ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

Facultad de Ingeniería en Electricidad y Computación

"ANALIZADOR DE PROTOCOLO SPI"

TESINA DE SEMINARIO

Previa a la obtención del Título de:

INGENIERO EN ELECTRÓNICA Y TELECOMUNICACIONES

Presentado por:

Diego Xavier Villegas Redrobán

José Enrique Dáger Pacheco

GUAYAQUIL – ECUADOR

AÑO 2013

AGRADECIMIENTO

Quiero agradecer a Dios, a mi familia sobre todo a mi madre que estuvo en cada momento de mi carrera apoyándome en cada uno de los momentos más importantes de mi vida, a mi padre por decirme que nunca me rinda ante cualquier adversidad, a mis hermanos por darme el apoyo necesario para que siga adelante y a cada una de las personas que conocí en este tiempo que seguimos el mismo sueño de convertirnos en ingenieros

Un especial agradecimiento al Ing. Ronald Ponguillo, por brindarnos su conocimiento en la realización de este proyecto

Diego Villegas Redrobán

Quiero agradecer a Dios, por darme toda la sabiduría, entendimiento e inteligencia para salir adelante en mi carrera, a mi familia, por confiar incondicionalmente en mí, a todas las personas que de una u otra manera han estado ahí para darme ganas de seguir adelante.

Agradezco al Ing. Ronald Ponguillo, por brindarnos su conocimiento para la culminación de éste proyecto.

José Enrique Dáger P.

DEDICATORIA

Dedico este proyecto a mi madre, a mi padre, a mis hermanos, a mis amigos y a cada una de las personas que creyeron en mí ya que sin ellos no estaría aquí cumpliendo una meta.

Diego Villegas Redrobán

Dedico éste proyecto a mi familia, especialmente a mi hermana Yessenia Yamile, a mis sobrinos Ubaldo Enrique y Evita Yamile, el amor que les tengo me inspira.

José Enrique Dáger P.

TRIBUNAL DE SUSTENTACIÓN

Ing. Ronald Ponguillo

PROFESOR DEL SEMINARIO DE GRADUACIÓN

Ing. Víctor Asanza

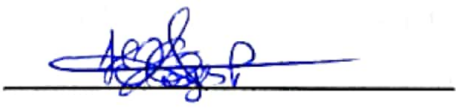
PROFESOR DELEGADO POR EL DECANO DE LA FACULTAD

DECLARACIÓN EXPRESA

"La responsabilidad del contenido de esta Tesina, nos corresponde exclusivamente; y el patrimonio intelectual del mismo a la ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL".

(Reglamento de exámenes y títulos profesionales de la ESPOL)

Diego Xavier Villegas Redrobán



José Enrique Dáger Pacheco

RESUMEN

El presente proyecto “Analizador de protocolo SPI” nos permite recolectar y analizar tramas de comunicación entre dispositivos que usen protocolo SPI.

El trabajo ha sido estructurado en 4 capítulos detallados a continuación:

En el capítulo 1, se trata de una manera muy breve los aspectos generales del proyecto que se busca implementar, así como los objetivos, la identificación del problema y una breve descripción de cómo va a ser resuelto el problema.

En el capítulo 2, se describe el fundamento teórico de los componentes de nuestro proyecto que incluye una descripción de cada uno de los elementos y la tecnología a utilizar. Se revisan conceptos importantes sobre Quartus II, NIOS II, FPGA´s y procesadores embebidos configurables. Así mismo se describe el funcionamiento del protocolo de comunicación SPI (Serial Peripheral Interface), y todo lo necesario para que se entienda de forma clara cuando se implemente el proyecto.

En el capítulo 3, se describe el diseño y la implementación del proyecto el cual va orientado a un ambiente de laboratorio donde cada una de las partes que componen el proyecto, están divididas en software y en hardware. El hardware consiste en el uso de la FPGA que contiene todos los componentes como el procesador, la memoria que va a ser creada para obtener los datos a partir de dispositivos externos que sean capaces de manejar protocolo SPI.

El software del proyecto se lo realiza en C para el núcleo del procesador NIOS II que se encuentra dentro de la FPGA.

En el capítulo 4, se realiza un análisis del funcionamiento del sistema, así como el análisis de los casos de prueba en diferentes dispositivos creados para medir las limitaciones y establecer las ventajas y desventajas del proyecto.

ÍNDICE GENERAL

ABREVIATURAS

ADC	ANALOG-TO-DIGITAL CONVERTER
ASIC	APPLICATION-SPECIFIC INTEGRATED CIRCUIT
CLK	CLOCK
CPLD	COMPLEX PROGRAMMABLE LOGIC DEVICE
CS	CHIP SELECT
CT	COMPUTED TOMOGRAPHIC
DDR	DOUBLE DATA RATE
FF	FLIP-FLOP
FPGA	FIELD PROGRAMMABLE GATE ARRAY
IDE	INTEGRATED DEVELOPMENT ENVIRONMENT
JTAG	JOINT TEST ACTION GROUP
LCD	LIQUID CRYSTAL DISPLAY
LED	LIGHT-EMITTING DIODE
MISO	MASTER INPUT SINGLE OUTPUT

MOSI	MASTER OUTPUT SINGLE INPUT
PAL	PROGRAMMABLE ARRAY LOGIC
PCB	PRINTED CIRCUIT BOARD
PET	POSITRON EMISSION TOMOGRAPHY
RRDY	RECEIVE READY
SCK	CLOCK
SDRAM	SYNCHRONOUS DYNAMIC RANDOM ACCESS MEMORY
SPI	SERIAL PERIPHERAL PROTOCOL
SS	SLAVE SELECT
TMT	TRANSMITTER SHIFT REGISTER READY
TRDY	TRANSMITTER READY
USB	UNIVERSAL SERIAL BUS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

En estos tiempos, las comunicaciones digitales han avanzado de tal manera que se han desarrollado innumerables protocolos de comunicación que se utilizan en distintas aplicaciones, pero con un objetivo en común, hacer más fáciles las cosas y mejorar la comunicación entre uno u otro dispositivo.

La importancia de analizar la información que se está enviando o recibiendo entre dispositivos en cualquier tipo de aplicación es un problema básico en la ingeniería en general, para monitoreo y control, detectar errores, comparar valores teóricos con valores prácticos, etc.

Para esto se han creado instrumentos que nos permiten analizar tal información, por ejemplo, para verificar si un amplificador de voltaje está amplificando cómo se espera, utilizamos un voltímetro que nos permite medir tanto el voltaje de entrada como de salida.

En comunicaciones, para verificar que los datos que se envían o reciben son los correctos, utilizamos un Analizador de Protocolo.

En nuestro proyecto construimos un Analizador de Protocolo SPI (Interfaz Serial Periférica), para lograr éste objetivo utilizamos la tarjeta de desarrollo de AlteraDE0_Nano que tiene embebida una FPGA(Arreglo de Puertas Programables por Campo) que puede ser programada o configurada de acuerdo a nuestras necesidades.

CAPÍTULO I

1. GENERALIDADES

En este capítulo se detalla de forma general el planteamiento del problema así como los límites y alcances del proyecto, objetivos y finalmente se busca la solución al problema planteado con la tecnología que estamos utilizando.

1.1 ALCANCE Y LIMITACIONES

El proyecto cumple con las siguientes características:

- Construimos un Analizador de Protocolo SPI que nos permite analizar las tramas de comunicación en cualquier tipo de dispositivo que esté usando éste protocolo.
- Nuestro analizador tiene 4 entradas las cuales corresponden a las líneas MOSI, MISO, chip select y reloj, éstas líneas son propias del Protocolo SPI, además tiene una entrada adicional que corresponde a la conexión de tierra.
- El caso base para nuestro estudio es un maestro y un esclavo.
- La implementación de las pruebas de éste proyecto se hicieron en un ambiente de laboratorio.

1.2. OBJETIVOS

1.2.1. OBJETIVOS GENERALES

- Construir un sistema embebido basado en el microprocesador NIOS II y bloques de lógica configurables que permita recolectar y analizar tramas en dispositivos que utilizan protocolo SPI.

1.2.2. OBJETIVOS ESPECIFICOS

- Entender el funcionamiento del protocolo SPI.
- Crear un hardware utilizando la herramienta QSYS que nos permita conectarnos con dispositivos que utilicen protocolo SPI.

- Crear un software utilizando la herramienta NIOS II IDE que nos permita analizar tramas de comunicación entre dispositivos que se comunican con protocolo SPI.
- Realizar pruebas con diferentes tipos de módulos comunicados con protocolo SPI para verificar el funcionamiento del proyecto.

1.3. ANTECEDENTES

Analizar tramas de comunicación entre dispositivos siempre ha sido un tema importante en las comunicaciones para: poder identificar la causa de algún problema, obtener datos acerca de nuestra red, hacer un seguimiento del funcionamiento de la red durante un período, identificar tendencias, comprobar conexiones generando paquetes de prueba haciendo un seguimiento de los resultados. Para cada una de estas características mencionadas existen los Analizadores de Protocolos.

En nuestro proyecto analizamos dispositivos que utilizan Protocolo SPI, para crear nuestro analizador utilizamos una FPGA que está dentro de una tarjeta de desarrollo de Altera (DE0-Nano).

Entonces, nuestro analizador va a estar embebido en una FPGA que es un dispositivo programable que está compuesto de bloques lógicos configurables y memoria embebida.

Un sistema embebido, el cual está dentro de un solo PCB, es un sistema muy complejo, es un microprocesador usado como un componente dentro de un sistema tecnológico (celulares, cámaras digitales, sistemas antibloqueos de frenos en automóviles, etc). Se dice también que la convergencia de muchas tecnologías y sistemas hace un sistema embebido.

1.4. IDENTIFICACIÓN DEL PROBLEMA

Cuando se requiere analizar, monitorear la información que se está enviando y/o recibiendo entre dispositivos o cuando se requiere detectar errores en la comunicación los cuáles se evidencian cuando nuestra aplicación no responde como se espera, es importante tener algún tipo de dispositivo que nos permita visualizar la información que se está procesando.

Para nuestro caso, el problema es analizar las tramas de comunicación entre dispositivos que utilizan protocolo SPI, para verificar que los datos que se envían y/o reciben son los correctos.

1.5. DESCRIPCIÓN BREVE DE LA SOLUCIÓN

Debido a las necesidades descritas, lo que se hace es utilizar un Analizador de Protocolo, en nuestro caso construimos un Analizador de Protocolo SPI con la DE0_Nano.

Lo que hicimos fue construir un módulo que nos permita recibir las 4 señales típicas de éste protocolo. El procesamiento de estas señales se las hizo por programación en NIOS II IDE, para recibirlas con protocolo SPI, procesarlas, y mostrarlas en la consola de NIOS II IDE.

CAPÍTULO II

2. MARCO TEÓRICO

En este capítulo se detalla, de una manera amplia, los principales conceptos de cada uno de los elementos que han sido utilizados en este proyecto, de tal manera que sea más fácil su comprensión y su uso al momento de ser leída. Se da información relevante sobre Quartus II 12.1, la herramienta de trabajo QSYS, NIOS II IDE, FPGA's, Procesadores Embebidos y sobre todo acerca del Protocolo SPI.

2.1. TARJETAS DE DESARROLLO Y EDUCACIÓN DE ALTERA

Las tarjetas de educación fueron desarrolladas por Altera para satisfacer las necesidades de la enseñanza de sistemas digitales. La plataforma provee de las

herramientas necesarias para la implementación de diseños lógicos que nos permitirá crear distintas y eficientes aplicaciones que incluyen:

- Funciones de alta calidad que disponen de una variedad de dispositivos de memoria, de audio y vídeo, así como de conectividad Ethernet y USB.
- PCB's de alta calidad con circuitos de protección para evitar daños en los conectores de E/S.
- Elementos de entrada y salida como botoneras, LED´s, interruptores e interfaces comúnmente utilizadas.

2.1.1. DE0 NANO

DE0-Nano de Altera introduce una plataforma desarrollada de FPGA de tamaño compacto adecuado para diseños de prototipos tales como robots y proyectos "portables" [16].

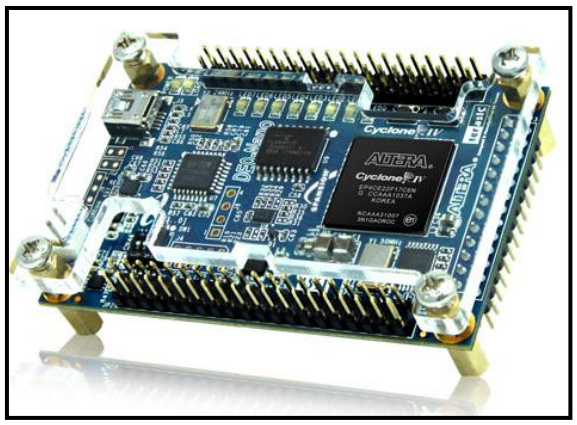


Figura 2.1 Tarjeta de desarrollo DE0-Nano de Altera

La DE0-Nano tiene una colección de interfaces que incluyen TRES conectores de expansión llamadas GPIO, que nos permite realizar diseños fuera de la tarjeta de Altera.

Así mismo posee periféricos de usos generales como LED's, pushbuttons y switches, para realizar procesos dentro de la tarjeta que permita ilustrar procedimientos básicos sobre el manejo de la DE0-Nano.

DE0-Nano cuenta con las siguientes características [16]:

Características	**Descripción**
FPGA	Cyclone IV EP4CE22F17C6 con EPCS64. USB-Blaster para la configuración de la FPGA 153 pines de E/S.
Interfaces E/S	Dos conectores de expansión de 40 pines (GPIO's) que proveen 72 pines de E/S. Pines de energía de 5V. Dos pines de suministro de energía de 3.3 V. 4 pines destinados a tierra.
Switches, LED's y pulsadores	8 LED's verdes. 2 Pulsadores. 1 Switch DIP de 4 posiciones.
Dispositivos de Memoria	SDRAM de 32 MB. EEPROM I2C de 2KB.
Clock	Oscilador de 50 MHz.
Convertidor A/D	Convertidor A/D (NS ADC128S022) de 12 bits de 8 canales que muestrea de 50 Ksps a 200 Ksps.

Suministro de Energía	Puerto USB tipo mini-AB (5V). Pin DC de 5V para cada GPIO. 2 pines externos (3.6 – 5.7V).
G-Sensor	Acelerómetro ADI ADXL345 de 3 ejes de alta resolución (13 bits).

Tabla 2.1 Características de la DE0 Nano.

2.2. FPGA

2.2.1. DEFINICIÓN

Un FPGA es un circuito integrado que puede configurarse para llevar a cabo cualquier función lógica y realizar funciones, donde el diseñador tiene el control de lo que se va a crear. La lógica programable de este dispositivo puede reproducir desde funciones tan sencillas, como las llevadas a cabo por una puerta lógica o un sistema combinatorial, hasta complejos sistemas en un chip [1].

Para poder conseguir dichas funciones, el diseñador debe configurar el circuito, normalmente siguiendo la especificación de un lenguaje de descripción de hardware, es algo así como “hacer electrónica digital a través de códigos” [2].

Todas las FPGA’s, independientemente del fabricante, tiene ciertos elementos en común, tienen un arreglo matricial de elementos lógicos, como los flip-flops y lógica combinatorial, que se configuran usando cierta tecnología de programación.

La programación de las interconexiones y de los elementos lógicos pueden o no ser permanentes, eso depende de la tecnología de programación usada.

En la siguiente figura se puede ver la composición y disposición de los componentes lógicos de una FPGA. Como se puede observar está compuesto de un arreglo matricial, interconexiones y bloques dedicados a funciones específicas como memoria, procesadores de señales, control de reloj, entre otras [3].

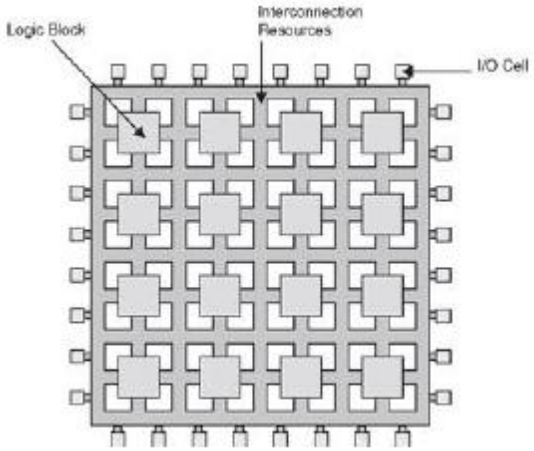


Figura 2.2 Estructura y componentes de una FPGA Altera

Las FPGA's se utilizan en aplicaciones similares a los ASIC's. Sin embargo, son más lentas, tienen un mayor consumo de potencia y no pueden abarcar sistemas tan complejos como ellos. A pesar de esto, las FPGA's tienen las ventajas de ser reprogramables (lo que añade una enorme flexibilidad al flujo de diseño), sus costes de desarrollo y adquisición son mucho menores para pequeñas cantidades de dispositivos y el tiempo de desarrollo es también menor.

Históricamente las FPGA surgen como una evolución de los conceptos desarrollados en las PAL y los CPLD [1].

2.2.2. CARACTERÍSTICAS

Las FPGA's son dispositivos orientados a satisfacer una muy amplia gama de aplicaciones, desde simple lógica combinatorial hasta sistemas con microprocesadores embebidos, transmisiones tipo Ethernet, transmisión de datos seriales a 3.5Gb/s, todos con el mismo dispositivo. Por esta razón las FPGA's poseen diversas características y entre sus principales se encuentran [3]:

- Control de impedancia programable por cada terminal de E/S.
- Gran cantidad de terminales de E/S, Desde 100 hasta 1400 terminales de E/S.
- Buffers de E/S programable: control de corriente, configuración de E/S, resistores pull-up y resistores pull-down configurables.
- Gran cantidad de Tablas de Búsqueda (Look-up tables) pueden llegar a 100000 o incluso más.
- Gran cantidad de Flip-Flops, los dispositivos más grandes podrían tener 40000 FF's o más.
- Procesador embebido en hardware.
- Interface DDR/DDR2 SDRAM que soporta interfaces de hasta 800 Mb/s.
- Transceptores para transmisión serie de muy alta velocidad.

2.2.3. APLICACIONES

Las FPGA's poseen un amplio rango de aplicaciones, debido a la versatilidad y a la flexibilidad de estos dispositivos, siendo así que una de sus principales aplicaciones se centra en el procesamiento digital de señales (DSP) en el campo de las telecomunicaciones. La elección de estos equipos se da gracias a su alta frecuencia de trabajo, a su capacidad de procesamiento en paralelo, y a su bajo precio en comparación con los ASIC's.

Otras de las aplicaciones que se dan gracias al uso de las FPGA's son [17]:

Sistemas de visión artificial: Actualmente existen cada vez más dispositivos que disponen de un sistema de visión artificial como cámaras de vigilancia, robots, etc. Muchos de estos dispositivos necesitan de sistemas para conocer su posición, reconocer objetos de su entorno, reconocer rostros de personas, y poder interactuar con ellos de forma adecuada.

Sistemas de imágenes médicas: En el mundo de la medicina se están utilizando cada vez más las FPGA's para el tratamiento de imágenes biomédicas obtenidas mediante procesos de PET, escáner CT, rayos X, imágenes tridimensionales, etc. Estos nuevos sistemas de requieren de mayor resolución y capacidad de procesamiento.

Reconocimiento de Voz: Esta técnica es empleada en el ámbito de la seguridad y recuperación de información, lo que resulta muy eficiente cuando se trabaja con

FPGA, ya que permite hacer una comparación de voz con un patrón previamente almacenado.

Codificación y Encriptación: La seguridad de la información es un ámbito importante cuando se trata de enviar un e-mail o realizar una compra por internet y aún más cuando tratamos temas de ámbito militar, aeronáutico y gubernamental. Las FPGA's aportan un gran apoyo en este terreno ya que pueden manejar grandes volúmenes de información para realizar operaciones complejas a muy alta velocidad.

Figura 2.3 Aplicaciones de las FPGA's

2.2.4. FPGA CYCLONE IV

La familia de dispositivos Cyclone IV ofrece las siguientes características [4]:

- Bajo costo y bajo consumo de potencia.
- De 6K a 150K elementos lógicos.
- Hasta de 594 Kbits de bloques de memoria embebida.
- Hasta 360 multiplicadores 18 x 18 para aplicaciones de procesamiento DSP.
- Bloques embebidos de Memoria M9K.

Los dispositivos ofrecen más de 8 transceptores de alta velocidad que proveen:

- Más de 3.125 Gbps de velocidad de datos.
- Inactividad Eléctrica o Idle.
- Canal dinámico de reconfiguración permitiendo el cambio de la velocidad de dato.
- Consumo de potencia de 150 mW por canal.
- Estructura de reloj flexible, para soportar múltiples protocolos en un sólo bloque transceptor.

Los dispositivos Cyclone IV E ofrecen un amplio rango de protocolos:

- Puertos Display (2.7 Gbps).
- Hasta 532 puertos de E/S dedicados al usuario.
- Soporte para interfaces DDR2 SDRAM de 200 MHz o más.
- Soporte para QDRII SRAM y DDR SDRAM de 167 MHz o más.

- Ofrecido en ambientes comerciales e industriales.

2.3. SISTEMAS EMBEBIDOS

Podemos decir que un sistema embebido consiste en un sistema basado en un microprocesador, cuyo hardware y software están específicamente diseñados y optimizados para resolver un problema concreto de forma eficiente [5]. Normalmente un sistema embebido interactúa continuamente con el entorno para vigilar o controlar algún proceso mediante una serie de sensores. Su hardware se diseña generalmente a nivel de chips o de tarjeta PCB, buscando minimizar el tamaño, el coste y maximizar el rendimiento y la fiabilidad para una aplicación particular [6].

Por ejemplo, en el caso de una lavadora, esta se compone de un chasis, de motores eléctricos y bombas de agua, de un frontal con varias teclas para que el usuario pueda poner en marcha la lavadora, contiene temporizadores y válvulas eléctricas que controlan el flujo del agua, y más componentes.

En este conjunto nos podemos fácilmente imaginar la necesidad de un circuito electrónico que contenga los diferentes programas de lavado del que disponga el electrodoméstico. Y por razones obvias, esta electrónica no sería otra cosa que un micro-ordenador, especialmente diseñado para dicho fin. Por lo tanto podemos hablar de un sistema embebido o empotrado en el interior de la lavadora [7].

2.4. MICROPROCESADOR EMBEBIDO NIOS II

2.4.1. INTRODUCCIÓN

Altera proporciona complejas infraestructuras que nos permiten crear una serie de sistemas embebidos, de acuerdo a las necesidades del diseñador, por medio de la combinación de una serie de componentes configurables dentro de sus FPGA's.

Así mismo Altera proporciona un entorno denominado QSYS, que permite la configuración a medida de la máquina basada en NIOS II, y gracias a las herramientas de QUARTUS II puede ser implementado sobre una FPGA.

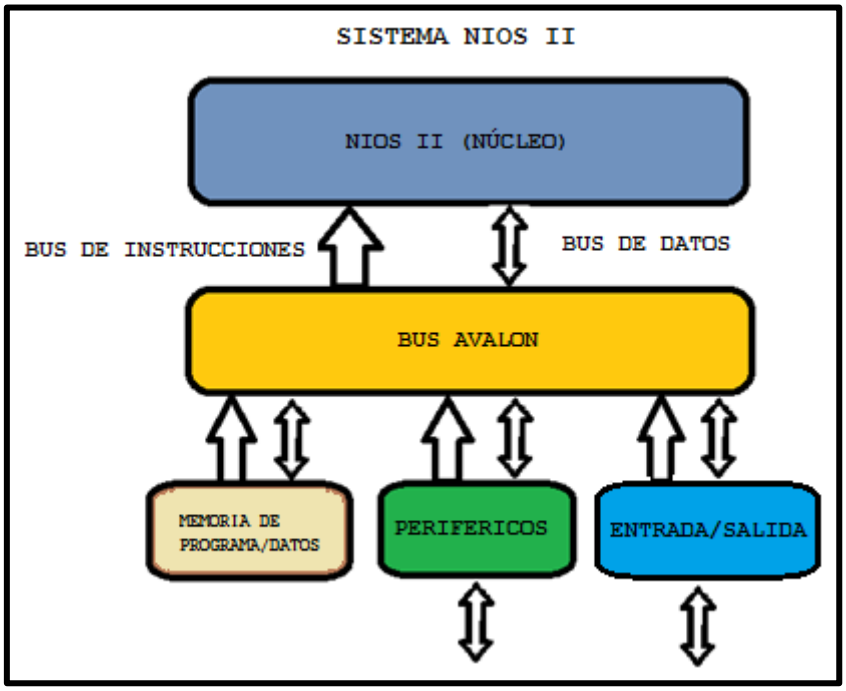


Figura 2.4 Microcontrolador Basado en NIOS II

El microcontrolador está compuesto por [8]:

- El núcleo procesador NIOS II.
- Memoria interna de programa y de datos.
- Periféricos integrados.
- Interfaces de entrada y salida para memorias externas.

Un microcontrolador basado en NIOS II es un sistema muy flexible que se adapta a las necesidades de cada diseño, no sólo porque se ajusta al tamaño del dispositivo, sino que deja al diseñador decidir cuando una implementación se debe realizar en software o en hardware, permitiendo elevar el rendimiento y ajustar los costos.

2.4.2. VERSIONES DEL PROCESADOR NIOS II

NIOS II es un procesador de 32 bits de propósitos generales diseñado por Altera. NIOS II incorpora muchas mejoras sobre la arquitectura NIOS original, haciéndola más adecuada para aplicaciones embebidas computarizadas de mayor complejidad para un sistema de control DSP.

El procesador NIOS II de Altera es uno de los procesadores más versátiles porque puede ser usado con una variedad de otros componentes de manera que se pueda crear sistemas completos y funcionales a través de este procesador.

NIOS II posee tres diferentes configuraciones con arquitectura Harvard que son [:

NIOS II/f (“fast”): Está diseñado para un máximo rendimiento a costa del tamaño del núcleo. Así mismo posee un “pipeline” de 6 etapas que proporciona opciones para mejorar su desempeño.

NIOS II/s (“standard”): Está diseñado para mantener el balance entre rendimiento y costo. Posee un “pipeline” de 5 etapas.

NIOS II/e (“economy”): Está diseñado para usar la menor cantidad de recursos de la FPGA, tal es así que carece de las operaciones de multiplicación y división.

2.4.3. CARACTERÍSTICAS DEL PROCESADOR NIOS II

Dado que NIOS II presenta arquitectura Harvard, presenta buses separados para instrucciones y datos cuyas características principales son [9]:

- Tamaño de palabra asignado de 32 bits.
- Juego de instrucciones RISC de 32 bits.
- 32 registros de propósito general de 32 bits (r0 – r31).
- 6 registros de control de 32 bits (ctl0 - ctl5).
- 32 fuentes de interrupción externa.
- Capacidad de direccionamiento de 32 bits.

- Operaciones de multiplicación y división de 32 bits.
- Instrucciones dedicadas para multiplicaciones de 64 y 128 bits.
- Instrucciones para operaciones de coma flotante en precisión simple.
- Acceso a variedad de periféricos integrados e interfaces para manejo de memorias y periféricos.

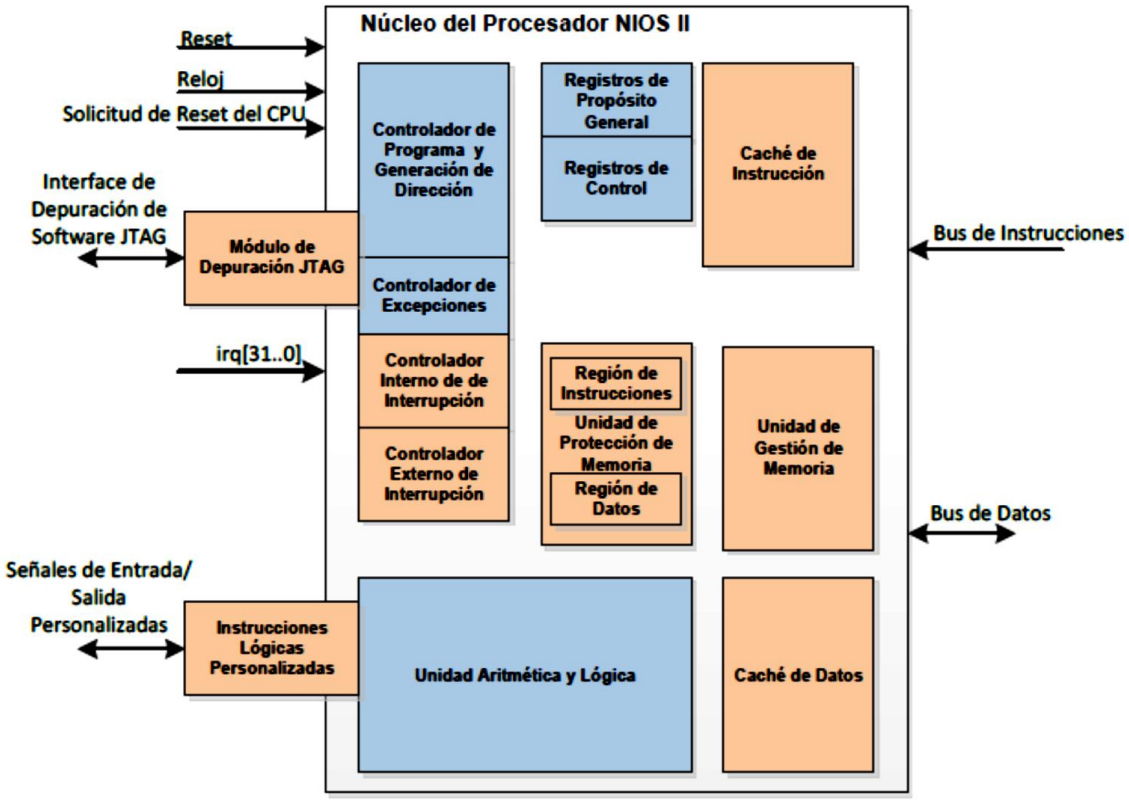


Figura 2.5 Arquitectura Interna del Procesador NIOS II

La figura muestra la arquitectura interna del microprocesador NIOS II en la cual se puede observar la unidad lógica aritmética, la región donde se procesan las instrucciones, controladores internos entre otras.

2.5. QUARTUS II 12.1

Este software incluye el procesador NIOS II y la herramienta de creación QSYS, para facilitar las implementaciones de sistemas complejos. Además, ofrece una interfaz gráfica de usuario así como una interfaz de comandos para cada una de las fases de diseño. La interfaz gráfica del software se presenta a continuación:

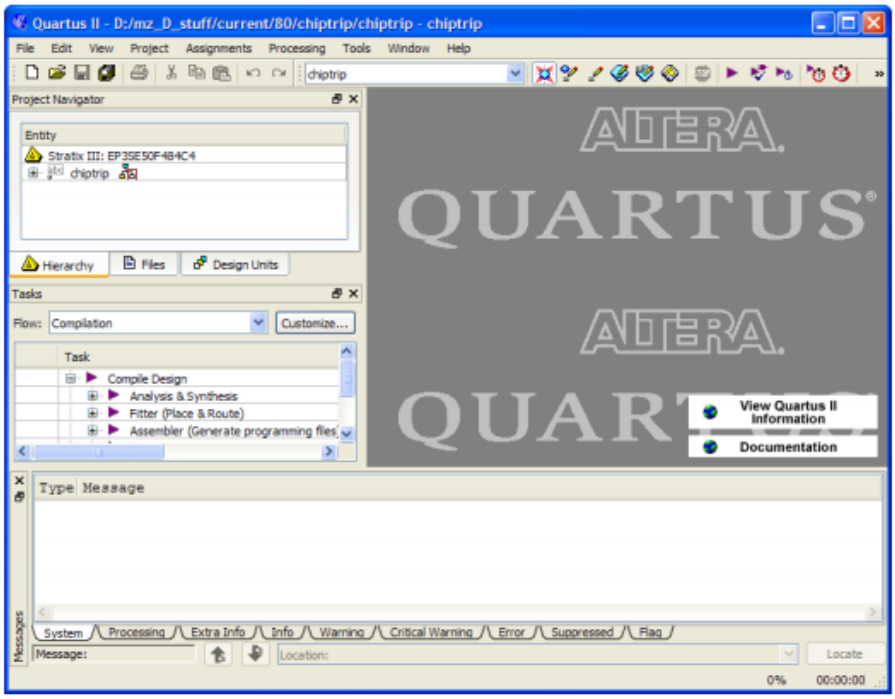


Figura 2.6 Interfaz gráfica de Quartus II 12.1

Quartus II nos permite realizar varias tareas al momento de la simulación de un circuito lógico, desde la descripción de los componentes del circuito en lenguaje VHDL, hasta la simulación de diagramas de tiempo del comportamiento del circuito. Quartus II nos ofrece muchas ventajas al momento de desarrollar un proyecto usando el microprocesador NIOS II ya que con la herramienta integrada QSYS podemos generar el hardware que requerimos para la implementación.

2.5.1. QSYS

Qsys es una nueva herramienta de integración de sistemas que nos ahorra tiempo y esfuerzo en el diseño de procesos en la FPGA, generando de forma automática interconexiones lógicas para conectar funciones y subsistemas de propiedad intelectual [19].

Qsys es la siguiente generación de la herramienta SOPC Builder que viene potencializada por una nueva tecnología de FPGA brindando un mayor rendimiento, mejor reutilización del diseño y una verificación más rápida en comparación con SOPC Builder [19].

Se puede decir que Qsys nos permite crear una computadora básica acorde a las necesidades del diseñador donde además de poder fabricar nuestros propios módulos, podemos quitar elementos dentro de la computadora que no van a ser utilizados de manera que al momento de realizar la compilación sea mucho más rápida y eficiente.

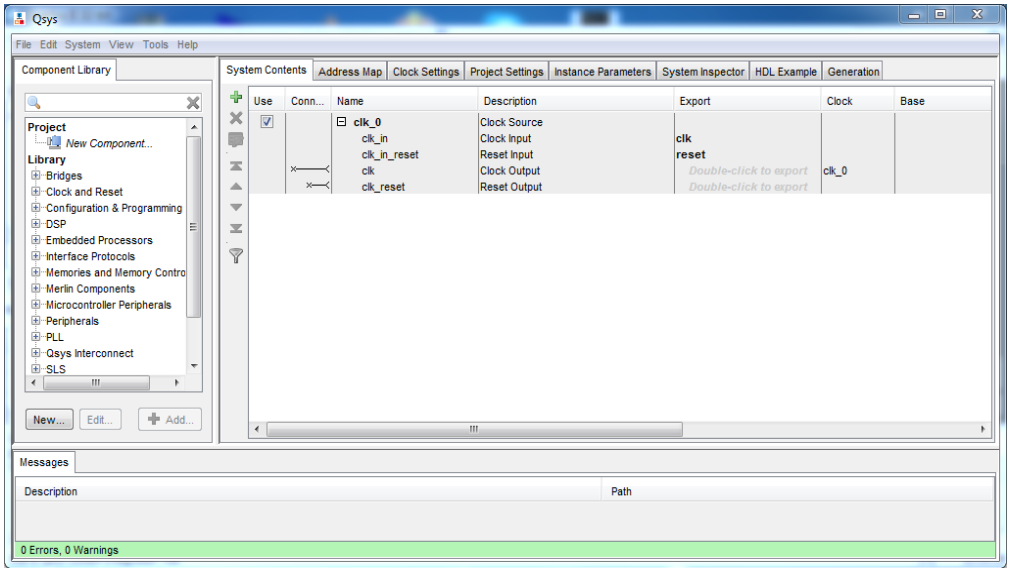


Figura 2.7 Ventana de QSYS

2.5.2. PROGRAMADOR DE QUARTUS

El programador de Quartus forma parte del diseño completo de Altera, donde se permite descargar nuevas configuraciones dentro de las FPGA's.

Luego de compilar el diseño gracias al software de Quartus II, podemos utilizar el programador de Quartus II para configurar nuestro dispositivo para comprobar su funcionalidad en un PCB [13].

Modos de Programación y Configuración

El programador de Quartus II puede soportar cinco clases de configuración, incluyendo JTAG, Pasivo Serial (PS), Activo Serial (AS), Configuración vía Protocolo (CvP), y modo In-Socket (ISM) [13].

Modo	FPGA	CPLD	Configuración del Dispositivo	Configuración del Dispositivo Serial
JTAG	✓	✓	✓	--
PS	✓	--	--	--
AS	--	--	--	✓
CvP	✓	--	--	--
Programación In-Socket	--	✓	✓	✓

Tabla 2.2 Lista de Programación y Configuración soportada por dispositivos Altera

Diseño de Claves de Seguridad

De igual manera Quartus II apoya la generación de programas de encriptación de archivos, configuración encriptada para las FPGA's de Altera, lo que incrementa el

diseño en las características de seguridad. Se puede usar también Quartus II para programar claves encriptadas dentro de las FPGA's.

Programación Opcional o Configuración de Archivos

El software de Quartus II puede generar cierto tipo de programación o ciertos archivos configurables en varios formatos que pueden ser usados dentro de otras herramientas programables que no tenga ninguna relación con el Programador de Quartus II. Al momento de realizar la compilación del diseño, el Ensamblador se encarga de generar automáticamente un archivo **.pof** o .**sof**por defecto [13].

Tipo de Archivo	**Generado por Software Quartus II**	**Soportado por Programador Quartus II**
.sof	✓	✓
.pof	✓	✓
.jam	✓	✓
.jbc	✓	✓
Configuración indirecta de archivo JTAG (.jic)	✓	✓
Archivo de Formato Vector Serial (.svf)	✓	--
Archivo de Configuración In-System (.isc)	✓	--
Archivo de Salida Hexadecimal (Intel-Format)(.hexout)	✓	--
Archivo Binario (.rbf)	✓	--
Archivo Binario para	✓	--

Reconfiguración Parcial (.rbf)		
Archivo de Text Tabular (.ttf)	✓	--
Archivo de Programación de Datos (.rpd)	✓	--

Tabla 2.3 Tipos de archivos generados por software Quartus II y soportados por el Programador Quartus II

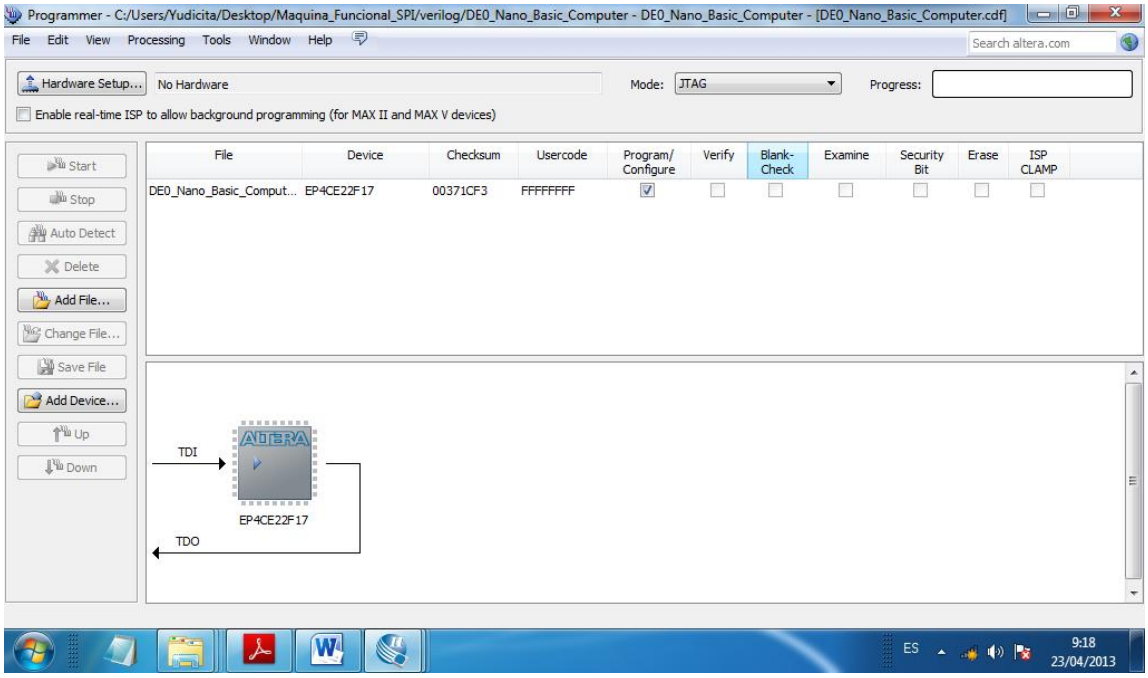


Figura 2.8 Ventana del Programador de Quartus

2.5.3. PIN PLANNER

La herramienta Pin Planner de Quartus II nos permite visualizar, planificar y asignar los pines de E/S del dispositivo con el que se está trabajando de una manera gráfica. Puede ser utillizado para una rápida localización de los pines tal como los pines de E/S, pines de suministro de energía VREF, pines DQ/DQS, pines de entrada de reloj, pines de disco duro. Y una vez que se define y se asigna cada uno de los pines a usar, se puede generar el archivo de diseño de alto nivel que será incluido en el proyecto [18].

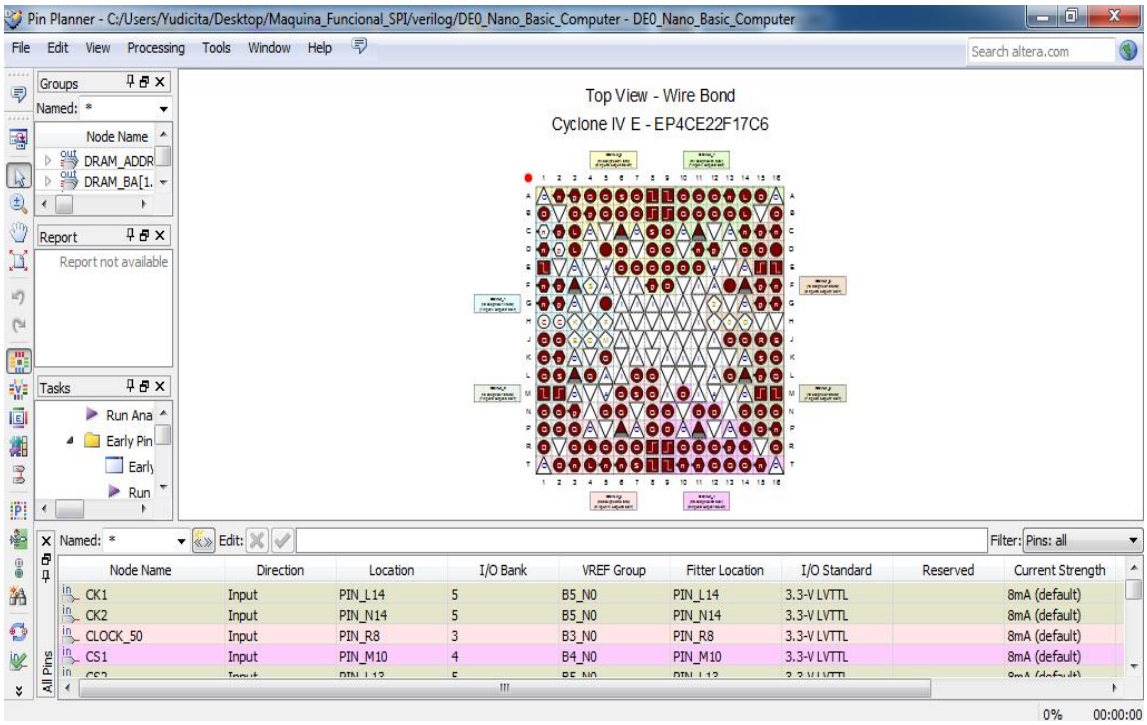


Figura 2.9 Ventana de Pin Planner

2.5.4. UNIVERSITY PROGRAM

La herramienta UniversityProgram de Altera provee de un completo apoyo para introducir a los estudiantes a la tecnología digital. Este tipo de apoyo incluye hardware, software y materiales de enseñanza. El hardware utilizado consiste en unas tarjetas de desarrollo y educación (DE0, DE1, DE2-115, DE4) diseñadas especialmente para investigación y enseñanza, para desarrollar prototipos de nuevas aplicaciones [11].

Los diseños de lógica digital que los ingenieros enfrentan hoy en día, como el aumento en la densidad de dispositivos o la complejidad en sus diseños obligan, a tener acceso a las últimas herramientas de hardware y software para desarrollar plataformas con los últimos avances en tecnología [12].

2.6. NIOS II IDE

NIOS II IDE es una interfaz de software desarrollado para el procesador NIOS II. Todas las tareas de desarrollo de software pueden ser realizadas dentro del IDE, incluyendo la edición, creación y depuración de programas.

NIOS II IDE está basado en el marco de Eclipse IDE y en las Herramientas de Desarrollo C/C++. NIOS II IDE hereda gran parte de su comportamiento de Eclipse, en la que se incluye los conceptos de workspaces, perspectivas y viewpoints [14].

Eclipse posee una interfaz gráfica llamada *workbench* donde cada ventana de trabajo de Eclipse contiene uno o más viewpoints. Ofrece también un conjunto de capacidades dirigido a cumplimiento de un determinado tipo de tarea.

Por ejemplo, Nios II C/C++ proporciona herramientas para la compilación de proyectos en Nios II C/C++.

Tener checkpoints en NIOS II, nos ayudan a organizar toda la información. Por ejemplo, Los registros de revisión dentro de la depuración permiten revisar y editar los valores de registros del procesador, mientras que la depuración de un proyecto examina cada archivo presente dejando así todo listo para que sea programado en la FPGA [14].

Proyectos y Workspaces en NIOS II IDE

Los Proyectos dentro de Nios II IDE, son un grupos de archivos tratados como una unidad, que contienen dentro de sí el código fuente, make files, object files, librerias y otros archivos relacionados. Estos proyectos contienen todos los recursos que se necesitan para crear, construir, ejecutar y depurar el Nios II IDE. Se puede crear toda clase de códigos fuente en C/C++ así como búsquedas de archivos en proyectos mediante la creación de un índice de este mismo proyecto.

Nios II IDE almacena sus proyectos en un directorio llamado un espacio de trabajo o *workspace*, donde se puede definir uno o más espacios de trabajo, seleccionar el área de trabajo que se utilizará para la sesión de IDE que está en uso todo gracias al cuadro de diálogo *Workspace Launcher*[14].

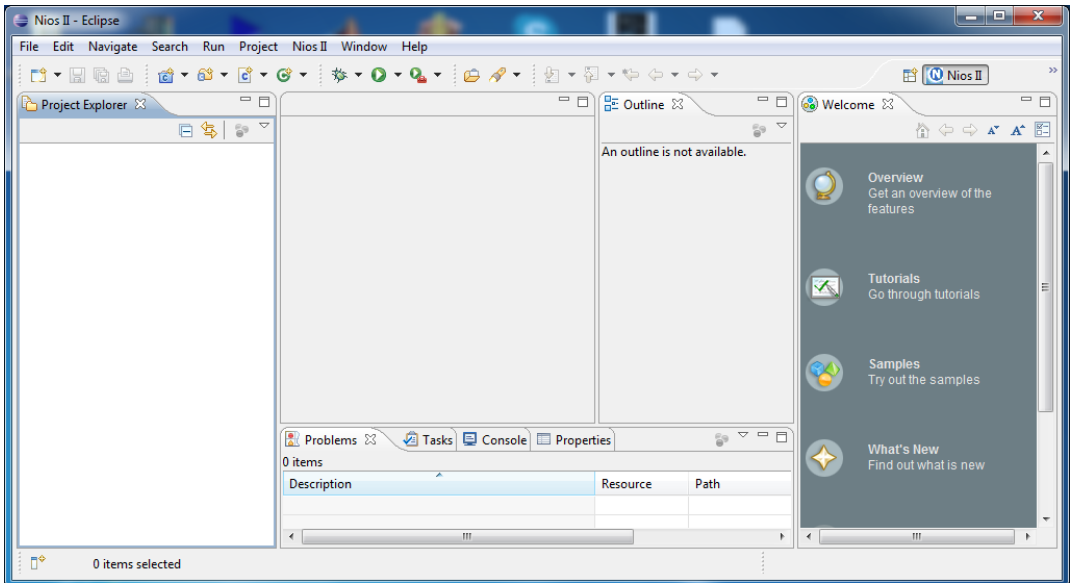


Figura 2.10 Ventana de NIOS II IDE

2.7. PROTOCOLO SPI

SPI es una interfaz serial sincrónica de propósito general. Durante una transferencia SPI, los datos transmitidos y recibidos son simultáneamente desplazados hacia afuera y hacia adentro serialmente. Una línea serial de reloj sincroniza el desplazamiento y muestreo de la información en dos líneas de datos seriales.

Motorola creo el puerto SPI a mediados de 1980 para ser utilizados en sus microcontroladores. El protocolo SPI es principalmente utilizado para permitir a los microcontroladores comunicarse con dispositivos periféricos.

Los dispositivos que usan protocolo SPI, se comunican utilizando una relación maestro-esclavo. Debido a que no tiene direccionamiento de dispositivos, SPI requiere más esfuerzo y más recursos de hardware que I2C cuando se trabaja con más de un esclavo. Pero SPI tiende a ser más simple y más eficiente que I2C en aplicaciones que se comunican punto a punto (un maestro un esclavo) por la misma razón, SPI no tiene direccionamiento de dispositivos, lo que significa que es más simple.

2.7.1. DETALLE DE SPI

Las interfaces SPI están disponibles en la comunicación de populares procesadores y microcontroladores. Es una comunicación de datos sincrónica serial que opera en full-duplex (la información puede enviarse y recibirse al mismo tiempo).

Los dispositivos se comunican utilizando una relación maestro-esclavo, en la cual el maestro inicia la trama de datos. Cuando el maestro genera el reloj y selecciona un dispositivo esclavo, los datos pueden ser enviados en una o ambas direcciones simultáneamente. En efecto, hasta donde SPI le concierne, los datos están siempre transfiriéndose en ambas direcciones. Les corresponde al maestro y al esclavo saber si un dato recibido es válido o no.

Entonces un dispositivo debe descartar el dato recibido en una trama de sólo transmisión o generar un byte “muerto” para una trama de sólo recepción.

El protocolo SPI es usado a distancias cortas o nivel de PCB, debido a que como es un protocolo que trabaja a alta frecuencia, quiere decir que a distancias largas y altas frecuencias, interviene el factor ruido.

2.7.2. SEÑALES DEL PROTOCOLO SPI

SPI especifica 4 señales:

- **Master Out Slave Input (MOSI):** Salida de datos desde el maestro hacia las entradas de los esclavos.
- **Master Input Slave Output (MISO):** Salida de datos desde el esclavo hacia el maestro.
- **Serial Clock (SCK):** Reloj manejado por el maestro hacia los esclavos, utilizados para sincronizar los datos.
- **Slave Select (SS):** Señal de selección manejada por el maestro hacia esclavos individuales, utilizados para seleccionar el esclavo objetivo.

La figura 2.12 muestra éstas señales en una configuración de un maestro un esclavo. El reloj es generado por el maestro, MOSI manda datos desde el maestro al esclavo, MISO envía datos desde el esclavo al maestro, un dispositivo esclavo es seleccionado cuando el master actica la señal de selector de esclavo.

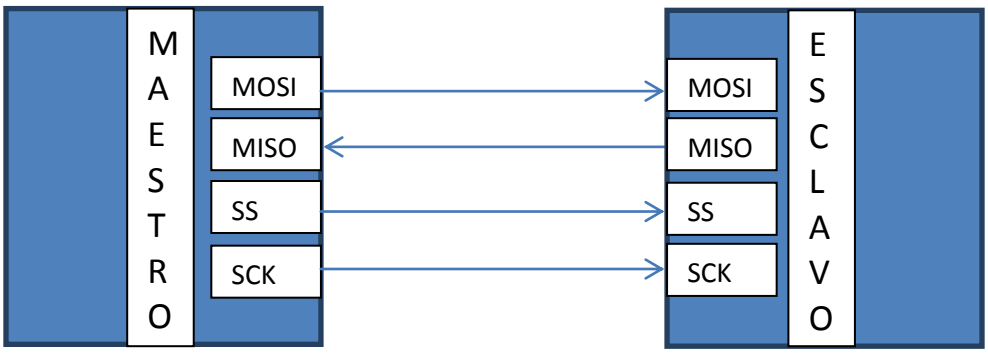


Figura 2.11 Comunicación SPI entre un maestro un esclavo

Con el selector de esclavo (SS), el correspondiente dispositivo periférico es seleccionado. Éste pin es activo en bajo. En el estado en que no está activo la señal MISO está en alta impedancia. El maestro decide con cual dispositivo periférico quiere comunicarse.

El reloj (CLK), está activo esté o no seleccionado el esclavo, el reloj sirve como sincronizador en la comunicación.

La mayoría de los dispositivos SPI tienen éstas 4 líneas. Hay dispositivos que tienen la señal MISO y la señal MOSI multiplexadas, por ejemplo, el sensor de temperatura LM74 de National Semiconductor. En otros dispositivos no tiene una de las cuatro líneas, por ejemplo, en algunos ADC's la línea MOSI no está y en los LCD que trabajan con éste protocolo la línea MISO no está [10].

CAPÍTULO III

3. DISEÑO E IMPLEMENTACIÓN

3.1. DISEÑO E IMPLEMENTACIÓN DE HARDWARE

En ésta sección describimos el diseño y la implementación del hardware de nuestro proyecto: especificaciones, diagrama de bloques e implementación en QSYS.

3.1.1. ESPECIFICACIONES

Con la herramienta de Quartus 12.1, QSYS, diseñamos en la FPGA un hardware que nos permite conectarnos con cualquier dispositivo que utilice protocolo SPI.

3.1.2. DIAGRAMA DE BLOQUES

3.1.2.1. DIAGRAMA DE BLOQUES GENERAL

El siguiente diagrama de bloques muestra, de manera simplificada, una configuración típica de dos dispositivos que utilizan protocolo SPI, y como debe ser conectado nuestro analizador.

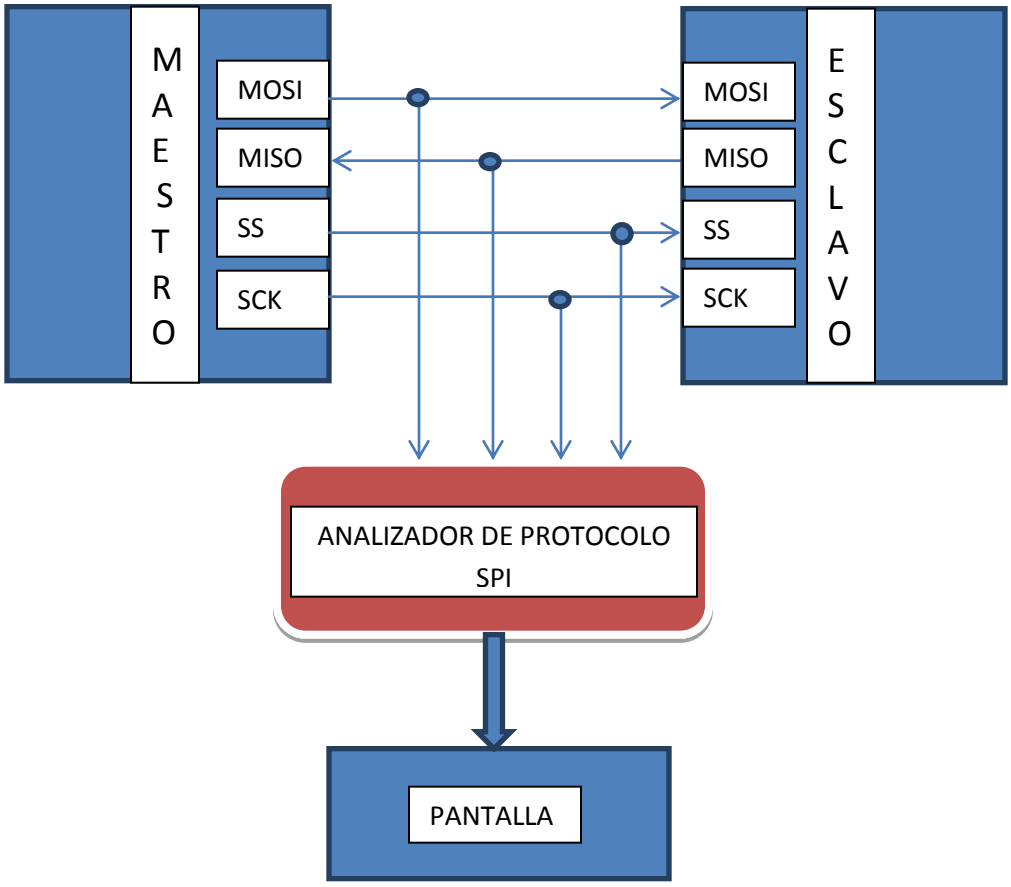


Figura 3.1: Diagrama de bloques general

3.1.2.2. DIAGRAMA DE BLOQUES DEL ANALIZADOR DE PROTOCOLO SPI

El siguiente diagrama de bloques muestra, de manera detallada, la configuración del hardware de nuestro analizador creado en QSYS.

La estrategia empleada para crear nuestro analizador es la siguiente:

- Nuestro analizador solo puede “escuchar” lo que se están diciendo entre maestro y esclavo, o sea, no va a enviar información, para esto necesitamos dos entradas MOSI, para conseguir esto se creó dos módulos SPI configurados como esclavo.
- Las líneas de dato (MOSI y MISO) de la configuración a analizar, deben estar conectadas a las entradas MOSI1 y MOSI2 que corresponden a los módulos SPI esclavos creados para hacer nuestro analizador
- Las dos salidas MISO del analizador no las usamos.
- Las líneas de control (relojes y los selectores de esclavo) de los 2 módulos SPI creados para hacer nuestro analizador deben estar conectados a las líneas de control (reloj y selector de esclavo) del sistema que vamos a analizar.
- La tierra de la configuración a analizar debe estar conectada a la tierra de nuestro analizador.
- Manipulamos por software los datos “escuchados” para imprimirlos en la consola de NIOS II IDE.

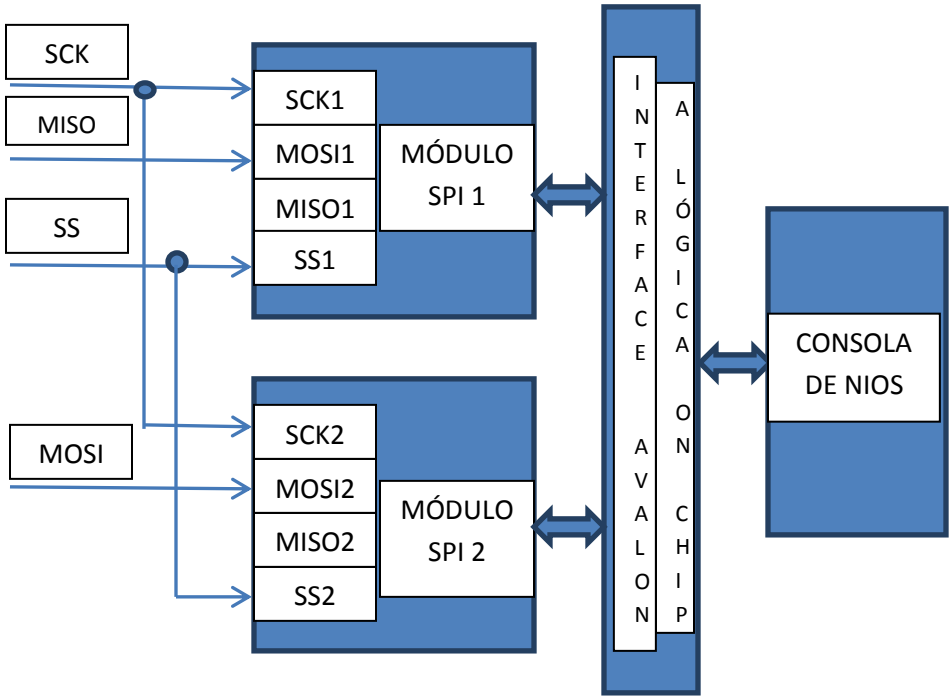


Figura 3.2: Diagrama de bloques del Analizador de Protocolo SPI

3.1.3. IMPLEMENTACIÓN DE HARDWARE EN QSYS

Para implementar nuestro hardware en QSYS, nos basamos en la computadora básica de la DE0 nano que viene preconfigurada cuando instalamos el UniversityProgram. Agregamos dos módulos SPI a la computadora básica, y luego quitamos todos los módulos que no vamos a usar, para que nos quede un hardware específico para analizar información entre dispositivos que usan protocolo SPI.

Seguimos los siguientes pasos para crear nuestra máquina:

- Abrimos la computadora básica de la DE0 nano.
- Abrimos la herramienta QSYS en Quartus II 12.1.
- Abrimos el archivo .qsys que se encuentra en la carpeta verilog de la computadora básica de la DE0 nano.

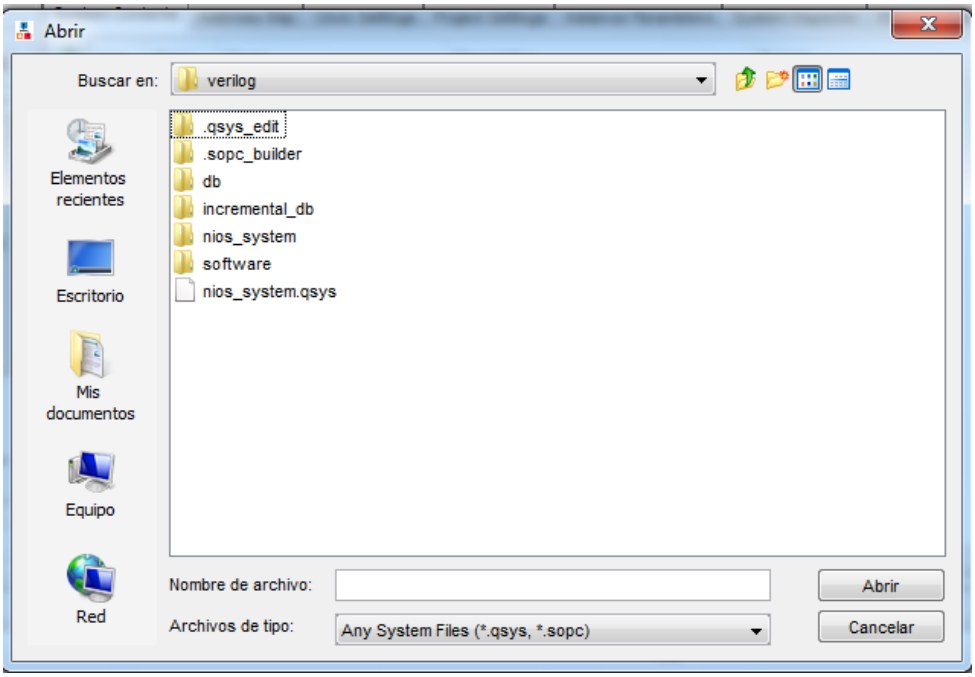


Figura 3.3: Carpeta que contiene el archivo .qsys

- Aquí nos saldrá la ventana de QSYS con todas las componentes de la computadora básica de la DE0 nano.

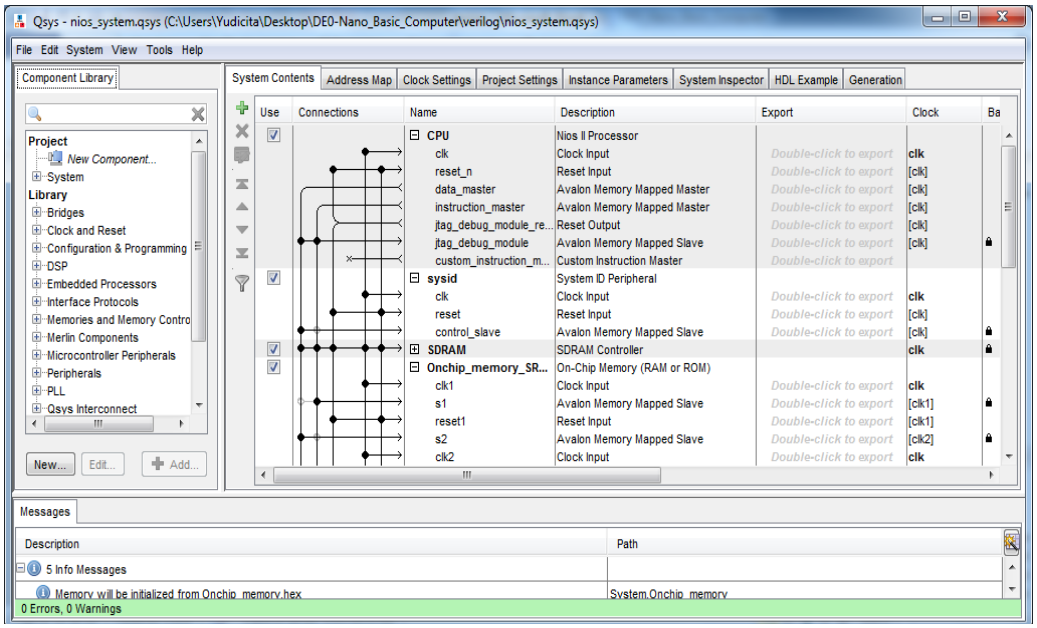


Figura 3.4: Ventana de Qsys de la computadora básica de la DE0 Nano

- Ahora nos vamos a la parte de Library/Interface Potocols/Serial/SPI (3 Wire Serial), hacemos doble click aquí en SPI (3 Wire Serial) y luego nos saldrá ésta ventana:

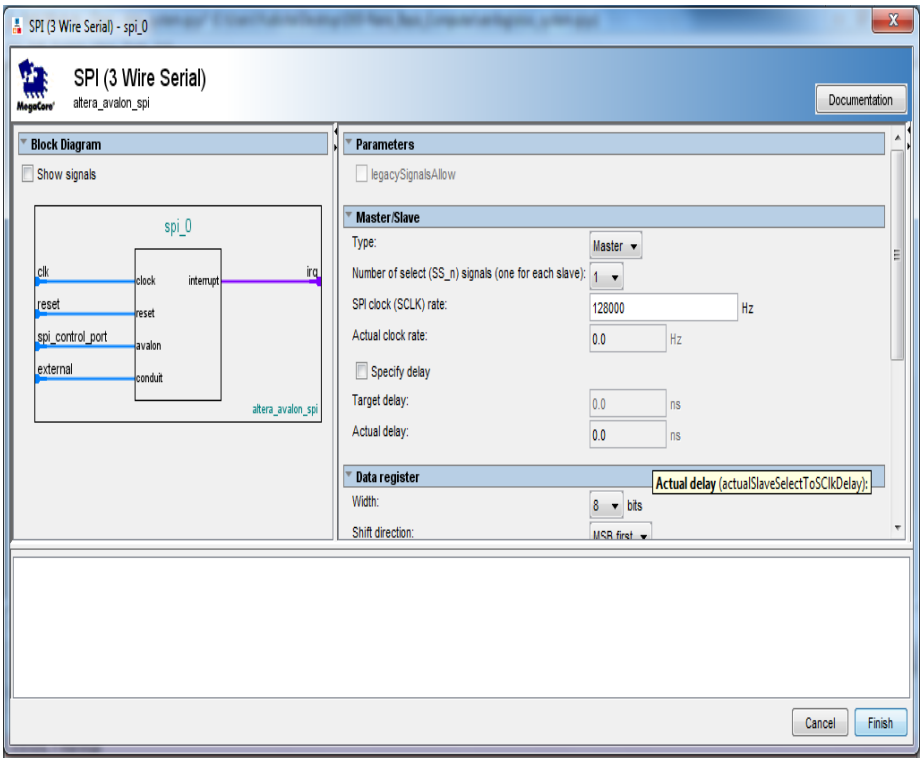


Figura 3.5: Ventana de SPI (3 WireSignal)

- Aquí configuramos nuestro módulo SPI, en nuestro caso solo tenemos que configurar nuestro módulo como esclavo y eso lo hacemos en la parte de **Type,** luego ponemos Finish**.**
- Repetimos el mismo procedimiento para el segundo módulo SPI.
- Nos van a aparecer errores y warnings.
- Para eliminar los errores y warnings debemos hacer las conexiones pertinentes, habilitar interrupciones y exportar si es necesario, todos los pasos que tenemos que seguir para eliminar errores y warnings lo podemos ver en la ventana de **Messages** de QSYS.

- Luego, tenemos que ir al menú System/**Assign Base Adresses**, ésta opción asigna rangos de direcciones para cada uno de los módulos de nuestra máquina, y evita que haya cruce de direcciones, eliminando los errores por cruces de direcciones.
- Después de haber eliminado todos los errores y warnings, eliminamos todos los módulos que no vamos a usar.
- Luego, en la ventana Generation ponemos Generate.

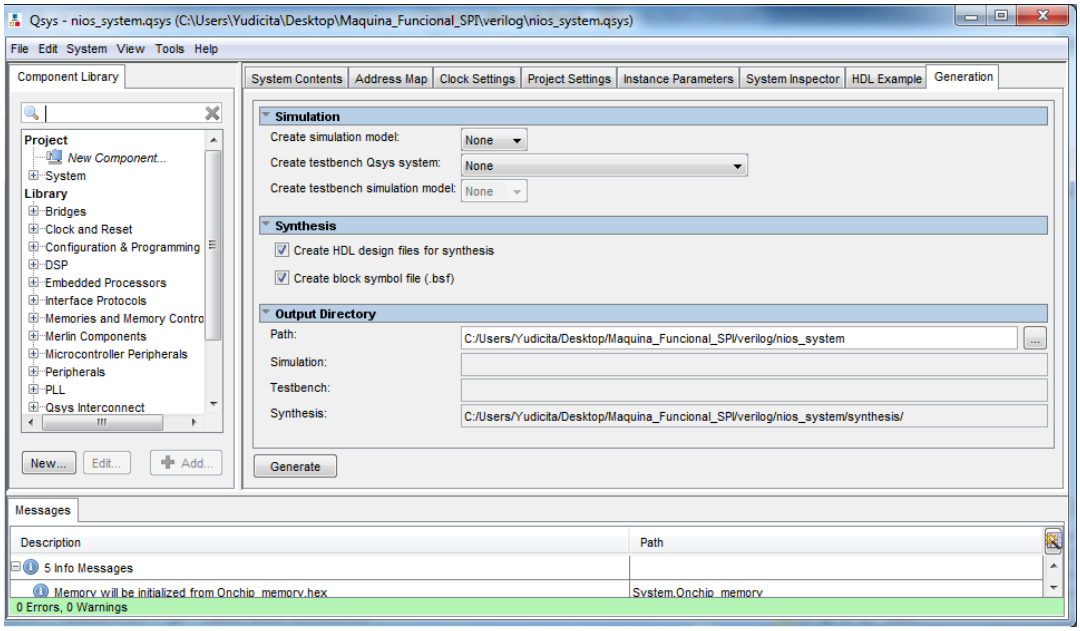


Figura 3.6: Pestaña de Generation de la ventana de QSYS

- Después, en la ventana de Quartus II 12.1, en la parte de Files hacemos click en el archivo .v de nuestra máquina, aquí tenemos que declarar nuestras variables correspondientes a los nuevos módulos creados y hay que eliminar las variables que corresponden a los módulos que eliminamos.

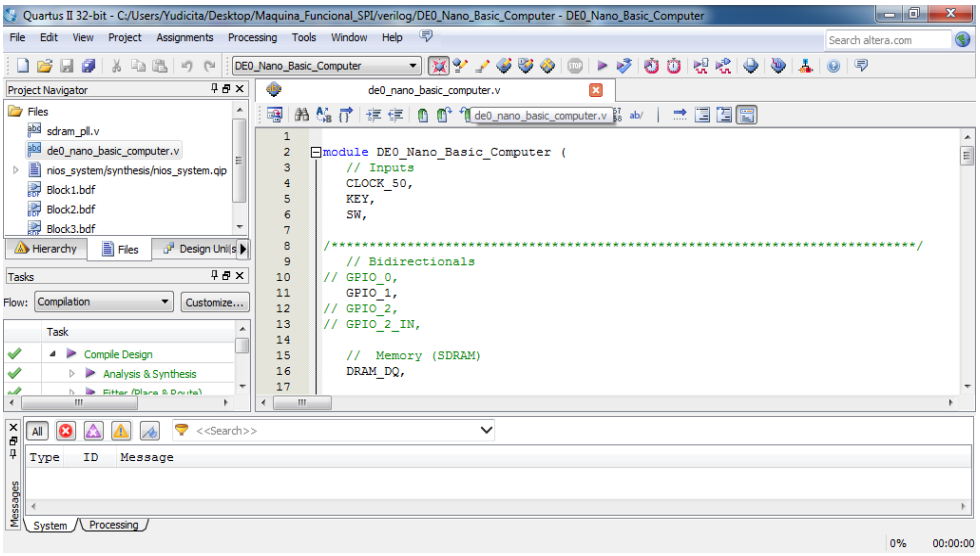


Figura 3.7: Ventana que muestra el archivo .v de nuestra máquina

- Luego mandamos a compilar.
- Nos saldrán errores debido a que no hemos asignado pines en el pin planner.
- Asignamos los pines en el pin planner y luego mandamos a compilar otra vez.
- Después de compilar, la máquina está lista para programarla en NIOS II IDE, nuestro hardware ya está completo.

3.2. DISEÑO E IMPLEMENTACIÓN DE SOFTWARE

En ésta sección describimos el diseño y la implementación del software de nuestro proyecto: especificaciones, diagrama de flujo e implementación en NIOS II IDE.

3.2.1. ESPECIFICACIONES

Con la herramienta NIOS II IDE, diseñamos un software que nos permite analizar tramas de comunicación entre dispositivos que se comunican con protocolo SPI.

3.2.2. DIAGRAMA DE FLUJO

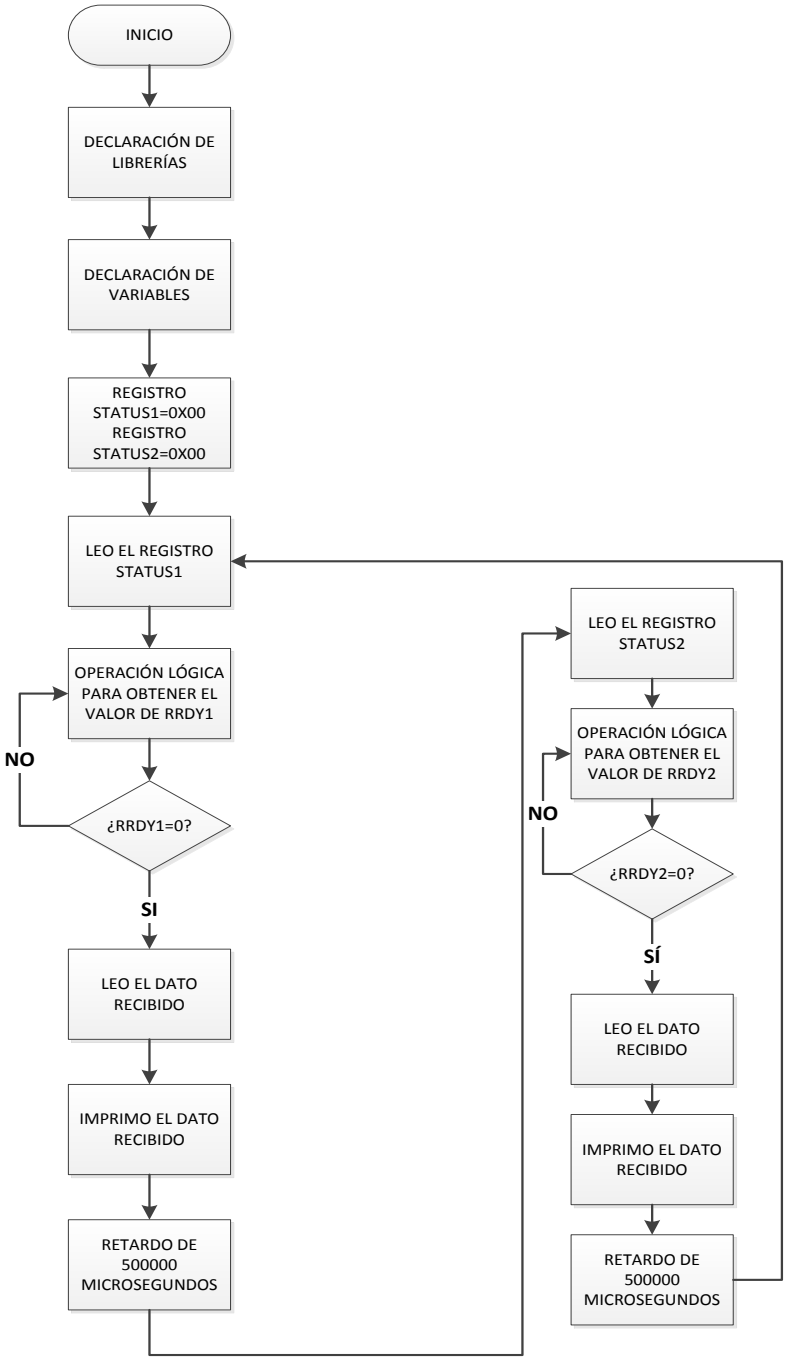


Figura 3.8: Diagrama de Flujo

3.2.2. IMPLEMENTACIÓN DE SOFTWARE EN NIOS II IDE

Para implementar nuestro software en NIOS II IDE, nos basamos en las funciones de las librerías que se crean cuando generamos nuestra máquina en QSYS y compilamos en Quartus II, las librerías que usamos son:

- system.h
- io.h
- altera_avalon_spi.h
- altera_avalon_spi_regs.h

Y las librerías estándar de c:

- stdio.h
- unistd.h
- ctype.h
- string.h

Con éstas librerías creamos un software específico que nos permite recolectar y analizar los datos que se envía y/o reciben en dispositivos que utilizan protocolo SPI.

Los pasos para crear nuestro software son:

- Abrimos NIOS II IDE.
- En la barra de menú ponemos File/New/NIOS II Aplication and BSP from Template.

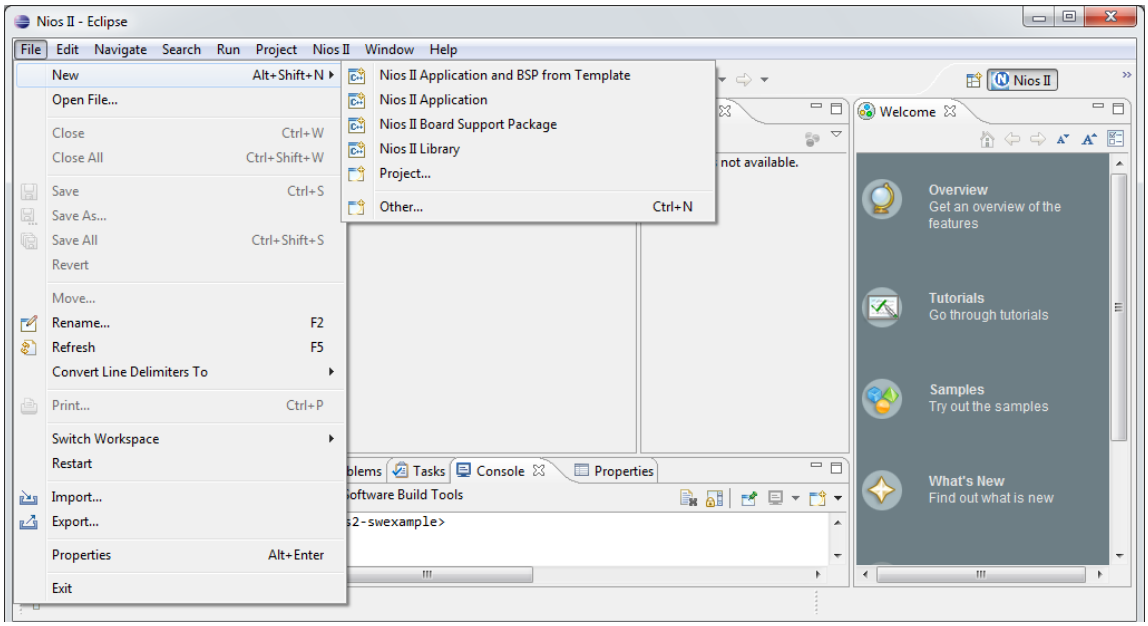


Figura 3.9: Paso 2, creación de software.

- En la parte de “SOPC information file name”, como muestra la figura 3.10, ponemos nuestro archivo con extensión .sopcinfo que contiene la información de la máquina que hemos creado.

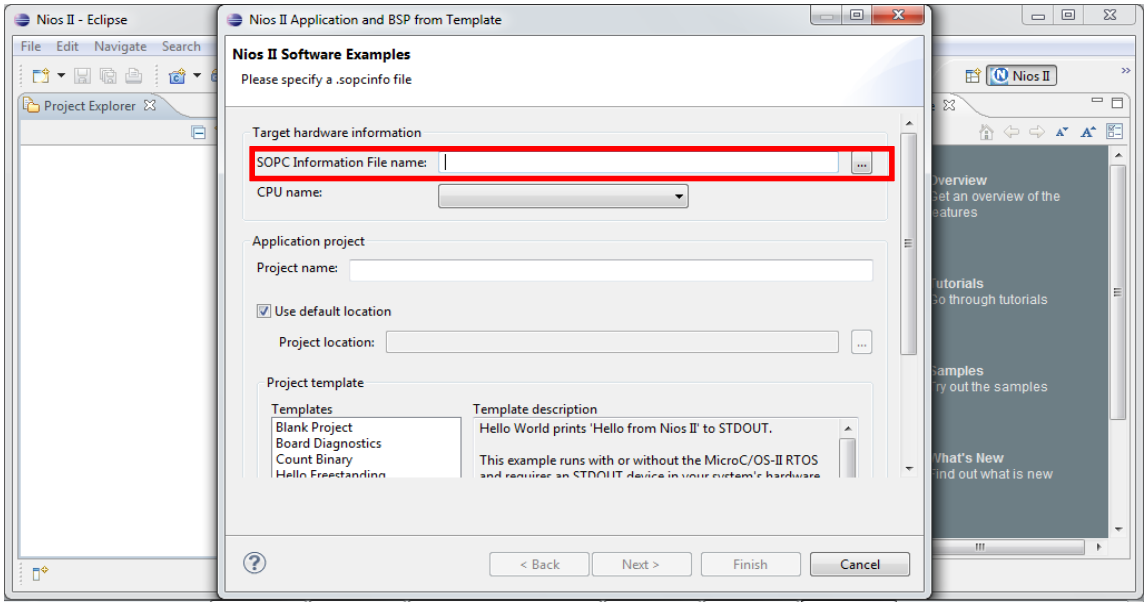


Figura 3.10: Paso 3, creación de software.

- Escribimos el nombre del proyecto.
- En la parte de “Project template”, como muestra la figura 3.11, escogemos en “Templates” la plantilla que dice “HelloWorld”.

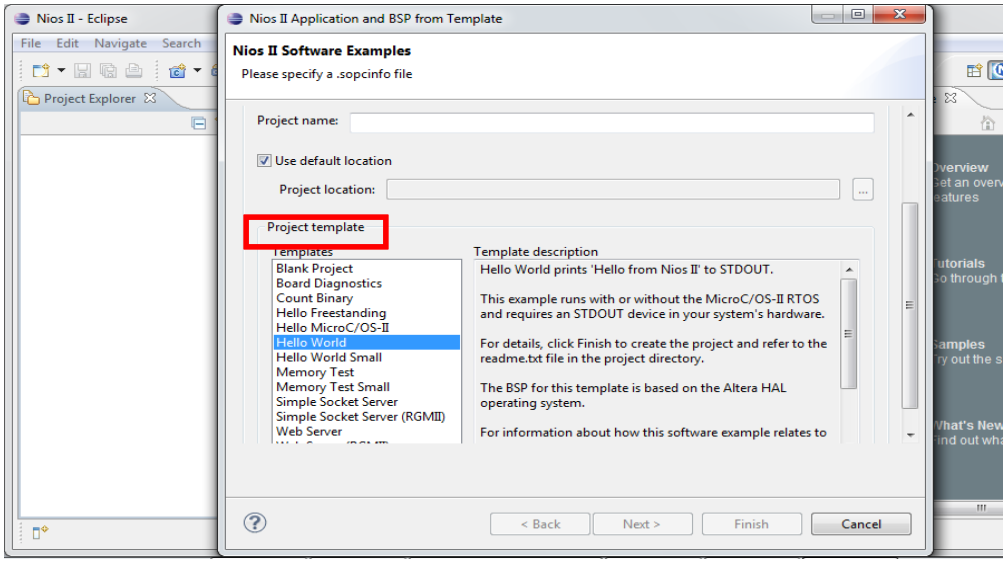


Figura 3.11: Paso 5, creación de software.

- Click en Finish.
- En la carpeta de trabajo abrimos el archivo “hello_world.c”, como se muestra en la figura 3.12.

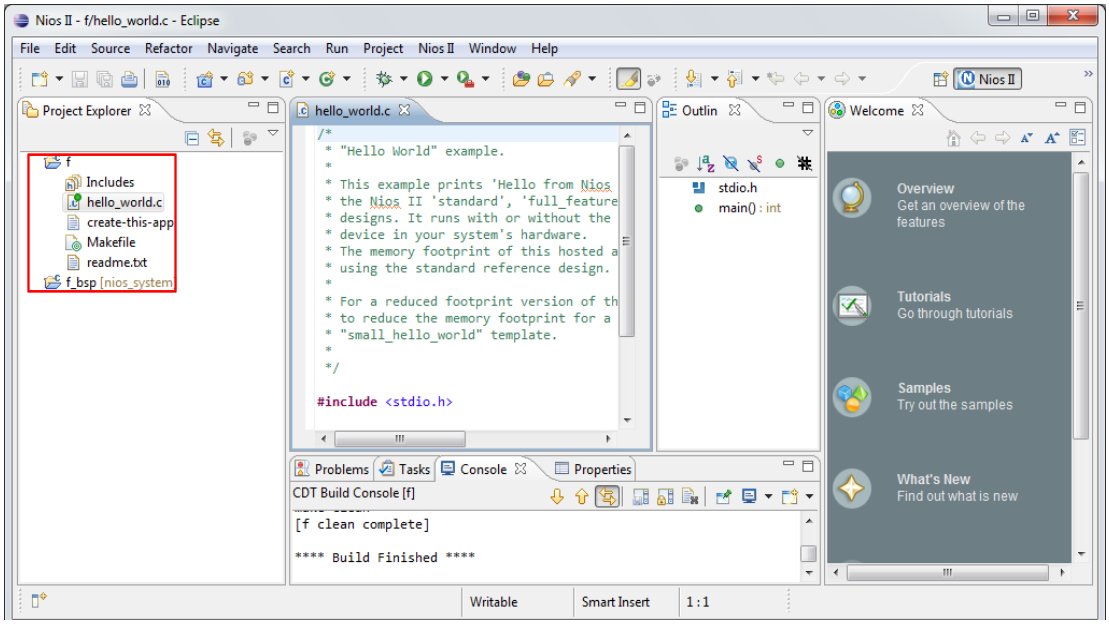


Figura 3.12: Paso 7, creación de software.

- En éste archivo vamos a hacer la programación de nuestro analizador.
- Para programar los dos módulos SPI de nuestro analizador seguimos la siguiente estrategia:
- Enceramos el registro STATUS.
- Leemos el registro STATUS.
- Chequeo si RRDY es 0, lo que significa que el registro RXDATA está vacío y está listo para leer datos.
- Leo el dato recibido.

- Imprimo en consola el dato recibido.

3.2.3.1. DESCRIPCIÓN DE LAS FUNCIONES UTILIZADAS

- **IOWR_ALTERA_AVALON_SPI_STATUS(base, data):** Ésta función nos permite escribir en el registro STATUS de un módulo SPI. Recibe como argumento la dirección **base** del módulo SPI al que queremos modificar su registro STATUS y **data** es el valor que queremos que tenga el registro STATUS.
- **IORD_ALTERA_AVALON_SPI_STATUS(base):** Ésta función nos permite leer el registro STATUS de un módulo SPI. Recibe como argumento la dirección **base** del módulo SPI del que queremos leer su registro STATUS.
- **ALTERA_AVALON_SPI_STATUS_RRDY_MSK:** Éste alias nos permite referirnos a una constante que tiene el valor de 0x80 en hexadecimal, lo que es 10000000 en binario.
- **IORD_ALTERA_AVALON_SPI_RXDATA(base):** Ésta función nos permite leer los datos que están llegando a un módulo SPI, si se trata de un esclavo, lee los datos desde la línea MOSI, si es un maestro, lee los datos desde la línea MISO. Recibe como argumento la dirección **base** del módulo SPI del que queremos leer datos.
- **IOWR_ALTERA_AVALON_SPI_SLAVE_SEL(base, data):** Ésta función nos permite escribir en la línea de selector de esclavo. Recibe como argumento la dirección **base** del módulo SPI y **data** es el dato que escribimos en el selector de esclavo del módulo SPI. Se escribe 0 para activar el módulo, 1 para desactivarlo.

- **ALTERA_AVALON_SPI_STATUS_TRDY_MSK:** Éste alias nos permite referirnos a una constante que tiene el valor de 0x40 en hexadecimal, lo que es 1000000 en binario.
- **IOWR_ALTERA_AVALON_SPI_TXDATA(base, data):** Ésta función nos permite escribir datos para transmitirlos, si se trata de un esclavo, escribe en la línea MISO, si es un maestro, escribe en la línea MOSI. Recibe como argumento la dirección **base** del módulo SPI y **data** es el dato que vamos a transmitir.
- **ALTERA_AVALON_SPI_STATUS_TMT_MSK:** Éste alias nos permite referirnos a una constante que tiene el valor de 0x20 en hexadecimal, lo que es 100000 en binario.
- **ALTERA_AVALON_SPI_STATUS_E_MSK:** Ésta alias nos permite referirnos a una constante que tiene el valor de 0x100 en hexadecimal, lo que es 100000000 en binario.

CAPÍTULO IV

4. PRUEBAS

En el presente capítulo se muestran las pruebas realizadas que permiten mostrar el funcionamiento del proyecto. Se observan los datos obtenidos por el analizador en diferentes dispositivos que utilizan protocolo SPI: una DE0 NANO, DE2-115 y PIC 16f887. Hicimos comparaciones de los resultados obtenidos cuando en el analizador no se conecta la tierra con los resultados obtenidos cuando se conecta la tierra. También se hicieron comparaciones entre los resultados obtenidos cuando tenemos una comunicación SPI a corta distancia con los resultados obtenidos con una comunicación SPI a larga distancia. Para entender esto, hemos escogido cuatro escenarios que nos permitirán comprender mejor el funcionamiento del proyecto los cuales se muestran a continuación:

Escenario A: Comunicación SPI en DE0 nano.

Escenario B: Comunicación SPI en DE2-115.

Escenario C: Comunicación SPI entre DE2-115 y DE0 nano.

Escenario D: Comunicación SPI entre PIC 16f87 y DE0 nano.

También se presentan comparaciones de los tiempos de compilación y recursos de la FPGA usados entre una máquina NIOS II de propósito general que contiene un analizador de protocolo SPI y una máquina NIOS II de propósito específico, que sólo contiene un analizador de protocolo SPI.

4.1. ESCENARIO A: ANÁLISIS DE COMUNICACIÓN SPI EN DE0 NANO

En éste escenario analizamos una comunicación SPI en una DE0 nano que tiene embebido un maestro y un esclavo que se comunican con protocolo SPI. Lo que se espera es que nuestro analizador lea lo que se están enviando entre maestro y esclavo. Hicimos una comparación entre los resultados que se obtienen cuando se conecta el analizador a la tierra de la configuración con los resultados que se obtienen cuando no se hace esto, se espera que las lecturas sean más precisas cuando la tierra de la configuración a analizar está conectada a la tierra de nuestro analizador.

En las siguientes figuras se muestra las conexiones y como luce físicamente el escenario:

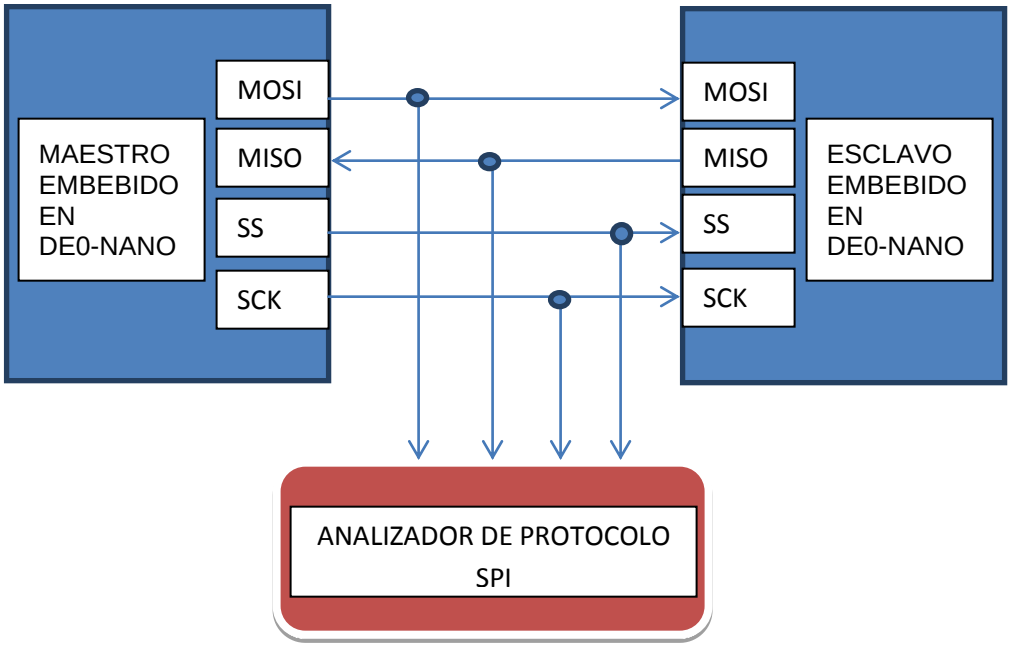


Figura 4.1: Conexiones del escenario A

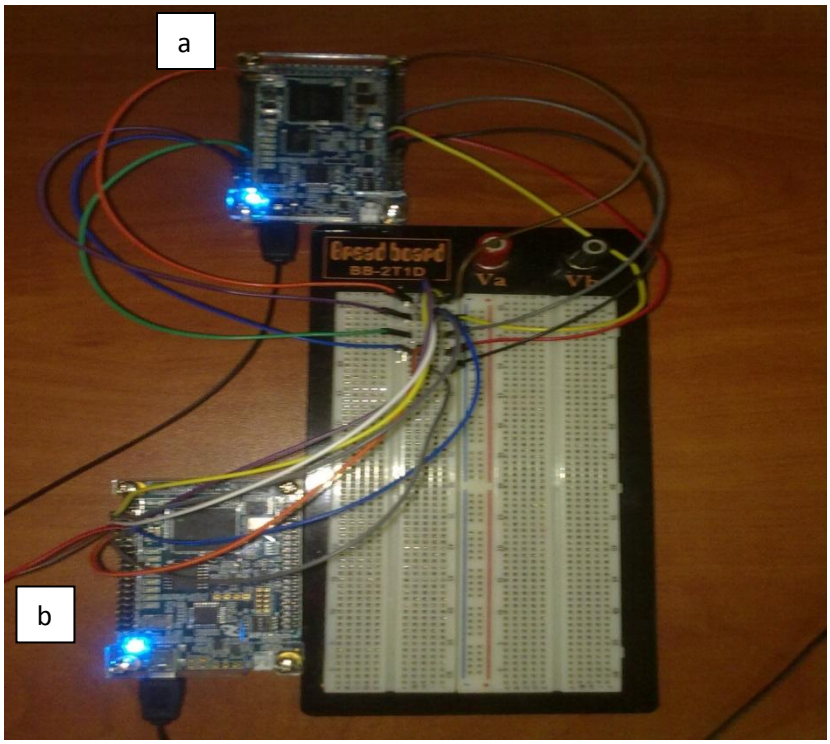


Figura 4.2: Escenario A; a) Maestro y esclavo SPI embebidos en DE0 nano; b) Analizador de Protocolo SPI embebido en DE0 nano

Una DE0 nano es la que tiene embebida la comunicación SPI entre un maestro y un esclavo, y la otra DE0 nano es la que tiene embebido nuestro analizador.

La siguiente tabla muestra los resultados obtenidos cuando el analizador no está conectado a tierra:

Dato enviado por el Maestro	Dato enviado por el Esclavo	Número de lecturas	Porcentaje de Error
0x78	0x50	100	20%
0x45	0x30	100	24%
0x75	0x58	100	27%
Promedio de error			23.66%

Tabla 4.1: Resultados obtenidos con el analizador no conectado a tierra, escenario A

La siguiente tabla muestra los resultados obtenidos cuando el analizador está conectado a tierra:

Dato enviado por el Maestro	Dato enviado por el Esclavo	Número de lecturas	Porcentaje de Error
0x78	0x50	100	2%
0x45	0x30	100	0.4%
0x75	0x58	100	1%
Promedio de error			1.13%

Tabla 4.2: Resultados obtenidos con el analizador conectado a tierra, escenario A

Resumen de los resultados:

	Porcentaje promedio de error
Analizador no conectado a la tierra de la configuración analizada	23.66%
Analizador conectado a la tierra de la configuración analizada	1.13%

Tabla 4.3: Resumen de resultados del escenario A

Se observa que se obtienen lecturas más precisas cuando conectamos la tierra de la configuración a analizar a la tierra del analizador.

4.2. ESCENARIO B: ANÁLISIS DE COMUNICACIÓN SPI EN DE2-115

En éste escenario analizamos una comunicación SPI en una DE2-115 que tiene embebido un maestro y un esclavo que se comunican con protocolo SPI. Lo que se espera es que nuestro analizador lea lo que se están enviando entre maestro y esclavo. Hicimos una comparación entre los resultados que se obtienen cuando se conecta el analizador a la tierra de la configuración con los resultados que se obtienen cuando no se hace esto, se espera que las lecturas sean más precisas cuando la tierra de la configuración a analizar está conectada a nuestro analizador.

En las siguientes figuras se muestra las conexiones y como luce físicamente el escenario:

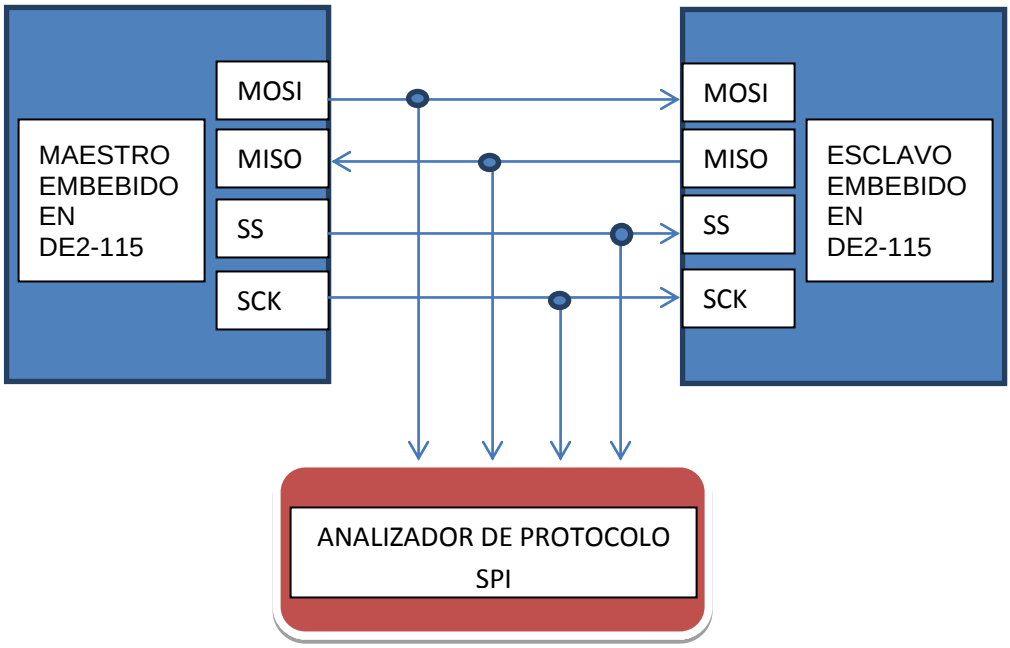


Figura 4.3: Conexiones del escenario B

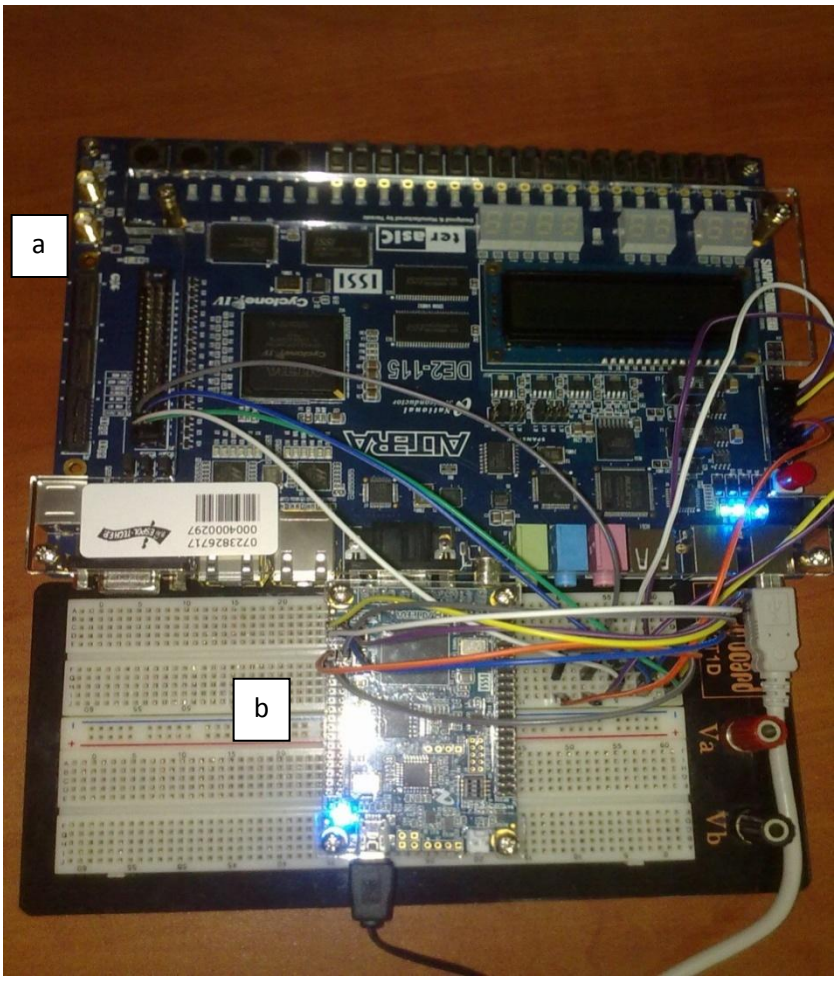


Figura 4.4: Escenario B, a) Maestro y esclavo SPI embebidos en DE2-115; b) Analizador de Protocolo SPI embebido en DE0 nano

La DE2-115 es la que tiene embebida la comunicación SPI entre un maestro y un esclavo, y la DE0 nano es la que tiene embebido nuestro analizador.

La siguiente tabla muestra los resultados obtenidos cuando el analizador no está conectado a tierra:

Dato enviado por el Maestro	Dato enviado por el Esclavo	Número de lecturas	Porcentaje de Error
0x20	0x78	100	25%
0x81	0x32	100	22%
0x43	0x14	100	29%
Promedio de error			25.33%

Tabla 4.4: Resultados obtenidos con el analizador no conectado a tierra, escenario B

La siguiente tabla muestra los resultados obtenidos cuando el analizador está conectado a tierra:

Dato enviado por el Maestro	Dato enviado por el Esclavo	Número de lecturas	Porcentaje de Error
0x20	0x78	100	5%
0x81	0x32	100	3%
0x43	0x14	100	6%
Promedio de error			4.66%

Tabla 4.5: Resultados obtenidos con el analizador conectado a tierra, escenario B

Resumen de los resultados:

	Porcentaje promedio de error
Analizador no conectado a la tierra de la configuración analizada	25.33%
Analizador conectado a la tierra de la configuración analizada	4.66%

Tabla 4.6: Resumen de resultados del escenario B

Se observa que se obtienen lecturas más precisas cuando conectamos la tierra de la configuración a analizar a la tierra del analizador.

4.3. ANÁLISIS DE COMUNICACIÓN SPI ENTRE DE2-115 Y DE0 NANO

En éste escenario analizamos una comunicación SPI entre una DE2-115 configurada como maestro, con una DE0 nano configurada como esclavo. Lo que se espera es que nuestro analizador lea lo que se están enviando entre maestro y esclavo. Hicimos una comparación entre los resultados que se obtienen cuando tenemos una comunicación SPI a corta distancia con una comunicación SPI a larga distancia, se espera que las lecturas sean más precisas cuando tenemos una comunicación a corta distancia, dado que el protocolo SPI es eficiente a cortas distancias.

En las siguientes figuras se muestra las conexiones y como luce físicamente el escenario:

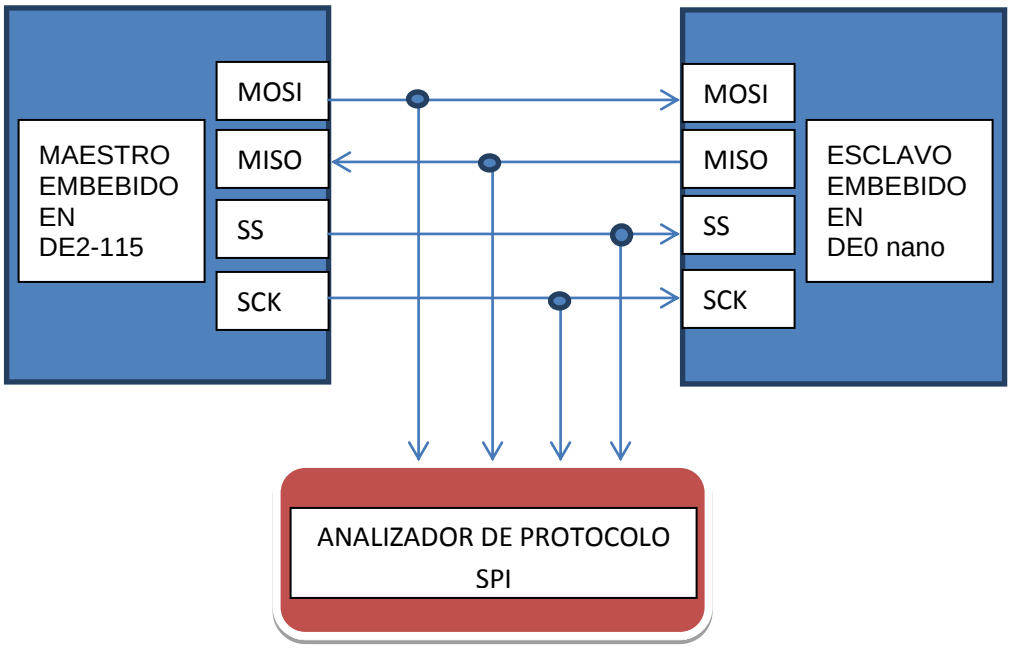


Figura 4.5: Conexiones del escenario C

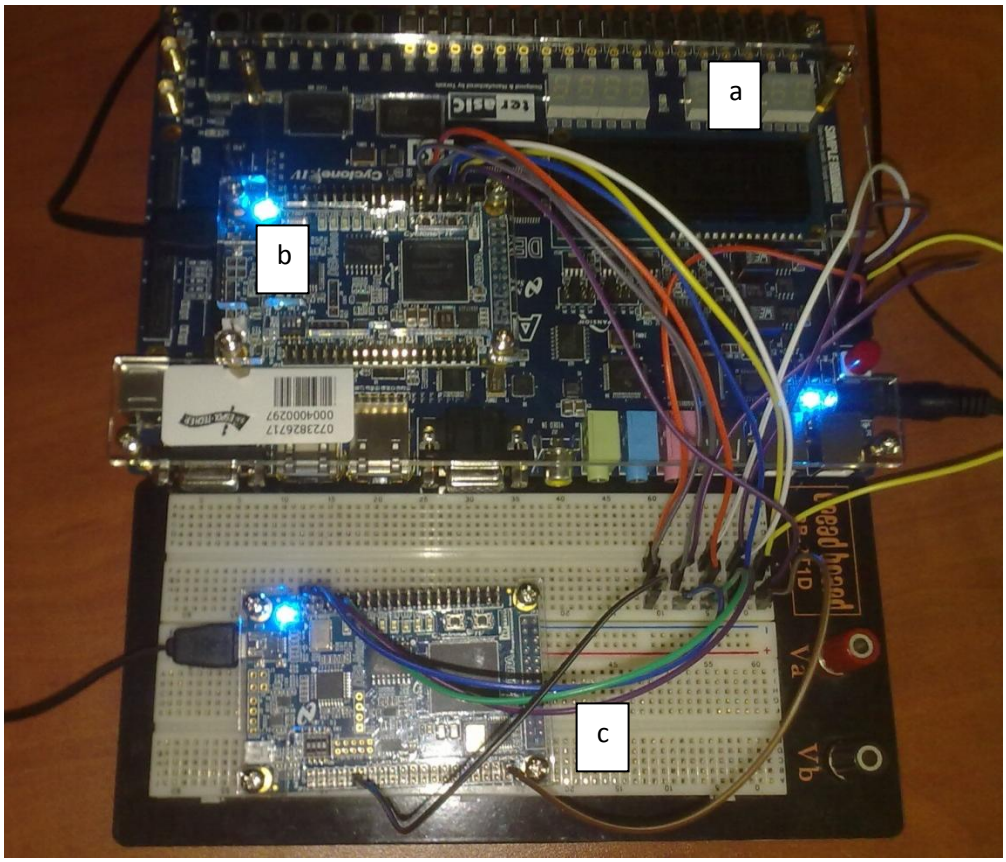


Figura 4.6: Escenario C, a) Maestro SPI embebido en DE2-115; b) Analizador de Protocolo SPI embebido en DE0 nano, c) Esclavo SPI embebido en DE0 nano

La DE2-115 es la que tiene embebida un maestro SPI, una DE0 nano tiene embebido un esclavo SPI, la otra DE0 nano es la que tiene embebido nuestro analizador.

La siguiente tabla muestra los resultados obtenidos cuando el analizador está conectado a 40 centímetros:

Dato enviado por el Maestro	Dato enviado por el Esclavo	Número de lecturas	Porcentaje de Error
0x14	0x58	100	9%
0x37	0x97	100	8%
0x12	0x45	100	6%
Promedio de error			7.66%

Tabla 4.7: Resultados obtenidos con el analizador conectado a 40 cm, escenario C

La siguiente tabla muestra los resultados obtenidos cuando el analizador está conectado a 60 centímetros:

Dato enviado por el Maestro	Dato enviado por el Esclavo	Número de lecturas	Porcentaje de Error
0x73	0x26	100	17%
0x83	0x19	100	18%
0x61	0x75	100	20%
Promedio de error			18.33%

Tabla 4.8: Resultados obtenidos con el analizador conectado a 60 cm, escenario C

La siguiente tabla muestra los resultados obtenidos cuando el analizador está conectado a 80 centímetros:

Dato enviado por el Maestro	Dato enviado por el Esclavo	Número de lecturas	Porcentaje de Error
0x42	0x98	100	28%
0x79	0x15	100	22%
0x63	0x35	100	25%
Promedio de error			25%

Tabla 4.9: Resultados obtenidos con el analizador conectado a 80 cm, escenario C

Resumen de los resultados:

	Porcentaje promedio de error
Analizador conectado a 40 cm	7.66%
Analizador conectado a 60 cm	18.33%
Analizador conectado a 80 cm	25%

Tabla 4.10: Resumen de resultados del escenario C

Se observa que se obtienen lecturas más precisas cuando conectamos el analizador a cortas distancias.

4.4. ANÁLISIS DE COMUNICACIÓN SPI ENTRE PIC 16F887 Y DE0 NANO

En éste escenario analizamos una comunicación SPI entre un PIC 16F887 configurado como master y una DE0 nano que tiene embebido esclavo SPI. Lo que

se espera es que nuestro analizador lea lo que se están enviando entre maestro y esclavo. Hicimos una comparación entre los resultados que se obtienen cuando tenemos una comunicación SPI a corta distancia con una comunicación SPI a larga distancia, se espera que las lecturas sean más precisas cuando tenemos una comunicación a corta distancia, dado que el protocolo SPI es más eficiente a cortas distancias.

En las siguientes figuras se muestra las conexiones y como luce físicamente el escenario:

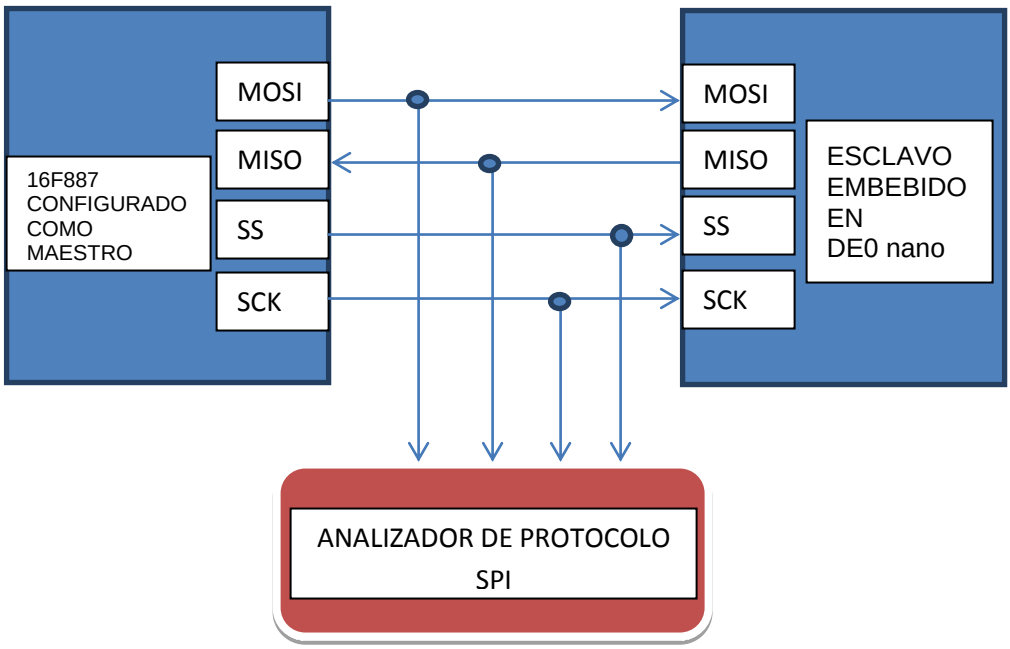


Figura 4.7: Conexiones del escenario D

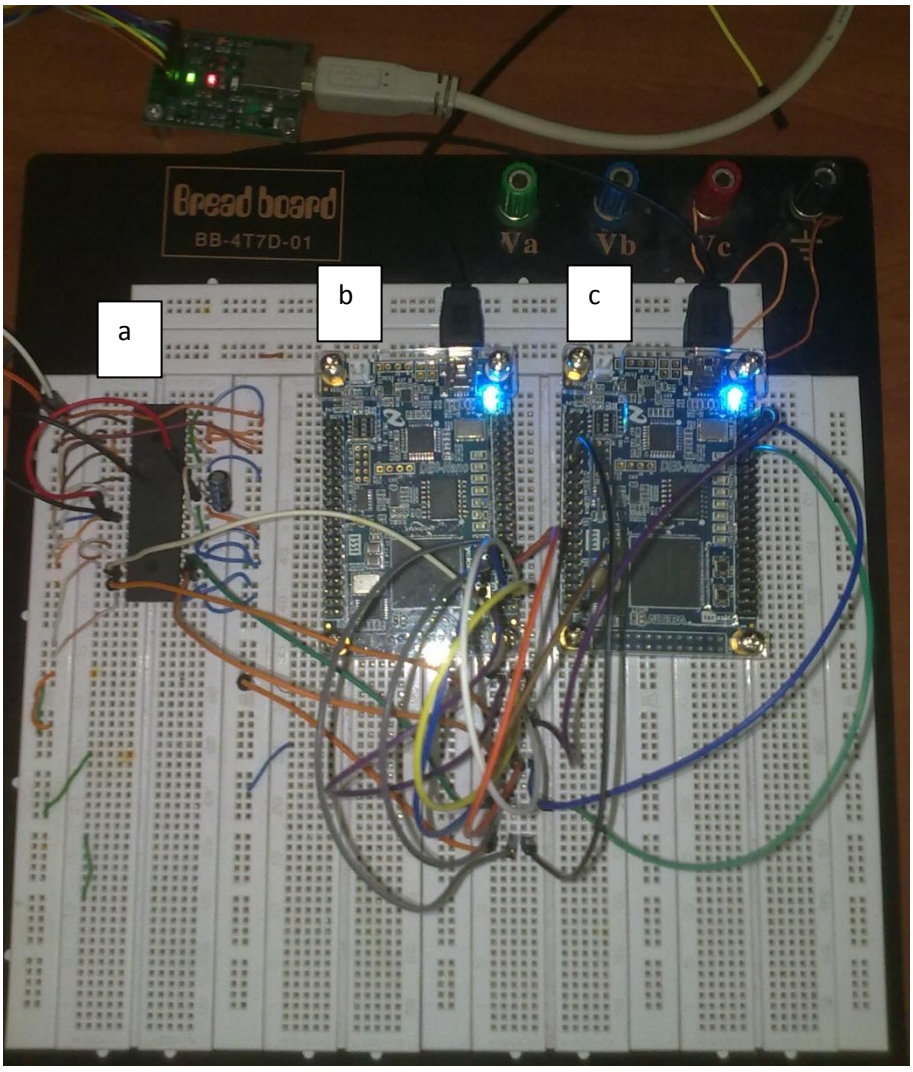


Figura 4.8: Escenario D, a) 16f887 configurado como Maestro SPI; b) Analizador de Protocolo SPI embebido en DE0 nano, c) Esclavo SPI embebido en DE0 nano

El PIC 16f887 es el master SPI, una DE0 nano tiene embebido un esclavo SPI, la otra DE0 nano tiene embebido nuestro analizador.

La siguiente tabla muestra los resultados obtenidos cuando el analizador está conectado a 30 centímetros:

Dato enviado por el Maestro	Número de lecturas	Porcentaje de Error
0x41	100	10%
0x41	100	7%
0x41	100	5%
Promedio de error		7.33%

Tabla 4.11: Resultados obtenidos con el analizador conectado a 30 cm, escenario D

La siguiente tabla muestra los resultados obtenidos cuando el analizador está conectado a 50 centímetros

Dato enviado por el Maestro	Número de lecturas	Porcentaje de Error
0x41	100	27%
0x41	100	25%
0x41	100	28%
Promedio de error		26.66%

Tabla 4.12: Resultados obtenidos con el analizador conectado a 50 cm, escenario D

La siguiente tabla muestra los resultados obtenidos cuando el analizador está conectado a 70 centímetros:

Dato enviado por el Maestro	Número de lecturas	Porcentaje de Error
0x41	100	30%
0x41	100	32%
0x41	100	38%
Promedio de error		33.33%

Tabla 4.13: Resultados obtenidos con el analizador conectado a 80 cm, escenario D

Resumen de los resultados:

	Porcentaje de error
Analizador conectado a 30 cm	7.33%
Analizador conectado a 50 cm	26.66%
Analizador conectado a 70 cm	33.33%

Tabla 4.14: Resumen de resultados del escenario D

Se observa que se obtienen lecturas más precisas cuando conectamos el analizador a cortas distancias.

4.5. RESULTADOS

Los resultados del escenario A y B nos hacen concluir que nuestro analizador debe trabajar necesariamente con la tierra de la configuración para obtener datos confiables.

Los resultados del escenario C y D nos hacen concluir que nuestro analizador debe trabajar a distancias menores que 40 cm para tener una lectura confiable de datos.

4.6. COMPARACIÓN DE TIEMPOS DE COMPILACIÓN Y RECURSOS USADOS DE LA FPGA ENTRE UNA MÁQUINA NIOS II DE PROPÓSITO GENERAL YUNA MÁQUINA NIOS II DE PROPÓSITO ESPECÍFICO.

En ésta sección vamos a hacer una comparación de los tiempos de compilación y recursos de la FPGA utilizados entre una máquina NIOS II de propósito general, en éste caso la DE0 nano Basic Computer, que tiene añadido un analizador de protocolo SPI, con una máquina NIOS II de propósito específico, que tiene embebido sólo el analizador de protocolo SPI.

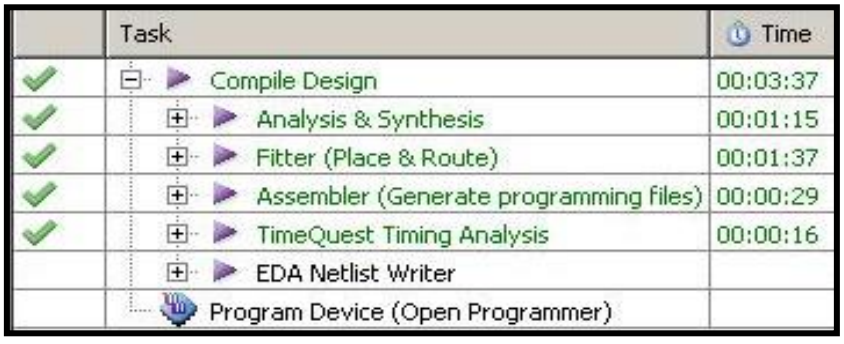


Figura 4.9: Tiempo de compilación de máquina NIOS II de propósito específico (Analizador de Protocolo SPI)

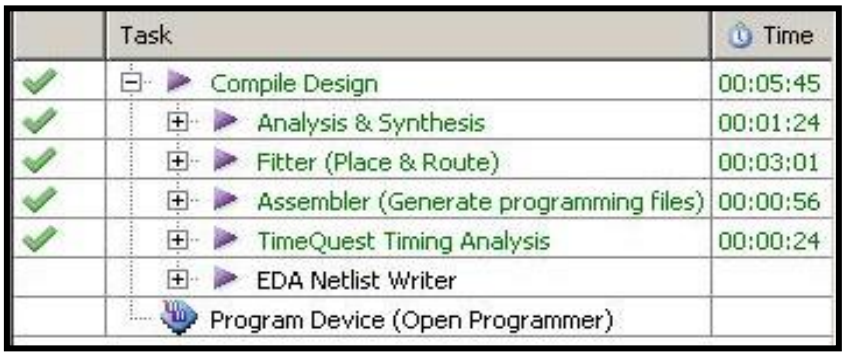


Figura 4.10: Tiempo de compilación de máquina NIOS II de propósito general con analizador SPI

Se puede ver que el tiempo de compilación entre las dos máquinas es relativamente diferente, esto se debe a que una de las máquinas (Máquina NIOS II de Propósito General) contiene en su interior todos los componentes básicos con los que fue creado inicialmente y los módulos con los que se trabajó en el proyecto, a diferencia de la máquina creada para el proyecto (Máquina NIOS II de Propósito Específico) donde todos aquellos módulos que eran innecesarios fueron eliminados y así mismo los módulos necesarios para el proyecto fueron agregados.

En la tabla 4.15 se puede ver de una manera más clara como el tiempo se reduce durante cada una de las etapas de compilación:

	Máquina NIOS II de Propósitos Generales	Máquina NIOS II SPI Definitiva
Analysis & Synthesis	00:01:24	00:01:15
Fitter (Place &Route)	00:03:01	00:01:37
Assembler (Generate Programming Files)	00:00:56	00:00:29
TimeQuest (Timing Analysis)	00:00:24	00:00:16
Compile Design	00:05:45	00:03:37

Tabla 4.15: Tiempo de Compilación entre las dos máquinas creadas

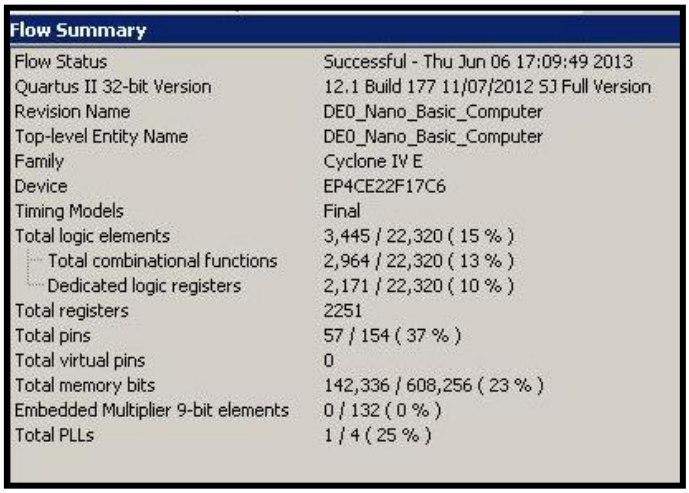
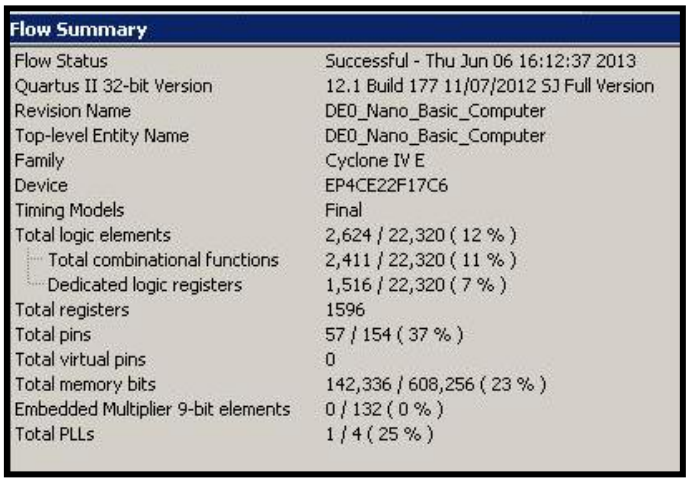


Figura 4.11: Uso de recursos de la FPGA por parte de la Máquina NIOS II de Propósito General



Flow Summary	
Flow Status	Successful - Thu Jun 06 16:12:37 2013
Quartus II 32-bit Version	12.1 Build 177 11/07/2012 SJ Full Version
Revision Name	DE0_Nano_Basic_Computer
Top-level Entity Name	DE0_Nano_Basic_Computer
Family	Cyclone IV E
Device	EP4CE22F17C6
Timing Models	Final
Total logic elements	2,624 / 22,320 (12 %)
Total combinational functions	2,411 / 22,320 (11 %)
Dedicated logic registers	1,516 / 22,320 (7 %)
Total registers	1596
Total pins	57 / 154 (37 %)
Total virtual pins	0
Total memory bits	142,336 / 608,256 (23 %)
Embedded Multiplier 9-bit elements	0 / 132 (0 %)
Total PLLs	1 / 4 (25 %)

Figura 4.12: Uso de recursos de la FPGA por parte de la Máquina NIOS II de Propósito Específico

Así como el tiempo de compilación se ve reducido al momento de crear nuestra máquina definitiva, los recursos que son utilizados por parte de la FPGA son notablemente visibles, ya que cuando se eliminan módulos que no son necesarios se liberan por así decirlo registros lógicos, funciones combinatoriales, pines, memoria, etc., lo que permite implementar dispositivos más pequeños, versátiles y sobre todo destinados únicamente a realizar el trabajo para el que fue diseñado.

	Máquina NIOS II de Propósito General	Máquina NIOS II de Propósito Específico
Funciones combinatoriales	2964/22320	2411/22320
Registros Lógicos	2171/22320	1516/22320
Elementos Lógicos	3445/22320	2624/22320
Registros Totales	2251	1596
Pines	57/154	9/154
Bits de Memoria	142336/608256	142336/608256

Tabla 4.16: Comparación entre recursos utilizados por cada una de las máquinas

Si quisiéramos utilizar una FPGA que se ajuste a los recursos que usa el proyecto, dejando un pequeño margen de expansión, se podría usarunade las siguientes FPGA´s:

Serie	Número	Precio por unidad(USD)
Cyclone®	EP1C3T100C8	11.77
Cyclone®	EP1C3T100C6N	16.00
Cyclone®	EP1C3T100C8N	10.70
Cyclone®	EP1C3T100C7	14.08
Cyclone®	EP1C3T100I7	17.60
Cyclone®	EP1C3T100C7N	12.80
Cyclone®	EP1C3T100I7N	16.00
Cyclone®	EP1C3T100C6	17.16
Cyclone®	EP1C3T100A8N	21.40

Tabla 4.17: Algunas FPGA´s que se ajustan a los recursos que usa el proyecto

CONCLUSIONES

1. En base a los objetivos planteados, se pudo generar un sistema embebido, que cumple con las funcionalidades necesarias para analizar dispositivos que se comunican con protocolo SPI.

2. Con el uso de la herramienta QSYS de QUARTUS II 12.1, se pudo desarrollar un hardware que nos permite conectarnos con dispositivos que se comunican con protocolo SPI, demostrando la eficacia de ésta herramienta para construir hardwares específicos para un proyecto determinado.

3. Con el uso de la herramienta NIOS II IDE, se pudo desarrollar un software en lenguaje C, que nos permitió recolectar y analizar tramas de comunicación entre dispositivos que utilizan protocolo SPI, lo que demuestra la versatilidad de éste lenguaje de alto nivel para programar incluso sistemas embebidos.

4. Comprendimos el protocolo SPI, que es un protocolo estándar en la industria, que lo utilizan microcontroladores y microprocesadores. Para hacer nuestro analizador, nos

valimos de que cuando un dispositivo esclavo no está seleccionado, éste sólo “escucha” lo que se está enviando entre el maestro y el esclavo seleccionado, es decir se comporta como un espectador, y no “dice” algo, sólo “escucha”.

5. El protocolo SPI es más eficiente a cortas distancias, debido a que trabaja a altas frecuencias, entonces a largas distancias y con frecuencias altas, el factor ruido interviene en la comunicación.

RECOMENDACIONES

1. Se recomienda revisar los diagramas esquemáticos de las tarjetas DE0_Nano, DE2 en el caso de conectar dispositivos externos para evitar corto circuitos, problemas de conexión, daños en los módulos integrados, todo esto con el fin de que no se produzca algún daño permanente.
2. Muchos de los problemas que se presentaron en la implementación se dieron a causa de las salidas de los dispositivos que no se estaban utilizando, por eso se recomienda en el caso de las tarjetas DE0-Nano o DE2-115, asignar las salidas necesarias cuando se crean las máquinas, en el caso de los microcontroladores, programar como salidas los pines que no se utilizan y conectarlos a tierra, ésto para disminuir el error en las lecturas de nuestro analizador.
3. Se recomienda tener un conocimiento de las librerías que se utilizan al momento de la programación sobre todo aquellas que interactúan con los registros del Bus Avalon, ya que es este es el que permite conectar todos los elementos dentro de la FPGA y un

mal manejo de estos registros puede llevar a un mal funcionamiento del proyecto al momento de ver los resultados.

4. Se recomienda retirar de la máquina básica aquellos módulos que no van a ser utilizados, ya que el tamaño del circuito es un factor importante cuando se está desarrollando un prototipo, porque influye directamente en el precio al momento de la implementación.

5. Se recomienda revisar los voltajes de alimentación con los que trabajan los dispositivos, ya que la alimentación que recibe cada uno de ellos puede ser diferente, lo que si bien no puede provocar daños en los dispositivos (DE0-Nano, 16F887), pero puede provocar errores de transmisión mostrando valores erróneos durante las pruebas.

TRABAJOS FUTUROS

La línea de trabajo de este proyecto es poder analizar las tramas que se transmiten de un dispositivo hacia otro, así que la eliminación de errores ha sido uno de los puntos vitales en donde hemos centrado nuestra atención, pero aún así se pueden realizar mejoras a fín de que sea más versátil.

- La utilización de LCD's es uno de los puntos en los que se podría mejorar, ya que los datos obtenidos por parte del analizador son presentados en la pantalla de consola de NIOS II, lo que es un inconveniente si se desea un analizador portátil lo que representaría una ventaja frente a otros dispositivos que realizan el mismo trabajo si se llegara a implementar masivamente.
- La implementación de un sistema que permita mantener los datos para que puedan ser interpretados en caso de que esos datos sean de vital importancia o si se desea realizar un estudio acerca del protocolo SPI.

- Crear un sistema que permita obtener datos SPI de forma inalámbrica ya que el dispositivo implementado en este proyecto lo hace de forma alámbrica y existen ciertos estándares de comunicaciones inalámbricos como WirelessUSB que se basan en protocolo SPI, que muchas veces requieren un análisis de sus tramas en casos de un mal funcionamiento.
- Crear una interfaz gráfica en la que se pueda ver las señales que intervienen en las transmisión de las tramas, lo que nos da una visión más amplia del funcionamiento del protocolo SPI, ya que se puede observar el CLOCK, SS (Slave Select), MOSI (Master Output Slave Input), MISO (Master Input Slave), y el preciso momento en el que se produce la transmisión lo que no ocurre si se lo hace a través de un LCD.

ANEXO A: CÓDIGO FUENTE DEL ANALIZADOR

```
#include"altera_avalon_spi_regs.h"
#include<stdio.h>
#include<unistd.h>
#include"system.h"
#include"io.h"
#include"altera_avalon_spi.h"
#include"altera_avalon_timer_regs.h"
#include"altera_up_avalon_parallel_port_regs.h"
#include<ctype.h>
#include<string.h>
//ProgramaPrinicipal
intmain(void)
{
alt_u32 rddata1=0x00;
alt_u32 rddata2=0x00;
alt_u32 status;
chari=1, data=0, pdata=0;
//Encero el registro STATUS
IOWR_ALTERA_AVALON_SPI_STATUS(SPI_0_BASE,0x0000);
IOWR_ALTERA_AVALON_SPI_STATUS(SPI_1_BASE,0x0000);
//Lazoinfinito
```

```
while(1)
{

do
{
//Leemos el registro STATUS
status = IORD_ALTERA_AVALON_SPI_STATUS(SPI_0_BASE);
}
//Chequeosi RRDY1 es 0, loquesignificaque el registro RXDATA1 estávacío
while((status & ALTERA_AVALON_SPI_STATUS_RRDY_MSK)==0 &&
(status& ALTERA_AVALON_SPI_STATUS_RRDY_MSK) == 0);
//Leo el datorecibido
rddata1= IORD_ALTERA_AVALON_SPI_RXDATA(SPI_0_BASE);
//Imprimoencónsola el datorecibido
printf("rddata_1=%x\n",rddata1);
//Retardo
usleep(500000);
do
{
//Leemos el registro STATUS
status = IORD_ALTERA_AVALON_SPI_STATUS(SPI_1_BASE);
}
```

```
//Chequeosi RRDY2 es 0, loquesignificque el registro RXDATA2 estávacío
while((status & ALTERA_AVALON_SPI_STATUS_RRDY_MSK)==0 &&
(status& ALTERA_AVALON_SPI_STATUS_RRDY_MSK) == 0);
//Leo el datorecibido
rddata2= IORD_ALTERA_AVALON_SPI_RXDATA(SPI_1_BASE);
//Imprimoencónsola
printf("rddata_2=%x\n",rddata2);
//Retardo
usleep(500000);


}
return 0;
}
```

ANEXO B: CÓDIGO DE COMUNICACIÓN SPI PARA NIOS II CÓMO MAESTRO

```
//Declaracióndelibrerías
#include<stdio.h>
#include<unistd.h>
#include"system.h"
#include"io.h"
#include"altera_avalon_spi_regs.h"
#include"altera_avalon_spi.h"
#include"altera_up_avalon_parallel_port_regs.h"
#include"altera_up_avalon_parallel_port.h"
//Programa principal
intmain(){
while(1){
alt_u8 data=0x56;
//Desactivoalesclavo
IOWR_ALTERA_AVALON_SPI_SLAVE_SEL(SPI_0_BASE, 0x1);
//Chequeosi TRDY es 1 loquesignificaque el registro TX estálisto, sinoespero
while((IORD_ALTERA_AVALON_SPI_STATUS(SPI_0_BASE))                    &
(ALTERA_AVALON_SPI_STATUS_TRDY_MSK != 0x40));
//Si el registro TX estálisto, escribo el datoen el registro TX
IOWR_ALTERA_AVALON_SPI_TXDATA(SPI_0_BASE, data);
//Chequeosi TMT es 1, loquesignificaque el registrodedesplazamientoestávacío
```

```
while((IORD_ALTERA_AVALON_SPI_STATUS(SPI_0_BASE))                    &
(ALTERA_AVALON_SPI_STATUS_TMT_MSK != 0x20));
//Activoalesclavo
IOWR_ALTERA_AVALON_SPI_SLAVE_SEL(SPI_0_BASE, 0x0);
//Chequeosi hay algún error
if              ((IORD_ALTERA_AVALON_SPI_STATUS(SPI_0_BASE))          &
(ALTERA_AVALON_SPI_STATUS_E_MSK == 0x100))
printf("A data overrun error has occured");}
return 0;
}
```

ANEXO C: CÓDIGO DE COMUNICACIÓN SPI PARA NIOS II COMO ESCLAVO:

```
//Declaracióndelibrerías
#include<stdio.h>
#include<unistd.h>
#include"system.h"
#include"io.h"
#include"altera_avalon_spi_regs.h"
#include<stdio.h>
#include<unistd.h>
#include"system.h"
#include"io.h"
#include"altera_avalon_spi.h"

//ProgramaPrinicipal
intmain(void)
{
alt_u32rddata=0x0;
alt_u32 status;
//Encero el registro STATUS
IOWR_ALTERA_AVALON_SPI_STATUS(SPI_SLAVE_BASE,0x0000);

while(1)
```

```
{
do
{
//Leemos el registro STATUS
status = IORD_ALTERA_AVALON_SPI_STATUS(SPI_SLAVE_BASE);
}
//Chequeo si RRDY es 0, lo que signific que está vacío
while((status & ALTERA_AVALON_SPI_STATUS_RRDY_MSK)==0 &&
(status& ALTERA_AVALON_SPI_STATUS_RRDY_MSK) == 0);
//Leo el datorecibido
rddata= IORD_ALTERA_AVALON_SPI_RXDATA(SPI_SLAVE_BASE);
//Imprimoencónsola el datorecibido
printf("rddata=%x\n",rddata);}
return 0;
}
```

ANEXO D: CÓDIGO DE COMUNICACIÓN SPI EN PIC 16F887 COMO MASTER

```
////////////////////////////////////////////////////////////////////////
//                    Serial Peripheral Interface                     //
// ////////////////////////////////////////////////////////////////////


//sbitChip_Select at RD0_bit;             // Pin RD0 esun pin deseleccionar el chip
sbitChip_Select_Direction at TRISD.b0;
sbitChip_Select at PORTD.b0;


// periféricoSelección_de_chip
//sbitChip_Select_Direction at TRISD0_bit; // Bit TRISC0 define el pin RC0 comoentrada
o salida
unsignedintvalue=0X41;                  // Dato a serenviadoesdetipounsignedint
short take, buffer;
voidmain() {
OSCCON =0X20;
ANSEL = ANSELH = 0;          // Todoslos pines de E/S son digitales
TRISB=1; // Configurarlos pines DEL PUERTO B comoentradas,
TRISa=0; // Configurarlos pines DEL PUERTO A Y D como salidas,
TRISd=0;
TRISc0_bit = TRISc1_bit =TRISc2_bit = TRISc6_bit=  TRISc7_bit = 0;
```

```
while(1){
Chip_Select = 0;            // Seleccionar el chip periférico
Chip_Select_Direction = 0;   // Configurar el pin CS# comosalida
  SPI1_Init();               // Inicializar el módulo SPI
  SPI1_Write(value);         // Envíar el valor al chip periférico
Delay_ms(250);
Chip_Select = 1;
 }

}
```

ANEXO E: NÚCLEO SPI DE NIOS II

REGISTRO STATUS

El registro estatus consiste de bits que indican condiciones particulares dentro de la comunicación SPI. El registro STATUS puede ser leído en cualquier momento por software, haciendo esto, no cambia el valor de los bits de éste registro.

Número de Bit	Nombre del Bit	Descripción
3	roe	Error de desbordamiento del receptor
4	toe	Error de desbordamiento del transmisor
5	tmt	Registro de desplazamiento del transmisor vacío
6	trdy	Transmisor listo
7	rrdy	Receptor listo
8	E	Error

Tabla 4.18: Bits del registro STATUS

BIT ROE

El bit **roe** se setea a 1 si el dato es recibido mientras el registro rxdata está lleno (esto es, mientras el bit rrdy es 0). Si existe error por desbordamiento de receptor, el nuevo dato sobrescribe al viejo. El bit **roe** es seteado a 0 cuando el software realiza una operación de escritura al registro STATUS.

BIT TOE

El bit **toe** es seteado a 1 cuando el dato es escrito en el registro txdata mientras el registro está aún lleno, (esto es, mientras el bit trdy es 0). Si existe un error por desbordamiento del transmisor, el nuevo dato es ignorado. El bit **toe** es seteado a 0 cuando el software realiza una operación de escritura al registro STATUS.

BIT TMT

El bit **tmt** es seteado a 0 mientras una transacción está en proceso y se setea a 1 cuando el registro de desplazamiento está vacío.

BIT TRDY

El bit **trdy** es seteado a 1 cuando el registro txdata está vacío.

BIT RRDY

El bit **rrdy**es seteado a 1 cuando el registro rxdata está lleno.

BIT E

El bit **e** es seteado a 0 cuando el software realiza una operación

BIBLIOGRAFÍA

[1] Wikipedia, Field Programmable Gate Array, http://es.wikipedia.org/wiki/Field_Programmable_Gate_Array, fecha de consulta junio 2012

[2] ElBitcoin.Org**,** Qué es un FPGA,

http://elbitcoin.org/que-es-un-fpga/, fecha de publicación marzo 2012

[3] Cristian Sisterna, MSc, FIELD PROGRAMMABLE GATE ARRAYhttp://dea.unsj.edu.ar/sisdig2/Field%20Programmable%20Gate%20Arrays_A.pdf, fecha de consulta junio 2012

[4] ALTERA Corporation, Cyclone IV FPGA DeviceFamily, http://www.altera.com/literature/hb/cyclone-iv/cyiv-51001.pdf, fecha de publicación mayo 2013

[5] Luis Muñoz, Herramientas de diseño para sistemas embebidos

http://www.slideshare.net/luisftube/herramientas-de-diseo-para-sistemas-embebidos, fecha de consulta junio 2013

[6] Wikipedia, Sistema embebido, http://es.wikipedia.org/wiki/Sistema_embebido, fecha de consulta junio 2013

[7] IDOSE Ingeniería, electrónica e informática, http://www.idose.es/faq/49-sistemas-embebidos/54-que-es-un-sistema-embebido, fecha de consulta junio 2013

[8] Jorge Rodríguez Araujo, Estudio del microprocesador NIOS II, 2010, fecha de consulta junio 2013.

[9] Altera Corporation, Nios II Processor Reference, http://www.altera.com/literature/hb/nios2/n2cpu_nii5v1.pdf, fecha de publicación mayo 2011.

[10] RenesasElectronicsCorporation, Serial Peripheral Interface (SPI) & Inter-IC (IC2) (SPI_I2C), http://documentation.renesas.com/doc/products/region/rtas/mpumcu/apn/spi_i2c.pdf, fecha de publicación abril 2010.

[11] Altera, Altera UniversityProgramLearningThroughInnovation, http://www.altera.com/education/univ/unv-index.html, Fecha de Consulta Mayo 2013.

[12] Altera, Altera UniversityProgram, http://vlsicad.eecs.umich.edu/BK/Slots/cache/www.altera.com/education/univ/unv-index.html, fecha de consulta Mayo 2013.

[13] Altera, Quartus II Programmer, http://www.altera.com/literature/hb/qts/qts_qii53022.pdf, Fecha de Consulta Mayo 2013.

[14] Altera, Usingthe NIOS II IntegratedDevelopmentEnviroment, http://www.altera.com/literature/hb/nios2/n2sw_nii52002.pdf, Fecha de Consulta Abril 2013.

[15] Altera, Cyclone IV FPGA FamilyOverview, http://www.altera.com/devices/fpga/cyclone-iv/overview/cyiv-overview.html, Fecha de ConsultaAbril 2013.

[16] Altera, DE0-Nano User Manual, ftp://ftp.altera.com/up/pub/Altera_Material/12.1/Boards/DE0-Nano/DE0_Nano_User_Manual.pdf, Fecha de Consulta Abril 2013.

[17]Genera, Aplicaciones Generales de una FPGA, http://www.generatecnologias.es/aplicaciones_fpga.html, Fecha de Consulta Marzo 2013.

[18] Quartus II Help, Assigning Device I/O Pins with Pin Planner, http://quartushelp.altera.com/13.0/mergedProjects/assign/ase/ase_pro_assigning_pins.htm, Fecha de ConsultaAbril 2013.

[19] Altera, Qsys, http://www.altera.com/products/software/quartus-ii/_subscription-edition/qsys/qts-qsys.html, Fecha de Consulta Mayo 2013.

[20] ESPOL, Datalogger usando NIOS II http://www.dspace.espol.edu.ec/handle/123456789/24353, fecha de consulta junio 2013.