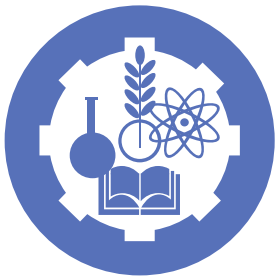
Instituto Tecnológico de Costa Rica
Escuela de Ingeniería en Electrónica



Diseño e implementación de un analizador de bus Secure Digital

Informe de Proyecto de Graduación para optar por el título
de Ingeniero en Electrónica con el grado académico de Licenciatura

Cadenux, LLC

Jorge Rivera Gutiérrez

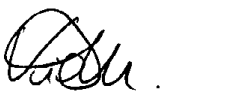
San José, Junio, 2006

Instituto Tecnológico de Costa Rica
Escuela de Ingenieria Electrónica
Proyecto de Graduación
Tribunal Evaluador

Proyecto de Graduación defendido ante el presente Tribunal Evaluador como requisito para optar por el título de Ingeniero en Electrónica con el grado académico de Licenciatura, del Instituto Tecnológico de Costa Rica.

Miembros del Tribunal

Luis Paulino Méndez Badilla
Profesor Lector



Victorino Rojas Madrigal
Profesor Lector

Carlos Badilla Corrales
Profesor Asesor

Los miembros de este Tribunal dan fe de que el presente trabajo de graduación ha sido aprobado y cumple con las normas establecidas por la Escuela de Ingeniería Electrónica.

Cartago, 27 de junio, 2006

Declaro que el presente Proyecto de Graduación ha sido realizado enteramente por mi persona, utilizando y aplicando literatura referente al tema, así como la información que ha suministrado la empresa Cadenux, LLC, y aplicando e introduciendo conocimientos propios.

En los casos en que he utilizado bibliografía, he procedido a indicar las fuentes mediante las respectivas citas bibliográficas.

En consecuencia, asumo la responsabilidad por el contenido de este proyecto.

San José, 27 de junio, 2006.

Jorge Manuel Rivera Gutiérrez
Cédula 1-1109-0460

Resumen

En el desarrollo de sistemas autocontenidos (embedded systems) se presentan necesidades de herramientas de desarrollo que permitan agilizar las tareas de programación y depuración de los programas. El presente proyecto consiste en el diseño e implementación de una herramienta de este tipo, que asista en el desarrollo de software para el control de tarjetas de memoria y comunicación conocidas como SD cards.

Para ello, se han utilizado circuitos lógicos programables de tipo FPGA programados en VHDL y un microcontrolador que captura y envia la información que se transmite entre un dispositivo y una tarjeta de este tipo y que pueda ser analizada en la computadora por el desarrollador y que este pueda depurar el software. Esta herramienta se comunica con la computadora por medio de un puerto USB de alta velocidad, utilizando para ello las capacidades del microcontrolador.

Palabras Clave: FPGA, microcontrolador, SD card, VHDL

Abstract

Development tools are needed to speed up the programming and debugging tasks in the development of embedded systems. This project consists of the design and implementation of a tool that will be able to assist in the development of software that controls communication and memory cards, also known as SD cards.

For this purpose, a VHDL programmed FPGA and a microcontroller have been used to capture and transfer the information transmitted between a device and a card of this type. This information is to be analyzed in the computer by the developer, to enable him to debug the software. This tool communicates with the computer by means of a high-speed USB port, using the capabilities of the microcontroller.

Keywords: FPGA, microcontroller, SD cards, VHDL

A mi abuelita Thelma,
por su ejemplo, su amor
y por siempre creer en mí.

Y a mi mamá,
por ser, como madre,
tan buena como ella,
por seguir con tanta humildad
ese ejemplo que ella nos dejó,
por su sacrificio y su constancia
que han sido la base de quien soy.

Un agradecimiento muy especial
a todo el personal de Cadenux,
por todo su apoyo, su disposición,
su ayuda técnica y su calidad humana.
Especialmente a Todd Fischer,
por la confianza que me tuvo
desde el primer momento.

Y a María Fernanda Salas,
por el apoyo que me diste,
y por tu amor incondicional
que me llena de vida.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

Las tarjetas Secure Digital o SD en el mercado de los dispositivos portátiles se han convertido en solución efectiva para las necesidades de expansión. Originalmente, la función principal de las tarjetas SD se limitó a almacenamiento de información, sin embargo, el bus utilizado para conectar estas tarjetas con los dispositivos móviles resultó muy útil y con capacidades más allá del almacenamiento. Esto permitió la aparición de tarjetas SD que expandieran no solo la capacidad de almacenamiento de los dispositivos, sino que extendieran sus capacidades. Estas tarjetas utilizan una extensión del bus original, llamada SDIO. Se encuentran en desarrollo en este momento muchos tipos de tarjetas, e inclusive algunos ya se encuentran en el mercado: cámaras digitales, interfaces para redes inálambricas (como Wi-Fi y Bluetooth), sistemas de localización GPS y otros. Una de las ventajas importantes de estas tarjetas es su capacidad para proteger su contenido a través de implementaciones de protección en la propia tarjeta. Esto elimina o disminuye la posibilidad de duplicar la tarjeta su fabricante incorpora la protección, para evitar violaciones a las leyes de derechos de autor.

Figura 1.1. Diagrama general del bus SD

1.1. Descripción del problema

Los dispositivos que utilizan tarjetas SD, de almacenamiento, entrada/salida o ambos, requieren controladores (*drivers*) para manejar todas las capacidades de cada tarjeta. Para escribir y depurar adecuadamente estos controladores, se requiere poder observar detalladamente la comunicación del dispositivo con la tarjeta. Esta depuración requiere no solo estudiar el comportamiento eléctrico de las señales (que podría realizarse de forma sencilla con un osciloscopio o un analizador lógico) sino también analizar el contenido de los paquetes de información que se transmiten en el bus.

En el desarrollo de sistemas autocontenidos (embedded systems) la necesidad de un sistema de análisis es aun mayor. En estos sistemas resulta más compleja la depuración de los controladores, puesto que no tienen una interfaz gráfica o un ambiente de desarrollo como el que tiene una computadora. La empresa Cadenux, en el desarrollo de controladores y paquetes de soporte para sistemas Linux autocontenidos, se encuentra en la necesidad de analizar el tráfico en el bus de estas tarjetas, para habilitar en el hardware de sus clientes la capacidad de manejar estos dispositivos.

El estudio del protocolo y las señales del bus SD es muy importante para la simplificación y la reducción del tiempo invertido en el desarrollo del *software* que maneja estos dispositivos, especialmente por la dificultad depurar la temporización de las señales. Los desarrolladores de controladores dedican mucho tiempo a observar señales en el analizador lógico o a utilizar el método de prueba y error para depurar los controladores de este tipo de dispositivos. Un analizador como el que se desarrolla en este proyecto les permite observar si se están cometiendo errores de temporización o de secuencia al establecer una comunicación con los dispositivos conectados al bus SD.

En el mercado existe una oferta muy limitada de analizadores de este tipo, a costos muy elevados. La empresa ha propuesto desarrollar uno para uso interno, y que eventualmente podría ser comercializado. La posibilidad de tener un analizador desarrollado por la empresa permitirá la actualización y personalización de este para satisfacer sus necesidades e implementar las mejoras que los empleados puedan proponer.

1.2. Solución seleccionada

Este proyecto tiene los siguientes requisitos:

- Reconocer información para las diferentes modalidades del bus.
- Comunicación con una computadora para registrar la información capturada, a través del bus USB.
- Usar el bus USB como fuente de poder.

- No interferir eléctricamente con el bus que está siendo analizado.
- Contar con un idicador de actividad en el bus, aun cuando no se esté registrando la información.

La empresa propuso basarse en un microcontrolador Cypress CY7C68013, el cual pertenece a la familia EZ-USB de Cypress e incorpora todas las características necesarias para la comunicación con una computadora usando USB y en un CPLD, que reciba las señales del bus a través de aisladores digitales y las transfiera al microcontrolador. Además, es necesario un extensor de bus, que es un dispositivo que se inserta en la ranura de la tarjeta y la tarjeta en él, estableciendo conexión eléctrica entre la tarjeta y el dispositivo, pero incluyendo pines para cada una de las líneas del bus, para establecer la conexión con analizadores lógicos o analizadores de protocolo.

El microcontrolador Cypress incluye varias formas de comunicarse a través del puerto USB. Se puede realizar el llenado de los buffers de transmisón a través del firmware que esté ejecutando el microcontrolador. También puede usarse el microcontrolador como una interfaz programable, de tal forma que envíe señales de lectura y escritura a través de pines dedicados a leer datos de otros sistemas, siendo en este caso el microcontrolador el *master* y el dispositivo externo el esclavo. Una última forma es configurarlo como esclavo, de tal forma que reciba señales del lectura y escritura a través de los pines dedicados.

Esta última opción resulta conveniente ya que con el dispositivo lógico programable pueden escribirse los datos directamente en los espacios de almacenamiento para ser enviados por el bus USB, y no se requiere de una sincronización entre los relojes de ambos dispositivos.

El protocolo de transferencia del bus SD incluye la posibilidad de que se escriba simultáneamente en el canal de datos y en el canal de comandos. Esto aumenta considerablemente el tamaño del diseño, ya que tienen que considerarse ambos canales independientemente. Para analizar cada canal se utiliza una máquina de estados que captura los datos, controlada por el reloj provisto por el bus. Además, es necesaria otra máquina de estados que envíe la información capturada al microcontrolador, utilizando un reloj independiente que permita que éste siga funcionando aun cuando no hay actividad en el bus.

Para implementar el diseño descrito anteriormente es insuficiente la lógica disponible en el CPLD que se encuentra disponible, por lo que es necesario utilizar un dispositivo más grande. Para efectos de desarrollo, se utilizó un FPGA Spartan-3 de Xilinx, el XC3S200, ya que está disponible entre el material adquirido para el proyecto, aunque posee más recursos de los necesarios.

Para el proceso de transmisión de la información se estableció una estructura de paquetes para informar a la computadora el tipo de transmisión que fue registrada.

En el computador, se programó un controlador sencillo que lee la información que es almacenada por el microcontrolador y la muestra en pantalla de forma ordenada.

En la figura 1.2 se muestra un diagrama general del sistema completo.

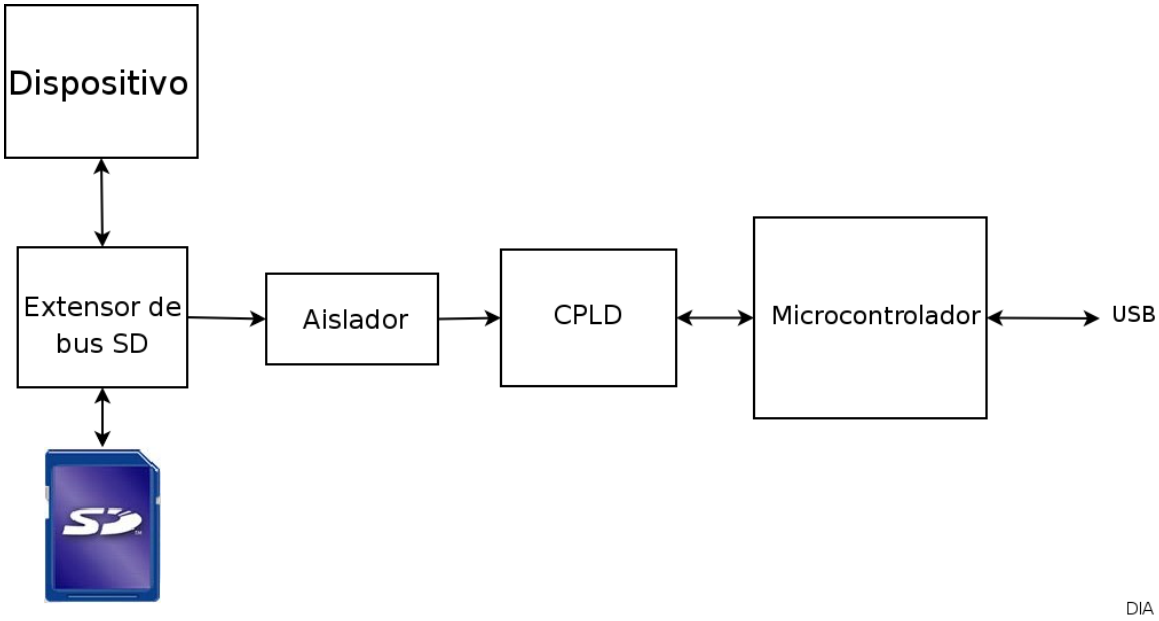


Figura 1.2. Diagrama general del sistema completo

Capítulo 2

Meta y Objetivos

2.1. Meta

Permitir el monitoreo y análisis del flujo de información en el bus SD de los dispositivos a los que la compañía brinda soporte.

2.2. Objetivo General

Diseñar y construir un prototipo funcional de un analizador de bus SD con conexión USB a la computadora.

2.3. Objetivos específicos

2.3.1. Objetivos específicos de hardware

- Aislar el bus del circuito analizador, de modo tal que reduzca la corriente que entre o salga del analizador desde o hacia el bus a una magnitud mucho menor que $10\,\mu$A.
- Preparar los datos que se obtengan del bus, para que puedan ser leídos de forma rápida por el microcontrolador, reduciendo al menos a la mitad el tiempo de lectura, utilizando un dispositivo de hardware programable.

2.3.2. Objetivos específicos de software

- Leer y procesar la información capturada del bus para ser transmitida a una computadora.
- Accesar la información capturada por el dispositivo desde el computador.

2.3.3. Objetivos específicos de implementación

- Comprobar el funcionamiento correcto del sistema usando un prototipo funcional.

2.3.4. Objetivos específicos de documentación

- Describir la estructura y funcionamiento del sistema, mediante una adecuada documentación.

Capítulo 3

Marco Teórico

El protocolo de comunicación SD es un estándar definido por la SD Card Association. El protocolo está descrito en una serie de documentos que se encuentran disponibles para sus miembros[7]. A continuación se presenta un resumen del funcionamiento de este protocolo y la descripción de algunos componentes que se utilizaron para el desarrollo del proyecto.

3.1. Bus SD

El bus SD, en su interfaz física, está basado en un estándar anterior, conocido como MulitMedia Card (MMC). El bus SD está diseñado para poder realizar transferencias de información utilizando diferentes protocolos, de esta forma se logra la compatibilidad con varios formatos de tarjeta. En una ranura SD se tiene la capacidad de leer tarjetas MMC, SD y SDIO, además de variantes de estas, como la MMC-RS (MMC Reduced Size, tamaño reducido), MMCplus, miniSD, microSD y otras, en algunos casos utilizando adaptadores debido a la forma física.[7] Sin embargo, en el caso de la MMCplus, no se aprovechan todas sus características, sino que funciona en una modalidad compatible con el MMC original.[5]

Los protocolos que se pueden utilizar para leer los diversos tipos de tarjetas, se detallan en la tabla 3.1.

Las tarjetas SD estándar tienen un total de nueve pines. Los nombres, números y funciones de esos pines se detallan en la tabla 3.2. En esta se detallan las funciones de los pines en cada modo. Como se observa en la tabla, en el modo SD están asignados cuatro pines para transmisión de datos y uno para transmisión de comandos, mientras que en el modo SPI únicamente están asignados un pin de entrada y uno de salida. Esto es una ventaja del modo SPI,

Tabla 3.1. Protocolos utilizados para la interfaz de los dispositivos

Tarjeta	SD	SDIO	mini y microSD	MMC	RS-MMC	MMCplus
Protocolos:	SPI SD 1-bit SD 4-bit	SPI SD 1-bit SD 4-bit	SPI SD 1-bit SD 4-bit	SPI MMC 1-bit	SPI MMC 1-bit	SPI MMC 1-bit

Tabla 3.2. Descripción de los pines de las tarjetas SD y MMC y sus funciones

Modo	SD[7]		SPI		MMC[4]	
PIN	Nombre	Función	Nombre	Función	Nombre	Función
1	CD/DAT3	Card Detect/ Datos Bit 3	$\overline{CS}$	Chip Select	RSV/NC	No conectada o siempre en alto
2	CMD	Command/ Response	DI	Entrada de datos	CMD	Command/ Response
3	VSS1	Tierra	VSS	Tierra	VSS1	Tierra
4	VDD	Alimentación	VDD	Alimentación	VDD	Alimentación
5	CLK	Reloj	SCLK	Reloj	CLK	Reloj
6	VSS2	Tierra	VSS2	Tierra	VSS2	Tierra
7	DAT0	Datos Bit 0	DO	Salida de datos	DAT0	Datos Bit 0
8	DAT1	Datos Bit 1	RSV	Reservado	ND	No disponible
9	DAT2	Datos Bit 2	RSV	Reservado	ND	No disponible

ya que está habilitado para un funcionamiento bidireccional simultáneo, sin embargo, para grandes transferencias en una sola dirección, el modo SD permite transferir cuatro veces más información en la misma cantidad de tiempo.

Un mismo dispositivo o anfitrión puede controlar varias tarjetas simultáneamente. En este caso, la línea CLK y las conexiones de tierra y alimentación pueden ser compartidas por todas las tarjetas, y cada tarjeta deberá tener líneas de datos y de comando independientes. El número de estas líneas depende del tipo de tarjeta.

3.1.1. Protocolo SD

En el protocolo SD, la línea denominada CMD es bidireccional, y por ella el sistema anfitrión envía comandos a la tarjeta. Cada comando consta de 48 bits, que incluyen un identificador de 6 bits, hasta 32 bits de argumento, un código CRC de 7 bits y 3 bits de señalización. El identificador de 6 bits permite seleccionar un total de 64 comandos diferentes. Cada tipo de comando recibe el nombre de CMD*nn*, donde *nn* es el número en base 10 del comando, por ejemplo CMD10 o CMD15.

Al inicializar o conectar una nueva tarjeta al bus, el bus entra en un modo de detección de tarjetas, en el que envía comandos por todas sus líneas CMD y espera respuestas de las tarjetas que tenga conectadas, para inicializarlas.

Tabla 3.3. Velocidades máximas de reloj para los diversos modos de funcionamiento en cada uno de los estándares

Estándar MODO	SD (v 1.0)	SD (v 1.10)	MMC[4] (v 3.31)
Identificación	400 kHz	400 kHz	400 kHz
Transferencia	25 MHz	50 MHz	20 MHz

Luego de identificarlas, tanto el anfitrión como la tarjeta pasan a modo de transferencia de datos.

Para algunos comandos enviados por el anfitrión la tarjeta debe enviar una respuesta. Esta respuesta es de 48 o 136 bits, dependiendo del comando que se haya enviado y es transmitido a través de la misma línea CMD. La codificación de la respuesta es dependiente del tipo de comando.

Ante solicitudes de transferencia de información los comandos y respuestas se envían a través de la línea CMD, mientras que los datos se transfieren ya sea por la línea DAT0 o por las líneas DAT[0-3], según se esté utilizando el modo estándar de 1 bit o el modo de bus ancho, de 4 bits. La definición del modo de transferencia que se va a utilizar será realizada a través de comandos transmitidos por la línea CMD al inicializar la tarjeta.

En la transferencia de información se utiliza una comprobación CRC de 16 bits, usando el polinomio CCITT. En el caso de la transmisión con el bus de 4 bits, se calcula un CRC para cada línea de datos individualmente.

La temporización de todas las transferencias se realiza a través de la línea CLK, que el anfitrión debe controlar, a una frecuencia que puede variar de acuerdo a su necesidad. Inclusive puede detener la señal CLK si lo necesita, para realizar la carga de datos o algún otro proceso. La velocidad del reloj está definida de acuerdo con la configuración definida en el proceso de identificación de tarjetas. En la tabla 3.3 se muestran las velocidades máximas de acuerdo con el protocolo utilizado y el modo en el que se encuentren.

Durante el proceso de identificación de tarjetas, cada tarjeta se identifica e indica su velocidad máxima y otros parámetros. El anfitrión puede alterar la velocidad de reloj e inclusive detenerlo, sin embargo, se requiere que al terminar una operación de cualquier tipo el anfitrión mantenga en funcionamiento el reloj por ocho ciclos completos, para permitirle a la tarjeta completar la operación.

3.1.2. Protocolo MMC

Dado que el protocolo SD está definido sobre el protocolo MMC, ambos son similares en su funcionamiento. La diferencia se encuentra en que el protocolo SD implementa algunas nuevas instrucciones para manejar las funciones nuevas que incorpora.

3.1.3. Protocolo SPI

En el caso del protocolo SPI, el funcionamiento es similar. Los comandos se envían por la línea SDI a la tarjeta, y esta responde por la línea SDO. Se utiliza el mismo formato de comandos y respuestas del protocolo SD, excepto porque todos los comandos que sean enviados por el anfitrión deben ser respondidos por la tarjeta. Esto incorpora dos nuevos formatos de respuesta, uno de 8 bits y otro de 16 bits. En estos se incluye una forma de comunicar al anfitrión el encuentro de un fallo de lectura, que en el bus SD es detectado por una temporización.

Además, se requiere el envío de una señal en la línea SDI, que indica el inicio de una transmisión de datos cuando el anfitrión le envía datos a la tarjeta. Si la tarjeta está ocupada grabando la información o en algún proceso, puede enviar una señal al anfitrión para indicarle que para continuar la transferencia de información debe esperar.

3.1.4. Protocolo SDIO

En términos generales, una tarjeta SDIO utiliza la misma configuración y protocolo que una tarjeta SD. Sin embargo, para poder interconectar un anfitrión con una tarjeta SDIO de forma más eficiente, se incorpora una línea de interrupción, mediante la cual una tarjeta puede solicitar la atención del afitrión cuando así lo requiera.[6]

La línea utilizada para esta función es la línea 8, que como se muestra en la tabla 3.2, corresponde a la línea DAT1. En modo de transferencia de 1 bit, no hay restricción de cuando se pueda solicitar una interrupción. En el modo de transferencia SPI sucede lo mismo, sin embargo, algunas tarjetas solo pueden solicitar interrupciones si se ha seleccionado esta con el correspondiente pin $\overline{\text{CS}}$. En el modo de transferencia de 4 bits, el pin DAT1 debe compartirse entre el bus de datos y la interrupción. Por ello, hay una limitación en cuanto a los momentos en que puede solicitarse la interrupción. El periodo de interrupción abarca aquellos momentos en que no hay transmisiones de datos, incluyendo los momentos en que se transmite una respuesta de parte de la tarjeta.

En este protocolo se incorporan nuevos comandos para la transferencia de información, para comunicarse con los diversos dispositivos que se diseñen para el bus. No se definen funciones específicas para tarjetas específicas, sino que se definen funciones generales de entrada y salida de datos, y el manejo de las funciones específicas de cada dispositivo se realiza en el software del anfitrión.

3.2. Componentes a utilizar

3.2.1. Microcontrolador Cypress

El microcontrolador Cypress CY7C68013A es un microcontrolador basado en microprocesador 8051, que incorpora un transductor USB 2.0 en el mismo chip y un controlador de interfaz serial (SIE). Esto simplifica de forma importante la programación, ya que tiene implementadas internamente las capacidades del bus USB. El SIE maneja la mayor parte de las funciones del protocolo USB 1.1 y 2.0 en hardware, y esto aumenta la eficiencia del dispositivo, ya que no tiene que dedicar tanto tiempo de procesamiento a la transmisión de datos por USB. El microcontrolador está disponible en tres empaquetados distintos, de 56, 100 y 128 pines.[2]

El consumo de corriente de este microcontrolador es siempre inferior a 85 mA. Esto cumple con la especifiación de USB para periféricos alimentados a través del mismo bus, lo cual permite simplificar de manera considerable la circuitería, ya que no es necesario incorporar elementos de alimentación externa. El funcionamiento del circuto se realiza a 3.3 V, y por ende, requiere de un regulador de tensión que reduzca al valor de tensión de 5 V que provee el USB a 3.3 V.

La velocidad de operación del microcontrolador puede ser 12 MHz, 24 MHz o 48 MHz y utiliza cuatro ciclos de reloj por instrucción.

La memoria interna del microcontrolador es de 16 kB. Esta memoria es compartida entre datos y código. El código puede ser cargado automáticamente a través de un controlador I^2C al momento del inicio del sistema, puede ser cargado via USB al inicializar o ser accesado directamente por un bus de datos y un bus de dirección, en el caso del empaquetado de 128 pines.

El controlador I^2C puede utilizarse para propósito general luego de ser cargado el código a la memoria. Esto es útil para la incorporación de periféricos que puedan necesitarse, tales como memorias para almacenar datos o un reloj de tiempo real.

3.2.2. Dispositivos de lógica programable Xilinx

Los dispositivos de lógica programable disponibles de Xilinx abarcan la tecnología CPLD (complex programmable logic device, dispositivo de lógica programable complejo) y la tecnología FPGA (field-programmable gate array, arreglo de compuertas programable en campo).[11, 10]

La lógica programable resulta muy útil en el desarrollo de hardware, ya que permite probar varias configuraciones

de diseño a bajo costo reprogramando el dispositivo para realizar todas las modificaciones que sean necesarias para llegar al comportamiento requerido.

Dentro de estos dispositivos se incluyen compuertas lógicas que pueden configurarse adecuadamente para realizar operaciones lógicas y matemáticas sencillas. También pueden implementarse en ellos diversos dispositivos electrónicos sencillos, como registros y otros.

Los FPGAs son dispositivos que, además de incorporar compuertas lógicas en una cantidad mucho mayor a los CPLDs, incluyen además características como memoria interna y funciones de alto nivel incorporadas, como sumadores o multiplicadores. Los FPGAs también son más dinámicos, ya que permiten un nivel de reprogramación mayor, inclusive durante el funcionamiento, ya sea como una actualización o como parte del funcionamiento dinámico del dispositivo.

Para la programación de estos dispositivos Xilinx tiene disponible un software gratuito llamado ISE WebPack, que permite la programación y modificación de CPLDs y FPGAs.

3.2.3. Aisladores Analog Devices ADuM

La serie ADuM de Analog Devices la conforman aisladores digitales basados en la tecnología *i*Coupler. Estos aisladores muestran una copia de la señal de entrada en la salida. Internamente los aisladores incluyen un acople a través de transformadores integrados en el chip, lo cual logra disminuir la corriente a valores muy pequeños, típicamente de 10 nA. Su consumo de corriente es mucho menor al de otros sistemas que cumplen la misma función, como los optoacopladores.[1]

Debido a que no requieren ningún dispositivo de control externo, simplifican considerablemente el diseño. Su velocidad de respuesta puede llegar a ser de hasta 120 Mbps. La alimentación del dispositivo tiene amplio rango entre 2.7 V y 5.5 V.

Capítulo 4

Procedimiento metodológico

A continuación se presentan algunos pasos que fueron seguidos para el desarrollo del proyecto. Se encuentran agrupados por etapas y no necesariamente en orden cronológico.

4.1. Reconocimiento y definición del problema

- Entrevistar a los empleados de la compañía con respecto a las necesidades que han tenido al trabajar con el bus SD.
- Observar las formas de onda reales que se presentan en el bus SD durante una transmisión, utilizando un analizador lógico.

4.2. Obtención y análisis de información

- Leer y estudiar detenidamente los detalles sobre el funcionamiento del bus SD en todas sus modalidades.
- Observar y estudiar el funcionamiento de otros analizadores de bus, como el analizador de bus USB.
- Reconocer los circuitos integrados propuestos para la solución del problema y estudiar su funcionamiento.
- Investigar otros posibles componentes que puedan utilizarse para resolver el problema.

4.3. Evaluación de las alternativas y síntesis de una solución

- Comparar los posibles componentes y estudiar sus capacidades. Definir ventajas y desventajas de cada uno.
- Seleccionar y hacer una lista de los componentes que más se amolden a las necesidades del proyecto.
- Diseñar la interfaz entre el bus y el circuito, de tal forma que no se causen alteraciones en el bus al estar el analizador conectado.
- Diseñar un circuito lógico que convierta la información que pasa por el bus en paquetes de ocho bits, que puedan ser leídos por el microcontrolador.
- Establecer un formato para ser utilizado en la transmisión de paquetes por medio del puerto USB y documentarlo para el diseño del programa en la computadora.
- Escribir un programa en el microcontrolador que interprete y prepare la información capturada en el bus para ser transmitida a la computadora.
- Escribir un programa en el computador que permita accesar la información capturada.

4.4. Implementación de la solución

- Comprobar el correcto aislamiento eléctrico entre el bus y el resto del circuito.
- Implementar el circuito lógico diseñado en un dispositivo lógico programable.
- Construir un prototipo del circuito diseñado, utilizando herramientas de desarrollo.
- Realizar pruebas al prototipo, que den como resultado el correcto funcionamiento del mismo.

4.5. Reevaluación y rediseño

- Estudiar la velocidad de respuesta del circuito lógico programable.
- Diseñar un método para aumentar la precisión de la medición de las señales presentes en el bus.
- Agregar un dispositivo de almacenamiento masivo de datos para almacenar información sobre tráfico en el bus y extraerla posteriormente.
- Analizar el tamaño del diseño para elegir los componentes más aptos para la optimización de costos de su implementación.

Capítulo 5

Descripción de la solución

La solución desarrollada se basa en la solución propuesta por la empresa, implementada utilizando los componentes y estrategias que se escogieron durante el desarrollo. En esta sección se explica la escogencia de éstas.

5.1. Análisis de soluciones y selección final

La información del bus debe ser procesada adecuadamente, para luego poder ser enviada a la computadora. Para este procesamiento se propuso la utilización de un CPLD. Además se requiere un dispositivo para enviar los datos usando un puerto USB de una computadora. Para esto se propuso utilizar un programa en un microcontrolador EZ-USB de Cypress.

5.1.1. Procesamiento de la información

Para procesar la información se diseñó un sistema en VHDL que fuera capaz de detectar y procesar la información proveniente del bus. Para la implementación del prototipo se adquirieron dos módulos de desarrollo de la marca Xilinx con los dispositivos descritos en la tabla 5.1

La implementación preliminar que se desarrolló excedió la capacidad de ambos CPLDs, sin embargo, requirió de menos de un 25 % de los recursos del FPGA Spartan-3. Las capacidades de los CPLDs se vio ocupada principalmente por su falta de capacidad de registros, ya que se hace necesario implementar grandes registros para almacenar las

Tabla 5.1. Características de los dispositivos de lógica programable disponibles.

Tipo	Familia	Modelo	Puertos I/O	Registros	Compuertas Lógicas	Velocidad [MHz]
CPLD	XC9500XL	XC9572XL	34	72	1600	178
CPLD	CoolRunner-II	XC2C256	118	256	6000	256
FPGA	Spartan-3	XC3S200	173	30000	200000	750

cadenas de hasta 136 bits. Además, la implementación de la interfaz con el microcontrolador requiere de gran cantidad multiplexores, para evitar el desplazamiento de registros luego de la captura de datos, ya que éste puede requerir mucho tiempo.

El Spartan-3 contiene una gran cantidad de registros disponibles en forma de RAM. Su escogencia para el desarrollo del proyecto fue determinada por su capacidad de almacenamiento y su alta disponibilidad de elementos lógicos.

5.1.2. Interfaz USB

La serie de microcontroladores EZ-USB fue escogida por la facilidad de implementación de sistemas que utilicen comunicación USB. El modelo CY7C68013A posee varias características que lo hacen ideal para el proyecto: puede ser programado para ser el dispositivo maestro y leer información de un dispositivo de almacenamiento y también para actuar como un dispositivo que se encargue del envío de información por el bus USB, escrita por un maestro externo.

La ventaja de la utilización del microcontrolador como dispositivo maestro es que la escritura del programa para el dispositivo se realiza en C y éste permite una amplia gama de funciones que permitirían procesar la información. Sin embargo, se requeriría también de una compleja temporización, ya que habría que proveer señales de reloj al sistema externo para leer los datos y basar el sistema de procesamiento en esa señal.

Por otro lado, utlizar el microcontrolador en modo esclavo implica la necesidad de describrir una mayor cantidad hardware en VHDL e implementar funciones de procesamiento en hardware. Sin embargo, todo el proceso está sincronizado por un reloj para el FPGA, el cual también puede utilizarse para sincronizar la escritura de la información en el micrcontrolador. Esto simplifica considerablemente la temporización de ambos dispositivos.

Se eligió utilizar el microcontrolador como esclavo. Esto implica una programación muy sencilla: basándose en un marco de programa provisto por el fabricante, tan solo fue necesario escribir rutinas que inicializaran los registros de configuración del dispositivo, para utilizar los espacios de almacenamiento para transmitir por el puerto USB adecuadamente.

5.2. Descripción del hardware

El hardware diseñado consta de dispositivos físicos conectados de una forma muy sencilla y de una lógica sintetizada en un FPGA. A continuación se describe el proceso seguido para ambos diseños.

5.2.1. Conexión física de los dispositivos

Para el método de transmisión de información que se elegió entre el microcontrolador y el FPGA, se requiere de una conexión de los dispositivos utilizando 16 líneas de datos y varias líneas de señalización: *slave write*, *slave read*, *slave output enable*, *packet end*, *FIFO address* y una señal de reloj. También se pueden utilizar opcionalmente 4 líneas (*Flag A*, *Flag B*, *Flag C* y *Flag D*) provenientes del microcontrolador que indican el estado de los espacios de almacenamiento.

Se dispuso de conexiones para todas las líneas, inclusive las opcionales. Esto requirió la selección de 27 puertos del FPGA para esto. Los puertos se seleccionaron de tal forma que se simplificara el cableado de los módulos de desarrollo. Estos módulos tienen conectores similares y fue posible hacer un cableado muy simple.

Además, es necesario establecer una conexión entre el FPGA y el bus que se está analizando. Para ello se utilizaron dos aisladores digitales de cuatro puertos. En uno se ubicaron las cuatro líneas de datos del bus SD. En el otro se ubicaron las líneas CMD y el reloj. Las salidas de los aisladores fueron conectadas a 6 puertos del FPGA, seleccionados también de forma que se simplificara el cableado y de tal forma que la señal de reloj del bus coincidiera con un puerto especializado del FPGA (GCK1) que puede brindar señal de reloj a toda la lógica del dispositivo. El módulo de desarrollo incluye un cristal de 50 MHz, cableado a otro puerto especializado del FPGA (GCK0). Esta señal se utilizó como parte de la señalización para las máquinas de estados.

Se utilizaron un total de 34 puertos del FPGA. Éste dispone de un total de 173. En la figura 5.1 se muestra un esquema general de la conexión del sistema. Durante el desarrollo del proyecto se utilizaron más puertos como puntos de control para la depuración del sistema. Estos puntos de control fueron asingados a diversos pines del FPGA. En algunos casos se usaron pines que tienen conexión externa en el módulo de desarrollo que fueron conectados al analizador lógico y en otros casos se usaron algunos que están conectados al puerto serial, a un display de cuatro dígitos y a varios LED's que están presentes en este módulo.

Además, en el hardware es necesario contar con las memorias donde está almacenada la programación del FPGA y del microcontrolador. En el caso del FPGA, esta es una memoria flash conectada por unos puertos dedicados del FPGA y que es leída al ser inicializado el dispositivo. En el caso del microcontrolador, es una memoria EEPROM conectada por I^2C que también es leída en la inicialización del dispositivo.

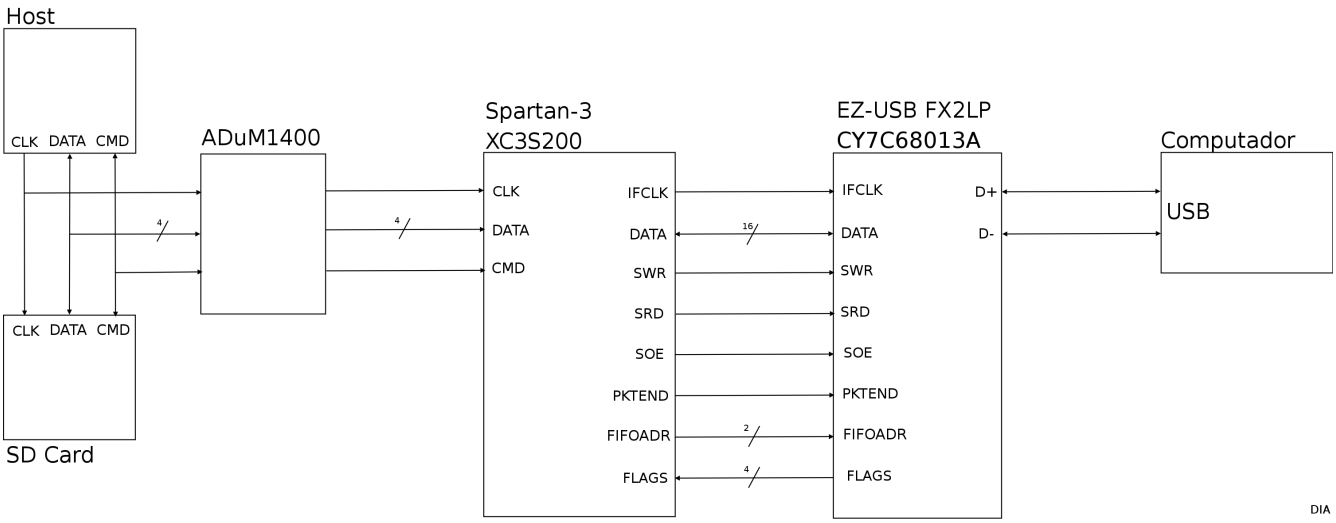


Figura 5.1. Diagrama del sistema completo

La alimentación de todos los dispositivos debe obtenerse del puerto USB. El puerto USB provee 5V, sin embargo, los dispositivos requieren de tres tensiones de alimentación distintos: el microcontrolador requiere de 3.3V y el FPGA requiere de 1.2V para alimentar el núcleo, una tensión auxiliar de 2.5V para optimizar ciertas funciones lógicas y una tensión que alimente la interfaz externa, que en este caso debe ser igual a la del microcontrolador. Para proveer estas múltiples tensiones se deben utilizar reguladores de tensión que ya están incorporados en los módulos de desarrollo.

5.2.2. Protocolo de envío de información

Para la escritura de la información en el puerto USB se estableció un protocolo. Este protocolo se describe utilizando BNF, como se muestra en la figura 5.2. El protocolo se estructuró en bloques de 16 bits, para que la escritura fuera por bloques con contenido relacionado. Así, el primer bloque contiene un inicio de paquete de 4 bits y 12 bits de banderas que indican el tipo de transmisión que se capturó, el segundo y el tercero contienen el registro del momento en que fue capturada la transmisión. Luego de estos se transmite el paquete de información, que consta de un máximo de 500 bytes, en 250 bloques de 16 bits y se concluye con dos paquetes que indican el tamaño, orden de secuencia y la fragmentación de paquetes, si se hizo, además de un indicador de fin de paquete.

5.2.3. Lógica sintetizada

La síntesis de la lógica requerida se realizó construyendo máquinas de estados de Mealy modificadas, de tal forma que de un estado se pueda pasar a varios estados, dependiendo de una serie de condiciones. El diseño realizado

<protocolo>	:==	<sop-sync-0.5><flags-1.5><timestamp-4>
		<data><sequence-0.5><lenght-1.5><frag-0.5><eop-1.5>
<sop-sync-0.5>	:==	0x5, inicio del paquete
<flags-1.5>	:==	descriptor del tipo de información:
		Bit 11: Comando
		Bit 10: Respuesta
		Bit 9: Datos modo 1 bit
		Bit 8: Datos modo 4 bits
		Bit 5: Interrupción SDIO
		Bit 3: Respuesta Larga
		Bits 7,6,4,2-0: reservados
<timestamp-4>	:==	registro de tiempo de 4 bits
<data>	:==	hasta 500 bytes de datos capturados
<sequence-0.5>	:==	4 bits de número de paquete cuando hay que dividir un paquete de datos
<lenght-1.5>	:==	longitud del paquete transmitido, contando encabezados
<frag-0.5>	:==	descriptor de fragmentación del paquete:
		Bit 3: este paquete es un fragmento
		Bit 2: este paquete es el último fragmento
		Bits 1,0: reservados
<eop-2>	:==	0xAAA, final de paquete

Figura 5.2. Protocolo de envío de información

consiste en tres máquinas de estados: una para la detección de actividad en la línea de comandos (*CMDdet*), una para la detección de actividad en la línea de datos (*DATdet*) y una para la comunicación con el microcontrolador (*uCinterface*). Para el registro del tiempo de inicio de cada comando o transimisión de datos se implementó un contador de 32 bits que cuenta a la velocidad del reloj del FPGA, que se definió en 50 MHz, lo cual permite ubicar las transmisiones con una presición de 20ñs.

Las máquinas de estados generadas se basaron en la creación de un tipo de señal llamado *state* y los diferentes valores que esta señal puede tomar corresponden a cada uno de los estados que la máquina de estados tiene que recorrer. Para llevar el control del estado de la máquina se crean dos variables: *CurrentState* para indicar el estado actual y *NextState* para indicar el estado siguiente. La escogencia del estado siguiente se realiza utilizando el estado actual y una o más variables externas o internas, mediante una construcción de tipo `case` para el análisis de los estados y construcciones de tipo `if` para la evaluación de las variables externas e internas. En algunos casos el siguiente estado puede depender de muchas variables y por ende, tener más de dos posibles estados siguientes. Esta es la modificación prinicipal hecha a las máquinas de estado teóricas, puesto que usualmente se opta por un modelo en que se evalúa una sola condición en cada estado para pasar al siguiente. Esta modificación incrementa la lógica de forma considerable, ya que deben implementarse complejos decodificadores para determinar uno u otro salto. Sin embargo, esta lógica es generada por el sintetizador, lo cual lo hace una tarea sencilla, pero que requiere de una cantidad grande de recursos del dispositivo programable. La ventaja más clara de esto es la posibilidad de reducir la cantidad de estados, y por ende aumentar la velocidad del dispositivo. Si se utilizaran varios estados intermedios para que los saltos fueran selecciones más simples, los estados intermedios causarían tiempos de espera en el procesamiento de los datos que podrían causar pérdidas de información o corrupción de los mismos.

CMDdet

La máquina de estados para detección de comandos tiene la dificultad de que debe recibir tanto comandos y respuestas de 48 bits, como respuestas especiales de la tarjeta que constan de 136 bits. Estas respuestas especiales son de dos tipos y no se diferencian en nada a una respuesta normal durante el inicio de la transmisión. El dispositivo anfitrión espera una respuesta de 136 bits después de solicitarlo enviando uno de tres comandos. Por ende, cada comando debe ser analizado para determinar si la siguiente actividad en el bus corresponderá a una respuesta de 48 bits o una de 136 bits.

Para simplificar el sistema, la salida de este módulo se definió como un bus de 136 bits, en el cual se transmite el comando o respuesta corta que se capturó en los 48 bits menos significativos del registro, o en el caso de una respuesta larga, en los 136 bits del bus.

Además, se dispuso un registro de 32 bits que almacena el valor actual del contador cuando se inicia el proceso de almacenamiento de un comando o respuesta y tres bits que indican que se ha completado la recepción de un comando, una respuesta corta y una respuesta larga.

La entrada al proceso de almacenamiento se da cuando se detecta un valor 0 en la línea CMD cuando hay un flanco positivo en el reloj. En el siguiente pulso del reloj se determina si el paquete que se está transmitiendo es un paquete de comando o de respuesta. Si éste es un 1, el paquete es una comando y se deben contar 46 bits más para concluir la transmisión. Si es un 0 y el comando anterior fue CMD2, CMD9 o CMD10, es una respuesta larga y deben contarse 134 bits.

Cuando un paquete ha sido almacenado, se notifica mediante el bit indicador del tipo de paquete que se capturó y se espera una confirmación por parte del módulo de comunicación con el microcontrolador y cuando se recibe se vuelve al estado de espera.

La señal de reloj que provee la señalización para el cambio de estados es un reloj compuesto, ya que durante la transmisión se usa el reloj del bus, para detectar el momento en el que se transmite cada bit, pero cuando concluye la transmisión el reloj del bus puede detenerse. Esto causaría que la máquina de estados se detenga y pierda información. Para subsanar este problema, la señalización en estos estados se obtiene del reloj principal del FPGA, que es el usado para señalizar los estados del módulo uCinterface.

En la figura 5.3 se ilustra el diagrama de estados de este módulo.

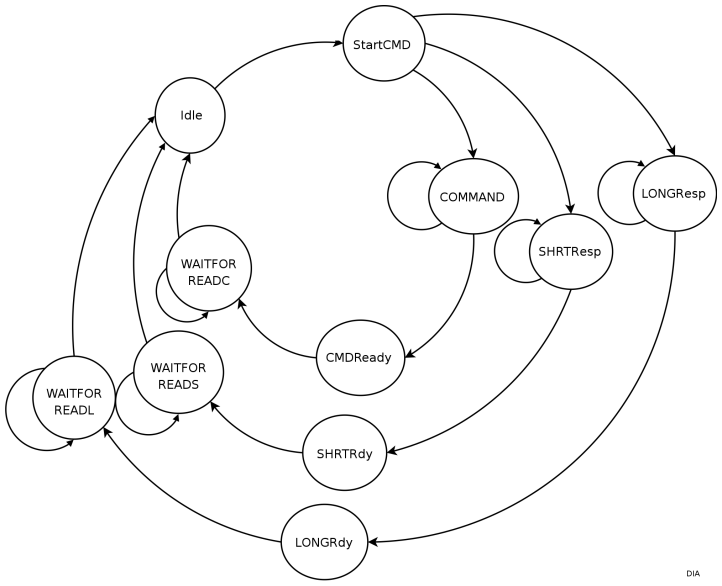


Figura 5.3. Diagrama de estados del módulo CMDdet

DATdet

La máquina de estados que detecta las transmisiones sobre las líneas de datos espera una de tres condiciones: que se presente un 0 en la línea DAT0, indicando el inicio de un paquete de datos en modo 1-bit; que se presente un 0 en la línea DAT1, indicando el inicio de una interrupción o que se presenten simultáneamente ceros en todas las líneas de datos, indicando el inicio de un paquete de datos en modo 4-bits.

Se destinaron dos registros de 16 bits que se llenan alternativamente, para que no se sobreescriba o se descarte información mientras se espera a que el módulo uCinterface lea la que se almacenó.

En el caso del modo de un bit se usa un estado para cada registro y un contador que se incrementa para saber cuando se han almacenado los 16 bits de cada uno. En el caso del modo de cuatro bits se usa un estado por cada grupo de bits de cada uno de los registros. Los estados se identifican con las letras HH, HL, LH y LL indicando en cual de los cuatro grupos de cuatro bits se está almacenando la información.

En ambos casos se espera una señal *STOP* proveniente del módulo uCinterface que le indica que no debe seguir almacenando datos al concluir el grupo de 16 bits que está siendo almacenado.

Cuando se inicia una interrupción no hay información que almacenar, sino que se regresa al estado de espera.

Para todos las transmisiones se implementa un estado especial de inicio, que se indica con una letra f al final del

nombre. Durante este estado se almacena en un registro de 32 bits el contenido del contador de tiempo. Este valor es el registro de tiempo que corresponde a la captura y es leído por el módulo uCinterface durante la inicialización de la transmisión.

En la figura 5.4 se ilustra el diagrama de estados de este módulo.

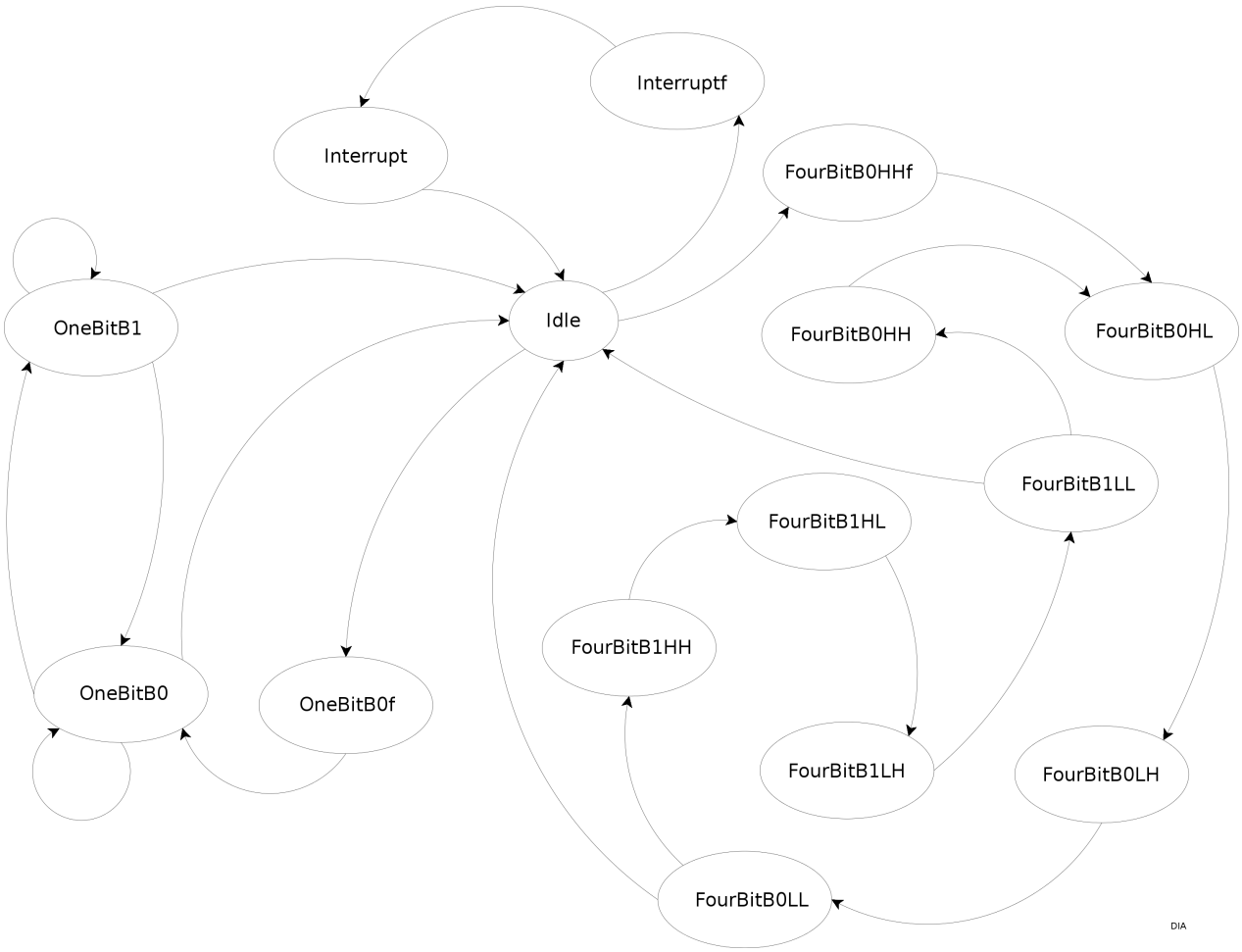


Figura 5.4. Diagrama de estados del módulo DATdet

uCinterface

Esta máquina de estados debe reconocer cuando hay un paquete capturado disponible para ser enviado al microcontrolador. Su estado de espera termina cuando una de las máquinas de estado de detección de paquetes notifica con una de las seis líneas que hay información lista.

El primer paso debe ser la configuración de la interfaz con el microcontrolador. Para ello se elige la dirección en la que se va escribir. Para paquetes de datos se eligió utilizar la dirección 0x86, correspondiente al valor 10 en las líneas de *fifo address*, y para paquetes de comandos o respuestas se eligió utilizar la dirección 0x82, correspondiente al valor 00.

Luego debe esperarse a que el microcontrolador confirme que esa dirección está disponible. Esto lo informa utilizando la línea *flag B*. Cuando esta línea es 1 se inicia el proceso de escritura, de acuerdo con la estructura especificada en la sección 5.2.2. Los primeros bloques que se escriben tienen una estructura común para todas las transmisiones, excepto porque cambian los valores. Estos son escogidos de acuerdo con las señales de aviso de los otros módulos.

En el caso de las transmisiones de comandos y respuestas, la información que va a ser transmitida es almacenada en registros del módulo interfaz durante la inicialización y luego es transmitida bloque por bloque al microcontrolador. Esto permite liberar la máquina de estados de captura para que reciba el siguiente bloque, lo cual es útil, especialmente si el bus USB tiene mucho tráfico o si la respuesta en la línea de comandos llega muy rápidamente.

En el caso de las transmisiones de datos, se almacena solo la información del tiempo durante la inicialización, mientras que las lecturas de los datos se transmiten directamente al microcontrolador conforme se capturan.

En el microcontrolador se encuentra asignada la señal *flag A* como un indicador de nivel programable. La programación de esta se desarrolla en la sección 5.3.1. Se ha programado adecuadamente para que dé un valor 1 cuando queda solo suficiente espacio en el bloque de alamcenamiento para almacenar los bloques de cierre de paquete. El bloque de almacenamiento máximo se obtiene cuando se utiliza la configuración de USB para funcionar en *high-speed*, este corresponde a 512 bytes. Para simplificar el diseño del dispositivo, se ha establecido como un requisito que el dispositivo se conecte a un puerto *high-speed*. Las transmisiones de comandos son mucho más pequeñas que el tamaño máximo del bloque, pero las transmisiones de datos pueden llegar a medir 512 bytes o más, así que deben ser fragmentados cuando sobrepasan los 500 bytes que se tienen como máxima transmisión en el bloque de almacenamiento de USB. Al transmitir una larga cadena de datos, esta señal se utiliza como indicador de que debe detenerse la captura para escribir los últimos datos del paquete e iniciar un nuevo paquete. Para ello se utilizan los estados alternativos (*SeqincLenght, EOPinc, afterEOPinc*) para distinguir entre el final de un paquete que ocupó fragmentación y uno que no.

La señalización para los cambios de estado se realiza con un reloj propio del FPGA que se obtiene de un oscilador de 50 MHz que está incluido en el módulo de desarrollo.

En la figura 5.5 se ilustra el diagrama de estados de este módulo.

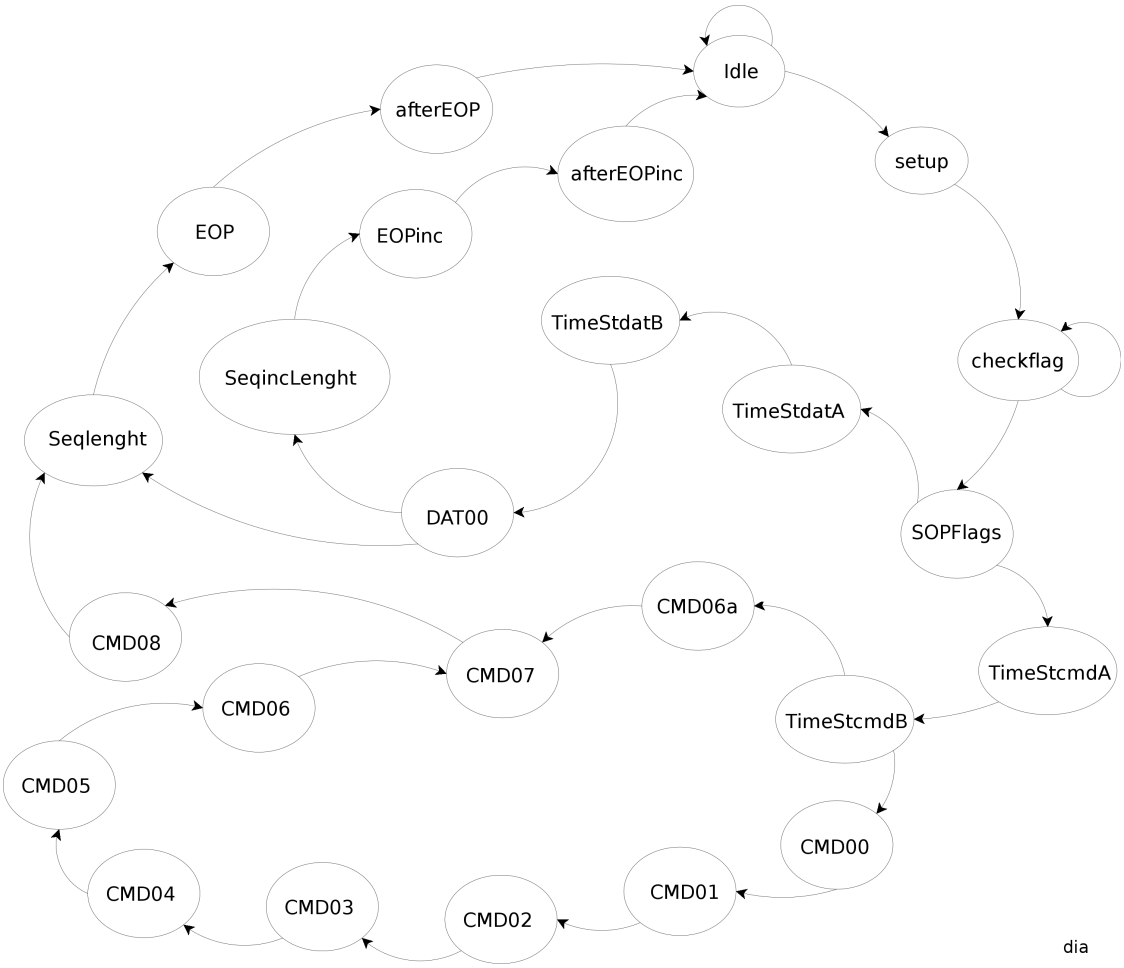


Figura 5.5. Diagrama de estados del módulo uCinterface

5.3. Descripción del software

El software del proyecto consta de dos partes diferentes: el software que se ejecuta en el microcontrolador y el software que se ejecuta en la computadora que recibe la información.

5.3.1. Microcontrolador

Este software está escrito en lenguaje C. El compilador usado fue el que provee el fabricante del microcontrolador con el sistema de desarrollo, creado por Keil Embedded Development Tools. Éste es una versión limitada de la herramienta C51 de esta empresa. Esta es completamente funcional, pero permite un tamaño máximo de objetos de 4 kilobytes.

El desarrollo de la rutina del microcontrolador se basó en un marco de programa (*framework*) provisto por Cypress. Esta rutina consta de 3 partes básicas: el archivo principal que contiene la función `main()`, el archivo de rutinas de servicio a interrupciones (ISRs) y el descriptor de USB.

main()

Este archivo, llamado `fw.c`, contiene varias llamadas a archivos de configuración. Es provisto por el fabricante, así como lo son también estos archivos de configuración. Este archivo debe modificarse adeacuadamente para incluir la incialización de todos los registros de función especial del microcontrolador que sean necesarios. Esto se logra con la función `main()`. Al inicializar, el microcontrolador lee el código correspondiente a esta función y lo ejecuta.

Los registros que fueron configurados en este caso fueron los registros que regulan el funcionamiento de los sectores de almacenamiento de datos que se van a enviar o recibir a través de USB. El microcontrolador puede utilizar 4 direcciones diferentes, pero con distintas características que son programables: la dirección (*in*, hacia la computadora o *out*, hacia el microcontrolador), la cantidad de paquetes que se pueden escribir en cada dirección mientras se espera el momento en que la información sea solicitada por la PC (dos, tres y hasta cuatro paquetes por dirección), la velocidad a la que puede conectar y otros. Se seleccionó configurar 3 direcciones en una combinación de 2 direcciones *in* (0x82 y 0x86) y una *out* (0x08), en la cual la direcciones *in* pueden almacenar 3 paquetes de 512 bytes cada una y la dirección *out* puede almacenar 2. Esto permite, por ejemplo, que si el software en la PC no ha leído la información de la dirección 0x82 cuando llega un paquete a la dirección 0x86 u otro paquete a la dirección 0x82, el primer paquete no es desechado, sino que queda a la espera de la lectura por parte de la PC. Esto puede suceder debido a que el protocolo USB es un protocolo manejado por el *master*, en este caso la computadora, y el paquete no puede ser enviado hasta que haya una lectura iniciada por el maestro.

Después de la inicialización del dispositivo, se ejecuta iterativamente usando la construcción `while(true)`, una función que consulta constantemente la presencia de información que haya sido guardada en los espacios de almacenamiento y deba ser procesada por el microcontrolador.

ISRs y otros

Este archivo, llamado `SDBA.c`, contiene las rutinas con que se atienden las interrupciones causadas por las distintas funciones del microcontrolador y además contiene el código de la función mencionada anteriormente que consulta la presencia de informacin.

En este caso no se requiere que estas rutinas tengan funciones especiales, ya que las características de interrupciones no se están utilizando y la lectura y escritura de la información se hace de forma independiente del

procesamiento del microcontrolador.

Descriptor de USB

Este archivo, llamado `descr.a51`, es un archivo de ensamblador que contiene datos en un orden específico y con etiquetas apropiadas para ser utilizados por el *framework*. Contiene la descripción de las características del dispositivo USB. Es importante coordinar correctamente el contenido de este archivo con la configuración programada por el `main()`, debido a que no se interpreta directamente de ahí, sino que se escribe independientemente. El sistema operativo va a recibir esta información, sin embargo, si hubiera una diferencia entre ésta y la configuración real, podrían darse resultados inesperados.

Este archivo también contiene el identificador del fabricante, el identificador de dispositivo y la descripción textual del dispositivo, es decir, el nombre con el que se identificará con el sistema operativo. En este caso se utilizaron valores temporales para estos datos, ya que la empresa deberá definir estos códigos si se lleva el producto al mercado.

5.3.2. Controlador en la PC

Todo el desarrollo del sistema y la documentación se ha realizado en la plataforma Linux, por ello, la programación del controlador se realizó en esta misma plataforma.

El controlador se realizó de una forma sencilla, basándose en una biblioteca de funciones de USB llamada `libusb`, que también puede utilizarse para compilar controladores para Windows y otras plataformas. Esta biblioteca provee funciones simples para abrir la comunicación con dispositivos USB y realizar transacciones de datos desde un programa que no esté integrado al sistema operativo. Esto permite realizar un programa muy sencillo para manejar un dispositivo propietario, sin necesitar de un controlador específico.

El controlador que se desarrolló lee la lista de dispositivos conectados y analiza uno a uno su identificador de fabricante. Cuando este coincide con el identificador temporal escogido como se menciona en la sección 5.3.1 empieza a solicitar lecturas de datos en las direcciones 0x82 y 0x86. Si la lectura se realiza con éxito, se despliega en pantalla el contenido del paquete.

Este programa podría permanecer en este estado sin fin, sin embargo, se limitó para fines prácticos, especialmente para confirmar la correcta conclusión del programa, luego de haber recibido los paquetes de información.

Capítulo 6

Análisis de Resultados

Los datos obtenidos por el sistema en las pruebas preliminares permiten analizar el contenido de los paquetes de información transmitidos, sin embargo, también se encontraron problemas de tensión en las líneas del bus SD.

6.1. Aislamiento del bus

Se probaron dos aisladores de similares características: el IL711-2 de NVE Electronics y el ADuM1400 de Analog Devices. El ADuM1400 no estaba disponible para la compra cuando fue realizado el pedido y por ello se optó por su alternativa. Sin embargo, el IL711 no dio el resultado requerido. Se realizó una medición de tensiones de las líneas de transmisión sin carga y con la carga de los dispositivos aisladores, para analizar la interferencia de estos con el bus. La medición se realizó con el bus en estado de espera, en el cual todas las líneas deben estar en un valor correspondiente a un 1 lógico.

En la tabla 6.1 se presentan los resultados obtenidos. Se puede observar que ambos dispositivos disminuyen considerablemente la tensión de las líneas, especialmente las de datos. Esto resulta problemático, porque se interfiere considerablemente en el funcionamiento del dispositivo. Esto provoca, especialmente con los dispositivos IL711, que la salida no sea una copia fiel de la entrada, sino que oscile inesperadamente.

Los aisladores digitales son alimentados tanto por el circuito del bus como por el circuito analizador. Estos utilizan la alimentación del bus como marco de referencia para realizar la comparación. Sin embargo, la conexión del dispositivo hace que baje un poco la tensión de alimentación del bus, causando inclusive que el sistema operativo no sea capaz de comunicarse con la tarjeta.

Tabla 6.1. Medición de tensión en el bus al conectar los circuitos aisladores.

Dispositivo aislador	VDD	CLK	CMD	D0	D1	D2	D3
ninguno	3.275	3.274	3.270	3.273	3.260	3.268	3.275
IL711	3.265	3.265	3.264	2.823	2.851	2.921	3.115
ADuM1400	3.271	3.268	3.266	3.051	3.105	3.250	3.243

Para evitar este problema se optó por alimentar el aislador con la tensión del analizador tanto del lado del analizador como del lado del bus. El resultado permitió disminuir la corriente de entrada de los aisladores y así permitir que la tensión en los canales de transmisión se mantuviera más alta. En la tabla 6.2 se muestran los resultados obtenidos, comparados con los obtenidos anteriormente. Dado que la tensión con que es alimentado el circuito analizador es la misma que la del bus de la tarjeta SD esto no presenta ningún problema de incompatibilidad de nivel.

Tabla 6.2. Medición de tensión en el bus al alimentar los circuitos aisladores ADuM1400 desde cada fuente.

Fuente	VDD	CLK	CMD	D0	D1	D2	D3
Bus	3.271	3.268	3.266	3.051	3.105	3.250	3.243
Analizador	3.275	3.271	3.272	3.273	3.215	3.264	3.245

Aún en estas condiciones se observaron inestabilidades que causaron problemas en la captura en el bus de datos. En la sección 6.2 se comenta los resultados de la lectura.

6.2. Lectura de datos

La lectura de los datos desde el computador se realizó a través de la biblioteca `libusb`, lo cual simplificó mucho la programación del controlador. El programa se hizo de forma muy sencilla, tal que mostrara en la pantalla el resultado de la lectura. También se puede redireccionar la salida de este programa a un archivo utilizando el operador > en la consola: `./driver > archivo.txt`.

Con este método se ha capturado alguna información. Sin embargo, la información mostrada incluye muchos caracteres aleatorios contenidos en el espacio posterior al paquete de información, debido a que se ha programado que se muestren los 512 caracteres de cada espacio de almacenamiento.

La lectura de estos datos se realizó utilizando una función de la biblioteca `libusb` que reporta el resultado de la operación con un número entero. Este valor corresponde a los datos útiles que se leyeron. Este es el caso de transmisiones de datos infreriores a 500 bytes que causan transmisiones USB que no alcanzan a completar el máximo espacio de almacenamiento. Las transmisiones de comandos y respuestas siempre son de este tipo, pues

miden 48 bits (6 bytes) y 136 bits (17 bytes) más 12 bytes de encabezados. Esto significa que los paquetes USB leídos por el programa para comandos y respuestas miden 18 y 30 bytes, respectivamente.

Es posible, entonces, filtrar las transmisiones haciendo uso de la respuesta numérica de la función, para que las mediciones sean más fáciles de interpretar por el programa que va a mostrar la información de forma ordenada.

```
     SOPFL -TIMESTAMP-              CRC|seqlen| EOP
EP    0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
0x82 58 00 98 a4 38 b5 d2 02 89 8a 00 d7 00 10 0a ea
0x82 54 00 98 a4 3c 88 12 00 00 09 00 d3 00 10 0a aa
0x82 58 00 98 b9 16 41 cc 02 89 8a 00 57 00 10 0a aa
0x82 54 00 98 b9 17 26 0c 00 00 0b 00 7f 00 10 0a ea
0x82 58 00 9e 65 6c 76 52 02 89 ea 00 e9 00 10 0a ea
0x82 54 00 9e 65 70 4a 12 00 00 09 00 d3 00 10 0a aa
0x82 58 00 9e 97 ad f4 4c 0b 89 fa 00 69 00 10 0a aa
0x82 54 00 9e 97 ae d9 0c 00 00 0b 00 7f 00 10 0a ea
0x82 58 00 b0 7a e7 f3 5a 02 89 fa 00 e9 00 10 0a aa
0x82 54 00 b0 7a eb c7 16 18 00 09 00 d3 00 10 0a aa
0x82 58 00 b0 ae 79 e2 4c 02 89 ea 00 69 00 10 0a aa
0x82 54 00 b0 ae 7a c7 0c 10 00 0b 00 7f 00 10 0a ea
0x82 58 00 b0 b2 a6 16 5a 02 8a ea 00 0b 00 10 0a aa
0x82 54 00 b0 b2 a9 e9 16 00 00 09 00 d3 00 10 0a aa
0x82 58 00 b0 cf 13 85 ec 02 8a ea 00 8b 00 10 0a aa
0x82 54 00 b0 cf 14 6a 0c 00 00 0b 00 7f 00 10 0a ea
```

Figura 6.1. Resultado de captura de comandos y respuestas filtrados.

Al aplicar este filtro se obtienen resultados como los mostrados en la figura 6.1, y que pueden ser validados confirmando que se cumple con la estructura de paquetes que se definió en la sección 5.2.2.

En el caso de las capturas de datos, se obtuvieron lecturas inestables debido a las inestabilidades en la conexión. Sin embargo, estos no se presentan en todos los casos. Una de las posibles fuentes de error es la longitud del cable de conexión entre el bus y los aisladores. Por restricciones de espacio, este se construyó con una longitud de aproximadamente 35 cm. La reducción de este cable puede significar una reducción de la inestabilidad.

Capítulo 7

Conclusiones y recomendaciones

7.1. Conclusiones

- El desarrollo de máquinas de estado con FPGAs simplifica tareas de gran complejidad que requerirían de grandes programas en un microcontrolador.
- El análisis del protocolo SD requiere de dos analizadores independientes que capturen los datos de los dos canales de comunicación simultáneamente.
- Las máquinas de estados modificadas permiten un funcionamiento más rápido del sistema, reduciendo el número de estados, pero aumentando de forma considerable la lógica del sistema.
- La interferencia de un dispositivo como el analizador en un bus de transmisión de datos puede causar la incapacidad de establecer una conexión efectiva entre los dispositivos.

7.2. Recomendaciones

- Para la construcción de prototipos usando dispositivos de lógica programable es conveniente contar con diversos tamaños de dispositivos si no se conoce el volúmen de la lógica que debe ser implementada.
- Para el desarrollo del producto se puede optar por un modelo de FPGA más pequeño, de la misma serie Spartan-3. Esto evita el desperdicio de recursos y favorece la reducción de los costos de producción del dispositivo.

- Es necesario realizar pruebas con diversos tipos y tecnologías de aisladores, para determinar el más conveniente para el dispositivo desarrollado.
- Se debe reducir al mínimo la longitud de los cables entre el bus y los aisladores.
- Incorporar todo el dispositivo en una tarjeta que incluya una extensión para el bus SD, de tal forma que las señales se puedan transmitir con fidelidad.

Bibliografía

[1] Analog Devices. Quad-Channel Digital Isolator – Data Sheet Rev. C. Massachusetts, 2005.

[2] Cypress Semiconductor Corporation. EZ-USB FX2LP USB Microcontroller – Data Sheet Rev. J. California, 2005.

[3] IEEE. About us, 2006. [en línea; accesado el 9 de febrero, 2006].

[4] MMCA Technical Committee. *The MultiMediaCard System Specification. Version 3.31.* MultiMedia Card Association, California, 2003.

[5] Multimedia Card Association. Multimedia card, 2006. [en línea; accesado el 9 de febrero, 2006].

[6] SD Association Technical Committee. *SD Specifications: Part E1. SDIO Specification. Version 1.10.* SD Card Association, California, 2004.

[7] SD Card Association Technical Comitee. *SD Specifications: Part 1. Physical Layer Specification. Version 1.10.* SD Card Association, California, 2004.

[8] Wikipedia. Backus-naur form — wikipedia, la enciclopedia libre, 2006. [en línea; accesado el 16 de junio, 2006].

[9] Wikipedia contributors. Wikipedia, the free encyclopedia, 2006. [en línea; accesado el 8 y 9 de febrero, 2006].

[10] Xilinx. Spartan-3 FPGA Family: Complete Data Sheet v 1.7, 2005.

[11] Xilinx. XC9500XL High-Performance CPLD Family Data Sheet v 2.0, 2005.

Apéndice A

Glosario y abreviaturas

B

bit Acrónimo de "binary digit", correspondiente a la mínima unidad de información. Corresponde a un dígito en el sistema binario, que puede tomar el valor de "0" o de "1". Comúnmente se utiliza para representar las velocidades de transmisión de información, en términos de "bits por segundo". Por lo general, se utiliza la letra "b" minúscula como su abreviatura, y por ende $^{b}/_{s}$, y sus múltiplos $^{kb}/_{s}$, $^{Mb}/_{s}$ para representar velocidades.[9].

Bluetooth Es una especificación industrial para redes de área personal (PAN) inalámbricas. Proveé una forma de interconectar e intercambiar información entre dispositivos como computadoras de mano, portátiles o de escritorio, teléfonos celulares, cámaras digitales, impresoras, auriculares, teclados y otros a través de una frecuencia de radio de corto alcance y bajo costo. Es también conocido como IEEE 802.15.1.[9].

BNF Backus-Naur Form. Es una metasintaxis para describir lenguajes formales. Se utiliza comúnmente para describir lenguajes de programación, sistemas de comando y protocolos de comunicación. Los elementos se describen entre signos <> y se describen de forma explícita o en términos de otros símbolos.[8].

byte Grupo de bits, generalmente 8, que representan una unidad de información. Un byte puede representar 256 valores distintos. Por lo general se utiliza la letra "B" mayúscula como su abreviatura, y por ende $^{B}/_{s}$, y sus múltiplos $^{kB}/_{s}$, $^{MB}/_{s}$ para representar velocidades de transmisión de datos.[9].

C

CCITT Comité consultatif international téléphonique et télégraphique. Actualmente llamado International Telecommunication Union - Telecommunication Standardization Sector (ITU-T), coordina estándares para la comunicación y telefonía internacional a manera de Recomendaciones. La ITU es una dependencia de la Organización de Naciones Unidas (ONU) y sus Recomendaciones son formalmente reconocidas.

CPLD Complex Programmable Logic Device. Dispositivo de lógica programable complejo. Dispositivo que contiene componentes programables para implementar expresiones y operadores lógicos.[11].

CRC Cyclic Redundancy Check. Comprobación de Redundacia Cíclica, es un procedimiento que se puede realizar a un conjunto de datos y que da como resultado un número relativamente pequeño que puede ser utilizado para confirmar la correcta recepción de información, lo cual puede evitar y corregir errores.[9].

E

EEPROM Electrically Erasable Read-Only Memory. Memoria de solo lectura borrable eléctricamente. Memoria no-volátil en la que cada localidad puede ser borrada eléctricamante de forma individual.

F

FPGA Field-Programmable Gate Array. Arreglo de compuertas programable en campo. Dispositivo que contiene componentes de lógica programable e interconexiones lógicas.[10].

G

GPS Global Positioning System. Es el primer sistema de navegación por satélite que se implementó. Usa una constelación de 24 satélites que transmiten señales de radio que permiten a los receptores de GPS determinar precisamente su localización (longitud, latitud y altitud) casi en cualquier parte del mundo. La precisión de la señal de GPS actualmente es de cerca de 5 m y ha venido mejorando constantemente desde su implementación. A través de técnicas de corrección de errores, la precisión puede ser mejorada hasta casi 1 cm en distancias cortas.[9].

I

I^2C Inter-Integrated Circuit Bus. Es un estándar para la comunicación entre circuitos integrados. Utiliza solamente dos líneas bidireccionales: reloj (SCL) y datos (SDA), ambas funcionando a 5 V y conectadas a través de resistencias a esta tensión. Permite la conexión de hasta 112 nodos en el mismo bus, a velocidades de 10 kb/s, 100 kb/s, 400 kb/s e inclusive 3.4 Mb/s.[9].

IEEE Institute of Electrical and Electronics Engineers, Inc. Es una organización sin fines de lucro para promover el avance de la tecnología en el área de la electrónica, electricidad y la computación.[3].

IEEE 802.11 Denota un conjunto de estándares de redes inalámbricas desarrolladas por el grupo de trabajo 11 del comité de estándares de redes locales de la IEEE. La familia 802.11 actualmente cuenta con seis técnicas de modulación que usan el mismo protocolo. Son muy populares en dispositivos como computadoras portátiles y de mano y otros dispositivos móviles.[9].

M

memoria flash Memoria no volátil de acceso aleatorio que puede ser borrada eléctricamante en bloques. Su costo es mucho menor que el de la EEPROM.

MMC Multimedia Card. Es un estándar de tarjetas de memoria flash. Salió al mercado en 1997, desarrollado por Siemens AG y SanDisk. Está basado en la memoria flash tipo NAND de Toshiba. Las dimensiones de las tarjetas MMC son 24 mm x 32 mm x 1.5 mm. Originalmente usaban una transferencia serial, pero nuevas versiones permiten transferencias de 4 y hasta 8 bits simultáneamente.[5, 9].

R

RAM Random Access Memory. Memoria de acceso aleatorio. Es un tipo de memoria volátil con tiempos de acceso muy cortos. Permite escoger cualquiera de sus celdas para ser leída en cualquier momento, en contraposición a las memorias de acceso secuencial, que requieren una lectura consecutiva de sus celdas.

S

SD Secure Digital. Es un estándar de tarjetas de memoria flash. Es usado en dispositivos portátiles, como cámaras y computadores de mano. Están basadas en el formato y dimensiones de las tarjetas MMC, pero son por lo general ligeramente más gruesas, hasta 2.1 mm. Hay diferentes velocidades, que

se denotan en múltiplos de 150 $^{kB}/_{s}$, como los lectores de discos compactos. Las velocidades varían desde 6x (900 $^{kB}/_{s}$) hasta 166x (25 $^{MB}/_{s}$). A diferencia de las MMC, algunas de las SD incorporan un interruptor para desabilitar la escritura de la tarjeta, y convertirla en memoria de solo lectura. Los dispositivos con ranuras para SD pueden usar también las tarjetas MMC, pero las tarjetas SD no pueden ser usadas en ranuras MMC. Hay varias variantes en tamaños de tarjetas, como la miniSD (20 mm x 21 mm x 1.4 mm) y la microSD (15 mm x 11 mm x 1 mm), que pueden usarse en una ranura SD normal usando un adaptador, ya que se conserva el mismo estándar de comunicación.[9].

SDIO Extensión del estándar SD para ser utilizado con dispositivos pequeños diseñados para el factor de forma de las tarjetas SD, como receptores de GPS, adaptadores Wi-Fi, Ethernet o Bluetooth, modems, cámaras digitales y otros.[9].

sistemas autocontenidos (embedded systems) Son sistemas electrónicos o computadoras de propósito específico en los que están completamente contenidas en el dispositivo que controlan. Estas computadoras están programadas para realizar unas pocas tareas específicas y ejecutan un programa más o menos invariable, generalmente conocido como firmware. El núcleo de estos sistemas es un microprocesador o un microcontrolador con su memoria de datos y programa incorporada en el sistema.

SPI Serial Peripheral Interface Bus (bus de interfaz de periféricos seriales). Es un estándar muy laxo para controlar casi cualquier dispositivo electrónico que acepte un flujo de bits serial con reloj. Puede recibir y enviar a la vez (es decir, opera en full-duplex), y requiere de poco espacio en un circuito integrado. Inclusive puede ser implementado con unos pocos pines de entrada y salida de un microcontrolador via software. Consta de una señal de reloj (CLK), una entrada de datos (SDI), una salida de datos (SDO), y una de selección de chip (CS) para cada circuito integrado que va a ser controlado.[9].

U

USB Universal Serial Bus. Es un estándar de bus serial para conectar dispositivos, especialmente a computadoras de todo tipo, pero no limitado a ellas. Puede trabajar a velocidades de 1.5 $^{Mb}/_{s}$, 12 $^{Mb}/_{s}$ y 480 $^{Mb}/_{s}$, e interconectar hasta 127 periféricos a un solo dispositivo anfitrión.[9].

V

VHDL Very High Speed Circuit Hardware Description Language. Lenguaje de descripción de hardware descrito por el estándar IEEE 1076. Utilizado en el desarrollo de sistemas de hardware sintetizado para su implementación en FPGAs, CPLDs y otros componentes de lógica programable.

W

Wi-Fi véase IEEE 802.11.

Apéndice B

Manual del usuario

Se presentan algunos detalles técnicos que deben tomarse en cuenta en el desarrollo del software que se utilice para el despliegue de la información capturada.

B.1. Lectura de la información

Este dispositivo requiere de una conexión USB 2.0, con capacidad para comunicación en modo *high-speed*.

La lectura debe realizarse periódicamente, haciendo uso de las instrucciones de lectura para dispositivos USB. La información está ubicada en los *endpoints* 2 y 6, que se identifican con los valores hexadecimales 0x82 y 0x86, respectivamente.

El tamaño de estos *endpoints* es 512 bytes, pero la cantidad de bytes leídos corresponde al tamaño del paquete enviado.

La estructura de los datos recibidos se ilustra en la figura B.1.

<protocolo>	:==	<sop-sync-0.5><flags-1.5><timestamp-4>
		<data><sequence-0.5><lenght-1.5><frag-0.5><eop-1.5>
<sop-sync-0.5>	:==	0x5, inicio del paquete
<flags-1.5>	:==	descriptor del tipo de información:
		Bit 11: Comando
		Bit 10: Respuesta
		Bit 9: Datos modo 1 bit
		Bit 8: Datos modo 4 bits
		Bit 5: Interrupción SDIO
		Bit 3: Respuesta Larga
		Bits 7,6,4,2-0: reservados
<timestamp-4>	:==	registro de tiempo de 4 bits
<data>	:==	hasta 500 bytes de datos capturados
<sequence-0.5>	:==	4 bits de número de paquete cuando hay que dividir un paquete de datos
<lenght-1.5>	:==	longitud del paquete transmitido, contando encabezados
<frag-0.5>	:==	descriptor de fragmentación del paquete:
		Bit 3: este paquete es un fragmento
		Bit 2: este paquete es el último fragmento
		Bits 1,0: reservados
<eop-2>	:==	0xAAA, final de paquete

Figura B.1. Protocolo de envío de información

B.2. Posibilidades de expansión

El dispositivo cuenta con un *endpoint* para recibir información, este corresponde al *endpoint* 8, identificado con el valor hexadecimal 0x08.

La escritura en este *endpoint* está reservada para la reconfiguración del dispositivo, para implementar la reprogramación de la memoria flash del FPGA y de la EEPROM del microcontrolador.

Estas características no han sido implementadas aun, pero se planea continuar el desarrollo del dispositivo para incluir estas características.

Apéndice C

Información del Proyecto

Información del estudiante			
Nombre:	Jorge Manuel Rivera Gutiérrez		
Cédula:	1-1109-0460	Carné ITCR:	9909117
Dirección:	Casa J-4, Los Colegios, Moravia		
	De la Farmacia La Guaria, 200 Oeste, 200 Norte, 25 Oeste		
Teléfono:	241-2102	Celular:	822-4785
e-mail:	jorgerivera@gmx.net		
	jorge.rivera@ridgerun.com		

Información del Proyecto	
Nombre:	Diseño e implementación de un analizador de bus Secure Digital
Área:	Electrónica Digital, Microcontroladores, Comunicación Digital

Información de la Empresa	
Nombre:	Cadenux, LLC.
Zona:	Guadalupe, Goicoechea, San José
Dirección:	De la esquina suroeste de los Tribunales, 650 Norte y 50 Este
Teléfono:	297-0204

Información del encargado en la empresa			
Nombre:	Marco Caamaño Castro		
Puesto que ocupa:	Gerente		
Departamento:	Investigación y desarrollo		
Profesión:	Ingeniero eléctrico	Grado académico:	Bachiller
Teléfono:	297-0204		
e-mail:	marco.caamano@ridgerun.com		

Apéndice D

Cadenux, LLC

Cadenux es una compañía dedicada al desarrollo de sistemas empotrados con Linux, en una amplia variedad de procesadores y plataformas (ARM7, ARM9, XSCALE, MIPS y otros). Sus paquetes de soporte y servicios de ingeniería reducen el tiempo a mercado y brindan soluciones optimizadas para las necesidades de sus clientes a nivel internacional. Mediante la inversión en investigación y en desarrollo y relación con la academia fomentan el desarrollo tecnológico del país.

El proyecto desarrollado está destinado a contribuir al desarrollo de controladores en la división de paquetes de soporte. El desarrollo posterior del proyecto será realizado por empleados de la compañía con el objetivo de utilizar el producto en las labores cotidianas de desarrollo de software y abrir la posibilidad de posicionarlo como un producto de mercado disponible para otras compañías que desarrollen software.

Anexo I

Datos técnicos de los dispositivos utilizados

CY7C68013A/CY7C68014A
CY7C68015A/CY7C68016A

EZ-USB FX2LP™ USB Microcontroller

1.0 Features (CY7C68013A/14A/15A/16A)

- USB 2.0–USB-IF high speed certified (TID # 40440111)
- Single-chip integrated USB 2.0 transceiver, smart SIE, and enhanced 8051 microprocessor
- Fit, form and function compatible with the FX2
  - Pin-compatible
  - Object-code-compatible
  - Functionally-compatible (FX2LP is a superset)
- Ultra Low power: I_{CC} no more than 85 mA in any mode
  - Ideal for bus and battery powered applications
- Software: 8051 code runs from:
  - Internal RAM, which is downloaded via USB
  - Internal RAM, which is loaded from EEPROM
  - External memory device (128 pin package)
- 16 KBytes of on-chip Code/Data RAM
- Four programmable BULK/INTERRUPT/ISOCHRONOUS endpoints
  - Buffering options: double, triple, and quad
- Additional programmable (BULK/INTERRUPT) 64-byte endpoint
- 8- or 16-bit external data interface
- Smart Media Standard ECC generation
- GPIF (General Programmable Interface)
  - Allows direct connection to most parallel interface
  - Programmable waveform descriptors and configuration registers to define waveforms
  - Supports multiple Ready (RDY) inputs and Control (CTL) outputs
- Integrated, industry-standard enhanced 8051
  - 48-MHz, 24-MHz, or 12-MHz CPU operation
  - Four clocks per instruction cycle
  - Two USARTS
  - Three counter/timers
  - Expanded interrupt system
  - Two data pointers
- 3.3V operation with 5V tolerant inputs
- Vectored USB interrupts and GPIF/FIFO interrupts
- Separate data buffers for the Set-up and Data portions of a CONTROL transfer
- Integrated I^2C controller, runs at 100 or 400 kHz
- Four integrated FIFOs
  - Integrated glue logic and FIFOs lower system cost
  - Automatic conversion to and from 16-bit buses
  - Master or slave operation
  - Uses external clock or asynchronous strobes
  - Easy interface to ASIC and DSP ICs
- Available in Commercial and Industrial temperature grade (all packages except VFBGA)

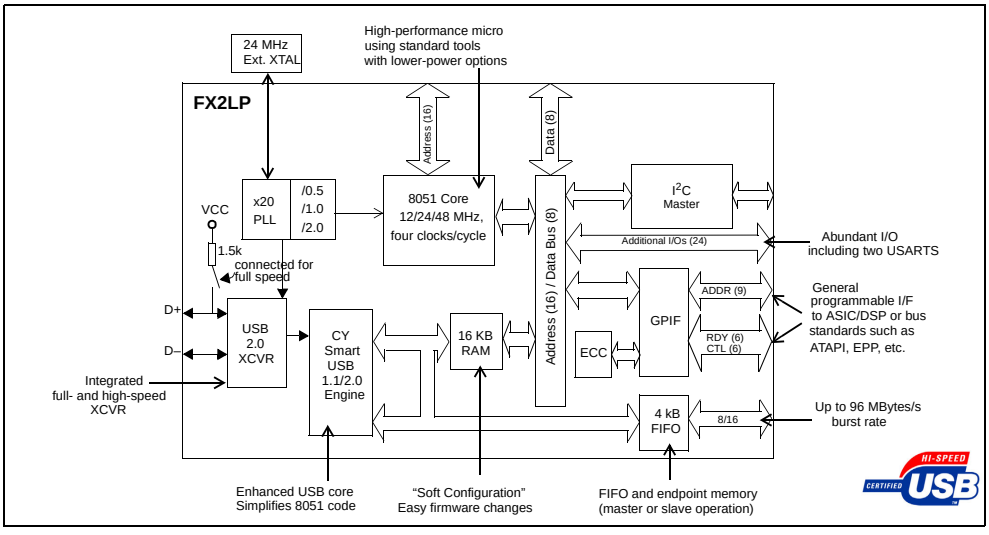


Figure 1-1. Block Diagram

Quad-Channel Digital Isolators

ADuM1400/ADuM1401/ADuM1402

FEATURES

- Low power operation
  - 5 V operation
    - 1.0 mA per channel max @ 0 Mbps to 2 Mbps
    - 3.5 mA per channel max @ 10 Mbps
    - 31 mA per channel max @ 90 Mbps
  - 3 V operation
    - 0.7 mA per channel max @ 0 Mbps to 2 Mbps
    - 2.1 mA per channel max @ 10 Mbps
    - 20 mA per channel max @ 90 Mbps
- Bidirectional communication
- 3 V/5 V level translation
- High temperature operation: 105°C
- High data rate: dc to 90 Mbps (NRZ)
- Precise timing characteristics
  - 2 ns max pulse width distortion
  - 2 ns max channel-to-channel matching
- High common-mode transient immunity: >25 kV/µs
- Output enable function
- 16-lead SOIC wide body package, Pb-free models available
- Safety and regulatory approvals
  - UL recognition: 2500 V rms for 1 minute per UL 1577
  - CSA component acceptance notice #5A
  - VDE certificate of conformity
    - DIN EN 60747-5-2 (VDE 0884 Part 2): 2003-01
    - DIN EN 60950 (VDE 0805): 2001-12; EN 60950: 2000
    - V_{IORM} = 560 V peak
  - TÜV approval: IEC/EN/UL/CSA 61010-1

APPLICATIONS

- General-purpose multichannel isolation
- SPI® interface/data converter isolation
- RS-232/RS-422/RS-485 transceiver
- Industrial field bus isolation

GENERAL DESCRIPTION

The ADuM140x[1] are 4-channel digital isolators based on Analog Devices' *i*Coupler® technology. Combining high speed CMOS and monolithic air core transformer technology, these isolation components provide outstanding performance characteristics superior to alternatives such as optocoupler devices.

By avoiding the use of LEDs and photodiodes, *i*Coupler devices remove the design difficulties commonly associated with optocouplers. The typical optocoupler concerns regarding uncertain current transfer ratios, nonlinear transfer functions, and temperature and lifetime effects are eliminated with the simple *i*Coupler digital interfaces and stable performance characteristics. The need for external drivers and other discrete components is eliminated with these *i*Coupler products. Furthermore, *i*Coupler devices consume one-tenth to one-sixth the power of optocouplers at comparable signal data rates.

The ADuM140x isolators provide four independent isolation channels in a variety of channel configurations and data rates (see the Ordering Guide). All models operate with the supply voltage on either side ranging from 2.7 V to 5.5 V, providing compatibility with lower voltage systems as well as enabling a voltage translation functionality across the isolation barrier. In addition, the ADuM140x provides low pulse width distortion (<2 ns for CRW grade) and tight channel-to-channel matching (<2 ns for CRW grade). Unlike other optocoupler alternatives, the ADuM140x isolators have a patented refresh feature that ensures dc correctness in the absence of input logic transitions and during power-up/power-down conditions.

[1] Protected by U.S. Patents 5,952,849 and 6,873,065. Other patents pending.

FUNCTIONAL BLOCK DIAGRAMS

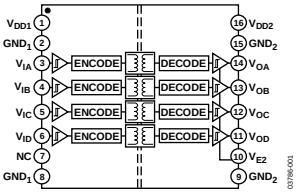


Figure 1. ADuM1400 Functional Block Diagram

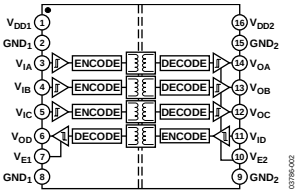


Figure 2. ADuM1401 Functional Block Diagram

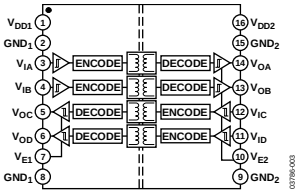


Figure 3. ADuM1402 Functional Block Diagram

One Technology Way, P.O. Box 9106, Norwood, MA 02062-9106, U.S.A.
Tel: 781.329.4700 www.analog.com
Fax: 781.461.3113

Spartan-3 FPGA Family: Introduction and Ordering Information

DS099-1 (v1.2) December 24, 2003
Advance Product Specification

Introduction

The Spartan™-3 family of Field-Programmable Gate Arrays is specifically designed to meet the needs of high volume, cost-sensitive consumer electronic applications. The eight-member family offers densities ranging from 50,000 to five million system gates, as shown in Table 1.

The Spartan-3 family builds on the success of the earlier Spartan-IIE family by increasing the amount of logic resources, the capacity of internal RAM, the total number of I/Os, and the overall level of performance as well as by improving clock management functions. Numerous enhancements derive from state-of-the-art Virtex™-II technology. These Spartan-3 enhancements, combined with advanced process technology, deliver more functionality and bandwidth per dollar than was previously possible, setting new standards in the programmable logic industry.

Because of their exceptionally low cost, Spartan-3 FPGAs are ideally suited to a wide range of consumer electronics applications, including broadband access, home networking, display/projection and digital television equipment.

The Spartan-3 family is a superior alternative to mask programmed ASICs. FPGAs avoid the high initial cost, the lengthy development cycles, and the inherent inflexibility of conventional ASICs. Also, FPGA programmability permits design upgrades in the field with no hardware replacement necessary, an impossibility with ASICs.

Features

- Revolutionary 90-nanometer process technology
- Very low cost, high-performance logic solution for high-volume, consumer-oriented applications
  - Densities as high as 74,880 logic cells
  - 326 MHz system clock rate
  - Three power rails: for core (1.2V), I/Os (1.2V to 3.3V), and auxiliary purposes (2.5V)
- SelectIO™ signaling
  - Up to 784 I/O pins
  - 622 Mb/s data transfer rate per I/O
  - Seventeen single-ended signal standards
  - Seven differential signal standards including LVDS
  - Termination by Digitally Controlled Impedance
  - Signal swing ranging from 1.14V to 3.45V
  - Double Data Rate (DDR) support
- Logic resources
  - Abundant logic cells with shift register capability
  - Wide multiplexers
  - Fast look-ahead carry logic
  - Dedicated 18 x 18 multipliers
  - JTAG logic compatible with IEEE 1149.1/1532 specifications
- SelectRAM™ hierarchical memory
  - Up to 1,872 Kbits of total block RAM
  - Up to 520 Kbits of total distributed RAM
- Digital Clock Manager (up to four DCMs)
  - Clock skew elimination
  - Frequency synthesis
  - High resolution phase shifting
- Eight global clock lines and abundant routing
- Fully supported by Xilinx ISE development system
  - Synthesis, mapping, placement and routing
- MicroBlaze processor, PCI, and other cores

Table 1: **Summary of Spartan-3 FPGA Attributes**

Device	System Gates	Logic Cells	CLB Array (One CLB = Four Slices)			Distributed RAM (bits[1])	Block RAM (bits[1])	Dedicated Multipliers	DCMs	Maximum User I/O	Maximum Differential I/O Pairs
			Rows	Columns	Total CLBs						
XC3S50	50K	1,728	16	12	192	12K	72K	4	2	124	56
XC3S200	200K	4,320	24	20	480	30K	216K	12	4	173	76
XC3S400	400K	8,064	32	28	896	56K	288K	16	4	264	116
XC3S1000	1M	17,280	48	40	1,920	120K	432K	24	4	391	175
XC3S1500	1.5M	29,952	64	52	3,328	208K	576K	32	4	487	221
XC3S2000	2M	46,080	80	64	5,120	320K	720K	40	4	565	270
XC3S4000	4M	62,208	96	72	6,912	432K	1,728K	96	4	712	312
XC3S5000	5M	74,880	104	80	8,320	520K	1,872K	104	4	784	344

Notes:
1. By convention, one Kb is equivalent to 1,024 bits.