Propuesta De Prácticas De Laboratorio Para Sistemas Embebidos Basados En Spartan-3E Starter Kit De Xilinx

Edwinn Andrey Silva Lamus

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍAS FISICOMECÁNICAS
ESCULA DE INGENIERÍA ELÉCTRICA, ELECTRÓNICA Y TELECOMUNICACIONES

Bucaramanga, 2009

Propuesta De Prácticas De Laboratorio Para Sistemas Embebidos Basados En Spartan-3E Starter Kit De Xilinx

Edwinn Andrey Silva Lamus

Trabajo de grado para optar al título de Ingeniero Electrónico

Director:
Mie (c) Carlos Augusto Fajardo Ariza

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍAS FISICOMECÁNICAS
ESCULA DE INGENIERÍA ELÉCTRICA, ELECTRÓNICA Y TELECOMUNICACIONES

Bucaramanga, 2009

RESUMEN

TÍTULO: PROPUESTA DE PRÁCTICAS DE LABORATORIO PARA SISTEMAS EMBEBIDOS BASADOS EN SPARTAN 3E STARTER KIT DE XILINX[1].

AUTOR: SILVA LAMUS, Edwinn Andrey.[2]

PALABRAS CLAVES: Sistemas Embebidos, FPGAs, PicoBlaze, S3ESK.

DESCRIPCIÓN:

Durante el desarrollo del presente proyecto se crearon y desarrollaron las prácticas para el laboratorio de Sistemas Embebidos basado en FPGA, verificando su concordancia con los objetivos planteados y las necesidades a las que se enfrenta quien se inicia en el diseño de soluciones para desarrollar en FPGAs. También, Se ha realizado la documentación necesaria para el uso de la plataforma Spartan-3E Starter Kit y el uso del microcontrolador PicoBlaze.

La propuesta de Prácticas de Laboratorio de Sistemas Embebidos basados en FPGAs más que una serie de guías a seguir al realizar un laboratorio, es una herramienta para quienes se inician en el área de Sistemas Embebidos, la cual les permitirá contar con una ayuda que les facilitará el diseño de Sistemas Embebidos sobre FPGAs. Estas prácticas planteadas no son una profundización en el estudio de diseño de Sistemas Embebidos, sino que ofrecen los primeros pasos por los que se debe ir al integrar hardware y software, así como hardware especifico y reconfigurable.

A través del proceso de formulación y desarrollo de las prácticas de laboratorio se resalta la validación del planteamiento de las FPGAs como los nuevos ejes centrales para el diseño y construcción de Sistemas Embebidos debido principalmente a su flexibilidad, reusabilidad y rápido tiempo de desarrollo.

---

[1]Trabajo de Grado

[2]Facultad de Ingenierías Físico-Mecánicas. Escuela de Ingeniería Eléctrica, Electrónica y Telecomunicaciones. Director: Mie (c) Carlos Augusto Fajardo Ariza

A**BSTRACT**

TITLE: PROPOSAL OF LABORATORY PRACTICES FOR EMBEDDED SYSTEMS BASED ON SPARTAN-3E STARTER KIT OF XILINX[3].

AUTOR: SILVA LAMUS, Edwinn Andrey.[4]

KEYWORKDS: Embedded Systems, FPGAs, PicoBlaze, S3ESK.

DESCRIPTION:

During the development of the present project, the practices for the FPGA based Laboratory of Embedded Systems were created and developed, verifying their concordance with the proposed objectives and the needs faced by any person who begins their solutions designs to develop FPGA's. Also, the necessary documentation has been made for the use of the Spartan-3E Starter Kit platform and the use of a PicoBlaze microcontroller.

The proposal of Practices for the FPGA based Laboratory of Embedded Systems, more than just a simple series of guides to follow while doing a lab practice, is a tool for those who start their journey in the area of Embedded Systems, which will let them count on an aid that will make their design of Embedded Systems on FPGA's easier. These proposed practices are not a profundization in the study of Embedded Systems Design, but offer the first steps that should be followed to integrate hardware and software, as well as specific and reconfigurable hardware.

Through the formulation process and the development of the laboratory practices, the validation of the proposal of FPGA's as the new central axis for the design and construction of Embedded Systems is highlighted, specially due to their flexibility, ability to be reused, and fast development times.

[3]Degree Work

[4]Faculty of Physical-Mechanical. Engineering Electrical, Electronic and Telecommunications School. Director: Mie (c) Carlos Augusto Fajardo Ariza

A mis padres por su constante apoyo,

A mis hermanas por su aliento,

A mis profesores y amigos por la motivación y apoyo desinteresado.

Agradecimientos

Especial agradecimiento a mis padres y familiares por su apoyo incondicional en este largo camino.

Agradezco a todas las personas que han contribuido en mi proceso de formación como ingeniero; a Carlos Augusto Fajardo por su acompañamiento, guía y colaboración al desarrollar el proyecto; y al profesor Jorge Hernando Ramón Suaréz por compartir su idea de realizar este proyecto, su motivación, orientación e inmensa paciencia durante el desarrollo del mismo.

Índice De Contenido

Índice De Figuras

Índice De Tablas

Capítulo 1

Introducción

FPGAs Y Los Sistemas Embebidos

A lo largo de las últimas décadas se han realizado múltiples desarrollos tecnológicos permitiendo el crecimiento de los Sistemas Embebidos, abarcado mayor cantidad de soluciones, reduciendo sus diferentes costos y masificando su uso. Ya sea por la reducción del área de los integrados o el crecimiento en las capacidades de procesamiento, actualmente abarcan las actividades que el hombre realiza o esta presente junto a él en la realización.

Los Sistemas Embebidos son actualmente los dispositivos con los que quizá más interactuamos hoy en día, hace ya algunos años el New York Times publicó que el americano promedio interactúa diariamente con más de 100 procesadores embebidos[1]; entra en contacto con ellos desde el preciso momento en que inicia su día, al ser despertado por su reloj programado, siguiendo con el uso de un horno microondas, cuando se dirige a un lugar los dispositivos de control con que cuenta la mayoría de sistemas de transporte, en su cotidianidad con el uso de computadores de mano, teléfonos móviles, sistemas de posicionamiento global, reproductores de audio, modems o routers para acceder a Internet y muchos otros dispositivos, como también sistemas más especializados de aplicaciones industriales y médicas. Muchos de estos sistemas pasan desapercibidos para el hombre, pero actualmente hace parte su vida sin llegar a notar

cuanto se la facilitan y sin saber cómo, cuándo o por qué puede ser de gran importancia, más aun desconociendo como se logran estos desarrollos.

Víctor Grimblatt, asegura que en la actualidad en una casa se pueden encontrar 40 microprocesadores en promedio, que no incluyen el computador personal ni los que se encuentran en un automóvil, además que esta cifra aumentará en una o dos ordenes de magnitud para la próxima década, estimando que para el año 2010 cada ser humano interactuará con 350 microprocesadores por día[25].

Un enfoque que en los últimos años está tomado fuerza para el diseño de Sistemas Embebidos, es el de las FPGAs, pues ofrecen gran cantidad de recursos al diseñador, desde millones de compuertas lógicas, memoria RAM y hasta microprocesadores, brindando la capacidad para ser una herramienta robusta y de alto desempeño. Esta tecnología tiene la posibilidad de implementar microprocesadores (Soft-Core) con los bloques lógicos programables de la FPGA, con la posibilidad de usar cualquier cantidad de estos según sea necesario y solo siendo limitado por la capacidad de la FPGA.

Adicionalmente las FPGAs están disponibles en diferentes empaquetados y de diversa complejidad, y el tiempo de desarrollo de un diseño es más corto y económico comparado con ASIC [4].

Pocos autores centran sus temáticas en el uso de FPGAs, uno de ellos Lewin A. R. W Edwards [6], realiza una corta introducción y resalta su uso para diseñar dispositivos para diversas aplicaciones evitando adquirir hardware específico para varias aplicaciones.

Algunas Universidades a nivel mundial, han optado por reemplazar los cursos de diseño de sistemas con microprocesadores, por cursos para diseño y construcción de Sistemas Embebidos basados en FPGAs. Como se planteó, y según Stephen A. Edwards[5] no existe literatura disponible para la enseñanza de Sistemas Embebidos que se encuentren

[5] Stephen A. Edwards, profesor de Department of Computer Science, Columbia University

orientada al uso de FPGAs, en tales cursos ha planteado que el estudiante desarrolle en el laboratorio diferentes prácticas y proyectos con estos dispositivos.

Algunas de las principales ventajas del diseño de Sistemas Embebidos basados en FPGAs:

- Software reconfigurable
- Alto Desempeño
- Ejecución simultánea de tareas en paralelo
- Alta fiabilidad, hardware “reflejado” en software.
- Bajo consumo de potencia
- Flexibilidad en el diseño
- Puertos Entrada/Salida reconfigurables

Capítulo 2

Conceptos Básicos

Sistemas Embebidos

Un sistema embebido es una combinación de hardware y software de computador, y periféricos mecánicos adicionales u otras partes, diseñados para realizar una función dedicada. En algunos casos, los Sistemas Embebidos son parte de un gran sistema o producto, como en el caso del sistema antibloqueo de frenos en un automóvil[8].
Se diferencia ampliamente con el computador personal, ya que está diseñado para cumplir una función específica, mientras el PC está disponible para realizar una variedad de tareas diferentes.

La existencia de procesador y software ocasionalmente no es advertida por quien usa el un Sistema Embebido. Para algunos casos es posible construir un sistema sin procesador ni software, con un circuito integrado que realice las funciones, pero perdería gran flexibilidad cuando se haga necesario efectuar cambios, siendo más fácil y económico cambiar líneas de código en el software que modificar el hardware[9].

Sistemas de Tiempo Real

Un Sistema en tiempo real se especifica en términos de su capacidad para hacer los cálculos o tomar decisiones en el momento oportuno. Los cálculos importantes poseen un plazo de realización, respuestas correctas pero demoradas son interpretadas como una

respuesta incorrecta. Un plazo puede ser “hard” o “soft”; para el caso donde las consecuencias más probables de una respuesta tardía pueda resulta en catástrofe se dice que es una restricción de tiempo “hard”, para consecuencias del otro extremo al caso hard, se dicen de restricción de tiempo “soft”.

Se debe garantizar el funcionamiento del hardware y software bajo todas las condiciones posibles a las que se somete el funcionamiento del equipo. Y, en la medida en que la vida humana dependa de la correcta ejecución del sistema, esta garantía debe estar respaldada por cálculos de ingeniería y descripción de su funcionamiento.

Componentes Básicos de un Sistema Embebido

Un sistema embebido esta compuesto básicamente por un procesador y un software. Para poder poseer un software necesita tener una ROM, para ejecutar el código, y una RAM para almacenar datos temporalmente durante la ejecución. Los Sistemas Embebidos tienen al menos una de cada una, y en caso de ser pequeña la cantidad necesaria para el procesamiento esta podrá estar contenida en el mismo chip del procesador, de lo contrario se encontraran en chips de memoria externos.

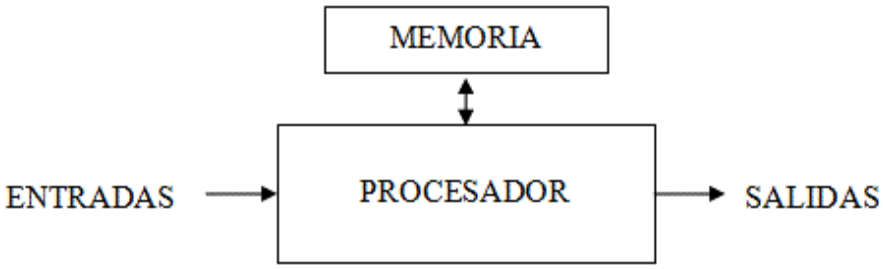


Figura 1 Esquema General de un Sistema Embebido

Todos los Sistemas Embebidos necesariamente tienen algunos tipos entradas y salidas. Las entradas del sistema usualmente provienen de sensores, señales de comunicación, perillas, botones de control, etc. Las salidas son típicamente señales a actuadores, displays o señales de comunicación.

El resto del hardware necesario para el sistema, es incorporado de acuerdo a las necesidades y criterios de diseño. Cada sistema debe cumplir con especificaciones completamente diferentes, cualquiera de ellas o todas pueden afectar las consideraciones o compromisos realizados durante el diseño. Una de las limitaciones con las que se trabaja es el costo de producción del sistema.

Una representación más completa de un sistema embebido, reuniendo las características antes mencionadas se observa en la Figura 2 Esquema de un Sistema Embebido

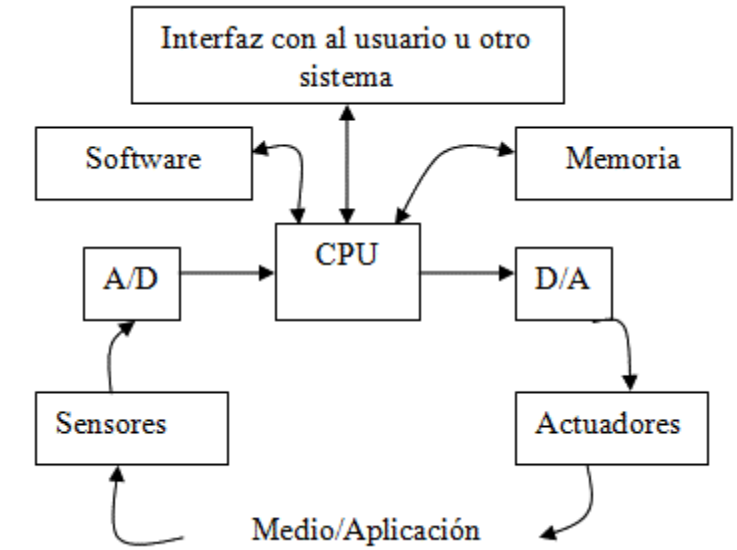


Figura 2 Esquema de un Sistema Embebido

Otros requerimientos de diseño

Poder de procesamiento.

El poder de procesamiento se define como la cantidad de procesamiento necesaria para realizar una tarea. Una manera de comparar el poder de procesamiento es en MIPS[6]. Otro aspecto a considerar en un procesador es el ancho de los registros, que pude ir desde 8 hasta 64 bit y un Sistemas Embebidos se puede construir de 16 o 32 bit.

[6] MIPS, Millones de Instrucciones por Segundo.

Memoria

La memoria en un Sistema Embebido es la capacidad de almacenamiento necesaria para ejecutar el software y almacenar los datos que se manipulan. Al diseñar se debe estimar la cantidad de memoria necesaria, y de ser necesario aumentar o disminuir la cantidad requerida.

Consumo de Potencia

El consumo de potencia es una medida de la potencia usada durante la operación. Es muy importante, especialmente en los casos de dispositivos portátiles que se alimentan por baterías. Una medida usada para dispositivos portátiles es en mW/MIPS.

Costo de Desarrollo

El costo de desarrollo esta definido como el valor del proceso de diseño del hardware y el software. Es fijo, en función de los costos en el tiempo, en grandes volúmenes de un producto el dinero no es importante, mientras que en otros proyectos, éste es el único sistema de medición exacta del costo, para productos de un pequeño número de unidades.

Time to market

El tiempo que demora en pasar un producto del momento en que es concebido como idea, a cuando está listo para estar a la venta. Este es un factor importante, pues esta relacionado al tiempo que tarda su desarrollo, aspecto muy importante a la hora de comercializar una idea tecnológica. En el caso de las FPGAs, el Time to Market juega a favor, al ser reducido drásticamente de meses a unos cuantos días, aunque al explorar una nueva área, este tiempo algunas veces es mayor en otro tipo de tecnologías.

Tiempo de Vida

Este tiempo es el que se espera el producto permanezca en funcionamiento, es decir, la expectativa de duración. Este requerimiento afecta todo tipo de decisiones de diseño, desde la selección de componentes de hardware a cuánto cuesta el desarrollo y cuánto puede producir.

Numero de Unidades

La compensación entre el costo de producción y el costo de desarrollo es más afectada por el número de unidades previsto que se producen y las que se venden.

Fiabilidad

En este requerimiento se trata de definir cuan confiable debe ser el producto final. En el caso de un juguete no siempre tiene que trabajar correctamente, pero en el caso de una parte de transbordador o un automóvil, debe hacer lo que se supone que haga para todo y cada momento.

Sistemas Operativos Embebidos

El movimiento *Open Source* ha generado una gama de diversos sistemas operativos embebidos de bajo costo o en algunos casos gratis, los más usados son los creados entorno al kernel de Linux. Este ha sido llevado a una gran cantidad de arquitecturas y existe una amplia documentación del proceso de instalación en diferentes plataformas; su amplio uso en Sistemas Embebidos ha producido características específicas para Linux Embebido.

Actualmente, se disponen de diferentes sistema operativos embebidos, del tipo Linux como: BlueCat, RTLinux o ucLinux y el sistema NetBSD derivado Unix. De tipo open source como eCos o del tipo propietario como Palm OS. Su uso o selección depende tanto del hardware disponible como de las funciones y característica de cada sistema operativo que sean útiles para solucionar los requerimientos planteados del sistema embebido.

En función del hardware disponible los sistemas operativos disponibles permiten ser ejecutados en gran variedad microcontroladores; en el caso de las distribuciones Linux cada una de ellas ofrece soporte para diferentes microcontroladores; también, es posible encontrar algunas distribuciones para FPGAs[7] ejecutadas en microcontroladores hard-

[7] FPGAs – Filed Programmable Gate Array

cores o soft-cores. En cuanto a las necesidades de hardware adicional, las distribuciones Linux necesitan de una unidad de gestión de memoria (MMU), tanto en los microcontroladores como en FPGAs, en este aspecto ucLinux es una distribución que no requiere de MMU. Además, pueden ser de tipo sistema operativo de tiempo real (RTO), algunos pueden ser considerados RTOs para pequeños tiempos de ejecución, sin embargo, en algunos casos se necesitan servicios que sean atendidos y ejecutados en una mínima y estable cantidad de tiempo, siendo estos verdaderos RTOs.

Linux es un sistema de amplio crecimiento, con una aplicación en una amplia gama de industrias, su alta adaptabilidad a las diferentes necesidades, y en gran medida la disponibilidad de manipular el código fuente para permitir a los desarrolladores lograr una mayor adaptación de las aplicaciones[1].

En los Sistemas Embebidos más sencillos no es necesario el uso de sistemas operativos, pueden llegar a ser escritos completamente partiendo desde cero, siendo así código de tipo propietario. En algunos casos usando librerías del lenguaje o de terceros, inclusive partiendo de código fuente de alguna aplicación que se asemeje y/o posea contenido necesario para el proyecto.

Field Programmable Gate Arrays, FPGA

Xilinx[8] en 1985 presenta las FPGAs como una nueva alternativa a los dispositivos lógicos programables (CPLDs) y ASICs. Ofrecen la ventaja de ser fácilmente programables y la posibilidad de ser reprogramadas una y otra vez, permitiendo a las diseñadores realizar múltiples variaciones y ajustes a su diseños[4].

“Una FPGA es una estructura regular de celdas lógicas e interconexiones. Que se encuentra bajo su completo control. Esto significa que usted puede diseñar, programar, y realizar cambios en su circuito cuando lo desee”[5].

[8] Xilinx – Mayor empresa de investigación, desarrollo y ventas de FPGAs

Existen dos tipos básicos de FPGAs: de una sola programación o reprogramable basadas en SRAM. Estos dos tipos de FPGAs difieren en la implementación de las celdas lógicas y la forma como se hacen las conexiones en el dispositivo. El mercado es ampliamente dominado por el tipo SRAM al poder ser reprogramada cuantas veces se desee. Una FPGA es reprogramada cada vez que se conecte a la alimentación, razón por la que necesitamos una memoria PROM o un sistema de memoria SRAM con cada FPGA. Sin embargo, en la actualidad Xilinx con su familia Spartan-3AN ofrece FPGAs no volátiles, a un costo mayor y una reducción de los recursos disponibles en comparación con la familia sin esta característica adicional.

Su uso se puede dividir en tres niveles. En el extremo inferior, de tipo de lógica programable para manejar simples tareas de codificación y decodificación. En el extremo superior son grandes FPGAs capaces de ejecutar toda clase sistemas o subsistemas, incluyendo procesadores Soft-Core. Algunos chips incorporan núcleos dedicados de procesadores (Hard Core). En el nivel intermedio las FPGAs son ideales para manejar datos, la sincronización de los subsistemas, y en general, para hacer más fácil la personalización del sistema en desarrollo.

Inicialmente la capacidad de las FPGAs era de miles de celdas lógicas y operaban a 40Mhz, llegando a costar más de U$ 150. Actualmente, pueden llegar a millones de celdas lógicas y operar en 450Mhz, con un costo de menos de U$ 10, con agregados como multiplicadores, memoria y procesadores.

En sus comienzos las FPGAs se tenían como un medio para obtener prototipos rápidos y sistemas de emulación de hardware, su poca capacidad y falta de herramientas, hacían que fuera altamente costoso su uso. Con dispositivos más grandes y la disminución de costos, las FPGAs abandonan el camino de los prototipos para ser parte de los dispositivos de producción [4].

Actualmente algunas FPGAs fabricadas por Xilinx y Altera[9] permiten desarrollar y mantener distribuciones Linux, por medio de sus procesadores core. Por parte de Xilinx en FPGAs Virtex-4 con PowerPC 405, y Altera con Nios para Stratix o Cyclone. Xilinx actualmente permite la opción de una MMU como parte del Soft-Core MicroBlaze en la FPGA, permitiendo el desarrollo de un Linux completo.

[9] Altera – Segunda mayor empresa fabricante y desarrolladora de FPGAs

Capítulo 3

Desarrollo Del Proyecto

En el fase inicial del desarrollo del proyecto se analizaron las principales necesidades a las que se enfrenta un diseñador al iniciarse en el área de desarrollo de Sistemas Embebidos; de igual manera se exploraron las capacidades que brinda la plataforma Spartan-3E Starter Kit[10], con el fin de determinar el contenido de la propuesta de prácticas de laboratorio a realizar con la plataforma.

De acuerdo al estudio preliminar, citado anteriormente, se plantearon los siguientes tópicos a desarrollar:

- Programación y/o configuración de los dispositivos con los que cuenta la plataforma.
- Diseño de hardware para el manejo de los puertos de la plataforma.
- Manejo de Microcontroladores o Microprocesadores soft-core, como eje central de un Sistema Embebido basado en FPGAs.
- Integración entre descripción de hardware especifico para manejo de puertos y el microcontrolador embebido.

[10] Plataforma disponible en la Escuela de Ingeniería Eléctrica, Electrónica y Telecomunicaciones,

Sistema de desarrollo

Para el desarrollo de este proyecto se partió del sistema de desarrollo Spartan-3E Starter Kit, el cual cuenta con una la plataforma de desarrollo Spartan 3E Starter Board y los Software Xilinx ISE™ y EDK.

La plataforma del Spartan-3E Starter Kit principalmente esta compuesta por:

- FPGA Xilinx Spartan-3E
- CPLD Xilinx CoolRunner-II
- Cuatro switch, cuatro botones y una perilla
- Pantalla LCD, Puerto VGA
- Puertos Seriales RS-232
- Conversores DA y AD
- Puerto PS/2
- Interfaz Física Ethernet
- Xilinx Flash PROM
- SPI serial Flash
- Intel StrataFlash Parallel NOR Flash PROM
- EEPROM 1-Wire SHA-1
- Conectores de expansión de 6 pines y 100 pines (Hirose FX2 Edge)

En este proyecto se realizó una descripción los dispositivos con los que cuenta esta plataforma (Para mayor información ver Anexo F: *Introducción a la S3ESK*).

Selección de las prácticas

Los sistemas de desarrollo poseen diferentes dispositivos que son configurados de la mejor forma posible para permitir la ejecución de gran variedad de aplicaciones. La Spartan 3E Starter Kit Board (S3ESK), posee una arquitectura bastante flexible debido a que cuenta con hardware reconfigurable como lo es una FPGA y un CPLD, además, diferentes opciones de memoria y formas de compartir y obtener información.

El estudio de la plataforma, junto con sus posibilidades de diseño, permitió establecer algunos de los conocimientos necesarios para usar esta plataforma en el diseño de Sistemas Embebidos. Se debe tener en cuenta que no se pretende cubrir “todos” los conocimientos sino más bien se han seleccionado algunos que son comunes en múltiples aplicaciones.

Partiendo del conocimiento y manejo de VHDL, se necesita aprender a diseñar Sistemas Embebidos, definir sus componentes, ya sea por medio de reingeniería de otros Sistemas Embebidos o abstrayendo de otras arquitecturas de Sistemas Embebidos. En general, se inicia el diseño de los componentes para un sistema dado y se propone una arquitectura de un sistema embebido basado en FPGA (componentes hardware, específico y/o reconfigurable).

El diseñador al plantear un Sistema Embebido necesita proponer una arquitectura que solucione un problema abordado, que en su forma básica está compuesta por un microcontrolador, memoria y puertos entrada/salida. Establecer el hardware para la arquitectura planteada, realizar su descripción VHDL o su integración física.

El Sistema Embebido necesita interactuar con el medio en que se desenvuelve, mediciones, comunicación y entrega de resultados. Implica establecer comunicación con actuadores y sensores u otros dispositivos que le suministren la información o datos a procesar, también interactuar con usuarios recibiendo y presentando resultado y/o información (Usando leds, pantallas, monitores, interruptores, pulsadores o diferentes tipo de teclado).
Microprocesador o microcontrolador, es la parte fundamental del Sistema Embebido siendo el eje central para el procesamientote datos, toma de decisiones. En este caso se selecciono el microcontrolador PicoBlaze, que brinda la posibilidad de crear, implementar y/o adaptar el software que integre el hardware disponible.

Para las necesidades planteadas y bajo la premisa del aprendizaje de “Aprender Haciendo”, se plantean las siguientes prácticas de laboratorio.

Prácticas a implementar

Configuración de la plataforma S3ESK

La puesta en funcionamiento de la plataforma de desarrollo S3ESK sucede después de ser energizada o de realizar un reset, cargando en la FPGA su configuración. La forma como es configurada la FPGA obedece al Modo de Configuración establecido en la S3ESK, que permite diferentes fuentes de almacenamiento para la configuración, en consecuencia se propone una práctica de laboratorio en donde se realicen las diferentes formas de configuración de la FPGA.

Generación de Señal VGA

VGA[11] es un sistema de video desarrollado por IBM, y convertido en un estándar de video ampliamente difundido y soportado por el hardware de video en los computadores personales. En diversos Sistemas Embebidos actuales se necesita de una interfaz visual al usuario con el fin de proporcionar una interacción con el entorno de una forma sencilla y de bajo costo. La generación de una señal de VGA para manejar un monitor es una solución y al realizarlo en bloque VHDL en una FPGA optimizamos el hardware usado, puesto que tendremos hardware dedicado de muy bajo costo.

Se plantea una práctica de laboratorio donde se analice la interfaz VGA. La señal VGA esta compuesta señales precisas de control según la resolución que se desea manejar. La arquitectura esta basada en la descripción de la señal VGA del documento [14], y su descripción es realizada en VHDL.

[11] Video Graphics Array

Microcontrolador PicoBlaze

Un enfoque de diseño que actualmente esta tomando mucha fuerza es la implementación de microcontroladores embebidos Soft-Core sobre FPGAs, lo cual está ofreciendo al diseñador la posibilidad de tener en una misma aplicación soluciones en software y en hardware. El microcontrolador usado es el PicoBlaze, proporcionado por Xilinx.

PicoBlaze, es una de las herramientas de enseñanza usadas en el laboratorio de Sistemas Embebidos basado en FPGAs, como se plantea en prácticas posteriores, para ser usado como parte funcional de los sistemas a implementar.

En un diseño con el microprocesador PicoBlaze es necesario poseer un código fuente, que al ser ensamblado se generen los archivos necesarios de configuración para agregar a un diseño VHDL, en el desarrollo de la práctica se plantea el uso de dos herramientas de software para este propósito; donde el código fuente es escrito usando un editor de texto, y se ensambla con el ejecutable de DOS KCPSM3, o con el software pBlazeIDE, que además de usarse para escribir el código fuente, permite simularlo y generar los archivos de configuración.

Alguna de las ventajas que posee el software pBlazIDE en su entorno de desarrollo integrado dedicado al KCPSM es la emulación de puertos de Entrada y Salida. Algunas instrucciones del pBlazIDE difieren en su nombre o sintaxis del KCPSM3, sin embargo, su funcionamiento es igual. pBlazIDE soporta varias variaciones del KCPSM, por tal motivo debe seleccionarse la versión PicoBlaze 3 para el uso en la FPGA Spartan-3E.

LCD y Teclado

En diversos Sistemas Embebidos es necesario capturar o informar valores, menús o instrucciones de uso. La S3ESK cuenta con un display de 16 caracteres y dos líneas y un puerto PS/2, planteando su uso conjunto como parte funcional de una aplicación que permita interactuar con el usuario captando y suministrando información, como ocurre en Sistemas Embebidos como celulares, reproductores de música, etc. Para éste propósito se

realiza una práctica de laboratorio donde se desarrollaren los procedimientos necesarios para lograr su correcto funcionamiento junto con el microcontrolador PicoBlaze.

VGA y Teclado

En un Sistema Embebido el manejo y manipulación de los datos e información es realizada por el microcontrolador o microprocesador. Una práctica donde se aproveche el microcontrolador PicoBlaze para procesar los datos de entrada, y dependiendo de estos modificar el contenido de la video memoria para poder observar en un monitor VGA el resultado obtenido, y en lo posible accesar alguna de las memorias de la S3ESK, dada la gran capacidad de almacenamiento que ofrecen éstas.

Capítulo 4

Resumen De Prácticas

Tabla 1 Resumen de Prácticas

Práctica de Laboratorio	Objetivos	Saberes
Configuración S3ESK	Configurar la plataforma S3ESK y sus diferentes modos de configuración de la FPGA Spartan-3E y accesar sus diferentes dispositivos (periféricos, puertos de entrada/captura y salida de datos). Objetivos Específicos	Comprensión
	Diseñar en VHDL un proyecto para verificar el funcionamiento de la plataforma S3ESK	Comprensión
	Configurar la FPGA Spartan-3E a través del USB-JTAG.	Aplicación
	Generar archivos de programación para las memorias disponibles en el S3ESK con el contenido de configuración de la Spartan-3E.	Conocimiento
	Configurar la FPGA Spartan-3E haciendo uso del modo de configuración Master Serial.	Aplicación

	Configurar la FPGA Spartan-3E haciendo uso del modo de configuración SPI.	` Aplicación
	Configurar la FPGA Spartan-3E haciendo uso del modo de configuración BPI.	Aplicación
Laboratorio VGA	Describir la composición de la señal de video VGA y la forma de generarla. Objetivos Específicos	Comprensión
	Crear un proyecto VHDL para generar la señal VGA	Aplicación
	Proponer mejoras en la descripción de hardware de una arquitectura definida.	Síntesis
	Describir la arquitectura de un sistema mediante el estudio de su proyecto VHDL	Análisis
Laboratorio Microcontrolador PicoBlaze	Crear un proyecto de diseño VHDL usando el microcontrolador PicoBlaze para la plataforma S3ESK. Objetivos Específicos	Aplicación
	Sintetizar, implementar y generar el archivo de configuración de la FPGA al usarse con un microcontrolador PicoBlaze.	Análisis, Aplicación
	Identificar las ventajas y desventajas del uso de los microcontroladores Soft-Core en sistemas basados en FPGAs.	Análisis, Percepción
	Identificar las ventajas y desventajas de las diferentes herramientas asociadas al microcontrolador PicoBlaze.	Análisis, Percepción

LCD y Teclado PS/2	Realizar e implementar un diseño en la S3ESK que permita de visualizar y capturar información para el funcionamiento de un proyecto VHDL. Objetivos Específicos	Aplicación
	Programar el microcontrolador PicoBlaze para que inicialice la pantalla LCD e implementar las funciones de la pantalla como subrutinas de programa que puedan ser usadas en otros proyectos VHDL	Aplicación, Adaptación
	Crear un modulo VHDL que permita capturar una señal de comunicación serial PS/2, y suministre la información contendida en la transmisión.	Aplicación Adaptación
	Realizar la interconexión entre diferentes módulos VHDL en un mismo proyecto VHDL.	Aplicación, Conocimiento
VGA y Teclado PS/2	Crear un proyecto que pueda recibir información por medio del teclado y visualizarla usando un monitor VGA. Objetivos Específicos.	Aplicación, Adaptación
	Implementar el uso de la interrupción por hardware con que cuenta el microcontrolador PicoBlaze para realizar las operaciones necesarias para las condiciones que se establezcan en el diseño.	Aplicación, Conocimiento
	Accesar y modificar el contenido de una memoria, ya sea de la FPGA o de la S3ESK.	Aplicación, Síntesis

Capítulo 5

Conclusiones

Durante el desarrollo del presente proyecto se crearon y probaron las prácticas para el laboratorio de Sistemas Embebidos basados en FPGAs, verificando su concordancia con los objetivos planteados y las necesidades a las que se enfrenta quien se inicia en el diseño de soluciones embebidas desarrolladas sobre FPGAs. También, se ha realizado la documentación necesaria para el uso de la plataforma S3ESK y el uso del microcontrolador PicoBlaze.

El microcontrolador PicoBlaze, es de fácil uso y permite la optimización para aplicaciones que requieren un uso reducido de recursos, y cuenta por si solo con amplia y comprensible documentación. Otras ventajas del PicoBlaze es que es altamente poderoso en comparación con los recursos que usa en la FPGA (85 slices y un bloque RAM) pues alcanza a ofrecer de 40-70 MPIS.

La propuesta de Laboratorios, es un recurso que facilita la iniciación en el desarrollo de Sistemas Embebidos, estas prácticas no pretender ser una profundización en el estudio de diseño basado en FPGAs, pero si plantean los primeros pasos a seguir al integrar hardware y software, así como hardware especifico y reconfigurable.

Quien se encuentre realizando el desarrollo de los sistemas necesita definir la partición entre el diseño de hardware y software para lograr obtener un alto rendimiento en la ejecución. Para ello es necesario adentrarse mucho más en el estudio de los Sistemas Embebidos en general, razón por la que se sugiere que la Universidad Industrial de

Santander por medio de la Escuela de Ingeniería Eléctrica, Electrónica y Telecomunicaciones dedique espacios para la enseñanza, la práctica y los aspectos que del conocimiento que involucran los Sistemas Embebidos.

La unión entre los recursos lógicos de una FPGA y un microprocesador embebido (como el PicoBlaze) ofrece grandes ventajas a la hora de hacer diseños embebidos, pues se cuenta con una mayor facilidad a la hora de hacer el particionamiento software / hardware.

Es claro que cada vez más los Sistemas Embebidos basados en FPGAs brindan aplicaciones en el área de control, automatización, comunicaciones, procesamiento de señales y/o imágenes, es decir se están convirtiendo en el puente entre las necesidades y las soluciones, razón por la cual su investigación al interior de la Universidad, por lo tanto es muy importante desarrollar la capacidad para crear y diseñar nuevas aplicaciones de herramientas, equipos e instrumentos para el desarrollo de la comunidad en Sistemas Embebidos. Las prácticas propuestas se fundamentan en brindar al estudiante una herramienta para ayudar a desarrollar sus capacidades para crear y diseñar Sistemas Embebidos.

Finalmente a través del proceso de realización del proyecto se resalta la validación del planteamiento de las FPGAs como los nuevos ejes centrales para el diseño y construcción de Sistemas Embebidos.

Referencias Bibliográficas

Documentos

[1] Carlos A. Fajardo, Jorge H. Ramón. Sistemas Embebidos: Su Importancia En El Mundo Actual Y Las Tendencias Tecnológicas Para Los Próximos Años. 2007

[2] Chris Conger, David Bueno y Alan D. George, Experimental Analysis of Multi-FPGA Architecture over RapidIO for Space-Based Radar Processing

[3] D. W. Lewis, Fundamentals of Embedded Software. Prentice Hall. 2002

[4] David Maliniak, Basic of Design. FPGAs. Tradeoff Abound in FPGA Design. Diciembre 4, 2003

[5] Karen Parnell, Nick Mehta. Introduction to Programmable Logic. Xilinx. Abril 2004

[6] Lewin A. R. W. Edwards, Embedded System Design on a Shoestring. Newnes 2003

[7] Matt Volckmann, Steve Balacco, Chris Rommel, Linux in the Embedded Systems Market, Sept. 20, 2007

[8] Michael Barr. Embedded Systems Glossary. Netrino Technical Library. Abril 21, 2004

[9] Michael Barr. Programing Embedded Systems in C and C++, O'Reilly. Segunda Edición 2006

[10] Stephen A. Edwards, Experiences Teaching an FPGA-based Embedded Systems Class

[11] Xilinx DS312-1 Spartan-3E FPGA Family Data Sheet, v3.4, Noviembre 9, 2006

[12] Xilinx DS312-2 Spartan-3E FPGA Family Data Sheet, v3.6, Mayo 29, 2007

[13] Xilinx UG129 PicoBlaze 8-bit Embedded Microcontroller User Guide for Spartan-3, Virtex-II, and Virtex-II Pro FPGAs, v1.1, Junio 10, 2004

[14] Xilinx UG230 Spartan-3E Starter Kit Board User Guide, v1.0, Marzo 9, 2006

[15] Xilinx UG332 Spartan-3 Generation Configure User Guide, v1.4, Julio 1, 2008

[16] Micron Technology, 512Mb DDR SDRAM (x4,x8,x16) Component Data Sheet, Rev A, 2007

[17] Intel, 3 Volt Intel® StrataFlash™ Memory 28F128J3A, 28F640J3A, 28F320J3A, Version-008, Abril 13, 2001

URLs

[18] Centro de Investigación en Computación, del Instituto Politécnico Nacional. http://posgrado.cic.ipn.mx/ Revisado en Febrero 2008

[19] Columbia University http://www.columbia.edu. Revisado en Febrero 2008

[20] Escuela Politécnica Superior http://www.eps.uam.es. Revisado en Febrero 2008

[21] Escuela Politécnica Superior – UAM, Sistemas Embebidos. Implementación en un FPGA del microcontrolador PicoBlaze.

http://arantxa.ii.uam.es/~edcd/lab/Practica3_PB_2008_09.pdf Revisado en Febrero 2008

[22] Sistemas Embebidos Basados en FPGAs y Linux. http://posgrado.cic.ipn.mx/?q=mcic.mse.tesis.mars3. Revisado en Dciciembre de 2008.

[23] Spartan-3E Starter Kit http://www.xilinx.com/s3estarter. Revisado en Noviembre de 2007

[24] University of Florida http://www.ufl.edu. Revisado en Diciembre de 2008

[25] Victor Grimblatt H., Introducción a los Sistemas Embebidos, Material Docente https://www.u-cursos.cl/ingenieria/2007/1/EL65P/1/material_docente Revisado en Febrero 2008

[26] Xilinx. https://www.xilinx.com. Revisado en Febrero 2008

Anexos

ANEXO A

PRÁCTICA No 1

CONFIGURACIÓN DE LA PLATAFORMA SPARTAN-3E STARTER KIT

INTRODUCCIÓN

El objetivo primordial de esta práctica es conocer la plataforma de desarrollo SPARTAN-3E STARTER KIT (S3ESK) haciendo uso de algunos de sus dispositivos (Memorias, puertos de entrada y salida de datos).

En esta práctica se configurará la FPGA disponible en la plataforma S3ESK utilizando diferentes modos de configuración. Para ello se implementa un diseño de hardware para manipular algunas de las entradas y salidas de la S3ESK. El diseño es realizado haciendo uso de lenguaje de descripción de hardware con la herramienta ISE de Xilinx, que permite la sintetización, implementación y generación de los archivos de configuración de la FPGA. Esta configuración inicialmente es realizada mediante la descarga directa del archivo de configuración y luego desde las memorias disponibles en la S3ESK, que son programadas con el contenido adecuado.

OBJETIVOS

OBJETIVO GENERAL

Configurar la plataforma S3ESK y sus diferentes modos de configuración de la FPGA Spartan-3E y accesar sus diferentes dispositivos (periféricos, puertos de entrada/captura y salida de datos).

OBJETIVOS ESPECIFICOS

- Diseñar en VHDL un proyecto para verificar el funcionamiento de la plataforma S3ESK
- Configurar la FPGA Spartan-3E a través del USB-JTAG.
- Generar archivos de programación para las memorias disponibles en el S3ESK con el contenido de configuración de la Spartan 3E.

- Configurar la FPGA Spartan-3E haciendo uso del modo de configuración Master Serial.
- Configurar la FPGA Spartan-3E haciendo uso del modo de configuración SPI.
- Configurar la FPGA Spartan-3E haciendo uso del modo de configuración BPI.

TRABAJO PRELIMINAR

Leer y estudiar los documentos *Introducción a la S3ESK* y *Manual de Configuración de la S3ESK*

PROCEDIMIENTO

Realice cada uno los numerales, luego describa y analice el proceso realizado.

1. Cree un proyecto VHDL que permita verificar el funcionamiento de los leds y pulsadores disponibles en la plataforma S3ESK.

2. Generar el archivo de configuración para la FPGA de la S3ESK con del proyecto creado.

3. Implemente la descripción VHDL realizada para la FPGA haciendo uso del modo de configuración USB-JTAG. Siga las instrucciones que se indicaron en el documento *Manual de Configuración de la S3ESK - Configuración USB-JTAG*. Verifique el funcionamiento del diseño realizado.

4. A partir del archivo de configuración creado, realice la configuración de la FPGA en modo:
   4.1. Master Serial
   4.2. SPI
   4.3. BPI UP y DOWN

   El procedimiento para realizar la programación en los modos indicados se describe en el documento *Manual de Configuración de la S3ESK*. Verifique el funcionamiento de cada modo de configuración.

** Compare los diferentes modos de configuración de la FPGA, ventajas y desventajas que se observaron.

ANEXO B

PRÁCTICA NO 2

GENERACIÓN DE SEÑAL VGA

INTRODUCCIÓN

VGA[1] es un sistema de video desarrollado por IBM, y convertido en un estándar de video ampliamente difundido y soportado por el hardware de video en los computadores personales. En diversos Sistemas Embebidos actuales se necesita de una interfaz visual al usuario con el fin de proporcionar una interacción con el entorno de una forma sencilla y de bajo costo. La generación de una señal de VGA para manejar un monitor es una solución y al realizarlo en bloque VHDL en una FPGA optimizamos el hardware usado, puesto que tendremos hardware dedicado de muy bajo costo.

En este laboratorio analizaremos un diseño para una interfaz VGA con una resolución de 640 por 480 y ocho colores. La señal VGA esta compuesta señales precisas de control según la resolución que se desea manejar. La arquitectura esta basada en la descripción de la señal VGA del documento *Guía de usuario de la plataforma Spartan-3E FPGA Starter Kit*, y su descripción es realizada en VHDL.

OBJETIVOS

OBJETIVO GENERAL

Describir la composición de la señal de video VGA y la forma de generarla.

OBJETIVOS ESPECIFICOS

- Crear un proyecto VHDL para generar la señal VGA.
- Proponer mejoras en la descripción de hardware de una arquitectura definida.
- Describir la arquitectura de un sistema mediante el estudio de su proyecto VHDL.

[1] Video Graphics Array

TRABAJO PRELIMINAR

Estudiar y analizar los documentos que contienen la información necesaria para comprender la señal de video VGA.

Archivo:
Guía de usuario de la plataforma Spartan-3E FPGA Starter Kit
http://www.digilentinc.com/Data/Products/S3EBOARD/S3EStarter_ug230.pdf

Libro:
FPGA Prototyping by VHDKL Examples Xilinx Spartan 3

ARQUITECTURA VGA

El generador de señal VGA, se encarga de sincronizar los datos de salida R,G,B con las señales de sincronismo Sinc_V y Sinc_H generadas. Esta compuesto por los bloques: Generador Sincronismo Vertical, Generador Sincronismo Horizontal, Generador de Direcciones, Generador Blanq y Asignar Color.

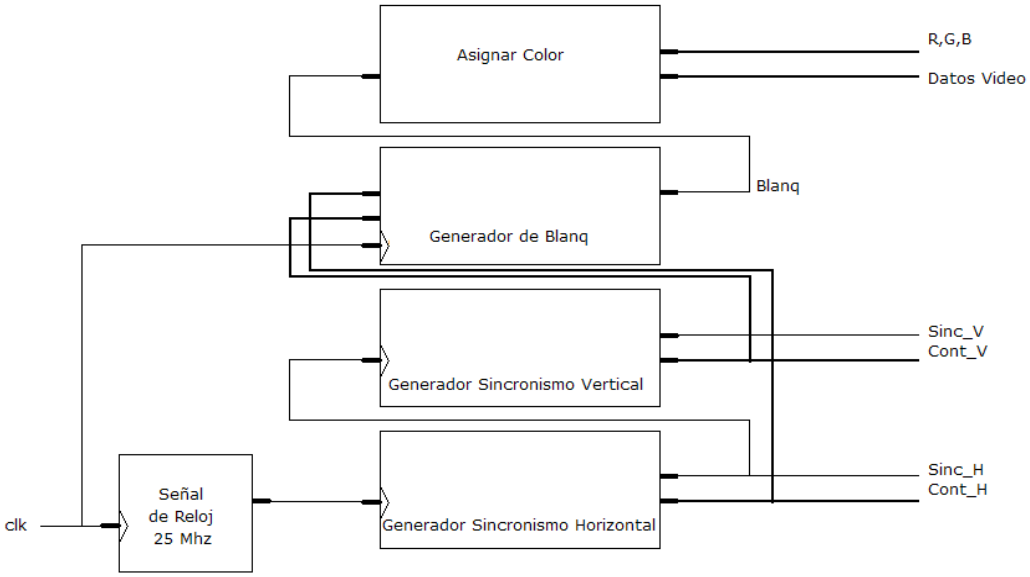


Figura 1 Generador de Señal de Video VGA

El bloque Señal de Reloj 25 MHz, se encarga de dividir la frecuencia del reloj del sistema, para permitir que los Generadores de Sincronismos obtengan tiempos acordes a los establecidos en la descripción de los tiempos VGA en el documento *Spartan-3E Starter KitBoard User Guide*.

Generadores de Sincronismo

Los Generadores de Sincronismo están encargados de aumentar y resetear los registros contadores o posicionadores, y generar el estado de las señales de sincronismo.
Acorde con la cantidad de píxeles o líneas y tiempos establecidos en el estándar VGA, el contenido de los registros determina el estado alto o bajo de las señales de sincronismo.

Bloque Blanq

El bloque Blanq genera una señal blanq que es usada para indicar la parte de las señales de sincronismo que es visible en el monitor.

Asignar Color

Los datos de color generados R, G y B son asignados y se validan durante los tiempos acordes a las señales de sincronismo, usando la señal *blanq* para realizar esta validación.

PROCEDIMIENTO

1. Analice el archivo VHDL entregado, verifique la concordancia de la descripción realizada con la arquitectura propuesta

2. Verifique mediante medición las señales generadas por el proyecto y el cumplimiento con los tiempos establecidos para estas.

3. Crear un proyecto VHDL, basándose en la arquitectura VGA que permita visualizar alguna de las siguientes opciones:
    a. Los patrones de video en un monitor. (También, conocida como carta de ajuste)
    b. Un cuadro rebotando dentro del área de la señal de video generada.

ANEXO C

PRÁCTICA No 3

INTRODUCCIÓN AL MICROCONTROLADOR PicoBlaze

INTRODUCCIÓN

Esta práctica se enfocada en el uso del microcontrolador PicoBlaze, que es una de las herramientas de enseñanza usadas en el laboratorio de Sistemas Embebidos basado en FPGAs, como se plantea en prácticas posteriores, puede ser usado como parte funcional de los sistemas a implementar.

En un diseño con microprocesador PicoBlaze es necesario poseer un código fuente, que al ser ensamblado se generen los archivos necesarios de configuración para agregar a un diseño VHDL. En el desarrollo de la práctica se plantean dos opciones para este propósito. Inicialmente el código fuente es escrito usando un editor de texto, y se ensambla con el ejecutable de DOS KCPSM3. Por otra parte, Mediatronix[1] ha creado una herramienta alternativa para esta tarea, el software pBlazeIDE, que además de usarse para escribir el código fuente, permite simularlo y generar los archivos de configuración.

Alguna de las ventajas que posee el software pBlazIDE en su entorno de desarrollo integrado dedicado al KCPSM es la emulación de puertos de Entrada y Salida. Algunas instrucciones del pBlazIDE difieren en su nombre o sintaxis del KCPSM3, sin embargo, su funcionamiento es igual. pBlazIDE soporta varias variaciones del KCPSM, por tal motivo debe seleccionarse la versión PicoBlaze 3 para el uso en la FPGA Spartan-3E.

El Simulador del Set de Instrucciones (ISS[2]) del pBlazIDE permite observar completamente el funcionamiento del código durante la etapa de desarrollo. La función ISS del pBlazIDE permite verificar el código fuente implementado reduciendo ampliamente el tiempo de desarrollo con el microcontrolador. Aunque, el pBlazIDE no simula la lógica de la FPGA, permite poner a prueba las funciones de entrada y salidas conectadas a la FPGA. Adicionalmente, para modelar los puertos de entrada y/o salida por el simulador es necesario usar las directivas propias del pBlazIDE.

---

[1] Mediatronix, una empresa de ingeniería y consultoría especializada en la electrónica

[2] ISS, por sus siglas en Ingles, Instruction Set Simulator

OBJETIVOS

OBJETIVO GENERAL

Crear un proyecto de diseño VHDL usando el microcontrolador PicoBlaze para la plataforma S3ESK.

OBJETIVOS ESPECIFICOS

- Sintetizar, implementar y generar el archivo de configuración de la FPGA al usarse con un microcontrolador PicoBlaze.
- Identificar las ventajas y desventajas del uso de los microcontroladores Soft-Core en sistemas basados en FPGAs.
- Identificar las ventajas y desventajas de las diferentes herramientas asociadas al microcontrolador PicoBlaze.

TRABAJO PRELIMINAR

Estudiar el documento *PicoBlaze, guía en español*, o el documento PicoBlaze 8-bit Embedded Microcontroller User Guide.

Un proyecto VHDL con microprocesador PicoBlaze, consiste en agregar el modulo KCPSM3 (CPU) y el modulo ROM, (instrucciones que seguirá durante la ejecución). El modulo KCPSM3 se encuentra disponible en el archivo KCPSM3.zip, y el modulo ROM que contiene el programa, es generado como se indica en las guías citadas. Estos dos módulos se agregan al flujo de diseño de la misma forma que se agrega cualquier otro modulo VHDL.

Una forma de crear un diseño usando el microprocesador PicoBlaze es iniciar con el uso del archivo plantilla S3ESK.vhd, que contiene la declaración de los módulos KCPSM3 y la memoria ROM, las señales de conexión entre estos dos módulos, y una descripción de la asignación de señales de entrada y salida al microcontrolador. Y para completar el proyecto se agrega al flujo de diseño los archivos embedded_kcpsm3.vhd, kcpsm3.vhd y el program_rom.vhd o como se halla nombrado el archivo .vhd generado.

PROCEDIMIENTO

Realice los puntos, luego describa y analice el proceso realizado.

1. Realice un diseño usando el microcontrolador PicoBlaze para generar la intermitencia de los leds del S3ESK, de tal forma que sea posible seleccionar diferentes configuraciones de intermitencia, “Luces del Ford”. Para ello inicialmente ensamble del código usando el KCPSM3 y compruebe su funcionamiento en el S3ESK.

2. Implemente las mismas instrucciones del código fuente anterior usando pBlazIDE, realice su ensamble y simule su funcionamiento.

3. Indique las principales diferencias entre la herramienta KCPSM3 y pBlazIDE, fortalezas y desventajas.

ANEXO D

PRÁCTICA No 4

LCD Y TECLADO PS/2

INTRODUCCIÓN

En diversos Sistemas Embebidos es necesario informar o visualizar valores, menús o instrucciones de uso, tareas que son fácilmente realizadas al implementar un Display LCD; cuyo uso se encuentra ampliamente difundido. Los displays LCD son encontrados en diferentes presentaciones en cuanto a formatos, modelos, colores y tamaños.
Por otra parte el puerto PS2 creado por IBM para la liberar el puerto serial de la conexión del teclado y aumentar la velocidad de transmisión de este, fue ampliamente difundido a los computadores personales, y actualmente a los sistemas de aplicación especifica. Cuenta con solo dos líneas para realizar la comunicación con un teclado o un mouse, que lo hace ideal para ser usado optimizando los recursos disponibles de pequeños sistemas.

Ya que la plataforma S3ESK cuenta con un display de 16 caracteres y dos líneas y un conector PS/2, se plantea su uso como parte funcional del una aplicación, como ocurre en un sistema embebido, para éste propósito en el laboratorio a desarrollar se realizaran los procedimientos necesarios para lograr su correcto funcionamiento en conjunto con el microcontrolador PicoBlaze.

OBJETIVOS

OBJETIVO GENERAL

Realizar e implementar un diseño en la S3ESK que cuente la capacidad de desplegar y capturar información para el funcionamiento de un proyecto VHDL.

OBJETIVOS ESPECIFICOS

- Programar el microcontrolador PicoBlaze para que inicialice la pantalla LCD e implementar las funciones de la pantalla como subrutinas de programa que puedan ser usadas en otros proyectos VHDL.

- Crear un modulo VHDL que permita captura una señal de comunicación serial PS/2, y suministre la información contendida en la transmisión.
- Realizar la interconexión entre diferentes módulos VHDL en un mismo proyecto VHDL.

TRABAJO PRELIMINAR

Lea la siguiente información y consulte los documentos sugeridos.

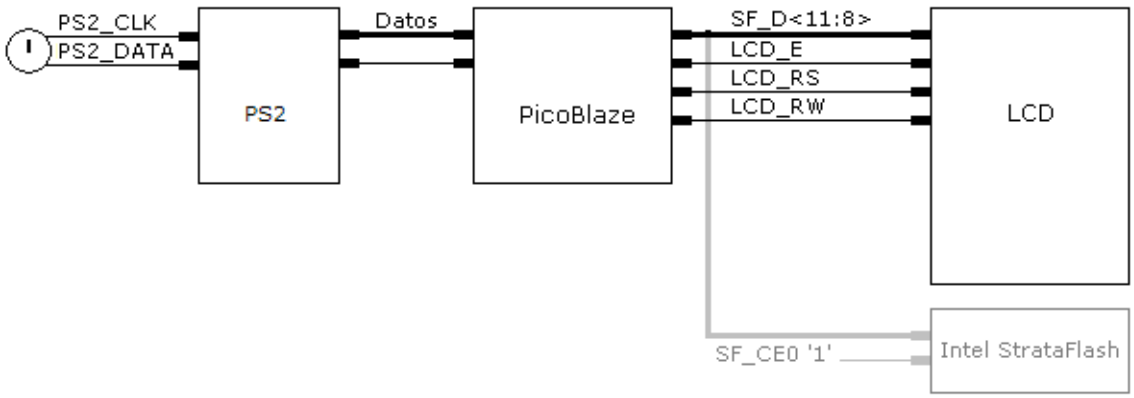


Figura 1 Arquitectura Microcontrolador con Teclado PS/2 y LCD

LCD

La S3ESK cuenta con un LCD de dos líneas por 16 caracteres. La FPGA controla el LCD mediante una interfaz de 4bit, la S3ESK implementa un interfaz de datos de 4bits con el fin de hacerla compatibilidad con otras plataformas de desarrollo Xilinx y minimizar la cantidad de pines en uso.
Una vez iniciado el LCD es una práctica forma de mostrar una variedad de información usando el estándar ASCII. Sin embargo, esta pantalla no es rápida, el desplazamiento en intervalos de medio segundo es una muestra clara, comparada con el reloj de 50MHz disponible en la plataforma.

La interfaz de datos del LCD es compartida con la memoria StrataFlash. Cuando la memoria StrataFlash se deshabilita, la aplicación en la FPGA tiene completo acceso de lectura y escritura al LCD.

Los pines de entrada/salida asignados y los estándares usados para el LCD a incluir en el UCF[1] son los siguientes:

```
NET "LCD_E"   LOC = "M18" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "LCD_RS"  LOC = "L18" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "LCD_RW"  LOC = "L17" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
```

[1] UCF User Contraints File

La interfaz de datos cuatro bits compartida con la memoria StrataFlash

```
NET "SF_D<8>"  LOC = "R15" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "SF_D<9>"  LOC = "R16" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "SF_D<10>" LOC = "P17" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "SF_D<11>" LOC = "M15" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
```

Procedimiento de inicialización, comandos de control y más información puede ser consultada en:

Controlador de caracteres LCD Sitronix ST7066U
http://www.sitronix.com.tw/sitronix/SASpecDoc.nsf/FileDownload/ST7066U614654/$FILE/ST7066Uv22.pdf
Guía de usuario de la plataforma Spartan-3E FPGA Starter Kit
http://www.digilentinc.com/Data/Products/S3EBOARD/S3EStarter_ug230.pdf

Puerto PS/2

La plataforma S3ESK dispone de un conector mini-DIN estándar de seis pines para usar como puerto PS/2. Con los pines 1 y 5 conectados a la FPGA.

Mouse y teclado usa dos hilos para la comunicación serial SP/2. Los tiempos del bus serial PS/2 son idénticos para las señales de ambos dispositivos y usan una palabra de 11 bits que incluye bit de inicio, parada y paridad impar. La comunicación es realizada de forma diferente para cada periférico.

La señal de reloj y datos solo son modificadas cuando ocurre una transmisión, de lo contrario permanece en un estado lógico alto. Los tiempos definen los requerimientos para la comunicación host-mouse y comunicación bidireccional del teclado. El mouse o teclado escriben un bit en la línea de datos cuando la señal de reloj está en alto, y el host lee la línea de datos cuando el reloj está en bajo.

Los pines asignados para el PS/2 a incluir en el UCF son los siguientes:

```
NET "PS2_CLK"  LOC = "G14" | IOSTANDARD = LVCMOS33 | DRIVE = 8 | SLEW = SLOW ;
NET "PS2_DATA" LOC = "G13" | IOSTANDARD = LVCMOS33 | DRIVE = 8 | SLEW = SLOW ;
```

Documentación del Puerto PS/2, como parámetros de tiempo del bus puede ser consultada en:

Protocolo de comunicación PS/2
http://www.computer-engineering.org/ps2protocol/

Interfaz Teclado PS/2
http://www.computer-engineering.org/ps2keyboard/

Interfaz Mouse PS/2
http://www.computer-engineering.org/ps2mouse/

PROCEDIMIENTO

En esta práctica se hará el diseño de una arquitectura compuesta por un microcontrolador PicoBlaze, un modulo VHDL para el puerto de entrada PS/2 y una interfaz para una pantalla LCD, es decir, se diseñará los módulos en VHDL necesarios para la adquisición de datos mediante un puerto PS/2, su transferencia al microcontrolador PicoBlaze y su visualización en una pantalla LCD.

1. Escriba el código ensamblador para el microcontrolador PicoBlaze que permita inicializar la pantalla LCD.
2. Escriba en código ensamblador las subrutinas que permita usar las funciones de la pantalla LCD, como son escribir carácter, borrar pantalla, desplazamiento etc.
3. Verifique el funcionamiento en conjunto del código de inicialización y subrutinas, escribiendo en el LCD su nombre, permita que rote y mediante el uso de un pulsador cambiar el mensaje por el nombre de la plataforma de trabajo.
4. Realice la descripción de un modulo que haciendo uso de las señales PS2_data y PS2_clock, entregue la información contenida en la comunicación.
5. Mediante el uso conjunto de las herramientas de software y de hardware creadas, construya la aplicación que permita mostrar en el LCD el carácter asignado al pulsar una tecla. (no el scan code, código asignado a la tecla)

ANEXO E

PRÁCTICA No 5

VGA Y TECLADO PS/2

INTRODUCCIÓN

En un Sistema Embebido el manejo y manipulación de los datos e información es realizada por el microcontrolador o microprocesador. En este laboratorio aprovecharemos el microcontrolador PicoBlaze para procesar los datos de entrada, y dependiendo de estos modificar el contenido de la video memoria para poder observar en un monitor VGA el resultado obtenido, y en lo posible accesar alguna de las memorias de la S3ESK, dada la gran capacidad de almacenamiento que ofrecen estas.

OBJETIVOS

OBJETIVO GENERAL

Crear un proyecto que pueda recibir información por medio del teclado y mostrarla usando un monitor VGA.

OBJETIVOS ESPECIFICOS

- Implementar el uso de la interrupción por hardware con que cuenta el microcontrolador PicoBlaze para realizar las operaciones necesarias para las condiciones que se establezcan en el diseño.
- Accesar y modificar el contenido de la una memoria, ya sea de la FPGA o de la S3ESK.

TRABAJO PRELIMINAR

Leer, analizar y entender la arquitectura planteada, y realizar sugerencias para obtener un mejor desempeño del propósito de esta al usar la S3ESK.

MICROCONTROLADOR PICOBLAZE CON SALIDA DE VIDEO VGA

La arquitectura posee el generador de Video VGA, video memoria y el microcontrolador PicoBlaze. El generador de Video VGA constantemente accede a la video memoria para obtener el estado del píxel que esta dibujando, y el microcontrolador puede acceder la video memoria para cambiar el contendido de esta y en consecuencia modificar la salida VGA.

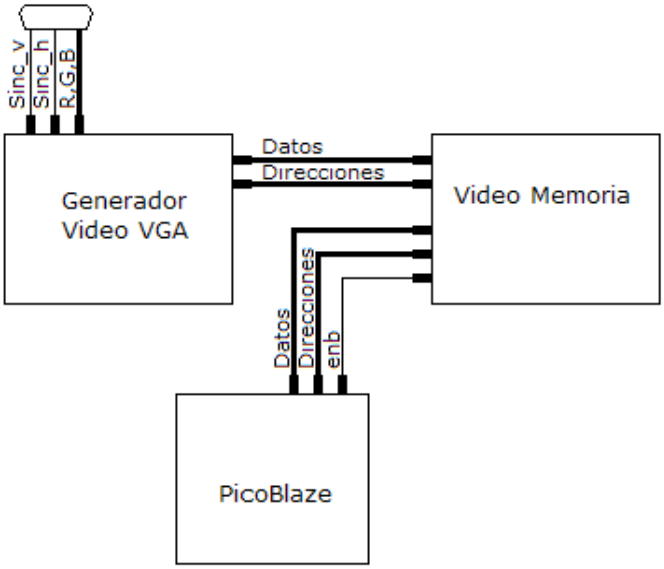


Figura 1 Arquitectura Microcontrolador PicoBlaze con Salida de Video VGA

GENERADOR DE VIDEO VGA

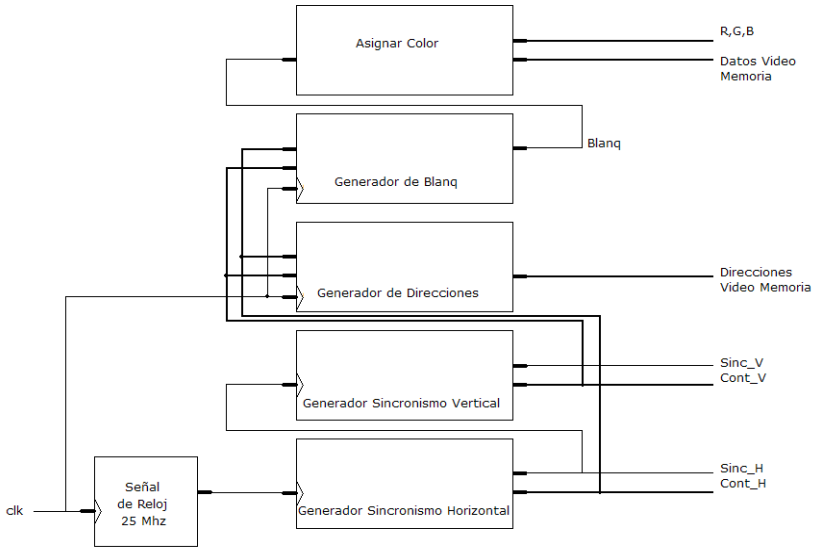


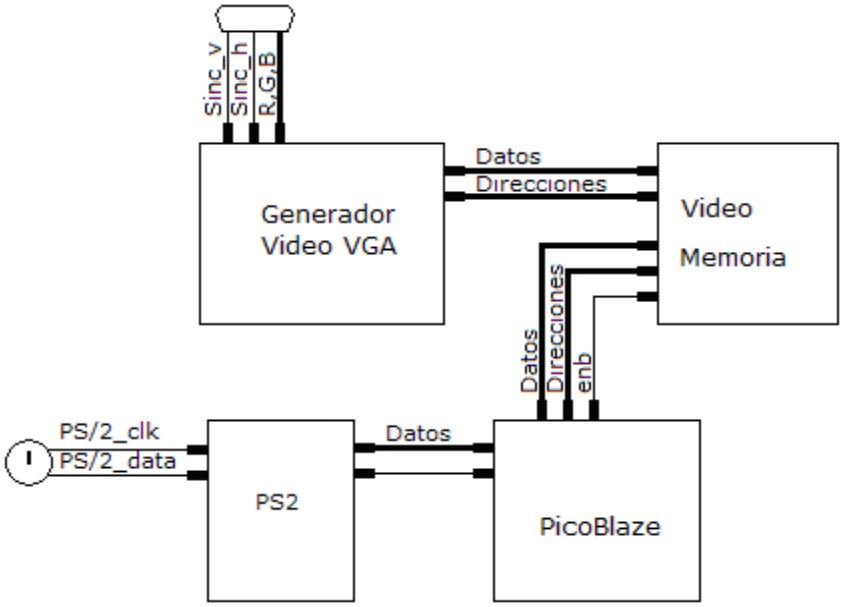
Figura 2 Generador de Señal de Video VGA

El generador de señal VGA, esta encargado de sincronizar los datos de salida R, G y B con las señales de sincronismo Sinc_V y Sinc_H, producidas por el mismo generador de video VGA.

El bloque Señal de Reloj 25 MHz, se encarga de dividir la frecuencia del reloj del sistema, para que en los Generadores de Sincronismos se obtengan tiempos acordes a los establecidos en la descripción de los tiempos VGA en el documento *Spartan-3E Starter KitBoard User Guide*. Los Generadores de Sincronismo están encargados de producir las señales de sincronismo y los contadores. El generador de direcciones haciendo uso de los contadores determina la posición de memoria que debe ser accedida para un determinar el color de un píxel en la pantalla. El bloque Blanq genera una señal *blanq* que es usada para indicar la parte de las señales de sincronismo que es visible en el monitor.
Los datos de color generados R, G y B son asignados de los datos recibidos de la video memoria, y se validan durante los tiempos acordes a las señales de sincronismo, valiéndose de la señal *blanq* para realizar esta validación.

PROCEDIMIENTO

1. Escriba un programa para el microcontrolador PicoBlaze y la lógica necesaria de entrada, para capturar la señal de entrada del ps/2 procedente de un teclado e interpretarla como los datos del movimiento de un puntero en el monitor.



2. Escriba un programa para el microcontrolador PicoBlaze que modifique el contenido de la video memoria para mostrar el logo de la UIS junto con un contador que permita controlar su incremento, decremento o reinicio por medio del teclado

ANEXO F

Introducción A La S3ESK

ÍNDICE DE CONTENIDO

ÍNDICE DE FIGURAS

INDICE DE TABLAS

INTRODUCCIÓN

Los sistemas de desarrollo poseen diferentes dispositivos que son configurados de la mejor forma posible para permitir la ejecución de gran variedad de aplicaciones. La Spartan 3E Starter Kit Board, posee una arquitectura bastante flexible debido a que cuenta con hardware reconfigurable como lo es una FPGA y un CPLD, diferentes opciones de memoria y formas de compartir y obtener información; nos permite decir que puede ser parte de una aplicación de sistema embebido si cuenta con un software que use el hardware descrito.

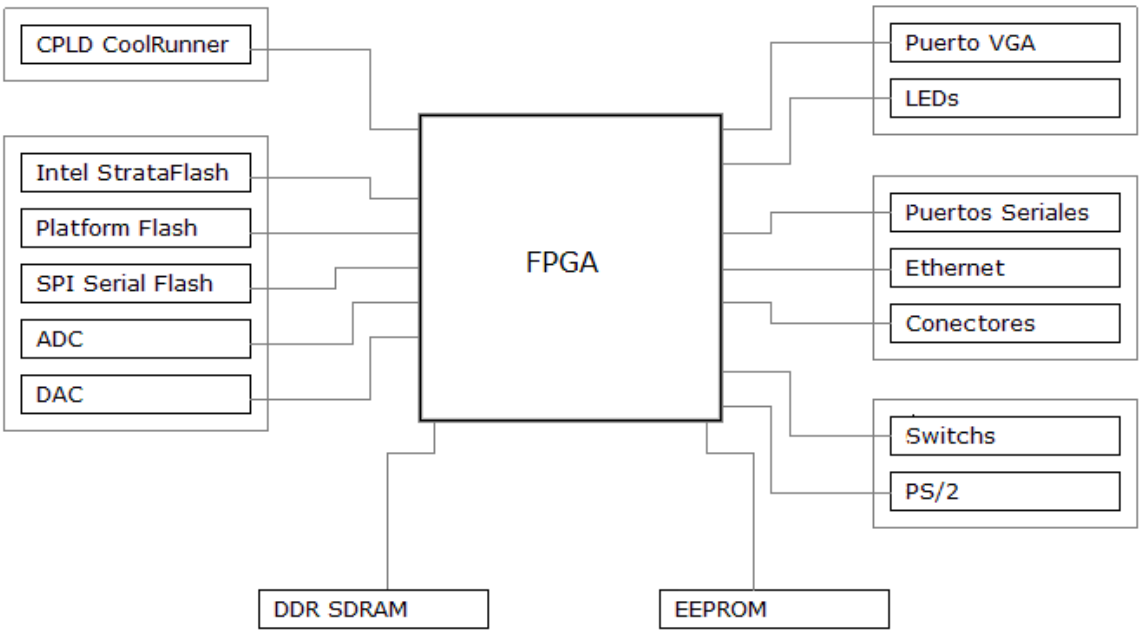


Figura 1: Arquitectura Spartan-3E Starter Kit Board

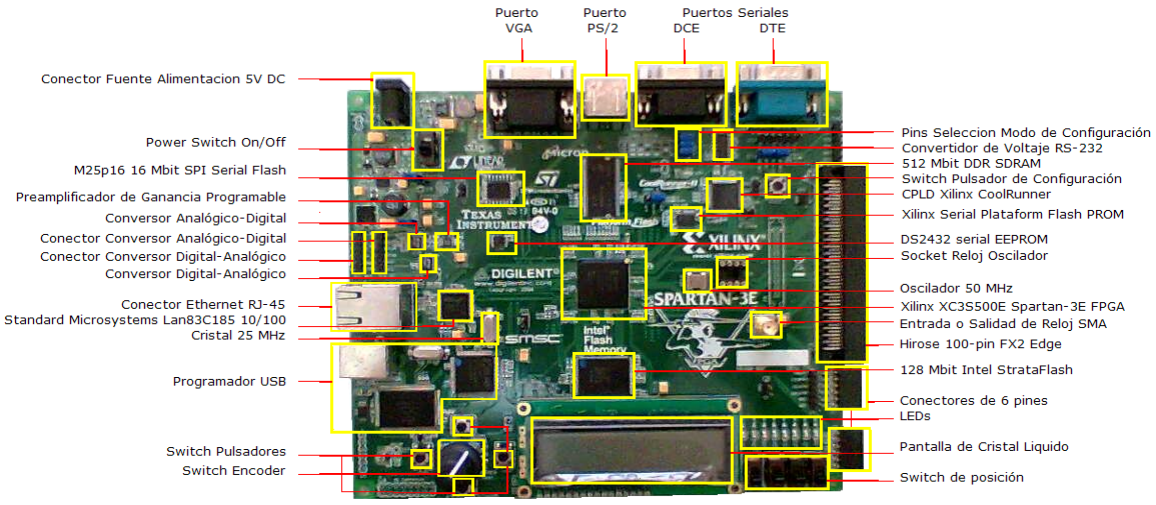


Figura 2: Board del Spartan 3E Starter Kit (S3ESK)

El dispositivo principal de lógica programable con el que cuenta la board es una FPGA Xilinx de la familia Spartan-3E, también dispone de un CPLD Xilinx de la familia CoolRunner-II. Las memorias disponibles en el sistema de desarrollo son de dos propósitos, configuración de la FPGA o como dispositivos de almacenamiento. Y como dispositivos de comunicación se tiene una interfaz física de EtherNET y conectores seriales, con sus convertidores de voltajes; para la entrada y salida de datos cuenta con un puerto VGA, pantalla de cristal liquido, ocho leds, cinco switch de posición, cuatro switch pulsadores, un switch encoder, tres conectores de 6 pines, un puerto PS/2, un conector de 100 pines Hirose FX2 Edge. Además, cuenta con un convertidor analógico-digital y un convertidor digital-analógico.
La S3ESK incluye en board una interfaz USB basada en JTGA. Para programación fuera de board mediante JTGA puede realizarse usando el Xilinx Plataform USB Cable.

Algunos textos e imágenes en el presente documento son una versión realizada y adaptada al español del documento UG230 Xilinx Spartan-3E Starter Kit Board User Guide, como documentación resultante del proyecto PROPUESTA DE PRACTICAS DE LABORATORIO PARA SISTEMAS EMBEBIDOS BASADOS EN SPARTAN 3E STARTER KIT DE XILINIX

LA FPGA

La FPGA disponible en la S3ESK es un dispositivo de la familia Spartan 3E de Xilinx, es una alternativa superior dentro de la gama Spartan al aumentar la cantidad de entradas/salidas y reducir significantemente el costo por celda lógica. Esta familia de FPGAs es una alternativa superior a un ASICs[1], reduce el costo inicial, los largos ciclos de desarrollo, y la inherente flexibilidad del convencional ASIC.

CARACTERÍSTICAS

Muy bajo costo, alto desempeño para soluciones de alto volumen, aplicaciones orientadas a consumidor

Provee avanzados procesos de tecnología de 90 nanómetros.

- Multi-voltaje, Multi-estándar pins SelectIO™
  - Arriba de 376 pins E/S o 156 señales diferenciales
  - LCVMOS, LVTTL, HSTL Y SSTL
  - 3.3 V, 2.5V 1.8V, 1.5V y 1.2
  - Velocidad de transferencia por pins E/S de 662 Mbit/s
  - LVDS, RSDS, mini-LVDS, diferencial HSTL/SSTL, E/S diferencial
  - Soporta Double Data Ratte (DDR)
  - Soporta arriba de 333 Mb/s DDR SDRAM
- Abundantes recursos lógicos
  - Densidad arriba de 33.192 celdas lógicas, un registro opcional de desplazamiento o soporte RAM
  - Multiplexores altamente eficientes
  - Rápida predicción de acarreo lógico
  - Aumento de 18 x 18 multiplicadores con la opción de pipeline
  - Programación JTAG IEEE 1149.1/1532
- Arquitectura de memoria SelectRAM ™
  - Hasta 648 Kbits de Bloques RAM
  - Hasta 231 Kbits of efficient distributed RAM
- Hasta ocho Digital Clock Managers (DCMs)
  - Eliminación de sesgo de reloj (Clock skew)
  - Frecuencia de síntesis, multiplicación, división
  - Alta resolución de desplazamiento de fase
  - Amplio Rango de Frecuencias (5 MHz to over 300 MHz)
- Ocho relojes globales más ocho adicionales para cada mitad del dispositivo.
- Interfaz de configuración PROMs de standards industriales
  - SPI serial Flash PROM

[1] ASIC Application Specific Integrate Circuit.

- NOR Flash PROM paralela x8 o x8/x16
- Xilinx Platform Flash con JTAG.

- Procesadores embebidos MicroBlaze™ y PicoBlaze™.

Tabla 1: Atributos generales de la FPGA disponible en la S3ESK

Dispositivo	System Gates	Celdas Logicas Equivalentes	Filas	Columnas	Total CLBs	Total Slices
XC3S500E	500K	10,476	46	34	1,164	4.656

RAM Distribuida	Bloque RAM	DCMs	Multiplicadores Dedicados	Máximas E/S
73K	360	4	20	232

ARQUITECTURA SPARTAN 3E

La arquitectura de la familia Spartan 3E consiste fundamentalmente de cinco elementos funcionales programables.

Bloques Lógicos Configurables (Configurable Logic Blocks CLBs), constituye el principal recurso lógico para implementación sincrona. Cada CBL contiene 4 slices, y cada slice contiene dos tablas Look-Up (LUTs) para implementación lógica y dos dedicadas para almacenar elementos que pueden ser usados como flip-flops o latches. Las LUTs pueden ser usadas como una memoria 16x1 (RAM16) o como registro de 16-bit (SRL16).

Bloques de Entradas/Salidas (Input/Output Blocks IOBs) controla el intercambio de datos entre los pines E/S y la lógica interna del dispositivo. Cada E/S soporta intercambio bidireccional de datos. Soporta una variedad standards de señales, incluyendo cuatro standards diferenciales de alto desempeño, incluidos los registros de doble velocidad de transferencia de datos (Double Date Rate DDR).

Bloque RAM, provee almacenamiento de datos desde dos bloques dual-port de 18 Kbits.

Bloques Multiplicadores, acepta dos números binarios de 18 bits como entradas y calcula el producto.

Bloques Manejadores de Reloj Digital (Digital Clock Manager Blocks DCMs) provisto de auto-calibración, completa solución digital distribuyendo, retrasando, multiplicando, dividiendo, y corriendo en fase señales de reloj.

Estos elementos estan organizados como muestra la Figura 3, las E/S bordean los arreglos de CBLs. Cada dispositivo tiene dos columnas de bloques RAM, cada columna consiste de 18Kbits. Cada bloque es asociado con un multiplicador dedicado. La familia posee una completa red de rutas que interconectan todos los cinco elementos funcionales, transmitiendo señales entre ellos. Cada elemento funcional tiene asociado una matriz de switch que permite múltiples conexiones para trazar.

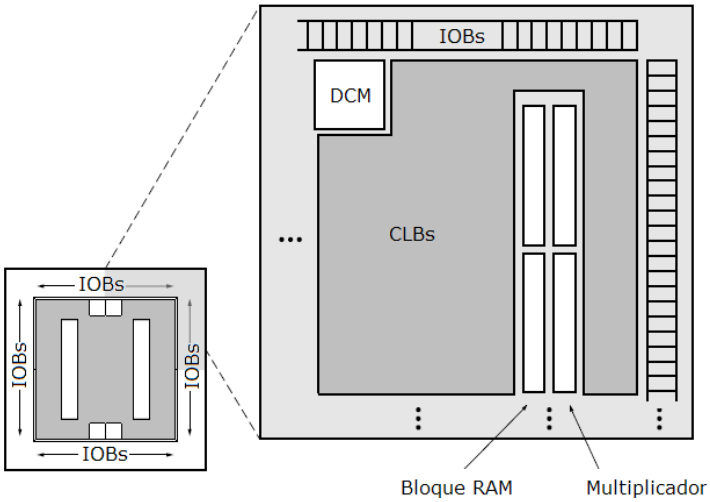


Figura 3: Estructura de una FPGA

CLBs

Los CLBs son agrupados en un arreglo regular de columnas y filas, como se muestra en la Figura 4.

Cada CLB comprende cuatro slices interconectadas, estas slices son agrupadas en pares, cada par es organizado como una columna con un acarreo independiente. El par izquierdo soporta lógica y funciones de memoria, estas slices son llamadas SLICEM. El par derecho soporta solo lógica y estas slices son llamadas SLICEL. De esta forma la mitad de las LUTs soportan lógica y memoria (RAM16 y SRL16). Los SLICEL reducen el tamaño de los CLB y minimiza el costo del dispositivo, y también puede proveer un mejor desempeño sobre el SLICEM. Para profundizar en el diseño, circuitería y funcionamiento de la FPGA consultar en [13].

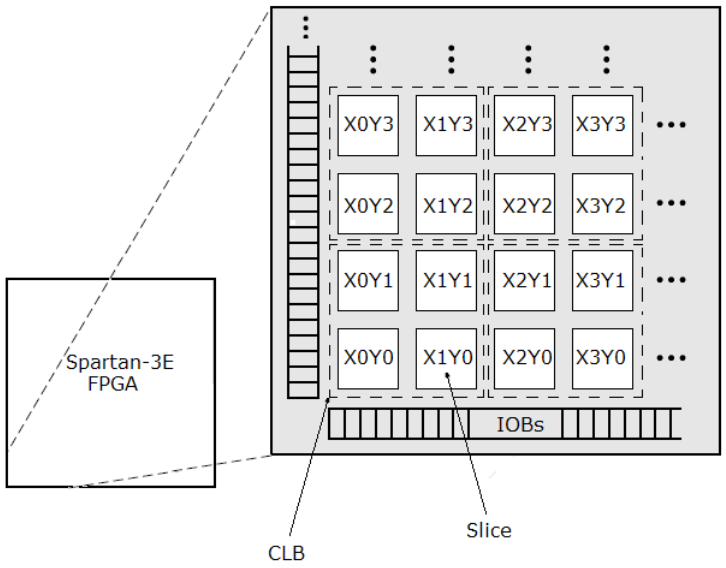


Figura 4: CLBs

CONFIGURACIÓN

Las FPGAs son programadas al cargar los datos de configuración, estos datos pueden ser almacenados externamente en una PROM o algún otro medio no-volátil, o fuera de board. Después de energizar, los datos de configuración son escritos en la FPGA usan alguno de los siguientes siete modos:

- Master Serial desde una Xilinx Platform Flash PROM
- Serial Peripheral Interface (SPI) desde una SPI serial Flash
- Byte Peripheral Interface (BPI), desde la parte alta o baja de una x8 o x8/x16 Parallel NOR Flash
- Slave Serial, típicamente descargado desde un procesador
- Slave Parallel, típicamente descargado desde un procesador
- Boundary Scan (JTAG), típicamente descargado desde un procesador o un sistema prueba.

Para el caso de la S3ESK se han implementado cinco de estas formas de configuración brindando la posibilidad de realizar diferentes y complejas aplicaciones.

La función de una FPGA Spartan-3E es definida por cargar los datos de configuración de aplicación específica. Este proceso de configuración usa una serie de pintes, algunos de ellos son dedicados para configuración, otros pines son solo usados y regresados a la aplicación para usar como E/S de propósito general después de completar la configuración.

Esta FPGA dispone de tres pines M2, M1 y M0, para seleccionar el modo deseado de configuración. La configuración de los pines para los diferentes modos se observan en Tabla 2.

En la S3ESK se tiene acceso a estos pines, aunque se cuenta con todas las opciones de configuración con la que cuenta la FPGA. Los valores de los pines de modo son tomados durante el inicio de la configuración cuando INIT_B es puesto en alto. Después de la completar la configuración, los pines de modo están disponibles como E/Ss.

Tabla 2: Descripción de los modos de configuración

	Master Serial	**SPI**	**BPI**	**Slave Parallel**	**Slave Serial**	**JTAG**
M[2:0] configuración de los pines	<0:0:0>	<0:0:1>	<0:1:0>=Alta <0:1:1>=Baja	<1:1:0>	<1:1:1>	<1:0:1>
Ancho de los Datos	Serial	Serial	Byte	Byte	Serial	Serial
Fuente de configuración de la memoria	Xilinx Platform Flash	Estándar Industrial SPI serial Flash	Standard Industrial NOR parallel Flash o Xilinx parallel Platform Flash	Alguna fuente vía microcontrolador, CPU, Xilinx parallel Platform Flash, etc.	Alguna fuente vía microcontrolador, CPU, Xilinx Platform Flash, etc.	Alguna fuete vía microcontrolador, CPU, SYStem ACE, etc.
Fuente De Reloj	Oscilador Interno	Oscilador Interno	Oscilador Interno	Reloj externo o pin CCLK	Reloj externo o pin CCLK	Reloj externo o pin TCK
Total de Pines E/S usados durante la programación	8	13	46	21	8	0
Modo de configuración	Slave Serial	Slave Serial	Slave Parallel	Slave Parallelor Memory Mapped	Slave Serial	JTAG

IMAGEN BITSTERAM DE CONFIGURACIÓN

Una FPGA Spartan-3E específica siempre requiere un número constante de bits de configuración, sin importar la complejidad del diseño. S3ESK dispone de una XC3S500E que requiere 2.270.208 bits de configuración.

Para la configuración la FPGA tiene seis pines dedicados, incluyendo los pines DONE y PROG_B, y los pines cuatro pines JTAG. Todos los otros pines son de configuración de doble propósito E/S y están disponibles después que el pin Done se pone un alto.

FUENTES DE RELOJ

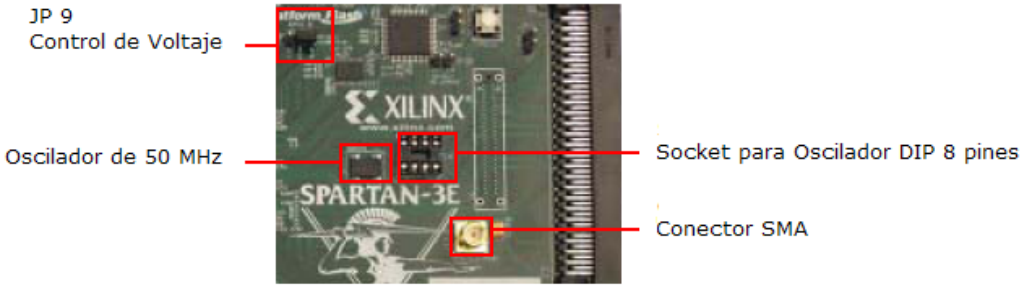


Figura 5: Ubicación en la S3ESK de las fuentes de reloj disponibles

La S3ESK posee tres posibles fuentes de reloj para los dispositivos. En board cuenta con un oscilador de 50 MHz de la serie Epson SG-8002JF que se encuentra conectador directamente a la FPGA en el pin C9.
Dispone de un conector estilo SMA[2] para una fuente reloj externa; también usado como salida para señales de reloj o señales de alta velocidad generadas por la Fpga. La conexión con la FPGA esta disponible en el pin A10.
Adicionalmente posee un socket DIP para hacer posible instalar un oscilador de este tipo, que esta conectado en el pin B8 de la FPGA.

Tabla 3: Fuentes de Reloj, Buffer Global y DCM Asociados

Reloj	Pin FPGA	Buffer Global	DCM Asociado
CLK_50MHZ	C9	GCLK10	DCM_X0Y1
CLK_AUX	B8	GCLK8	DCM_X0Y1
CLK_SMA	A10	GCLK7	DCM_X1Y1

El voltaje para todos los pines E/S del banco 0 son controlados por el Jumper JP9, y los recursos de reloj también son controlados por el JP9, el se encuentra establecido normalmente a 3.3V. En la board el dispositivo oscilador es de 3.3V y no funcionara de de la forma esperada cuando el Jumper JP9 este a 2.5V

La S3ESK también cuenta con 2 cristales osciladores mas, usados para la interfaz EtherNET y para el microcontrolador que realiza la comunicación USB encargada de la configuración de la FPGA.

[2] SMA, (SubMiniature Version A) Conector coaxial de RF.

UCF Constraints

Las entradas de reloj requieren dos tipos diferentes de constraints. La constraits de localización define la asignación de pines E/S y el estándar de E/S. Las constraints de periodo definen el periodo del reloj, consecuentemente la frecuencia del reloj, y el ciclo de duración de la señal de reloj entrante.

Localización:

```
NET "CLK_50MHZ"     LOC = "C9"    | IOSTANDARD = LVCMOS33 ;
NET "CLK_SMA"       LOC = "A10"   | IOSTANDARD = LVCMOS33 ;
NET "CLK_AUX"       LOC = "B8"    | IOSTANDARD = LVCMOS33 ;
```

Esta configuración asume que el jumper JP9 esta puesto a 3.3V, si no es el caso, se tiene que ajustar la configuración IOSTANDARD acorde.

DISPOSITIVOS DISPONIBLES EN LA S3ESK

MEMORIAS

Las memorias disponibles en la S3ESK pueden ser de dos usos, que son configuración de la FPGA o almacenamiento de datos; algunas de ellas pueden usarse para los dos propósitos.
Xilinx 4 MBit Serial Platform Flash PROM
16 MByte (128 Mbit) Nor Flash (Intel StrataFlash)
16 Mbit SPI Serial Flash (STMicro)
64 MByte (512 Mbit) DDR (Double Data Rate) SDRAM

Xilinx Serial Platform Flash PROM

Es una solución que brinda Xilinx para la configuración de las diferentes familias de FPGAs que produce. La XCF04S permite almacenar el archivo de configuración de la FPGA con la que cuenta la S3ESK y ofrece diferentes modos de realizar esta configuración, que dependerá de las necesidades del sistema en el que se use. La S3ESK tiene establecido el modo Master Serial, de otros seis modos posibles de uso de la Flash PROM, no de la FPGA en si, para realizar la configuración mediante el uso de esta PROM.
La Platform Flash PROM XCF04S tiene una densidad de 4 MBit, suficientes para contener los datos de configuración de la FPGA disponible en la S3ESK. La configuración de la FPGA es el único uso disponible para esta Flash PROM

Intel StrataFlash Parallel NOR Flash PROM

Es posible usar para:

- Realizar la configuración de la FPGA. Es posible usarse para almacenar una o dos configuraciones para la FPGA y dinámicamente cambiar entre los dos usando MultiBoot Spartan-3E.
- Almacenar y ejecutar código del procesador MicroBlaze directamente desde el dispositivo StrataFlash.
- Almacenar código del procesador MicroBlaze y realizar "Shadowing Code"[3] en la memoria DDR antes de ejecutar el código.
- Almacenar datos para la FPGA.

[3] Shadowing Code, el código asociado a una cache RAM es almacenado en una memoria no volátil, después de energizada la aplicación es descargado en la RAM donde se ejecuta por el procesador [1][10]

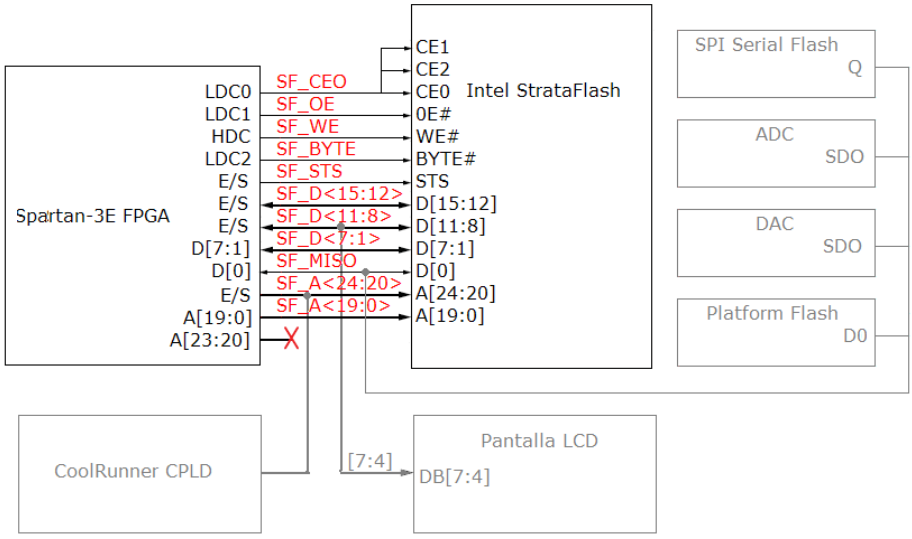


Figura 6: Conexiones de la StrataFlash, la FPGA y otros dispositivos de la S3ESK

En general la StrataFlash esta conectada a la XC3S500E para soportar configuración BPI (Byte Peripheral Interface) y provee 128 Mbits, además la interfaz en board permite usar una StrataFlash de 256 Mbits. Los cuatro últimos bits de dirección de la FPGA A[23:19] no se encuentran conectados directamente a la StrataFlash; el CPDL controla estos pines durante la configuración.

Tabla 4: Conexiones StrataFlash

Categoría	StrataFlash Nombre Señal	Función
Dirección	SF_A<24:20>	Compartidos con el CPLD. El CPLD maneja los pines durante la configuración de la FPGA. También conectados con la FPGA pines E/S.
Dirección	SF_A<0:19>	Conectado a la FPGA para soportar la configuración BPI
Datos	SF_D<8:15>	Ocho bit altos de media palabra de 16bits cuando la StrataFlash es configurada para datos x16. Conectados a la FPGA SF_D<11:8> conectados a la pantalla LCD
Datos	SF_D<1:7>	7 bits altos de un byte o 8 bits bajos de media palabra de 16 bits. Conectada a la FPGA pines D[7:1] para soportar configuración BPI
Datos	SPI_MISO	Bit 0 de un byte y de media palabra de 16 bits. Conectado a la FPGA pin D0/DIM para soportar configuración BPI. Compartido con otro periférico SPI y Platform Flahs PROM
Control	SF_CE0	Habilitar Chip StrataFlash. Conectado con pin LDC0.
Control	SF_WE	Habilitar escritura StrataFlash. Conectado con pin HDC.
Control	SF_OE	Habilitar Chip StrataFlash. Conectado con pin LDC1

Control	SF_BYTE	Habilitar Byte StrataFlash. 0: x8 data 1: x16 data
Control	SF_STS	Señal de estado de la StrataFlash

Conexiones Compartidas

La memoria StrataFlash comparte algunas conexiones con otros componentes de la S3ESK.

La pantalla LCD comparte las conexiones de datos con las señales SF_D<11:8> de la memoria StrataFlash. La FPGA controla el acceso a la StrataFlash o a la pantalla LCD usando la señales, SF_CE0, SF_OE#, SF_WE# y LCD_RW. Para usar la pantalla LCD estas tres señales de control de la StrataFlash debe ser puesta en alto para que la memoria no interfiera con la pantalla LCD y para evitar que inadvertidamente modificar el contenido de la memoria.

El CPLD controla las cinco líneas superiores dirección de la StrataFlash durante la configuración. Las cuatro líneas superiores de dirección del modo BPI desde la FPGA no están conectadas.

El display LCD utiliza cuatro bits de datos que se comparten con las señales SF_D <11:8> de la StrataFlash PROM. La FPGA controla el acceso a la StrataFlash PROM o el display LCD utilizando las señales SF_CE0 y LCD_RW.
Si SF_CE0 y LCD_RW se encuentran en alto la FPGA lee desde el display LCD, y si SF_CE0 y LCD_RW se encuentra en bajo la FPGA accede a la StrataFlash PROM.

La línea de datos menos significativa, SF_D<0>, es compartida con la señal de salida de datos de los periféricos SPI, SPI_MISO, y la salida serial de la Platafom Flash PROM. Las aplicaciones de la FPGA debe asegurarse que solo una fuente de datos esta activa en algún momento.

La línea más significativa de dirección, SF_A<24>, no esta físicamente usada en la StrataFlash PROM de 16Mbytes. Esto es provisto para una migración a una StrataFlash más grande en un igual Package Footprint.

UCF Localización

Dirección

```
NET "SF_A<24>" LOC = "A11"    | IOSTANDARD = LVCMOS33       | DRIVE = 4    | SLEW = SLOW ;
NET "SF_A<23>" LOC = "N11"    | IOSTANDARD = LVCMOS33       | DRIVE = 4    | SLEW = SLOW ;
NET "SF_A<22>" LOC = "V12"    | IOSTANDARD = LVCMOS33       | DRIVE = 4    | SLEW = SLOW ;
NET "SF_A<21>" LOC = "V13"    | IOSTANDARD = LVCMOS33       | DRIVE = 4    | SLEW = SLOW ;
NET "SF_A<20>" LOC = "T12"    | IOSTANDARD = LVCMOS33       | DRIVE = 4    | SLEW = SLOW ;
NET "SF_A<19>" LOC = "V15"    | IOSTANDARD = LVCMOS33       | DRIVE = 4    | SLEW = SLOW ;
NET "SF_A<18>" LOC = "U15"    | IOSTANDARD = LVCMOS33       | DRIVE = 4    | SLEW = SLOW ;
NET "SF_A<17>"  LOC = "T16"   | IOSTANDARD = LVCMOS33       | DRIVE = 4    | SLEW = SLOW ;
NET "SF_A<16>" LOC = "U18"    | IOSTANDARD = LVCMOS33       | DRIVE = 4    | SLEW = SLOW ;
NET "SF_A<15>" LOC = "T17"    | IOSTANDARD = LVCMOS33       | DRIVE = 4    | SLEW = SLOW ;
NET "SF_A<14>" LOC = "R18"    | IOSTANDARD = LVCMOS33       | DRIVE = 4    | SLEW = SLOW ;
NET "SF_A<13>" LOC = "T18"    | IOSTANDARD = LVCMOS33       | DRIVE = 4    | SLEW = SLOW ;
NET "SF_A<12>" LOC = "L16"    | IOSTANDARD = LVCMOS33       | DRIVE = 4    | SLEW = SLOW ;
NET "SF_A<11>" LOC = "L15"    | IOSTANDARD = LVCMOS33       | DRIVE = 4    | SLEW = SLOW ;
NET "SF_A<10>" LOC = "K13"    | IOSTANDARD = LVCMOS33       | DRIVE = 4    | SLEW = SLOW ;
NET "SF_A<9>"  LOC = "K12"    | IOSTANDARD = LVCMOS33       | DRIVE = 4    | SLEW = SLOW ;
NET "SF_A<8>"  LOC = "K15"    | IOSTANDARD = LVCMOS33       | DRIVE = 4    | SLEW = SLOW ;
NET "SF_A<7>"  LOC = "K14"    | IOSTANDARD = LVCMOS33       | DRIVE = 4    | SLEW = SLOW ;
NET "SF_A<6>"  LOC = "J17"    | IOSTANDARD = LVCMOS33       | DRIVE = 4    | SLEW = SLOW ;
NET "SF_A<5>"  LOC = "J16"    | IOSTANDARD = LVCMOS33       | DRIVE = 4    | SLEW = SLOW ;
NET "SF_A<4>"  LOC = "J15"    | IOSTANDARD = LVCMOS33       | DRIVE = 4    | SLEW = SLOW ;
NET "SF_A<3>"  LOC = "J14"    | IOSTANDARD = LVCMOS33       | DRIVE = 4    | SLEW = SLOW ;
NET "SF_A<2>"  LOC = "J12"    | IOSTANDARD = LVCMOS33       | DRIVE = 4    | SLEW = SLOW ;
NET "SF_A<1>"  LOC = "J13"    | IOSTANDARD = LVCMOS33       | DRIVE = 4    | SLEW = SLOW ;
NET "SF_A<0>"  LOC = "H17"    | IOSTANDARD = LVCMOS33       | DRIVE = 4    | SLEW = SLOW ;
```

Datos

```
NET "SF_D<15>" LOC = "T8"     | IOSTANDARD = LVCMOS33       | DRIVE = 4    | SLEW = SLOW ;
NET "SF_D<14>" LOC = "R8"     | IOSTANDARD = LVCMOS33       | DRIVE = 4    | SLEW = SLOW ;
NET "SF_D<13>" LOC = "P6"     | IOSTANDARD = LVCMOS33       | DRIVE = 4    | SLEW = SLOW ;
NET "SF_D<12>" LOC = "M16"    | IOSTANDARD = LVCMOS33       | DRIVE = 4    | SLEW = SLOW ;
NET "SF_D<11>" LOC = "M15"    | IOSTANDARD = LVCMOS33       | DRIVE = 4    | SLEW = SLOW ;
NET "SF_D<10>" LOC = "P17"    | IOSTANDARD = LVCMOS33       | DRIVE = 4    | SLEW = SLOW ;
NET "SF_D<9>"  LOC = "R16"    | IOSTANDARD = LVCMOS33       | DRIVE = 4    | SLEW = SLOW ;
NET "SF_D<8>"  LOC = "R15"    | IOSTANDARD = LVCMOS33       | DRIVE = 4    | SLEW = SLOW ;
NET "SF_D<7>"  LOC = "N9"     | IOSTANDARD = LVCMOS33       | DRIVE = 4    | SLEW = SLOW ;
NET "SF_D<6>"  LOC = "M9"     | IOSTANDARD = LVCMOS33       | DRIVE = 4    | SLEW = SLOW ;
NET "SF_D<5>"  LOC = "R9"     | IOSTANDARD = LVCMOS33       | DRIVE = 4    | SLEW = SLOW ;
NET "SF_D<4>"  LOC = "U9"     | IOSTANDARD = LVCMOS33       | DRIVE = 4    | SLEW = SLOW ;
NET "SF_D<3>"  LOC = "V9"     | IOSTANDARD = LVCMOS33       | DRIVE = 4    | SLEW = SLOW ;
NET "SF_D<2>"  LOC = "R10"    | IOSTANDARD = LVCMOS33       | DRIVE = 4    | SLEW = SLOW ;
NET "SF_D<1>"  LOC = "P10"    | IOSTANDARD = LVCMOS33       | DRIVE = 4    | SLEW = SLOW ;
NET "SPI_MISO" LOC = "N10"    | IOSTANDARD = LVCMOS33       | DRIVE = 6    | SLEW = SLOW ;
```

Control

```
NET "SF_BYTE"  LOC = "C17"    | IOSTANDARD = LVCMOS33       | DRIVE = 4    | SLEW = SLOW ;
NET "SF_CE0"   LOC = "D16"    | IOSTANDARD = LVCMOS33       | DRIVE = 4    | SLEW = SLOW ;
NET "SF_OE"    LOC = "C18"    | IOSTANDARD = LVCMOS33       | DRIVE = 4    | SLEW = SLOW ;
NET "SF_STS"   LOC = "B18"    | IOSTANDARD = LVCMOS33       | DRIVE = 4    | SLEW = SLOW ;
NET "SF_WE"    LOC = "D17"    | IOSTANDARD = LVCMOS33       | DRIVE = 4    | SLEW = SLOW ;
```

STMicroelectronics SPI Serial Flash

Se usa en una variedad de aplicaciones. La SPI Flash es una alternativa para configurar la FPGA. La SPI Flash también esta disponible para la FPGA después de la configuración para una variedad de propósitos, como son:

- Almacenar datos
- Almacenar códigos de identificación, números seriales, direcciones IP, etc.
- Almacenar el código de para procesador MicroBlaze y puede ser movido a DDR SDRAM.

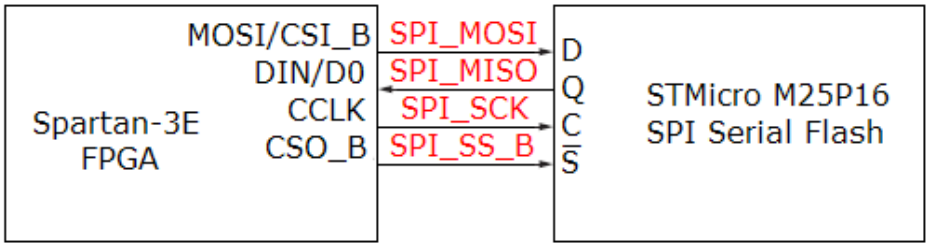


Figura 7: Conexiones de la M25P16 con la FPGA

La M25P16 es una memoria de 16MBit (2M x 8), con avanzado mecanismo de protección de escritura, accesada por alta velocidad en un bus SPI. Puede ser programada de 1 a 256 bytes al tiempo. Esta organizada en 32 sectores, cada sector de 256 páginas, y cada página de 256 byte [9].

Descripción de las señales

Salida serial de datos SPI_MISO, señal de salida usada para trasmitir datos en serie. Los datos son desplazados a la salida en un flanco de a la salida en el flanco de bajada del reloj serial, SPI_SCK.
Entrada serial de datos SPI_MOSI, se usa para recibir datos de forma serial en el dispositivo. Recibe instrucciones y los datos para la programación. Los valores son validados en el flanco de subida del PSI_SCK.
Reloj serial SPIS_SCK, esta señal proporcional la medida del tiempo de la interfaz serial. Instrucciones, direcciones o datos presentes en la entrada de serial SPI_MOSI son validados en el flanco positivo del reloj serial. Los datos en son puestos en la Salida serial de datos en el flanco de bajada del SPI_SCK.
Chip Select SPI_SS_B, cuando esta señal esta en alto, el dispositivo es deseleccionado y la salida serial de datos es una alta impedancia. A menos que un programa interno, el borrado o ciclo de escritura de registros de estado este en proceso, el dispositivo estará en modo Standby. Después de encendido, un flanco de bajada en Chip Select es necesario antes de iniciar cualquier instrucción.

Conexiones Compartidas

Después de la configuración la SPI Serial Flash puede ser usada por la aplicación. En la S3ESK el bus SPI es compartido con otros dispositivos con periféricos SPI. Para acceder a la memoria SPI Serial Flash después de la configuración la FPGA debe deshabilitar los otros dispositivos con los que se comparte el bus SPI, así:

- DAC_CS en alto, deshabilita el convertidor Digital-Analógico
- AMP_CS en alto, deshabilita el preamplificador de ganancia programable
- AD_CONV en bajo, deshabilita el convertidor Analógico-Digital
- SF_CE0 en alto, deshabilita la memoria StrataFlash Parallel Flash PROM
- FPGA_INIT_B en alto, deshabilita la memoria Platform Flash PROM

UCF Localización

```
NET "SPI_MISO" LOC = "N10"    | IOSTANDARD = LVCMOS33 ;
NET "SPI_MOSI" LOC = "T4"     | IOSTANDARD = LVCMOS33       | SLEW = SLOW  | DRIVE = 6 ;
NET "SPI_SCK"  LOC = "U16"    | IOSTANDARD = LVCMOS33       | SLEW = SLOW  | DRIVE = 6 ;
NET "SPI_SS_B" LOC = "U3"     | IOSTANDARD = LVCMOS33       | SLEW = SLOW  | DRIVE = 6 ;
NET "SPI_ALT_CS_JP11" LOC = "R12" | IOSTANDARD = LVCMOS33   | SLEW = SLOW  | DRIVE = 6 ;
```

64 MByte (512 Mbit) DDR (Double Data Rate) SDRAM

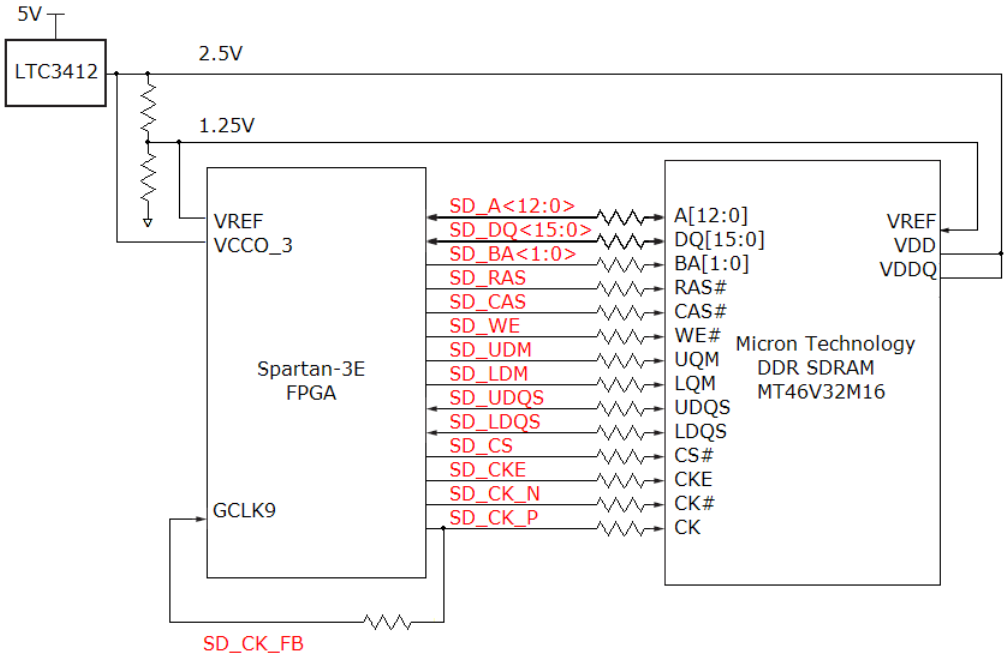


Figura 8: Conexiones entre la SDRAM y la FPGA

La S3ESK incluye una memoria DDR SDRAM, con una interfaz de datos de 16bit. Esta se encuentra conectada a los pines del banco 3 de E/S de la FPGA. El banco 3 y la DDR SDRAM están conectados a alimentación de 2.5V, generados por un regulador LTC3412 desde la alimentación de 5V de la board. Un divisor de voltaje es el encargado de establecer 1.25V de referencia común para la FPGA y la DDR SDRAM.

Descripción Funcional [7]

La DDR SDRAM utiliza una arquitectura de doble velocidad de transferencia de datos para lograr una alta velocidad de operación. La Arquitectura de doble velocidad de transferencia de datos es esencialmente una arquitectura 2n-prefetch[4] con una interfaz diseñada para la transferencia de dos palabras por ciclo de reloj en los pines de E/S. Una sola lectura o acceso de escritura para la DDR SDRAM consta de un solo 2n-bit de ancho; un ciclo de reloj de transferencia de datos en el interior de DRAM central y los dos n-bits de ancho correspondientes, un medio ciclo de reloj transfiere los datos a los pines de E/S.

[4] Arquitectura Prefetch, técnicas usadas para incrementar la velocidad de una RAM, mas de una palabra es descarga de la memoria en cada ciclo de dirección y transferencia para el selector de datos en el buffer de salida [2].

Un dato strobe bidireccional (DQS) se transmite al exterior, junto con los datos, para su uso en la captura de datos en el receptor, DQS es un strobe transmitido de la DDR SDRAM durante la lectura y por el controlador de la memoria durante la escritura. DQS es un flanco alineado con los datos de lectura y un flanco centrado con los datos durante la escritura.

La DDR SDRAM funciona con un reloj diferencial (CK y CK#); el cruce de CK hacia un alto y de CK# hacia un bajo se denomina flanco positivo de CK. Los comandos (direcciones y señales de control) son registrados en cada flanco positivo de CK. Los datos de entrada se registran en ambos flancos de DQS, y los datos de salida son referenciados para ambos flancos de DQS, así como para ambos flancos de CK.

Leer y escribir accesos a la DDR SDRAM son ráfagas orientadas; los accesos inician desde una posición seleccionada y continúa por un número programado de posiciones en una secuencia programada. Inicia accediendo con el registro de un comando activo, la cual podrá ser seguida de una lectura o escritura. Los bits de dirección registrados coinciden con el comando activo son usados para seleccionar el banco y la fila para ser accedidas. Los bits de registrados coinciden con la lectura o escritura son usados para seleccionar el banco e inicio de columna para el acceso ráfaga

La DDR SDRAM permite la programación de lectura o escritura por ráfaga de longitudes de dos, cuatro u ocho posiciones. La función AUTO PRECHARGE puede ser activada para poder proporcionar un tiempo libre fila de precargado que se inicia al final de la ráfaga de acceso.

Como ocurre con el estándar DEG SDRAMs, el segmentado, la arquitectura multibanco de la DDR SDRAMs permite la operación simultánea, ofreciendo así un alto ancho de banda efectivo para el precargado oculto de la fila y tiempo de activación.

Tabla 5: DDR (Double Data Rate) SDRAM

Categoría	PIN DDR SDRAM	Función
Dirección	A<0:12>	Entradas de dirección, proporciona la dirección de fila para el comando ACTIVE, y la dirección de columna y el bit de auto cargado (A10) para comandos de READ/WRITE, para seleccionar una posición fuera del arreglo de memoria en el respectivo banco. A10 muestreada durante le comando PRECHARGE determina si la pre-carga se aplica a un banco (A10 en bajo, bancos seleccionados por BA<0:1>) o todos los bancos (A10 en alto). Las entradas de dirección también proporcionan el código de operación durante el comando LOAD MODE REGISTER
Control	BA<0:1>	Dirección de banco, define a cual banco el comando ACTIVE, READ, WRITE o PRECHARGE se está aplicando. También define el modo de registro.
Control	CK, CK#	Reloj, CK, CK# son entradas de reloj diferenciales. Todas las señales control y dirección de entrada son tomadas en el cruce de el flanco positivo de CK y el flanco negativo de CK#. Los datos de salida (DQ y DQS) son referenciados por el cruce de CK y CK#
Control	CKE	Habilitar Reloj, CKE alto activa y CKE bajo desactiva el reloj interno, buffers de entrada y manejadores salidas.
Control	CS#	Chip select, Cs habilita y deshabilita el decodificador de comandos. Todos los comandos son enmascarados cuando CS# esta en alto. Provee selección de bancos externos en sistemas de múltiples bancos.
Control	DM, LDM, UDM	DM es una mascara de señal de entrada para la escritura de datos, la entrada es enmascarada cuando DM esta en alto junto con los datos de entrada durante el acceso.
Control	RAS#, CAS#, WE#	Entrada de comandos, define los comandos que se entra.
Datos	DQ<0:15>	Entrada/Salida de Datos, Bus de datos
Control	DQS, LDQS, UDQS	Dato strobe, salida cuando lee datos, entrada cuando escribe datos. DQS es un flanco alineado con los datos leídos, centrado en datos escritos. Este es usado para capturar los datos. Para x16 LDQS es DQS para DQ<0:7> y UDQS es DQS para DQ<8:15>

UCF Localización

Dirección

```
NET "SD_A<12>"      LOC = "P2"    | IOSTANDARD = SSTL2_I ;
NET "SD_A<11>"      LOC = "N5"    | IOSTANDARD = SSTL2_I ;
NET "SD_A<10>"      LOC = "T2"    | IOSTANDARD = SSTL2_I ;
NET "SD_A<9>"       LOC = "N4"    | IOSTANDARD = SSTL2_I ;
NET "SD_A<8>"       LOC = "H2"    | IOSTANDARD = SSTL2_I ;
```

```
NET "SD_A<7>"       LOC = "H1"    | IOSTANDARD = SSTL2_I ;
NET "SD_A<6>"       LOC = "H3"    | IOSTANDARD = SSTL2_I ;
NET "SD_A<5>"       LOC = "H4"    | IOSTANDARD = SSTL2_I ;
NET "SD_A<4>"       LOC = "F4"    | IOSTANDARD = SSTL2_I ;
NET "SD_A<3>"       LOC = "P1"    | IOSTANDARD = SSTL2_I ;
NET "SD_A<2>"       LOC = "R2"    | IOSTANDARD = SSTL2_I ;
NET "SD_A<1>"       LOC = "R3"    | IOSTANDARD = SSTL2_I ;
NET "SD_A<0>"       LOC = "T1"    | IOSTANDARD = SSTL2_I ;
```

Datos

```
NET "SD_DQ<15>"     LOC = "H5"    | IOSTANDARD = SSTL2_I ;
NET "SD_DQ<14>"     LOC = "H6"    | IOSTANDARD = SSTL2_I ;
NET "SD_DQ<13>"     LOC = "G5"    | IOSTANDARD = SSTL2_I ;
NET "SD_DQ<12>"     LOC = "G6"    | IOSTANDARD = SSTL2_I ;
NET "SD_DQ<11>"     LOC = "F2"    | IOSTANDARD = SSTL2_I ;
NET "SD_DQ<10>"     LOC = "F1"    | IOSTANDARD = SSTL2_I ;
NET "SD_DQ<9>"      LOC = "E1"    | IOSTANDARD = SSTL2_I ;
NET "SD_DQ<8>"      LOC = "E2"    | IOSTANDARD = SSTL2_I ;
NET "SD_DQ<7>"      LOC = "M6"    | IOSTANDARD = SSTL2_I ;
NET "SD_DQ<6>"      LOC = "M5"    | IOSTANDARD = SSTL2_I ;
NET "SD_DQ<5>"      LOC = "M4"    | IOSTANDARD = SSTL2_I ;
NET "SD_DQ<4>"      LOC = "M3"    | IOSTANDARD = SSTL2_I ;
NET "SD_DQ<3>"      LOC = "L4"    | IOSTANDARD = SSTL2_I ;
NET "SD_DQ<2>"      LOC = "L3"    | IOSTANDARD = SSTL2_I ;
NET "SD_DQ<1>"      LOC = "L1"    | IOSTANDARD = SSTL2_I ;
NET "SD_DQ<0>"      LOC = "L2"    | IOSTANDARD = SSTL2_I ;
```

Control

```
NET "SD_BA<0>"      LOC = "K5"    | IOSTANDARD = SSTL2_I ;
NET "SD_BA<1>"      LOC = "K6"    | IOSTANDARD = SSTL2_I ;
NET "SD_CAS"        LOC = "C2"    | IOSTANDARD = SSTL2_I ;
NET "SD_CK_N"       LOC = "J4"    | IOSTANDARD = SSTL2_I ;
NET "SD_CK_P"       LOC = "J5"    | IOSTANDARD = SSTL2_I ;
NET "SD_CKE”        LOC = "K3"    | IOSTANDARD = SSTL2_I ;
NET "SD_CS"         LOC = "K4"    | IOSTANDARD = SSTL2_I ;
NET "SD_LDM"        LOC = "J2"    | IOSTANDARD = SSTL2_I ;
NET "SD_LDQS"       LOC = "L6"    | IOSTANDARD = SSTL2_I ;
NET "SD_RAS"        LOC = "C1"    | IOSTANDARD = SSTL2_I ;
NET "SD_UDM"        LOC = "J1"    | IOSTANDARD = SSTL2_I ;
NET "SD_UDQS"       LOC = "G3"    | IOSTANDARD = SSTL2_I ;
NET "SD_WE"         LOC = "D1"    | IOSTANDARD = SSTL2_I ;
NET "SD_CK_FB"      LOC = "B9"    | IOSTANDARD = LVCMOS33 ;
```

DS2432 1-Wire SHA-1 EEPROM

La S3ESK incluye una Maxim D2432 serial EEPROM con SHA-1 Engine. La DS2432 EEPROM usa una interfaz Maxim 1-Wire, es decir, una interfaz de comunicación y alimentación en un solo hilo. La DS2432 EEPROM ofrece un de muchas posibles formas para proteger copias del bitstream de configuración de la FPGA, haciendo difícil la clonación. La nota de aplicación XAPP780, FPGA IFF Copy Protection Using Dallas Semiconductor/Maxim DS2432 Secure EEPROMs, ofrece un posible método de implementación [14].

DS2432 combina 1024 bits de EEPROM, 64 bit secretos, un registro de control de 8-bytes y hasta cinco registro de lectura/escritura para el usuario, un motor SHA-1 de 512 bits, y una completa interfaz 1-Wire. Cada DS2432 tiene su propio número de registro 64-bit ROM que es impreso en fábrica en el chip para proporcionar una garantía unida de identidad para absoluta trazabilidad. Los datos se transmiten serialmente vía protocolo 1-Wire, que requiere solo un dato dirigido y retornado a tierra. La DS4232 tiene adicionalmente un área de memoria llamada Scratchpad[5] que actúa como un buffer cuando se escribe en la memoria principal, la pagina de registros o cuando se instala un nuevo secreto. Los datos son primero escritos en el Scratchpad desde donde se puede volver a leer. Después que los datos son verificados, un comando copia del Scratchpad y transfiere los datos a la posición final en la memoria, permitiendo que DS2432 reciba un 160-bit MAC[6]. El cómputo del MAC involucra datos secretos y adicionales almacenados en incluyendo el número de registro del dispositivo. Solo un nuevo secreto puede ser cargado sin proporcionar un MAC. El motor SHA-1 también puede ser activado para computar los 160-bit del MAC cuando lee una página de memoria o para calcular un nuevo secreto, en lugar de cárgalo. Las aplicaciones del SD2432 incluyen seguridad de propiedad intelectual, después de la salida del mercado del manejo de los consumidores, y la falsificación de datos de soporte [6].

UCF Localización

```
NET "DS_WIRE" LOC = "U4"    IOSTANDARD = LVCMOS33 |   LEW = SLOW | RIVE = 8 ;
```

[5] Scratchpad, Bloc de notas o apuntes, una memoria interna de alta velocidad usada para almacenar temporalmente información preliminar [2].
[6] MAC, Message Authentication Codes.

CONVERSIÓN ANALÓGICA-DIGITAL

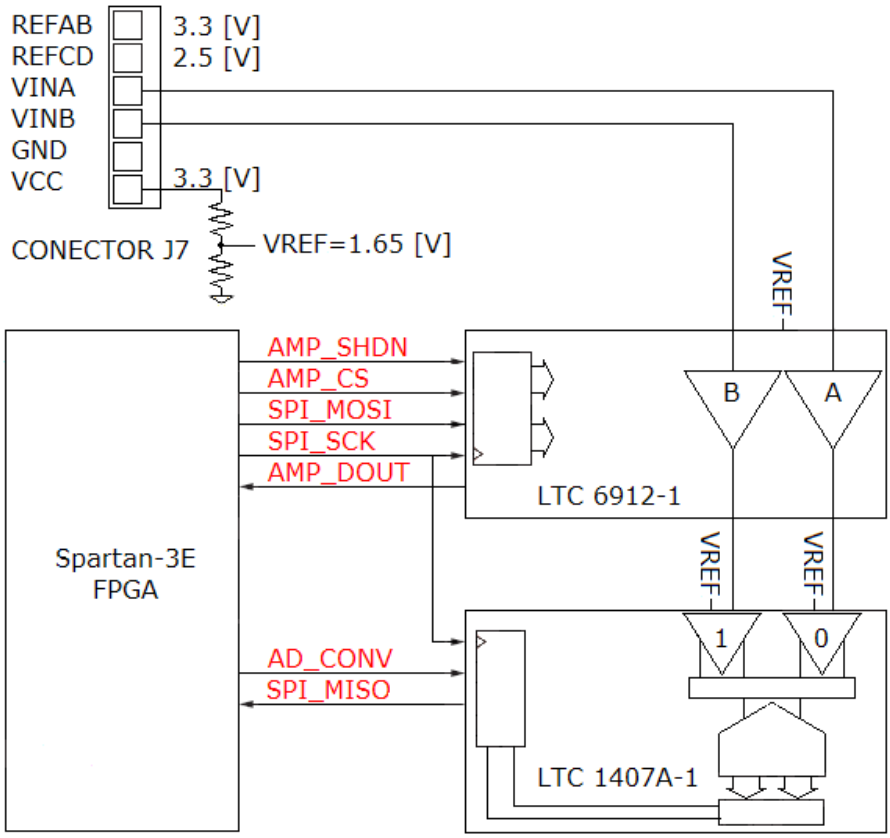


Figura 9: Conexiones de la FPGA y el modulo de conversión analógica-digital

La S3ESK dispone de un modulo para la conversión analógica-digital, compuesto por un preamplificador de ganancia programable y un conversor analógico-digital, que son programados y controlados desde la FPGA usando una interfaz de comunicación serial.

Una ganancia G, es establecida mediante la configuración del preamplificador. Por medio de un divisor de tensión, 1.65 [V] se aplica en las entradas inversoras de cada canal del conversor, y en las entradas no inversoras de los preamplificadores, de esta forma se establece el voltaje de referencia de 1.65 [V]. Mediante En esta configuración se tiene obtiene un rango de entrada de ± 1.25 [V] alrededor de 1.65 [V].
A la salida del ADC se presenta una salida digital de 14 bit que representan las muestras tomadas en la entrada del canal. Entonces el LSB que entrega el ADC es 2.5V/16384=153 μV.

El voltaje en VINA y VINB muestreado es convertido a una representación digital de 14 bits como lo indica la función:

$$D[13:0] = \frac{G \times (V_{IN} - 1.65V)}{153\mu V}$$

Dual Programmable Gain Amplifiers

El LTC6912 es una familia de amplificadores de ganancia digitalmente programable de dos canales, bajo ruido, que son de fácil uso y ocupa muy poco espacio en board. La ganancia para ambos canales son independientemente programables usando una interfaz SPI para seleccionar ganancia de voltaje de 0, 1, 2, 10, 20, 50 y 100 V/V. Todas las ganancias son inversas.

El control de ganancia dentro del amplificador ocurre por conmutación del conjunto de resistencias combinadas dentro o fuera de un lazo cerrado de un amplificador operacional usando conmutación análoga MOS[7]. El ancho de banda de cada amplificador depende de la ganancia que le sea configurada.

Descripción de la Interfaz SPI

El control de ganancia para cada amplificador es independientemente programable usando la interfaz SPI. Los niveles lógicos para la interfaz SPI del LTC6912-1 son compatibles con TTL y CMOS. Cuando AMP_CS es puesto a un nivel bajo, los datos seriales en SPI_MISO van siendo ingresados a un registro de desplazamiento de 8-bits, en el flanco derecho de SPI_SCK, con transmisión del MSB[8] primero. El MSB del contenido del registro es puesto en AMP_DOUT en cada flanco de bajada de SPI_SCK.

Los cuatro bits más significativos configuran la ganancia del amplificador del canal B, y los cuatros bits menos significativos configuran la ganancia del amplificador del canal A. En la Tabla 7 se detalla la ganancia nominal y los respectivos códigos de cada ganancia.

Se debe asegurar que SPI_SCK se encuentre en un bajo antes que AMP_CS sea puesto en un bajo, para evitar un pulso extra del reloj interno a la entrada del registro de desplazamiento de 8-bits.

AMP_DOUT siempre se encuentra activo en todos los estados, por lo tanto no puede ser conectado a otras salidas SPI.

Un LTC6912 puede estar en Daisy-Chain[9] con otro LTC6912 u otros dispositivos usando interfaz serial para conectar Dout al Din del siguiente circuito integrado y AMP_CS permanece común a todos los circuitos integrados en el Daysi-Chain.

La interfaz de señales entre la FPGA y el amplificador es compartida con otros dispositivos por el bus SPI.

---

[7] MOS Metal-Oxide Semiconductor, Semiconductor de Óxido Metálico.
[8] MSB Most Significant Bit, el bit más significativo.
[9] Daisy Chain, es un esquema donde un dispositivo A es conectado a un dispositivo B, y un dispositivo B a un dispositivo C, etc. Las Conexiones solo pueden ser entre dispositivos que se preceden, y no pueden formase lazos. Son usadas principalmente para transmitir datos digitales.

Control de la Interfaz SPI

Para configurar la ganancia en cada amplificador es enviado un comando de un byte, que consiste de dos campos de 4-bits. El bit más significativo, B3, es enviado primero.
La señal AMP_DOUT transmite la ganancia configurada anteriormente. Estos valores pueden ser ignorados para la mayoría de aplicaciones.
El intercambio del bus SPI inicia cuando la FPGA pone en un bajo la señal AMP_CS. El amplificador captura los datos en SPI_MOSI en el flanco de subida de la señal de reloj SPI_SCK. Los datos serial son presentados en AMP_DOUT, en el flanco de bajada de SPI_SCK.

Tabla 6: Características de tiempo para la interfaz SPI del LTC6912-1

SÍMBOLOS	PARAMETROS	TIEMPO [ns]	
		Mínimo	Máximo
t1	SPI_MOSI válido para SPI_SCK	30	
t2	SPI_MOSI válido para SPI_SCK	0	
t3	SPI_SCK alto	50	
t4	SPI_SCK bajo	50	
t5	Ancho del pulso AMP_CS	40	
t6	LSB SPI_SCK a AMP_CS	40	
t7	LD bajo a SPI_SCK	20	
t8	Retardo de salida en AMP_DOUT		85
t9	SPI_SCK bajo a AMP_CS bajo	0	

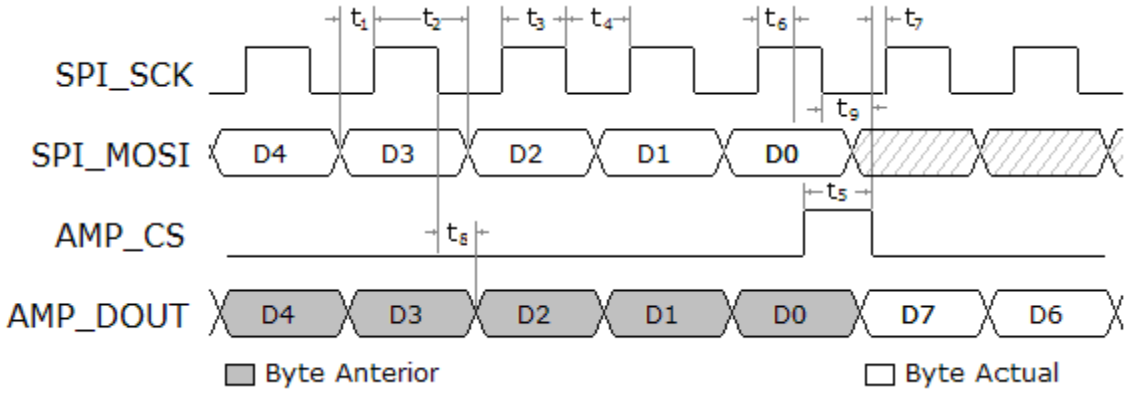


Figura 10: Diagrama de tiempos para la interfaz SPI del LTC6912-1

Ganancia Programable

Las señales presentes en VINA o VINB en el conector J7 son amplificadas respecto a 1.65V.

Tabla 7: Configuración para la Ganancia Programable del Preamplificador

A3	A2	A1	A0	GANANCIA NOMINAL (G)		RANGO VOLTAJE DE ENTRADA	
B3	B2	B1	B0	V/V	dB	B3	B2
0	0	0	0	0	-120		
0	0	0	1	-1	0	0.4	2.9
0	0	1	0	-2	6	1.025	2.275
0	0	1	1	-5	14	1.4	1.9
0	1	0	0	-10	20	1.525	1.775
0	1	0	1	-20	26	1.5875	1.7125
0	1	1	0	-50	34	1.625	1.575
0	1	1	1	-100	40	1.6375	1.6625
1	0	X	X	0	-120		

UCF Localización

```
NET "SPI_MOSI" LOC = "T4"     | IOSTANDARD = LVCMOS33       | SLEW = SLOW  | DRIVE = 6 ;
NET "AMP_CS"   LOC = "N7"     | IOSTANDARD = LVCMOS33       | SLEW = SLOW  | DRIVE = 6 ;
NET "SPI_SCK"  LOC = "U16"    | IOSTANDARD = LVCMOS33       | SLEW = SLOW  | DRIVE = 8 ;
NET "AMP_SHDN" LOC = "P7"     | IOSTANDARD = LVCMOS33       | SLEW = SLOW  | DRIVE = 6 ;
NET "AMP_DOUT" LOC = "E18"    | IOSTANDARD = LVCMOS33 ;
```

Linear Tech LTC1407A-1 Dual A/D

El LTC1407A tiene dos entradas diferenciales separadas que son muestreadas simultáneamente en el flanco de subida de la señal AD_CONV. Estas dos muestras tomadas son convertidas en una velocidad de 1.5 Msps[10] por canal. El dispositivo convierte 0V a 2.5V entradas unipolares diferenciales. La interfaz serial envía a la salida

[10] Msps, Megasamples per second. Mega muestras por segundo.

los dos resultados de las conversiones en 32 ciclos compatibles con el estándar de interfaz serial SPI.

Descripción de la Interfaz SPI

El LTC1407A dispone de una interfaz SPI para la transmisión de las conversiones realizadas. Las entradas SPI_SCK, AD_CONV y la salida SPI_MISO, implementan esta interfaz.

La conversión se inicia al en un flanco de subida de AD_CONV, cualquier otro flanco de subida en AD_CONV después de iniciar la conversión es ignorado hasta que no ocurran 32 ciclos de SPI_SCK.

Un flanco de subida en SPI_SCK inicia el proceso de conversión y también la actualización de cada bit que se transmite en SPI_MISO. Después de iniciar la conversión, poner un alto en AD_CONV, en el tercer ciclo de SPI_SCK se inicia el envío del las dos tramas de datos de 14 bits, enviando los MSB primero. Los datos de las dos conversiones serán recibidos en 32 o mas ciclos de reloj por cada muestra.

Al conectar la fuente de alimentación del LTC1407A, la salida SPI_MISO es puesta inmediatamente en alta impedancia. SPI_MISO se mantiene en alta impedancia hasta que se inicia una nueva conversión. Las dos tramas de transmitidas son separadas por alta impedancia durante dos ciclos de reloj.
Se deben usar las especificaciones del SPI_SCK para tener un SPI_MISO válido. De tal forma se garantiza siempre que SPI_MISO para ser válido por el siguiente flanco de subida de SPI_SCK. Los datos de salida de 32 bits son compatibles con el puerto serial de 16 bits o 32 bits de la mayoría de procesadores.

Control de la Interfaz SPI

El convertidor simultáneamente toma la muestra de los dos canales e inicia su conversión. La conversión es realizada y enviada con una latencia de una muestra, es decir el resultado de la conversión se transmite hasta la siguiente vez que se inicie una nueva conversión. El resultado enviado es la representación digital de la muestra análoga como un número binario de 14 bits en complemento a dos.

Tabla 8: Características de tiempo para la interfaz SPI del LTC1407A-1

SÍMBOLOS	PARAMETROS	Mínimo	Máximo	Unidades
fSAMPLE	Frecuencia máxima de muestreo por canal	1.5		MHz
tTHROUGHPUT	Mínimo periodo de muestreo (Adquisición+ Conversión)		667	ns
tSCK	Periodo del Reloj	19.6	10000	Ns
tCONV	Tiempo de conversión	32		Ciclos de SCK
t1	Mínimo ancho del pulso Positivo o Negativo de SCK	2		ns
t2	CONV a SCK	3	10000	ns
t3	SCK antes de CONV	0		ns
t4	Mínimo ancho del pulso Positivo o Negativo de CONV	4		ns
t5	SCK a el modo Sample	4		ns
t6	CONV a el modo Hold	1.2		ns
t7	Intervalo del 32avo SCK a CONV	45		ns
t8	Mínimo retardo desde SCK a Bits 0 valido	8		ns
t9	SCK a Hi-Z en SDO	6		ns
t10	El bit anterior en SDO permanece válido después del flanco de subida SCK	2		ns

Conexiones Compartidas

El bus SPI esta compartido por otros dispositivos, por tal razón, es importante que se desactiven otros dispositivos cuando la FPGA se comunica con el amplificador o el conversor para evitar conflictos en el bus. A pesar de que la PROM StrataFlash es un dispositivo paralelo, el bit de datos menos significativo se comparte con la señal SPI_MISO. La Plataforma Flash PROM sólo esta potencialmente activa si la FPGA se configura para el modo de configuración Master Serie

Los otros dispositivos que comparten el bus SPI pueden ser desactivados así:

- SPI_SS_B en alto, deshabilita la memoria SPI Serial Flash
- AMP_CS en alto, deshabilita el preamplificador de ganancia programable
- DAC_CS en alto, deshabilita el convertidor Digital-Analógico
- SF_CE0 en alto, deshabilita la memoria StrataFlash Parallel Flash PROM
- FPGA_INIT_B en alto, deshabilita la memoria Platform Flash PROM

UCF Localización

```
NET "AD_CONV"  LOC = "P11"    | IOSTANDARD = LVCMOS33       | SLEW = SLOW  | DRIVE = 6 ;
NET "SPI_SCK"  LOC = "U16"    | IOSTANDARD = LVCMOS33       | SLEW = SLOW  | DRIVE = 8 ;
NET "SPI_MISO" LOC = "N10"    | IOSTANDARD = LVCMOS33 ;
```

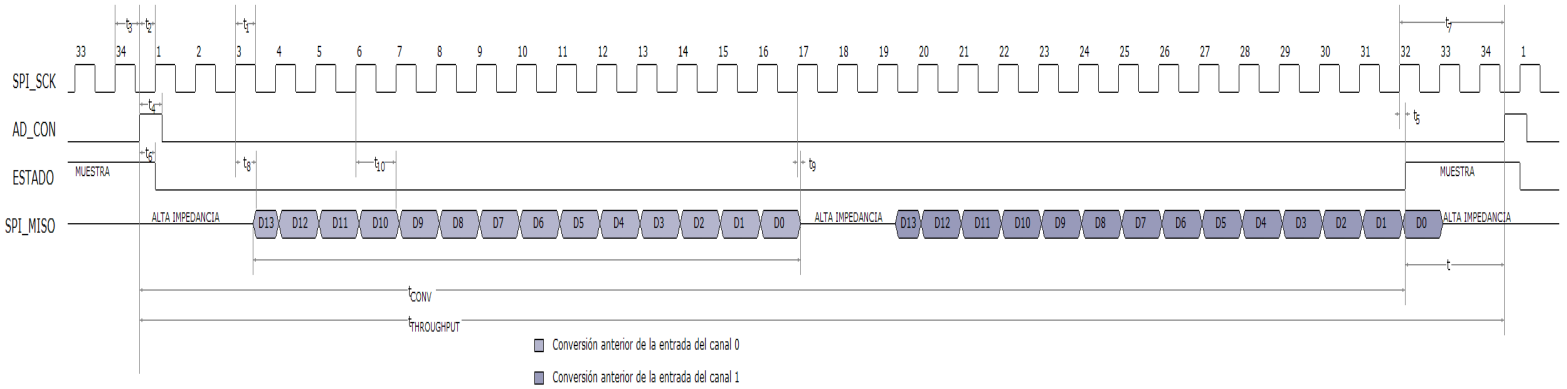


Figura 11: Diagrama de tiempos Linear Tech LTC1407A-1

CONVERSIÓN DIGITAL-ANALÓGICA

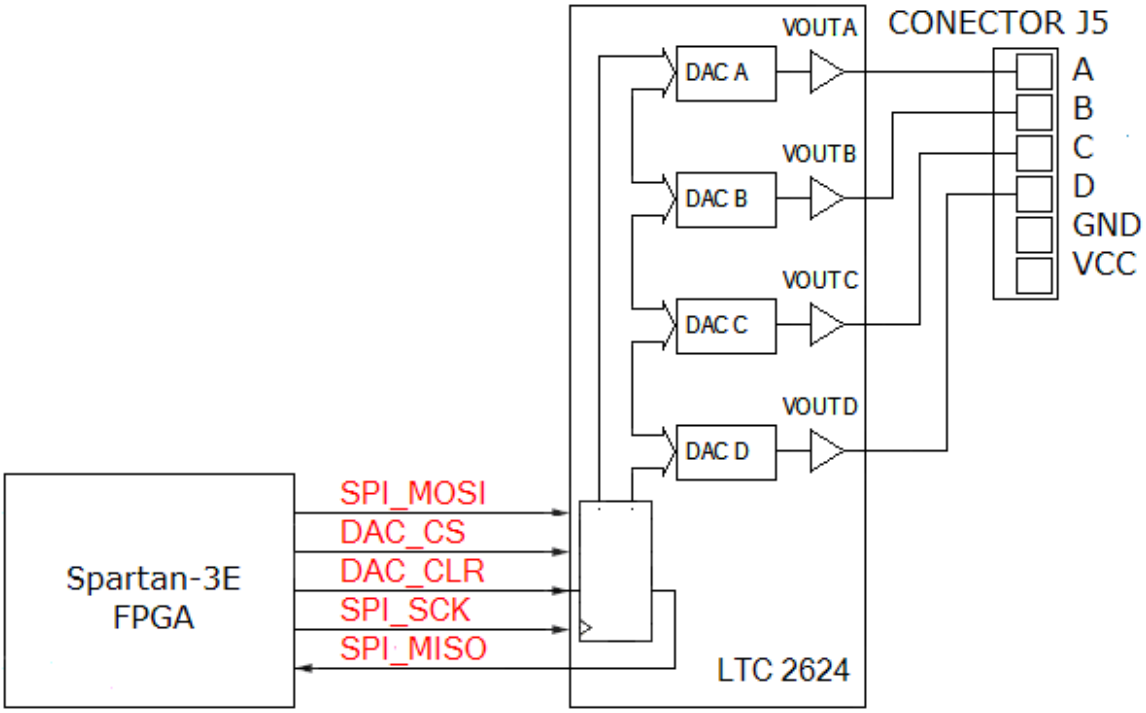


Figura 12: Conexiones entre el conversor digital-analógico y la FPGA

La S3ESK dispone de un integrado para realizar la conversión digital-análoga, el LTC 2624. El LTC2624 son cuatro DAC de 12bits, con referencias separadas para cada DAC, pero que en la S3ESK han sido agrupadas. Incorpora una interfaz serial SPI que puede operar hasta a 50 MHz.

La función de transferencia digital a análoga esta dada por:

$$Vout_{(IDEAL)} = \left(\frac{k}{2^N}\right)\left(REF_X - REFLO\right) + REFLO$$

Donde k es el decimal equivalente del código binario a la entrada del DAC, N es la resolución, es decir 4096, y REFx es el voltaje de referencia. REFLO es la tensión establecida para el cero en la escala de voltaje de todos los DACs. En la S3ESK ha sido conectada a GND.

La tensión REF para las salidas A y B es de 3.3 V ±5%, para C y D es de 2.5 V ±5%.

Descripción de la Interfaz SPI

DAC_CS actúa como una entrada de selección del dispositivo, activando la alimentación, habilitando la entrada de reloj, el buffer de salida y habilitando la entrada del registro de desplazamiento. Los datos se reciben mientras DAC_CS permanece en bajo en los siguientes 24 flancos de subida del reloj, SPI_SCK. La entrada esta definida como cuatro bits de comando, cuatro bits de dirección y el código de 16 bits, ordenando de MSB a LSB los 12 bits del dato y seguido por 4 bits para completar. El flanco de subida en DAC_CS termina la recepción de datos, e indica al dispositivo cargar la acción indicada en los 24 bits de entrada.

Comandos y Direcciones

Tabla 9: Códigos de los Comandos del LTC2624

COMANDOS				FUNCIÓN
C3	C2	C1	C0	
0	0	0	0	Escribe en el registro de entrada n
0	0	0	1	Actualiza el registro DAC n
0	0	1	0	Escribe en el registro de entrada n, Actualiza todos los registros DAC
0	0	1	1	Escribe el registro de entrada n y actualiza el registro DAC n
0	1	0	0	Apagado n
1	1	1	1	No opera

Tabla 10: Códigos de las Direcciones del LT2624

DIRRECIONES				ASIGNA
A3	A2	A1	A0	
0	0	0	0	DAC A
0	0	0	1	DAC B
0	0	1	0	DAC C
0	0	1	1	DAC D
1	1	1	1	Todos los DAC

Los primeros cuatro comandos son de escritura y actualización. Una operación de escritura carga un dato de 16 bits desde el registro de desplazamiento de 32 bits al registro DAC seleccionado.
Una operación de actualización copia los datos desde el registro de entrada al registro del DAC. Una vez copiado en el registro, el dato se convierte en el código activo de entrada, y se convierte en un voltaje analógico a la salida del DAC.

El ancho mínimo de entrada es 24 bits, sin embargo, puede ampliarse a 32 bits. Con un ancho de 32 bits, se agregan 8 bits al inicio de la transferencia, seguido por los 24 bits del dato como se describió. Un ancho de 32-bits es necesario para una operación en Daisy-Chain, y también para conexión serial con la gran cantidad de microprocesadores.

Control de la Interfaz SPI

Tabla 11: Características de tiempo para la interfaz SPI del LTC2624

SÍMBOLOS	PARAMETROS	TIEMPO [ns]	
		Mínimo	Máximo
t1	SPI_MOSI válido para SPI_SCK	4	
t2	SPI_MOSI válido para SPI_SCK	4	
t3	SPI_SCK alto	9	
t4	SPI_SCK bajo	9	
t5	Ancho del pulso DAC_CS	10	
t6	LSB SPI_SCK alto a ADC_CS alto	7	
t7	ADC_CS bajo a SPI_SCK alto	7	
t8	Retardo de salida en SPI_MOSI		20
t9	Ancho del pulso de ADC_CLR	20	
t10	DAC_CS alto para un flanco positivo de PSI_SCK	7	

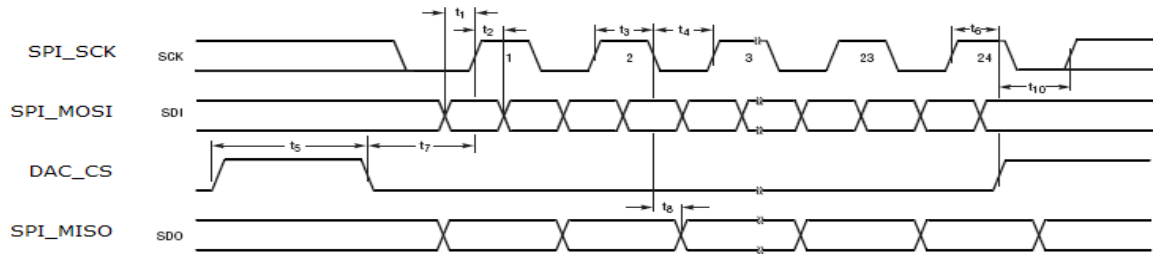


Figura 13: Diagrama de tiempos para la interfaz de comunicación SPI del LTC2624

Conexiones Compartidas

El bus SPI esta compartido por otros dispositivos, por tal razón, es importante que se desactiven otros dispositivos cuando la FPGA se comunica con el amplificador o el conversor para evitar conflictos en el bus. A pesar que la PROM StrataFlash es un dispositivo paralelo, el bit de datos menos significativo se comparte con la señal SPI_MISO. La Plataforma Flash PROM sólo esta potencialmente activa si la FPGA se configura para el modo de configuración Master Serie

Los otros dispositivos que comparten el bus SPI pueden ser desactivados así:

- SPI_SS_B en alto, deshabilita la memoria SPI Serial Flash
- AMP_CS en alto, deshabilita el preamplificador de ganancia programable
- AD_CONV en bajo, deshabilita el convertidor Analógico-Digital
- SF_CE0 en alto, deshabilita la memoria StrataFlash Parallel Flash PROM
- FPGA_INIT_B en alto, deshabilita la memoria Platform Flash PROM

UCF Localización

```
NET "SPI_MISO" LOC = "N10"    | IOSTANDARD = LVCMOS33 ;
NET "SPI_MOSI" LOC = "T4"     | IOSTANDARD = LVCMOS33 |     SLEW = SLOW    | DRIVE = 8 ;
NET "SPI_SCK"  LOC = "U16"    | IOSTANDARD = LVCMOS33 |     SLEW = SLOW    | DRIVE = 8 ;
NET "DAC_CS"   LOC = "N8"     | IOSTANDARD = LVCMOS33 |     SLEW = SLOW    | DRIVE = 8 ;
NET "DAC_CLR"  LOC = "P8"     | IOSTANDARD = LVCMOS33 |     SLEW = SLOW    | DRIVE = 8 ;
```

CPLD

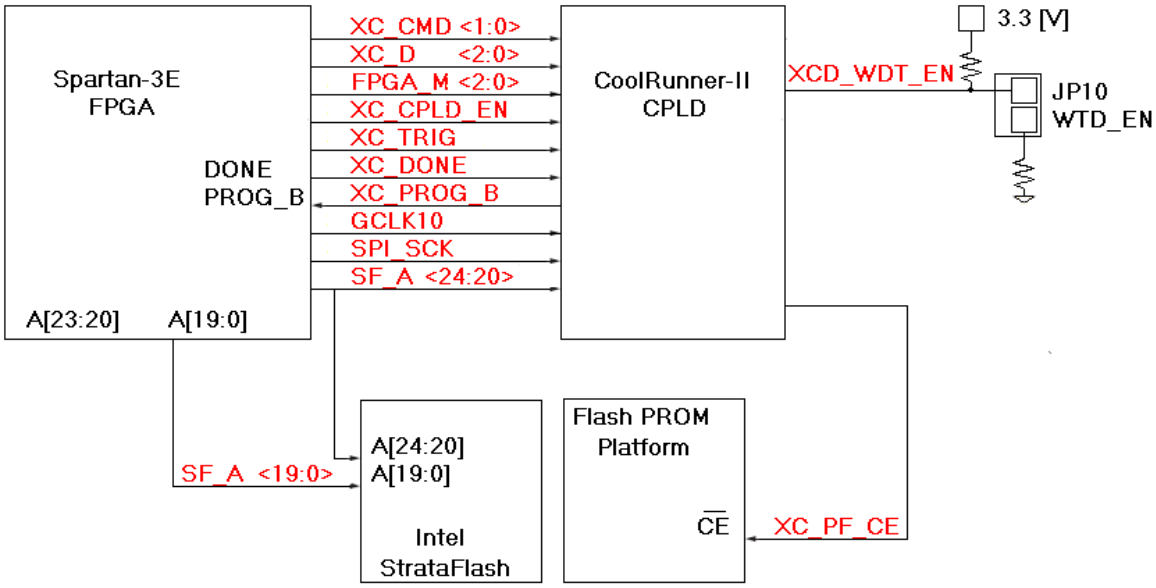


Figura 14: CPLD conexiones con la FPGA y gestión de la configuración BPI

La board del S3ESK incluye un CPLD de la familia CoolRunner-II, se encuentra disponible para su programación y uso por parte del diseñador, también es usado para coordinar el comportamiento de las memorias del S3ESK durante la configuración de la FPGA, esta operación ocupa una pequeña porción del CPLD.

El CPLD esta encargado de:

Cuando la FPGA se encuentra en el modo de configuración Master Serie (M <2:0> = 000), generar una señal activa baja para habilitarla XCF04S Platform Flash PROM. La Platform Flash PROM está desactivada en los otros modos de configuración. El CPLD ayuda a reducir el número de jumpers en la board y simplifica la interacción de todas las posibles memorias fuentes de configuración de la FPGA.

Cuando la FPGA es configurada en el modo BPI_UP (FPGA_M <2:0> = 010, done = 0), pone las cinco líneas superiores de dirección de la StrataFlash PROM, A [24:20], a 00000 binario. Cuando la FPGA es configurada en el modo BPI-Up (FPGA_M <2:0> = 011, done = 0), pone las cinco líneas superiores de dirección de la StrataFlash PROM, A [24:20], a 11111 binario. Fija las cincos líneas superiores de dirección en ZZZZZ en los otros modos de configuración no BPI o cuando el pin DONE de la FPGA esta en alto.

Luego de la función implementada en el CPLD, el usuario tiene a su disposición de 13 a 21 pines E/S y 58 macroceldas.
El Jumper JP10 (WDT_EN) define el estado de la señal WDT_EN del CPLD. De forma predeterminada, este jumper esta abierto y la señal se establece en lógica alta.

La salida XC_PROG_B del CPLD, si es usada, debe estar configurada como open-drain, es decir nunca en un alto. Esta señal esta conectada directamente al pin de programación PROG_B.
El bit más significativo de la dirección de la StrataFlash PROM, SF_A<24>, es la misma señal FX2_IO del conector FX2. La StrataFlash de 16 Mbytes físicamente solo usa 24 bits, SF_A<23:0>. La dirección SF_A<24> esta disponible para migrar a una StrataFlash PROM de mayor densidad.

UCF Localización, FPGA al CPLD

```
NET "XC_CMD<1>"         LOC = "N18"   | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "XC_CMD<0>"         LOC = "P18"   | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "XC_D<2>"           LOC = "F17"   | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "XC_D<1>"           LOC = "F18"   | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "XC_D<0>"           LOC = "G16"   | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "FPGA_M2"           LOC = "T10"   | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "FPGA_M1"           LOC = "V11"   | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "FPGA_M0"           LOC = "M10"   | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "XC_CPLD_EN"        LOC = "B10"   | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "XC_TRIG"           LOC = "R17"   | IOSTANDARD = LVCMOS33 ;
NET "XC_GCK0"           LOC = "H16"   | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "GCLK10"            LOC = "C9"    | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "SPI_SCK"           LOC = "U16"   | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "SF_A<24>"          LOC = "A11"   | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "SF_A<23>"          LOC = "N11"   | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "SF_A<22>"          LOC = "V12"   | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "SF_A<21>"          LOC = "V13"   | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "SF_A<20>"          LOC = "T12"   | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
```

UCF Localización CPLD

```
NET "XC_WDT_EN"         LOC = "P16"   | IOSTANDARD = LVCMOS33 ;
NET "XC_CMD<1>"         LOC = "P30"   | IOSTANDARD = LVCMOS33       | SLEW = SLOW ;
NET "XC_CMD<0>"         LOC = "P29"   | IOSTANDARD = LVCMOS33       | SLEW = SLOW ;
NET "XC_D<2>"           LOC = "P36"   | IOSTANDARD = LVCMOS33       | SLEW = SLOW ;
NET "XC_D<1>"           LOC = "P34"   | IOSTANDARD = LVCMOS33       | SLEW = SLOW ;
NET "XC_D<0>"           LOC = "P33"   | IOSTANDARD = LVCMOS33       | SLEW = SLOW ;
NET "FPGA_M2"           LOC = "P8"    | IOSTANDARD = LVCMOS33       | SLEW = SLOW ;
NET "FPGA_M1"           LOC = "P6"    | IOSTANDARD = LVCMOS33       | SLEW = SLOW ;
NET "FPGA_M0"           LOC = "P5"    | IOSTANDARD = LVCMOS33       | SLEW = SLOW ;
NET "XC_CPLD_EN"        LOC = "P42"   | IOSTANDARD = LVCMOS33       | SLEW = SLOW ;
NET "XC_TRIG"           LOC = "P41"   | IOSTANDARD = LVCMOS33       | SLEW = SLOW ;
NET "XC_DONE"           LOC = "P40"   | IOSTANDARD = LVCMOS33       | SLEW = SLOW ;
NET "XC_PROG_B"         LOC = "P39"   | IOSTANDARD = LVCMOS33       | SLEW = SLOW ;
NET "XC_GCK0"           LOC = "P43"   | IOSTANDARD = LVCMOS33       | SLEW = SLOW ;
NET "GCLK10"            LOC = "P1"    | IOSTANDARD = LVCMOS33       | SLEW = SLOW ;
NET "SPI_SCK"           LOC = "P44"   | IOSTANDARD = LVCMOS33       | SLEW = SLOW ;
NET "SF_A<24>"          LOC = "P23"   | IOSTANDARD = LVCMOS33       | SLEW = SLOW ;
NET "SF_A<23>"          LOC = "P22"   | IOSTANDARD = LVCMOS33       | SLEW = SLOW ;
NET "SF_A<22>"          LOC = "P21"   | IOSTANDARD = LVCMOS33       | SLEW = SLOW ;
NET "SF_A<21>"          LOC = "P20"   | IOSTANDARD = LVCMOS33       | SLEW = SLOW ;
NET "SF_A<20>"          LOC = "P19"   | IOSTANDARD = LVCMOS33       | SLEW = SLOW ;
NET "SPI_MISO"          LOC = "N10"   | IOSTANDARD = LVCMOS33 ;
NET "SPI_MOSI"          LOC = "T4"    | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 8 ;
NET "SPI_SCK"           LOC = "U16"   | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 8 ;
NET "DAC_CS"            LOC = "N8"    | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 8 ;
NET "DAC_CLR"           LOC = "P8"    | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 8 ;
```

ETHERNET PHYSICAL LAYER INTERFACE

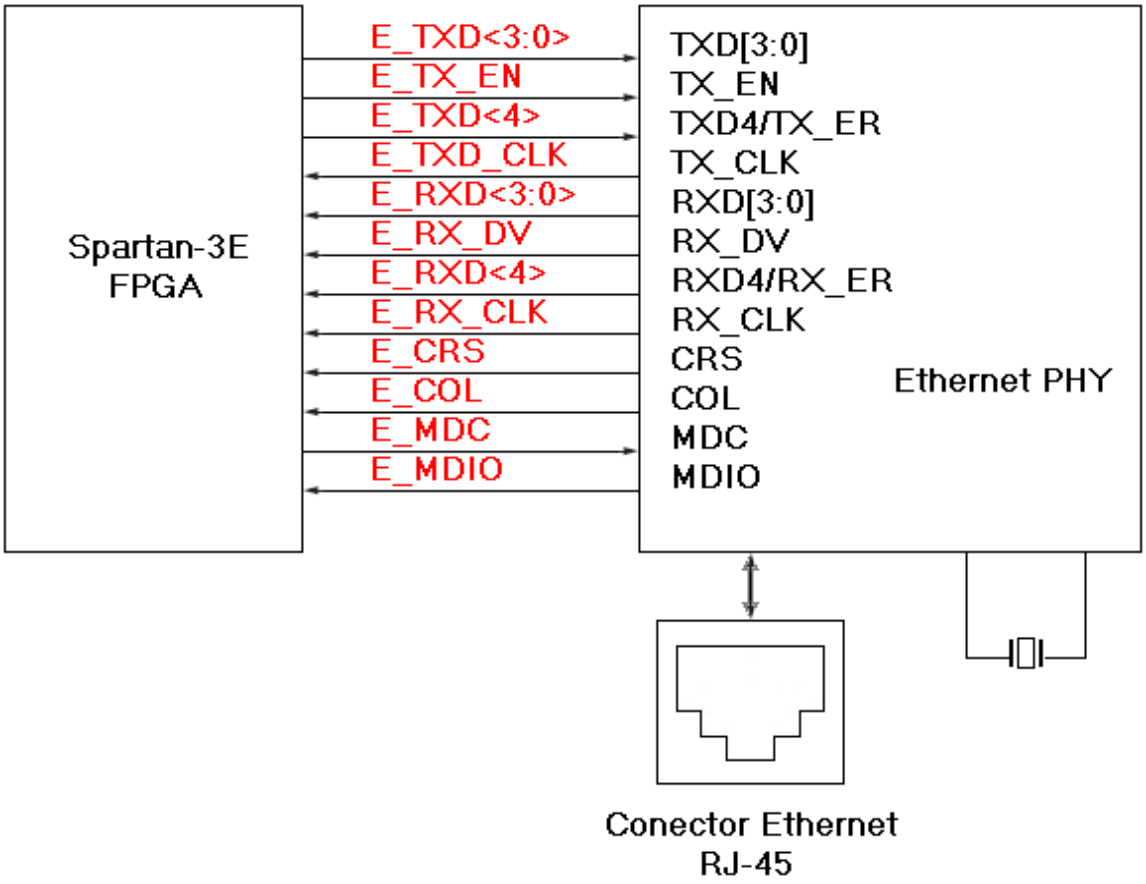


Figura 15: Interfaz de conexión entre el dispositivo EtherNET y la FPGA

La board del S3ESK dispone de un Standard Microsystems LAN83C185 10/100 EtherNET PHY y un conector RJ-45. Con un Control de acceso al medio EtherNET (MAC) implementado en la FPGA, la board puede tener la opción de conectarse a una red EtherNET estándar. Los tiempos son controlados por un oscilador de cristal de 25 MHz

La FPGA se conecta al LAN83C185 EtherNET PHY usando una MII[11]

Tabla 12: Conexiones del LAN83C185 EtherNET PHY

Señal	Función
E_TXD<4>	Datos transmitidos al PHY. E_TXD<4> también es el transmisor de error MII
E_TXD<3>	

[11] MII Media Independent Interface

E_TXD<2>	
E_TXD<1>	
E_TXD<0>	
E_TX_EN	Habilita la transmisión
E_TX_CLK	Reloj de Transmisión, 25 MHz en 100Base-Tx
E_RXD<4>	Receptor de los Datos del PHY
E_RXD<3>	
E_RXD<2>	
E_RXD<1>	
E_RXD<0>	
E_RX_DV	Válida los Datos Recibidos
E_RX_CLK	Reloj del Receptor, 25 MHz en modo 100Base-Tx
E_CRS	
E_COL	Detectar colisión MII
E_MDC	Gestión del Reloj
E_MDIO	Gestión de los Datos de Entrada y Salida

UCF Localización

```
NET "E_COL"    LOC = "U6"     | IOSTANDARD = LVCMOS33 ;
NET "E_CRS"    LOC = "U13"    | IOSTANDARD = LVCMOS33 ;
NET "E_MDC"    LOC = "P9"     | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 8 ;
NET "E_MDIO"   LOC = "U5"     | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 8 ;
NET "E_RX_CLK" LOC = "V3"     | IOSTANDARD = LVCMOS33 ;
NET "E_RX_DV"  LOC = "V2"     | IOSTANDARD = LVCMOS33 ;
NET "E_RXD<0>" LOC = "V8"     | IOSTANDARD = LVCMOS33 ;
NET "E_RXD<1>" LOC = "T11"    | IOSTANDARD = LVCMOS33 ;
NET "E_RXD<2>" LOC = "U11"    | IOSTANDARD = LVCMOS33 ;
NET "E_RXD<3>" LOC = "V14"    | IOSTANDARD = LVCMOS33 ;
NET "E_RXD<4>" LOC = "U14"    | IOSTANDARD = LVCMOS33 ;
NET "E_TX_CLK" LOC = "T7"     | IOSTANDARD = LVCMOS33 ;
NET "E_TX_EN"  LOC = "P15"    | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 8 ;
NET "E_TXD<0>" LOC = "R11"    | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 8 ;
NET "E_TXD<1>" LOC = "T15"    | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 8 ;
NET "E_TXD<2>" LOC = "R5"     | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 8 ;
NET "E_TXD<3>" LOC = "T5"     | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 8 ;
NET "E_TXD<4>" LOC = "R6"     | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 8 ;
```

CONFIGURACIÓN DE LA FPGA

La board del S3ESK ha incluido varias de las posibilidades de configuración de la FPGA para demostrar las capacidades de esta. La S3ESK tiene tres fuentes de memoria de configuración para demostrar la capacidad de funcionamiento en conjunto. También incluye en la board una conexión USB basada en interfaz de programación JTAG. Con este conjunto de opciones la configuración de la S3ESK puede realizarse así:

Descargando el diseño FPGA directamente a la FPGA Spartan-3E vía JTAG, usando la interfaz USB. La interfaz USB también permite programación en sistema de la Platform Flash PROM y el Xilinx XC2C64A CPLD.
Programando la Platform Flash PROM, entonces la configuración de la FPGA se realiza desde la imagen almacenada en la Flash PROM usando el modo Master Serial.
Programando la SPI Serial Flash PROM, la configuración de la FPGA se realiza desde la imagen almacenada en la SPI Serial Flash PROM usando el modo SPI.
Programando la StrataFlash Parallel NOR Flash PROM, la configuración de la FPGA se realiza desde la imagen almacenada en StrataFlash Parallel NOR Flash PROM usando los modos de configuración BPI Up o BPI Down.

El modo de configuración de la FPGA esta determinado por la configuración de los jumpers de modo J30.
El pin DONE esta conectado al led DONE, que se activa al finalizar la configuración de la FPGA.
Presionando el pulsador PROG se hace a la FPGA reiniciar el proceso de configuración.

USB

La USB disponible en la board nos permite configurar la FPGA, la Flash PROM y el CPLD que se encuentran en la S3ESK. La descarga de los archivos de configuración se realiza haciendo uso del software de programación IMPACT y el cable USB.

Para iniciar la configuración, debe seleccionar el modo Master Serial en los jumpers J30. Una vez establecido el modo, conectar el cable USB a la board y aplicar alimentación a la board.

Si el software IMPACT es invocado dentro del Project Navigator de ISE, automáticamente el software reconocerá los tres dispositivos que se encuentran en el archivo de programación del JTAG. En otro caso, en el Boundary Scan haga clic derecho y ejecute Initialize Chain o presione el juego de teclas Crtl+I; el software reconocerá los dispositivos.

Si la selección de archivos de configuración no se inicio automáticamente, haga clic derecho en el dispositivo que desee configurar, FPGA o CPDL, y seleccione Assign New Configuration File, y seleccione el archivo de configuración que desee.

Para iniciar la programación del dispositivo, haga clic derecho sobre el y seleccione Program. IMPACT realiza un reporte del proceso de programación. El realizar la programación directamente de la FPGA toma tan solo unos segundos, que tiene una amplia diferencia en comparación de la programación desde la SPI SERIAL Flash PROM o de la StrataFlash Parallel NOR Flash PROM, usando el microcontrolador PicoBlaze y las aplicaciones implementadas para programar las memorias.

IMPACT indicara la programación exitosa cuando esta se realice, por medio del mensaje, Program Succeeded, y en la board el led Done pasara a estar destellante.

Platform Flash PROM

Para realizar la configuración de la FPGA usando el modo Master Serial, la memoria Flash PROM debe ser programada con el archivo de configuración de la FPGA, dicha programación se realiza usando la interfaz USB de la misma forma que se uso programar la FPGA, pero antes debe generarse el respectivo archivo que se almacena en la Flash PROM.

La FPGA proporciona un reloj de salida, CCLK, cuando carga la configuración de una PROM. El CCLK interno de la FPGA oscila aproximadamente a 1.5 MHz, Como la PROM externa soporta una alta frecuencia, entonces puede ser incrementada la frecuencia del CCLK como una forma de reducir el tiempo de configuración de la FPGA. La Xilinx XCF04S soporta una frecuencia de CCLK de 25MHz.

Para incrementarla frecuencia de CCLK haciendo uso del Project Manajer de ISE, hacer clic derecho en Generator Programming File y seleccionar Properties con clic izquierdo. Seleccionando Configuration Options del árbol de Process Properties, cambiar el campo Configuration Rate a 25, para de esta forma incrementar la frecuencia a aproximadamente 25MHz. clic en OK para confirmar el cambio en las propiedades de configuración y hacer doble clic en Generate Programming File para generar el nuevo bitstream a partir del cual se obtendrá la imagen a almacenar en la Flash PROM.

Después de generar el archivo de programación, cambiando o no la frecuencia de CCLK, doble clic en Generate PROM, ACE or JTAG File para ejecutar el software IMPACT. En IMPACT doble clic en PROM File Formatter, por medio de esta se seleccionan las opciones de archivo para las diferentes formas de configuración, tanto dispositivos, densidad de estos y todas las otras opciones que se hacen necesarias establecer.
Para la Flash Platform PROM, selecciona Xilinx PROM, el formato de archivo MCS, indicar la ubicación donde se desea guardar y el nombre que llevara el archivo, luego en

Next, para continuar con la preparación. Continúe seleccionando la familia de la PROM y el correspondiente dispositivo, y agregue a la lista haciendo clic en Add, y luego en Next. Verifique las opciones de la generación de archivo y haga clic en Finish. En los cuadros de pregunta, rechace agregar otro archivo de dispositivo. La configuración ya esta completa, ahora se genera el archivo de configuración en Opearions/Generate File; el archivo para programar la memoria será creado y guardado en la ubicación que se indico.

El siguiente paso es descargar el archivo a la XCF04S, que se realiza siguiendo los pasos descritos en la configuración por USB.

SPI Serial Flash PROM

La memoria SPI Flash PROM puede ser programada desde perspectivas diferentes, con el uso de las herramientas de programación del Project Navigator, ya sea por consola o interfaz grafica, o haciendo uso de el microcontrolador PicoBlaze.

El almacenar la configuración de la FPGA en la SPI Serial FLASH PROM puede realizarse haciendo uso de una aplicación creada por Ken Chapman, creador del microcontrolador PicoBlaze.
La aplicación establece una comunicación serial RS-232, por medio de la cual se permite el acceso completo a las memorias, lectura y escritura, tanto de una dirección como de toda la memoria.
El primer paso que se lleva a cabo en esta forma de realizar la configuración es programar la FPGA con la aplicación requerida para el acceso a la memoria, se puede usar dos métodos, configurar la FPGA vía JTAG o configurar primero la Platform Flash PROM con una imagen de la aplicación para luego cargarla en la FPGA. Es recomendable almacenar la configuración en la Platform dado qué es una forma más rápida de reconfigurar la FPGA para hacer uso nuevamente de la aplicación en caso de ser necesario.
Generar el archivo de configuración mcs adecuado para el dispositivo, usando la herramienta Impact, y enviar de los datos de programación entre una estación de trabajo y el microcontrolador PicoBlaze para que los escriba en la memoria SPI Serial FLASH PROM.

StrataFlash Parallel NOR Flash PROM

El almacenar la configuración de la FPGA en la StrataFlash Parallel NOR Flash PROM puede realizarse haciendo uso de la aplicación NOR FLASH Programmer for Spartan-3E Starter Kit, creada por Ken Chapman.
El primer paso que se lleva a cabo en esta forma de realizar la configuración es programar la FPGA con la aplicación requerida para el acceso a la memoria, se puede usar dos métodos, configurar la FPGA vía JTAG o configurar primero la Platform Flash

PROM con una imagen de la aplicación para luego cargarla en la FPGA. Es recomendable almacenar la configuración en la Platform dado qué es una forma más rápida de reconfigurar la FPGA para hacer uso nuevamente de la aplicación en caso de ser necesario.
Usando la herramienta Impact generar el archivo de configuración .mcs adecuado para el dispositivo y para el modo de configuración BPI, up o down. Luego enviar de los datos de programación entre una estación de trabajo y el microcontrolador PicoBlaze, para ser escritos en la memoria SPI Serial FLASH PROM.

REFERENCIAS BIBLIOGRAFICAS

[1] A Complete Portfolio of Compatible Serial EEPROMs and Flash Memories for Flexible Storage of Parameters and Code

[2] Andrew Marshall, Sreedhar Natarajan, SOI Design: Analog, Memory and Digital Techniques 2002. USA Pág. 279

[3] Linear Technology Corporation, LTC1407/LTC1407A - Serial 12-Bit/14-Bit, 3Msps Simultaneous Sampling ADCs with Shutdown. Linear Technology Corporation, LT 0506 Rev A

[4] Linear Technology Corporation, LTC2604 Quad 16-Bit Rail-to-Rail DACs in 16-Lead SSOP, LT 0108 Rev C

[5] Linear Technology Corporation, LTC6912 - Dual Programmable Gain Amplifiers with Serial Digital Interface, LT/LT 1005 Rev A

[6] Maxim, DS2432 1Kb Protected 1-Wire EEPROM with SHA-1 Engine, Junio 6, 2008

[7] Micron Technology, 512Mb DDR SDRAM (x4,x8,x16) Component Data Sheet

[8] SMSC, LAN83C25 High Performance Single Chip Low Power 10/100 EtherNET Physical Layer Transceiver (PHY), Junio 12, 2008

[9] STMicroelectronics, M25P16, Mayo 2004

[10] The Code Shadowing Approach: Choose The Correct Non-Volatile Memory.

[11] Xilinx DS090 CoolRunner-II CPLD Family, v3.1, Septiembre 11, 2008

[12] Xilinx DS128 Platform Flash In-System Programmable Configuration PROMs, Julio 7, 2008

[13] Xilinx DS312-2 Spartan-3E FPGA Family Data Sheet, v3.6, Mayo 29, 2007

[14] Xilinx UG230 Spartan-3E Starter Kit Board User Guide, v1.0, Marzo 9, 2006

[15] Xilinx UG332 Spartan-3 Generation Configure User Guide, v1.4, Julio 1, 2008.

ANEXO G

Tutorial Para Crear Un Proyecto VHDL en ISE

TUTORIAL PARA CREAR UN PROYECTO VHDL EN ISE

CREAR UN DISEÑO USANDO XILINX ISE

Para la creación, síntesis e implementación de proyectos de diseño de descripción de hardware se usa el paquete de software Xilinx ISE. Por medio del cual se selecciona el dispositivo a usar, se sintetiza la descripción de hardware, se realiza la configuración de pines y tiempos necesaria para el proyecto.

Pasos para la creación de proyectos usando Xilinx ISE:

Crear el proyecto (*New Project Wizard*)
File > New Proyect. Inicia una serie de pasos para indicar el nombre del proyecto en el que se trabaja, la ubicación donde se almacenaran los archivos generados, la forma principal de descripción del proyecto, el lenguaje de descripción, etc.

La primera ventana, *New Project Wizard – Create New Proyect*, asignar:

- *Project Name*. El nombre que llevara el proyecto. Se debe seguir las convención de creación de archivos, el nombre debe iniciar con una letra (A-Z, a-z) y contener solo caracteres alfa numéricos (A-Z, a-z, 0-9) y subraya (_).
- *Project Location*. La ubicación del proyecto en la estación de trabajo. Ingresa la ubicación de la carpeta o buscarla mediante el explorador. Importante, cada proyecto debe tener su propia subcarpeta, pueden ocurrir conflictos de archivos si múltiples proyectos comparten una carpeta.
- *Top-Level Source Type.* El tipo de fuente principal del proyecto, puede selecciónar cuatro diferentes tipos de la lista:
    - HDL, si el diseño principal es un archivo VHDL, Verilog o ABEL (para CPLDs). Puede incluir otros módulos secundarios de diferentes tipos de archivo, otros archivos HDL, esquemáticos y , IP Cores y archivos EDIF[1]
    - Schematics, si su archivo principal es una archivo esquemático. Puede incluir otros módulos secundarios de diferentes tipos de archivo, otros archivos HDL, esquemáticos, cajas negras[2]. Project Navigator automáticamente convierte los archivos schematics en el diseño a una estructura HDL antes de la implementación, debe especificarse una herramienta de síntesis cuando se trabaja con proyectos schematics.
    - EDIF y NGC/NGO, seleccionar si se convirtió un proyecto a alguno de estos tipos de archivo. Usar este tipo de archivos permite saltar el proceso de síntesis del Project Navigator e iniciar el proceso de implementación.

[1] EDIF iniciales de Electronic Design Interchange Format.
[2] Cajas Negras, nombre con el que se conocen a los IP Cores y archivos EDIF.

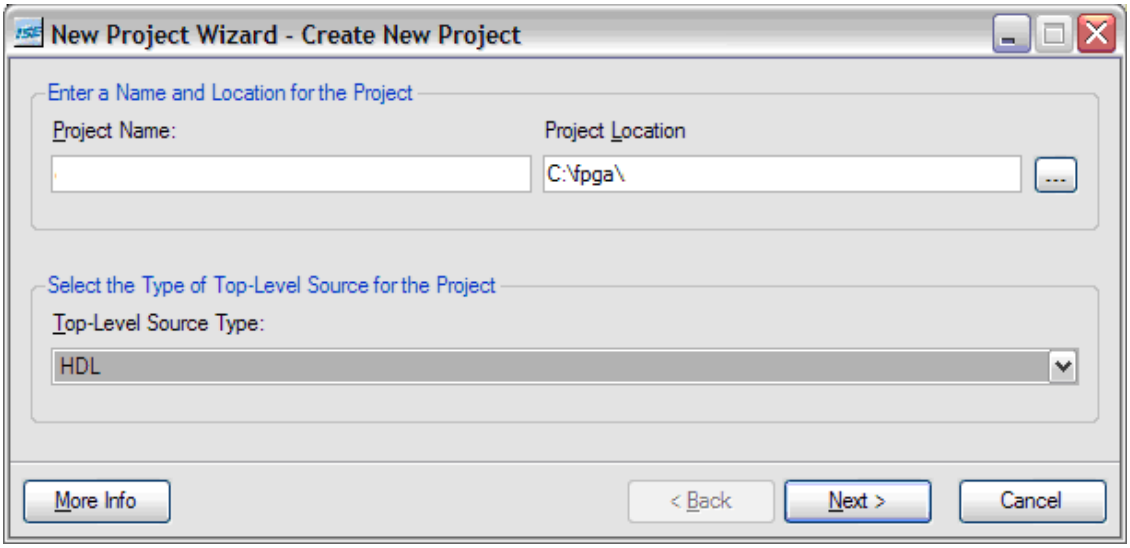


Figura 1: New Project Wizard – Create New Project

Clic en *Next* para continuar.

Como los diseños usados son del tipo HDL o Schematic, consecuentemente el siguiente paso será la ventana *New Project Wizard – Device Properties*, se establecen las propiedades del dispositivo:

- *Property Name*
- *Product Category*, permite realizar una búsqueda rápida de las características del dispositivo que se desea usar.
  - Para la FPGA:
    - Family: Spartan-3E
    - Device: XC3S500E
    - Package: FG320
    - Speed: -4
  - Para el CPLD
    - Family: CoolRunner2 CPLDs
    - Device: XC2C64A
    - Package: VQ44
    - Speed: -5
- *Top-Level Source Type*, se establece automáticamente.
- *Synthesis Tool*, Herramienta de síntesis, seleccionar una de las herramientas de síntesis y lenguaje HDL del proyecto. VHDL/Verilog es un lenguaje de flujo mixto. Para simular el comportamiento, el simulador debe soportar múltiple lenguaje de simulación.
  - XST (Xilinx® Synthesis Technology), esta disponible con la instalación del software ISE Foundation™. Soporta proyectos que incluyan archivos de diseño esquemáticos y proyectos que incluyan fuentes archivos de lenguaje mixto, es decir archivos VHDL o Verilog en el mismo proyecto.
  - Synplify y Synplify Pro (Synplicity®, Inc.). Synplicity® no soporta proyectos que incluyan archivos de lenguaje mixto. Synplicity Pro®

soporta proyectos que incluyen archivos de lenguaje mixto. Ninguno de los dos soportar archivos de diseño esquemáticos.
    - Precision (Mentor Graphics®, Inc.). Soporta proyectos que incluyan archivos de diseño esquemáticos y proyectos que incluyan fuentes archivos de lenguaje mixto, es decir archivos VHDL o Verilog en el mismo proyecto.

  Las herramientas de síntesis Synplify, Synplify Pro y Precision, no son incluidas en el Project Navigator.
- *Simulator*. Simulador, seleccionar una de las herramientas simulación:
    - ISE Simulator (Xilinx®, Inc.), este simulador permite ejecutar el proceso de simulación como parte del flujo de diseño ISE.
    - ModelSim (Mentor Graphics®, Inc.), integra la simulación como un proceso del flujo de diseño ISE usando cualquiera de las siguientes ediciones ModelSim®: ModelSim Xilinx Edition (MXE), ModelSim MXE Starter, ModelSim PE, or ModelSim SE™.
    - NC-Sim (Cadence®, Inc.) y VCS (Synopsys®, Inc.) no son simuladores integrados con ISE y se dejen ejecutar independientemente.
    - Other. Otros, seleccionar esta opción si no tiene ISE Simulator o ModelSim instalado o si se quiere ejecutar la simulación fuera de Project Navigator. Indica al Project Navegator desactivar los procesos integrados de simulación para el proyecto.

  El uso de un simulador no integrado a ISE se debe especificar, para garantizar que los archivos generados estén escritos en el formato correcto.
  ModelSim no se incluye en la instalación de Project Navigator.
- *Preferred Language*. Seleccionar un lenguaje para establecer un lenguaje preferido. La propiedad del proyecto, Preferred Language, controla la configuración predeterminada de la propiedad de proceso que genera la salida HDL. Si la opcion de la herramienta de síntesis y/o simulación son establecidas en un solo lenguaje, el lenguaje establecido para generar los archivos HDL será automáticamente cambiando al adecuado.
    - Verilog, seleccionar si ambas herramienta de síntesis y de simulación se establecen a lenguaje mixto y se quiere que el idioma por defecto sea Verilog.
    - VHDL, seleccionar si ambas herramienta de síntesis y de simulación se establecen a lenguaje mixto y se quiere que el idioma por defecto sea VHDL.
    - N/A, esta opción es mostrada si ambas herramientas síntesis y simulación son establecidas a un solo lenguaje.
- *Enable Enhanced Design Summary*. Resumen de diseño. Seleccionar para mostrar el número de errores y advertencias para cada uno de los informes de Resumen de diseño (Design Summary).
- *Enable Message Filtering*. Filtrado de mensajes. Activar esta opción para mostrar el número de mensajes filtrados en el Design Summary. Se tiene que ejecutar el software para mostrar el número de mensajes filtrados.
- *Display Incremental Messages*. Mostrar mensajes incrementales. Activar esta opción para mostrar el número de mensajes nuevos para la ejecución mas reciente

del software en el Resumen de diseño. Se tiene que ejecutar el software para mostrar el número de mensajes filtrados.

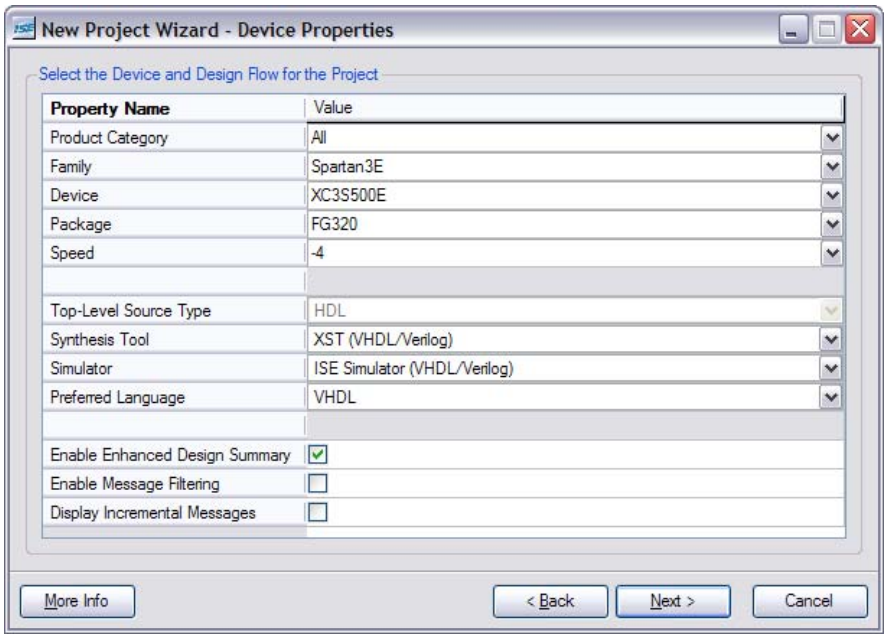


Figura 2: Project Wizard – Device Properties

Estas opciones pueden ser modificadas posteriormente en el área de fuentes. Source for: Post-Route Simulation, y doble clic en el dispositivo.

Clic en *Next* para continuar. Al estar creando un proyecto HDL o esquemático se tiene la opción de crear un nuevo archivo fuente para el proyecto, usando *New Source Wizard*, se puede continuar sin crear un archivo haciendo clic en Next. Para crear el archivo el primer paso es hacer clic en el boton *New Source*.

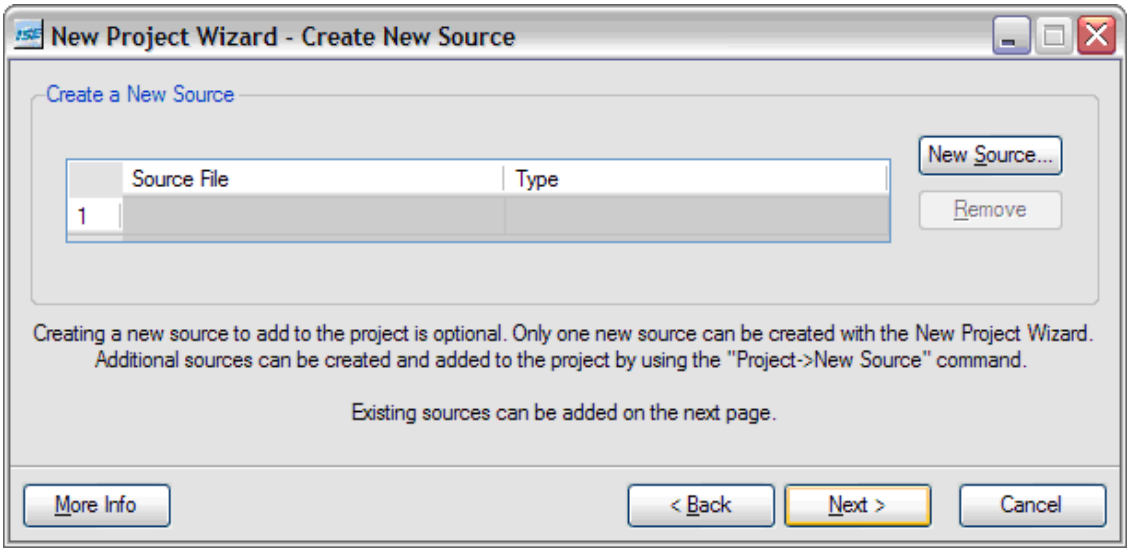


Figura 3: New Project Wizard – Create New Source

En la ventana *New Source Wizard – Select Source Type*:

- Seleccione el tipo de archivo.
- *File Name*. Ingrese el nombre del archivo
- Location. La ubicación es la que se estableció en la creación del proyecto.
- La casilla Add to project indica que el archivo es agregado a el proyecto. Clic en Next.
- En algunos casos, una nueva ventana solicita información adicional para crear el tipo de archive seleccionado, como un tipo de modulo y asociación o nombres de puertos, se ingresan los datos requeridos y clic en *Next*.
- Clic*Finish*.

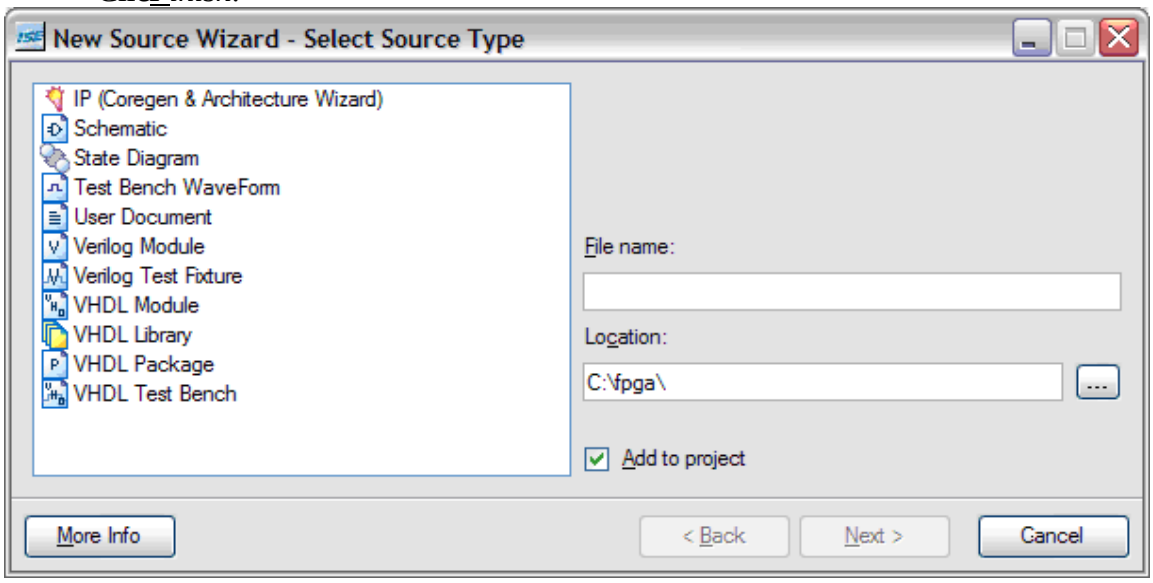


Figura 4: New Source Wizard – Select Source Type

New Project Wizard solo permite la creación de un archivo fuente. Sin embargo, una vez creado el proyecto se pueden crear nuevos archivos fuente.

En el siguiente paso se pueden agregar uno o mas archivos existente a el proyecto que se esta creando. Si no se agregan nuevos archivos al proyectohagaclic en *Next*. Para agregar las fuentes ya existentes haga clic en el boton *Add Source*, explore, seleccione el archivo o archivos y hacer clic en *Open*.
Se muestran los archivos que se agregaran al proyecto, si la casilla *Copy to Project* se encuentra activa los archivo son copiados a la carpeta del proyecto.
Para continuar con el New Project Wizard hacer clic en *Next*.

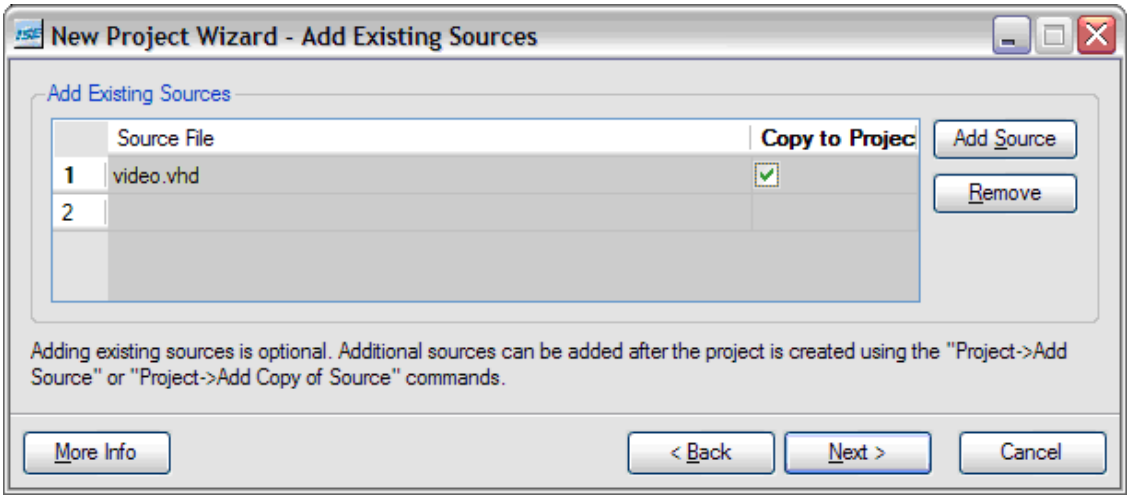


Figura 5: New Project Wizard – Add Existing Sources

Finalmente se presenta un resumen de las propiedades del proyecto, donde se puede corroborar el proceso realizado. El proyecto es creado después de hacer clic en *Finish*

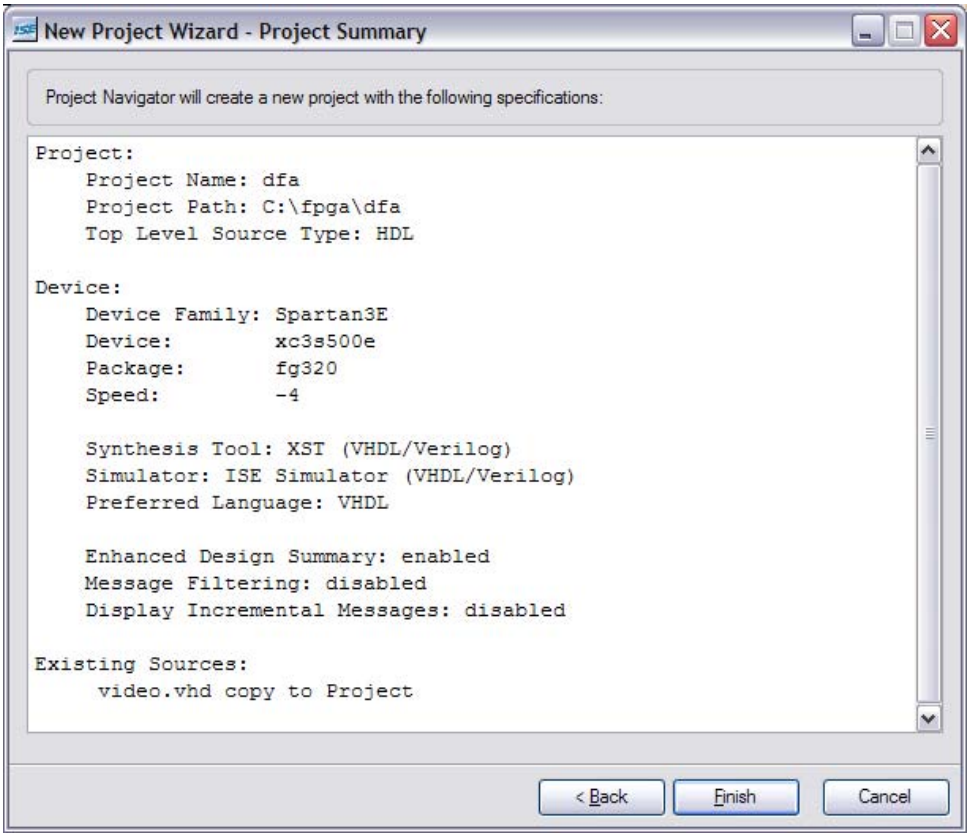


Figura 6: New Project Wizard – Project Summary

CREAR UN ARCHIVO FUENTE HDL

Para crear un archivo fuente HDL siga los siguientes pasos:

Haga clic derecho en la ventana *Sources* y seleccione *New Source*. El archivo puedo ser creado durante la creación del proyecto al hacer de clic en le boton *New Source* de la ventana *New Project Wizard.*

En la ventana *New Source Wizard – Select Source Type*, seleccione VHDL Module.

Escriba el nombre del archivo.

Verifique que la casilla *Add to Project* esté seleccionada.

Haga clic en *Next*.

Declare los puertos para el diseño, ingresando la información necesaria para cada puerto, Figura 7.

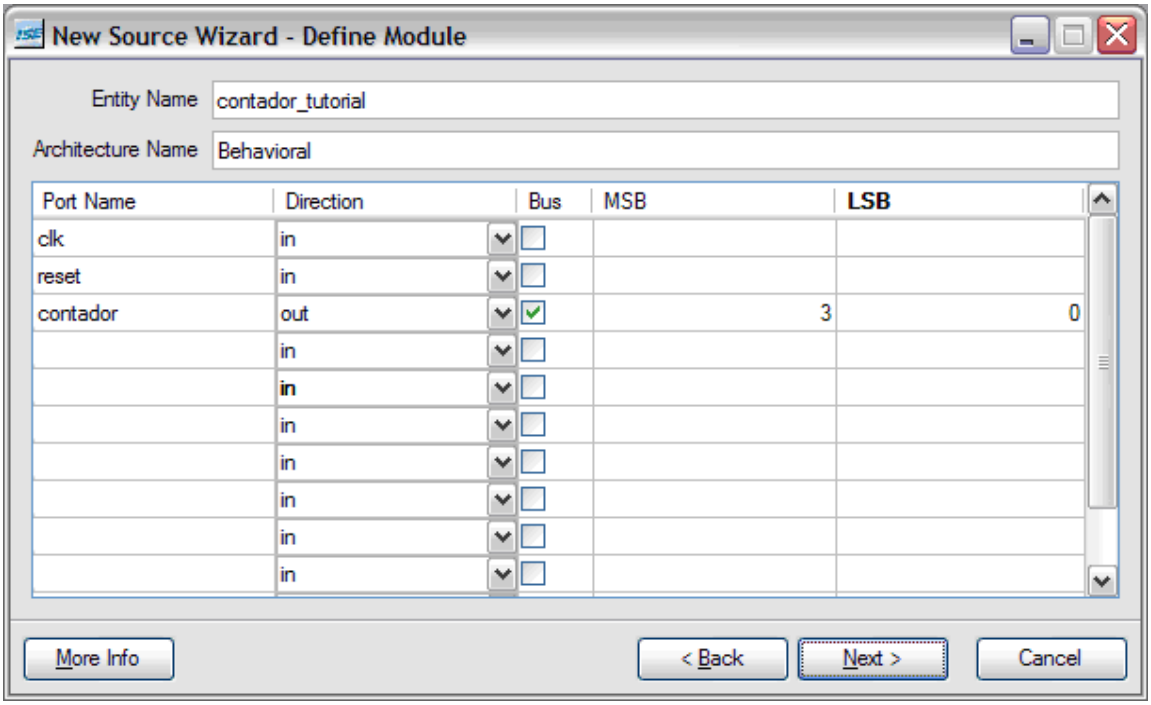


Figura 7: New Source Wizard – Define Module

Haga clic en *Next*, luego en *Finish* en el cuadro de dialogo New Source Wizard – Summary, para completar la creación del archivo, Figura 8

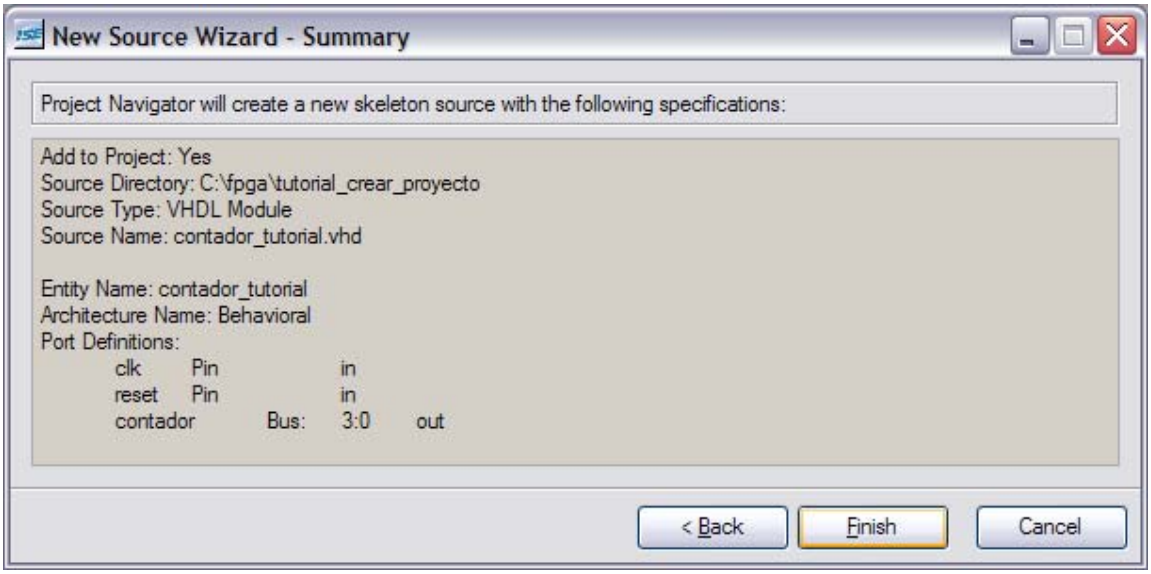


Figura 8: New Source Wizard - Summary

En un proyecto HDL, el archivo HDL de alto nivel está encargado de agregar los diferentes componentes que comprende el proyecto y describir la forma en que interactúan entre ellos, o es el que contiene una única descripción de hardware

VERIFICAR LA SINTAXIS DE UN MODULO

Cuando el archivo HDL esta completo se verifica la sintaxis del diseño, para encontrar los errores que este pueda contener.

Verifique en la ventana *Sources* que este seleccionada *Synthesis/Implementation* en la lista *Source for*, Figura 9.

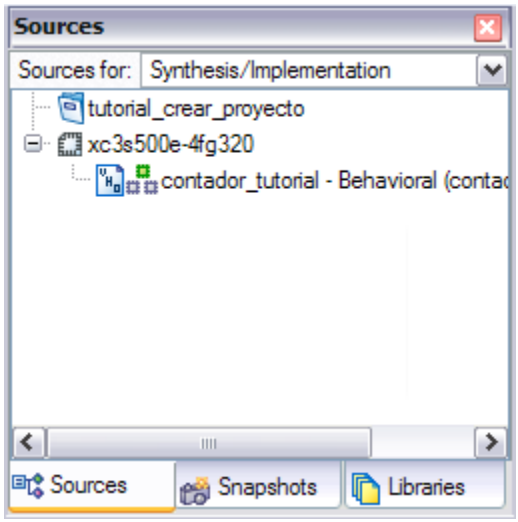


Figura 9: Sources

Seleccione el archivo de alto nivel del diseño HDL en la ventana *Sources*, para mostrar en la ventana *Processes* los procesos relacionados a este archivo.

Haga clic en "+" el proceso Synthesize-XST para expandir este grupo de procesos, Figura 10, haga doble clic en el proceso Check Syntax, para verificar la correcta sintaxis del lenguaje de descripción. Se deben corregir los errores que encuentre en los archivos fuente. De no realizar las correcciones necesarias no es posible realizar la simulación del diseño o su implementación.

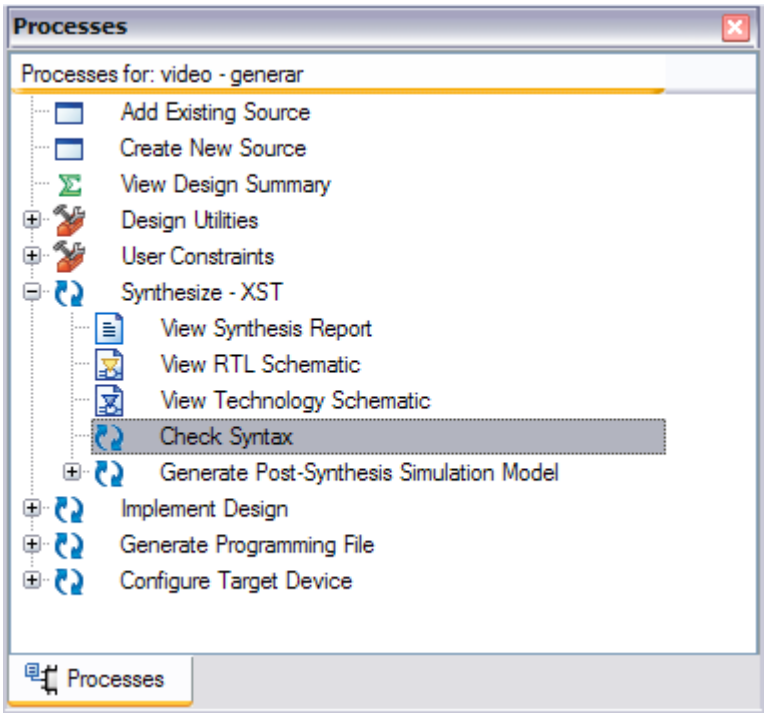


Figura 10: Processes

ASIGNAR LA UBICACIÓN DE LOS PINES

Para asignar los pines a los puertos del diseño siga los siguientes pasos:

Seleccione el archivo HDL de alto nivel del diseño.
Haga doble clic en el proceso Floorplan IO – Pre-Synthesis, del grupo de procesos *User Constraints*. Se iniciará el software Xilinix Pinout and Area Constraints Editor (PACE), Figura 11.

En la ventana *Design Object List – I/O Pins*, ingrese la ubicación de cada pin en la columna Loc siguiendo la siguiente información:

La entrada clk conectada al pin T9 de la FPGA
La entrada reset conectada al pin V16 de la FPGA
La salida Contador<0> conectada al pin K12 de la FPGA

La salida Contador<1> conectada al pin P14 de la FPGA
La salida Contador<2> conectada al pin L12 de la FPGA
La salida Contador<0> conectada al pin N14 de la FPGA

Seleccione **File**>**Save**. Seleccione **XST Default** <>y haga clic en *OK*, para terminar cierre PACE.

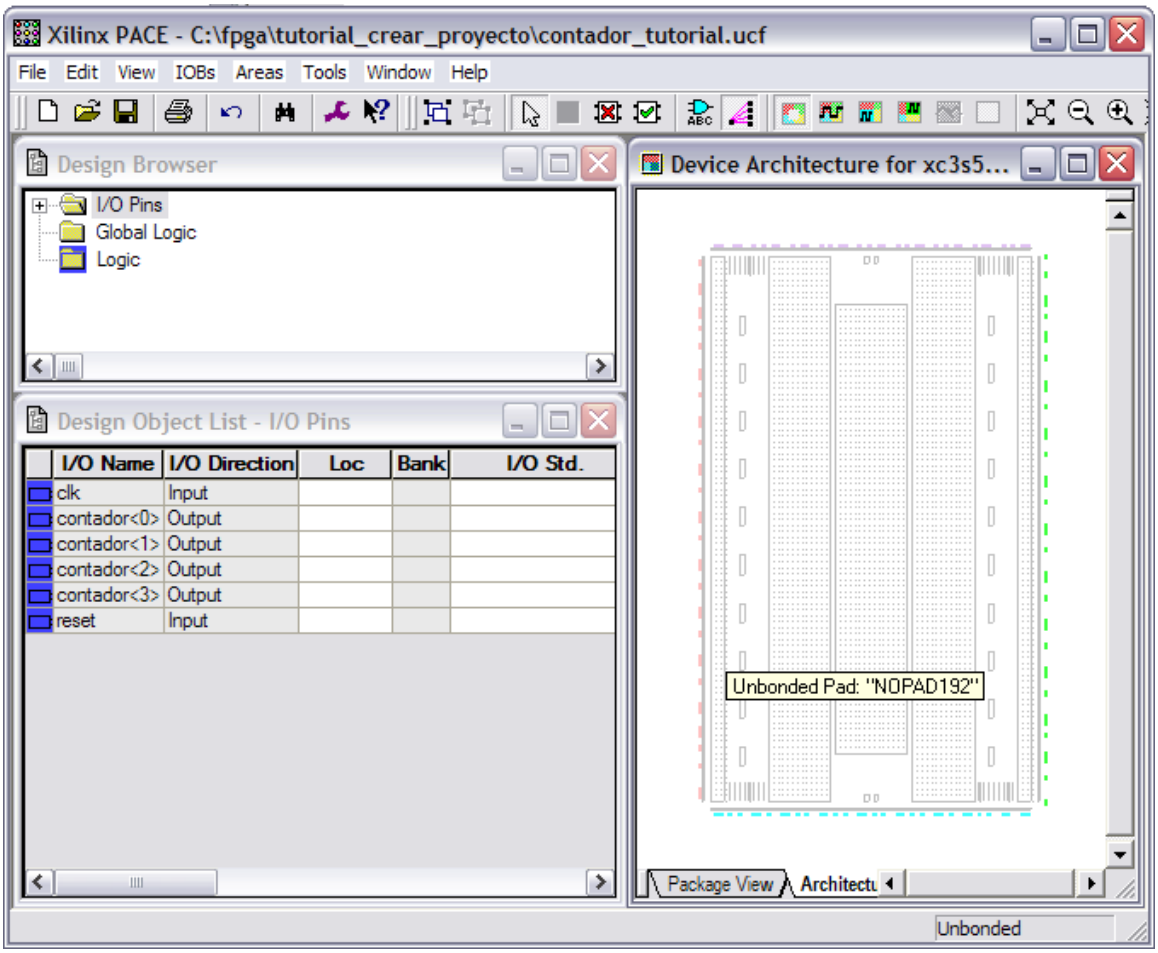


Figura 11: Xilinx PACE

IMPLEMENTAR EL DISEÑO

Seleccione el archivo HDL de alto nivel del diseño.

Haga doble clic sobre el grupo de procesos *Implement Design*

Para realizar la implementación, los errores encontrados por el proceso *Implement Design* deben ser corregidos para poder descargar el diseño al FPGA. El globo verde es la indicación que el proceso se realizo de forma optima y no contiene errores. Un globo amarillo indicara que el proceso encontró algunos peligros, es conveniente revisar estas indicaciones para en pasos futuros no encontrar mayores problemas. Finalmente un globo

rojo detiene la ejecución del proceso al encontrar errores y no podrá continuar realizándose hasta darles solución a todos ellos, Figura 12.

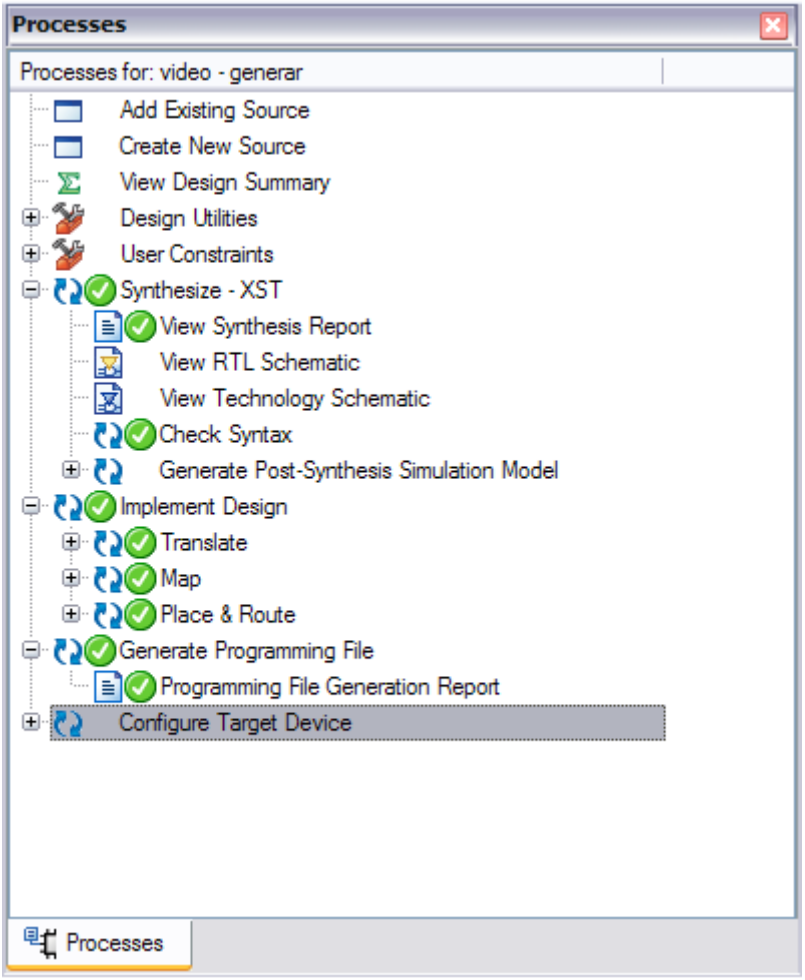


Figura 12: Processes Programming File Generation Report, Proyecto listo para ser configurado en la FPGA

ANEXO H

Manual De Configuración De La S3ESK

MANUAL DE CONFIGURACIÓN DE LA S3ESK

La board del Spartan-3E Starter Kit (S3ESK), es una plataforma de Xilinx que junto con el software ISE permite diseñar, simular e implementar Sistemas Embebidos (SE). Esta plataforma posee grandes posibilidades de diseño pues cuenta con una FPGA de la Familia Spartan-3E, un CPLD de la familia CoolRunner–II, y adicionalmente memorias e interfaces de entrada y salida de datos.

La plataforma S3ESK cuenta con:

- FPGA Xilinx Spartan-3E
- CPLD Xilinx CoolRunner-II
- Cuatro switch, cuatro pulsadores y una perilla
- Pantalla LCD.
- Puerto VGA
- Puertos Seriales RS-232
- Conversores DA y AD
- Puerto PS/2
- Interfase Física Ethernet
- Xilinx Flash PROM
- SPI serial Flash
- Intel StrataFlash Parallel NOR Flash PROM
- EEPROM 1-Wire SHA-1
- Conectores de expansión de 6 pines y 100 pines (Hirose FX2 Edge)

Propósito de este manual:

En el proceso de diseño de SE basados en FPGAs encontramos una etapa encargada de almacenar y realizar la configuración de la FPGA. En este manual se presentan las diferentes formas de configuración con que las que cuenta la S3ESK y se muestra cómo llevar a cabo cada una de ellas.

Antes de la Configuración

Para realizar la configuración de la FPGA es necesaria una estación de trabajo que cuente con el software de programación iMPACT que hace parte del paquete de herramientas de Xilinx ISE.

Identifique:

- Conector de alimentación de la S3SK
- Switch de encendido/apagado
- Led de encendido
- Puerto USB de programación

- Jumpers de configuración J30
- Botón pulsador de programación
- Led de programación (DONE).

Los tres últimos ítems son fundamentales para establecer el modo de configuración, ejecutarla y verificar que esta fue realizada.

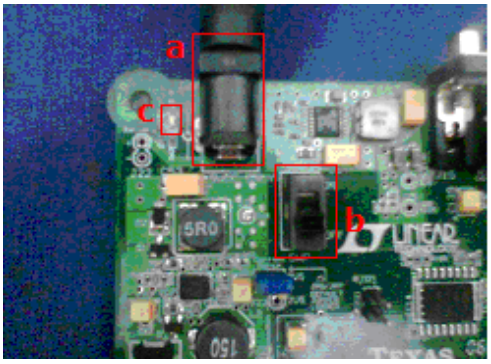


Figura 1: Conector de alimentación (a), Switch de encendido/apagado, Led de encendido (c)

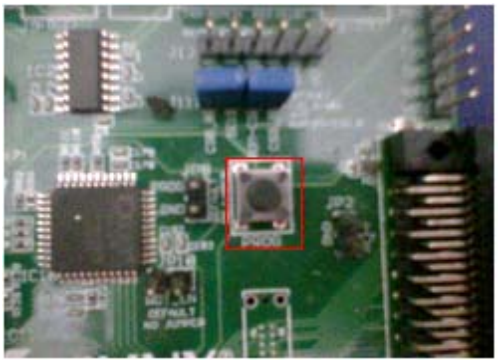


Figura 2: Pulsador de programación.

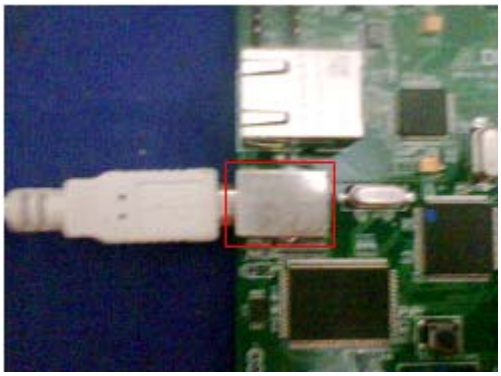


Figura 3: Puerto USB de programación.

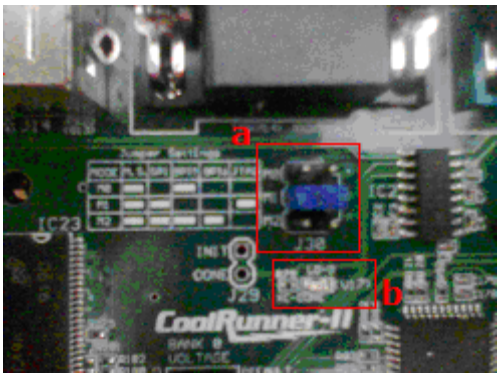


Figura 4: Jumpers J30 de configuración (a) y led de programación (DONE) (b).

Modos de Configuración

La familia Spartan-3E permite diversas formas de configuración, la S3ESK cuenta con varios de estos diferentes modos de configurar la FPGA. En la tabla se presenta el resumen de los modos como se configuran los jumpers J30 y los dispositivos que interactúan.

Selección del modo de configuración mediante la Tabla 1.

Tabla 1. Modos de Configuración.

Modo de Configuración	**Jumpers J30 M2 M1 M0**	**Origen de la Configuración**
JTAG	0 1 0	Programación desde el puerto USB vía JTAG
Master Serial	0 0 0	Platform Flash PROM
SPI	1 1 0	SPI Serial Flash PROM, iniciando en la dirección 0
BPI Up	0 1 0	StrataFlash Parallel Flash Prom, iniciando en la dirección 0 y incrementando por posición de dirección. El CPLD controla las líneas A[24:20] durante la configuración BPI
BPI Down	0 1 1	StrataFlash Parallel Flash Prom, iniciando en la dirección 1FF_FFFF y aumenta por posición de dirección. El CPLD controla las líneas A[24:20] durante la configuración BPI

Configuración USB-JTAG

El primer modo de configuración a implementar es el realizado mediante la interfase JTAG, para establecer una programación rápida, aunque volátil dado que no es posible almacenarla en alguna de las memorias disponibles y tampoco en la S3ESK, como sucede en las FPGA Spartan-3AN.

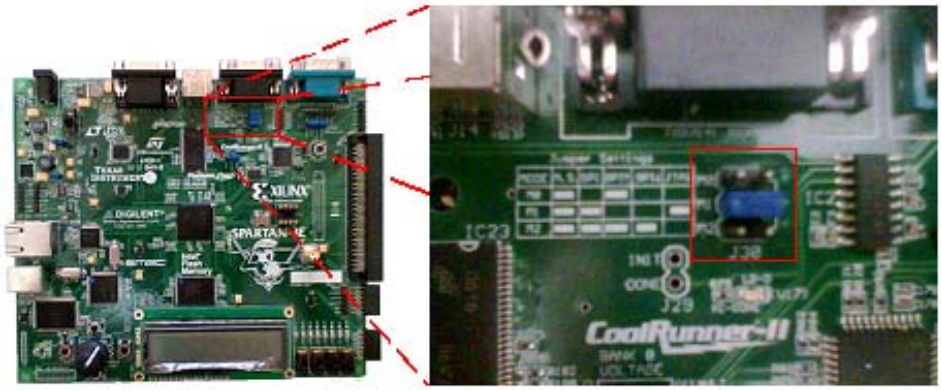


Figura 5: Jumpers J30 Configuración USB-JTAG

Para iniciar la configuración, se debe seleccionar el modo Master Serial en los jumpers J30. Una vez establecido el modo, se conecta el cable USB desde el puerto USB en la S3SK a un puerto USB de la estación de trabajo. Luego se conecta el adaptador AC/DC de 5V al conector de alimentación de la board, y se pone el switch de encendido a la posición ON.

Luego de Generar el archivo de programación en el Project Navigator de ISE, se hace doble clic en Manage Configure Project (iMPACT), del proceso Configure Target Device, para iniciar el software iMPACT, Figura 6. La acción Configure device using Boundary-Scan (JTGA) es la que permite realizar la configuración, se selecciona y se hace clic en *Finish*, Figura 7. Automáticamente el software reconocerá los tres dispositivos que se encuentran en el archivo de programación del JTAG.

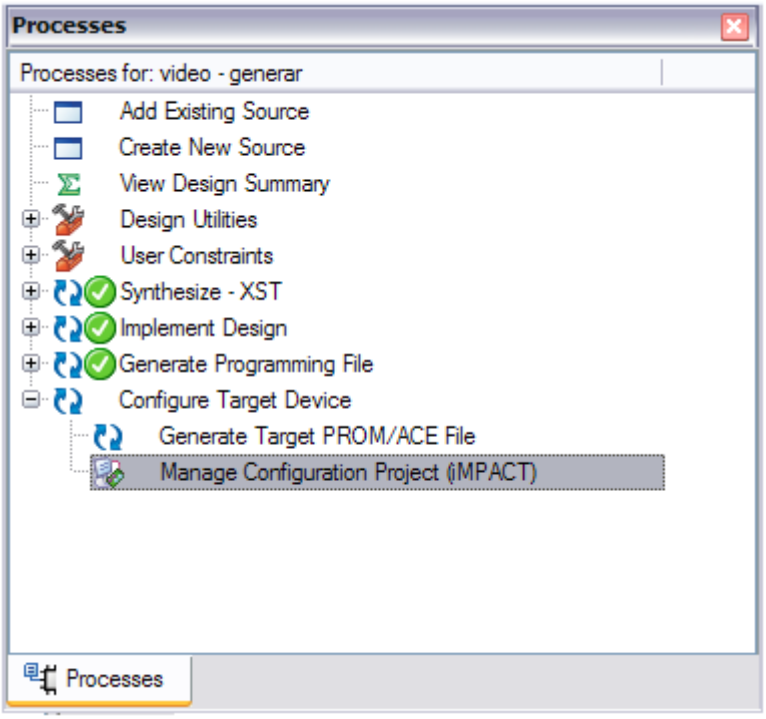


Figura 6: Proceso del proyecto que se realiza

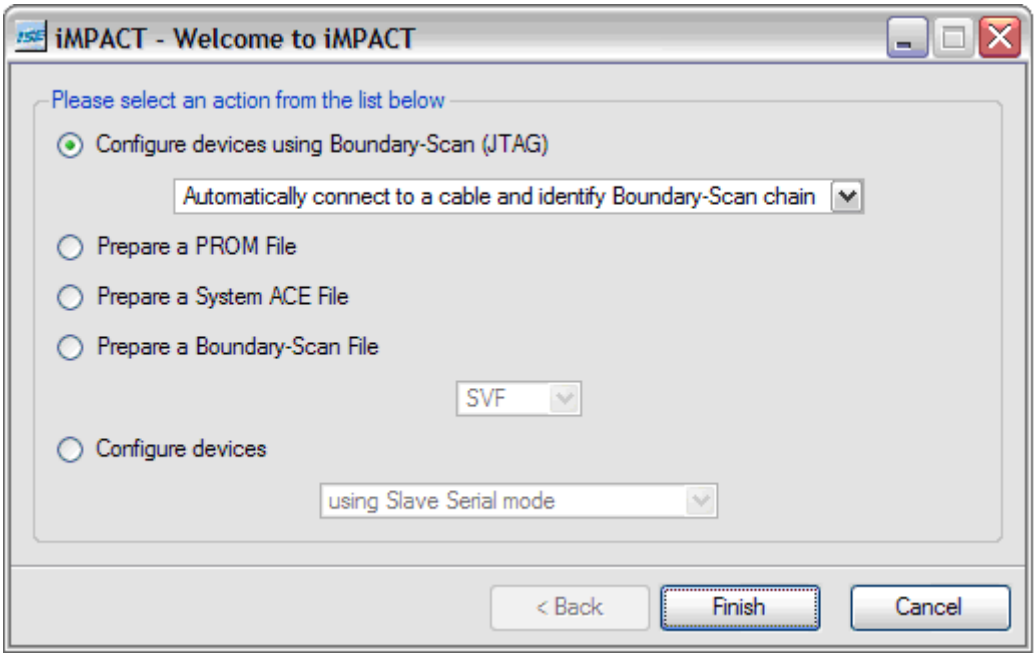


Figura 7: iMPACT

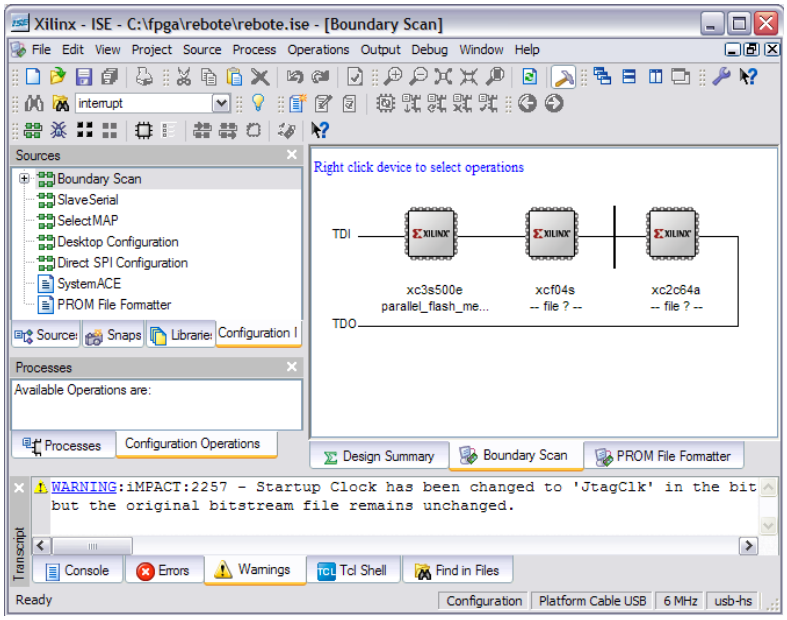


Figura 8: iMPACT

Si los dispositivos no son reconocidos automáticamente se hace clic derecho y se ejecuta Initialize Chain, se selecciona el archivos de configuración para el dispositivo que va a configurar; si la selección de archivo de configuración no se inicio se hace clic derecho sobre su imagen, y se selecciona Assign New Configuration File, y se indica el archivo de configuración, Figura 9.

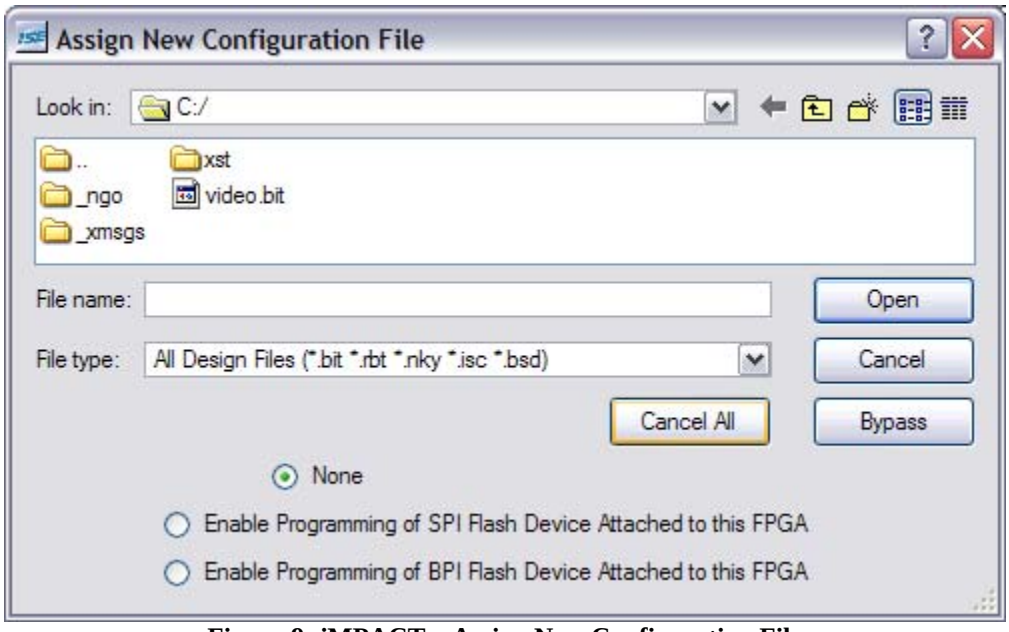


Figura 9: iMPACT – Assign New Configuration File

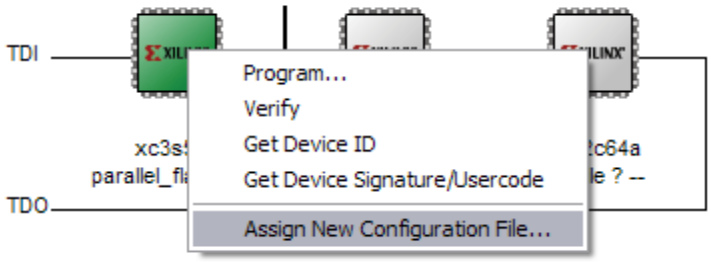


Figura 10: iMPACT – Asignar un nuevo archivo de configuración

Para iniciar la programación del dispositivo, se hace clic derecho sobre él y se selecciona Program. iMPACT realizar directamente la programación de la FPGA, que toma tan solo unos segundos y realiza un reporte del proceso.

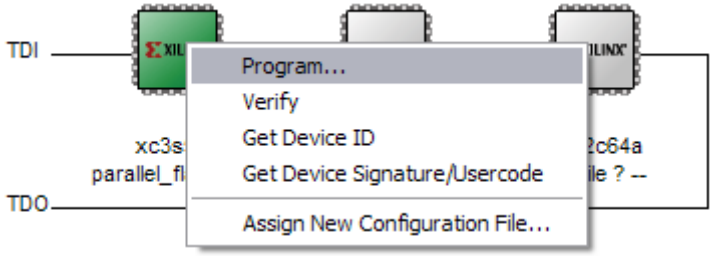


Figura 11: iMPACT – Programar

iMPACT indicará la programación exitosa cuando esta se realice, por medio del mensaje, Program Succeeded, y en la board el led Done pasara a estar destellante.

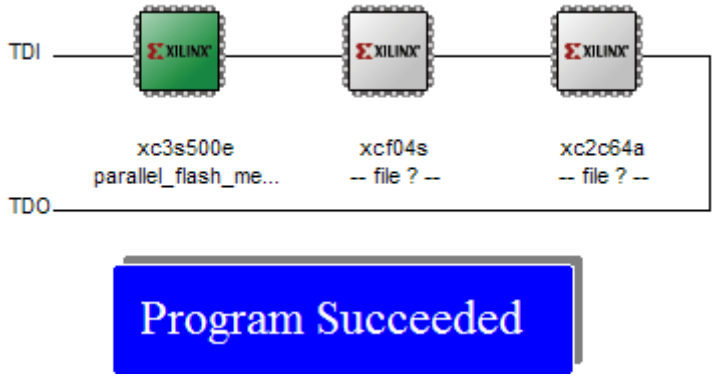


Figura 12: iMPACT – Programación realizada

Figura 13: S3ESK Led DONE

Configuración Master Serial

Para realizar la configuración de la FPGA usando el modo Master Serial, la memoria Platform Flash PROM debe ser programada con el archivo de configuración de la FPGA, dicha programación se realiza usando la interfase USB.

Generación del archivo de configuración:

La FPGA proporciona un reloj de salida, CCLK, cuando carga la configuración de una Flash PROM. El CCLK interno de la FPGA oscila aproximadamente a 1.5 MHz, la Platform Flash PROM del S3ESK soporta una alta frecuencia, pudiéndose incrementar la frecuencia del CCLK como una forma de reducir el tiempo de configuración de la FPGA. Para realizar la configuración en modo Master Serial se dispone de la memoria Xilinx XCF04S, que soporta una frecuencia CCLK de 25MHz.

Para incrementar la frecuencia de CCLK haciendo uso del Project Manajer de ISE, se hace clic derecho en Generator Programming File y se selecciona Properties con clic izquierdo. El árbol de Process Properties se selecciona Configuration Options, modificando el campo Configuration Rate al valor 25, de esta forma se incrementa la frecuencia a aproximadamente a 25MHz, Figura 14. Al hacer clic *OK* se confirmar el cambio en las propiedades de configuración. Y haciendo doble clic en Generate Programming File se generar el nuevo bitstream a partir del cual se obtendrá la imagen a almacenar en la Flash PROM.

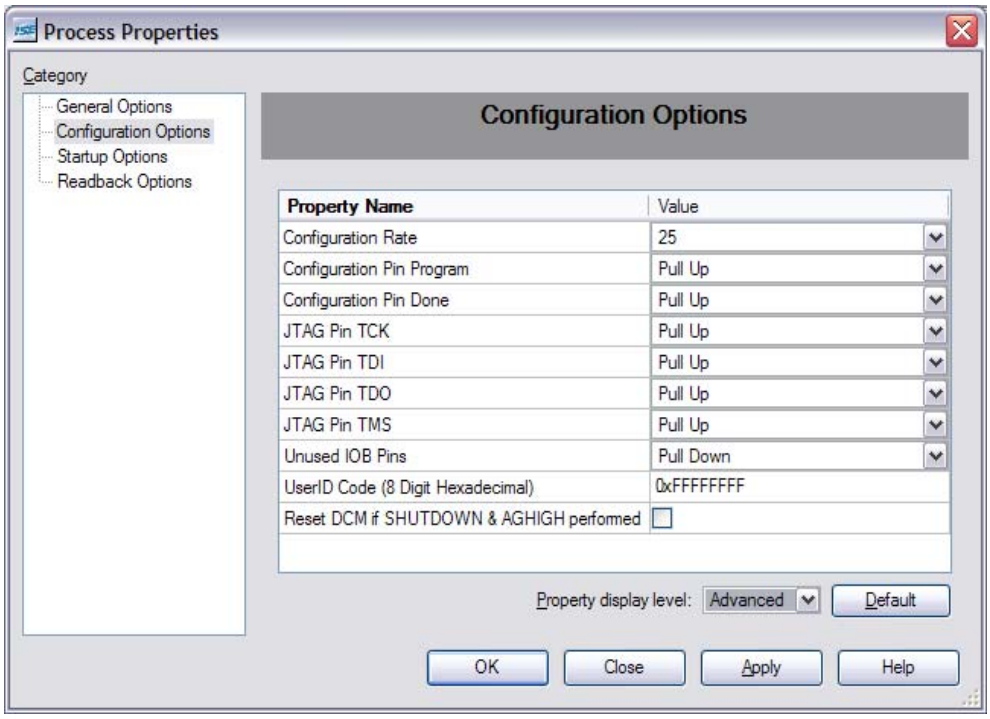


Figura 14: Process Properties

Después de generar el archivo de programación, se hace doble clic en Manage Configuration Project para ejecutar el software iMPACT. Se inicia la ventana de la Figura 15, seleccionando Prepare a PROM File y se haciendo clic en *Finish*.

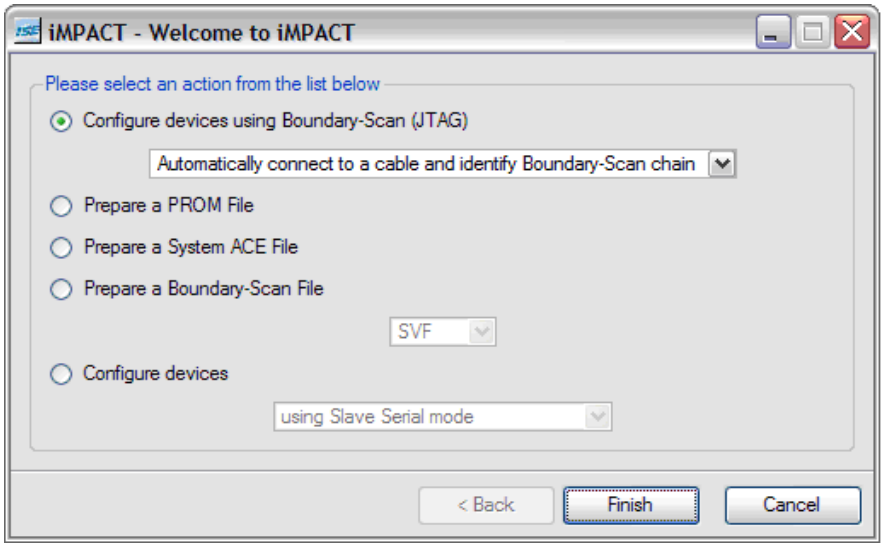


Figura 15: iMPACT Wizard

Se mostrara una ventana de opciones como en la Figura 16, si no sucede automáticamente en iMPACT se hace doble clic en PROM File Formatter, por medio de ésta se seleccionan las opciones de archivo para las diferentes formas de configuración, tanto dispositivos, densidad de estos y todas las otras opciones que sean necesarias establecer.

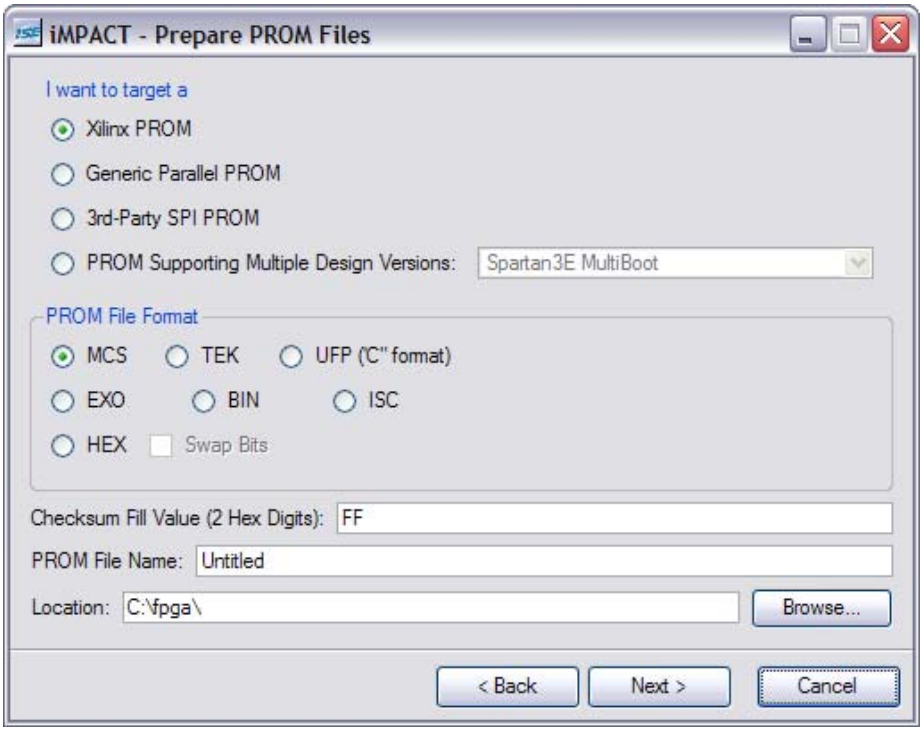


Figura 16: iMPACT – Prepare PROM Files

Para la Flash Platform PROM, seleccione Xilinx PROM, el formato de archivo MCS, indique la ubicación donde se desea guardar y el nombre que llevara el archivo, luego haga clic en Next, para continuar con la preparación. Continúe seleccionando la familia de la PROM, xcf, y el correspondiente dispositivo, xcf04s, se agrega a la lista haciendo clic en Add, y luego en Next, Figura 17. Verifique las opciones de la generación de archivo y haga clic en *Finish*, Figura 18.

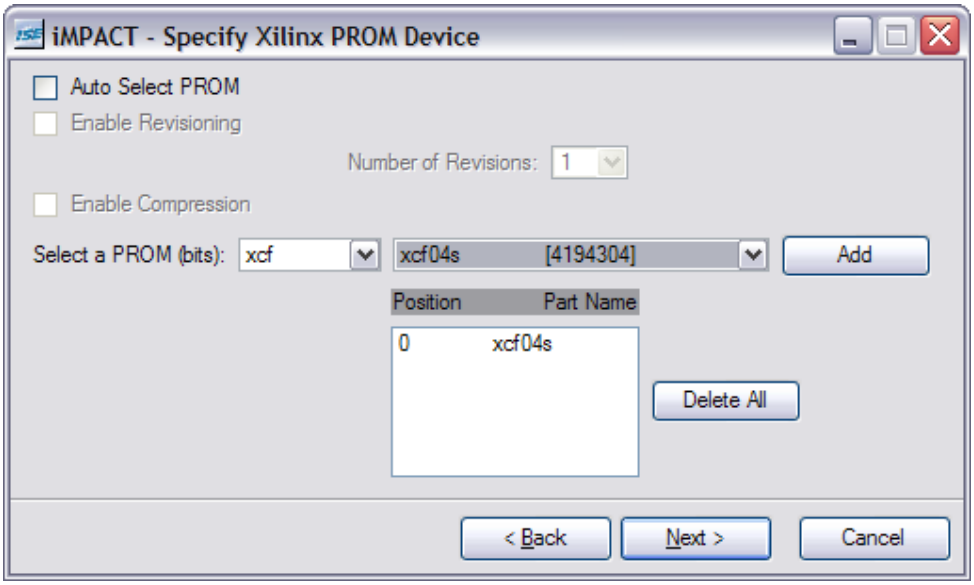


Figura 17: iMPACT – Specify Xilinx PROM Device

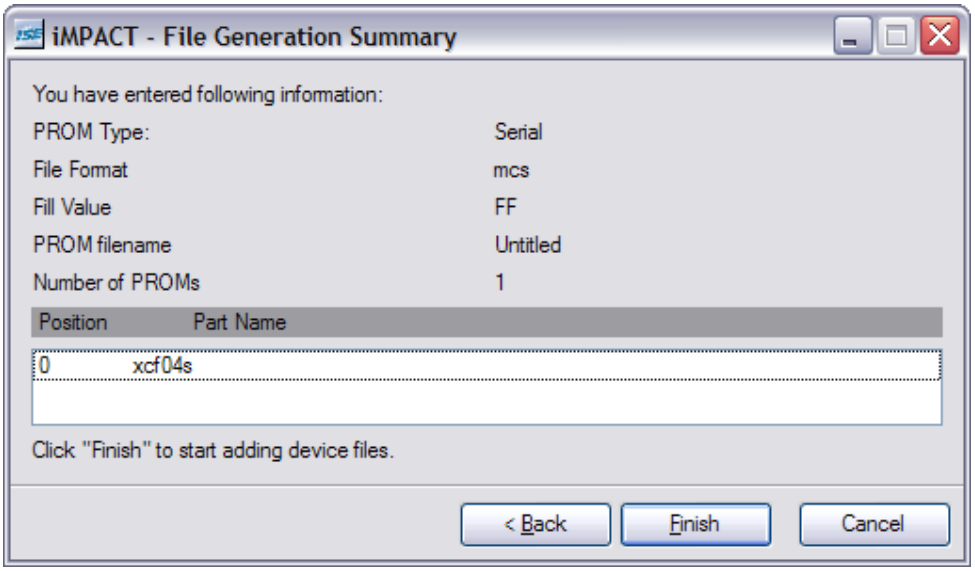


Figura 18: iMPACT – File Generation Summary

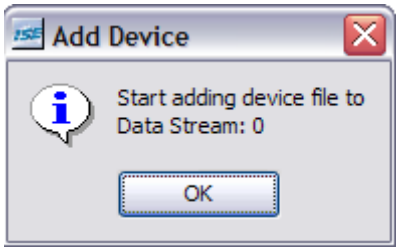


Figura 19: iMPACT – Add Device

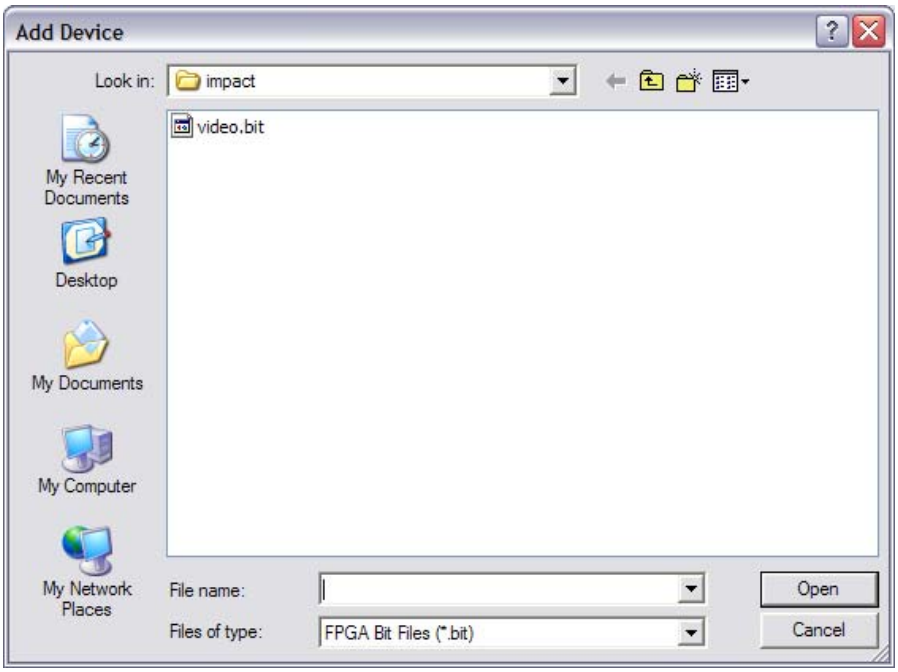


Figura 20: iMPACT – Add Device

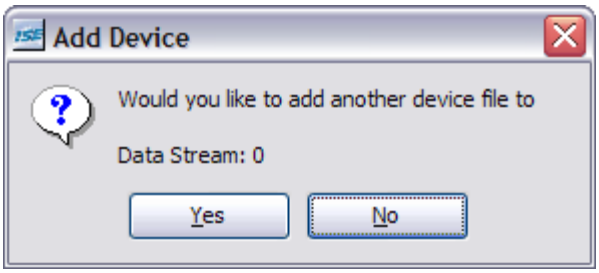


Figura 21: iMPACT – Add Device

Se inicia el agregar los dispositivos para el archivo de configuración, solo es necesario agregar un dispositivo, clic en *OK*, Figura 19; luego seleccione el archivo .bit de la configuración de la FPGA que se cargar en la Flash Platform PROM, Figura 20. En la ventana de pregunta, Figura 21, se rechaza agregar otro archivo de dispositivo. La configuración ya esta completa, ahora se genera el archivo de configuración ejecutando Operations/Generate File; el archivo para programar la memoria será creado y guardado en la ubicación que se indico.

El siguiente paso es descargar el archivo a la XCF04S, que se realiza siguiendo los pasos descritos en la configuración de un dispositivo usando USB-JTAG.

Configuración SPI

Para la configuración SPI es necesario programar la memoria SPI Serial Flash. La programación de la memoria SPI serial Flash puede realizarse de varias formas:

- Usando el software XSPI, por medio de un cable de programación JTGA paralelo.
- Usando la aplicación de los diseños de referencia basada en el microcontrolador PicoBlaze. Descargar la programación a través del puerto serie de la estación de trabajo a la FPGA, y el microcontrolador realiza la programación de la memoria SPI serial Flash., como se describe en este documento.

Para la configuración de la memoria SPI Serial Flash usando SPI FLASH Programmer (diseñada por Ken Chapman) se hace necesario utilizar el archivo de configuración adecuado para la programación del dispositivo, que se genera con el uso de la herramienta iMPACT.

Generación del archivo de configuración:

El archivo de configuración de la FPGA que se usa para generar la programación de la memoria SPI Serial Flash es creado con las siguientes opciones de configuración.

ConfigRate: CCLK Frequency 12
StartupClk: CCLK
DriveDone: Activar Drive Done Pin High

Estas opciones se asignan en las propiedades para generar el archivo de programación, Figura 14. Haga clic derecho en Generator Programming File y seleccione Properties con clic izquierdo. En las categorías de Process Properties seleccione:
Configuration Options y establezca Configuration Rate a 12
Setup Options, seleccione CCLK en la lista FPGA Start-Up Clock y active la casilla de Driver Done Pin High
Genere el archivo de configuración haciendo doble clic en Generator Programming File.

Convertir el archivo de configuración:

Después de generar el archivo de configuración en el Project Navigator, hacer doble clic en Generate PROM, ACE or JTAG File para ejecutar el software iMPACT. En la ventana de la Figura 22, seleccione Prepare a PROM File y haga clic den *Finish.*

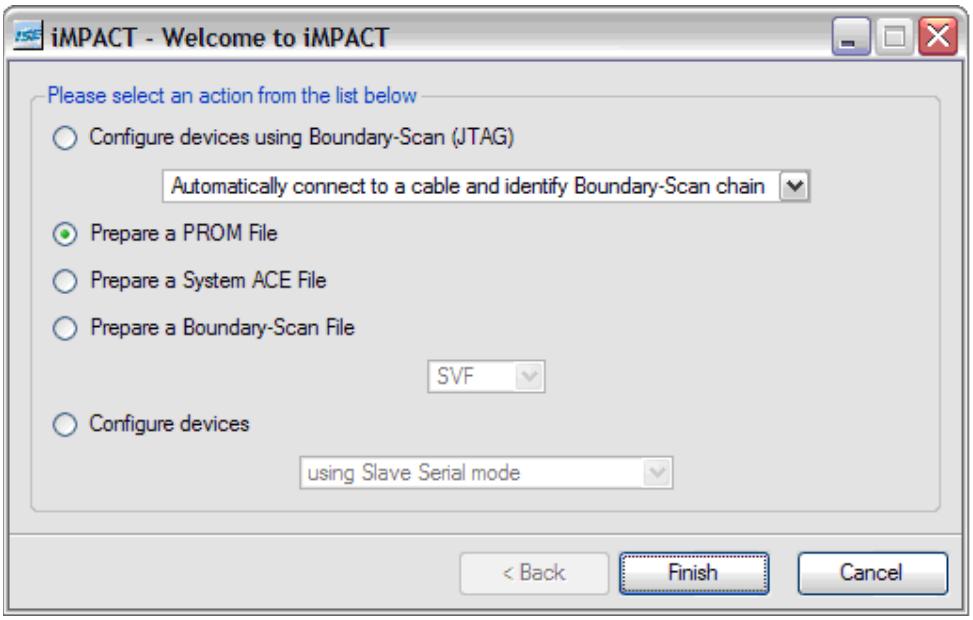


Figura 22: iMPACT – Welcome to iMPACT

Seleccione 3rd-Party SPI PROM, el formato MCS e indique la ubicación donde se desea guardar y el nombre que llevara el archivo como se muestra en la Figura 23.

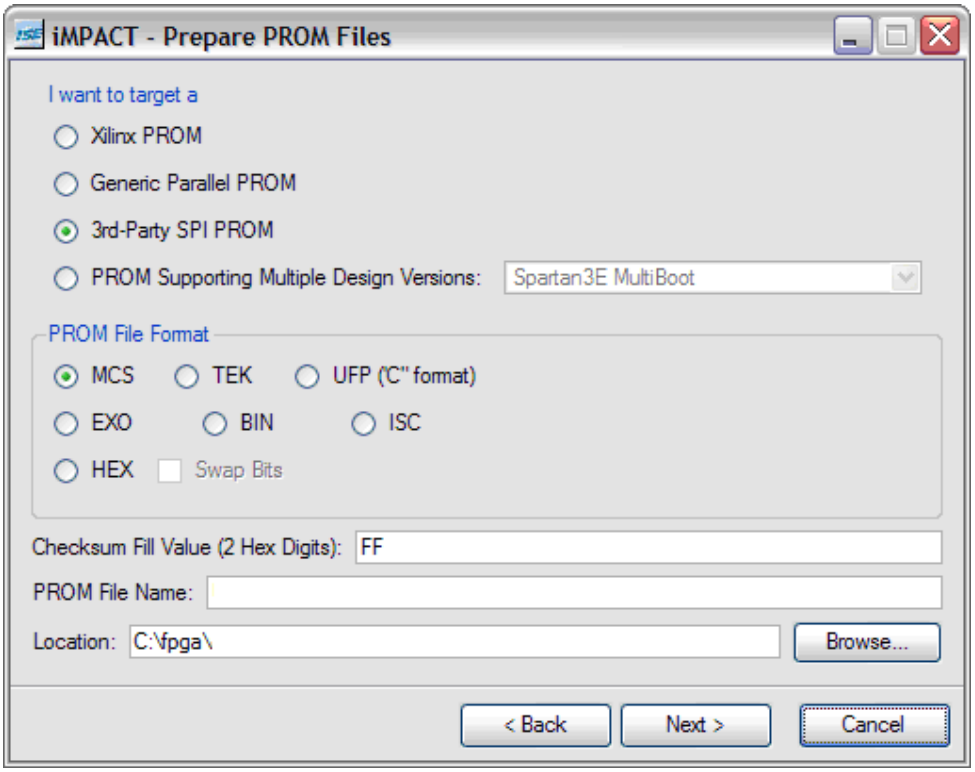


Figura 23: iMPACT – Prepare PROM Files

Continúe seleccionando la densidad de la memoria SPI PROM 128 M bits, Figura 24.

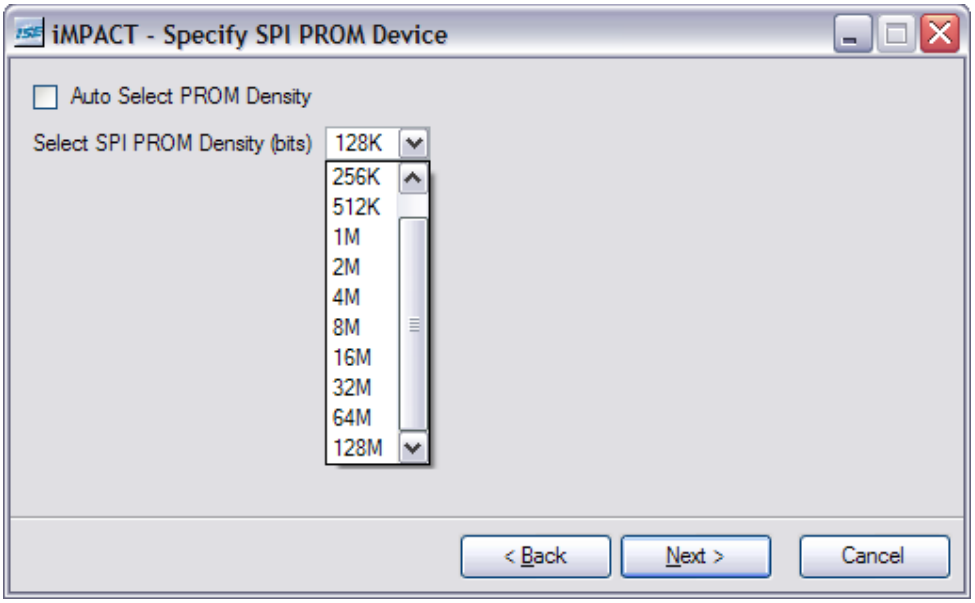


Figura 24: iMPACT – Specify SPI PROM Device

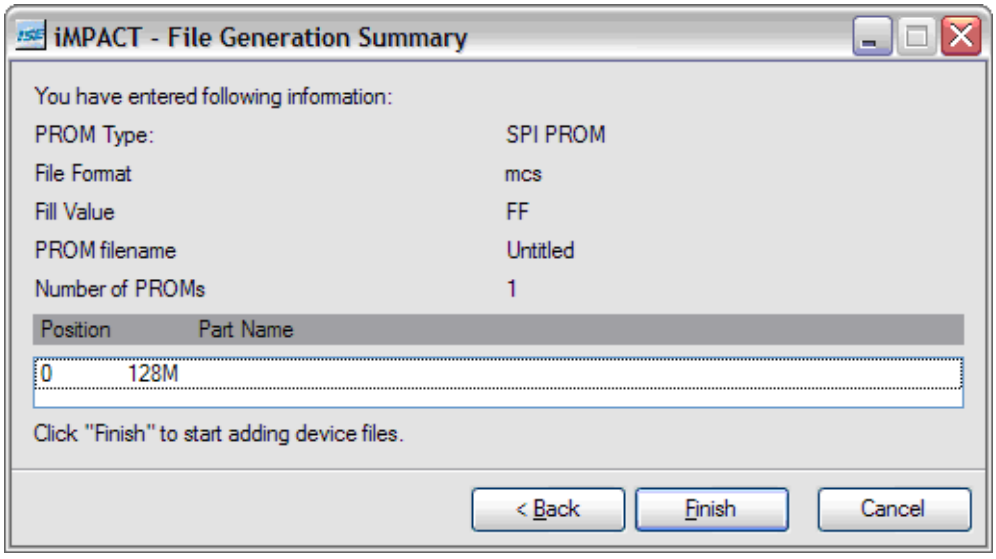


Figura 25: iMPACT – File Generation Summary

Verifique las opciones de la generación de archivo y haga clic en *Finish*, Figura 25. Agregue el archivo del dispositivo, *OK* en ventana de la Figura 26, luego seleccione el archivo .bit de la configuración de la FPGA que se cargar en la memoria SPI Serial Flash Figura 27. Para completar genere el archivo de configuración ejecutando Operations/Generate File. El archivo para programar la memoria con la configuración de la FPGA es creado y guardado en la ubicación que se indicó.

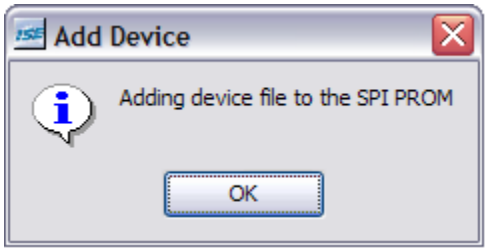


Figura 26: iMPACT - Add Device

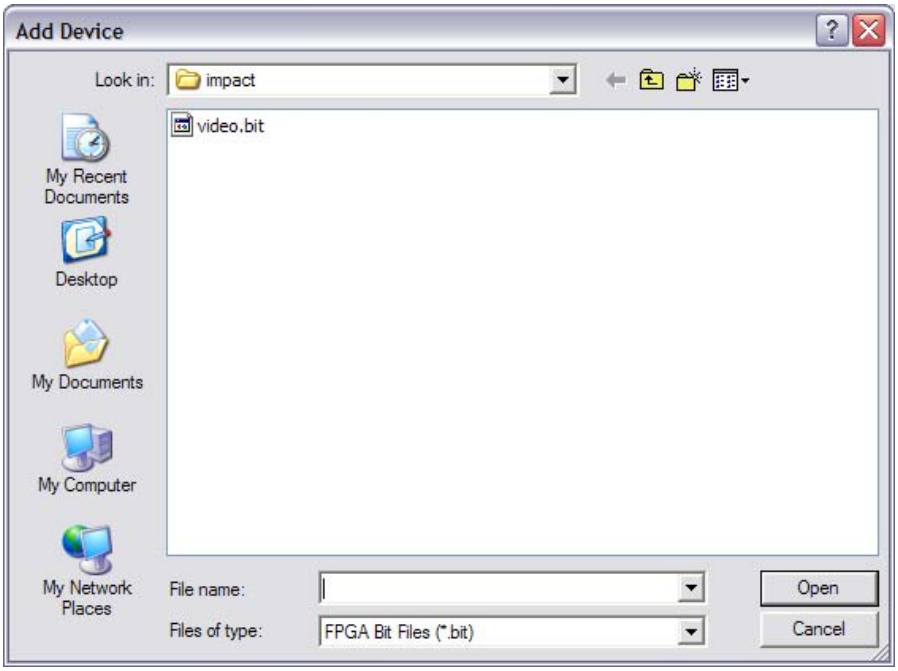


Figura 27: iMPACT – Add Device

Para programar la memoria SPI Serial Flash se usa algún software para comunicación serial RS-232 para enviar el archivo a la FPGA, y ésta a la memoria. En el caso del los PCs puede usarse el HiperTerminal, la configuración del puerto serial debe ser la siguiente:

Bits per second : 115200
Data bits: 8
Parity: None
Stop bits: 1
Flow control: **XON/XOFF**

Establecer el software de comunicación en Conectar (HiperTermianl en Llamar) y cargar la FPGA con la configuración contenida en s3esk_spi_flash_programmer.zip, que es la encargada de leer y escribir la memoria SPI Serial Flash. Puede realizar usando el modo USB-JTAG o el SPI MASTER. El software de comunicación mostrara en el prompt, el mensaje de inicio de la aplicación y las opciones del menú, Figura 28.

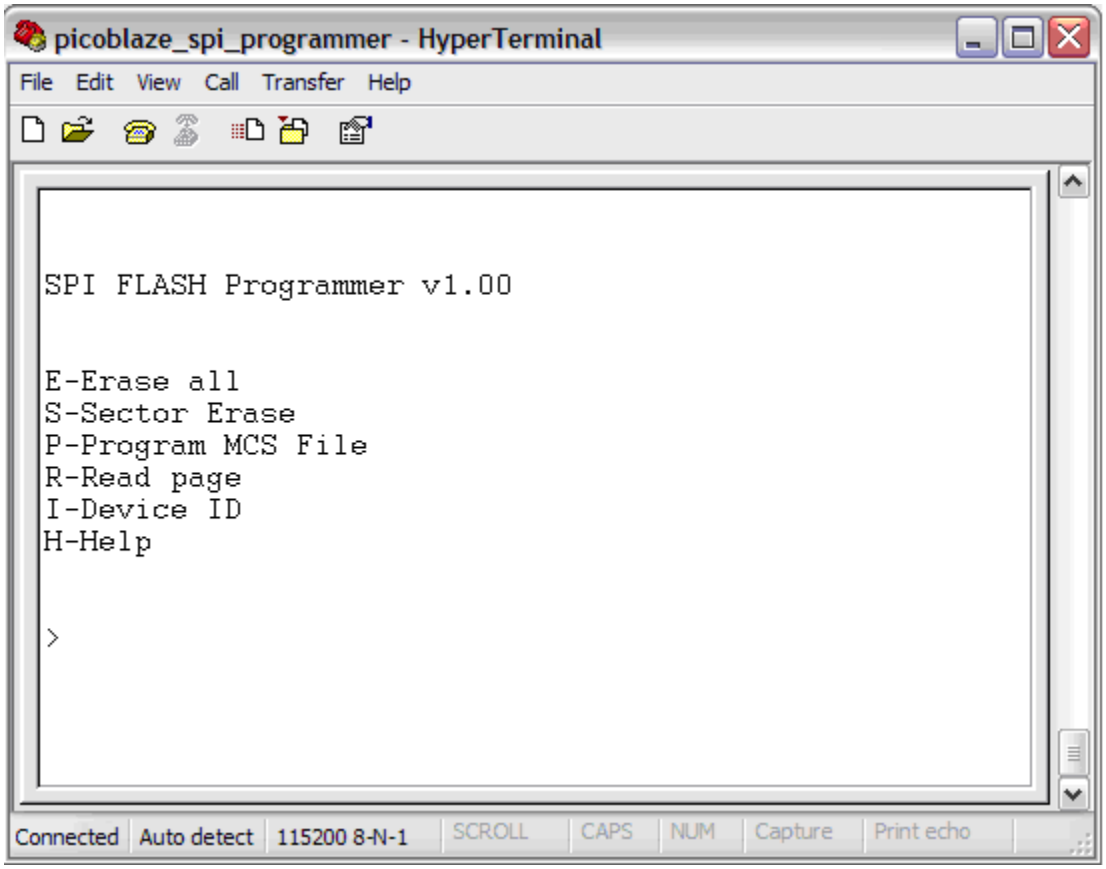


Figura 28: Interfaz de comunicación entre la FPGA y el PC.

El SPI FLASH Programmer, dispone de los siguientes comandos:

- E-Erase all, realiza el borrado de todo el dispositivo SPI FLASH.
- S-Sector Erase, borra los sectores del 0 al 5 de la memoria SPI FLASH. En estos sectores se encuentra el archivo de configuración de la FPGA.
- P-Program MCS File, es el comando que permite programar la memoria SPI FLASH con el archivo de configuración para la FPGA, para ser cargado al realizar el encendido de la board o al presionar el botón PROG.
- R-Read page, lee un bloque de memoria de 256 bytes.
- I-Device ID, Lee el código de identificación de la memoria SPI FLASH M25P16, se puede usar para verificar el funcionamiento de la comunicación entre la FPGA y el dispositivo SPI FLASH.

Programación de la memoria SPI FLASH:

Para una optima programación de la memoria y lograr la configuración de la FPGA mediante el modo SPI, es necesario borrar el contenido de la memoria, o al menos el contenido de la memoria encargado de configurar la FPGA, puede usarse el comando E o el S. Enseguida se hace uso del comando P para enviar la programación de la memoria. Este se inicia en el modo de espera, para seleccionar el archivo de programación creado

anteriormente. El envío se realiza usando la herramienta *Transferir* del HiperTerminal, mediante la opción *Enviar archivo de Texto*, se selecciona el archivo .mcs de la ubicación indicada al generar este archivo.
El proceso de transferencia de archivo usando un puerto Serial puede tomar varios minutos. 80 segundos. Sin embargo, el uso de con convertidor USB-Serial en conjunto con el software HiperTerminal puede alcanzar hasta los 30 minutos, dependiendo de la configuración y velocidad que dispositivo puede llegar a alcanzar. Con el fin evitar este inconveniente se recomienda el uso del software de comunicación Serial Hercules.

Configuración BPI

Para realizar la configuración de la memoria SPI Serial Flash usando SPI FLASH Programmer (diseñada por Ken Chapman). se hace necesario crear el archivo de configuración adecuado para la programación del dispositivo, que se genera con el uso de la herramienta iMPACT.

En el modo de configuración BPI, se realiza previamente la programación de la memoria Parallel Nor Flash usando la aplicación de los diseños de referencia Nor Flash Programmer de Ken Chapman. Se descarga la programación a través del puerto serie de la estación de trabajo a la FPGA, y ésta realiza la programación de la memoria Parallel Nor Flash.

El archivo para la programación del dispositivo debe contener las siguientes opciones de configuración:

ConfigRate: Default
StartupClk: CCLK
DriveDone: Desactivar Drive Done Pin High

Estas opciones se asignan en las propiedades para generar el archivo de programación, Figura 14. Haga clic derecho en Generator Programming File y seleccione Properties con clic izquierdo y en las categorías de Process Properties seleccione:

Configuration Options y establezca Configuration Rate a 12
Setup Options, seleccione CCLK en la lista FPGA Start-Up Clock y active la casilla de Driver Done Pin High
Genere el archivo de configuración haciendo doble clic en Generator Programming File, después de generar el archivo de configuración en el Project Navigator, se hace doble clic en Generate PROM, ACE or JTAG File para ejecutar el software iMPACT.

Al abrirse la ventana de la Figura 29, seleccione Prepare a PROM File y haga clic en *Finish*.

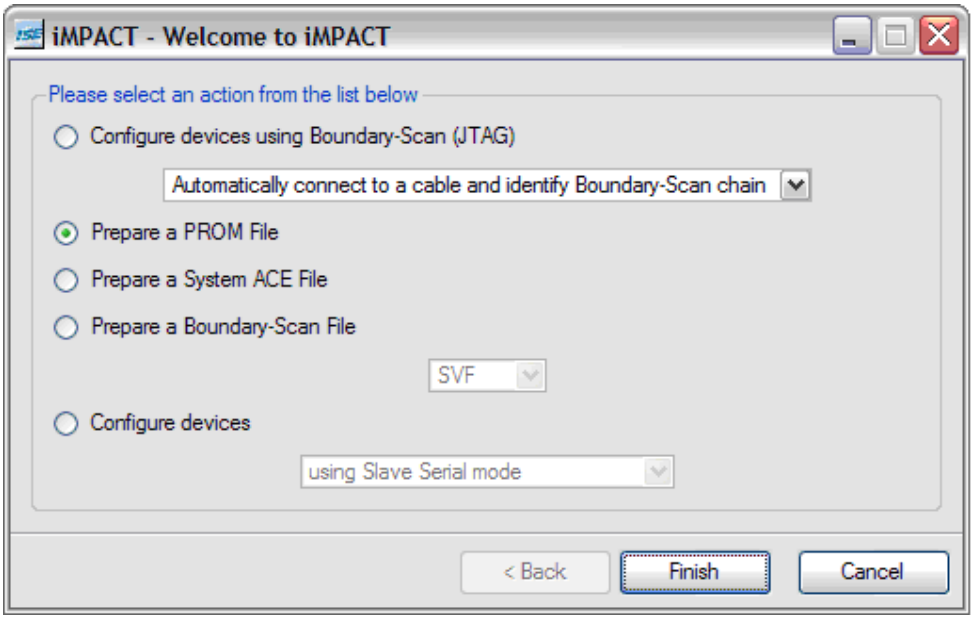


Figura 29: iMPACT – Welcome to iMPACT

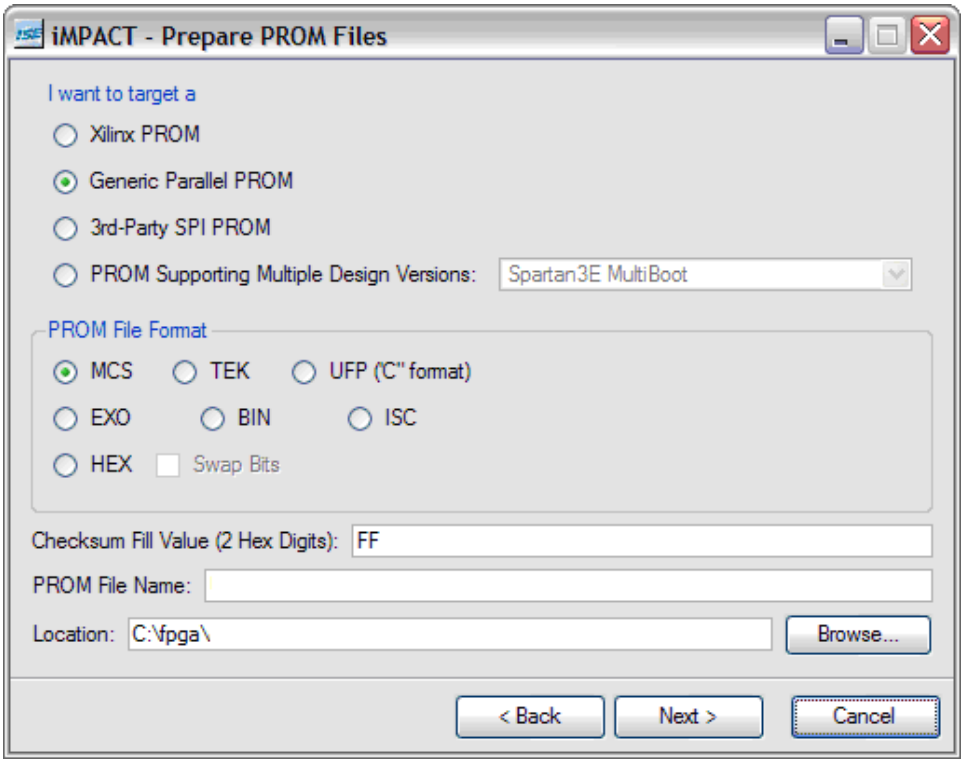


Figura 30: iMPACT – Prepare PROM Files

Seleccione Generic Parallel PROM, el formato MCS y se indica la ubicación donde se guarda y el nombre que llevará el archivo, como se muestra en la Figura 30, enseguida haga clic en *Next*.

Se continúa seleccionando la densidad de la memoria Parallel PROM 16 M, se agrega haciendo clic en Add. Se activa la casilla *Create BPI-Mode PROM* y se selecciona la

Familia *Spartan 3E*. Por ultimo se escoge la dirección del modo BPI, Up o Down como se observa en la Figura 31.

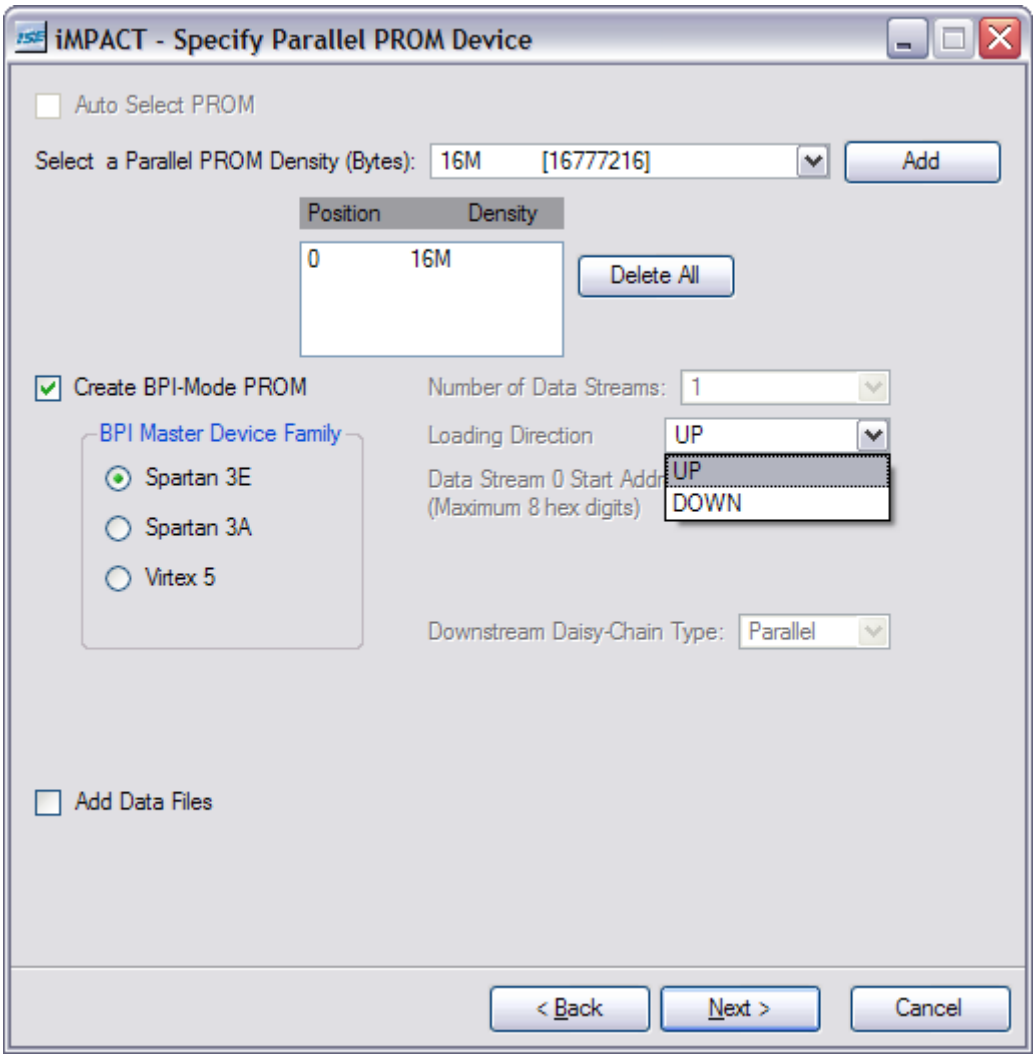


Figura 31: iMPACT – Specify Parallel PROM Device

Luego es posible verificar las opciones de la generación de archivo y luego se debe hacer clic en *Finish*, Figura 32.

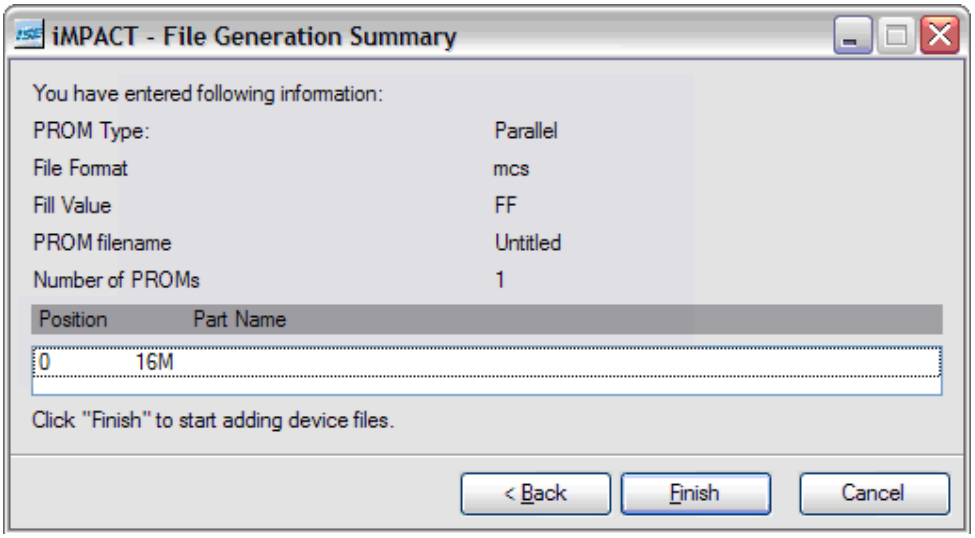


Figura 32: iMPACT – File Generation Summary

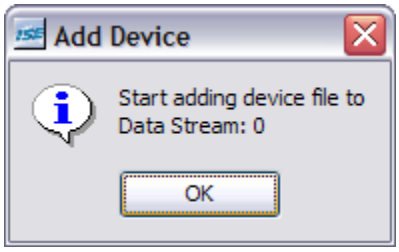


Figura 33: iMPACT - Add Device

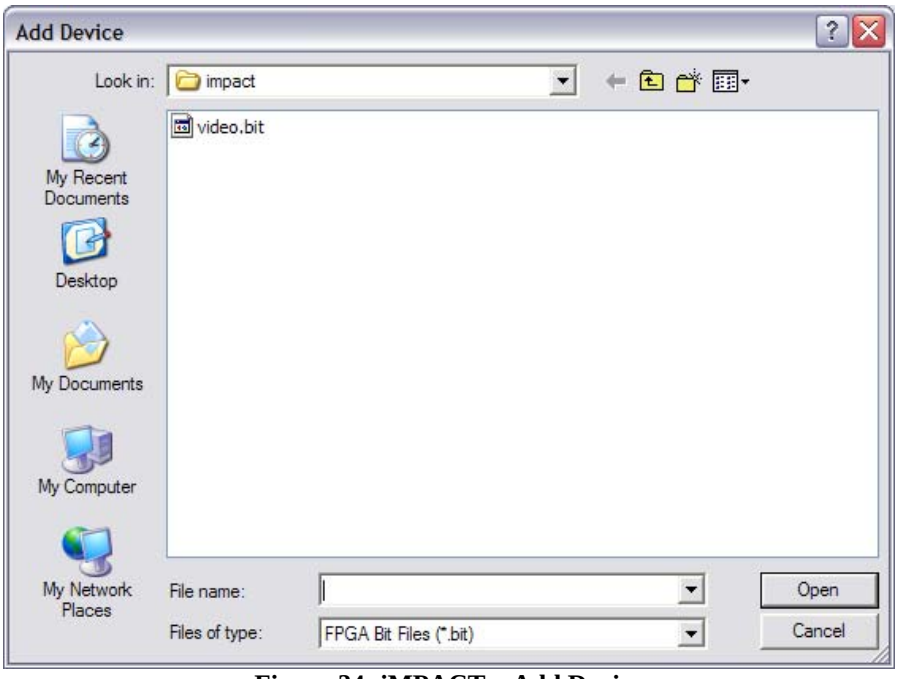


Figura 34: iMPACT – Add Device

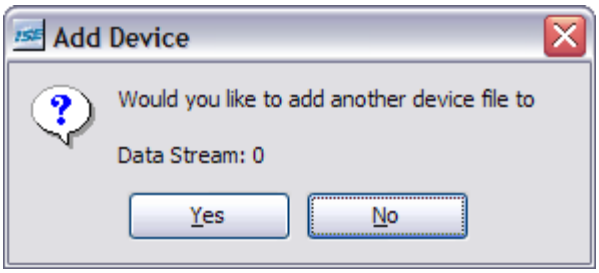


Figura 35: iMPACT – Add Device

Posteriormente se agregan los dispositivos para el archivo de configuración, se hace clic en *OK*, Figura 33. Luego se selecciona el archivo .bit de la configuración de la FPGA que se carga en la memoria Paralell Nor Flash, Figura 34. En este paso se debe rechazar agregar otro archivo de dispositivo, Figura 35. Finalmente, genere el archivo de configuración ejecutando Operations/Generate File; el archivo para programar la memoria será creado y guardado en la ubicación que se indico.

Seguido esto, se usa el software de comunicación para serial RS-232. En el caso de los PCs puede usarse el HiperTerminal, la configuración del puerto serial debe ser la siguiente:

Bits per second : 115200
Data bits: 8
Parity: None
Stop bits: 1
Flow control: **XON/XOFF**

Se inicia el software de comunicación en Conectar (HiperTermianl en Llamar) y se configura la FPGA con la aplicación encargada de controlarla y modificarla, que se encuentra en el archivo s3esk_parallel_nor_programmer.zip.

Al terminar la configuración de la FPGA se inicia la comunicación, permitiendo enviar comandos y recibir o enviar datos entre el PC y la memoria Parallel Nor Flash.

El software de comunicación mostrara en el prompt, el mensaje de inicio de la aplicación y las opciones del menú, Figura 36.

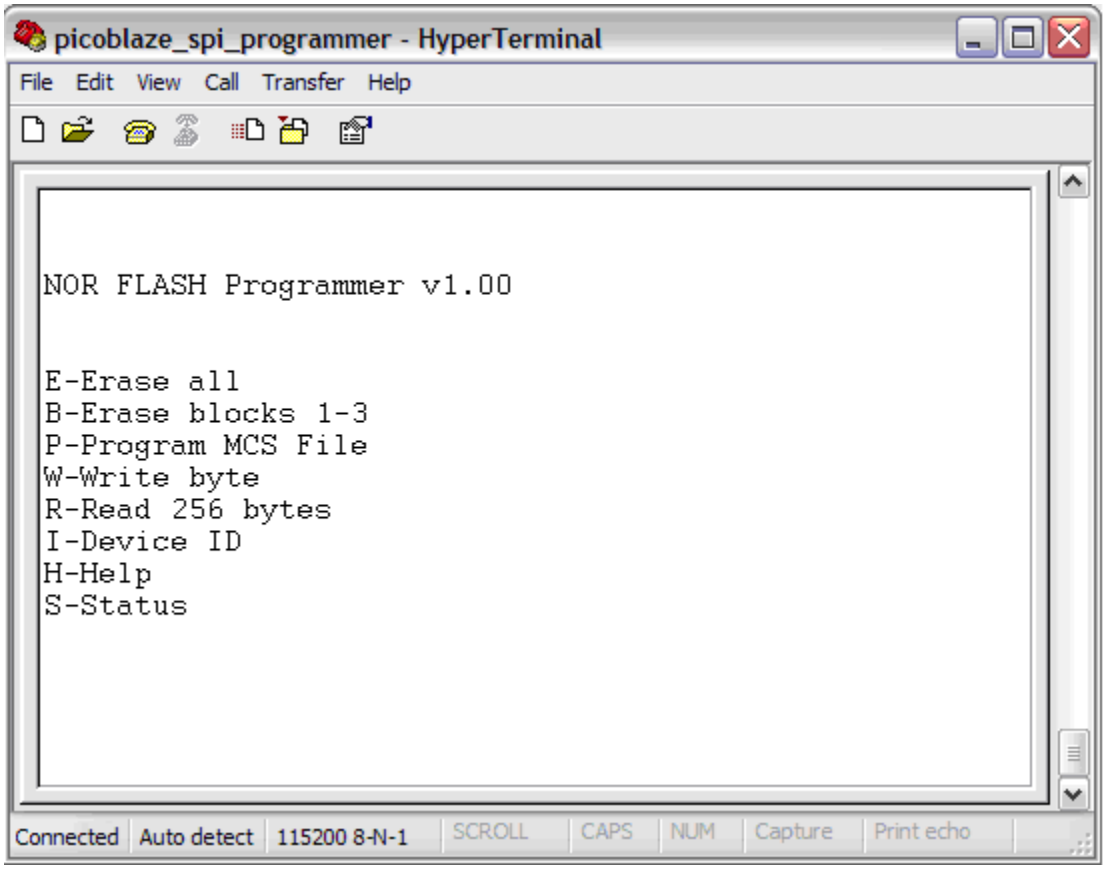


Figura 36: Interfaz de comunicación entre la FPGA y el PC.

El Nor Flash Programmer, permite acceder a las posiciones de la memoria StrataFlash Paralell Nor Flash, usando los diferentes comandos implementados en la FPGA.

El NOR FLASH Programmer, dispone de los siguientes comandos:

E - Erase all, realiza el borrado de todo el dispositivo Paralell Nor Flash.
S - Erase blocks 1-3, borra los sectores del 0 al 3 de la memoria Parallel Nor Flash. En estos sectores se encuentra el archivo de configuración BPI UP de la FPGA.
P - Program MCS File, es el comando que permite programar la memoria SPI FLASH con el archivo de configuración para la FPGA, para ser cargado al realizar el encendido de la board o al presionar el botón PROG.
W – Write byte. Permite escribir un byte en cualquier dirección de la memoria.
R - Read 256 bytes, lee un bloque de memoria de 256 bytes.
I-Device ID. Lee el código de identificación de la memoria Intel StrataFlash, se puede usar para verificar el funcionamiento de la comunicación entre la FPGA y el dispositivo SPI FLASH.
S – Status, Lee el registro de estado de la memoria Intel StrataFlash.

Programación la memoria Parallel Nor Flash:

Para una optima programación de la memoria y lograr la configuración de la FPGA usando los modos BPI, es necesario borrar el contenido de la memoria, o al menos el contenido de la memoria encargado de configurar la FPGA, puede usarse el comando E o el B. Seguido, el uso del comando P para enviara la programación de la memoria, se inicia el modo de espera para seleccionar el archivo de programación creado anteriormente. El envío se realiza usando la herramienta *Transferir* del HiperTerminal, mediante la opción *Enviar archivo de Texto*, se selecciona el archivo .mcs de la ubicación indicada al generar este archivo.

El proceso de transferencia de archivo usando un puerto Serial puede tomar varios minutos. 80 segundos. Sin embargo, el uso de con convertidor USB-Serial en conjunto con el software HiperTerminal puede alcanzar hasta los 30 minutos, dependiendo de la configuración y velocidad que dispositivo puede llegar a alcanzar. Con el fin evitar este inconveniente se recomienda el uso del software de comunicación Serial Hercules.

ANEXO I

PicoBlaze, Guía en Español

ÍNDICE DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

El Constant (k) Coded Programmable State Machine (KCPSM3) es un microcontrolador RISC embebido de 8 bits para FPGAs de Xilinx. Es un microcontrolador compacto, de una buena relación costo-eficiencia y puede ser usado en procesamiento complejo de datos que no requiera tiempos críticos altos de ejecución.

Es totalmente embebido, como un modulo VHDL en una FPGA y no requiere componentes externos. Es altamente flexible y de fácil funcionamiento. Su funcionamiento puede ampliado usando la lógica de la FPGA, conectándose a otros módulos VHDL, ya sean Verilog, VHD o IP CORE, o recursos de la FPGA. También, usando la lógica de salida de la FPGA para interactuar con otros dispositivos.

Algunos textos e imágenes en el presente documento son una versión realizada y adaptada al español de los documentos PicoBlaze 8-Bit Microcontroller for Virtex-E and Spartan-II/IIE Devices por Ken Chapman y UG129 Xilinx PicoBlaze 8-bit Embedded Microcontroller User Guide, como documentación resultante del proyecto PROPUESTA DE PRACTICAS DE LABORATORIO PARA SISTEMAS EMBEBIDOS BASADOS EN SPARTAN 3E STARTER KIT DE XILINIX

ARQUITECTURA DEL MICROCONTROLADOR

El PicoBlaze esta optimizando para eficiencia y bajos costos de desarrollo. Ocupa 96 Fpga Slices, 2% de la FPGA XC3S500E. En una aplicación típica un solo bloque RAM de la FPGA almacena 1024 bits de instrucciones de programa. Puede llevar a cabo de 44 a 100 instrucciones por segundo dependiendo del dispositivo de la Familia FPGA y grado de velocidad.

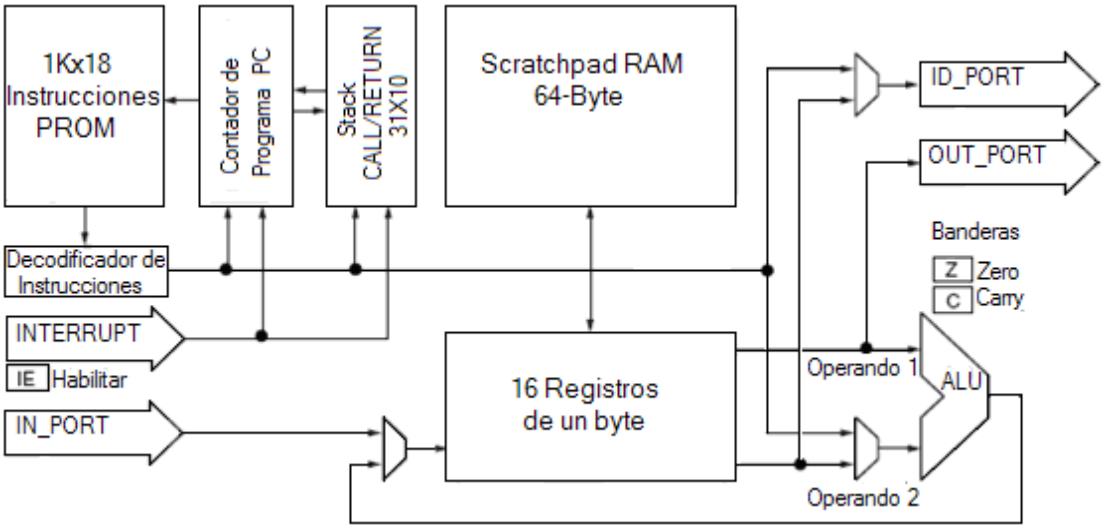


Figura 1: Arquitectura PicoBlaze

CARACTERÍSTICAS

- 1Kbyte de programación para almacenar en la memoria, automáticamente cargada durante la configuración de la FPGA
- 16 Registros para Propósito General de un byte
- Unidad Lógico Aritmética
- Banderas CARRY y ZERO
- Reset
- 256 puertos de entrada y 256 puertos de salida
- Interrupción (respuesta de 5 ciclos de reloj)
- Memoria RAM scratchpad [1]64-byte
- Comportamiento previsible, dos ciclos por instrucción.

[1] Memoria interna de alta velocidad usada para almacenar temporalmente información preliminar.

BLOQUES FUNCIONALES

Registros de Propósito General

El microcontrolador PicoBlaze incluye 16 registros de propósito general de un byte, llamados s0 a SF. Estos registros pueden ser renombrados usando una directiva de lenguaje ensamblador, que le permite mayor facilidad de manejo al programador. Todas las operaciones de los registros son completamente intercambiables, ningún registro es reservado para tareas especiales o tiene prioridad sobre algún otro registro. No hay un acumulador dedicado, cada resultado es computado en el registro especificado.

Memoria de Programa

El microcontrolador ejecuta hasta 1024 instrucciones de la memoria de la FPGA, por lo general almacenadas en un solo bloque RAM. Cada Instrucción PicoBlaze tiene un ancho de 18bits. Las instrucciones son compiladas durante el diseño para la FPGA y son cargadas automáticamente durante el proceso de configuración de la FPGA.
Para requerimientos de espacio para grandes programas son típicamente diseccionados usando múltiples microcontroladores cada uno asociado a un bloque de memoria para distribuir en varias tareas el sistema.

Unidad Lógico Aritmética

La Unidad Lógico Aritmética (ALU) realiza todos los cálculos del microcontrolador, incluyendo:
Operaciones de aritmética básica (suma y resta), tiene la opción de incluir la bandera CARRY como una entra para dar soporte a operaciones que requieren más de 8 bits.
Operaciones lógicas (AND, OR y XOR)
Comparación aritmética y test de operaciones entre bits
Operaciones de rotación y desplazamiento

Todas las operaciones son realizadas usando un operador indicado por algún registro específico (sX) y el resultado es devuelto al mismo registro. Si una instrucción requiere un segundo operando, entonces el segundo operando es un segundo registro (SY) o una de constante inmediata de 8 bits (kk).

Banderas

Las operaciones de la ALU afectan las banderas de cero (ZERO) y acarreo (CARRY). La bandera ZERO indica que el resultado de la última operación fue cero.

La bandera CARRY indica varias condiciones dependiendo de la instrucción ejecutada. Es establecida cuando en el desborde de operaciones aritméticas; también, es usada para capturar el bit movido afuera del registro durante una instrucción desplazamiento o rotación. Durante la instrucción TEST, la bandera CARRY es usada para indicar si los 8 bits resultantes temporalmente tienen paridad impar.

El estado de las banderas puede ser usado para determinar ejecuciones de secuencias del programa, usando condicionales o no-condicionales en el control de flujo de programa.

La bandera de interrupción INTERRUPT_ENABLE habilita la entrada de interrupciones.

Memoria ScratchPad

El microcontrolador PicoBlaze suministra una memoria RAM de propósito general de 64-byte, direccionada directa o indirectamente desde el registro usando las instrucciones STORE y FETCH.

La instrucción STORE escribe el contenido de cualquiera de los registros a cualquiera de las 64 posiciones de RAM. La instrucción FETCH lee cualquiera de las 64 posiciones de la memoria y lo escribe a cualquiera de los 16 registros. Esto permite gran número de variables dentro del procesador y reservar todos los espacios I/O para señales de entradas y salidas reales.
El direccionamiento es directo con una constante inmediata (ss), o indirecto usando el contenido de alguno de los 16 registros (sY).

Entradas / Salidas

Los puertos de entrada y salida extienden las capacidades del microcontrolador PicoBlaze y permite al microcontrolador conectarse a periféricos o a otras FPGA. El microcontrolador soporta hasta 256 puertos de entrada y 256 puertos de salida o una combinación de puertos entrada/salida.
La salida PORT_ID suministra la dirección del puerto. Durante una operación de entrada, el microcontrolador lee el dato desde el puerto IN_PORT a un registro específico, sX. Una operación de lectura es indicada por un pulso en READ_STROBE. El uso de esta señal en un diseño no es siempre vital, indica que el dato fue adquirido por el microcontrolador. Durante la operación de salida, el microcontrolador escribe el contenido de un registro específico, sX, al puerto OUT_PORT. Un pulso en WRITE_STROBE indica la operación de salida. Este señal es usada en los diseños en las interfaz lógicas de salida para asegurar que solo el dato valido pasa a otro sistema.

Contador de Programa

El contador de programa (PC) apunta a la siguiente instrucción a ejecutar. El PC se incrementa predeterminadamente a la siguiente ubicación de la instrucción cuando se ejecuta una instrucción. Solo las instrucciones JUMP, CALL, retorno (RETURN o RET) y retorno de interrupción (RETURNI o RETI) y los eventos de interrupción y reset modifican el comportamiento predeterminado. El PC no puede ser modificado directamente por el código de la aplicación.
Los 10 bits de PC soportan un máximo espacio de código de 1024 instrucciones, desde 000 a 3FF. Si PC alcanza el máximo de la memoria 3FF, este regresa a la ubicación 000.

Pila CALL/RETURN

La pila "CALL/RETURN" almacena hasta 31 direcciones de instrucción, lo que permite anidar un "CALL" de secuencias de hasta 31 niveles de profundidad. La pila se utiliza también durante la operación de una interrupción, al menos uno de estos niveles deben reservarse cuando la interrupción se encuentra activa

Dado que la pila también se utiliza durante una operación de interrupción, por lo menos uno de estos niveles deben ser reservados interrumpe cuando están activados.

Interrupción

El microcontrolador PicoBlaze tiene una entrada INTERRUPT. Puede usarse en combinación con otras señales si se requiere. Permite al microcontrolador manejar eventos externos asíncronos. En la práctica se recomienda sincronizar todas las entradas del PicoBlaze, incluyendo la interrupción.

Una interrupción fuerza al microcontrolador a llamar la ultima posición de la memoria de programa, “CALL 3FF”, donde el usuario define la acción a seguir. Al realizar la interrupción se almacena el estado de las banderas y se deshabilita cualquier otra interrupción. El comando de retorno de interrupción (RETURNI o RET) se asegura que al final de un ISR[2] se restaure el estado de las banderas y se habilite el control de interrupción.

Reset

Automáticamente después de completar el proceso de configuración de la FPGA el microcontrolador PicoBlaze se resetea. La entrada RESET fuera al microcontrolador al estado inicial. El PC es puesto en la dirección 0, las banderas limpiadas, interrupciones deshabilitadas, y la pila CALL/RETURN es reseteada.

[2] Interruption Service Routine

Los registros y la memoria Scratchpad no son afectadas por Reset.

Según Ken Chapman creador del modulo microcontrolador PicoBlaze, es una maquina de estados de basada en constantes. Así:

- Valor de datos constantes para usar en una operación ALU.
- Dirección de puerto constante para tener acceso a una información o control de la lógica externa del módulo de PicoBlaze.
- Valor de dirección constante para controlar la ejecución secuencial del programa

Todas las instrucciones bajo cualquier condición de operación se ejecutan en dos ciclos de reloj. El poder determinar el tiempo de ejecución de un programa de forma tan sencilla es de gran valor, sobre todo cuando es parte de una aplicación de tiempo real donde los tiempos de ejecución son calculados con gran exactitud.

SEÑALES Y CONEXIÓN DEL MICROCONTROLADOR PICOBLAZE

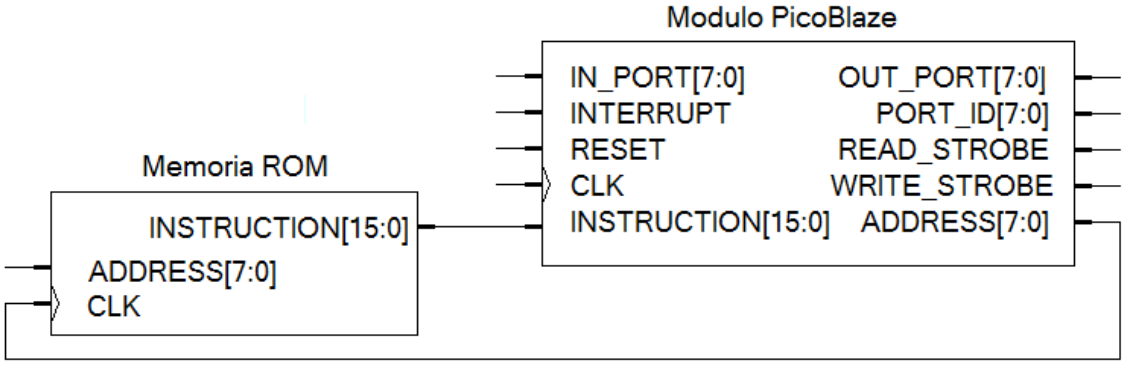


Figura 2: Diagrama de Bloques del PicoBlaze

IN_PORT[7:0] Puerto de entrada de datos. Presenta los datos de entrada validos en este puerto durante una instrucción de entrada. Los datos son capturados en el flanco de reloj de subida CLK.

INTERRUPT. Entrada de Interrupción. Si la bandera INTERRUPT_ENABLE es establecida para el programa, genera un evento INTERRUP. Si la bandera INTERRUP_ENABLE es limpiada, esta entrada es ignorada.

RESET. Entrada Reset. Para resetear el microcontrolador y genera un evento RESET. Un Reset es realizado automáticamente seguido a la configuración de la FPGA.

CLK. Entrada de Reloj. La frecuencia puede ser desde DC a la frecuencia de operación máxima reportada por el software de desarrollo Xilinx ISE. Todos los elementos sincrónicos son sincronizados por el flanco de subida de reloj. No hay requerimientos del ciclo de reloj más allá del mínimo ancho de pulso requerido por la FPGA.

OUT_PORT[7:0]. Puerto de salida de datos. Los datos de salida aparecen en este Puerto por dos ciclos de reloj durante la instrucción de salida.

PORT_ID[7:0] puerto de salida de dirección. Las dirección del puerto de E / S que aparecen en este puerto por dos ciclos de CLK durante una instrucción de salida o entrada.

READ_STROBE Cuando se establece en alto esta señal indica que los datos de entrada en el puerto IN_PORT[7:0] fueron capturados a un registro de datos durante una instrucción IN. Esta señal se utiliza típicamente para reconocer las operaciones de lectura desde FIFOs.

WRITE_STROBE Cuando se establece en alto esta señal valida los datos de salida en el puerto OUT_PORT[7:0] durante una instrucción de salida. Se usa para la captura de los datos de salida dentro de la FPGA en el ciclo de subida de CLK cuando WRITE_STROBE esta en alto.

INTERRUPT_ACK Indica Interrupción. Al ponerse en alto, esta señal reconoce que se produjo un evento INTERRUPT. Esta señal se utiliza opcionalmente para borrar la fuente que produce la entrada de interrupción.

En un flujo de diseño es usado el macro PicoBlaze. El modulo es una descripción VHDL (KCPSM3.vhd o KCPSM3.v), que ha sido escrita para aplicación óptima y predecible en FPGAS Xilinx. El código VHDL no debe ser modificado bajo ninguna circunstancia.

El modulo KCPSM3 contiene los elementos del microcontrolador PicoBlaze, no incluye el almacenamiento de las instrucciones.
La declaración de los componentes del modulo vhd es la siguiente:

```
component KCPSM3
	port (
		address : out std_logic_vector( 9 downto 0);
		instruction : in std_logic_vector(17 downto 0);
		port_id : out std_logic_vector( 7 downto 0);
		write_strobe : out std_logic;
		out_port : out std_logic_vector( 7 downto 0);
		read_strobe : out std_logic;
		in_port : in std_logic_vector( 7 downto 0);
		interrupt : in std_logic;
		interrupt_ack : out std_logic;
		reset : in std_logic;
		clk : in std_logic
		);
	end component;
```

La lista instaciación de componentes KCPSM3:

```
processor: kcpsm3
	port map(
		address => address_signal,
		instruction => instruction_signal,
		port_id => port_id_signal,
		write_strobe => write_strobe_signal,
		out_port => out_port_signal,
		read_strobe => read_strobe_signal,
		in_port => in_port_signal,
		interrupt => interrupt_signal,
		interrupt_ack => interrupt_ack_signal,
		reset => reset_signal,
		clk => clk_signal
	);
```

El modulo PicoBlaze ROM se utiliza dentro del flujo de diseño VHDL. El software ensamblador del PicoBlaze genera un archivo VHDL en donde un bloque de memoria RAM y su contenido inicial esta definido. El archivo VHDL se usa en síntesis lógica y la simulación del microcontrolador.

La declaración de los componentes del modulo ROM es la siguiente:

```
component prog_rom
	port (
		address : in std_logic_vector( 9 downto 0);
		instruction : out std_logic_vector(17 downto 0);
		clk : in std_logic
	);
end component;
```

La lista instaciación de componentes ROM:

```
program: prog_rom
	port map(
		address => address_signal,
		instruction => instruction_signal,
		clk => clk_signal
	);
```

El nombre del modulo ROM en la declaración de componentes e instaciación es debido al nombre del archivo fuente del código ensamblador del PicoBlaze, el ensamblador genera el modulo ROM con el nombre del archivo fuente.

HERRAMIENTAS DE DESARROLLO

Existen tres principales herramientas de desarrollo en la creación de código para aplicaciones del microcontrolador PicoBlaze. Xilinx ofrece dos de estas herramientas, el KSCPM3 para el entorno DOS de Windows y el entorno de desarrollo Xilinx Generator para DSP que incluye el ensamblador PicoBlaze y un modelo de simulación para desarrollo de MathWorks MATLAB/Simulink.
Mediatronix ofrece el software pBlazIDE, una herramienta de desarrollo grafica que incluye un ensamblador y un completo simulador de set de instrucciones (ISS).
En el año 2005 Francesco Poderico publicó el compilador C Pccomp alpha para el microcontrolador PicoBlaze, si embargo no ha tenido una amplia acogida ni desarrollo.
También existen algunas otras herramientas de desarrollo para el entorno GNU/Linux como PacoBlaze.

KSCPM3

El KCPSM3 es un ensamblador, un archivo ejecutable DOS, que junto con tres archivos base, generan el modulo ROM VHDL que contiene las instrucciones para el microcontrolador PicoBlaze.
Los archivos fuente son escritos usando alguna herramienta de texto disponible en la estación de trabajo, y son guardados usando la extensión PSM, con un límite de ocho caracteres alfanuméricos.

Para ensamblar un archivo, se abre una ventana MS-DOS y se escribe:

```
KSCPM3 [nombre_archivo].psm
```

El ensamblador se detiene al encontrarse cualquier error. Un mensaje indica la razón del error y la línea que se analizaba cuando ocurrió. Es necesario solucionar el error y volver a ensamblar el archivo.
Para almacenar el reporte del ensamblador, la salida puede ser redirigida a un archivo de texto usando el operador redirección, el cual envía las salidas del comando a un archivo de texto que se especifique para abrir con un editor de texto.

```
KCPSM3 <nombre_archivo>.psm> salida_K.txt
```

El ensamblador KCPSM3 es sensible a mayúsculas, interpreta como operandos diferentes a una palabra escrita con mayúsculas de otra sin ellas.

KCPSM3 usa cuatro archivos para la creación del modulo ROM, un archivo de programa .psm y tres plantillas de archivo que describen la forma de inicializar un bloque de memoria RAM que contiene las instrucciones de programa.

El ensamblar un archivo .psm genera 15 archivos, en los que se encuentran los archivos de flujo de diseño, vhd o verilog, definición del contenido de la memoria, reporte de errores y una versión formateada del archivo fuente.

Los archivos ROM_form.vhd y ROM_form.v, suministran las plantillas para la generación de los archivos VHDL, estos archivos deben estar contenidos en la misma carpeta donde se encuentra el ensamblador y el archivo fuente .psm. Las plantillas definen un bloque RAM Single Port[3] de la Spartan-3E configurada como una ROM. Se puede modificar estas plantillas para la definición de memoria que se quiera. Los archivos suministrados por Xilinx incluyen notas adicionales de cómo funciona las plantillas.

Estas plantillas son igualmente usadas en conjunto con pBlazIDE para generar el modulo ROM. También existen otras versiones de estas plantillas que incorporan opciones de multiprogramación de microcontroladores o programación vía JTAG, solución que reduce el tiempo de desarrollo.

El ensamblador lee la plantilla y copia la información generada en el archivo de salida. No se realiza chequeo de la sintaxis, por lo que cualquier modificación puede conllevar al no funcionamiento o errores en el microcontrolador. El ensamblador también genera los archivos hexadecimal y decimal, .hex y .dec, con la misma información para otras utilidades.

El archivo <nobmre_archivo>.log proporciona los detalles sobre el proceso de ensamblado que se realizo. En el se puede observar cómo cada instrucción y directiva fue usada. La dirección y código de operación asociada a cada línea del programa y los valores reales de direcciones, registros, y las constantes definidas son especificadas.

Constant.txt es un archivo que contiene el valor de las constantes declaradas con el uso de la directiva CONSTANT, y Labels.txt suministra una lista de las etiquetas y las direcciones asociadas a estas.

Pass.dat son archivos internos para el ensamblar y representan etapas intermedias el del proceso de ensamble. Pueden ser usados para identificar la forma en que el ensamblador ha interpretado el archivo de programa sintaxis, así solucionar errores obtenidos durante el ensamble ya que brindan formación precisa de cada operación que se realiza.

Luego de generar los archivos VHDL y .coe, el ensamblador genera una nueva versión del programa fuente que tiene una mejor apariencia y luce mas ordenado. Este archivo puede ser tomado como el archivo fuente original para continuar con el desarrollo; en esta versión encontramos:

[3] Memoria RAM de un solo puerto.

- Formato de etiquetas y comentarios
- Pone todos los comandos en mayúscula
- Espacio correcto entre operandos
- Los registros están en formato sX
- Conversión de las constantes hexadecimales a mayúsculas.

pBlazIDE

El software pBlazIDE de Mediatronix es un entorno de desarrollo libre y gráfico para el sistema operativo Windows. Sus principales características son las siguientes:

Identificación de sintaxis por colores
Completo Simulator de las instrucciones (ISS)
    Interrupciones de prueba (Breakpoints)
    Visualización de los registros
    Visualización de la Memoria ScratchPad
Función de conversión de sintaxis a pBlazIDE al importar archivos KCPSM3

El ensamblador pBlazIDE proporciona una directiva, usando la palabra VHDL, para establecer el nombre del archivo de salida y la entidad VHDL. El archivo template.vhd es la plantilla VHDL del modulo ROM, el archivo target.vhd es el archivo de salida VHDL, derivado de la plantilla template.vhd.

```
VHDL "template.vhd", "target.vhd", "entity_name"
```

El software de desarrollo pBlazIDE soporta diferentes variaciones de la arquitectura PicoBlaze, por eso debe ser seleccionada la versión PicoBlaze 3 para el uso en la FPGA Spartan-3E. En el menú seleccione Settings > PicoBlaze 3.

Durante el desarrollo del código el ISS pBlazIDE es la mejor herramienta para la simulación del microcontrolador PicoBlaze. pBlazIDE brinda ventajas como el rápido desarrollo del código, el ISS, simulación paso a, breakpoints, total interacción con los registros, banderas y los valores de la memoria RAM Scratchpad, funciones E/S para System-level Simulation vía archivo y es gratis, frente a su única debilidad asociada a la imposibilidad de modelar los comportamientos lógicos asociados al microcontrolador PicoBlaze.

El pBlazIDE ISS no soporta la simulación integrada del microcontrolador PicoBlaze con otra lógica de la FPGA. Esta simulación puede realizarse usando el simulador ModelSim, que permite la simulación del diseño completo.

SET DE INSTRUCCIONES

ADDRESS SPACES

El microcontrolador PicoBlaze tiene cinco distintos espacios de dirección. Para cada espacio de dirección operan instrucciones específicas, Tabla 1

Tabla 1: Espacio de Direcciones e Instrucciones relacionadas

Address Space	Dimensión	Modo de Dirección	Instrucciones que Operan
Instrucciones	1Kx18	Direct	JUMP CALL RETURN RETURNI Evento INTERRUPT Evento RESET Y todos los incrementos del PC a la siguiente ubicación.
Registros	16x8	Direct	LOAD AND OR XOR TEST ADD ADDCY SUB SUBCY COMPARE SR0 SR1 SRX SRA RR SL0 SL1 SLX SLA RL INPUT OUTPUT STORE FETCH
Memoria Scratchpad RAM	64x8	Directo / Indirecto	STORE FETCH
Entradas / Salidas	256x8	Directo / Indirecto	INPUT OUTPUT

Pila CALL/RETURN	31x10		CALL RETURN RETURNI Evento RESET Habilitar evento INTERRUPT

PROCESAMIENTO DE DATOS

Las instrucciones de procesamiento operan en los 16 registros de propósito general, solo estas instrucciones modifican el estado de las banderas ZERO y CARRY relacionado a cada instrucción. Las instrucciones para el procesamiento de datos están divididas así:

- Instrucciones Lógicas
- Instrucciones Aritméticas
- Instrucciones de prueba y comparación
- Instrucciones de desplazamiento y rotación

Instrucciones Lógicas

Las instrucciones lógicas realizan las operaciones lógicas AND, OR o XOR entre dos operandos. El primer operando es cualquiera de los registros, el segundo operando es otro registro o una constante. Además, de las operaciones lógicas, las instrucciones se usan para:

- Complementar o invertido un registro
- Borrar un registro
- Establecer o borrar bits específicos dentro de un registro

Todas las operaciones lógicas ser realizan bit a bit. Si el resultado de la operación es cero se establece la bandera ZERO. La bandera CARRY es siempre limpiada por una instrucción lógica.

AND

```
AND sX,sY     ; AND lógica bit a bit del registro sX y el registro sY
AND sX,kk     ; AND lógica bit a bit del registro sX y una constante inmediata
kk
```

Registros y Banderas Afectadas
Registros sX, PC
Banderas: ZERO según el resultado, CARRY siempre 0

OR

```
OR sX,sY      ; OR lógica bit a bit del registro sX y el registro sY
OR sX,kk      ; OR lógica bit a bit del registro sX y una constante inmediata kk
```

Registros y Banderas Afectadas
Registros sX, PC
Banderas: ZERO según el resultado, CARRY siempre 0

XOR

```
XOR sX,sY     ; XOR lógica bit a bit del registro sX y el registro sY
XOR sX, kk    ;XOR lógica bit a bit del registro sX y una constante inmediata kk
```

Registros y Banderas Afectadas
Registros sX, PC
Banderas: ZERO según el resultado, CARRY siempre 0

Invertir Registros

El microcontrolador no tiene una instrucción para invertir en un registro sX. Usando la instrucción XOR sX, FF se despeña una función equivalente.

```
LOAD s0,AA    ;Carga el registro s0=10101010
XOR s0,FF     ;Invierte el contenido de s0=01010101
```

Invertir Bits

El microcontrolador no tiene una instrucción para invertir un bit en un registro sX. Usando la instrucción XOR se puede realizar una operación equivalente, realizar XOR con una mascara especifica para el bits o bits que se quieren invertir. Un bit ’1’ en la mascara invierte el bit correspondiente en el registro sX, ‘0’ para no modificar los bits de sX.

```
XOR s0,80     ;Invierte el bit mas significativo del registro s0
```

Limpiar Registros

El microcontrolador no tiene una instrucción para limpiar un registro sX. Con el uso de la instrucción XOR sX, sX se obtiene el mismo resultado. Al realizar XOR se limpia el registro y la bandera ZERO se establece

```
XOR s0,s0     ; Limpia el registro s0 y se establece la bandera ZERO
```

También se puede limpiar el registro sX sin afectar la bandera ZERO haciendo uso de la instrucción LOAD.

Establecer un bit

El microcontrolador no tiene una instrucción para establece un bit en un registro sX. Con el uso de la instrucción OR se puede obtener este resultado, realizar OR con una mascara especifica para el bits o bits que se quieren establecer. Un bit ’1’ en la mascara invierte el bit correspondiente en el registro sX, ‘0’ para no modificar los bits de sX.

```
OR s0,80      ; Establece el bit mas significativo del registro s0
```

Limpiar un bit

El microcontrolador no tiene una instrucción para limpiar un bit en un registro sX. Con el uso de la instrucción AND se obtiene el mismo resultado, realizar AND con una mascara especifica para el bits o bits que se quieren limpiar. Un bit ’0’ en la mascara invierte el bit correspondiente en el registro sX, ‘1’ para no modificar los bits de sX.

```
AND s0,7F     ; Limpia el bit mas significativo del registro s0
```

Instrucciones Aritméticas

El microcontrolador PicoBlaze posee instrucciones para realizar operaciones de suma y resta entre bytes. Las combinaciones de instrucciones y uso de las características de la FPGA permiten lograr realizar operaciones de multiplicación y división. Para desempeño un alto desempeño de operaciones aritméticas se considera el uso del procesador RISC de 32-bit MicroBlaze.

Instrucciones de suma ADD y ADDCY (ADDC)

El microcontrolador PicoBlaze posee dos instrucciones ADD y ADDCY (ADDC para el ensamblador pBlazIDE) que realizan el calculo de la suma de dos operandos de 8 bits, sin o con acarreo respectivamente, El primer operando es cualquiera de los registros, el segundo operando es otro registro o una constante. El resultado de esta operación afecta las banderas ZERO y CARRY. Cuando el resultado es mayor que 255, desborde, la bandera CARRY se establece. Si el resultado de la suma es 0 ó 256, el registro sX es cero, la bandera ZERO se establece. La instrucción ADDCY (ADDC) es una suma con acarreo. Si la bandera CARRY esta establecida se suma uno al resultado de la suma.

ADD

```
ADD sX,sY     ; Suma de registros        sX = sX + sY
ADD sX,kk     ; Suma inmediata    sX = sX + kk
```

Registros y Banderas Afectadas
Registros sX, PC
Banderas: ZERO y CARRY según el resultado

ADDCY (ADDC)

```
ADDCY sX,sY   ; Suma de registros        sX = sX + sY + CARRY
ADDCY sX,kk   ; Suma inmediata    sX = sX + kk + CARRY
```

Registros y Banderas Afectadas
Registros sX, PC
Banderas: ZERO y CARRY según el resultado
ADDC es la instrucción equivalente para pBlazIDE

Instrucciones de resta SUB y SUBCY (SUBC)

El microcontrolador PicoBlaze posee dos instrucciones SUB y SUBCY (SUBC para el ensamblador pBlazIDE) que realizan el calculo de la resta de dos operandos de 8 bits, sin o con pedir restando respectivamente, El primer operando es cualquiera de los registros, el segundo operando es otro registro o una constante. El resultado de esta operación afecta las banderas ZERO y CARRY. Cuando el resultado es menor que cero la bandera CARRY se establece. Si el resultado de la resta es 0 ó -256, la bandera ZERO se establece. La instrucción SUBCY (SUBSC) es una resta pidiendo. Si la bandera CARRY esta establecida se resta uno al resultado de la resta.

SUB

```
SUB sX,sY     ; Resta de registros sX = sX - sY
SUB sX,kk     ; Resta inmediata   sX = sX - kk
```

Registros y Banderas Afectadas
Registros sX, PC
Banderas: ZERO y CARRY según el resultado

SUBCY (SUBC)

```
SUBCY sX,sY   ; Resta de registros       sX = sX - sY - CARRY
SUBCY sX,kk   ; Resta inmediata   sX = sX - kk - CARRY
```

Registros y Banderas Afectadas
Registros sX, PC
Banderas: ZERO y CARRY según el resultado
SUBC es la instrucción equivalente para pBlazIDE

Suma de 16 bits

La instrucción ADCCY es comúnmente usada para lograr suma de n-bytes. Las suma de enteros de 16 bits ser realiza sumando cada parte baja de los 16 bits para obtener la parte menos significativa de los 16 bits LSB, y la cada parte alta de los 16 bits para obtener la parte mas significativa MSB.

```
LOAD s0, lsb_a      ;LSB del número a
LOAD s0, msb_a      ; MSB del número a
LOAD s0, lsb_b      ; LSB del número b
LOAD s0, msb_b      ; MSB del número b

ADD s0, s2          ; s0 es el LSB de la suma de dos numero de 16bits a y b
ADDCY s1, s3        ; s1 es el MSB de la suma de dos numero de 16bits a y b
```

Resta de 16 bits

La instrucción SUBCY es comúnmente usada para realizar la resta de n-bytes. Las resta de enteros de 16 bits ser realiza restando cada parte baja de los 16 bits para obtener la parte menos significativa de los 16 bits LSB, y la cada parte alta de los 16 bits para obtener la parte mas significativa MSB.

```
LOAD s0, lsb_a      ;LSB del número a
LOAD s0, msb_a      ; MSB del número a
LOAD s0, lsb_b      ; LSB del número b
LOAD s0, msb_b      ; MSB del número b

SUB s0, s2          ; s0 es el LSB de la resta de dos numero de 16bits a y b
SUBCY s1, s3        ; s1 es el MSB de la resta de dos numero de 16bits a y b
```

Incrementos/Decrementos

El microcontrolador no tiene una instrucción para realizar incrementos o decrementos en un registro sX. Con el uso de las instrucciones ADD y SUB se tiene esta operación.

```
SUB s0, 01          ; Se decremento el registro s0
ADD s0, 01          ; Se incrementa el registro s0
```

Para una variable de 16 bits:

```
ADD s0, 01          ; Se incrementa LSB
ADDCY s1, 00        ; Se el MSB solo si genero acarreo la operación anterior.
```

Negativo

El microcontrolador no tiene una instrucción dedicada a negar el valor de un registro sX. Sin embargo, con el uso de las operaciones lógicas y aritméticas se puede obtener el negativo de un byte.

```
XOR s0, FF          ; Invierte todo los bits del registro s0, complemento uno
ADD s0, 01          ; Suma uno al registro s0, complemento a dos
```

No Opera (NOP)

El microcontrolador PicoBlaze no tiene una instrucción específica que realice la instrucción NOP. NOP es una instrucción que no afecta registros o banderas y que requiere un ciclo de instrucción para realizarse. Una instrucción NOP algunas es útil para equilibrar el código para obtener un determinado tiempo de ejecución.

Algunas formas de obtener este comportamiento:
Cargando un registro en si mismo no afecta el registro ni el valor de las banderas.

```
LOAD s0, s0
```

Prueba y Comparación

El microcontrolador PicoBlaze posee dos instrucciones que permite probar los bits de un registro y comparar el valor con otro registro o una constante. Las instrucciones TEST o COMPARE solo afectan las banderas ZERO y CARRY.

Test

La instrucción TEST se desempeña haciendo una AND lógica bit a bit entre dos registros operandos. El primer operando es cualquiera de los registros, el segundo operando es otro registro o una constante. No se afecta ninguno de los registros solo las banderas. La bandera ZERO se establece si todos los bits temporales resultantes son cero.
Si el segundo operando contiene un único bit ‘1’, entonces la bandera CARRY es la prueba de si el correspondiente bit en el primer operando es ‘1’.
En una aplicación mas completa de la instrucción TEST, la bandera CARRY genera paridad impar para los bits incluidos del primero operando, sX. El segundo operando actúa como una mascara, si el bit del segundo operando es ‘0’ el correspondiente bit en sX no se incluye para general el valor de paridad; si el bit del segundo operando es ‘1’, entonces el bit correspondiente en sX es incluido en la generación de paridad.

```
TEST sX, sY   ; Test entre registros
TEST sX, kk   ; Test con una constante

TEST s0, FF   ; La mascara FF Incluye todos los bit en la generación de paridad
LOAD s0, 05   ; s0=00001010, se prueba contra la mascara 00000100
TEST s0, 04   ; Como en el bit 3 del primer operando se encuentra un ‘1’
              ; Se establece la bandera CARRY, y además se limpia ZERO
```

Registros y Banderas Afectadas
Registros PC
Banderas: ZERO y CARRY según el resultado

COMPARE (COMP)

La instrucción COMPARE(COMP) se desempeña haciendo una resta de los dos operandos, el resultado no afecta a algún registro, solo a la banderas ZERO y CARRY.
La bandera ZERO se establece cuando los dos operandos son idénticos, y cuando el segundo operando es mayor que el primer operando se establece la bandera CARRY.
Para un fácil entendimiento de los resultados se puede recurrir a la Tabla 2.

```
COMPARE sX, sY      ; Comparación entre registros
COMPARE sX, kk      ; Comparación con una constante
```

Tabla 2: Operación de las Banderas para la instrucción COMP

Resultado	Bandera ZERO	Bandera CARRY
Operando > sX	0	1
Operando < sX	0	0
sX = Operando	1	0
sX ≠ Operando	0	1 ó 0

Registros y Banderas Afectadas
Registros PC
Banderas: ZERO y CARRY.
COMP es la instrucción equivalente para pBlazIDE

Instrucciones de desplazamiento y rotación

El microcontrolador PicoBlaze soporta un amplio set de funciones de desplazamiento. Todas las instrucciones de desplazamiento modifican el contenido solo un registro, y afectan las banderas CARRY y ZERO.

Instrucciones de desplazamiento

SL0 – SR0

SL0 y SR0, son instrucciones de desplazamiento, a izquierda y derecha respectivamente. Ponen un cero a los extremos contrarios a la dirección del desplazamiento. El bit que sale del registro se desplaza a CARRY

SL1 – SR1

SL1 y SR1, son instrucciones de desplazamiento, a izquierda y derecha respectivamente. Ponen un uno a los extremos contrarios a la dirección del desplazamiento. El bit que sale del registro se desplaza a CARRY

SRX – SLX

SRX y SLX, realizan una operación de desplazamiento aritmético, a derecha e izquierda respectivamente. Los bits de los extremos contrarios al desplazamiento son copiados nuevamente en el registro la posición que ocupan inicialmente. El bit que sale del registro se desplaza a CARRY.

SLA – SRA

La instrucción SLA y SRA hace que el registro sX y la bandera CARRY se comporten como un registro de nueve bits para realizar una rotación, a izquierda o derecha, CARRY es el MSB cuando rota a la izquierda, y el LSB cuando se rota a la derecha

```
SL0 sX ; Desplazamiento a izquierda llenando con cero
SL1 sX ; Desplazamiento a izquierda llenando con cero
SLX sX ; Desplazamiento a izquierda, manteniendo el bit 0
SLA sX ; Desplazamiento a izquierda por todos los bits incluyendo CARRY.

SR0 sX ; Desplazamiento a derecha llenando con cero
SR1 sX ; Desplazamiento a derecha llenando con cero
SRX sX ; Desplazamiento a derecha, manteniendo el bit 0
SRA sX ; Desplazamiento a derecha por todos los bits incluyendo CARRY
```

Registros y Banderas Afectadas
Registros sX, PC
Banderas: ZERO según el contenido del registro, y CARRY según el bit desplazado fuera del registro.

Limpiar y Establecer CARRY

En algunos programas de las aplicaciones es necesario establecer o limpiar la bandera CARRY, esto es posible realizar de la siguiente forma.

Hacer AND del registro con si mismo, limpia la bandera CARRY sin afectar el contenido del registro.

```
AND s0, s0   ; Limpia la bandera CARRY sin afectar el registro
```

Existen diversos métodos para establecer la bandera CARRY, en su mayoría se afecta algún registro.

```
LOAD sX, 00
COMPARE sX, 01 ; Se establece la bandera CARRY como se indica en la Tabla 2
```

MOVIMIENTO DE DATOS

El movimiento de datos es una de las partes importantes del funcionamiento del microcontrolador, es la forma como se intercambia información entre sus componentes. Dependiendo de donde y hacia donde van los datos se usan las diferentes instrucciones:

- Al registro, LOAD
- Entre la memoria ScratchPad y el registro, STORE y FETCH
- Entre el microcontrolador y la FPGA, INPUT (IN) y OUTPUT (OUT)

LOAD

La instrucción LOAD mueve datos entre y a los registros del microcontrolador, carga el contenido en cualquiera de los registros. El valor cargado es cualquier registro o una constante. No tiene ningún efecto sobre las banderas. Al no afectar el estado de las banderas puede usarse en cualquier parte del programa para cargar datos en un registro.
El uso de LOAD para carga un registro con una constante en un registro no tiene efecto sobre el tamaño del programa o el desempeño de este, razón por la cual se usa para limpiar los registros o asignarles nuevos valores.

```
LOAD sX,sY    ; Mueve el contendido del registro sY  al registro sX
LOAD sX,kk    ; Carga en el registro sX la constante kk.
```

Registros y Banderas Afectadas
Registros sX, PC
Banderas, no afecta ninguna.

STORE

La instrucción STORE mueve datos de los registros a la memoria RAM ScratchPad. Escribe el registro sX a la ubicación de la Scratchpad RAM especificada por el operando. La RAM ScratchPad tiene 64 posiciones de memoria, por tanto los dos bits mas significativos, son descartados. La dirección máxima para direccionar el movimiento de datos es 3F.

```
STORE sX, (sY)      ; Escribe el registro sX a la ubicación de la RAM
                    ; ScratchPad indicada en el contenido de sY
STORE sX, kk        ; Escribe el registro sX a la ubicación de la RAM
                    ; ScratchPad indicada por la constante kk
```

Registros y Banderas Afectadas
Registros sX, PC
Banderas, no afecta ninguna.
La sintaxis para el uso en pBlazIDE no incluye uso de los paréntesis.

FETCH

La instrucción STORE lee los datos de una ubicación de la RAM ScratchPad al registro sX. La RAM ScratchPad tiene 64 posiciones de memoria, por tanto los dos bits mas significativos, son descartados. La dirección máxima para direccionar el movimiento de datos es 3F.

```
FETCH sX, (sY)      ; Copia al registro sX el contenido de una posición
                    ; de la RAM ScratchPad indicada en el registro sY
FETCH sX, (sY)      ; Copia al registro sX el contenido de una posición
                    ; de la RAM ScratchPad indicada en la constante kk.
```

Registros y Banderas Afectadas
Registros sX, PC
Banderas, no afecta ninguna.
La sintaxis para el uso en pBlazIDE no incluye uso de los paréntesis.

INPUT (IN)

La instrucción INPUT establece la salida del puerto PORT_ID por el valor especificado por en un registro o una constante. El valor del puerto de entrada IN_PORT es leído y puesto en el registro sX. Las banderas no son afectadas por esta operación.
Interfaz lógica decodifica la dirección de PORT_ID para proporcionar el valor correcto en IN_PORT.

```
INPUT sX, (sY)      ; Pone en PORT_ID el contenido de sY
                    ; Copia el valor en IN_PORT al registro sX

INPUT sX, kk        ; Pone en PORT_ID el valor de la constante kk
                    ; Copia el valor en IN_PORT al registro sX
```

Registros y Banderas Afectadas
Registros sX, PC
Banderas, no afecta ninguna.
La instrucción en pBlazIDE es IN, la sintaxis no incluye paréntesis. La salida READ_STROBE es puesta en alto durante el segundo ciclo de reloj CLK de los dos ciclos que tarda la operación de lectura.

OUTPUT (OUT)

La instrucción OUTPUT establece la dirección en la salida del puerto PORT_ID por el valor especificado por en un registro o una constante. Se escribe el contenido del registro sX en el puerto de salida OUT_PORT. La lógica de la FPGA debe realizar el direccionamiento de la salida con el valor de IN_PORT y la señal WRITE_STROBE.

```
OUTPUT sX, (sY)     ; Escribe el contenido del registro sX a OUT_PORT
                    ; y del registro SY a PORT_ID

OUTPUT sX, kk       ; Escribe el contenido del registro sX a OUT_PORT
                    ; y valor de la constante kk a PORT_ID
```

Registros y Banderas Afectadas
Registros sX, PC
Banderas, no afecta ninguna.
La instrucción en pBlazIDE es OUT, la sintaxis no incluye paréntesis. La salida WRITE_STROBE es puesta en alto durante el segundo ciclo del reloj CLK de los dos ciclos que tarda la operación de escritura.

CONTROL DE FLUJO

El contador de Programa (PC) indica la siguiente instrucción que se ejecuta y controla directamente el flujo del programa. El microcontrolador automáticamente pasa a realizar la siguiente instrucción almacenada, aumenta el contador de programa a la siguiente posición (PC = PC + 1). PC no puede ser accedido directamente. A pesar de esto, es posible modificar la operación por defecto del flujo de programa cargado el PC con un valor diferente, se logra con el uso de las instrucciones JUMP y CALL/RETURN
Un evento de Interrupción también logra modificar el flujo del programa.

Las instrucciones JUMP, CALL y RETURN son ejecutadas condicionalmente, dependiendo de una condición establecida al estado de las banderas CARRY o ZERO, o incondicionalmente en caso de no indicar una condición., ver Tabla 3.

Tabla 3: Condiciones de Ejecución

Condición	Descripción
sin indicar	Siempre es verdad. Se ejecuta la instrucción incondicionalmente
C	CARRY = 1. Se ejecuta la instrucción si la bandera CARRY esta establecida
NC	CARRY = 0. Se ejecuta la instrucción si la bandera CARRY esta limpia
Z	ZERO = 1. Se ejecuta la instrucción si la bandera ZERO esta establecida
NC	ZERO = 0. Se ejecuta la instrucción si la bandera ZERO esta limpia

JUMP

La instrucción JUMP modifica la ejecución secuencial por un salto a una dirección específica del programa. Cada instrucción de JUMP debe indicar una dirección de 10 bits en formato de tres dígitos hexadecimal o una etiqueta que el ensamblador convertirá a tres dígitos hexadecimal.

La instrucción JUMP es condicional y se ejecuta solo para la condición específica. Si la condición es falsa, la instrucción JUMP no tiene algún efecto, transcurren los dos ciclos de reloj CLK de la operación y continuar el programa con su flujo normal.
Si se ejecuta JUMP, entonces el PC es cargado con la dirección de la etiqueta. El microprocesador ejecutara las instrucciones a continuación de la etiqueta.
Las Banderas y los registros no son afectados, Ni la pila de CALL/RETURN.
Todos los saltos son absolutos, no relativos.

```
JUMP ETIQUETA       ; Salto incondicional a ETIQUETA
JUMP C, ETIQUETA    ; Si CARRY = 1, Salta a ETIQUETA
JUMP NC, ETIQUETA   ; Si CARRY = 0, Salta a ETIQUETA
JUMP Z, ETIQUETA    ; Si ZERO = 1, Salta a ETIQUETA
JUMP NZ, ETIQUETA   ; Si ZERO = 0, Salta a ETIQUETA
```

Registros y Banderas Afectadas
Registros PC
Banderas, no afecta ninguna.

CALL/RETURN

La instrucción CALL altera el flujo de programa de forma temporal, salta a una función subrutina y regresa a la instrucción siguiente a la llamada.

Si la instrucción CALL es realizada el PC es puesto en la parte alta de la pila CALL/RETURN, y se carga con la dirección en la etiqueta, pasando el microcontrolador a ejecutar las instrucciones de la etiqueta indicada. El microcontrolador realizara las instrucciones de la subrutina hasta que encuentre una instrucción RETURN. Cada instrucción CALL debe tener su correspondiente RETURN. La instrucción RETURN también es un condicional, y solo se realiza para las condiciones especificadas, ver Tabla 3. La instrucción RETURN termina la subrutina, y toma la parte superior de la pila CALL/RETURN, incrementa el valor, y carga el valor en el PC. El flujo de programa regresa a la instrucción inmediatamente después de la instrucción CALL.

La pila CALL/RETURN tiene un tamaño máximo de 31 direcciones, si esta cantidad de llamadas anidadas es superada, la pila es sobre escrita, implica tener cuidado con no exceder las llamadas anidadas sin perder las posición a las que debe regresar el programa.

CALL

El uso de la instrucción CALL debe indicar una dirección de 10 bits en formato de tres dígitos hexadecimal o una etiqueta que el ensamblador convertirá a tres dígitos hexadecimal.
La instrucción CALL es condicional y se ejecuta solo para la condición específica. Si la condición es falsa, la instrucción CALL no tiene algún efecto, transcurren los dos ciclos de reloj CLK de la operación y continuar el programa con su flujo normal. Las Banderas

y los registros no son afectados. Sin embargo, la subrutina llamada puede modificar los registros y las banderas.

```
CALL ETIQUETA       ; Salto incondicional a ETIQUETA
CALL C, ETIQUETA    ; Si CARRY = 1, Salta a ETIQUETA
CALL NC, ETIQUETA   ; Si CARRY = 0, Salta a ETIQUETA
CALL Z, ETIQUETA    ; Si ZERO = 1, Salta a ETIQUETA
CALL NZ, ETIQUETA   ; Si ZERO = 0, Salta a ETIQUETA
```

Registros y Banderas Afectadas
Registros PC, pila CALL/RETURN
Banderas, no se afecta ninguna.
El numero máximo de llamadas a subrutinas anidadas en de 31 niveles, debido al tamaño de la pila CALL/RETURN.

RETURN

La instrucción RETURN es la instrucción complemento de la instrucción CALL, que debe estar siempre acompañada por RETURN. El nuevo valor de PC es el último valor de la pila incrementado, garantizando que el programa ejecute la instrucción siguiente a la instrucción CALL que llamo la subrutina. RETURN no afecta el estado de las banderas.

Dependiendo del la condición el programa regresa desde la subrutina que fue llamada. Si la condición no es valida, el programa continúa en la siguiente instrucción

```
RETURN    ; Toma PC de la pila y ejecuta la instrucción PC+1
RETURN C  ; Si CARRY = 1, Toma PC de la pila y ejecuta la instrucción PC+1
RETURN NC ; Si CARRY = 0, Toma PC de la pila y ejecuta la instrucción PC+1
RETURN Z  ; Si ZERO = 1,Toma PC de la pila y ejecuta la instrucción PC+1
RETUNR NZ ; Si ZERO = 0,Toma PC de la pila y ejecuta la instrucción PC+1
```

Registros y Banderas Afectadas
Registros PC, pila CALL/RETURN
Banderas, no se afecta ninguna.
RET es la instrucción equivalente para pBlazIDE.

INTERRUPCIÓN

Un evento de Interrupción modifica el flujo de programa, para un realizar una tarea que se encuentra por fuera del flujo por defecto del programa. El microcontrolador PicoBlaze posee una sola entrada de señal de interrupción, en aplicación donde son necesarias mas de una señal de interrupción se combinan las señales usando lógica simple para obtener una señal de interrupción. Después de un RESET del microcontrolador la interrupción es desactivada, esta se activa usando la instrucción ENABLE INTERRUPT, y para desactivar en cualquier parte del programa se usa la instrucción DISABLE INTERRUPT.

Una señal de interrupción en la entrada INTERRUPT debe permanecer activa por lo menos dos ciclos de reloj CLK para garantizar que esta es reconocida y generar un evento INTERRUP. Una interrupción hace que el microcontrolador PicoBlaze ejecute la instrucción CALL 3FF, que llama a la dirección 3FF de forma incondicional, la ubicación 3FF es la usada para almacenar la ISR. Para regresar de esta rutina se usa la instrucción RETURNI, este comando garantiza el fin del ISR, restaura el estado de las banderas y habilita el las futuras interrupciones.
Cuando se ejecuta RETURNI se recarga el PC almacenado en la pila, y se regresa a la siguiente instrucción del flujo de programa donde ocurrió la interrupción.

Al iniciar el proceso de Interrupción el microcontrolador PicoBlaze realiza algunas funciones automáticamente. Almacena el estado actual de las banderas ZERO y CARRY y deshabilita cualquier futura interrupción. De igual forma el valor actual de PC se almacena en la pila CALL/RETURN.
El microcontrolador PicoBlaze establece en alto la señal INTERRUPT_ACK durante el segundo ciclo del evento INTERRUP para indicar que la interrupción fue reconocida. La señal INTERRUPT_ACK puede ser usada para limpiar el manejo lógico de la interrupción externa.

Cuando la aplicación programada no necesita de interrupciones la señal INTERRUP se pone a un bajo.

PUERTOS DE ENTRADA Y SALIDA

El microcontrolador PicoBlaze soporta hasta 256 puertos de entrada y 256 puertos de salida, que pueden ser combinados para crear puertos entrada/salida.

Los tiempos críticos de diseño, establecen limitaciones de tiempo para el PORT_ID y los datos a dos ciclos de reloj CKL. La sola necesidad de los indicadores de lectura y el de escritura son limitados a un ciclo de reloj. Para el máximo rendimiento y simplificar las limitaciones de tiempo, se agrega un registro en la señal cuando sea posible.

Aun fuera del microcontrolador el diseño mantiene la interfaz lógica compacta con un con buen desempeño.

PUERTO PORT_ID

El puerto PORT_ID suministra la identificación del puerto o dirección del puerto asociado a una operación de instrucción entrada o salida. El puerto PORT_ID es válido por dos ciclos de reloj, permitiendo suficiente tiempo para cualquier interfaz de decodificación lógica y para conexiones con RAM asíncronas. Los dos ciclos permiten leer operaciones desde cualquier RAM asíncrona, como un bloque RAM.
Las operaciones de entrada y salida soportan direccionamiento directo e indirecto. La dirección de puerto es suministrada inmediatamente como una constante de ocho bits o indirectamente como el contenido de cualquiera de los 16 registros. El direccionamiento indirecto es ideal cuando accede a un bloque de memoria, ya sea un periférico en el puerto de direcciones contiguo o un bloque de memoria distribuida interna o externa a la FPGA

Agregar periféricos externos al microcontrolador PicoBlaze es relativamente sencillo. El único problema es la descodificación del valor del PORT_ID usando los requerimientos mínimos en la lógica de la aplicación. La decodificación depende del número de entradas, salidas o puertos bidireccional

Tabla 4: Decodificación de acuerdo al número de entradas

Número de Puertos	Entrada	Salida
0 a 1	No requiere Multiplexado	No requiere Decodificación
2 a 8	Un multiplexor de entrada binaria	Una línea codificada por POR_ID
9 a 255	Un árbol de Multiplexores en cascada	

ENTRADA DE DATOS

Una operación de entrada transfiere los datos del puerto IN_PORT a cualquiera de los 16 registros definidos en por sX. La salida PORT_ID esta definida por un registro sY o por una constante inmediata, selecciona la fuente para la entrada deseada. Las Fuentes de entrada generalmente son seleccionadas a través de un multiplexor, utilizando una parte de los bits puerto de salida PORT_ID para seleccionar una fuente específica. El tamaño del multiplexor es proporcional al número de posibles fuentes de entrada, que tiene una incidencia directa en rendimiento.

La entrada de datos esta asociada a la salida READ_STROBE, que pone un pulso alto en el segundo ciclo de la operación de entrada. La señal READ_STROBE indica que el microcontrolador PicoBlaze ha recibido los datos.

SALIDA DE DATOS

En una operación de salida se presenta el contenido de cualquiera de los 16 registros en el puerto de salida OUT_PORT. El puerto POR_ID, se define por el un registro sY o por una constante de 8-bits, seleccionando el destino de la salida. Un pulso en el puerto WRITE_STROBE indica que los datos sobre el puerto OUT_PORT son válidos y están listos para su captura.

DIRECTIVAS DE PROGRAMACIÓN

ASIGNACIÓN DE CÓDIGO A UNA DIRECCIÓN ESPECÍFICA

La directiva de asignación a una dirección establece las instrucciones que preceden a la directiva en la dirección de memoria que se indico. Esto es necesario en caso como asignar el vector de reset y el vector de interrupción.

Para el interprete KCPSM3

```
ADDRESS 3FF
```

Para el interprete pBlazIDE

```
ORG $3FF
```

ASIGNAR NOMBRES A LOS REGISTROS

El microcontrolador PicoBlaze posee 16 registros de propósito general nombrados s0 a sF, para mejorar la escritura del código y para facilitar su reuso, se nombra o asigna un alias los registros del PicoBlaze.

Para el interprete KCPSM3

```
NAMEREG sX,<mireg>  ; El registro sX es renombrado <mireg>
```

Para el interprete pBlazIDE

```
<mireg> EQU sX      ; El registro sX es renombrado <mireg>
```

DEFINIR CONSTANTES

Es posible asignar nombres a valores constantes que se usan en el código, que permite interpretación del código por parte del desarrollador, haciéndolo mas fácil de comprender y de realizar su documentación.

Para el interprete KCPSM3

```
CONSTANT <miconstante> , kk      ; La constante kk es renombrada <miconstante>
```

Para el interprete pBlazIDE

```
<miconstante> EQU   kk           ; La constante kk es renombrada <miconstante>
```

DEFINIR PUERTOS EN PBLAZIDE

Al usar el software pBlazIDE para hacer uso del ISS es necesario definir los puertos de salida y entrada que se van asignar durante el código, son usados estos puerto durante el ISS.

Puertos de entrada

La directiva DSIN define el nombre del puerto y su dirección para un puerto de solo lectura. Mediante esta directiva pBlazIDE modela una entra que se conecta al microcontrolador en el puerto IN_PORT.

```
Puerto1 DSIN $01    ; Se define un puerto de lectura como Puerto1 y se asigna
                    ; la dirección de puerto $01
```

Puertos de Salida

La directiva DOUT define el nombre del puerto y su dirección para un puerto de solo escritura. Mediante esta directiva pBlazIDE modela una salida que se conecta al microcontrolador en el puerto OUT_PORT.

```
Puerto2 DSOUT $01   ; Se define un puerto de escritura como Puerto2 y se asigna
                    ; la dirección de puerto 01
```

PLANTILLAS PARA INICIAR UN PROGRAMA

Sintaxis KCPSM3

```
NAMEREG sX, <name>  ; Renombra el registro sX con <name>
CONSTANT <name>, kk ; Define la constante <name> con el valor kk

ADDRESS 000         ; EL programa siempre inicia en el vector de reset 000
ENABLE INTERRUPT    ; Al usar interrupciones se debe asegurar la habilitación
                    ; del evento INTERRUT
INICIO:             ; Etiqueta
                    ; Código del Programa

JUMP INICIO         ;La aplicación embebida se repite indefinidamente

ISR:                ; Rutina del servicio de interrupción
                    ;Define el comportamiento del microcontrolador durante
                    ; el evento INTERRUT

RETURNI ENABLE      ; Regreso del la rutina ISR, finaliza el evento INTERRUPT
ADDRESS 3FF         ; Vector de Interrupción, asignado al final de la memoria
JUMP ISR            ; Salta al servicio de interrupción
```

Sintaxis pBlazeIDE

```
<name> EQU    sX     ; Renombra el registro sX con <name>
<name> EQU    kk     ; Define la constante <name> con el valor kk

<name> DSIN   <port_id>    ; Habilita como entrada el puerto <port_id>
<name> DSOUT  <port_id     ; Habilita como salida el puerto <port_id>
<name> DSIO   <port_id>    ; Habilita el puerto de salida y lectura <port_id>

                    ; Asignar la plantilla a usar par implementar la ROM,
                    ; el nombre que lleva el archivo que contiene la ROM,
                    ; y la entidad mediante la cual está definida

VHDL “template.vhd”, “target.vhd”, “entity_name”

ORG    0            ; EL programa siempre inicia en el vector de reset 000
EINT                ; Al usar interrupciones se debe asegurar la habilitación
                    ; del evento INTERRUT
INICIO:             ; Etiqueta
                    ; Código del Programa

JUMP INICIO         ;La aplicación embebida se repite indefinidamente

ISR:                ; Rutina del servicio de interrupción
                    ;Define el comportamiento del microcontrolador durante
                    ; el evento INTERRUT

RETI ENABLE         ; Regreso del la rutina ISR, finaliza el evento INTERRUPT
ORG 3FF             ; Vector de Interrupción, asignado al final de la memoria
JUMP ISR            ; Salta al servicio de interrupción
```

RUTINAS DE SOFTWARE PARA EL pBlazIDE

Rutinas y subrutinas para el Display LCD
Inicializar el display LCD usando el modo de comunicación de 4-bits, incluye la rutina de borrar el display.

Función	28,	0 modo 4-bit, 1 dos líneas, 0 matriz de 5x7
Modo Entry	06,	1 incrementa, 0 no desplaza el display
Control de display	0C,	1 display encendido, 0 cursor off, 0 cursor blink off
Borrar display	01	

Registros usados s0, s1, s2, s3, s4

```
LCD_reset:
       CALL      delay_20ms          ; Espera por mas de 15ms para iniciar el
       LOAD      s4, 48              ; Display LCD
       CALL      LCD_write_inst4     ; envia '3'
       CALL      delay_20ms          ; espera mas de 4.1ms
       CALL      LCD_write_inst4     ; envia '3'
       CALL      delay_1ms           ; espera mas de 100us
       CALL      LCD_write_inst4     ; envia '3'
       CALL      delay_40us          ; espera mas 40us
       LOAD      s4, 32
       CALL      LCD_write_inst4     ; envia '2'
       CALL      delay_40us          ; espera mas 40us
       LOAD      s5, 40              ; Pone la función
       CALL      LCD_write_inst8
       LOAD      s5, 6               ; Modo Entry
       CALL      LCD_write_inst8
       LOAD      s5, 12              ; Control del Display
       CALL      LCD_write_inst8

       LCD_clear:
       LOAD      s5, 1               ; Borrar el Display
       CALL      LCD_write_inst8
       CALL      delay_1ms           ; Espera mas de 1.64 ms para borrar
       CALL      delay_1ms           ; el Display LCD
       RET
```

Rutina LCD_cursor
Esta rutina establece la posición del cursor, suministrada en el registro s5. La parte superior del byte indica la línea y la parte inferior la posición en del carácter en la línea.
Registros utilizados s0, S1, S2, S3, S4

```
LCD_cursor:
       TEST      s5, 16              ; Verificar si la dirección es de la
       JUMP      Z, set_line2        ; linea 1, sino pasar a linea 2
       AND       s5, 15              ; rango de direcciones $80 a 8F
       OR        s5, $80             ; para la linea 1
       CALL      LCD_write_inst8     ; Escribe la Instrucción set cursor
       RET
set_line2:
       AND       s5, 15              ; Rango de direcciones C0 a CF
```

```
       OR        s5, $C0             ; Para la linea 2
       CALL      LCD_write_inst8     ; Escribe la instrucción set cursor
       RET
```

Rutina LCD_write_data
Escribe datos de 8 bits en la pantalla LCD, suministrados en el registro s5. Registros utilizados s0, S1, S4, S5

```
LCD_write_data:
       LOAD      s4, s5              ; Instrucción Enable=0 RS=0,
       AND       s4, $F0             ; Escribe RW=0, E=0
       OR        s4, 12              ; Dato Enable=1 RS=1, RW=0 Write, E=0
       OUT       s4, LCD_output_port ; Rs y Rw en alto por mas de 40ns
                                     ; despues del pulso
       CALL      LCD_pulse_E         ; Escribe la parte alta del byte
       CALL      delay_1us           ; Espera por mas de 1us
       LOAD      s4, s5              ; toma la parte baja del byte
       SL1       s4                  ; Enable=1
       SL1       s4                  ; RS=1
       SL0       s4                  ; RW=0
       SL0       s4                  ; E=0
       OUT       s4, LCD_output_port ; Rs y Rw en alto por mas de 40ns
                                     ; despues del pulso
       CALL      LCD_pulse_E         ; Escribe la parte baja del byte
       CALL      delay_40us          ; espera por mas de 40us
       LOAD      s4, $F0             ; Instucción Enable=0 RS=0
       OUT       s4, LCD_output_port ; RW=0, E=0
       RET
```

Rutina LCD_write_inst8
Escribe una instrucción de 8-bits en la pantalla LCD. La instrucción se suministra en el registro s5. Registros utilizados s0, S1, S4, S5

```
LCD_write_inst8:
       LOAD      s4, s5              ; Instrucción de 8 bits
       AND       s4, $F0             ; Enable=0 RS=0 Instruction,
                                     ; RW=0, E=0
       OR        s4, LCD_drive       ; Enable=1
       CALL      LCD_write_inst4     ; Escribe la parte alta del byte
       CALL      delay_1us           ; wait >1us
       LOAD      s4, s5              ; select lower nibble with
       SL1       s4                  ; Enable=1
       SL0       s4                  ; RS=0 Instruction
       SL0       s4                  ; RW=0 Write
       SL0       s4                  ; E=0
       CALL      LCD_write_inst4     ; write lower nibble
       CALL      delay_40us          ; wait >40us
       LOAD      s4, $F0             ; Enable=0 RS=0 Instruction,
                                     ; RW=0, E=0
       OUT       s4, LCD_output_port ;
       RET
```

Subrutina LCD_write_inst4
Escribe una instrucción de 4-bits en la pantalla LCD. Es utilizada durante la inicialización y como parte de la subrutina de instrucción de 8-bits. Los cuatro bits de la instrucción se

encuentran en la parte superior del registro s4.
Registros utilizados s4

```
LCD_write_inst4:
       AND       s4, $F8             ; Escribe 4 bits de la instucción
       OUT       s4, LCD_output_port ; Instrucción Enable=1 RS=0,
                                     ;  RW=0, E=0
       CALL      LCD_pulse_E         ; RS y RW en alto por mas de 40ns
       RET                           ; despues del pulso
```

Subrutina LCD_pulse_E
La subrutina realiza un pulso en la señal E, necesariamente mayor a 230ns, se usa 1us. El registro s4 define el estado actual de la pantalla LCD en el puerto de salida. Registros utilizados s0, s4

```
LCD_pulse_E:
       XOR       s4, LCD_E           ; E=1, Pulso E para valida las
       OUT       s4, LCD_output_port ; Instucciones y Datos enviados
       CALL      delay_1us
       XOR       s4, LCD_E           ; E=0
       OUT       s4, LCD_output_port
       RET
```

Rutina LCD_read_data8
Lee un dato de 8 bits de la pantalla LCD a partir de la posición actual de la memoria LCD, posición del cursor. Se almacenan en el registro s5.

```
LCD_read_data8:
       LOAD      s4, 14              ; Dato Enable=1 RS=1, RW=1 Read, E=0
       OUT       s4, LCD_output_port ; RS y RW en alto por mas de 40ns
                                     ; despues del pulso
       XOR       s4, LCD_E           ; E=1
       OUT       s4, LCD_output_port
       CALL      delay_1us           ; Espera por mas de 260ns
                                     ; para los datos de acceso
       IN        s5, LCD_input_port  ; Lee la parta baja del byte
       XOR       s4, LCD_E           ; E=0
       OUT       s4, LCD_output_port
       CALL      delay_1us           ; Espera por mas de 1us
       XOR       s4, LCD_E           ; E=1
       OUT       s4, LCD_output_port
       CALL      delay_1us           ; Espera por mas 260ns
                                     ; para los datos de acceso
       IN        s0, LCD_input_port  ; Lee la parta alta del byte
       XOR       s4, LCD_E           ; E=0
       OUT       s4, LCD_output_port
       AND       s5, $F0             ; Une la parte alta y baja del byte
       SR0       s0
       SR0       s0
       SR0       s0
       SR0       s0
```

```
       OR        s5, s0
       LOAD      s4, 4               ; Dato Enable=0 RS=1, RW=0 Write, E=0
       OUT       s4, LCD_output_port ; Detiene la lectura
       CALL      delay_40us          ; espera por mas de 40us
       RET
```

Convertir a valor ASCII y mostrarlo en el LCD

Convierte un valor hexadecimal del registro s0 en un carácter ASCII. El valor puede ser cualquiera en el intervalo 00 a FF y se convierte a dos caracteres ASCII. Toma la parte alta del registro y devuelve su representación ASCII en el registro s2. La representación ASCII de la parte baja del registro es devuelta en s1. Las representaciones ASCII del 0 al 9, son 30 al 39 hexadecimal, se obtiene sumando 30 hexadecimal al valor decimal. Las representaciones ASCII de la 'A' a la 'F', son 41 a 46 hexadecimal, que requieren una nueva adición de 07 a los 30 ya añadido. Registros utilizados s0, S1 y S2.

```
hex_byte_to_ASCII:
       LOAD      s1, s0      ; Cambia de registro el valor Hexa
       SR0       s0                  ; toma la parte alta del byte
       SR0       s0
       SR0       s0
       SR0       s0
       CALL      hex_to_ASCII        ; Convierte el número Hexa a ASCII
       LOAD      s2, s0              ; El ASCII del número se mueve a s2
       LOAD      s0, s1              ; Retoma el valor Hexa
       AND       s0, 15              ; toma la parte baja del byte
       CALL      hex_to_ASCII        ; Convierte el número Hexa a ASCII
       LOAD      s1, s0              ; El ASCII del número se mueve a s1
       RET


hex_to_ASCII:
       SUB       s0, 10              ; Prueba si el valor esta
                                     ; en el rango 0 a 9
       JUMP      C, number_char
       ADD       s0, 7               ; Caracter ASCII A a F
                                     ; en el rango 41 a 46
       number_char:
       ADD       s0, 58              ; Caracter ASCII 0 a 9
                                     ; en el rango 30 to 40
       RET


disp_hex_byte:
       CALL      hex_byte_to_ASCII
       LOAD      s6, s1              ; Cambia el registro del caracter
       LOAD      s5, s2              ; Muestra en el Display en caracter
       CALL      LCD_write_data
       LOAD      s5, s6              ; Muestra en el Display en caracter
       CALL      LCD_write_data
       RET
```

Demoras por software.

La definición de valores constantes se refleja en el reloj aplicado al KCPSM3. Cada instrucción se ejecuta en 2 ciclos de reloj haciendo el cálculo altamente predecible. El

numero seis de la siguiente ecuación es debido a el número de instrucciones del código indicado.
delay_1us_constant=(clock_rate-6)/4.
El reloj en MHz, para la S3ESK delay_1us_constant es 0B.

```
delay_1us:  LOAD s0, 0B          ;Retardo de 1us, registro usado s0
wait_1us:   SUB s0, 01
            JUMP NZ, wait_1us
            RET

delay_40us: LOAD s1, 28          ; Retardo de 40us, registros usados s0, s1
wait_40us:  CALL delay_1us
            SUB s1, 01
            JUMP NZ, wait_40us
            RET

delay_1ms:  LOAD s2, 19          ; Retardo de 1ms, registros usados s0,s1,s2
wait_1ms:   CALL delay_40us
            SUB s2, 01
            JUMP NZ, wait_1ms
            RET

delay_20ms: LOAD s3, 19          ; Retardo de  20ms, registros usados s0, s1
wait_20ms:  CALL delay_1ms ; s2, s3
            SUB s3, 01
            JUMP NZ, wait_20ms
RET

delay_1s:   LOAD s3, 19          ; Retardo aproximado de 1s, registros usados
wait_1s:    CALL delay_20ms      ; s0, s1, s2, s3, s4
            SUB s3, 01
            JUMP NZ, wait_1s
            RET
```

PLANTILLAS PARA DISEÑO VHDL

El siguiente proceso describe el comportamiento lógico para las señales de entrada del microcontrolador conectadas al puerto IN_PORT. También, incluye el uso de la señal READ_STROBE para generar una señal que confirma la lectura realizada de uno de los puertos.

```
INPUT_PORTS: process(clk)
begin
if clk'event and clk='1' then
       case port_id(1 downto 0) is
              when "00" =>    in_port <=input_0;
              when "01" =>    in_port <=input_1;
              when "10" =>    in_port <=input_2;
              when others =>in_port <= "XXXXXXXX";
       end case;

       if (read_strobe='1' and port_id(1 downto 0)="01") then
              read_input_1<= '1';
       else
              read_input_1<= '0';
       end if;
end if;
end process INPUT_PORTS;
```

El siguiente proceso describe el comportamiento lógico para las señales de salida del microcontrolador conectado a un bus 24bits de ancho y para unas señales individuales:

```
OUTPUT_PORTS: process(clk)
begin
if clk'event and clk='1' then
       if write_strobe='1' then

              if port_id(0)='1' then
                     output_0(23 downto 16) <= out_port;
              end if;

              if port_id(1)='1' then
                     output_0(15 downto 8) <= out_port;
              end if;

              if port_id(2)='1' then
                     output_0(7 downto 0) <= out_port;
              end if;

              if port_id(4)='1' then
                     pin_0 <= out_port(0);
                     pin_1<= out_port(1);
                     pin_2<= out_port(2);
              end if;
       end if;
end if;
end process OUTPUT_PORTS;
```

El siguiente proceso describe el comportamiento lógico para el control de una señal que genera un evento INTERRUP, la señal que genera la interrupción es event_interrupt.

```
interrupt_control: process(clk)
begin
```

```
if clk'event and clk='1' then
       if interrupt_ack='1' then
              interrupt <= '0';
       elsif event_interrupt='1' then
              interrupt <= '1';
       else
              interrupt <= interrupt;
       end if;
end if;
end process interrupt_control;
```

El siguiente proceso describe el comportamiento lógico para las señales de salida del microcontrolador que controlan y transfieren datos a una pantalla LCD.

```
output_ports: process(clk)
begin
if clk'event and clk='1' then
       if write_strobe='1' then
              if port_id(1)='1' then
                     lcd_output_data <= out_port(7 downto 4);
                     lcd_drive <= out_port(3);
                     lcd_rs <= out_port(2);
                     lcd_rw_control <= out_port(1);
                     lcd_e <= out_port(0);
              end if;
       end if;
end if;
```

end process output_ports;

DESIGN SUMMARY

VIDEOVGA Project Status			
Project File:	videovga.ise	**Current State:**	Placed and Routed
Module Name:	embedded_kcpsm3	• **Errors:**	No Errors
Target Device:	xc3s500e-4fg320	• **Warnings:**	114 Warnings
Product Version:	ISE 9.1i	• **Updated:**	Vie 13. Jun 13:49:11 2008

VIDEOVGA Partition Summary
No partition information was found.

Device Utilization Summary				
Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of Slice Flip Flops	73	9,312	1%	
Number of 4 input LUTs	106	9,312	1%	
Logic Distribution				
Number of occupied Slices	96	4,656	2%	
Number of Slices containing only related logic	96	96	100%	
Number of Slices containing unrelated logic	0	96	0%	
Total Number of 4 input LUTs	176	9,312	1%	
Number used as logic	106			
Number used as a route-thru	2			
Number used for Dual Port RAMs	16			
Number used for 32x1 RAMs	52			
Number of bonded IOBs	30	232	12%	
IOB Flip Flops	3			
Number of Block RAMs	1	20	5%	
Number of GCLKs	1	24	4%	
Total equivalent gate count for design	74,820			
Additional JTAG gate count for IOBs	1,440			

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		

POWER REPORT

Design:	**C:\fpga\copia videovga con picoblaze\embedded_kcpsm3.ncd**
Preferences:	**embedded_kcpsm3.pcf**
Part:	**3s500efg320-4**
Data version:	**ADVANCED,v1.0,10-03-03**

Power summary:	**I(mA)**	**P(mW)**
Total estimated power consumption:		**81**
Vccint 1.20V:	**26**	**31**
Vccaux 2.50V:	**18**	**45**
Vcco25 2.50V:	**2**	**5**
Clocks:	**0**	**0**
Inputs:	**0**	**0**
Logic:	**0**	**0**
Outputs:		
Vcco25	**0**	**0**
Signals:	**0**	**0**
Quiescent Vccint 1.20V:	**26**	**31**
Quiescent Vccaux 2.50V:	**18**	**45**
Quiescent Vcco25 2.50V:	**2**	**5**

Thermal summary:	
Estimated junction temperature:	**27C**
Ambient temp:	**25C**
Case temp:	**26C**
Theta J-A range:	**26 - 26C/W**