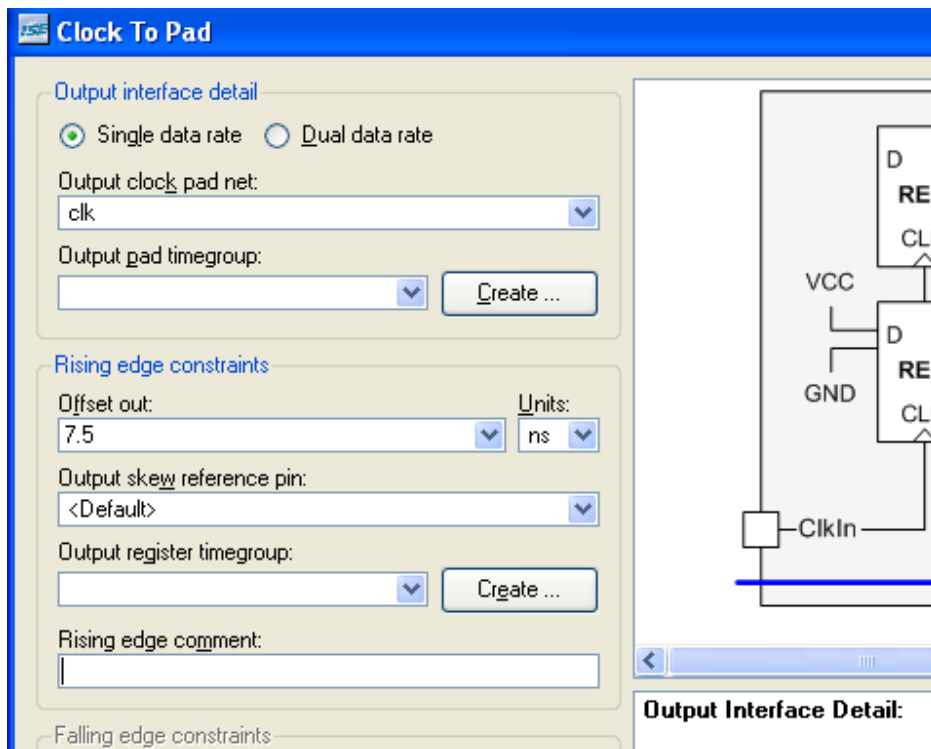


Figura 4.3-10. OFFSET limitaciones a cabo el diálogo

El editor de restricciones de ahora debería tener las tres restricciones formularse del siguiente modo:

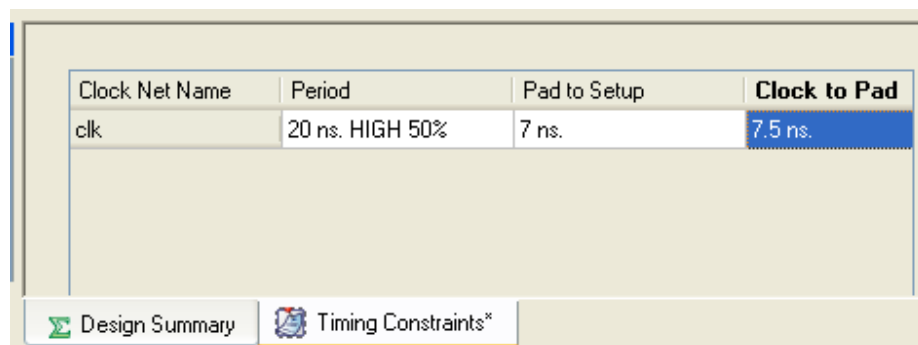


Figura 4.3-11. Limitaciones de tiempo se introducen Mundial

En el LAB3 directorio, pinouts.txt abrir con una utilidad como plataforma de palabra.

Copia de los obstáculos mencionados en el archivo de la UCF, por debajo de las restricciones temporales introducidos por el editor de restricciones.

```
NET "clk" TNM_NET = clk;
TIMESPEC TS_clk = PERIOD "clk" 20 ns HIGH 50%;
OFFSET = IN 7 ns VALID 20 ns BEFORE "clk" RISING;
OFFSET = OUT 7.5 ns AFTER "clk";

NET "clk" LOC = "c9" | IOSTANDARD = LVCMOS33 ;
NET "leds<0>" LOC = "f12" | IOSTANDARD = LVTTTL | SLEW = SLOW | DRIVE = 8 ;
NET "leds<1>" LOC = "e12" | IOSTANDARD = LVTTTL | SLEW = SLOW | DRIVE = 8 ;
NET "leds<2>" LOC = "e11" | IOSTANDARD = LVTTTL | SLEW = SLOW | DRIVE = 8 ;
NET "leds<3>" LOC = "f11" | IOSTANDARD = LVTTTL | SLEW = SLOW | DRIVE = 8 ;
NET "leds<4>" LOC = "c11" | IOSTANDARD = LVTTTL | SLEW = SLOW | DRIVE = 8 ;
NET "leds<5>" LOC = "d11" | IOSTANDARD = LVTTTL | SLEW = SLOW | DRIVE = 8 ;
NET "leds<6>" LOC = "e9" | IOSTANDARD = LVTTTL | SLEW = SLOW | DRIVE = 8 ;
NET "leds<7>" LOC = "f9" | IOSTANDARD = LVTTTL | SLEW = SLOW | DRIVE = 8 ;
NET "rs232_rx" LOC = "R7" | IOSTANDARD = LVTTTL ;
NET "rs232_tx" LOC = "M14" | IOSTANDARD = LVTTTL ;
NET "rst" LOC = "v16" | IOSTANDARD = LVTTTL | PULLDOWN ;
NET "switches<0>" LOC = "l13" | IOSTANDARD = LVTTTL | PULLUP ;
NET "switches<1>" LOC = "l14" | IOSTANDARD = LVTTTL | PULLUP ;
NET "switches<2>" LOC = "h18" | IOSTANDARD = LVTTTL | PULLUP ;
NET "switches<3>" LOC = "n17" | IOSTANDARD = LVTTTL | PULLUP ;
NET "switches<4>" IOSTANDARD = LVTTTL ;
NET "switches<5>" IOSTANDARD = LVTTTL ;
NET "switches<6>" IOSTANDARD = LVTTTL ;
NET "switches<7>" IOSTANDARD = LVTTTL ;
```

Figura 4.3-13. Entre las limitaciones en el archivo de la UCF

✓ Guardar y cierre el archivo UCF

4.3.9 IMPLEMENTAR EL DISEÑO Y ANÁLISIS DE LA SINCRONIZACIÓN PASO 4

Aplicar el diseño. Mira a través de la Post-MAP Static Timing Report y el Post-Place de carreteras y estática Timing Informe para completar el cuadro 1 y el gráfico 2 en esta sección.

- ✓ En los procesos para la ventana de la Fuente, ampliar el proceso de **Implementación de diseño**, y ampliar el **proceso** de la hoja
- ✓ Si usted no ve el proceso de Implementación del diseño, asegúrese de que *loopback.vhd / .v* está seleccionada en las fuentes de la ventana del proyecto.
- ✓ Expandir el Post-Generar el mapa de procesos Regulación estática y haga doble clic en Analizar Post-Mapa de Regulación estática

La realización de estas medidas aplica el diseño a través de MAP, genera el Post-Map Static Timing Report, y abre el informe en el calendario Analyzer. Utilice el informe para verificar que sus limitaciones de tiempo son realistas y evitar la pérdida de Lugar de carreteras y el tiempo.

Complete el siguiente cuadro escribiendo el pedido y los valores reales. Al entrar en el periodo, consultar la restricción de que fue colocado en CLKFX_BUF que esta es la salida de DCM utilizados para el diseño del reloj.

Gráfico 1	Restricción PERIODO	DESPLAZAMIENTO EN restricción	OFFSET restricción OUT
Pidió			
Actual			

Compare sus respuestas con las de la sección de respuestas de este laboratorio.

- ✓ La salida **post-Timing Mapa Analyzer**
- ✓ En los procesos para la ventana Origen, ampliar el **Lugar de carreteras** y el proceso
- ✓ Ampliar la **Generación Post-Place de carreteras** y proceso de **Regulación estática** y haga doble clic en **Analizar Post-Place de carreteras y Regulación estática**
- ✓ Complete el siguiente cuadro escribiendo el pedido y los valores reales. Al entrar en el periodo, consultar la restricción de que fue colocado en CLKFX_BUF que esta es la salida de DCM utilizados para el diseño del reloj.

Gráfico 2	Restricción PERIODO	DESPLAZAMIENTO EN restricción	OFFSET restricción OUT
Pidió			
Actualm			

Tus respuestas con las de la sección de respuestas de este laboratorio.

Nota: Si su diseño no responde a tiempo después de lugar de carreteras y, a continuación, aflojar la restricción de la ruta y no volver a ejecutar la aplicación.

- ✓ La salida **post-Lugar de carreteras y Timing Analyzer**

4.3.10 REALICE EL PASO DE SIMULACIÓN HDL 5

- ✓ Ahora que el hardware cumple el calendario, que ahora se desarrollará una asamblea PicoBlaze
- ✓ Programa para completar la primera de las tres pruebas de loop-back. El programa de plantilla
- ✓ Contiene una serie de definiciones constante para su conveniencia y está estructurado de manera

- ✓ Que se puede aplicar cada una de las tres pruebas de forma independiente.
Entrará
- ✓ Código de montaje en la plantilla y ejecutar el ensamblador para volver a generar el programa de
- ✓ Los archivos de la memoria. A continuación, se simulará el diseño para comprobar que el interruptor DIP
- ✓ Ajustes se hicieron eco de la salida del LED.

```

;=====
, Programa de montaje real va aquí..
;=====

cold_start: CARGA s0, all_clear; cero fuera reg s0

, Task LAB # 2

; Escribir código para una salida de corto (10 caracteres
, O menos) mensajes al puerto serie.

lazo: CARGA s0, all_clear; cero fuera reg s0 (NOP)

, Task LAB # 1

; Escribir código para leer el estado de conmutación y
, Y luego escribirla en el puerto de control LED.

rs232_echo: CARGA s0, all_clear; cero fuera reg s0 (NOP)

, Task LAB # 3

; Escribir código para comprobar si un byte ha sido

```



```
; Recibido por la UART. Si es así, escribir
; De nuevo a la UART transmitir puerto. Si no
, No hacen nada y sólo...
```

Ciclo saltar; bucle de nuevo

```
;=====
;
;=====
```

Figura 4.3-14. Loop-back PicoBlaze plantilla de aplicación

Crear el código necesario para realizar un bucle de nuevo la prueba de la Tarea N ° 1 y montarlo. Una vez que el archivo de ROM se ha generado, se llevará a cabo una simulación del comportamiento para probar el lazo de devolución de aplicación.

Escriba el código de tarea # 1 de program.psm edición en el subdirectorio Assemble para leer el estado de cambiar a un registro PicoBlaze y luego escribir el estado para el puerto de control LED.

Nota: Consulte las constantes de la plantilla de montaje para los valores de puerto y la documentación PicoBlaze para obtener instrucciones.

Sugerencia: Usted sólo tiene que escribir dos líneas de código (consulte el manual de usuario KCPSM3)

- ✓ Una vez que el código se ha escrito, volver a montar el programa de
- ✓ Cambiar a modo de **simulación de comportamiento** y haga doble clic en

Simular modelo de comportamiento

- ✓ Analizar la forma de onda de salida y de la consola, teniendo en cuenta la configuración del switch se hizo eco en los LED.

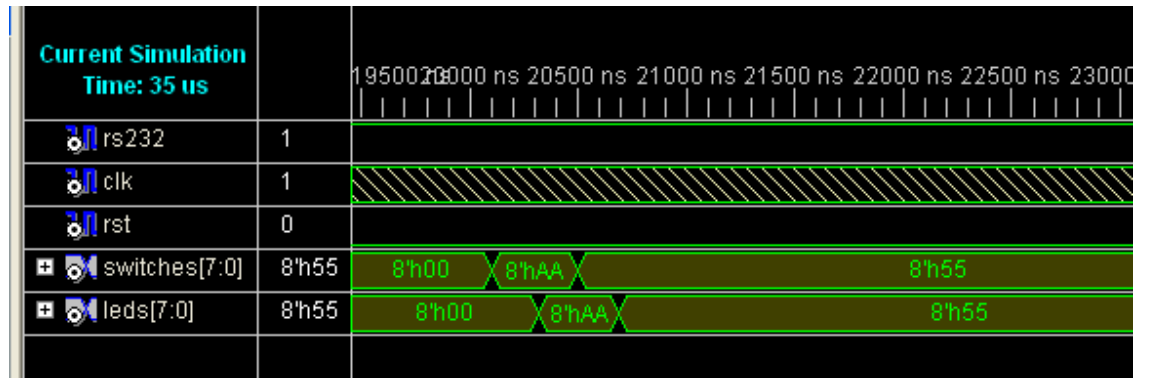


Figura 4.3-15. Cambie la configuración de eco de nuevo a los LEDs

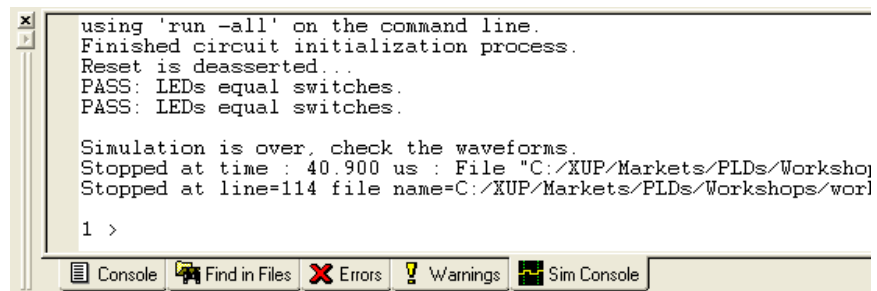


Figura 4.3-16. Ver mensajes en la consola de simulación

- ✓ Cierre la ventana cuando haya terminado.

4.3.11 GENERAR UN ARCHIVO DE PROGRAMACIÓN DE PASO 6

El flash PROM plataforma Xilinx proporcionar un método reprogramable para almacenar grandes corrientes de bits de configuración Xilinx FPGA. El espartano Digilent-3E Junta está equipado con un 4 Mbit xcf04s PROM flash plataforma que puede almacenar una secuencia de bits de un xc3s500e, que exige 2.270.208 bits de configuración. En este paso, que va a utilizar iMPACT para generar un archivo con formato Intel MCS al programa de la PROM.

- ✓ Haga doble clic en **Generar Meta PROM / ACE archivo** en proceso de **configurar el dispositivo de destino**.

ISE en primer lugar generar el flujo de bits y luego abra

Seleccione **Preparar un archivo RPM** y haga clic en Siguiente para continuar

- ✓ Deje los valores por defecto (fig. 4.3-17) con **Xilinx PROM** y **MCS** seleccionados, opcionalmente, proporcionar un nombre de archivo PROM, y, a continuación, haga clic en Siguiente para continuar.

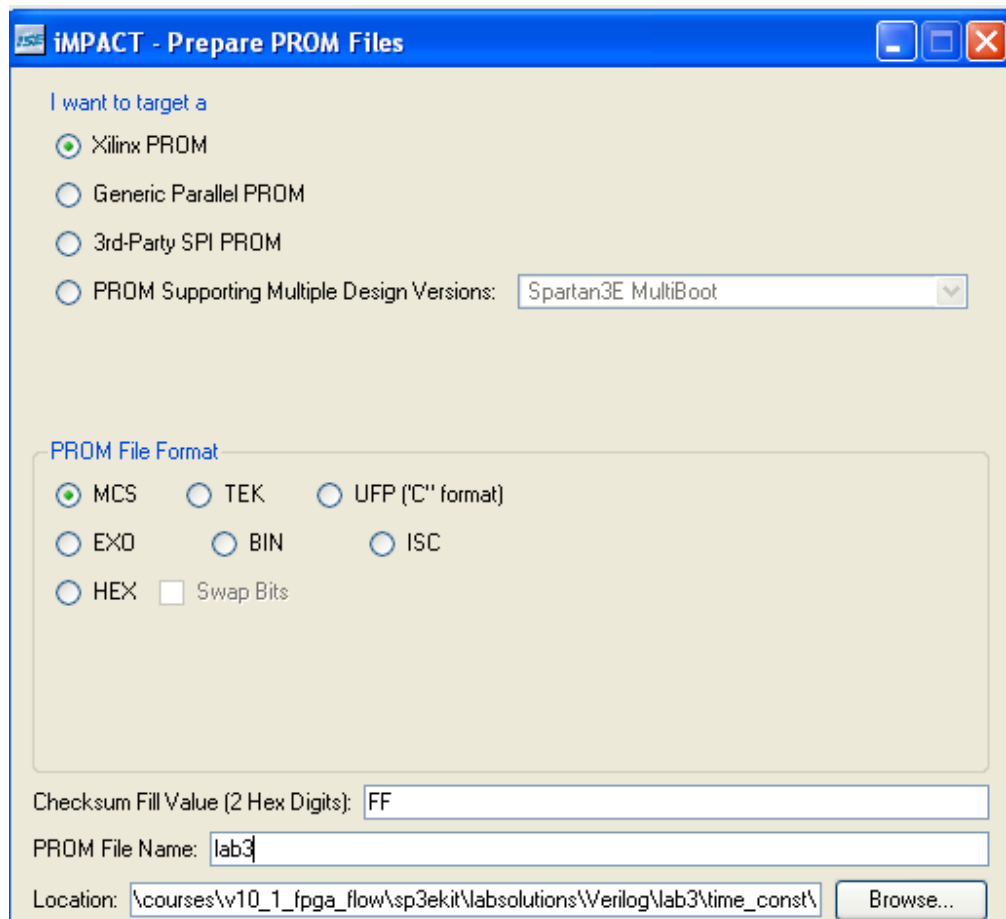


Figura 4.3-17. Preparación de los archivos RPM

- ✓ Deje la opción por defecto **estoy usando un PROM Xilinx en modo de serie** y haga clic en **Siguiente**.
- ✓ Seleccione la **xcf04s** Plataforma Flash PROM de la lista desplegable y haga clic en el botón **Añadir** (figura 4.3-18). <Siguiente> Haga clic para continuar

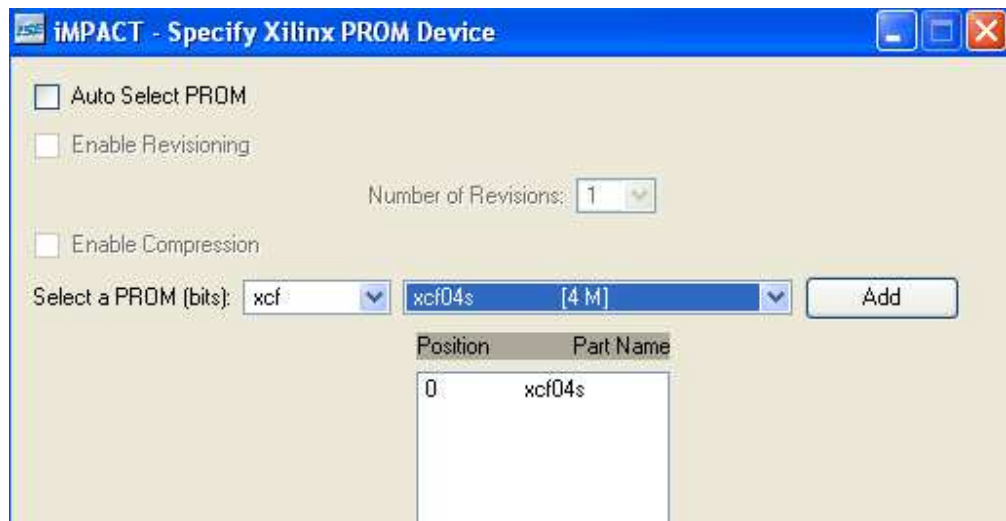


Figura 4.3-18. Especifique la PROM xcf04s para la Digilent Spartan-3E bordo

- ✓ Haga clic en Next, Siguiente "y <Finish> después de especificar el dispositivo PROM.
- ✓ Haga clic en <OK> y agregar la loopback.bit.
- ✓ <No> Haga clic en cuando se abre el cuadro de diálogo para que NO agregar otro indirecto.

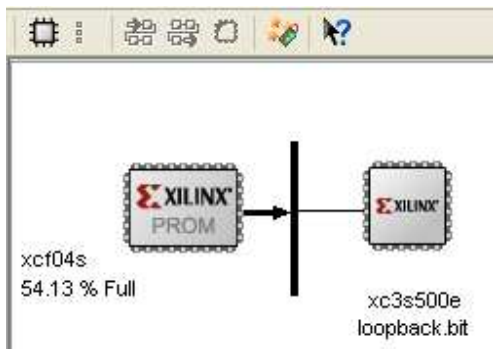


Figura 4.3-19. Bitstream se asocia ahora con la RPM

- ✓ Haga doble clic en **Generar el archivo...** en la ventana de Procesos para generar el archivo de MCS. Deberías ver el siguiente mensaje "PROM de generación de archivos con éxito".

4.3.12 PROBAR EL DISEÑO DE HARDWARE EN EL PASO 7

En este paso, podrás cambiar al modo de configuración y configurar el flash de la plataforma

PROM usando el MCS archivo generado en el último paso. A continuación, configurar el FPGA de la PROM y probar la aplicación de bucle invertido en la junta Digilent.

- ✓ Potencia y conectar el Spartan-3E bordo
- ✓ Iniciar la cadena haciendo doble clic en **Boundary-Scan** y seleccionando **Iniciar la cadena** después de hacer clic derecho sobre el espacio en blanco (Figura 4.3-20)

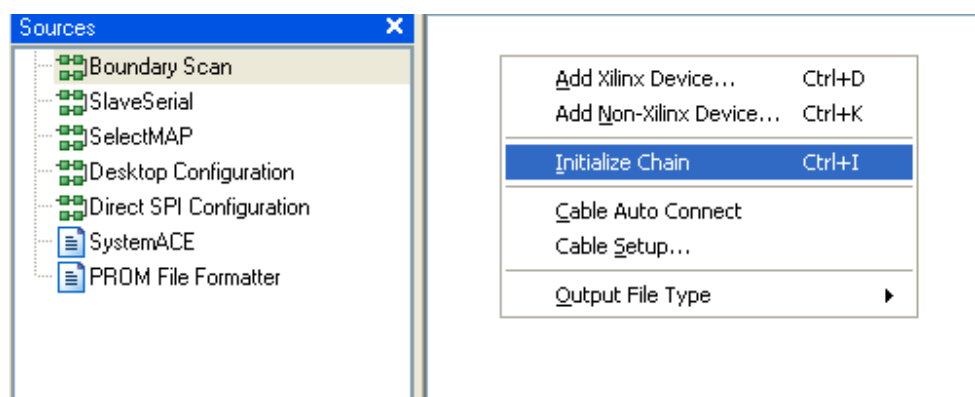


Figura 4.3-20. Inicializar Boundary Scan Cadena

- ✓ Añadir el **.Mcs** archivo en el dispositivo de xcf04s Plataforma Flash, pasando por alto el Spartan-3E y dispositivos CPLD. <OK> Haga clic para cerrar el cuadro de diálogo **Propiedades de programación**.
- ✓ Haga clic en <OK> Programa de la PROM. Haga clic en el xcf04s en la ventana de IMPACT y seleccione **Program**. En el cuadro de diálogo **Propiedades de programación** para borrar y el programa de la PROM.
- ✓ Compruebe que los puentes de configuración de modo se configura de manera que el flujo de bits se carga desde la Plataforma Flash en el poder hacia arriba (consultar con Digilent Spartan-3E guía del usuario). Reciclaje del poder en el tablero Digilent reconfigurar el Spartan-3E a través de la plataforma Flash PROM y la tapa de los interruptores para encender los LED de encendido / apagado.

4.3.13 CONCLUSIÓN

En este laboratorio, que utiliza el Editor de restricciones Xilinx para entrar en las restricciones temporales globales. También examinó el -Map and Post-Place & Route Timing Reports

Restricciones temporales son la mejor manera de comunicar sus expectativas de rendimiento a las herramientas de aplicación.

Usted debe verificar que sus limitaciones de tiempo son realistas, mientras que las herramientas de aplicación son colocación y el trazado de su diseño para la primera vez. Usted puede obtener una estimación del rendimiento momento del Post-Map Static Timing Report.

Después de la implementación es completa, limitaciones de tiempo debe ser verificada con el Post-Place de carreteras y estática Calendario informe o un informe personalizado momento generado por el momento Analyzer.

4.3.14 RESPUESTAS

Laboratorio de respuestas enumeradas representan soluciones de muestra solamente. Sus resultados pueden variar dependiendo de la versión del software, Service Pack, o el sistema operativo que esté utilizando.

1. Complete la fila titulada Post-Mapa en la siguiente tabla:

Gráfico 1	Restricción PERIODO	DESPLAZAMIENTO EN restricción	OFFSET restricción OUT
Pidió	18,18 NS	7 ns	7,5 ns
Actual	~ 9,97 ns	~ 4,6 ns	~ 4.5 ns

2. Post completo la fila titulada-P & R en el siguiente cuadro:

Gráfico 2	Restricción PERIODO	DESPLAZAMIENTO EN restricción	OFFSET restricción OUT
Pidió	~ 18,18 ns	7 ns	7,5 ns
Actual	~ 13,52 ns	~ 5,51 ns	~ 7,22 ns

Práctica 4 : Técnicas de Síntesis

Orientación de la Spartan-3E Starter Kit

4.4 LA SÍNTESIS DE TÉCNICAS DE LABORATORIO

4.4.1 INTRODUCCIÓN

Este laboratorio estudia el proceso de usar las opciones de síntesis para mejorar la depuración y la síntesis de los resultados.

4.4.2 OBJETIVOS

Después de completar este laboratorio, usted será capaz de:

- ✓ Mantenga la Jerarquía y las opciones de la síntesis de expansiones
- ✓ Lea el informe de síntesis por software XST

4.4.3 PROCEDIMIENTO

Este laboratorio consta de cuatro pasos primarios:

1. Complete el diseño
2. Sintetizar con las opciones predeterminadas
3. Cambiar las opciones de la síntesis
4. Ver los resultados de síntesis en el visor de RTL

Después de cada instrucción general para un procedimiento determinado, se encuentra el paso de acompañamiento-por paso y cifras ilustran los que se detallan para la realización de la instrucción general. Si usted se siente confiado acerca de una instrucción específica, no dude en saltarse el paso por paso y pasar a la instrucción general siguiente en el procedimiento

Nota: Si usted no puede completar el laboratorio en este momento, usted puede descargar los archivos de laboratorio de este módulo desde el sitio del Programa en la Universidad de Xilinx <http://www.xilinx.com/univ>

4.4.4 COMPLETE EL DISEÑO

PASO 1



Poner en marcha el proyecto ISE Navigator TM y abra el archivo de proyecto `synth_lab.ise`.

❶ Para abrir el Xilinx ISE software, seleccione **Start** → **Programs** → **Xilinx ISE Design Suite 10.1** → **ISE** → **Project Navigator**

❷ Seleccione **File** → **Open Project**

Verilog usuarios: Buscar en `c:\xup\pgaflow\labs\verilog\lab4`

VHDL usuarios: Buscar en `c:\xup\pgaflow\labs\vhdl\lab4`

Seleccione `synth_lab.ise` y haga click en Open



Actualización de la `program.psm` archivo creado en LAB3 para completar la tarea # 2 para mostrar el mensaje de "Xilinx Reglamento" y montar el programa para generar el archivo de ROM del programa. Agregue el archivo de ROM con el proyecto.

❶ Abra `program.psm` (localizado en `Assembler directory`) y agregue el código para completar la tarea #2, refiriéndose a la documentación de la información técnica sobre PicoBlaze y el Assembler.

Sugerencia: Todos los caracteres ASCII se incluyen en la lista de constantes situado en la parte superior del programa. Sólo dos instrucciones (de carga y salida) están obligadas a mostrar un carácter único.

❷ Abra un símbolo del sistema y vaya al directorio en ensamblador, que contiene el programa `d`

❸ Escriba el siguiente comando en el indicador para montar el programa y generar los archivos de ROM

```
> kcpsm3 program
```

❹ Añadir el archive de `program.v` (or `.vhd`) al proyecto si no esta añadido.

4.4.5 SINTETIZAR E IMPLEMENTAR USO DE LAS OPCIONES POR DEFECTO

PASO 2



Sintetizar el diseño y revisión de los resultados con el informe de síntesis.

- 1. Con *loopback.v/vhd* seleccionado, expanda el **Synthesize** process y haga doble-click en **View Synthesis Report**.

ISE primero sintetizara el diseño antes de abrir el reporte.

- 2. En el campo de búsqueda que esta encima del editor de texto escribirá *Timing Summary* y presionara **<Enter>**



1. Crear una nota aquí de la frecuencia de reloj estimado para la comparación posterior.

- 3. Ingrese *Device Utilization* en el campo de búsqueda y presiona **<Enter>** hasta que encuentre el sumario



2. Ingrese el número de recursos en la tabla de abajo.

Slices	
Slice Flip Flops	
4 input LUTs	
IOBs	
BRAMs	
Global Clocks	
DCMs	

- ④ Cierre el reporte
- ⑤ Haga doble-click en **View/Edit Placed Design (Floorplanner)** debajo del proceso de lugar & ruta.
- ⑥ Revise el diseño de jerarquía y ubicación de piso y note que le diseño en marcha se indicara por los colores (mismo color para todos los diseños de módulos).

Note que la fuente seleccionada (ie. mux) en el diseño de jerarquía se resaltara en la fuente en la vista del panel de piso mostrando in/out if enabled.

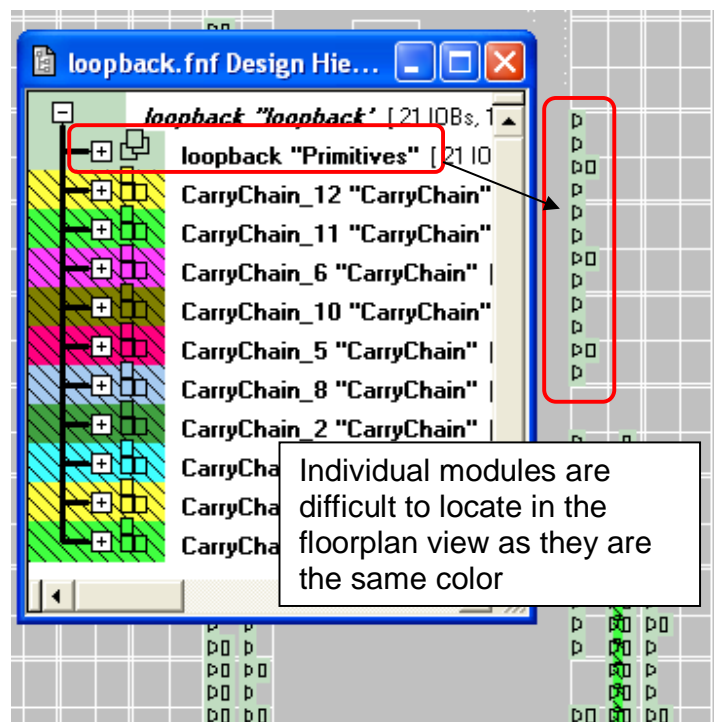


Figura 4. 4-1. Design Hierarchy and Floorplan View of Verilog Design

- ⑦ Exit the Floorplanner

4.4.6 CAMBIAR LAS OPCIONES DE SÍNTESIS

PASO 3



En general, un diseño HDL es una colección de bloques jerárquicos, y preservar la jerarquía da la ventaja de un procesamiento rápido debido a que la optimización se hace en piezas separadas que reduce la complejidad. Sin embargo, muy a menudo, la fusión de los bloques de la jerarquía mejora los resultados de ajuste (PTerms menos y macrocélulas dispositivo, mejor frecuencia), porque los procesos de optimización (colapsando, factorización) se aplican a nivel mundial en toda la lógica.

- ❶ De forma predeterminada, el diseño es sintetizado a un netlist aplanada. Vamos a cambiar una opción de síntesis para generar un netlist jerárquico y observar los efectos de área y de tiempo.
- ❷ En Opciones de síntesis, se **Mantiene Jerarquía Sí**, como se muestra en la **Figura 4.4-2** Under **Synthesis Options** , set **Keep Hierarchy** to **Yes** as shown in **Figure 4.4-2**

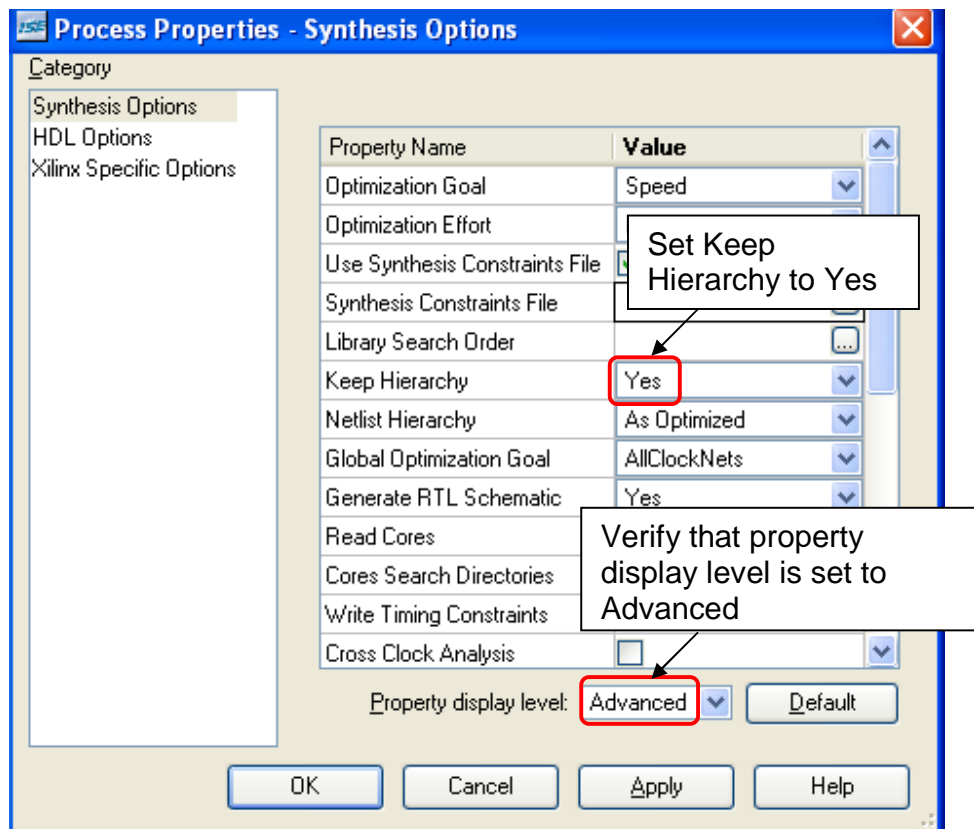


Figura 4.4-2. Propiedades Sintentizadas

- ③ Haga Click en <OK> y resintente el diseño
- ④ Abra el reporte de sintensis e ingrese **Timing Summary** en campo de ingreso arriba del editor de texto,
Y presione <Enter>



3. Haga una nota aqui para el estimado de la frecuencia de reloj para una comparación posterior.

⑥ Ingrese el dispositivo de utilización en el campo Find What y haga en click **Find Next**



4. Ingrese el número de recursos en la tabla de abajo

Slices	
Slice Flip Flops	
4 input LUTs	
IOBs	
BRAMs	
Global Clocks	
DCMs	

⑥ Ingrese *Fanout* en el campo Find What y haga click en **Find Next**



5. Que recursos tiene el máximo fanout?



Fanouts puede causar problemas rute habilidad, por lo tanto XST trata de limitar expansiones mediante la duplicación de las puertas o mediante la inserción de tampones. Este límite no es un límite de la tecnología, sino una guía para XST. Puede ocurrir que este límite no es exactamente respetados, especialmente cuando este límite es pequeño (inferior a 30).

En la mayoría de los casos, la duplicación de la puerta de las redes de conducción con expansiones realiza grandes expansiones de control. Si la duplicación no se puede realizar, a continuación, topes será insertado. Estos topes serán protegidos contra la lógica de recorte en el nivel de ejecución mediante la definición de un atributo Mantener en el netlist NGC.

En esta sección usted reducirá el máximo de fanout.

- 1 Haga click derecho en **Synthesize – XST** y seleccione **Properties**
- 2 En la rama **Xilinx Specific Options**, cambia el valor predeterminado de 500 a 50 para un **Max Fanout**. haga Click en <OK>

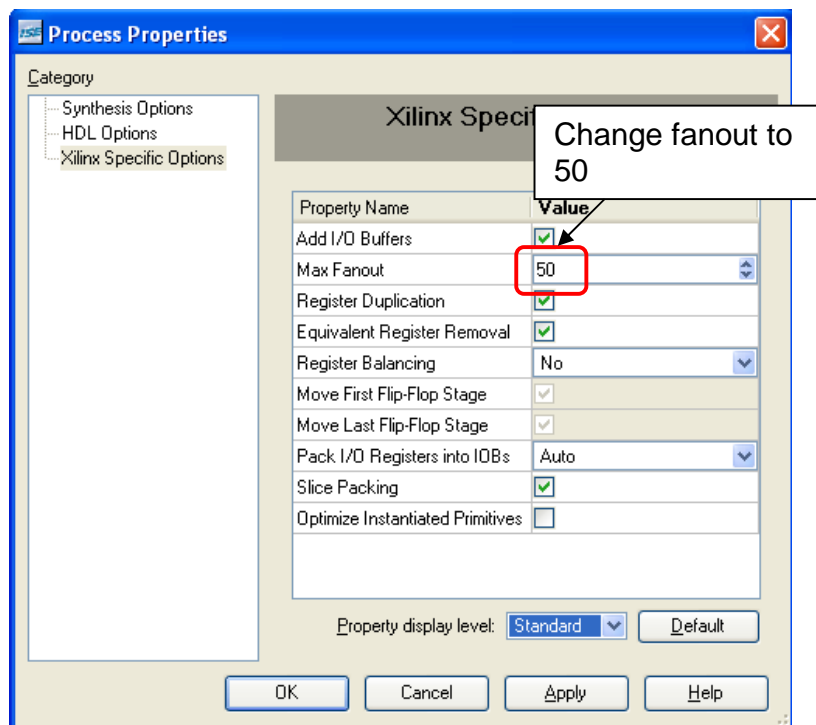


Figura 4-3. Opciones Xilinx específicas

- ③ Haga doble click en **Synthesize – XST** para resintetizar el diseño.
- ④ Abra el reporte de sintensis y desempeño para buscar el **fanout**



6. Que es un fanout de la fuente que anteriormente tiene un mínimo fanout?

- ⑤ En el reporte de sintensis , desempeñe una búsqueda para **timing summary**

7. Cual es la frecuencia máxima de reloj?

- ⑥ Expanda **Implement Design** y haga doble click en **Place & Route**
- ⑦ Abra el planificador de suelo y revise el diseño de jerarquía notando el color del código que indica que modulo en el diseño ha sido sintetizado por medio de bloques de jerarquía.

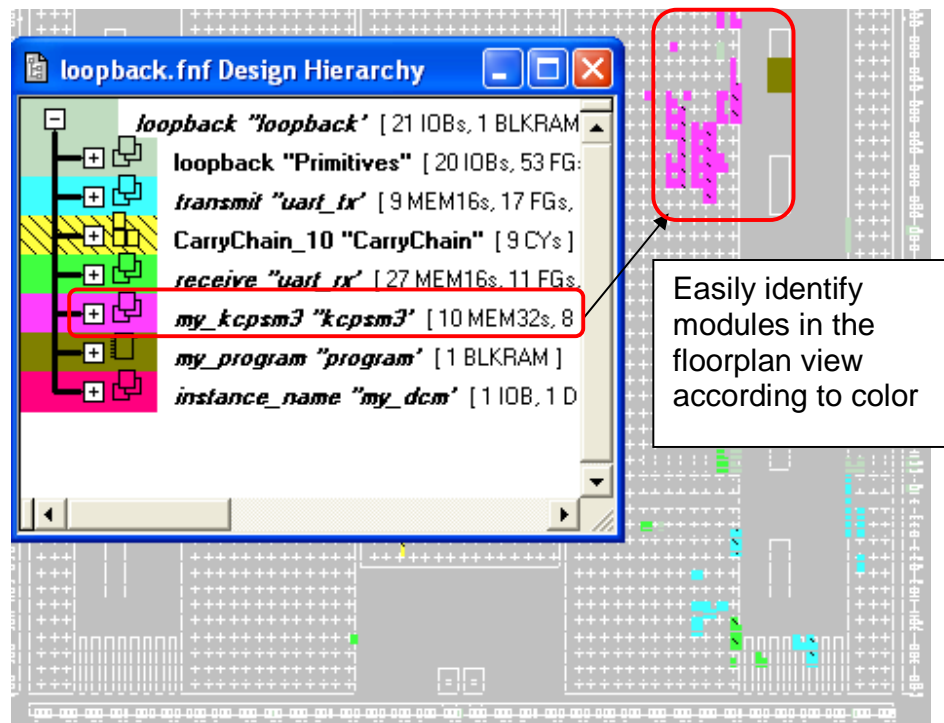


Figura 4. 4-4. Diseño de la Jerarquía y plano de diseño de Verilog

4.4.7 DESCARGA Y PRUEBA DE SISTEMA

PASO 5



En este paso usted, generara la configuración y el indirecto del FPGA a través de JTAG

Cable de descarga.

- ❶ configurar el modo de puentes de configuración (J30 en Spartan-3E Starter Kit) para permitir la configuración a través del cable de descarga.
- ❷ Conecte el cable de descarga y de potencia hasta el tablero Digilent Spartan-3E.
- ❸ Inicie una sesión de HyperTerminal con la siguiente configuración
 - Baud rate 9600
 - 8 data bits
 - No parity
 - 1 stop bit

- No flow control
- ④ Haga doble click en **Manage configuration project (iMPACT)** bajo el proceso **Configure Target Device** .
 - ⑤ Cuando la ventana de configuración aparezca verifique usando **boundary-Scan (JTAG)** es seleccionado, click en <Next> y luego <Finish>
 - ⑥ Asigne el **loopback.bit** al **xc3s500e** (Primer dispositivo de la cadena JTAG) y puentee la plataforma flash PROM y CPLD.
 - ⑦ Haga Click en <OK> para cerrar la ventana **Programming Properties** .

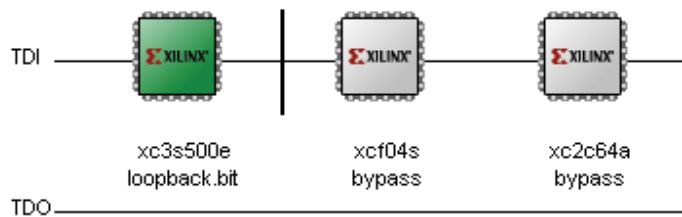


Figura 4-5. Asignar la configuración de archivo loopback.bit

- ⑧ Haga click derecho en el dispositivo **xc3s500e** y seleccione el programa

Nota: Usted deberá ver el mensaje **Xilinx Rules** en la ventana de hyperterminal

- ⑨ Voltee los interruptores en el tablero Digilent y observe el encendido de los LEDS

4.4.8 CONCLUSIÓN

Especificación de las opciones de síntesis puede ayudar en la depuración de los diseños y los objetivos de rendimiento. El informe de síntesis ofrece una estimación de los recursos consumidos y plazos previstos.

4.4.9 RESPUESTAS



1. Haga una nota estimada de la frecuencia de reloj para una comparación posterior.

Período mínimo (verilog): ~9.28 ns (Frecuencia Máxima: ~107.76 MHz)

Periodo Mínimo (VHDL): ~8.15 ns (Frecuencia Máxima: ~122 MHz)

Note que los resultados Irán variando dependiendo del sistema de su PC

2. Ingrese los números estimados de la fuente en la tabla de abajo

Slices	163 (Verilog)/ 161 (VHDL)
Slice Flip Flops	147 (Verilog)/ 147 (VHDL)
4 input LUTs	305 (Verilog)/ 300 (VHDL)
IOBs	21 (Verilog and VHDL)
BRAMs	1 (Verilog and VHDL)
Global Clocks	2 (Verilog and VHDL)
DCMs	1 (Verilog and VHDL)

3. Haga una del estimado de la frecuencia de reloj. El desarrollo mejoro o empeoro? porque?

Periodo mínimo (Verilog): ~9.8 ns (Frecuencia Máxima: ~102 MHz)

Periodo mínimo (VHDL): ~8.68 ns (Frecuencia Máxima: ~115 MHz)

4. Ingrese los valores estimados de la fuente en la tabla de abajo

Slices	167 (Verilog)/ 163 (VHDL)
Slice Flip Flops	147 (Verilog)/ 147 (VHDL)
4 input LUTs	305 (Verilog)/ 299 (VHDL)
IOBs	21 (Verilog and VHDL)
BRAMs	1 (Verilog and VHDL)
Global Clocks	2 (Verilog and VHDL)
DCMs	1 (Verilog and VHDL)

5. Que recursos tienen el fanout mínimo?

Verilog

LUT3:I2->O con fanout of 84

VHDL

LUT3:I2->O con fanout of 84

6. Que es el fanout de la fuente que anteriormente tenia un máximo fanout?

Verilog

LUT3:I2->O con fanout of 43

VHDL

LUT3:I2->O con fanout 43

*Práctica 5: Practica de CORE
Sistema Generador Lab*

Orientación del kit de arranque Spartan-3E

4.5 CORE SISTEMA GENERADOR DE LABORATORIO

4.5.1 INTRODUCTION

En la guía de laboratorio a través del proceso de creación de un núcleo con el sistema de generador de Zilina CORE™ y la inserción en el núcleo de su diseño.

4.5.2 OBJETIVOS

Después de completar este laboratorio, usted será capaz de:

- ✓ Crear un núcleo utilizando CORE Generator
- ✓ Crear instancias de un núcleo en un diseño HDL existente
- ✓ Realizar la simulación del comportamiento en un diseño de HDL que contiene un núcleo
- ✓ Probar el diseño en hardware

4.5.3 PROCEDIMIENTO

En esta práctica, que va a utilizar el sistema central generador para crear una memoria RAM bloque inicia con una aplicación de software, una instancia en un diseño PicoBlaze, y prueba en el Digilent del tablero Spartan-3E . En esta práctica, que va a utilizar el sistema central generador para crear una memoria RAM bloque inicia con una aplicación de software, una instancia en un diseño PicoBlaze, y prueba en el Digilent del tablero del Spartan-3E .

Este laboratorio consta de cuatro pasos principales: Se revisará el diseño, generar el núcleo, una instancia del bloque de RAM principal, y realizar la simulación del comportamiento en el módulo de bucle invertido nuevo. Después de cada instrucción general para un procedimiento determinado, se encuentra el paso de acompañamiento-por paso y cifras ilustran los que se detallan para la realización de la instrucción general. Si usted se siente confiado acerca de una instrucción específica, no dude en saltarse el paso por paso y pasar a la instrucción general siguiente en el procedimiento.

Nota: Si usted no puede completar el laboratorio en este momento, usted puede descargar los archivos de laboratorio de este módulo desde el sitio del Programa en la Universidad de Xilinx <http://www.xilinx.com/univ>

4.5.4 CREAR UN NÚCLEO USANDO CORE GENERATOR PASO 1



Usando el programa desde el laboratorio anterior, se completará la tercera tarea que bucle apoya golpes de teclado para HyperTerminal. A continuación, se reunirán el programa para generar un archivo. Coe. Finalmente, se le va a crear un programa que utiliza el nombre ROM Core Generator, inicializar con el archivo. COE.

El ROM_form.coe archivo de plantilla se ha actualizado en el directorio / lab5/Assembler para reflejar el formato del COE trabajo necesarios para el bloque de memoria v2.7 Generator. En el nuevo formato, sólo dos parámetros pueden ser especificados: memory_initialization_radix y memory_initialization_vector. Usted puede acceder al bloque de memoria v2.7 Generator de hoja de datos a través del núcleo generador para obtener más información..

❶ Para abrir el Xilinx ISE software, seleccione **Start → Programs → Xilinx ISE Design Suite 10.1i → ISE → Project Navigator**

❷ Select **File → Open Project** y seleccione **coregen.ise**

Verilog usuarios: buscar en `c:\xup\fpflow\labs\verilog\lab5\coregen`

VHDL usuarios: buscar en `c:\xup\fpflow\labs\vhdl\lab5\coregen`

❸ Abra el programa en el archivo psm (localizado en el directorio de proyectos) y complete la tarea #3 escribiendo un pedazo de código de software que se hará eco de nuevo a HyperTerminal lo que se escribe en el teclado.

Nota: Consulte los comentarios en el programa para obtener instrucciones

- 4 Abra una ventana de comandos, vaya al directorio que contiene el programa, y montamos el programa introduciendo el comando siguiente en el símbolo del sistema de programa:> kcpasm3

Nota: el ensamblador genera varios archivos incluyen. COE, que se utilizará, generado a partir de una memoria de núcleo generador

- 5 Haga doble-click en **Create New Source** para abrir el asistente de nueva fuente, seleccione el **IP (CoreGen & Architecture Wizard)** y escriba *program* en nombre de archivo (vea Figura 5-1), y haga click en **Next**.

Si usted no ve el proceso de creación de Fuente Nueva, asegúrese de que una fuente de HDL archivo está seleccionado en la ventana de Fuentes en el Project.

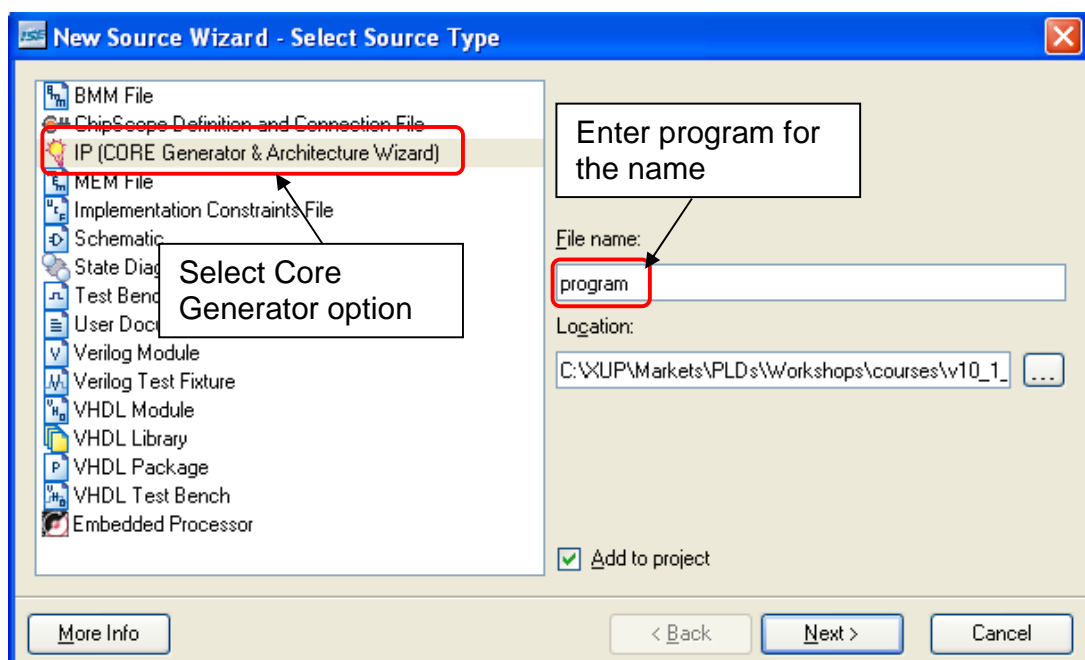


Figura 4.5-1. Nuevo cuadro de diálogo Fuente

- En la selección de tipo de núcleo expanda **Memories & Storage Elements**, expanda **RAMs & ROMs**, y seleccione **Block Memory Generator v2.7** (vea Figura 4.5-2). Haga Click en **Next** luego **Finish**.

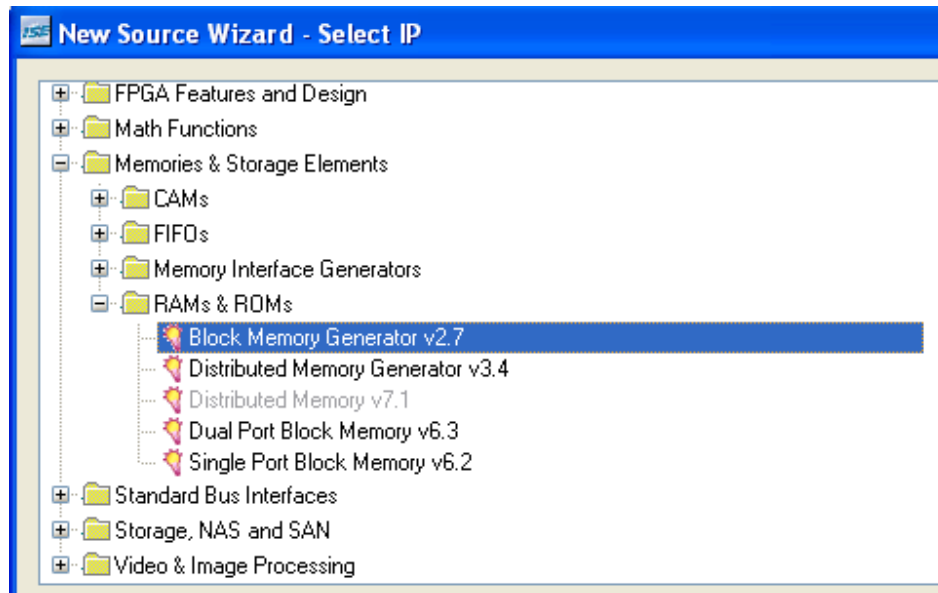


Figura 4.5-2. Seleccione el cuadro de diálogo Tipo de núcleo

- **Configurar el único bloque de memoria del puerto**

Especifique el nombre de la memoria escribala en la **Pág. 1** y haga click en <Next>

Component Name: *program*
Memory Type: Single Port ROM

Especifique el Puerto A los paramentos en la **Pág. 2** y de click en <Next>.

- Read Width: **18**
- Read Depth: **1024**
- Enable: Always Enabled

Especifique los parámetros del Puerto Ben la **Pág. 3** y de click en <Next>.

- Read Width: **18**
- Read Depth: **1024**
- Enable: **Always Enabled**

③ Haga Click en **Load Init File** y escoja el archivo **PROGRAM.COE** del directorio

④ Haga Click en botón de **Show** y seleccione **memory_initialization_vector** de la caja de abajo para ver el contenido que se leerá después de configurar la FPGA.

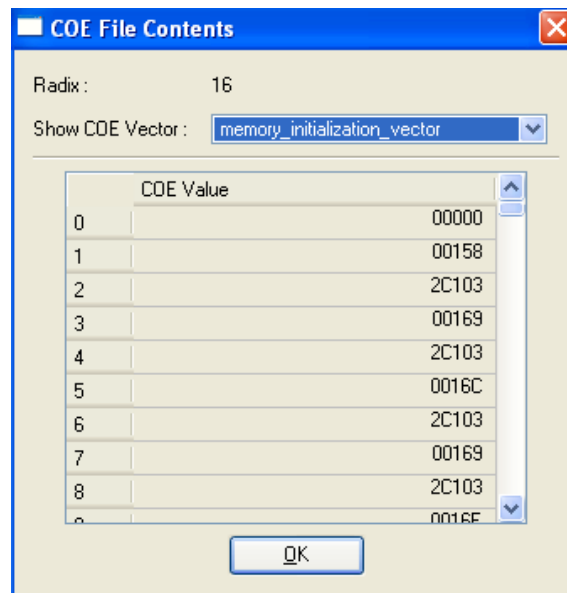


Figura 4.5-3. Bloquear el contenido de inicialización de RAM

⑤ Haga Click en **<next>** y luego en **<finish>**

Nota: el archive *program.xco* automáticamente se adjuntara en el ISE Project.

4.5.5 UNA INSTANCIA DE UN NÚCLEO DE RAM BLOCK EN FUENTE VERILOG PASO 3A

Los usuarios de VHDL: Ir al paso 3b, "Bloque de instancias RAM Core en fuente VHDL"

Una instancia de la base que ha generado anteriormente en `loopback.v`.



- 1 En la ventana de Project en Sources, haga doble-click en `loopback.v`

El archivo `loopback.v` se abrirá en la ventana de editor de texto.

- 2 Seleccione **Edit** → **Language Templates**



El de instancias de plantilla para el núcleo se encuentra en la sección de COREGEN de esta ventana.

Expanda **COREGEN**, expanda **VERILOG Component Instantiation**, y seleccione **program** (vea Figura 4.5-3)

Si el núcleo generador de instancias de plantilla no aparece, entonces usted puede acceder a la plantilla (ver archivo. Veo) en el directorio Del proyecto.

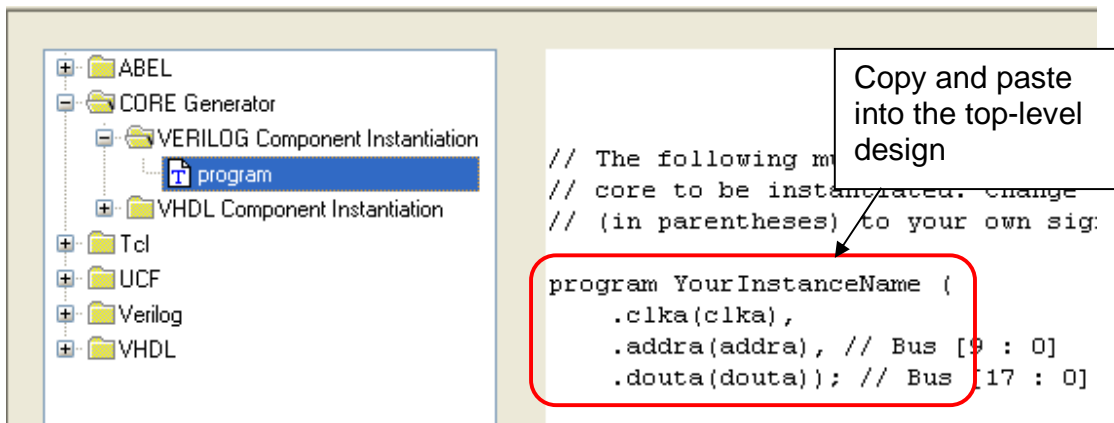


Figura 4.5-3a. Plantillas de idiomas

- 4 Copie y pegue las **plantillas en el archivo** `loopback.v` *bajo el comentario*, “//Instantiate ROM here”
- 4 Edite las **instancias** para que se vea así:

```
program my_program (
    .clka(clk55MHz),
    .addra(address),
    .douta(instruction)
);
```

- ⑥ Seleccione **File** → **Save**

4.5.6 INSTANCIACIÓN BLOQUE RAM CORE EN FUENTE VHDL PASO 3B

Verilog usuarios: Ir al paso 4, "Realizar la simulación del comportamiento."



Una instancia de la base que ha generado en el paso 2 en `fifo_2048x8.vhd`.

- ① En la fuente en la ventana de Project, haga doble-click en *loopback.vhd*

El archivo *loopback.vhd* se abra en la ventana de editor de texto.



- ② Seleccione **Edit** → **Language Templates**

La creación de instancias de plantilla para el núcleo se encuentra en la sección de COREGEN de esta ventana.

Expanda **COREGEN**, expanda **VHDL Component Instantiation**, y selección *program*. (vea **Figura 5-3b**)

Si el núcleo generador de instancias de plantilla no aparece, entonces usted puede acceder a la plantilla (ver archivo. VHO) en el directorio del proyecto.

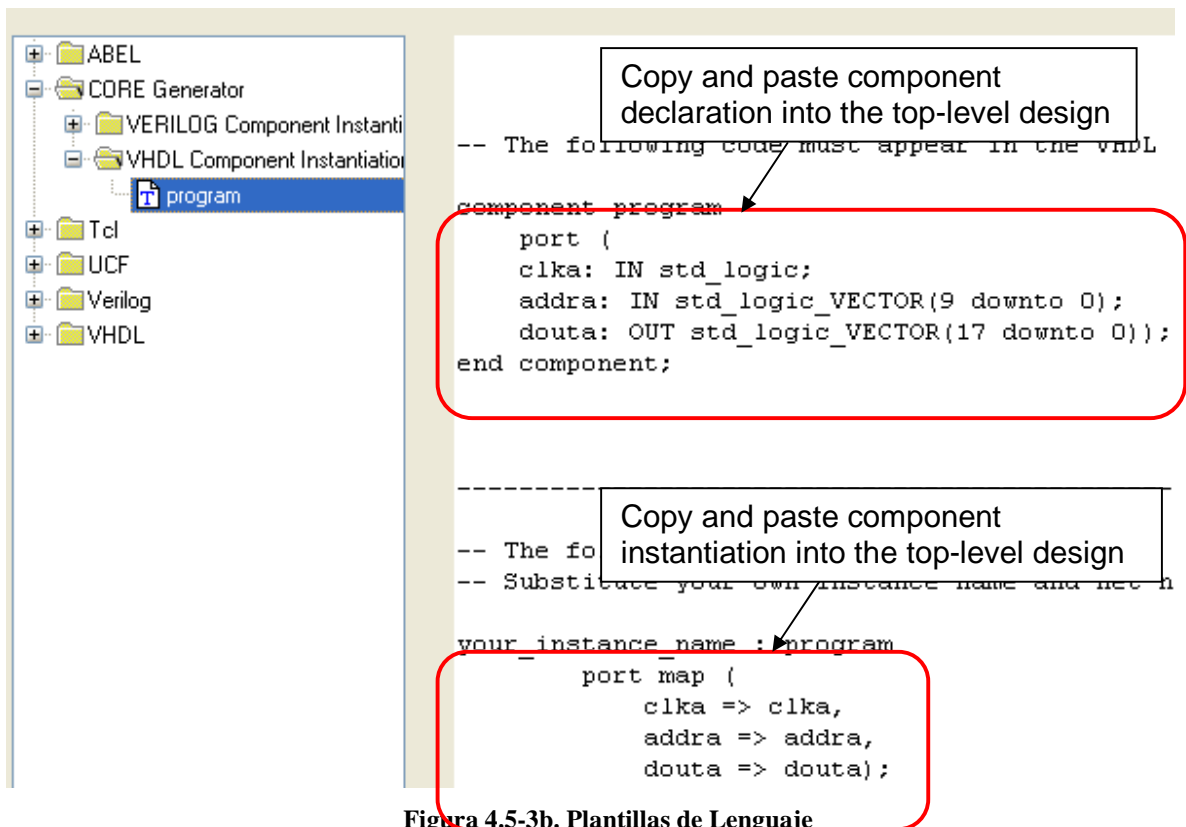


Figura 4.5-3b. Plantillas de Lenguaje

- ④ Copie y pegue la **plantilla** en el archivo *loopback.vhd*

Pegue el **component declaration** en la cabecera de la arquitectura después del comentario “---- Insert component declaration for program here”.

Pegue el **component instantiation** en la cabecera de la arquitectura después del comentario “-- insert component instantiation for program here”.

- ④ Edite la **instancia** como se muestra a continuación que conecta los puertos duales del bloque ROM para el diseño:

```
my_program : program
port map
(
  clka => clk55MHz,
  addra => address,
  douta => instruction
);
```

- ⑥ Seleccione **File** → **Save**

4.5.7 REALIZAR LA SIMULACIÓN DEL COMPORTAMIENTO

PASO 4



Simular el diseño utilizando el banco de pruebas previstas.

- 1 Abra *testbench.v/vhd* y revise la funcionalidad del banco de prueba.

Algunas cosas pasaran acerca del banco de prueba:

- ✓ **Los interruptores de banco de pruebas cambia, espera a que algunos ciclos de reloj, y el control de los indicadores LED para ver si coinciden con la configuración del switch**

El RS232 TX y RX señales no son impulsados por el banco de pruebas en este ejemplo

- 2 en las fuentes de Ventana de Proyecto, seleccione *program.xco*

en los procesos de la ventana Fuente, ampliar la caja de herramientas CORE y haga doble clic en **Ver Verilog / VHDL Functional /VHDL Functional Model**

Este archivo hace referencia a los modelos de la biblioteca de la simulación XilinxCoreLib y se utiliza de forma automática cuando se ejecuta la simulación del comportamiento desde el interior de la Navigator de proyecto de software de la ISE TM.



Los usuarios de VHDL: Si el archivo no aparece en el editor de texto, haga clic en Ver VHDL Functional Model y seleccione Abrir, sin actualizar.



Uso de la testbench.v o testbench.vhd archivo, ejecutar una simulación del comportamiento de 50000 ns. Ver las formas de onda para confirmar que el núcleo está conectado.

- ❶ Cambiar a modo de simulación mediante la selección de simulación del comportamiento
- ❷ Introduzca una simulación en tiempo de ejecución de 50000 ns en las propiedades del modelo de comportamiento. Haga clic en Simular modelo de comportamiento en el proceso de Xilinx ISE Simulator para acceder a las propiedades.
- ❸ Ejecutar la simulación del comportamiento, haga doble clic sobre Simular modelo de comportamiento
- ❹ Examinar las formas de onda para comprobar que la configuración de los interruptores se hacen eco de los LEDs.
- ❺ Cambie de nuevo al modo de ejecución.

4.5.8 PRUEBE LA APLICACIÓN EN HARDWARE PASO 5



Abra una sesión de HyperTerminal. Generar el flujo de bits y descargar a la Junta Digilent y probar la aplicación.

- ❶ Inicie una sesión de HyperTerminal y configurar con los siguientes valores
 - COM Properties
 - ✓ Baud rate: 9600
 - ✓ Stop bits: 1
 - ✓ Parity: none

- ✓ Flow control: none
Settings – ASCII Setup
 - ✓ Append line feeds to incoming line ends: checked
- ➊ loopback.v / VHD seleccionado como el Con el diseño de nivel superior, haga doble clic en Administrar configuración del proyecto (IMPACT), en proceso de configurar el dispositivo de destino para generar el flujo de bits e invocar de IMPACT.
 - ➋ Haga Click en <Finish> dejando intacto las opciones de **Configure devices using boundary-scan (JTAG)**
 - ➌ Asignar el loopback.bit al Spartan-3 xc3s500e y puentear el PROM y CPLD
 - ➍ Haga click derecho en el dispositivo Spartan-3e en impact y seleccione el programa haga. Click en <OK>.

Nota: Usted debería ver el mensaje "Xilinx Rules" Que aparece en la ventana de HyperTerminal. Cualquier mensaje que se escribe a través del teclado también se mostrará.

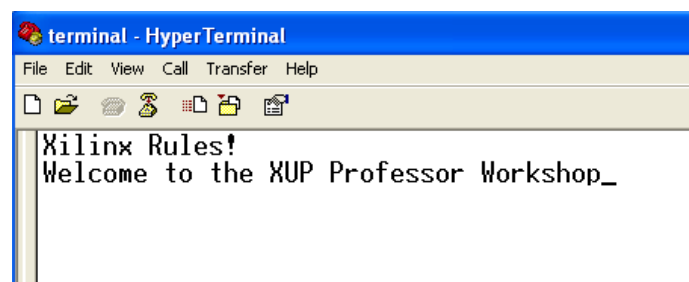


Figura 5-4. Ver la salida de la ventana de HyperTerminal

CONCLUSIÓN

Usted puede utilizar el sistema generador CORE™ para configurar y generar núcleos. Puede acceder a la creación de instancias de plantillas para diseños Verilog o VHDL utilizando las plantillas de idiomas o de la VHO VEO y archivos generados en el directorio del proyecto. Los modelos de simulación funcional para los núcleos se proporcionan en la biblioteca XilinxCoreLib.

Práctica 6: Chipscope depuración de laboratorio

Orientación XUP Spartan-3E

4.6 CHIPSCOPE DEPURACIÓN DE LABORATORIO

4.6.1 INTRODUCCIÓN

En la guía de laboratorio a través del proceso de inserción de chipscope núcleos Pro en su diseño y realizar una verificación sobre el chip.

4.6.2 OBJETIVOS

- ✓ Después de completar este laboratorio, usted será capaz de:
- ✓ Crear un nuevo Chipscope-Pro fuente en ISE
- ✓ Crear ILA y núcleos icono mediante Chipscope-Pro y la inserta en un diseño PicoBlaze
- ✓ Especifique las opciones de activación en Chipscope Analyzer
- ✓ Descargar el flujo de bits y ejecutar el diseño de hardware
- ✓ Realizar una verificación sobre el chip y ver las formas de onda en Chipscope Analyzer

4.6.3 PROCEDIMIENTO

En este laboratorio, se modifica una aplicación de software dirigidos a PicoBlaze y el uso Chipscope-Pro para realizar una verificación sobre el chip.

Este laboratorio consta de cuatro pasos principales:

1. Crear una nueva fuente Chipscope
2. Conectar y configurar un núcleo de la ILA
3. Configure las opciones de activación Chipscope Analyzer
4. Realizar una verificación sobre el chip

Después de cada instrucción general para un procedimiento determinado, se encuentra el paso de acompañamiento-por paso y cifras ilustran los que se detallan para la realización de la instrucción general. Si usted se siente confiado acerca de una instrucción específica, no dude en saltarse el paso por paso y pasar a la instrucción general siguiente en el procedimiento.

Nota: Si usted no puede completar el laboratorio en este momento, usted puede descargar los archivos de laboratorio de este módulo desde el sitio del Programa en la Universidad de Zilina <http://www.xilinx.com/univ>

4.6.4 DISEÑO GENERAL

Usted ampliar el laboratorio de diseño de 5 añadiendo una chipscope fundamentales de la ILA para el bus de salida PicoBlaze A continuación, podrás configurar el disparador para capturar los datos cuando el texto se introduce a través de HyperTerminal. Usted debe ver el texto resultante se muestra en ChipScope cuando el buffer está lleno.

CREAR UN CHIPSCOPE NUEVA PRO PASO FUENTE 1

Crear un nuevo ChipScope-Pro proyecto a través del Navegador de proyectos.

- ✓ Abra el software Xilinx ISE seleccionando **Inicio** □ **Programas** □ **Xilinx ISE Design Suite - 10.1i ISE -Project Navigator**
- ✓ Abra el proyecto seleccionando **Archivo** □ **Abrir proyecto**

Usuarios Verilog: Vaya a *C: \ xup \ fpgaflow \ labs \ verilog \ lab6 \ chipscope*

Los usuarios de VHDL: Vaya a *C: \ xup \ fpgaflow \ labs \ VHDL \ lab6 \ chipscope*

- ✓ *ChipScope.ise* Seleccione y haga clic en Abrir

- ✓ Crear un nuevo **Chipscope Definición y fuente de conexión** mediante la selección de proyectos **fuentes nueva** y entrar en el **loopback_cs** nombre. <Siguiete> Haga clic para continuar.

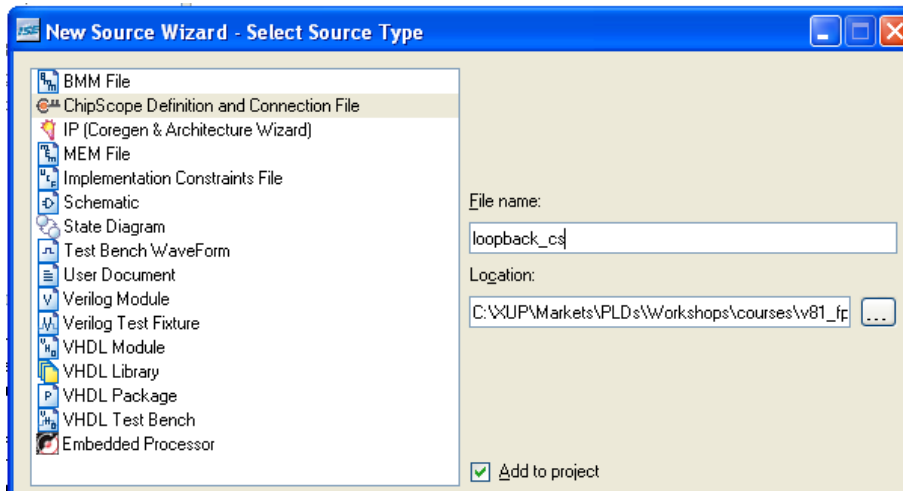


Figura 4.6-1. Nuevo cuadro de diálogo Fuente

- ✓ **Loopback** Seleccione la fuente. Haga clic en Next, Siguiete "y luego <Finish>. Chipscope A-Pro fuente se añadirá a las fuentes en la ventana del proyecto.

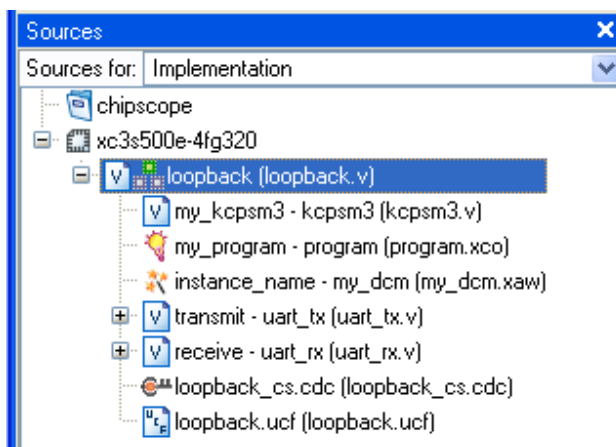


Figura 4.6-2. Chipscope Definición y conexión (. CDC) añadido a Verilog proyecto

4.6.6 CONFIGURAR Y CONECTAR UN PASO ILA CORE 2



Conecte la base de la ILA a la salida de PicoBlaze.

Haga doble clic en el archivo **loopback_cs.cdc** en las fuentes en la ventana de proyecto para abrir el proyecto de inserción.

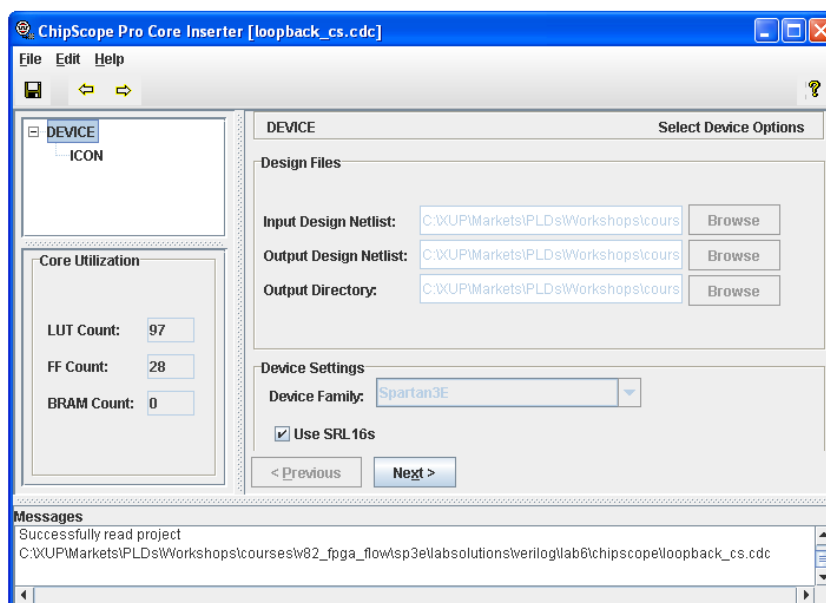


Figura 4.6-3. Chipscope-Pro Core de inserción

Nota: Los proyectos guardados en el núcleo de inserción mantener toda la información pertinente acerca de los archivos fuente, archivos de destino, los parámetros y configuraciones básicas.

Haga clic en Next, Siguiente”. Dejando el reloj JTAG BUFG Deshabilitar la opción de inserción desactivada, haga clic en Nueva Unidad de la ILA. Aviso en la ventana de la izquierda cómo una instancia de la base de la ILA, U0: ILA, se añade al sistema.

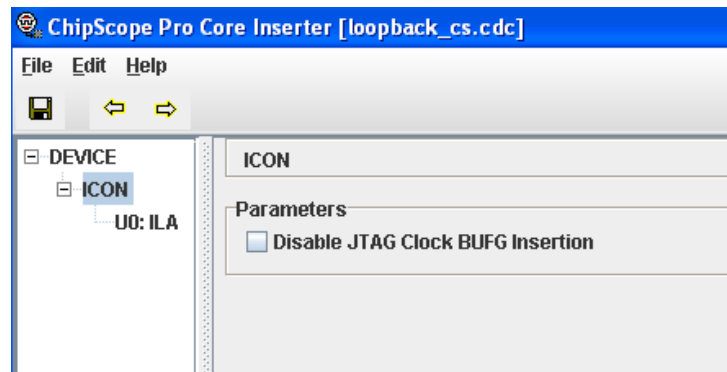


Figura 4.6-4. Insertar un nuevo analizador lógico integrado (ILA), Unidad

Nota: la desactivación de la BUFG reloj JTAG inserción causas de las herramientas ISE para enrutar el reloj JTAG usando los recursos de enrutamiento normal en vez de reloj mundial de enrutamiento recursos. Esta opción debería estar seleccionada únicamente si los recursos de enrutamiento mundiales son escasos.

Haga clic en Next, Siguiente "para configurar los parámetros de activación

Cada ILA o ILA / ATC núcleo puede tener hasta 16 puertos de activación independiente que puede tener la configuración de forma independiente. Los puertos de activación individual son los autobuses que se componen de señales individuales o bits que puede ir desde 1 hasta 256 bits. Cada puerto disparador puede ser conectado a 1 a 16 unidades de partido.

Una unidad de partido es una comparación que se conecta a un puerto disparador y se utiliza para detectar eventos en ese puerto disparador. Los resultados de las unidades de uno o más partido se combinan para formar el total de eventos condición de disparo que se utiliza para controlar la captura de datos. Las comparaciones diferentes funciones o del partido que puede ser realizado por las unidades de selección de puertos de activación depende del tipo de partido unidad. La ILA y la ILA / CCI núcleos de apoyo a los seis tipos de unidades de partido. En este laboratorio, se configurar el núcleo de la ILA para activar a través de algunas señales de control de UART.

Ajuste los siguientes parámetros ILA gatillo como sigue a continuación, haga clic en Next, Siguiente”

Entrada del disparador y del partido Unidad de Configuración

- ✓ Número de puertos de disparo de entrada: 3

Trigger Puerto	Trigger Ancho	Match # Unidades	Contra Ancho	Tipo de concordancia
TRIG0	1	1	Personas de movilidad reducida	base
TRIG1	1	1	Personas de movilidad reducida	Básico
TRIG2	1	1	Personas de movilidad reducida	Básico

Condición de activación Configuración

- ✓ Habilitar Trigger Secuenciador: Revisado (Esto le permite especificar una secuencia de eventos que permitan disparo)
- ✓ Número máximo de niveles de Secuenciador: 2
- ✓ Condición de Configuración de almacenamiento de calificación
- ✓ Habilitar el almacenamiento Calificación: Revisado (Esto le permite especificar qué datos se almacenan en el búfer interno)

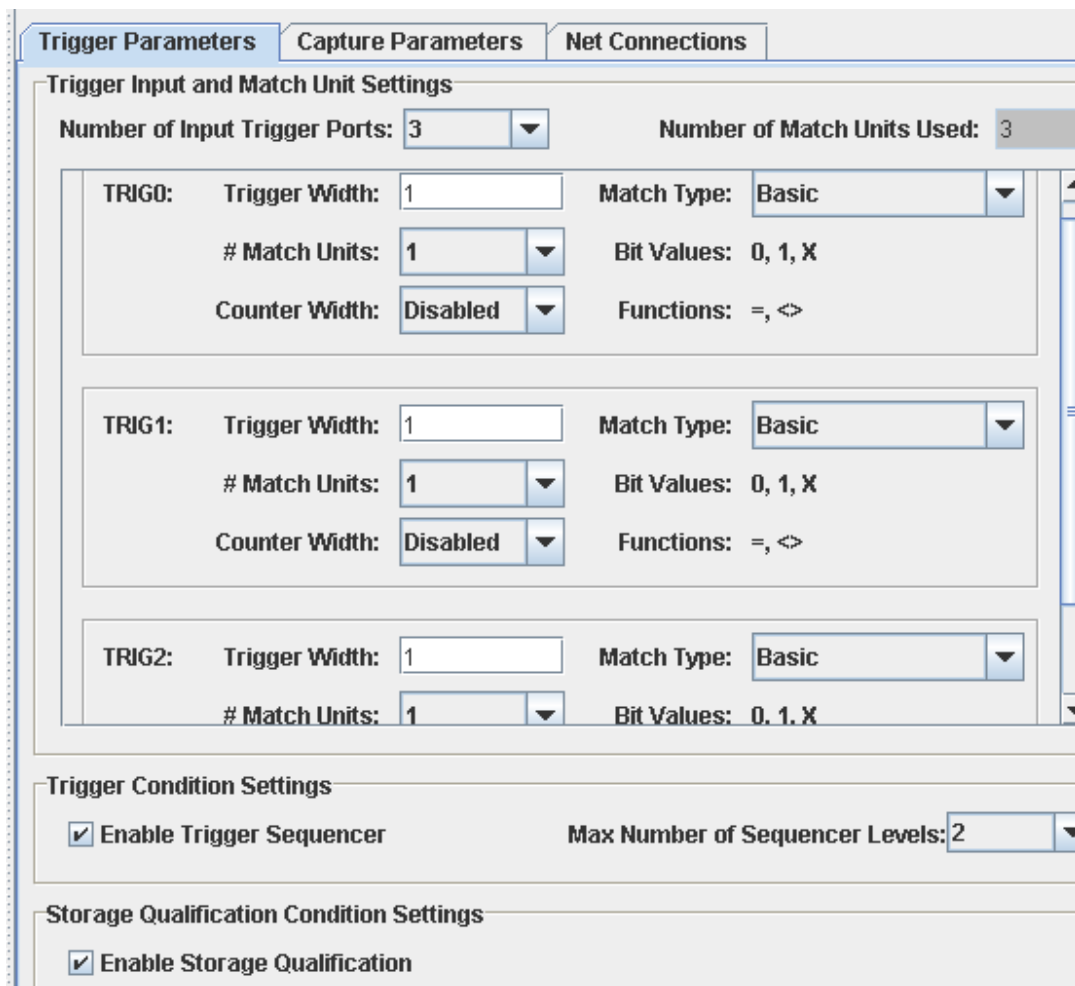


Figura 4.6-5. Especificar los parámetros de activación

El número máximo de palabras de datos de ejemplo que el núcleo de la ILA puede almacenar en el buffer de la muestra se denomina *profundidad de datos*. La profundidad de los datos determina el número de bits de datos aportados por ancho de cada unidad de bloque de memoria RAM utilizada por la unidad de la ILA. El número máximo de datos ejemplos de palabras que se pueden capturar depende del número y tamaño de bloque de memoria RAM, que varía de acuerdo a la familia de dispositivos y la densidad.

- ✓ Establecer los parámetros de captura de los siguientes y haga clic en Next, Siguiente "

- ✓ Profundidad de los datos: 512
- ✓ En la muestra: Flanco de reloj
- ✓ Mismos datos que la activación del puerto: sin control
- ✓ Ancho de datos: 8

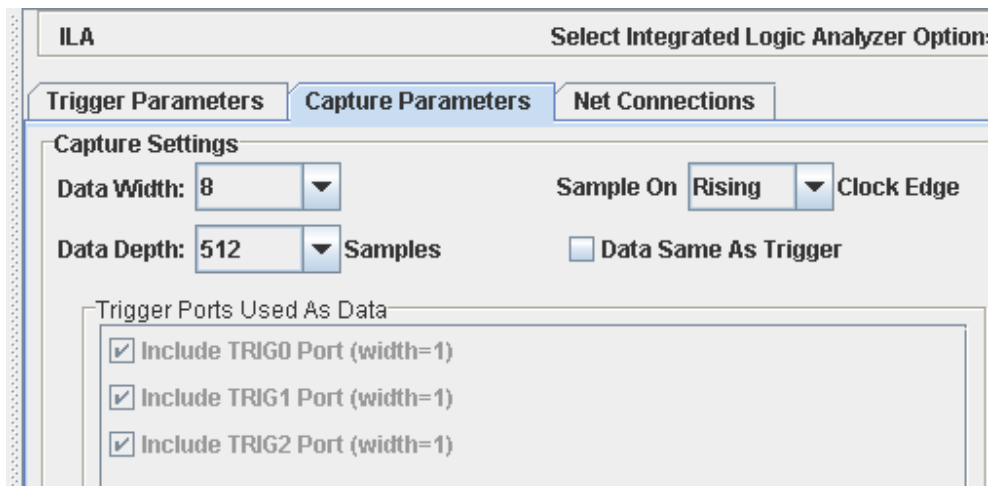


Figura 4.6-6. Especifique los parámetros de activación

La red de conexiones ficha le permite elegir la señal para conectarse a la base de la ILA. Si el disparo es independiente de datos, reloj, disparo, y los datos deben ser especificados. Conexiones que no se han hecho aparecerá en rojo.

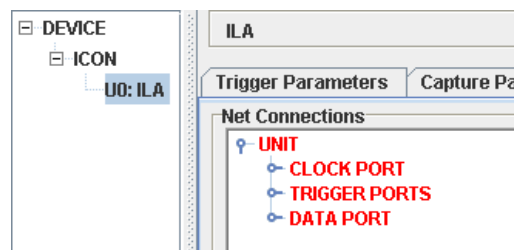


Figura 4.6-7. Conexiones de Red Desconectado

- ✓ Haga clic en la ficha **Conexiones Modificar**

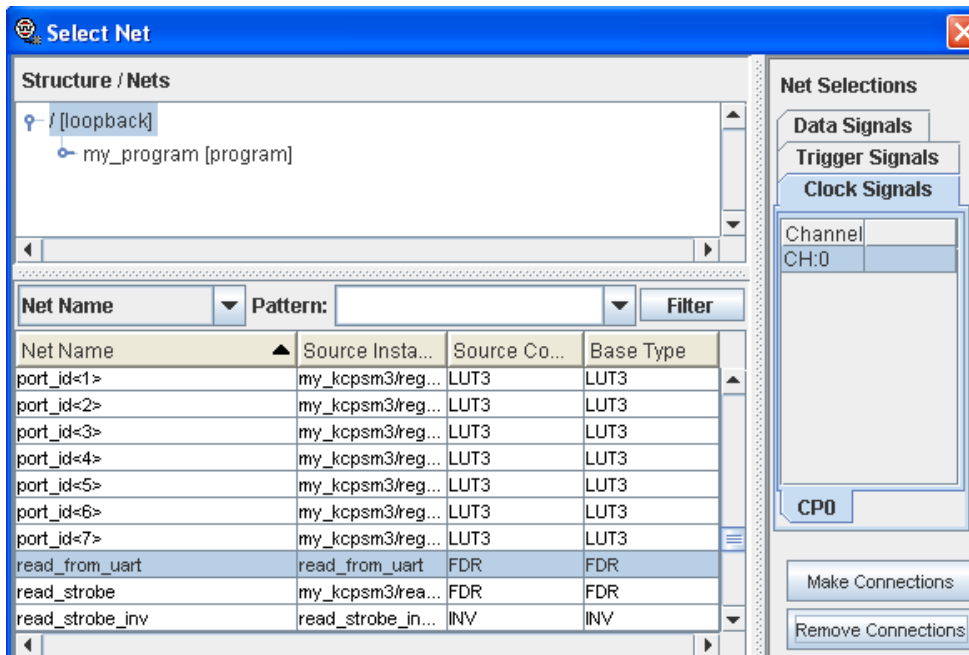


Figura 4.6-8. Conexiones de red

El cuadro de diálogo **Seleccionar Net** proporciona una interfaz fácil de elegir para conectarse a las redes de la ILA, la ILA / ATC o núcleos de CTM2. La estructura jerárquica del diseño se puede recorrer utilizando la estructura / panel de Redes. Las redes de todo el diseño de la jerarquía en la estructura seleccionada aparecen en la tabla en el panel inferior izquierdo. Las señales de reloj y de activación / señales de datos partituras ilustran las conexiones de red entre el diseño y el núcleo de la ILA.

Con las **señales del reloj** en la ficha **Red**, seleccione **selecciones**, resalte la entrada de **clk55MHz** en la lista de redes y haga clic en el **botón** Crear **Conexiones** para conectar la señal del reloj en el diseño para el puerto de reloj del núcleo de la ILA.

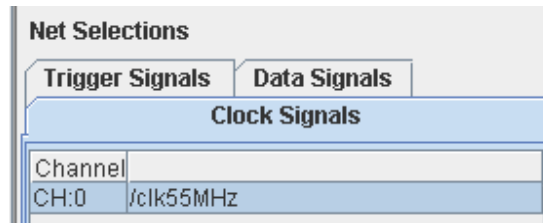


Figura 4.6-9. Conecte el reloj

Haga clic en la pestaña de activación de señales, y conecta los tres puertos de activación de la siguiente manera:

- ✓ TP0: data_present (esta señal indica que los datos se presentan en el módulo de uart_rx)
- ✓ TP1: read_from_uart (entrada a uart_rx que indica que una operación de lectura se producirá)
- ✓ TP2: write_to_uart (entrada a uart_tx que indica que una operación de escritura se producirá)

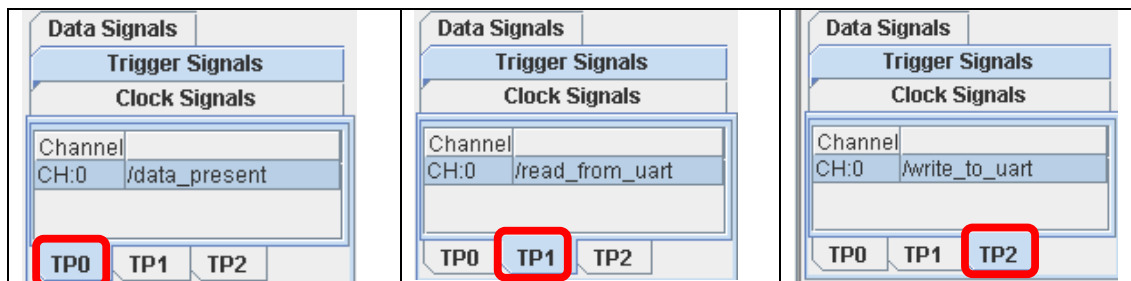


Figura 4.6-10. Conectar los puertos de disparo

- ✓ Haga clic en la ficha **señales de datos** y conectar el puerto de salida del controlador de PicoBlaze al puerto de datos del núcleo de la ILA (ver Figura 4.6-11), y haga clic en <OK>:

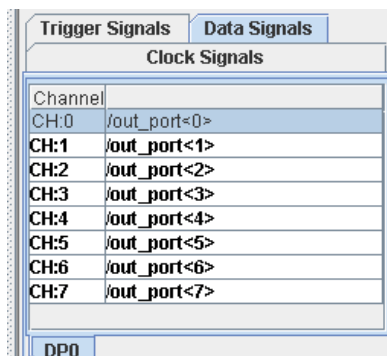


Figura 6-11. Conecte el puerto de salida PicoBlaze

Usted notará que el reloj, Trigger, y puertos de datos en conexiones de red se destacan en negro, con indicación de conexiones válidas. Haga clic en **Return to Project Navigator** y guardar el archivo.

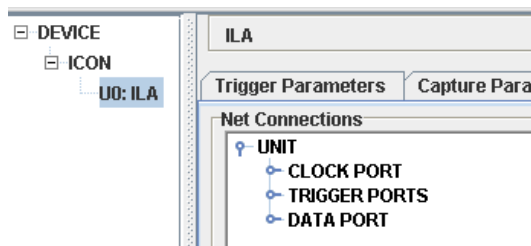


Figura 6-12. Conexión entre el diseño y el núcleo de la ILA de plantilla

4.6.7 ESPECIFIQUE CHIPSCOPE ANALYZER OPCIONES PASO 3

Va a descargar el flujo de bits utilizando Chipscope y configurar el núcleo de la ILA para activar cuando el UART lee el texto del HyperTerminal.

- ✓ Con el alto nivel de archivo (loopback.v / VHD) seleccionado, haga doble clic en **Analyze Design Using Chipscope** en la ventana de Procesos

- ✓ Conecte el cable de descarga en el puerto paralelo del PC y conexión JTAG de la Spartan-3E bordo, y luego encienda la tarjeta.
- ✓ Haga clic en el botón **Open Cable/Search JTAG Chain**

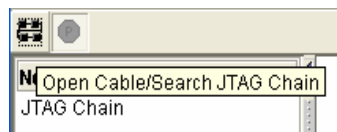


Figura 4.6-13. Establecer conexión JTAG

- ✓ Haga clic en <Accept.>, señalando que el Spartan-3E dispositivo es el primer dispositivo de la cadena de tres dispositivos.

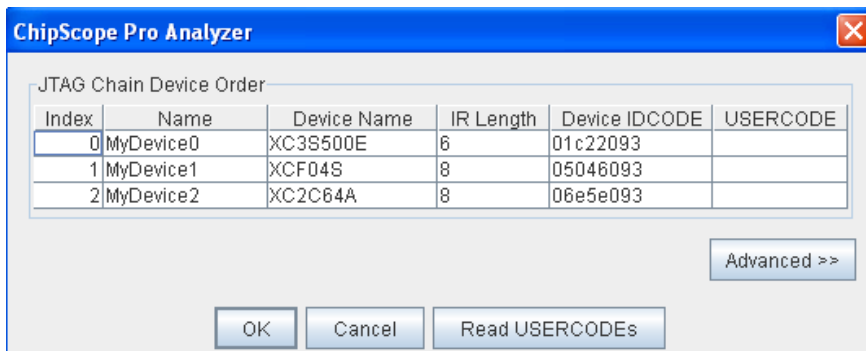


Figura 6-14. Impacto detecta dispositivos en la cadena JTAG

- ✓ Haga clic en el dispositivo **xc3s500e** y seleccione Configurar.
- ✓ **Seleccione Nuevo**, haga clic en **Archivo** y seleccione el archivo loopback.bit indirecto desde el directorio del proyecto. <OK> Haga clic en.

Cada ILA Pro Chipscope, ILA / ATC, y el núcleo IBA tiene su propia ventana de configuración de activación, lo que proporciona una interfaz gráfica para el usuario para configurar los factores desencadenantes. El mecanismo de activación en el interior de cada núcleo Pro Chipscope puede ser modificado en tiempo de ejecución sin tener que recompilar el diseño. Componentes para el mecanismo de activación:

- ✓ Funciones del partido: Define el partido o el valor de la comparación de cada unidad de partido.
- ✓ Condiciones de activación: Define la condición de activación global basada en una ecuación binaria o una secuencia de uno o más partido de las funciones.
- ✓ La configuración de captura: Define cuántas muestras de la captura, como las ventanas de captura de muchos, y la posición del gatillo en las ventanas del mismo.

En este diseño, se le dispara a la configuración de la captura de texto en el puerto de salida PicoBlaze, después de haber entrado a través de HyperTerminal.

- ✓ Vaya a **Archivo** → **Importar** y **loopback_cs.cdc** importar el archivo desde el directorio del proyecto. Este archivo contiene los nombres neto de la PicoBlaze señales conectadas al gatillo y puerto de datos del núcleo de la ILA.
- ✓ Especificar las unidades del partido de la siguiente manera:
 - ✓ M0: TriggerPort0 (data_present): Valor 1
 - ✓ M1: TriggerPort1 (read_from_uart): Valor 1
 - ✓ M1: TriggerPort1 (write_to_uart): Valor 1

Match Unit	Function	Value	Radix	Counter
M0:TriggerPort0	==	1	Bin	disabled
/data_present		1		
M1:TriggerPort1	==	1	Bin	disabled
/read_from_uart		1		
M2:TriggerPort2	==	1	Bin	disabled
/write_to_uart		1		

Figura 4.6-16. Instalación de las Unidades del partido

Haga clic en el campo en la **ecuación de disparo Estado**, establecer la ecuación de **M0** → **M1** en la ficha **del secuenciador**, a continuación, haga clic en <OK>.

Trigger Condition: TriggerCondition0

Boolean Sequencer

Number of Levels: 2 Use Contiguous Match Events Only

Level	Match Unit	Negate
1	M0	<input type="checkbox"/>
2	M1	<input type="checkbox"/>

Trigger Condition Equation

M0 -> M1

Figura 4.6-17. Condición de la ecuación de disparo

Compruebe el campo situado junto a **Almacenamiento**, **Clasificación**, seleccione la **ecuación y**, de verificación y **M2**. <OK> Clic. De este modo, el núcleo de la ILA para capturar datos en el búfer de datos sólo cuando está presente, y no en cada flanco de reloj único.

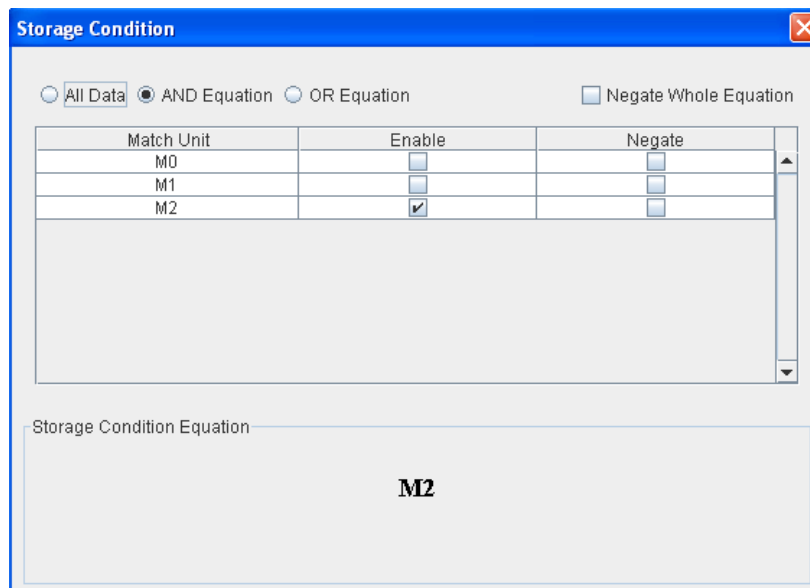


Figura 4.6-18. La ecuación de almacenamiento de calificación

4.6.8 Realizar un chip Paso Verificación 4

Usted brazo del gatillo y ver las formas de onda de los datos capturados.

- ✓ Realizar las siguientes acciones para crear un autobús desde el 8 señales out_port
- ✓ Seleccione las señales de out_port <0> a través de out_port <7> por lo que se puso de relieve que
- ✓ Haga clic en las señales de relieve y seleccione **Agregar a** **A New Bus** para crear el nuevo autobús **output_port**

Bus/Signal	X	O
/out_port		
/out_port<0>	0	0
/out_port<1>	0	0
/out_port<2>	0	0
/out_port<3>	0	0
/out_port<4>	0	0
/out_port<5>	0	0
/out_port<6>	0	0
/out_port<7>	0	0

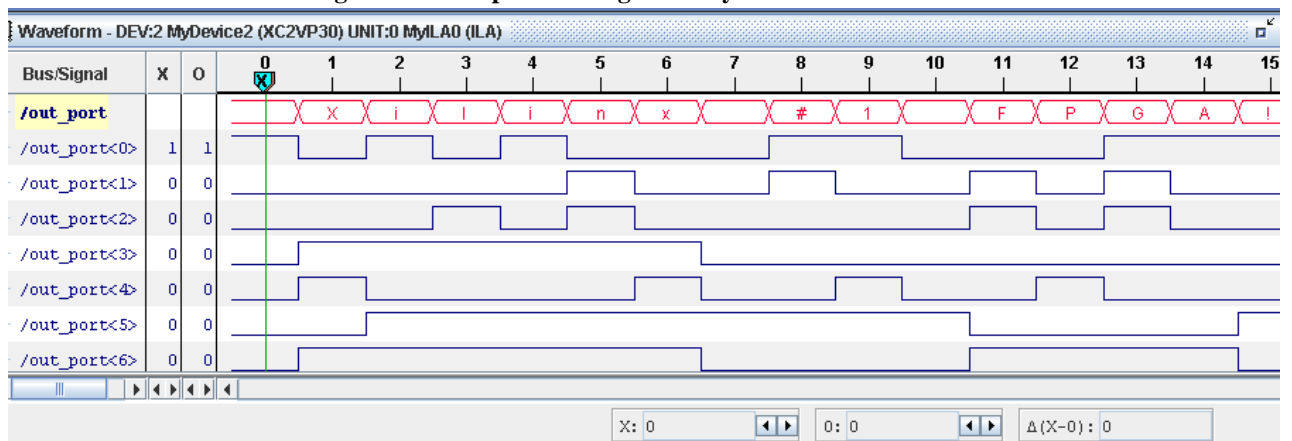
Figura 4.6-19. Crear un autobús

- ✓ Ajuste de la profundidad de amortiguamiento a 16

▶ Capture	Type: <input type="text" value="Window"/>	Windows: <input type="text" value="1"/>	Depth: <input type="text" value="16"/>	Position: <input type="text" value="0"/>
	Storage Qualification: <input type="text" value="M2"/>			

Figura 4.6-20. Seleccione Buffer Depth

Figura 4.6-21. Aplicar configuración y activación del brazo



4.6.9 CONCLUSIÓN

Ha insertado el ILA y núcleos de icono en el diseño PicoBlaze, establecer las condiciones de activación en Chipscope Analyzer, realizó una verificación sobre el chip, y analizaron las formas de onda en Chipscope-Pro Analyzer.

CONCLUSIONES

FPGA nos ha permitido explorar el espacio de diseño de este tipo de sistemas, analizando la influencia de distintas alternativas arquitecturales y diferentes opciones de implementación con objeto de optimizar el uso de los recursos que proporcionan los dispositivos actuales.

Aunque no son capaces de superar en velocidad de proceso a un computador, tienen la ventaja de que pueden integrar todo un sistema computacional dentro de un único chip, lo cual supone un ahorro en espacio, consumo y calor. Por otra parte son sistemas mucho más abiertos que los microprocesados son reprogramables.

Al poderse emplear HDLs para programarlos, tienen la ventaja de que en algunas aplicaciones su ejecución es mucho más rápido, puesto que la FPGA va a comportarse como un circuito electrónico hecho a la medida de la aplicación a satisfacer. Cabe mencionar que muchos modelos de procesador se han probado primero en FPGA antes de construir los chips, lo que indica la potencia de este tipo de dispositivos.

Como hemos visto en los campos de aplicación de las FPGA llegan hasta donde podamos imaginar: visión artificial, telecomunicaciones, proceso digital de señales, robótica, tarjetas gráficas..., el límite está en la imaginación del diseñador.

Otra de sus ventajas es la interfaz de programación, casi todos los dispositivos se programan mediante interfaz, que además permite comunicarlos e incluso conectarlos en serie para programar varios dispositivos conectados al mismo bus.

En muchos FPGA, estos componentes lógicos programables, también incluyen elementos de memoria, los cuales pueden ser simples flip-flops (biestables) o bloques de memoria más complejos.

Un FPGA que tiene una gran cantidad de canales de interconexión, cualquier circuito de aplicación específica puede ser implementado, siempre y cuando esta disponga de los recursos necesarios.

El mercado de los FPGA se ha colocado en un estado donde hay dos productores de FPGA de propósito general que están a la cabeza del mismo, y un conjunto de otros competidores quienes se diferencian por ofrecer dispositivos de capacidades únicas. Xilinx, es uno de los dos grandes líderes en la fabricación de FPGA.

Los laboratorios nos permiten verificar el funcionamiento de nuestra descripción de hardware sin necesidad de pasar a la síntesis.

Todos los simuladores importantes convierten nuestro código HDL en un binario ejecutable que se comporta como nuestro hardware simulando los cambios de señales. A través del uso de testbenches (bancos de prueba) es posible realizar una verificación del funcionamiento de nuestro hardware e informar si algo no funciona como se esperaba.

Como conclusión final decir que nos ha resultado bastante interesante la realización de este trabajo, y nos hemos dado cuenta de las posibilidades del empleo de este tipo de dispositivos, así como la potencia de los mismos.

PLAN DE TRABAJO

Proyección, análisis, y generalización de la idea de tesis	4 Semanas
Prácticas de laboratorio de FPGAS de Xilinx	5 semanas
Desarrollo de tesis	4 Semanas

BIBLIOGRAFIA

http://books.google.com.ec/books?id=vYgweLqkRzMC&dq=fpga&printsec=frontcover&source=bl&ots=C9RdPHlsHX&sig=EaoYStLZ_6olXctB1a71xYPEkIE&hl=es&ei=j0LBSqL2JNWf8AaYyLipAQ&sa=X&oi=book_result&ct=result&resnum=4#v=onepage&q=&f=false

http://www.ni.com/fpga/esa/what_is.htm

<http://www.latticesemi.com/products/fpga/index.cfm>

<http://translate.google.com.ec/translate?hl=es&sl=en&u=http://www.webopedia.com/TERM/F/FPGA.html&ei=yUfBSua5MY-18AbiyJzDAQ&sa=X&oi=translate&resnum=9&ct=result&prev=/search%3Fq%3DFPGA%26hl%3Des%26sa%3DG>

<http://www.webopedia.com/TERM/F/FPGA.html>

<http://fpgalibre.sourceforge.net>

http://www.cio.mx/3_enc_mujer/files/extensos/Sesion%202/S2-ING03.pdf

http://www.miky.com.ar/fpga_2004.pdf

http://www.salle.url.edu/~gpazienza/seminarios/presentaciones/P_FPGAs_1_0.pdf

http://www.gelbukh.com/polibits/37_11.pdf

<http://www.elo.utfsm.cl/~elo212/docs/elo212-lab04-0109.pdf>

Libros

- ✓ [Wiley] Synthesis of Arithmetic Circuits - FPGA, ASIC and Embedded Systems (2006)DDU
- ✓ ASIC FPGA.Verification-Guide.to.Component.Modeling – 2005
- ✓ FPGA Express VHDL Reference Manual, *May 1999*

- ✓ Electronics - Digital - CPLD and FPGA - B. Zeilman - An Introduction to FPGA Design

Artículos

- ✓ Tale of tools, Review 2005, *FPGA and STRUCTURED ASIC JOURNAL*
- ✓ EE Times: Startup defines next-generation FPGA, *David Bursky 18/09/2006*
- ✓ How Programmable Logic Works, *Michel Barr, 1999*