UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA PARA MONITOREO Y CONTROL BASADO EN FPGA

T E S I S

QUE PARA OBTENER EL TÍTULO DE:

INGENIERO ELÉCTRICO ELECTRÓNICO

(Á R E A: E L E C T R Ó N I C A)

PRESENTA:

Héctor Alonso Valera Martínez

DIRECTOR DE TESIS: M. I. LAURO SANTIAGO CRUZ

MÉXICO, D.F. 2014

Agradecimientos

Primeramente a DIOS por permitirme estar aquí, y ahora. Gracias a la **UNAM** por darme la oportunidad de estudiar una carrera y, por supuesto, a la **Facultad de Ingeniería**, en donde pasé una etapa muy importante de mi vida.

A mi mamá y a mi papá, a mis hermanos y, en especial, a mi hermana ***PATY*** por todo el apoyo que me dio, ***GRACIAS***.

Al maestro Lauro y al Instituto de Ingeniería por brindarme un lugar para realizar este proyecto. A los profesores de la Facultad, que de ellos algo aprendí.

A Bob, Cuchillo, Fabio y Preciado por las anécdotas que pasamos en el laboratorio. A Alfredo, Bubu, Julio, Mario Oscar, Rosa, Adrián, Charly, Disdis, Isma, Iván, Job, Lalo, Mario, Nophal, Oscar, Tio, Belem, Cabañas, Cesar, Javier, Naye, Piochas, Sandra, Sócrates, Greta, JuanCa, Laura, Ricardo, Andrés, Ángel, Buky, Jipi, Richi, porque sin ustedes la Universidad no hubiera sido tan divertida como lo fue.

Agradezco a la DGAPA-UNAM la beca recibida.

Investigación realizada gracias al Programa de Apoyo a Proyectos de Investigación e Innovación Tecnológica (PAPIIT) de la UNAM, IG100914, Integración de un sistema para la obtención y monitoreo de datos de vehículos automotores, basado en los protocolos CAN y OBD-II.

Contenido

Índice de figuras

Capítulo I

Capítulo II

Capítulo III

Capítulo IV

Capítulo V

Índice de tablas

Capítulo II

Capítulo III

Capítulo IV

Capítulo V

Apéndice B

Prólogo

El rápido avance en la tecnología y las nuevas necesidades que surgen al ser humano hace que la vida útil de los circuitos electrónicos se reduzca rápidamente. Uno de los aspectos que actualmente se está manejando es incluir la mayor cantidad de funcionalidades en un dispositivo, ya sea móvil, como en el caso de los teléfonos celulares, ó fijo, como en el caso de las televisiones, computadoras, consolas de videojuegos, etcétera, lo que provoca el cambio ó la actualización constante, tanto de los circuitos electrónicos como del software que los controlan, para realizar nuevas tareas. Una solución para el cambio y la actualización de los circuitos electrónicos es utilizar los dispositivos lógicos programables, en específico, el más versátil de todos ellos: el arreglo de compuertas programables en campo ó FPGA (por sus siglas en inglés). Las principales características del FPGA es la capacidad de realizar cualquier función digital y la capacidad de reconfigurarse, es decir, cambiar y/o mejorar su función.

El presente trabajo tiene por objetivo general introducir al estudio del FPGA abarcando dos aspectos fundamentales que, en conjunto, concluyen un sistema de monitoreo y control. El primero es la manera de configurar el FPGA, ya que existen varias maneras para realizarlo, entre las que destacan los lenguajes de descripción de hardware (HDL por sus siglas en inglés) y el lenguaje de programación C. El segundo es el circuito impreso que contiene al FPGA, el cual posee características particulares entre las que destacan el número de capas de cobre y el orden de las mismas.

Tomando en cuenta los dos aspectos mencionados, el presente trabajo se divide en seis capítulos que a continuación se describen:

- Capítulo I. En este capítulo se muestra un breve resumen de las características más relevantes de los dispositivos lógicos programables, entre ellos el FPGA.
- Capítulo II. Este capítulo está dedicado a los conceptos teóricos básicos para el entendimiento y el desarrollo del trabajo. Se abordan características más detalladas del FPGA, así como también comunicaciones serie, lenguajes de descripción de hardware, sistemas embebidos, protocolos de configuración para el FPGA, entre otros.
- Capítulo III. Aquí se presenta a detalle las características y el diseño del sistema de monitoreo y control. Se presentan también los objetivos que se pretenden alcanzar con la realización del presente trabajo y la plataforma seleccionada para alcanzarlos. Se muestran las dos formas en que se abordó el diseño del sistema de monitoreo y control con todo lo que implica cada una de ellas para cumplir con los objetivos planteados. Las formas utilizadas fueron: lenguajes de descripción de hardware y lenguaje de programación C. Por último, se muestra el sistema de monitoreo y control obtenido con la forma de diseño seleccionada.
- Capítulo IV. Dedicado al diseño y ensamblado de los circuitos impresos, tanto de los componentes que lo integran como del sistema en conjunto. Se muestra a detalle los

problemas y soluciones que se encontraron al realizar los circuitos impresos, así como también los problemas y soluciones que se encontraron al realizar algunas interfaces del sistema.

- Capítulo V. En este capítulo se presentan las pruebas realizadas tanto a la plataforma seleccionada como a la plataforma desarrollada. Se muestra la manera de utilizar la plataforma desarrollada y las características funcionales de la misma.
- Capítulo VI. Este capítulo final muestra los resultados obtenidos, las conclusiones generadas y algunos comentarios finales del trabajo realizado.

Después de los seis capítulos se presenta la bibliografía y las referencias consultadas para el desarrollo del presente trabajo.

Se concluye con los apéndices que complementan y ayudan al entendimiento del presente trabajo. En el primer apéndice se muestra un glosario de términos para una consulta rápida de ellos, en el segundo apéndice se enlistan las terminales del FPGA utilizado y en el último apéndice se muestra el software desarrollado para el sistema de monitoreo y control.

Capítulo I
Introducción

En este capítulo se presenta una breve introducción a los dispositivos llamados arreglo de compuertas programables en campo (FPGA: *Field Programmable Gate Array*), así como una reseña sobre los dispositivos lógicos programables.

1.1. Introducción

Un FPGA es un circuito integrado destinado a la implementación de hardware personalizado, con la capacidad de ser reconfigurado una infinidad de veces. Reconfigurar un FPGA significa cambiar su funcionamiento para soportar una nueva aplicación, es decir, tener algunas piezas nuevas de hardware colocadas dentro del circuito FPGA que cumplan con la funcionalidad de la aplicación nueva. Los FPGA hacen posible tener un diseño personalizado de alta densidad en un solo circuito electrónico, teniendo la posibilidad de cambiar este diseño cuando sea necesario, incluso cuando la aplicación se esté ejecutando. El término *Field Programmable* se refiere a la propiedad de poder cambiar el funcionamiento del dispositivo en el campo. La flexibilidad de tener un hardware personalizado y cambiable es el factor que ha determinado la popularidad de los dispositivos FPGA en una amplia gama de campos de aplicación.

En la última década se han detectado cada vez mas áreas de uso para los dispositivos FPGA. Algunos ejemplos de las áreas de uso incluyen: el remplazo de diseños viejos de múltiples circuitos en un diseño moderno de un solo circuito, el procesamiento digital de señales e imágenes, aplicaciones multimedia, comunicaciones de alta velocidad, equipos de redes como *routers* y conmutadores, implementación de protocolos para periféricos así como la implementación de co-procesadores y de micro-controladores. Un área reciente de aplicación es en el cómputo reconfigurable, la cual consiste en reconfigurar múltiples veces el dispositivo FPGA durante su operación normal y así realizar diferentes tareas de cómputo en diferentes lapsos de tiempo.

Existen gran variedad de dispositivos FPGAs, en cuanto a tamaño y características tanto internas como externas. Lo que tiene en común todos estos dispositivos, es que están compuestos por bloques de lógica programable. Estos bloques, que normalmente contienen algunos registros y algunos elementos lógicos configurables, son colocados juntos en una red, utilizando conexiones programables. El número de componentes lógicos en los más recientes dispositivos FPGA han permitido la implementación de sistemas completos en un solo circuito integrado.

1.2. Dispositivos Lógicos Programables

Los dispositivos lógicos programables (PLD: *Programmable Logic Device*) son circuitos de uso general para implementar circuitos lógicos. Se introdujeron por primera vez al mercado a principios de 1970. Los PLDs contienen compuertas lógicas e interruptores programables, estos últimos permiten que las compuertas lógicas, en el interior del PLD, se conecten juntas para implementar el circuito lógico necesario. Entre los PLDs que existen comercialmente se encuentran los siguientes: arreglo lógico programable, lógica de arreglo programable, dispositivo lógico programable complejo y arreglo de compuertas programables en campo.

1.2.1. Arreglo lógico programable

El primer PLD que se desarrolló fue el arreglo lógico programable (PLA: P*rogrammable Logic Array*). Con base en la idea de que cualquier función lógica se puede realizar en forma de suma de productos, un PLA está constituido por un juego de compuertas AND que alimentan un conjunto de compuertas OR. Las entradas del PLA pasan por un grupo de *buffers* (que proporcionan el valor verdadero así como el valor negado de cada entrada) hacia un bloque de circuito llamado plano AND. El plano AND genera términos producto, cada uno de los cuales puede configurarse para implementar cualquier función AND de las entradas. Los términos producto sirven como entradas a un plano OR el cual produce las salidas. Cada salida puede configurarse para realizar cualquier suma de productos realizados en el plano AND y por ende cualquier función de suma de productos de las entradas al PLA. La única restricción en cuanto a las funciones que se pueden implementar es el tamaño del plano AND. Los parámetros típicos de un PLA comercial son de 16 entradas, 32 términos productos y 8 salidas.

En la figura 1.1 se muestra el diagrama de conexiones de un PLA sencillo, en donde se observan las conexiones (marcadas con una X) entre las entradas del PLA y los *buffers*, de los *buffers* al plano AND y del plano AND al plano OR.

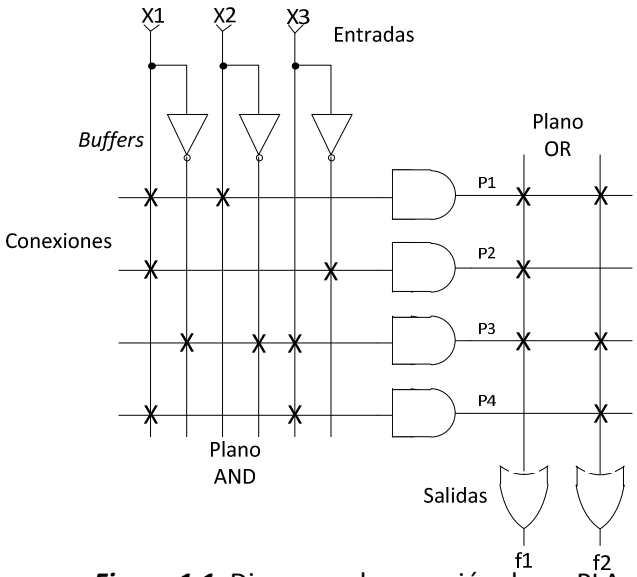


Figura 1.1. Diagrama de conexión de un PLA.

Las conexiones en los PLA se realizaban mediante interruptores programables que, generalmente, eran implementados con tecnología anti-fusible y se programaban por medio de voltajes altos para realizar las conexiones necesarias.

El PLA es eficiente en términos del área necesaria para implementarlo en un circuito integrado, por ello, con frecuencia se incluye un PLA como parte de circuitos integrados más grandes como en los micro-procesadores. En este caso se crea un PLA de modo que las conexiones a las compuertas AND y OR son fijas, es decir, no programables.

1.2.2. Lógica de arreglo programable

Las desventajas de los interruptores programables de los dispositivos PLA, tales como la fabricación incorrecta y la velocidad en los circuitos implementados, llevaron al desarrollo de un dispositivo similar en el que el plano AND es programable pero el plano OR es fijo. Tal dispositivo se conoce como dispositivos de lógica de arreglo programable (PAL: *Programmable Array Logic*). Debido a que los PAL son más simples de fabricar, son menos costosos que los PLA y tienen un mejor rendimiento. Para compensar su menor flexibilidad debido a que el plano OR es fijo, se fabrican en diversos tamaños, con varias entradas y salidas y diferentes entradas a las compuertas OR. Una limitación en los PAL es que un solo término producto no puede ser compartido entre dos términos suma. Si dos sumas contienen un término producto común, ese producto debe ser generado dos veces. En la figura 1.2 se muestra un ejemplo de un PAL donde se observa que las compuertas OR se encuentran fijas a los términos producto del plano AND.

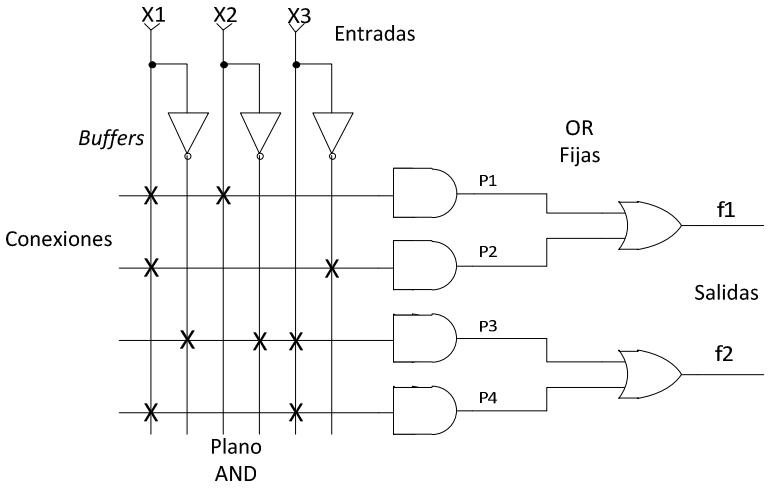


Figura 1.2. Diagrama de conexión de un PAL.

Macrocelda

En muchas PAL se agregan circuitos adicionales a la salida de cada compuerta OR para brindar mayor flexibilidad. Es usual emplear el término "macrocelda" ("macrocell") para referirse a la compuerta OR combinada con los circuitos adicionales. Se puede definir a una macrocelda como los circuitos lógicos asociados con una terminal de salida que contiene *flip-flops* (elemento básico

para el almacenamiento de datos digitales) y cierto número de opciones programables. Esto minimiza el número de dispositivos necesarios en un diseño debido a que las salidas de un solo dispositivo puede ser configurado en diferentes formas. Las opciones típicas de configuración incluyen la habilidad de utilizar o evitar el *flip-flop*; selección del modo de operación del *flip-flop* (D, T, SR, o JK) y selección de la salida del *flip-flop* (salida verdadera Q o salida complementada $\bar{Q}$).

En la figura 1.3 se muestra un ejemplo de una macrocelda básica, donde se observar la conexión entre la compuerta OR y el *flip-flop*. Un multiplexor 2 a 1 selecciona como salida de la PAL la salida de la compuerta OR o la del *flip-flop*. La línea de selección de dicho multiplexor puede ser programada para que tenga un valor lógico de "0" o "1". Existe también una compuerta *buffer* tres estados (con valores posibles de "1", "0" y "HIGH Z" ó alta impedancia) que se conecta entre el multiplexor y la terminal de salida del PAL. Por último, en la misma figura se muestra la realimentación que existe entre la salida del multiplexor al plano AND. Esta realimentación permite que la función lógica producida por el multiplexor se use internamente en la PAL, lo que posibilita la implementación de circuitos que tienen múltiples estados ó niveles.

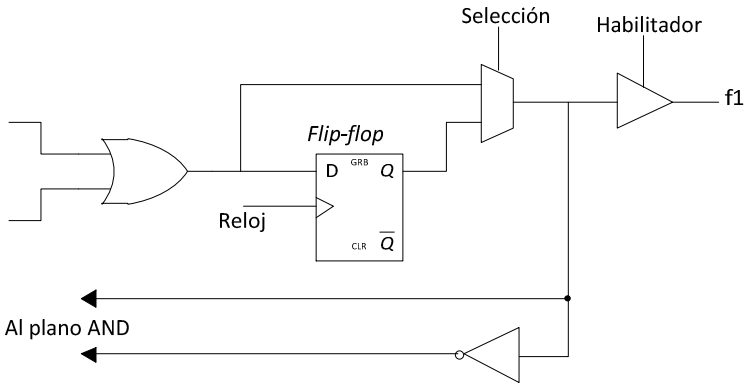


Figura 1.3. Macrocelda de un PAL.

1.2.3. Dispositivo lógico programable complejo

Un dispositivo lógico programable complejo (CPLD: *Complex Programmable Logic Device*), comprende múltiples bloques en un solo circuito integrado, con recursos de cableado interno para conectarlos. Cada bloque es similar a un PLA o a un PAL, por comodidad se referirán a estos bloques simplemente como "Bloque PAL".

En la figura 1.4 se muestra la estructura de un CPLD. Se incluyen cuatro bloques PAL que se conectan a los recursos de interconexión. Cada bloque PAL también está conectado a un sub-circuito llamado bloque de entrada salida, que a su vez está conectado a las terminales de entrada y salida del circuito integrado.

En la figura 1.5 se muestra un ejemplo de los recursos de cableado y las interconexiones a un bloque PAL de un CPLD. El bloque PAL de la figura 1.5 incluye tres macroceldas, los CPLD reales tiene alrededor de 16 macroceldas, cada una de las cuales consta de una compuerta OR de cuatro entradas, los CPLDS comerciales tienen entre 5 y 20 entradas. La salida de la compuerta OR se

conecta a una compuerta XOR (OR exclusiva), para esta compuerta, si ambas entradas son "1" produce una salida "0". Como se muestra en la figura 1.5, una entrada a la compuerta XOR se puede programar para que tenga un valor lógico de "1" ó "0". Si el valor de entrada es "1" la compuerta XOR complementa, es decir, invierte la salida de la compuerta OR. En cambio, si la entrada a la compuerta XOR es "0" la compuerta XOR no tiene efecto alguno en la salida.

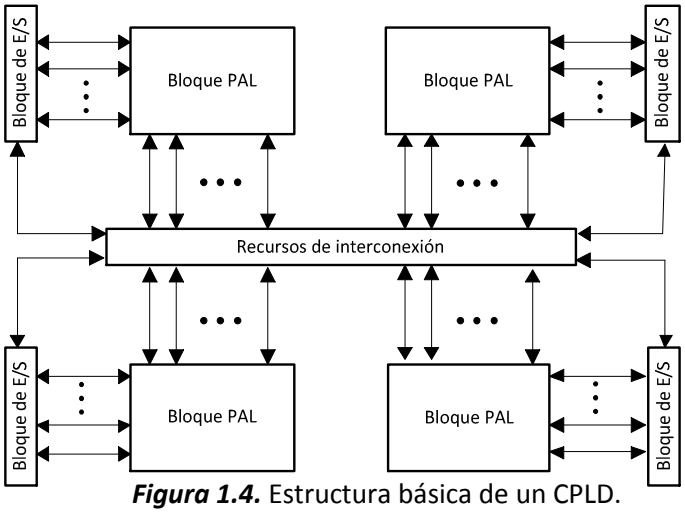


Figura 1.4. Estructura básica de un CPLD.

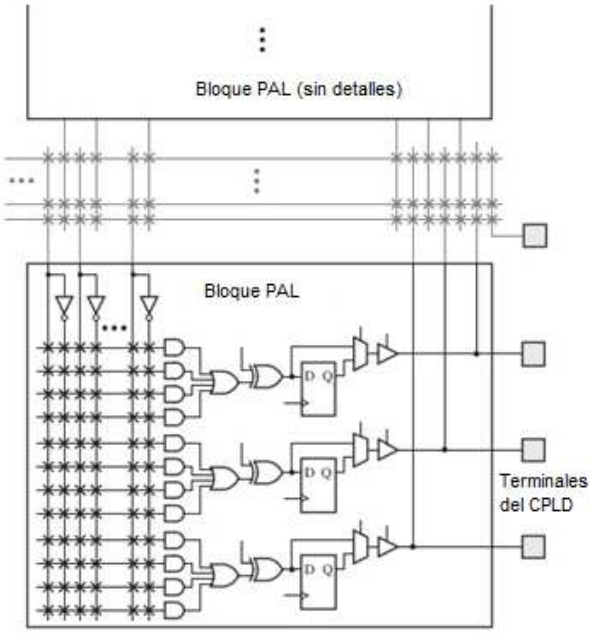


Figura 1.5. Recursos de interconexión de un CPLD.

La macrocelda de la figura 1.5 también incluye un *flip-flop*, un multiplexor y un *buffer* tres estados. El *flip-flop* se utiliza para almacenar el valor de salida de la compuerta OR, mientras que el *buffer* tres-estados se conecta a una terminal del CPLD. El *buffer* actúa como interruptor que permite que cada terminal se use como salida o como entrada. Para usar una terminal como salida, el correspondiente buffer se activa y se comporta como un interruptor que se enciende. Si la

terminal se utiliza como entrada, el *buffer* se deshabilita y se comporta como un interruptor que se apaga. En este caso una fuente externa puede dirigir una señal hacia la terminal que pude conectarse a otra macrocelda por medio de cables de interconexión. Cabe advertir que cuando una terminal se utiliza como entrada la macrocelda asociada con esa terminal no puede usarse y por lo tanto se desperdicia. Algunos CPLD incluyen conexiones adicionales entre macroceldas y el cableado que evita desperdiciarlas en tales situaciones.

Los recursos de interconexión contienen interruptores programables que se usan para conectar los bloques PAL. El número de interruptores se elige a fin de ofrecer flexibilidad suficiente para circuitos típicos sin desperdiciar muchos interruptores.

La programación de un CPLD se realiza mediante la transferencia de información desde un entorno de desarrollo integrado (IDE: *Integrated Development Environment*) hacia el CPLD. El método de programación para los CPLD es por medio del estándar IEEE 1149.1, conocido como grupo de acción de prueba conjunta (JTAG: *Joint Test Action Group*). En capítulos posteriores se da una descripción de este estándar. Una vez que el CPLD se programa retiene permanentemente la configuración aun cuando la fuente de poder del circuito integrado se apague (programación no volátil).

1.2.4. Arreglo de compuertas programables en campo

Un FPGA es un dispositivo lógico programable que soporta la implementación de circuitos lógicos hasta cierto punto grandes. Los FPGA son muy diferentes a los PLA, PAL y CPLD, ya que no contienen planos AND ni OR. En vez de esto, ofrecen bloques lógicos para la implementación de las funciones requeridas.

La estructura general de un FPGA se ilustra en la figura 1.6. Contiene principalmente tres tipos de recursos: bloques lógicos, bloques de entrada salida y recursos de interconexión. Los bloques lógicos están dispuestos en un arreglo bidimensional, en tanto que los recursos de interconexión están organizados como canales de enrutamiento horizontales y verticales, entre filas y columnas de bloques lógicos. Los bloques de entrada salida se encuentran alrededor del dispositivo, con el propósito de controlar el flujo de datos tanto a la entrada como a la salida del propio dispositivo, así como de los bloques lógicos. Los recursos de interconexión contienen alambres e interruptores programables que permiten que los bloques lógicos se interconecten de muchas formas. Existen también conexiones programables entre los bloques de entrada salida y los recursos de interconexión. Los FPGAs sirven para implementar circuitos lógicos con un tamaño de más de un millón de compuertas equivalentes.

Cada bloque lógico en un FPGA tiene un pequeño número de entradas y salidas. El bloque lógico más utilizado comercialmente es el basado en tablas de consulta (LUT: *Lookup Table*). Las LUT contienen celdas de almacenamiento que sirven para implementar una pequeña función lógica. Cada celda puede contener un solo valor lógico: “0” ó “1”. El valor almacenado se utiliza como salida de la celda de almacenamiento. Pueden crearse LUTs de varios tamaños en los que el tamaño se define mediante el número de entradas.

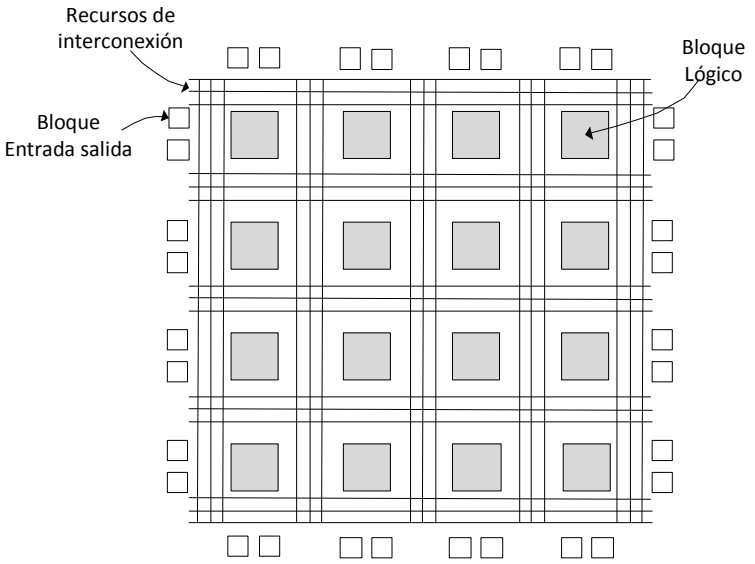
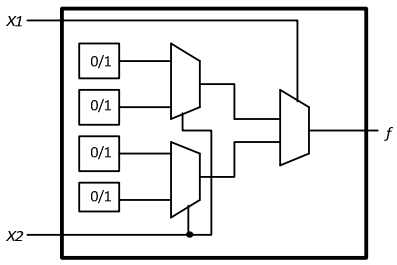


Figura 1.6. Estructura general de un FPGA.

En el inciso *a* de la figura 1.7 se muestra una estructura de una LUT de 2 entradas y una salida, mientras que en el inciso *b* muestra una función f_1, la cual se quiere implementar en dicha LUT.



a)

x1	*x2*	*f1*
0	0	1
0	1	0
1	0	0
1	1	1

b)

Figura 1.7. Estructura y función a implementar en una LUT.

Debido a que una función lógica de dos variables tiene cuatro filas (inciso *b* de la figura 1.7), la LUT debe tener cuatro celdas para almacenar el valor de la función que se desea implementar. Las variables de entrada se usan como entradas de selección a los multiplexores, los cuales, según el valor de entrada, eligen el contenido de una de las cuatro celdas de almacenamiento como salida de la LUT. La figura 1.8 muestra la implementación de la función f_1 en una LUT de 2 entradas y una salida.

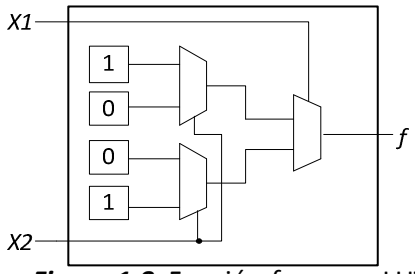


Figura 1.8. Función f_1 en una LUT.

En los FPGA comerciales, las LUTs tienen entre cuatro y cinco entradas, por lo tanto, requieren de un total de 16 y 32 celdas de almacenamiento para la implementación de funciones de dichas entradas. Es importante señalar que la mayoría de los FPGAs tienen LUTs volátiles, es decir, que pierden el contenido que almacenan siempre que la fuente de energía que alimenta al FPGA se apague. Existen FPGA con tecnología no volátil, los cuales se utilizan para diseños especiales, en donde no se requiere la reprogramación del FPGA.

Además de las LUTs, los FPGAs cuentan en los bloques lógicos con circuitos adicionales, como *flip-flops*, multiplexores, compuertas, entre otros circuitos. En la figura 1.9 se muestra un bosquejo de estos componentes.

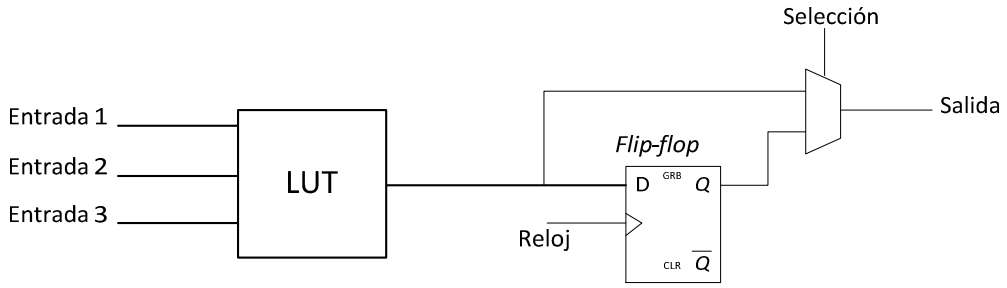


Figura 1.9. *Flip-flop* y multiplexor en un bloque lógico.

Para realizar un circuito lógico en un FPGA, cada función lógica debe ser lo suficientemente pequeña para encajar en un solo bloque lógico. Cuando se implementa un circuito, los bloques lógicos y los canales de enrutamiento se programan para cumplir las funciones necesarias del circuito diseñado.

Compuertas equivalentes

Una forma de cuantificar el "tamaño" de un circuito es suponer que se construirá usando compuertas lógicas simples y luego estimar cuántas de ellas se necesitan. Una medida que suele usarse es el número total de compuertas NAND de dos entradas que se necesitarían para construir el circuito, esta medida se llama número de compuertas equivalente y se utiliza regularmente para referirse a la capacidad de los PLDs.

Como dato histórico, a Ross Freeman (1948 – 1989) co-fundador de la empresa Xilinx, se le da el crédito por la invención del FPGA. Su FPGA [Fre89] incluía elementos lógicos programables y una estructura de interconexiones programable. Se programaba con tecnología SRAM (*Static Random Acces Memory*) y no con anti-fusible. Esto permitía que se fabricara en procesos estándares de escala de integración muy grande (VLSI: *Very Large Scale Integration*) teniendo menos costo y proporcionando más opciones de fabricación. Esta tecnología permitía que el FPGA pudiera ser reprogramado mientras "estaba" en el circuito.

En el siguiente capítulo se dan a conocer los conceptos teóricos básicos para el desarrollo y compresión del presente trabajo de tesis.

Capítulo II
Antecedentes

En este capítulo se presentan los conceptos teóricos básicos para la realización del proyecto. Se describirá la arquitectura interna del FPGA, así como la forma de configurarlo. También se abordarán los conceptos relacionados con las interfaces de comunicación serie y algunos componentes de propósito general.

2.1. Arquitectura de un FPGA

En este capítulo se describirán, además de los IOB y CLB, algunos recursos adicionales que se presentan frecuentemente en los FPGAs. Se describirán dichos recursos en los dispositivos fabricados por la empresa Xilinx de la familia SPARTAN 3E.

2.1.1. Bloques lógicos configurables

Los bloques lógicos configurables (CLB: *Configurable Logic Block*) ó elementos lógicos (LE*: Logic Element*) son los componentes principales de un FPGA; están organizados en una matriz regular de filas y columnas, como se muestra en la figura 2.1. En los FPGA de Xilinx, familia SPARTAN 3E, cada CLB contiene cuatro partes ó *slices* que están agrupados en pares y cada par está organizado en columnas. El par izquierdo soporta funciones lógicas y de memoria y es llamado SLICEM. El *slice* derecho sólo soporta funciones lógicas y se le llama SLICEL, como se muestra en la figura 2.2. Los componentes adicionales de un *slice* son: multiplexores, lógica de acarreo y compuertas de aritmética, los cuales se suman a la capacidad del *slice*. Cada *slice* contiene dos LUTs y dos elementos de almacenamiento (registros) que pueden ser utilizados como *flip-flop* o como *latch*. Cuatro de las ocho LUTs de un CLB se pueden utilizar como memoria RAM de 16x1 ó como registro de corrimiento de 16 bits. Alrededor de cada LUT existen conexiones para las señales que van desde y hacia ésta. La combinación de una LUT y un elemento de almacenamiento se conocen como celda lógica. En la figura 2.3 se muestran de forma esquemática los recursos de cada *slice*.

Durante el proceso de configuración del FPGA, se escribe en la memoria de la LUT la función requerida y la lógica alrededor de ella se configura para conectar la señal correctamente a los demás bloques necesarios.

2.1.2. Bloques de entrada salida

Los bloques de entrada/salida (IOB: *Input Output Block*) tienen la función de interconectar señales desde la lógica interna del FPGA hacia las terminales de salida del empaquetado del mismo FPGA y

viceversa. Hay exclusivamente un IOB para cada terminal del empaquetado. Cada IOB tiene su propia memoria de configuración, la cual almacena el estándar del nivel de voltaje, LVTTL, LVCMOS, SSTL, LVDS, etcétera, con el cual debe cumplir la terminal y la dirección de la comunicación de la misma (bidireccional o mono direccional).

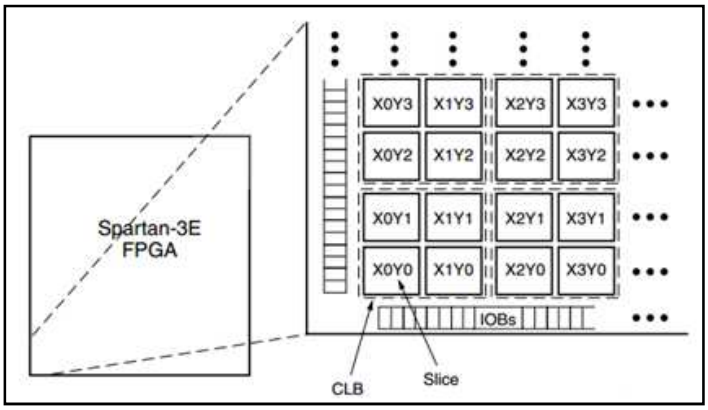


Figura 2.1. Distribución de los CLB.

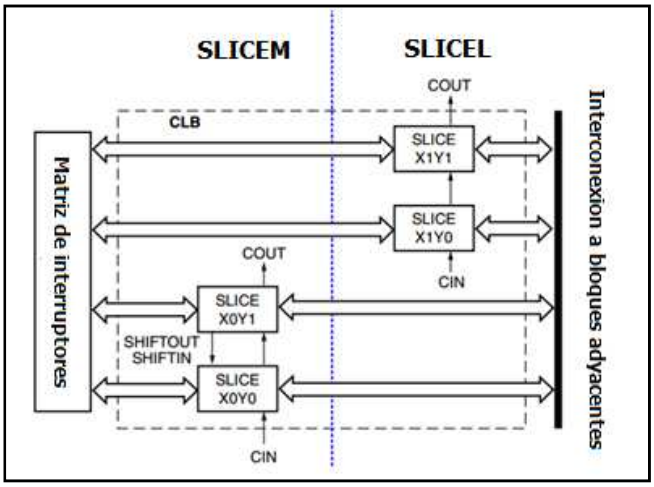


Figura 2.2. *Slices* en un CLB.

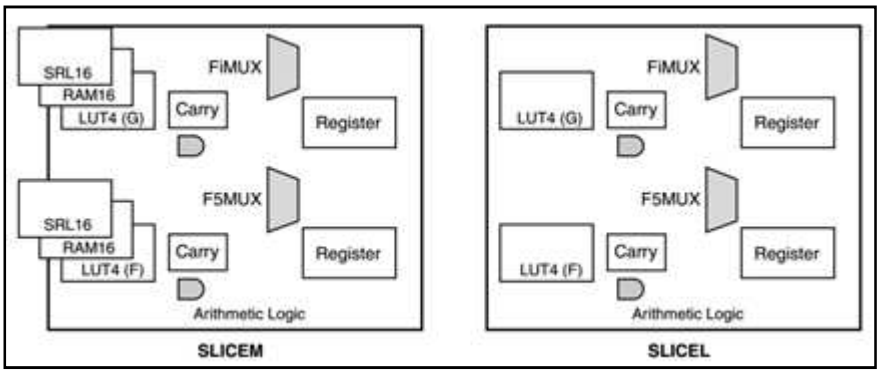


Figura 2.3. Recursos de un *slice* SLICEM y de un SLICEL.

El mayor reto en el diseño de la arquitectura de los IOB es la gran variedad en los estándares de voltaje de entrada y salida que pueden manejar. Cada estándar requiere umbrales de voltajes diferentes, tanto de entrada como de salida. Para cumplir con estas diferencias se requieren diferentes voltajes de alimentación, así como voltajes de referencia, si el estándar así lo demanda. Otros estándares pueden necesitar diodos, que permiten cierta tolerancia en los valores altos y bajos de voltaje. Muchos de los estándares se basan en señales diferenciales para mejorar la inmunidad al ruido y permitir el incremento de la velocidad en la transmisión de datos, estos factores también se deben tomar en cuenta para el diseño de los IOB.

La utilidad del FPGA depende de la flexibilidad para manejar diferentes estándares. Una vez que son seleccionados, es necesario determinar qué terminales soportarán dichos estándares. Los FPGA recientes han adoptado un esquema de entradas/salidas en bancos. Cada banco comparte voltajes de alimentación y voltajes de referencia. Un solo banco no puede manejar todos los estándares simultáneamente pero diferentes bancos pueden tener diferentes alimentaciones para manejar los diferentes estándares.

Los IOB en la familia SPARTAN 3E tienen tres rutas principales: ruta de salida, ruta de entrada y ruta tres estados. Cada ruta tiene sus elementos de almacenamiento que pueden actuar como registros o *latches*.

La ruta de entrada se ilustra en la figura 2.4, lleva el dato desde la terminal del empaquetado del FPGA, a través de un elemento de retardo opcional, a la línea I. Después hay rutas alternativas a través de un par de elementos de almacenamiento para las líneas IQ1 e IQ2. Las salidas de la ruta de entrada del IOB: I, IQ1 e IQ2 conducen a la lógica interna del FPGA.

La ruta de salida ilustrada en la figura 2.5, inicia con las líneas O1 y O2, lleva los datos desde la lógica interna del FPGA, a través de un multiplexor, a un controlador tres estados de la terminal del IOB. El multiplexor proporciona la opción de insertar un par de elementos de almacenamiento. También se observan protecciones para descarga electrostática (diodos) y resistencias de *PULL UP* y de *PULL DOWN*.

La ruta tres estados, figura 2.6, determina cuándo el controlador de salida (mostrado en la figura 2.5 como *Programmable Output Driver*) está en alta impedancia. Las líneas T1 y T2 llevan los datos desde la lógica interna del FPGA a través de un multiplexor hacia el controlador de salida. El multiplexor proporciona la opción de insertar un par de elementos de almacenamiento.

Los IOB en la SPRTAN 3E están organizados en 4 bancos, cada banco mantiene separado su propio voltaje de alimentación y su voltaje de referencia para los diferentes estándares soportados. La distribución de los bancos de entradas/salidas se muestra en la figura 2.7. Los estándares soportados por la familia SPARTAN 3E son: LVCMOS, LVTTL, HSTL y SSTL; los diferenciales que soporta son: LVDS, RSDS, mini-LVDS, HSTL y SSTL.

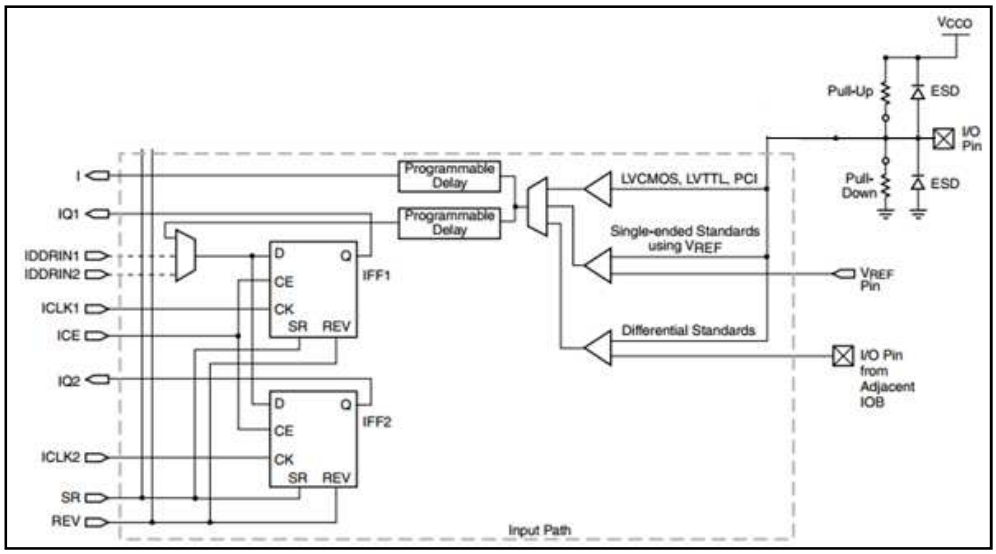


Figura 2.4. Ruta de entrada del IOB.

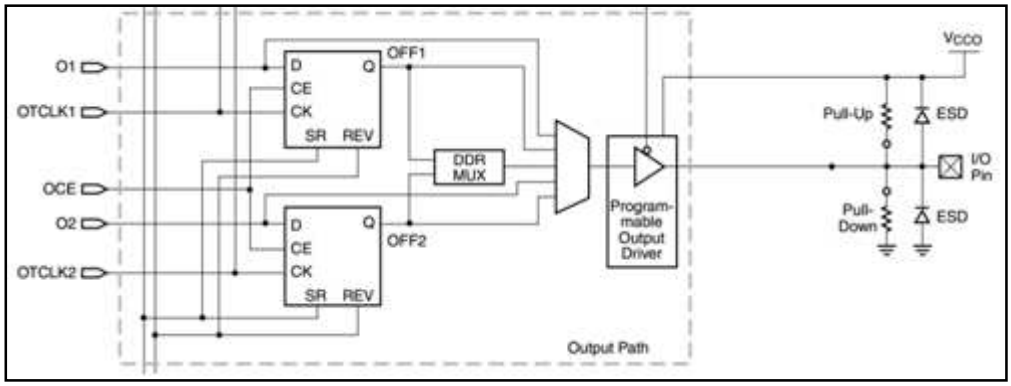


Figura 2.5. Ruta de salida del IOB.

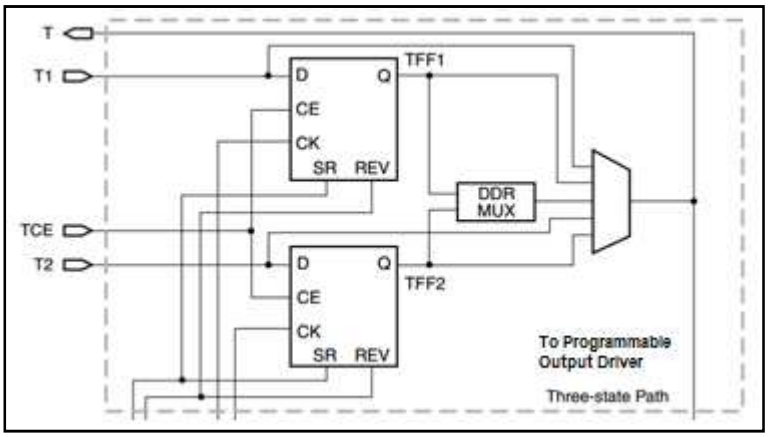


Figura 2.6. Ruta tres estados del IOB.

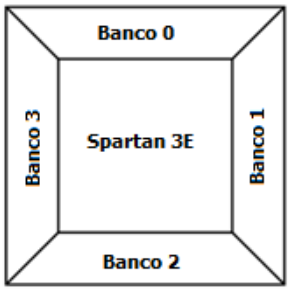


Figura 2.7. Distribución de bancos de IOB.

2.1.3. Recursos de interconexión

Los recursos de interconexión dentro del FPGA permiten conexiones arbitrarias entre los CLB, los IOB, las mismas conexiones y los recursos adicionales que contenga el FPGA. Los recursos de interconexión consisten en alambres e interruptores programables y definen la posición relativa de los canales de ruteo en relación con la posición de los CLB, definen también, la conexión de cada canal con otros canales y el número de alambres en cada canal. Los detalles finos de los recursos de interconexión especifican la longitud de los alambres, la cantidad de interruptores y los patrones entre los alambres y las terminales de los CLB.

Existen varios modos de diseñar los recursos de interconexión, se describirán los cuatro modos principales: directo, segmentado, jerárquico e isla.

Directo

En el modo directo, figura 2.8, se hacen grupos de conexiones que cruzan al dispositivo en todas direcciones. Los CLB colocan los datos en los canales más adecuados de acuerdo al destino de los mismos. Esta implementación incluye regularmente algunas conexiones adicionales cortas que conectan los bloques más cercanos. La principal ventaja de la conexión directa es que la resistencia y capacitancia parásita son casi constantes, lo cual conduce a una mejor predicción en los tiempos de propagación de la señal.

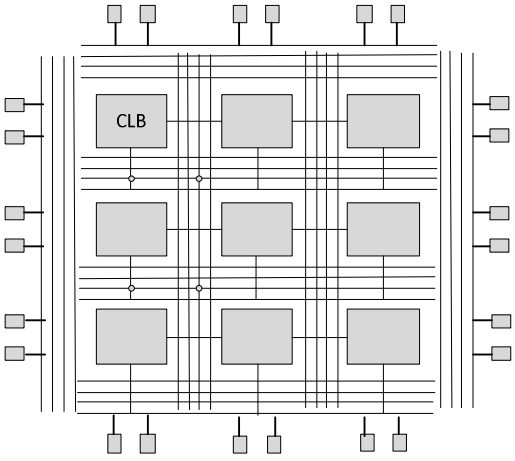


Figura 2.8. Modo directo.

Segmentado

La figura 2.9 ilustra el modo segmentado. Está basado en líneas que pueden ser interconectadas utilizando matrices programables. En este tipo de conexiones hay líneas que cruzan completamente al dispositivo, con el fin de maximizar la velocidad de comunicación y limitar el ruido en la señal. Ofrece un consumo de potencia reducido debido a que la resistencia y la capacitancia de las líneas de interconexión son sólo de la longitud necesaria para la conexión entre bloques.

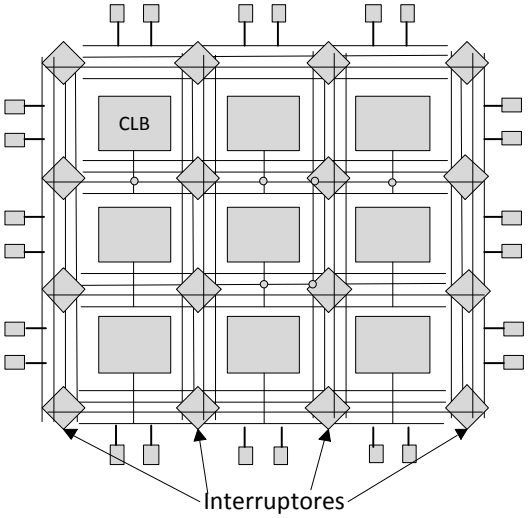


Figura 2.9. Modo segmentado.

Jerárquico

El modo jerárquico separa los CLB en distintos grupos. Las conexiones entre CLBs dentro de un grupo se pueden hacer utilizando segmentos de alambres del nivel más bajo de la jerarquía. Las conexiones entre CLB de distintos grupos necesitan de recorridos de uno o mas niveles de la jerarquía. En la figura 2.10 se ilustra el modo jerárquico donde se aprecia que sólo un nivel (el nivel 1 de la jerarquía) está conectado entre CLBs. Aunque cualquier nivel de jerarquía generalmente proporciona un retraso constante entre los mismos miembros, la distancia física y las diferencias resultantes en la capacitancia y la resistencia de la interconexión provocan una variación en el retardo entre los bloques.

Isla

Como se muestra en la figura 2.11, los CLBs están acomodados en una malla de dos dimensiones, con los recursos de conexión distribuidos uniformemente a lo largo de la misma malla. Comúnmente tiene los canales de ruteo en los cuatro lados de los CLBs. Generalmente utiliza segmentos de alambre de diferentes longitudes para cada conexión. Debido a que los alambres de enrutamiento son de diferentes longitudes y están físicamente cercanos a los CLBs, se pueden formar una gran variedad de conexiones eficientemente.

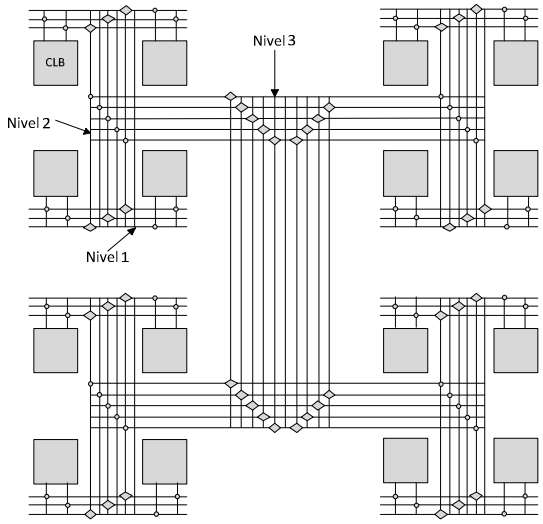


Figura 2.10. Modo jerárquico.

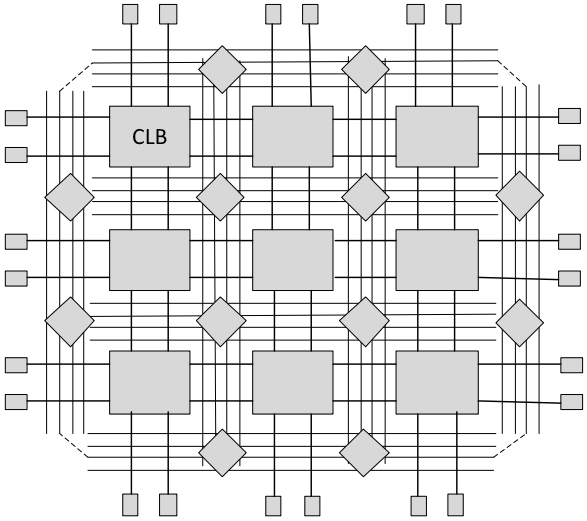


Figura 2.11. Modo isla.

Adicionalmente a estos modos de interconexión, algunos FPGA contienen redes de interconexión dedicadas a las señales de reloj que se distribuyen a lo largo de todo el dispositivo. Estas redes están cuidadosamente diseñadas para que tengan retardos (*skew*) pequeños. Estas líneas se pueden conectar directamente a *flip-flops* y las redes sólo pueden ser controladas por un número limitado de recursos del FPGA.

Líneas de conexión en la SPARTAN 3E

En el FPGA SPARTAN 3E existen cuatro tipos de líneas para la interconexión entre recursos: líneas largas, líneas hex, líneas dobles y líneas directas.

Líneas largas: se extienden horizontal y verticalmente a lo largo de la matriz y se conecta a uno de cada seis CLBs. Debido a su baja capacitancia, estas líneas son adecuadas para transportar señales de alta frecuencia con efectos de carga mínimos, como el *skew*.

Líneas hex: conectan uno de cada tres CLBs tanto horizontal como verticalmente.

Líneas dobles: conectan uno de cada dos CLBs, tanto horizontal como verticalmente, en las cuatro direcciones, brindando una gran flexibilidad.

Líneas directas: conectan un CLB con sus vecinos de forma vertical, horizontal ó diagonal. Estas líneas a menudo conducen las señales desde la fuente a una línea doble, hex o larga y viceversa, de una línea doble, hex ó larga a una línea directa.

En la figura 2.12 se muestran los 4 tipos de líneas de conexión.

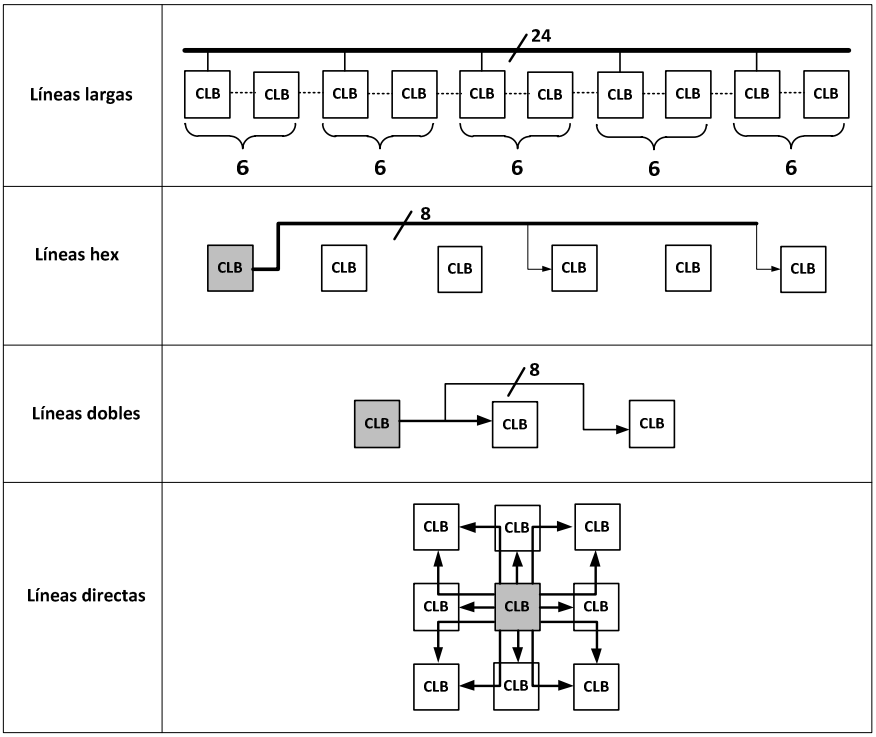


Figura 2.12. Líneas de conexión en la SPARTAN 3E.

2.1.4. Recursos adicionales

Los recursos adicionales que se describirán son: bloques de memoria embebida, multiplicadores embebidos y administrador digital de reloj.

Bloques de memoria embebida

Muchos diseños requieren el uso de cierta cantidad de memoria dentro del circuito integrado. Utilizando los recursos internos del FPGA (como las LUTs) es posible construir una memoria de

tamaño variable, sin embargo, esta cantidad de memoria suele ser insuficiente y agotarse rápidamente. La solución a este problema es proporcionar una cantidad fija de memoria embebida dentro del FPGA. Este tipo de memoria es llamada bloques de RAM ó BRAM.

La SPARTAN 3E incorpora de 4 a 36 bloques de memoria RAM dedicada (dependiendo del dispositivo), que está organizada en bloques de 18 kbit cada uno, con una estructura de puerto doble. Físicamente están colocados en una o dos columnas, dependiendo del tamaño del dispositivo, al lado de los multiplicadores embebidos, como se muestra en la figura 2.13.

Multiplicadores embebidos

Los multiplicadores son muy útiles en aplicaciones de procesamiento digital de señales, donde los algoritmos tienen gran cantidad de operaciones aritméticas como las sumas y las multiplicaciones. Incluir multiplicadores embebidos reduce en gran medida el consumo de CLBs, donde también se pueden realizar dichas operaciones. Los multiplicadores se encuentran regularmente colocados junto a los bloques de memoria BRAM, ilustrados también en la figura 2.13.

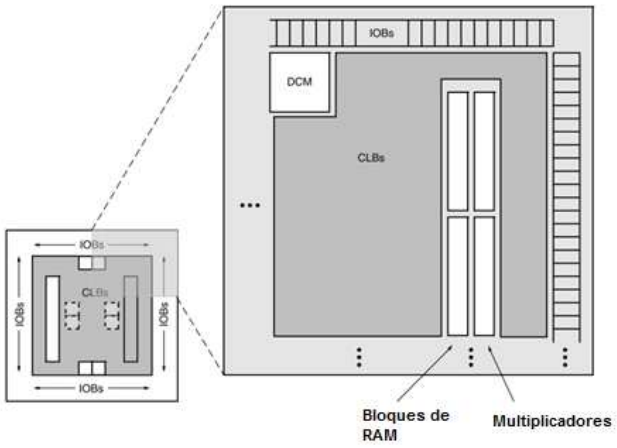


Figura 2.13. Ubicación de los BRAM y multiplicadores.

Administrador digital de reloj

La mayoría de los sistemas digitales cuenta con una sola fuente de reloj externa, la cual proporciona una frecuencia fija de operación. Sin embargo, algunos diseños pueden requerir diferentes frecuencias en sus entradas de reloj para los diferentes módulos que componen al mismo. Un administrador digital de reloj (DCM: *Digital Clock Manager*) proporciona valores de frecuencias variadas a partir de una sola entrada de reloj. Los DCM proporcionan tres funciones básicas: eliminación de retardos (*skew*), síntesis de frecuencias (multiplicación y división de frecuencias) y cambio de fase de la señal.

Para el caso de la familia SPARTAN 3E dependiendo el dispositivo son los DCM con los que cuentan cada miembro. Por ejemplo, el dispositivo de menor capacidad cuenta solo con dos DCM

colocados en la parte superior e inferior, mientras que el dispositivo de mayor capacidad dentro de esta familia de FPGAs cuenta con ocho DCM colocados en pares en los bordes del dispositivo.

2.1.5. Tecnologías de configuración

En esta sección se presentan las diferentes tecnologías que se utilizan para el almacenamiento de datos para la configuración de los FPGAs.

SRAM

La tecnología de configuración más utilizada en los FPGA comerciales es la tecnología SRAM. Esta tecnología se utiliza mucho debido a que proporciona una rápida configuración, así como una infinidad de veces. La información se almacena en celdas de memoria SRAM. Cuando se implementa una red de interconexión por medio de transistores de paso, la SRAM controla el encendido/apagado de dichos dispositivos. Las desventajas que presenta la tecnología SRAM son: el consumo de energía debido a las corrientes de fuga es grande, comparada con otras tecnologías la celda SRAM es grande (entre 6 y 12 transistores) y la principal desventaja, que al quitar la energía, la configuración almacenada se pierde, por lo cual se tiene que volver a configurar.

Flash

La principal ventaja de la tecnología flash es la no volatilidad de la configuración del dispositivo, es decir, cuando la energía se retira del dispositivo, la configuración del usuario permanece dentro del mismo. A diferencia de la tecnología SRAM, la celda de la memoria flash se puede construir con pocos transistores, por tal motivo, el consumo de energía debido a corrientes de fuga disminuye. Algunas desventajas de la memoria flash es que el ciclo para la escritura tiene un límite (alrededor de cientos de miles de millones) y por lo regular es lento comparado con la SRAM. Para realizar una operación de escritura con tecnología flash, se necesitan voltajes de mayor magnitud que los utilizados en la circuitería habitual, esto implica tener estructuras adecuadas para generar dichos voltajes.

Anti-fusible

La tecnología anti-fusible tiene la característica de funcionar completamente inverso al funcionamiento de un fusible, es decir, el anti-fusible está normalmente abierto (desconectado), durante el proceso de programación se aplica una corriente muy grande o un láser funde el anti-fusible para formar una conexión eléctrica entre las terminales del anti-fusible. Las ventajas de esta tecnología son que los anti-fusibles se pueden construir en tamaños muy pequeños comparados con las celdas SRAM y no requieren ningún transistor (por lo tanto no hay consumos de energía por corrientes de fuga), no es susceptible a partículas de altas energías que provocan anomalías en otras tecnologías, esto los hace apropiados para aplicaciones espaciales. La desventaja principal de la tecnología anti-fusibles es que no es reconfigurable, una vez que el anti-fusible es fundido no se puede volver a desconectar, lo cual los descarta en el campo de la reconfiguración y en la construcción de prototipos.

2.2. Lenguajes de descripción de hardware

Debido al incremento en la capacidad de integrar cada vez mas componentes en un solo circuito, los sistemas digitales se han hecho más y más complejos. Diseñar estos sistemas a nivel de compuertas lógicas es tedioso y casi imposible. Por este motivo, el uso de lenguajes de descripción de hardware (HDL: *Hardware Description Language*) es esencial para el diseño de los mismos. Un HDL permite diseñar y depurar sistemas digitales con un alto nivel de abstracción (entre menos detalles de un diseño, la abstracción es mayor) para después convertirlo en un diseño a nivel de compuertas lógicas. Síntesis es el proceso por el cual se genera un circuito lógico, a nivel de compuertas, a partir de un diseño con un alto nivel de abstracción. Existen muchos HDLs en el mercado: VHDL, Verilog, System C, Handel C, ABEL, etcétera. Los HDL más utilizados en la industria son VHDL y Verilog. En el presente trabajo se dará una explicación únicamente de VHDL.

2.2.1. VHDL

VHDL es el acrónimo de *Very High Speed Integrated Circuit (VHSIC) Hardware Description Language (HDL)*, es decir, lenguaje de descripción de hardware de circuitos integrados de muy alta velocidad. Fue desarrollado a principios de los años 1980 por el departamento de defensa de los Estados Unidos junto con tres grandes empresas: IBM, Texas Instrument e Intermetrics. Su intención era proporcionar uniformidad en los diseños digitales. La primer versión disponible al público de VHDL fue lanzada en 1985, en 1987 el Instituto de Ingenieros Eléctricos y Electrónicos (IEEE: *Institute of Electrical and Electronics Engineers*) lo adoptó como estándar y lo llamó IEEE 1076. Una actualización se realizó en 1994 y se renombró como IEEE 1076-1993.

Algunas ventajas que ofrece VHDL son: diseño y simulación a un alto grado de abstracción, las descripciones creadas en VHDL se pueden utilizar en otros diseños; como es un estándar, puede ser utilizado por diversas herramientas de síntesis para implementar circuitos digitales. VHDL fue diseñado para que sea independiente de la tecnología en uso, si un diseño es descrito con VHDL e implementado con la tecnología actual, esa descripción puede ser utilizada como punto de partida para las nuevas tecnologías.

VHDL permite describir un sistema digital en diferentes niveles de abstracción: algorítmica, flujo de datos y estructural. Es importante mencionar que la filosofía más utilizada para realizar diseños digitales con VHDL es la llamada alto-bajo (*Top-Down*). Esta filosofía permite realizar un diseño con un alto grado de abstracción e implementarlo partiendo de esta misma descripción, para que posteriormente se incremente el nivel de detalles según sea necesario.

Algorítmico

El nivel más alto de abstracción soportado por VHDL es llamado algorítmico. Cuando se describe con este nivel de abstracción se tiene que describir el circuito en términos de su funcionamiento ó comportamiento. Una de sus características principales es que su descripción contiene procesos que son concurrentes entre sí, es decir, que se ejecutan simultáneamente. Dentro de estos

procesos existen sentencias que se ejecutan secuencialmente. Estas sentencias secuenciales se caracterizan porque son del tipo si-entonces-si no (*if –then-else*).

Flujo de datos

Con este tipo de abstracción se describen los circuitos en términos de cómo los datos se mueven a través del sistema, es decir, se describe cómo la información es transferida entre los registros del sistema. Es llamado también, transferencia entre registros (RTL: *Register Transfer Logic*). Este nivel de abstracción es un nivel intermedio entre el algoritmo y el estructural, ya que define el comportamiento de los módulos, así como la conexión entre ellos. La característica de este nivel de abstracción es que utiliza únicamente sentencias concurrentes de tipo cuando-si no (*when-else*) o con ecuaciones booleanas.

Estructural

Este tipo de descripción se utiliza para describir un diseño en términos de sus componentes, es decir, un conjunto de componentes interconectados entre sí con señales. Una lista de conexiones (*netlist*) consiste en dar una lista de componentes, sus interconexiones y las entradas y salidas del diseño. En principio se utilizaron estas *netlist* para describir circuitos digitales. Cabe aclarar que esos componentes a los que se hace referencia deben estar contenidos en una biblioteca ya sea creada por el diseñador o proporcionada por la herramienta utilizada.

Sin importar qué nivel de abstracción se utilice, en cualquier diseño realizado con VHDL se debe definir el símbolo ó entidad, llamada *entity* en VHDL, del circuito a diseñar. En esta entidad se definen las entradas y salidas que posee el circuito. Es importante aclarar que una entidad es única para cada circuito pero se pueden tener diferentes vistas ó arquitecturas, llamada *architecture* en VHDL, de dicho circuito, es decir, puede existir una descripción estructural y otra algorítmica del mismo circuito.

En la figura 2.14 se muestra un multiplexor que se puede implementar con los tres tipos de abstracción descritos anteriormente.

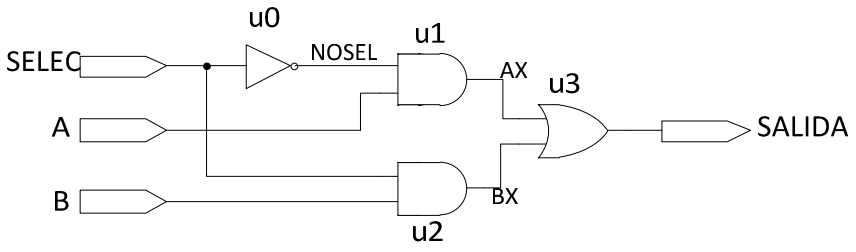


Figura 2.14. Multiplexor.

En el inciso *a* de la figura 2.15 se muestra la entidad del multiplexor de la figura 2.14, la cual es común a los tres niveles de abstracción. En el inciso *b* se muestra la arquitectura algorítmica, en el inciso *c* se muestra la de flujo de datos y por último, en el inciso *d*, se muestra la estructural.

a)
```
ENTITY MUX IS          ---ENTIDAD LLAMADA MUX
PORT( A: IN BIT;       ---
      B: IN BIT;       ---SALIDA Y ENTRDAS DE TIPO
      SELEC: IN BIT; ---BIT
      SALIDA: OUT BIT);
END MUX;               ---FIN DE LA ENTIDAD
```

b)
```
ARCHITECTURE ALGO OF MUX IS --ARQUITECTURA ALGORITMICA
BEGIN                       --DE LA ENTIDAD MUX
   PROCESS(A,B,SELEC)       --PROCESO CONCURRENTE
   BEGIN
      IF SELEC='0' THEN     --
         SALIDA<=A;         --SENTENCIAS
      ELSE                  --SECUENCIALES
         SALIDA<=B;         --
      END IF;               --
   END PROCES;
END ALGO;
```

c)
```
ARCHITECTURE FLUJO OF MUX IS --ARQUITETURA FLUJO DE
BEGIN                         --DATOS DE LA ENTIDAD
                              --MUX
   SALIDA<=A WHEN SELEC='0' ELSE --SENTENCIAS
            B;                   --CONCURRENTES
END FLUJO;
```

d)
```
ARCHITECTURE ESTRUCTURAL OF MUX IS  --ARQUITECTURA
SIGNAL AX, BX, NOSEL: BIT;          --ESTRUCTURAL
BEGIN                               --DE LA ENTIDAD MUX
   U0: ENTITY INV PORT MAP(E=>SELEC, Y=>NOSEL);       --
   U1: ENTITY AND2 PORT MAP(E1=>A, E2=>NOSEL, Y=>AX); --COMPONENTES Y
   U2: ENTITY AND2 PORT MAP(B, SELEC, BX);            --CONEXIONES ENTRE
   U3: ENTITY OR2 PORT MAP(E1=>AX, E2=>BX, Y=>SALIDA);--ELLOS
END ESTRUCTURAL;
```

Figura 2.15. Multiplexor con VHDL.

2.3. Configuración del FPGA

El desarrollo de un sistema basado en FPGA es un proceso complicado que consta de transformaciones y algoritmos complejos, las herramientas de software, comúnmente denominadas IDE, son imprescindibles para automatizar y agilizar estos procesos. Existen diversos IDEs en el mercado para el diseño con FPGAs, todos ellos comparten los procesos de diseño, simulación e implementación.

2.3.1. Flujo de diseño con herramientas IDE

El flujo de diseño consiste en los procesos generales para realizar la implementación de un diseño utilizando IDEs. Existen otros procesos, los cuales, dependiendo de la aplicación, se deben realizar, por ejemplo, análisis de tiempos y análisis de potencia. A continuación se enlistan los procesos básicos del flujo de diseño:

- Diseño del circuito con HDL. En esta etapa se generan los archivos HDL, ya sea a partir de diagrama de estados, diagramas esquemáticos o directamente con la sentencias del HDL utilizado.

- Simulación del circuito HDL. Se crea un banco de prueba para verificar el correcto funcionamiento de la lógica diseñada. El banco de prueba puede ser de forma gráfica o en forma de sentencias.

- Síntesis. Se conoce también como síntesis lógica, el software transforma los archivos HDL a componentes genéricos como compuertas lógicas y *flip-flops*.

- Implementación. Consiste en tres sub-procesos: *translate*, *map* y *place and route*:

    - El proceso *transalte* convierte los múltiples archivos del diseño en una sola *netlist*.
    - El proceso *map* o mapeo, distribuye el diseño en los recursos internos del FPGA (CLB e IOB).
    - Por último, el proceso *place and route* deriva del diseño físico del FPGA, se coloca el diseño en locaciones físicas y determina las rutas para conectar las señales del diseño.

- Generación del *bitstream*. Ya que se han asignado los recursos internos del FPGA, se genera el archivo que contiene toda esta información. Contiene la información tanto del diseño como de las conexiones programables que se requieran. A este archivo de configuración se le conoce como *bitstream*.

- Por último se descarga el *bitstrem* al dispositivo a utilizar. Para la descarga del *bitstream* se utilizan diferentes métodos dependiendo la aplicación.

En la figura 2.16 se muestre un diagrama de flujo donde se encuentran los procesos básicos del flujo de diseño anteriormente mencionados.

Un ejemplo concreto donde se utilizan estos recursos de descarga de configuración externa es en la tarjeta de desarrollo SPARTAN 3E MicroBalze, donde existen varias formas de configurar el FPGA con componentes externos. Estos métodos de configuración son: *Master Serial Mode, SPI Serial Flash Mode, Byte-Wide Peripheral Interface Paralle model, Slave Parallel Mode, Slave Serial Mode y JTAG Mode.*

Master Serial

En este modo de configuración, el FPGA se configura desde una memoria externa, la cual es una Xilinx PLATFORM Flash PROM. El FPGA proporciona la señal de salida CCLK de su oscilador interno hacia dicha memoria. Como respuesta, la memoria proporciona los datos en forma serie al FPGA en su terminal de entrada DIN.

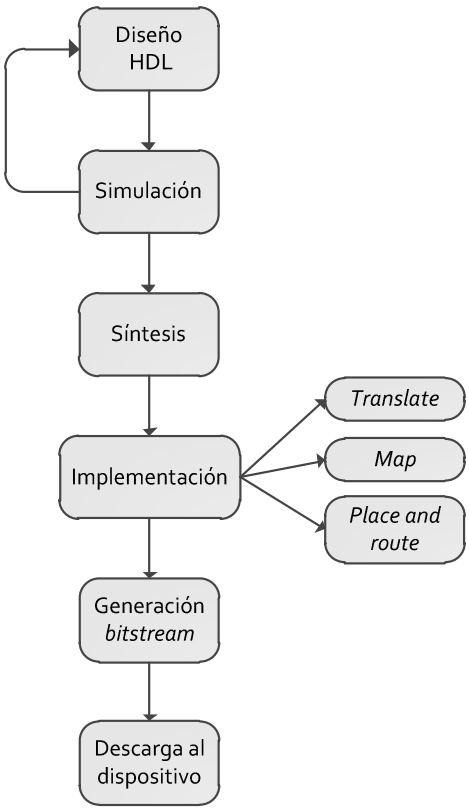


Figura 2.16. Diagrama de flujo procesos básicos.

SPI Serial Flash

El FPGA se configura a sí mismo desde una memoria PROM Flash serie SPI de la empresa ST microelectronics. El FPGA proporciona la salida de reloj en la terminal CCLK desde el oscilador interno a la entrada de reloj de la PROM SPI.

Byte-Wide Peripheral Interface

El FPGA se configura a sí mismo desde una memoria NOR Flash PROM paralela. El FPGA genera hasta 24 bits de direcciones para acceder a la memoria externa. Está diseñada primordialmente para memorias estándar NOR Flash PROM y soporta ambos tamaños: byte (x8) y byte/media palabra (x8/x16).

Parallel Slave

En el modo esclavo paralelo, un dispositivo externo (como un microprocesador ó un micro-controlador) escribe los datos de la configuración en el FPGA utilizando una interface de 8 bits.

Slave Serial

En el modo esclavo serial, un dispositivo externo (como un microprocesador ó un micro-controlador) escribe de manera serie los datos para la configuración del FPGA. Los datos se presentan en la entrada DIN del FPGA.

JTAG

La SPARTAN 3E tiene cuatro terminales dedicadas al puerto JTAG, que están siempre disponibles para recibir los datos para la configuración del FPGA.

Nos enfocaremos en el modo JTAG, ya que es de interés particular en el presente trabajo de tesis.

2.3.2. JTAG

El estándar IEEE 1149.1 lleva por nombre: *Standard Test Access Port and Boundary-Scan Architecture* y se conoce como JTAG. Se estandarizó en 1990 y en 1994 se agregó un suplemento que contiene una descripción sobre los modelos *Boundary Scan Description Language* (BSDL).

El JTAG es utilizado para pruebas de sub-módulos de circuitos integrados y es muy útil como herramienta para depuración de aplicaciones embebidas. Otras aplicaciones de JTAG incluyen: prueba de circuitos conectados en cadena, prueba del funcionamiento de la lógica del circuito, programación en el sistema (ISP: *In System Programming*) y prueba de funcionalidad del circuito.

Una interfaz JTAG es una interfaz especial de cuatro, o cinco terminales, agregadas a un circuito integrado en su fabricación. Esta interfaz permite que varios circuitos integrados, en una tarjeta, puedan tener sus líneas JTAG conectadas en cadena, con el objetivo de que una sola sonda de prueba JTAG se conecte a un solo circuito integrado para así acceder a todos los demás circuitos conectados.

Los elementos básicos que componen una interface JTAG son: Test Access Port (TAP), el controlador de TAP, el registro de instrucciones, el decodificador de instrucciones, el registro *Boundary Scan* (BSR) y el registro *BYPASS*. El TAP contiene cuatro terminales especificadas por el protocolo: TDI, TDO, TMS y TCK. Una quinta terminal opcional es TRST, la cual no todos los dispositivos manejan. Así se definen las cinco terminales del protocolo:

- TDI: *Test Data In*, es la entrada de datos serie para todas las instrucciones y datos del protocolo JTAG.

- TDO: *Test Data Out*, es la salida de datos serie para todas las instrucciones y datos del protocolo JTAG.

- TCK: *Test Clock*, es la señal de reloj para el controlador TAP y los registros JTAG.

- TMS: *Test Mode Select*, es la terminal que determina la secuencia a través del controlador TAP.

- TRST: *Test Reset*, reinicio asíncrono de la prueba (es una terminal opcional).

Para controlar el flujo de datos se utiliza un controlador TAP, este controlador consiste en una máquina finita de 16 estados, manejada principalmente con las señales TCK y TMS, figura 2.17. El valor de TMS, en el borde positivo de TCK, determina el siguiente estado en la secuencia de la máquina.

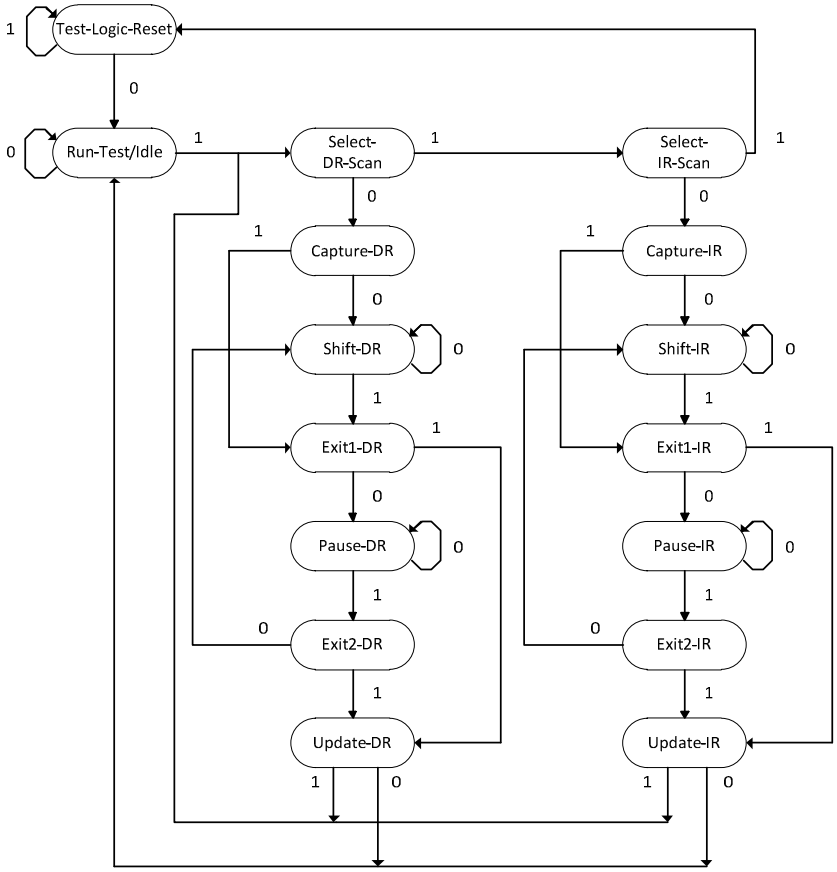


Figura 2.17. Máquina de estados JTAG.

A continuación se da una breve explicación sobre los estados de la máquina, figura 2.17.

- *Test-Logic-Reset*: todas las pruebas lógicas son deshabilitadas en este estado, habilitando así el funcionamiento normal del dispositivo. Se puede alcanzar este estado manteniendo la señal TMS en valor “1” lógico y con cinco pulsos en el borde positivo de la señal TCK. Es por eso que la señal TRST es opcional.

- *Run-Test/Idle*: en este estado, la lógica de prueba es activada en el circuito sólo si están presentes ciertas instrucciones. Por ejemplo, si una instrucción activa la auto-prueba (*selft-test*), ésta es ejecutada cuando el controlador llega a este estado.

- *Select-DR-Scan*: controla si entra al estado Capture-DR (ruta de datos) ó al estado *Select-IR-Scan*.

- *Select-IR-Scan*: controla si entra al estado *Capture IR* (ruta de instrucciones) ó regresa al estado *Test-Logic-Reset*.

- *Capture-IR*: en este estado, el *shift register bank* en el registro de instrucciones carga en paralelo un patrón de valores fijos en el borde positivo de TCK.

- *Shift-IR*: aquí, el registro de instrucción se conecta entre TDI y TDO y el patrón capturado es desplazado en cada borde de subida del TCK. La instrucción disponible en el TDI también se mueve en el registro de instrucción.

- *Exit1-IR*: controla si entra al estado *Pause-IR* ó *Update-IR*.

- *Pause-IR*: permite el desplazamiento del registro de instrucciones para ser detenido temporalmente.

- *Exit2-IR*: controla si entra al estado *Shift-IR* o *Update-IR*.

- *Update-IR*: la instrucción en el registro de instrucción es retenida en el *latch bank* del registro de instrucción en cada borde de bajada del TCK. Esta instrucción se convierte en la instrucción actual una vez que está retenida.

- *Capture-DR*: los datos son cargados paralelamente en el registro de datos seleccionado por la instrucción actual en el borde de subida del TCK.

- *Shift-DR*, *Exit1-DR*, *Pause-DR*, *Exit2-DR* y *Update-DR*: son similares a los estados *Shift-IR*, *Exit1-IR*, *Pause-IR*, *Exit2-IR* y *Update-IR* de la ruta de instrucciones.

La manera general de operar de la máquina de estados se describe a continuación: al inicio se envía en serie, por medio de la terminal TDI, la instrucción a realizar en forma binaria. La instrucción se aplica a los bloques involucrados hasta que es cargada completamente, debido a que una instrucción está conformada por seis bits. Ya que se cargó completamente la instrucción, los bloques seleccionados del circuito se configuran para responder a dicha instrucción. En algunos casos, es necesario ingresar datos para realizar la acción solicitada por la instrucción, estos datos se ingresan también por la terminal TDI. Después de que la instrucción se ha ejecutado, el

resultado se puede examinar por medio del desplazamiento de datos en la terminal TDO. Se puede realizar la misma instrucción con diferentes datos sin necesidad de cargar nuevamente la instrucción.

Como ya se ha mencionado, existen dos tipos de registro dentro del protocolo JTAG: registro de instrucciones y registro de datos. El registro de instrucciones mantiene la instrucción actual, su contenido es usado por el controlador TAP para decidir qué hacer con las señales que se reciben. El contenido del registro de instrucción va a definir a que señales de los registros de datos se debe pasar. Existen por lo menos tres registros de datos: el BSR, el registro BYPASS y el registro *IDCODES*. El BSR es el registro principal y es usado para mover datos desde y para el dispositivo. El BYPASS es un registro de 1 bit que pasa la información de TDI a TDO y el registro *IDCODES* contiene el identificador (ID) del dispositivo.

En la figura 2.18 se muestra una arquitectura típica JTAG, se observan los registros anteriormente mencionados, así como el controlador TAP y las cuatro terminales necesarias del protocolo.

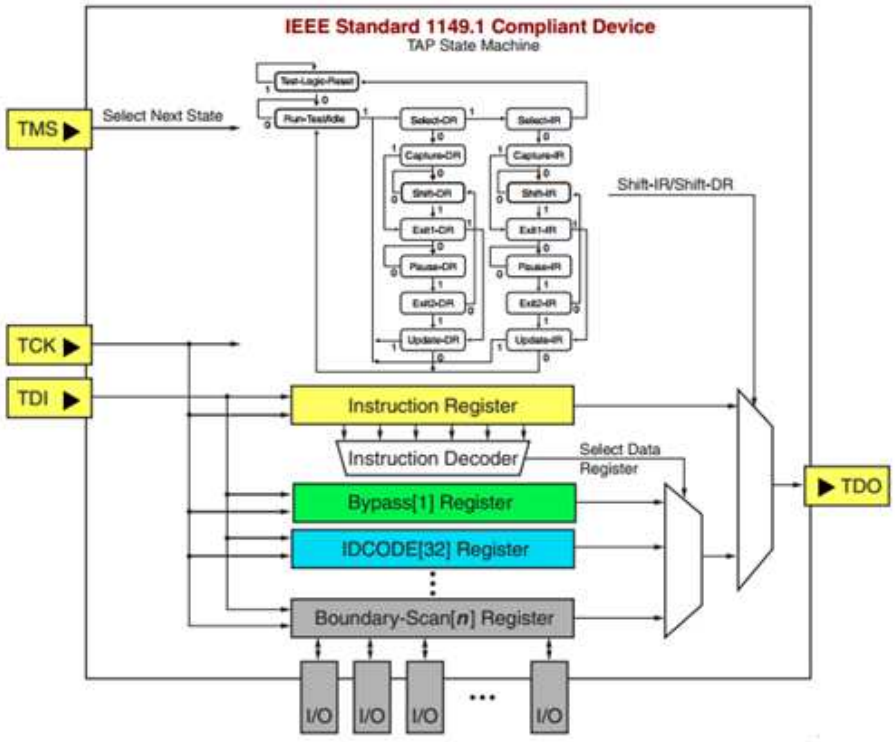


Figura 2.18. Arquitectura típica JTAG.

JTAG en la SPRATN 3E

La familia SPARTAN 3E es compatible con el protocolo JTAG, soporta comandos propios del estándar JTAG, así como comandos específicos de Xilinx. Las instrucciones propias del protocolo como: EXTEST, INTEST, SAMPLE/PRELOAD, BYPASS, IDCODE, USERCODE y HIGHZ están incluidas en la familia. También incluye dos registros utilizables por el usuario (USER1 y USER2).

La familia SPARTAN 3E contiene todos los registros necesarios para el estándar JTAG y además contiene registros opcionales para simplificar la verificación y las pruebas realizadas a dichos circuitos. Los registros que contiene la familia SPARTAN 3E son: BSR, *Instruction*, BYPASS, *Identificatión*, C*onfiguration*, *User code*, USER1 y USER2, que a continuación se describen:

BSR

Cada IOB contiene lógica adicional que conforma el BSR. Las operaciones del BSR son independientes de la configuración individual de cada IOB. De inicio, cada IOB es configurado como bidireccional. Se puede configurar cada IOB por medio del JTAG para que se comporte como entrada, salida ó tres estados (alta impedancia). En la figura 2.19 se muestra la lógica del BSR.

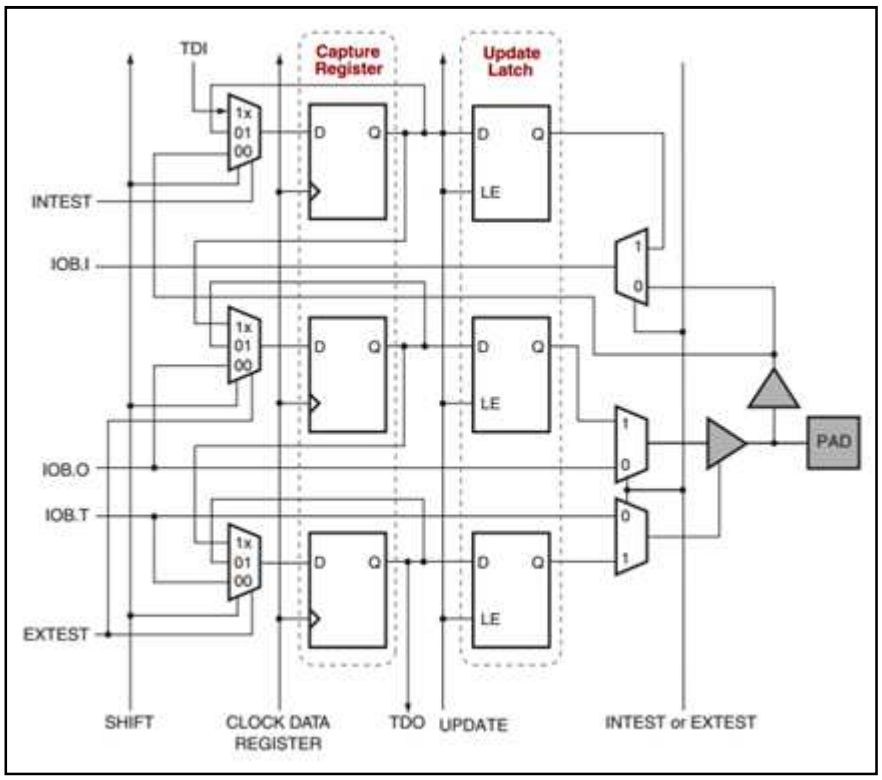


Figura 2.19. Lógica *Boundary Scan* por cada IOB.

Registro *Instruction*

El registro *Instruction* es conectado entre TDI y TDO durante la secuencia de escaneo de la instrucción. Para llamar una operación, se carga el código de operación (OPCODE: *Operation Code*) en el registro *Instruction*, el cual es de seis bits.

Registro BYPASS

Consiste solamente en un *flip-flop* conectado entre TDI y TDO. Permite el paso de los datos en forma serie desde TDI hasta TDO durante la instrucción BYPASS.

Registro *Identificatión* (ID)

La SPARTAN 3E tiene un registro de 32 bits llamado IDCODE, el cual contiene un valor fijo y asignado por el fabricante del dispositivo para identificar eléctricamente al mismo fabricante y qué tipo de dispositivo es. Este registro es útil para una fácil identificación del dispositivos al que se esté examinando o configurando. Se puede leer este código con la instrucción IDCODE. El último bit del código IDCODE siempre es 1, por el estándar JTAG; los últimos dígitos del código de identificación en la familia SPARTAN 3E son 0x093 en formato hexadecimal.

Registro JTAG *Configuration*

Es un registro de 32 bits y permite el acceso al bus de configuración y la operación de relectura. Cuando la instrucción CFG_IN se activa, el registro JTAG *Configuration* es solamente de entrada. Cuando el CFG_OUT se activa el registro JTAG *Configuration* es sólo de salida.

Registro *User Code*

Este registro permite al usuario especificar un código de identificación. El código de usuario se programa dentro del dispositivo y se puede leer para su verificación posterior. El código de usuario es embebido en el *bitstream* durante la generación del mismo y sólo es válido después de la configuración. Si no es programado el código de usuario, el registro contiene el valor de 0xFFFFFFFF en formato hexadecimal.

Registro USER1 y USER2

Estos registros están disponibles sólo después de la configuración. Si son utilizados en la aplicación, deben ser implementados utilizando la lógica del FPGA.

2.4. Sistemas embebidos

Un sistema embebido es un sistema dedicado a un propósito en específico. El usuario final del sistema interactúa regularmente con interfaces limitadas como controles remotos o interruptores, no interactúa directamente con el sistema. A pesar de que el sistema puede ser capaz de realizar más tareas, está normalmente restringido a realizar una tarea limitada. Existen muchos ejemplos en la vida cotidiana de sistemas embebidos, como son: reproductores de DVD, Reproductores MP3, consolas de videojuegos, tarjetas de acceso, sistemas de frenado en automóviles, etcétera.

2.4.1. Sistemas embebidos en un FPGA

Debido a que un FPGA tiene la propiedad de configurarse y reconfigurarse las veces que sean necesarias, los recursos pueden ser utilizados de una manera eficiente. Esta propiedad los convierte en una excelente base para implementar sistemas embebidos. Gracias a los recursos contenidos en un FPGA como BRAM, multiplicadores, DCM y hasta procesadores, es posible implementar un sistema embebido completo y funcional, en un único circuito integrado.

Extendiendo aún más las propiedades de un FPGA para utilizarlo en sistemas embebidos, se encuentran los núcleos de propiedad intelectual (IP: *Intellectual Property*), también conocidos como módulos ó simplemente, núcleos. Estos IP se refieren a una especificación de hardware que realiza una tarea muy particular, por ejemplo, un controlador CAN (*Controller Area Network*), un decodificador de imágenes JPEG: *Joint Photographic Experts Group* ó un generador de señales senoidales. Los IP pueden ser de dos tipos: *hard core* ó *soft core*. Los del tipo *hard core* son físicamente parte del silicio del dispositivo, es decir, se fabrican a nivel transistor dentro del FPGA. Los *soft core* son implementados en los recursos configurables del FPGA como en los generadores de funciones ó en las memorias, por lo tanto, están hechos con un HDL, por lo general en VHDL.

Entre las ventajas de un *hard core* se encuentran: altos niveles de optimización y rendimiento, es similar a un circuito integrado estándar, los retardos para acceder a las funciones son muy pequeños. Su principal desventaja cosiste en que es fijo, no se puede modificar, ni agregar ni quitar características.

Las principales ventajas de un *soft core* son: altamente portables debido a que están hechos con un HDL, se pueden implementar en diferentes arquitecturas de FPGA, se pueden personalizar agregando o quitando características. Su principal desventaja es que no están optimizados debido a la portabilidad y que los resultados pueden variar dependiendo de las herramientas utilizadas en la implementación.

En la figura 2.20 se muestra un ejemplo de los dos tipos de IP que se pueden encontrar en un FPGA. Se observa que el *soft core* está colocado en los recursos configurables del FPGA, mientras que el *hard core* se encuentra colocado en vez de los recursos configurables.

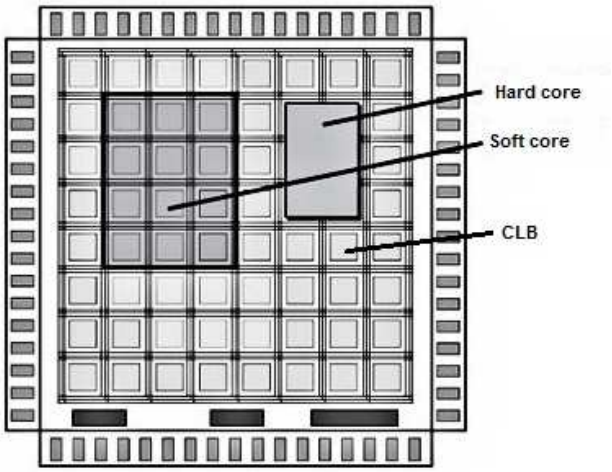


Figura 2.20. *Hard* y *soft core* en un FPGA.

2.5. Comunicación SPI

El protocolo SPI (*Serial Peripheral Interface*) fue desarrollado por la empresa Motorola para proporcionar una interface simple entre micro-controladores y periféricos. Debido a la flexibilidad y sencillez de este protocolo, muchos componentes ajenos a Motorola lo utilizan.

El SPI es un protocolo síncrono, ya que todas las transmisiones están referencias a una señal de reloj generada por el dispositivo maestro. Es implementado con una estructura maestro-esclavo, donde el maestro es el que controla la transmisión de datos. Consiste en cuatro líneas principales, las cuales son nombradas como: MOSI, MISO, CLK ó SCLK y CS. A continuación se da una explicación de cada señal:

- MOSI: *Master Out Slave In*, esta señal es generada por el maestro y es recibida por el esclavo.

- MISO: *Master In Slave Out*, esta señal es generada por el esclavo y es recibida por el maestro.

- CS: *Chip Select*, es la señal con la cual se activa o desactiva un esclavo. Por lo general se activa a un esclavo cuando la señal CS está en un nivel lógico bajo y se desactiva cuando está en un nivel lógico alto.

- SCLK: *Slave Clock*, es la señal de reloj que sincroniza la transferencia de datos y es generada por el maestro, regularmente de entre 8, 16 y 32 pulsos.

En la figura 2.21 se muestra la conexión típica del protocolo SPI, en ella, se observa al maestro, las cuatro líneas necesarias, las direcciones de las mimas y al esclavo.

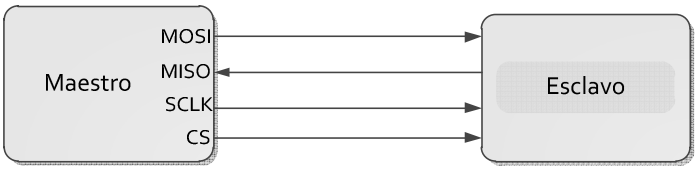


Figura 2.21. Comunicación SPI.

Debido a que cuenta con las líneas MOSI y MISO por separado, se logra una comunicación *full dúplex*, la cual permite enviar y recibir datos al mismo tiempo.

Se puede expandir las conexiones a más de un esclavo. Esto se logra compartiendo las líneas MOSI, MISO y SCLK. Cada esclavo debe tener su propia línea CS, así como se muestra en la figura 2.22. Un inconveniente del protocolo SPI es que tantos esclavos sean conectados, tantas líneas CS se deben tener y por consecuencia aumentar conexiones.

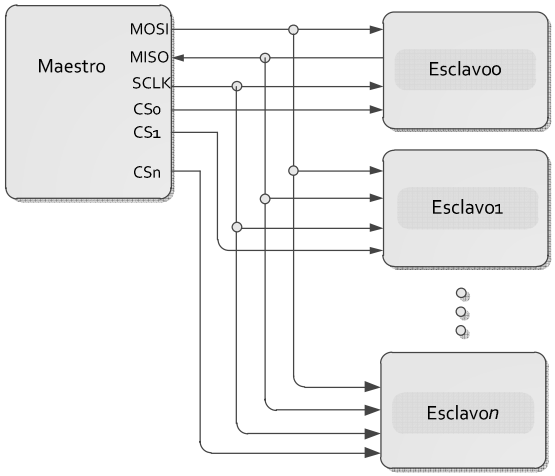


Figura 2.22. Comunicación SPI multi-esclavos.

Una trama típica de una comunicación SPI se muestra en la figura 2.23. Se observa que los datos, tanto MOSI como MISO, están sincronizados con la señal SCLK. También se observa que la activación de CS se realiza en un nivel lógico bajo.

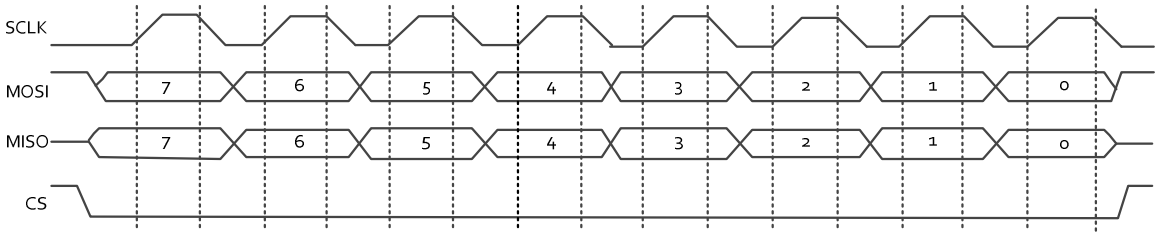


Figura 2.23. Comunicación SPI típica.

El protocolo SPI tiene cuatro formas de operar, las cuales dependen de la polaridad (CPOL: C*lock Polarity*) y fase (CPHA: *Clock Phase*) de la señal de reloj SCLK. Estos modos hacen posible seleccionar en qué momento del ciclo de reloj se transmite la información. CPOL determina si el estado de reposo (*idle*), de la señal de reloj, está en un nivel lógico bajo, CPOL=0, ó si está en un nivel lógico alto, CPOL=1. CPHA determina en qué flanco de la señal de reloj son válidos los datos en la línea MOSI, si CPHA=0 los datos son válidos en el primer flanco del reloj (ya sea de subida o de bajada) y con CPHA=1 los datos son válidos en el segundo flanco (ya sea de subida o de bajada) de la señal de reloj. A continuación se resumen estos modos de operación y se muestran en la figura 2.24.

- Modo 1: CPOL=0 y CPHA=0. El estado de reposo del reloj está en nivel lógico bajo y los datos son válidos en el flanco de subida de la señal de reloj.

- Modo 2: CPOL=0 y CPHA=1. El estado de reposo del reloj está en nivel lógico bajo y los datos son válidos en el flanco de bajada de la señal de reloj.

- Modo 3: CPOL=1 y CPHA=0. El estado de reposo del reloj está en nivel lógico alto y los datos son válidos en el flanco de bajada de la señal de reloj.

- Modo 4: CPOL=1 y CPHA=1. El estado de reposo del reloj está en nivel lógico alto y los datos son válidos en el flanco de subida de la señal de reloj.

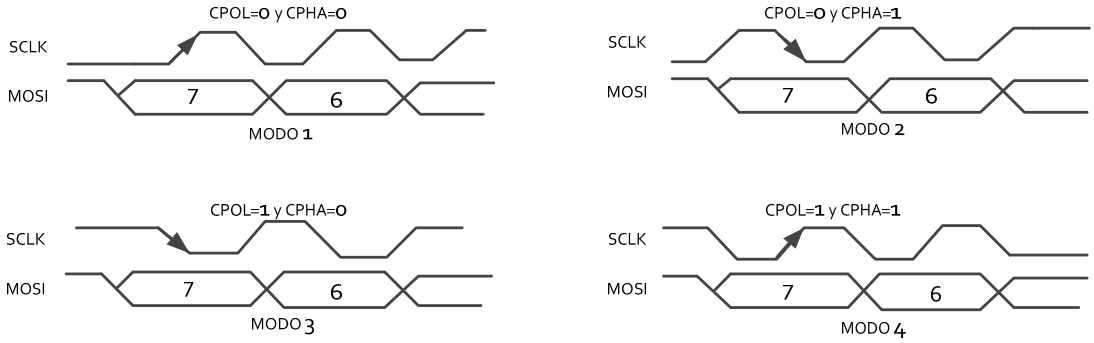


Figura 2.24. Modos de configuración SPI.

2.6. Comunicación asíncrona

Toda comunicación en serie convierte datos paralelos en datos serie y viceversa. Es utilizada cuando no es conveniente una comunicación paralela, tanto por costo como por espacio físico. Para sistemas embebidos, una comunicación serie es la opción fácil y de bajo costo para conectar un sistema computacional como parte de la aplicación o para depuración.

La forma simple de una interfaz serie es un transmisor receptor asíncrono universal (UART: *Universal Asynchronous Receiver Transmitter*). Se le denomina asíncrono ya que no se transmite una señal de reloj junto con los datos seriales, por lo tanto el receptor debe detectar los datos sin una señal de sincronización.

La UART consta principalmente de dos secciones: el receptor (Rx) que convierte los datos seriales en datos paralelos y el transmisor (Tx) que convierte los datos paralelos en datos seriales para su transmisión. También contiene información del estado, por ejemplo, si el receptor está lleno (los datos se han recibido) ó si el transmisor está vacío (la transmisión se ha completado). El principal componente de un transmisor UART es un registro de corrimiento que es cargado paralelamente y después es desplazado en cada pulso del reloj. El receptor UART recibe una serie de bits en un registro de corrimiento que después se puede leer.

Un problema asociado con la transmisión serie asíncrona es en la reconstrucción de los datos, es decir, en la recepción de los mismos. El problema principal se encuentra en la dificultad de detectar los límites entre cada bit. Por ejemplo, si la línea de datos se encuentra en un valor lógico bajo por un lapso de tiempo, la parte de recepción de la UART debe ser capaz de detectar si los datos fueron “00” o fueron “000”. Por lo tanto, debe ser capaz de conocer cuándo inicia y termina

cada bit. En el caso de la comunicación asíncrona, cada dispositivo tiene una señal propia de reloj, las cuales deben operar exactamente a la misma frecuencia.

Regularmente una comunicación asíncrona se utiliza cuando se envía un carácter, típicamente de ocho bits a la vez, y cuando el tiempo entre cada transmisión es variable. Para transmitir se utiliza un bit de inicio, y uno o dos bits de paro para finalizarla. El receptor sincroniza su señal de reloj al momento de recibir el bit de inicio y después muestrea los bits de datos. Al recibir el bit, o bits, de parada, el receptor asume que la transmisión fue exitosa y se tiene un carácter válido. Por el contrario, si no se recibe correctamente la secuencia de paro, el receptor asume que se encuentra fuera de la frecuencia correcta y se declara un error de cuadro: *framing error*.

En la figura 2.25 se muestra una trama típica de una comunicación asíncrona. Se aprecia el bit de inicio, los bits de datos y el o los bits de paro.

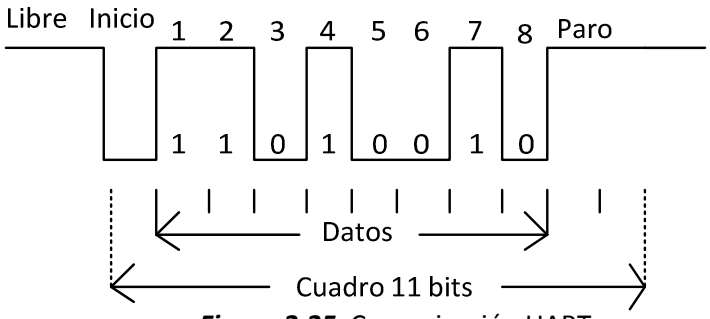


Figura 2.25. Comunicación UART.

En cualquier transmisión de datos existe la posibilidad de errores, en especial en un medio ruidoso como en los cables serie. Para detectar los errores se implementa un sistema simple en el cual se agrega un bit extra a la trama de datos. Este bit extra es llamado bit de paridad, con lo cual se validan los bits de información recibidos. El bit de paridad y los bits de información son calculados por la UART que enviará la información. La UART que recibe la información también calcula el bit de paridad de los bits de datos y lo compara con el bit que ha recibido. Si los bits de paridad coinciden, se considera una transmisión correcta, si no coinciden, el receptor considera que se han perdido bits de información y por lo tanto existe un error. Para que una transmisión sea exitosa, el transmisor y el receptor deben de establecer el mismo tipo de paridad.

Los tipos de paridad principal son: par e impar. En cualquier byte, es decir, ocho bits de información, existe un número par o impar de bits “1”. El bit de paridad se agrega al byte de información para que siempre tenga un número par de bits “1” (paridad par) ó un número impar de bits “1” (paridad impar).

Estándar RS-232

Es una interfaz de comunicación serie estándar que se utiliza desde 1960. Es utilizado para comunicaciones serie que sean menores a 20 metros, con una velocidad menor a 20 kilo bits por segundo y una capacitancia de carga en la línea menor a 2 500pF.

Para un transmisor en el estándar RS-232, un valor lógico alto está entre -5V y -15V, donde el valor nominal es -12V, mientras que un valor lógico bajo es entre +5V y +15V donde el valor típico es +12V. Para un receptor RS-232, un valor lógico alto está entre -3V y -15V, donde el valor nominal es -12V y un valor lógico bajo está entre 3V y 15V, donde el valor típico es de 12V.

El estándar RS-232 fue destinado principalmente para conectar dispositivos llamados equipo terminal de datos (DTE: *Data Terminal Equipment*) con el equipo de comunicación de datos (DCE: *Data Communication Equipment*). Comúnmente los DTE eran terminales y los DCE eran módems. En la actualidad un DTE suele ser una computadora personal y el DCE suele ser un equipo de medición, máquinas y en general dispositivos periféricos.

En la tabla 2.1 se muestran las señales que el estándar establece para dos tipos de conectores: el DB25 y el DB9. El más extendido y utilizado es el conector DB9, ya que ofrece un menor espacio y un menor costo.

Señal	Función	DB25	DB9
Tx	Transmitted Data	2	3
Rx	Received Data	3	2
RTS	Reques to Send	4	7
CTS	Clear to Send	5	8
DTR	Data Terminal Ready	20	4
DSR	Data Set Ready	6	6
DCD	Data Carrier Detect	8	1
RI	Ring Indicator	22	9
FG	Frame Ground (chassis)	1	-
SG	Signal Ground	7	5

Tabla 2.1. Señales del estándar RS-232.

La mayoría de las señales son para control; para formar una comunicación simple entre dos equipos, sólo se necesitan tres señales: Tx, Rx y SG.

2.7. Convertidor digital analógico

El convertidor digital analógico (DAC: *Digital to Analog Converter*) es un circuito diseñado para producir un voltaje analógico proporcional a las entradas binarias que se le aplican. El voltaje de salida varía en pasos. El paso más pequeño es controlado por el bit menos significativo (*LSB: Least Significant Bit*) de la entrada binaria. Existen dos métodos principales de conversión digital a analógica: amplificador sumador ponderado y la red de escalera R-2R.

Amplificador sumador ponderado

En la figura 2.26 se muestra un amplificador sumador con entradas lógicas ponderadas. Este circuito convierte un número binario, en este caso de 4 bits, a un voltaje equivalente de salida de

16 niveles discretos. Las entradas binarias controlan cuatro interruptores conectados a un voltaje de DC de 1V. Cada interruptor está marcado con su correspondiente peso lógico. Los interruptores se implementan con interruptores de estado sólido que controlan las líneas de entradas binarias.

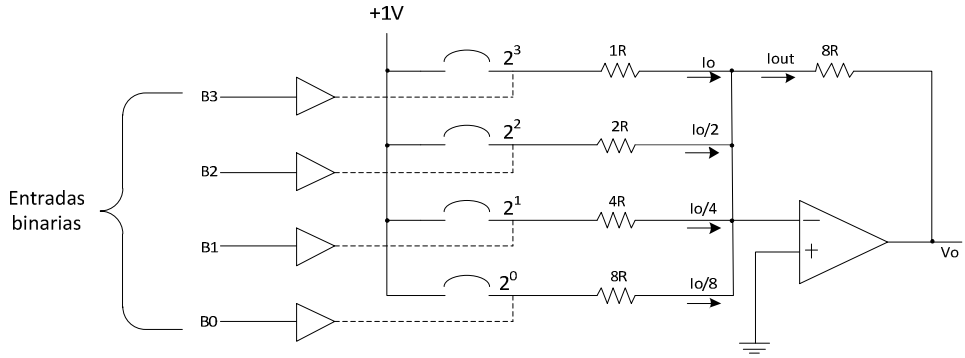


Figura 2.26. Amplificador sumador ponderado.

Cuando una línea binaria está en estado lógico alto, el interruptor de estado sólido es cerrado y cuando una línea binaria está en estado lógico bajo el interruptor es abierto.

Analizando este circuito se obtiene que la salida es, por ejemplo, -4V si el interruptor marcado con 2^2 es activado, debido a que produce un voltaje en la salida de V_0=-8R/2R=-4V. Si dos interruptores se activan, la salida total será la suma de la salida de los dos componentes individualmente, por ejemplo, 2^2 y 2^1, la salida será de -6V ya que V0=-8R/2R=-4V para 2^2 y V0=-8R/4R=-2V para 2^1. Si la salida se requiere positiva, se puede cambiar el valor de referencia de 1V a -1V.

Utilizando este modelo se pueden agregar cualquier número de entradas binarias, sin embargo, así como se incrementan las entradas binarias, se vuelve más difícil mantener la relación apropiada en los resistores. Por este motivo estos DAC son utilizados en un número relativamente pequeño de entradas binarias.

Red de escalera R-2R

Este convertidor utiliza un circuito denominado red de escalera para proporcionar el peso apropiado de las entradas binarias. La ventaja principal de esta red radica en que sólo se necesitan dos valores de resistencias, sin importar el número de entradas binarias a utilizar. Por este motivo, este DAC es más práctico en convertidores con un número grande de entradas lógicas.

En la figura 2.27 se muestra este arreglo de resistencias, donde los interruptores, también implementados con componentes de estado sólido, controlan las entradas binarias. Estos interruptores tienen dos posiciones: cuando la entrada binaria está en valor lógico “1” se activa el interruptor correspondiente, el cual conecta el voltaje de referencia, si el valor lógico es “0” el

interruptor conecta a la tierra del sistema. El bit más significativo (MSB: *Most Significant Bit*) controla el interruptor más cercano al amplificador, mientras que el LSB controla el interruptor más lejano del amplificador.

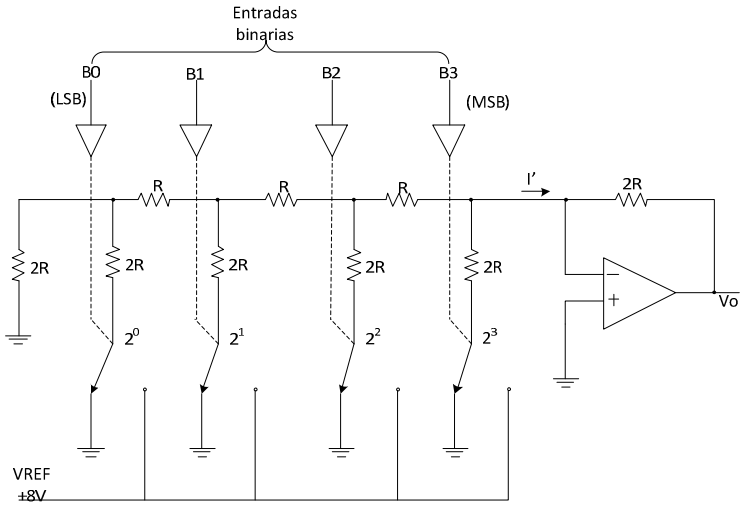


Figura 2.27. Arreglo R-2R.

Una forma de onda como la mostrada en la figura 2.28 es generada por esta configuración cuando se realiza una secuencia continua en las entradas binarias. Esta forma de onda es llamada generador de escalera. Este tipo de DAC tiene muchas aplicaciones, entre las que destacan los convertidores analógico digital.

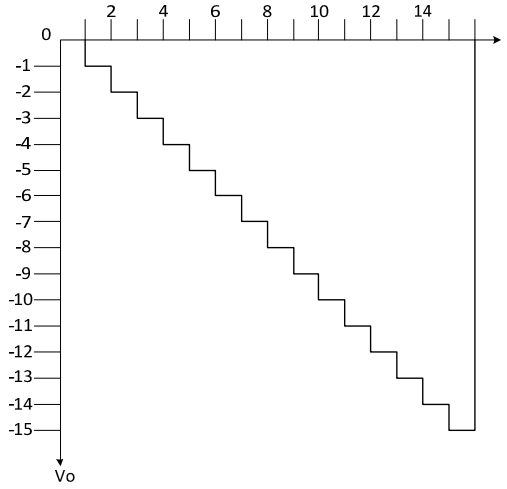


Figura 2.28. Forma de onda en escalera.

Algunas de las especificaciones más importantes para los DAC son: precisión absoluta, precisión relativa, resolución, linealidad, monotonicidad y el desvío (*offset)*, las cuales se describen a continuación:

- Precisión absoluta figura 2.29: es especificada en términos de la diferencia entre la salida analógica actual y la salida analógica esperada, esto para un valor binario específico. Se expresa en ± 1 LBS.

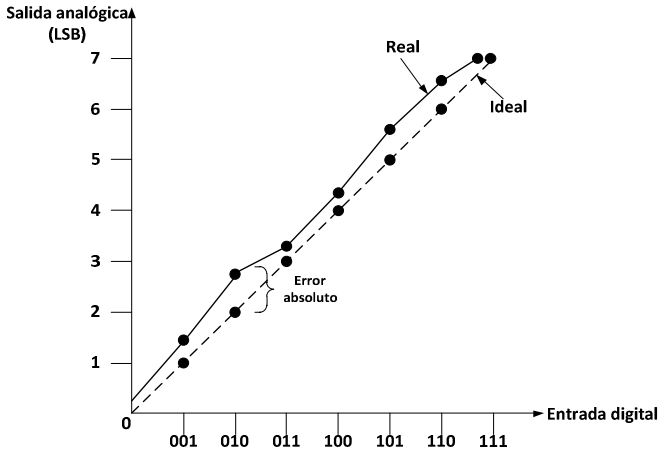


Figura 2.29. Precisión absoluta.

- Resolución: la resolución de un DAC es el incremento más pequeño que puede tener la salida analógica. La resolución básica de un DAC está dada por:

$$Resolución = \frac{V_0(FS)}{2^N} \tag{2.1}$$

donde:

V_0(FS) (FS: *Full Scale*) = voltaje de salida en escala completa y
N = número de bits

- Linealidad: es la medida de la desviación de la línea recta ideal y real de la conversión que va desde el cero hasta el punto de escala completa, figura 2.30. Se puede considerar una de las especificaciones más importantes de un DAC.

- Monotonicidad: en un DAC monótono, la salida siempre se incrementa cuando las entradas binarias se incrementan. En la figura 2.31 se muestra la salida de un DAC no monótono, junto con una salida ideal.

- Desvío: el desvío, o el *offset*, causa que toda la conversión se desplace a un nivel fijo de DC. Regularmente es ajustable poniendo las entradas binarias en nivel lógico "0" y ajustando o anulando el voltaje de salida a cero, figura 2.32.

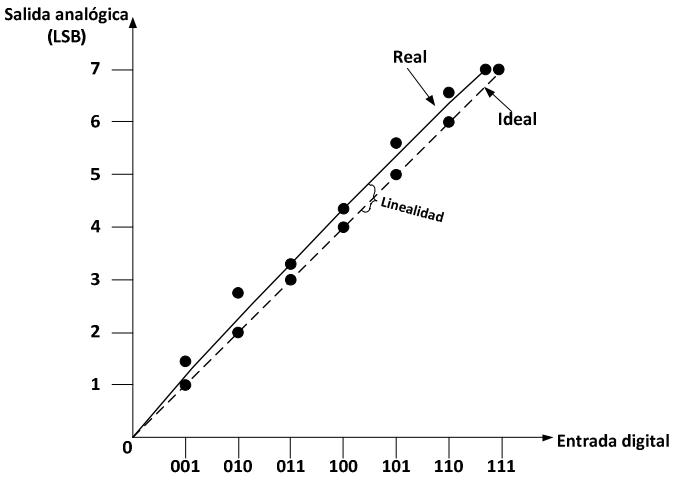


Figura 2.30. Linealidad.

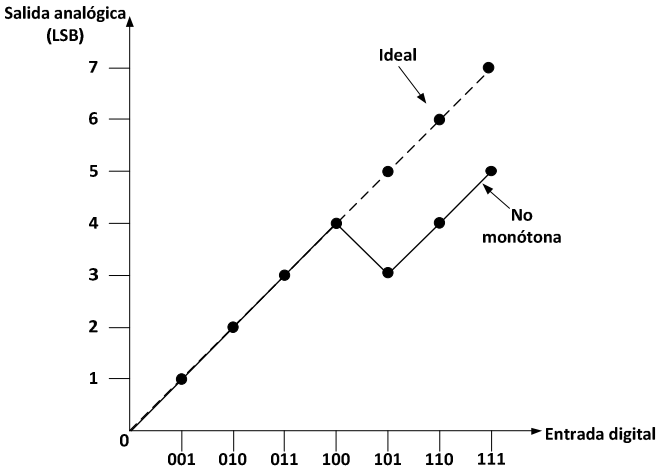


Figura 2.31. Monotonicidad.

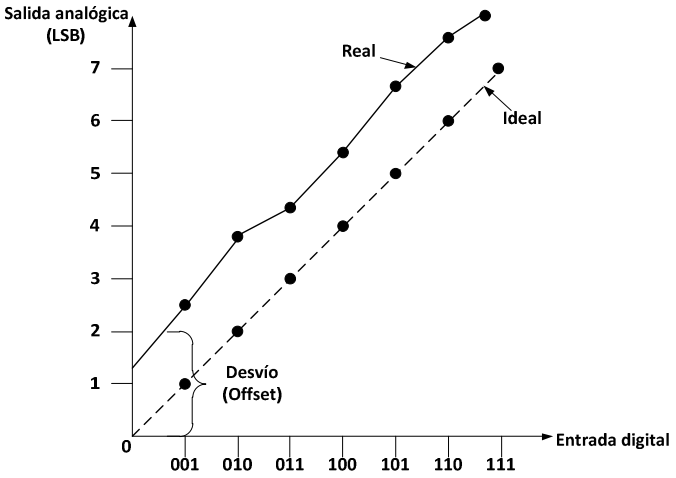


Figura 2.32. Desvío (Offset).

2.8. Convertidor analógico digital

El convertidor analógico digital (ADC: A*nalog to Digital Converter*) a diferencia del DAC, convierte una señal de entrada analógica a una salida digital. Los procesos de conversión en un ADC regularmente son más complejos que en los DAC.

Las especificaciones más importantes y que se deben considerar en un ADC son: resolución, error de cuantificación, linealidad, tiempo de adquisición y tiempo de conversión. A continuación se da una breve explicación de estas especificaciones.

- Resolución: indica el cambio mínimo que se puede detectar de la señal de entrada. Se puede calcular con la siguiente fórmula:

$$Resolución = \frac{V_{FS}}{2^N} \tag{2.2}$$

donde:

V_{FS} = voltaje a escala completa (FS: *Full Scale*) y
N = número de bits

- Linealidad: es la desviación en la transición de los valores de las salidas reales y la transición de los valores de la conversión ideal, figura 2.33.

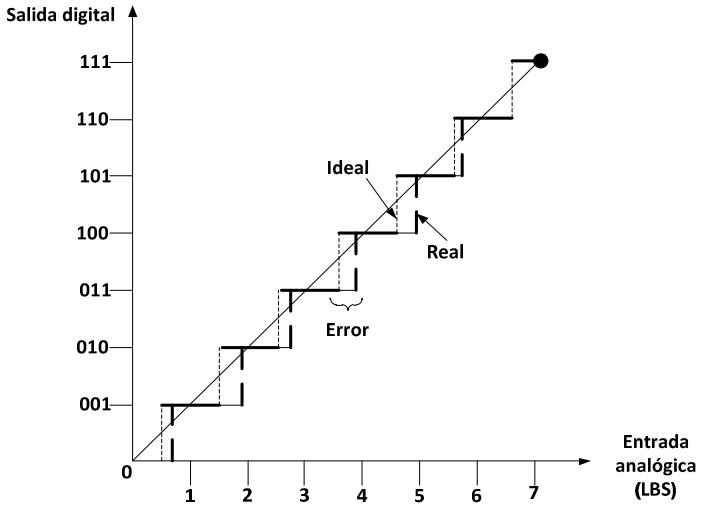


Figura 2.33. Linealidad.

- Error de cuantificación: es el error que se produce cuando el valor real de la muestra no equivale a ninguno de los valores disponibles en el convertidor. Su valor regularmente es de ± ½ LSB.

- Precisión: es la comparación entre la salida real del ADC y la salida esperada.

- Tiempo de adquisición: es el tiempo durante el cual el ADC mantiene el valor de la señal de entrada constante para ser convertida.

- Tiempo de conversión: es el tiempo desde que se aplica la señal de entrada hasta que la salida está disponible.

Existen diversos tipos de ADC, que si bien, tienen el mismo número de bits (lo cual implica la misma resolución), la arquitectura que se utiliza al implementar la conversión cambia y, por lo tanto, cambian otros parámetros como linealidad, tiempo de adquisición, tiempo de conversión etcétera. Las arquitecturas que destacan en la implementación de ADC son: rampa, doble rampa, aproximaciones sucesivas, sigma-delta y flash.

El tiempo de conversión en los ADC rampa y doble rampa es del orden de 300ms. Se utilizan principalmente en donde el tiempo de conversión no es prioridad, como en los multímetros portátiles. La precisión de estos ADC depende fuertemente de los generadores de rampas y de la estabilidad de la frecuencia de reloj.

La principal ventaja del ADC sigma delta es su alta resolución. Se pueden encontrar ADC de hasta 24 bits. Utiliza un proceso de sobre-muestreo, el cual disminuye su velocidad en la conversión, del orden de milisegundos. Debido a este mismo proceso de sobre-muestreo, su precisión es muy alta. Se utiliza principalmente en procesamiento digital de señales (DSP: *Digital Signal Process*).

Los ADC flash son los más rápidos ya que utiliza comparadores en su construcción y sólo están limitados por el tiempo de respuesta de los mismos. La principal desventaja es que es muy caro al implementar; se requieren 2^n-1 comparadores, donde *n* es el número de bits. Se utilizan en radares o aplicaciones de video.

El ADC por aproximaciones sucesivas tiene un periodo de conversión del orden de microsegundos y se abordará con más detalle ya que es de interés particular.

2.8.1. Convertidor por aproximaciones sucesivas

La arquitectura de aproximaciones sucesivas es la utilizada frecuentemente para resoluciones consideradas medias y altas, alrededor de 8 y 16 bits y con tasas de muestreo en el rango de 5 mega-muestras por segundo (Msps: *Mega Samples Per Second*). También proporciona un consumo de potencia reducido así como una construcción relativamente sencilla. Estas características hace ideal esta arquitectura en diversas aplicaciones, tales como instrumentos portátiles de bajo consumo, controles industriales y sistemas de adquisición de datos.

La figura 2.34 muestra el diagrama de bloques de la arquitectura del ADC de aproximaciones sucesivas.

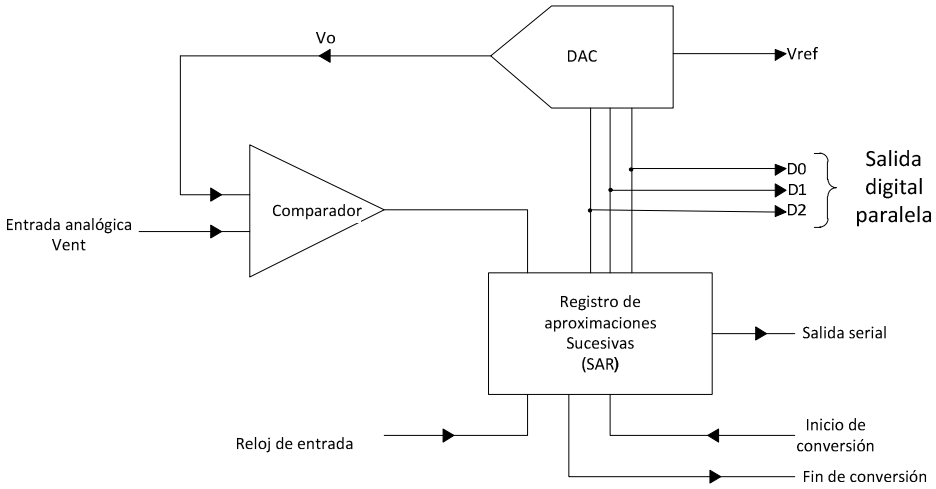


Figura 2.34. Arquitectura aproximaciones sucesivas.

Consta de tres bloques principales: un DAC, un comparador y un registro de aproximaciones sucesivas (SAR). Se requiere un mínimo de tres terminales de control: inicio de conversión, fin de conversión y una terminal externa de reloj que establece el tiempo para la conversión.

Funcionamiento

Se inicia el ciclo de conversión con la señal inicio de conversión. El SAR conecta la secuencia de números digitales a las entradas del DAC. La salida del DAC, Vo, se compara con el voltaje analógico de entrada, Vent. El comparador le dice al SAR cuando Vent es mayor o menor que la salida Vo. Para cada bit de salida se tiene que realizar una comparación, por ejemplo, si se tienen 3 bits de salida, se tienen que realizar 3 comparaciones.

Las comparaciones se realizan comenzando con el bit más significativo en un nivel lógico "1", haciendo que la salida del DAC, Vo, se encuentre a media escala de su capacidad, es decir, Vref/2. La comparación se realiza para determinar si Vent es mayor que o menor que Vref/2. Si Vent es mayor que Vref/2 la salida del comparador se establece en un nivel lógico alto y el bit más significativo permanece en "1". Si Vent es menor que Vref/2 la salida del comparador se establece en un nivel lógico bajo y el bit más significativo se coloca a nivel "0". El control lógico del SAR se coloca ahora en el siguiente bit hacia abajo para realizar otra comparación siguiendo el mismo criterio. Las comparaciones se realizan hasta alcanzar el bit menos significativo. Al finalizar la comparación el SAR envía la señal que finalizó la conversión.

Tiempo de conversión

Se necesita un pulso de reloj para que el registro SAR compare cada bit. No obstante casi siempre se requiere un pulso adicional para restablecer el registro antes de llevar a cabo la conversión. Por lo tanto la relación es:

$$\boldsymbol{Tc = T(N + 1)} \qquad (2.3)$$

donde:

Tc = periodo de conversión,
T = periodo de la señal de reloj y
N = número de bits.

En la figura 2.35 se muestra la operación de la arquitectura aproximaciones sucesivas con tres bits. Se observa la señal de reloj, la de inicio de conversión y fin de conversión. Se nota claramente que se necesita un pulso por cada comparación así como la lógica que se deben tomar en cada comparación.

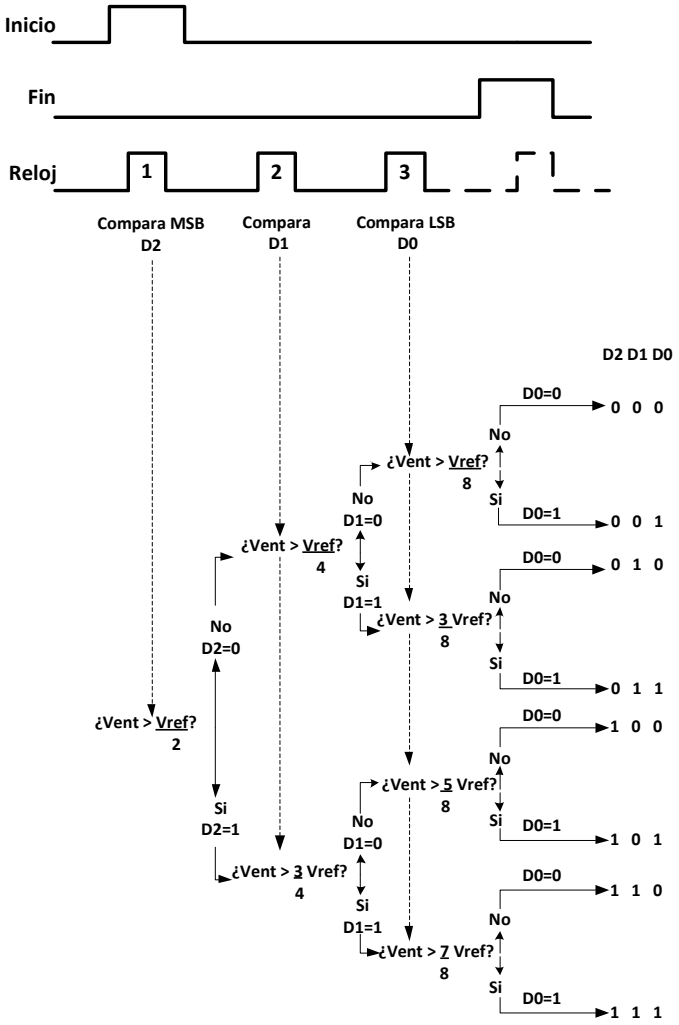


Figura 2.35. Funcionamiento aproximaciones sucesivas.

2.9. Display de cristal líquido

Un display LCD es una interfaz formada principalmente por una pantalla de cristal líquido (LCD: *Liquid Crystal Display*), organizada en una matriz de puntos, y un micro-controlador, que regularmente se encuentra incorporado en la misma placa de la pantalla LCD. El micro-controlador

es el encargado de gestionar los detalles finos del display LCD, como polarizar los puntos de la pantalla, generar los caracteres, desplazar el cursor, etcétera. El usuario sólo envía una serie de comandos o instrucciones para realizar funciones de alto nivel, como por ejemplo limpiar pantalla, enviar carácter, ubicación del cursor, entre otras.

Existen display LCD de distintos formatos: 2x8, 1x16, 2x16, 4x20, donde el primer número es el número de líneas que tiene la pantalla y el segundo número es el número de columnas que contiene la pantalla. El de interés particular es de 2 filas y 16 columnas, es decir, 2x16, que se muestra en la figura 2.36.

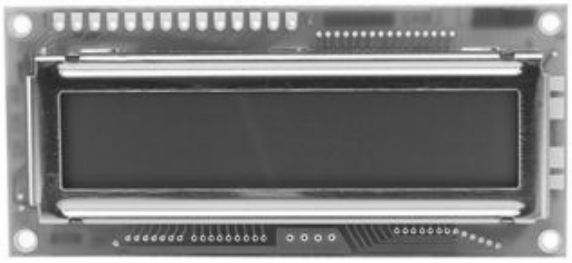


Figura 2.36. Display LCD de 2x16.

Para interactuar con el display LCD, comúnmente se cuenta con una comunicación paralela de 8 bits aunque, de estos 8 bits, se pueden utilizar solamente 4 bits. El display también cuenta con tres señales de control: habilitación (*enable*), escritura/lectura (*write/read*) y selección de registro (*register select*). En la tabla 2.2 se muestran estas señales con la correspondiente terminal del display LCD.

Terminal	Nombre	Función
1	GND	Tierra
2	VCC	Alimentación
3	Vo	Ajuste del contraste
4	RS	Selecciona entre una instrucción (“0”) o un dato (“1”)
5	R/W	Selecciona entre leer (“1”) y escribir (“0”) en el display LCD
6	E	Habilitación del display LCD
7	DB0	Terminal 0 para datos
8	DB1	Terminal 1 para datos
9	DB2	Terminal 2 para datos
10	DB3	Terminal 3 para datos
11	DB4	Terminal 4 para datos
12	DB5	Terminal 5 para datos
13	DB6	Terminal 6 para datos
14	DB7	Terminal 7 para datos
15	LED1	Ánodo para LED
16	LED2	Cátodo para LED

Tabla 2.2. Terminales del display LCD 2x16.

La alimentación del display LCD debe ser de 5V y las entradas de datos y de control deben ser también de 5V.

El display LCD contiene tres regiones de memoria: DD RAM, CG ROM y CG RAM que a continuación se definen y describen.

DD RAM

La memoria RAM de datos del display (DD RAM: *Display Data RAM*) almacena el código del carácter que será desplegado. El código del carácter almacenado en la DD RAM hace referencia al mapa de bits que se encuentra en la CG ROM o en la CG RAM. Para poder desplegar un carácter, primero se le indica a la DD RAM la localidad en donde se quiere visualizar. Físicamente la DD RAM del display LCD 2x16, figura 2.37, tiene 80 localidades con 40 caracteres disponibles por línea (los números están en base hexadecimal). La primera línea visible por defecto, tiene una localidad de inicio en 0x00 y una de término en la 0x0F, mientras que la segunda línea visible por defecto, tiene una localidad de inicio en 0x40 y una de término en 0x4F. Las demás posiciones se pueden visualizar realizando una función de desplazamiento del cursor.

	Localidades visibles																Localidades no visibles
1	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10 . . . 27
2	40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F	50 . . . 67
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17 . . . 40

Figura 2.37. Localidades de la DD RAM.

CG ROM

La ROM de generador de carácter (CG ROM: *Character Generator ROM*) contiene el mapa de bits de cada carácter predefinido que el display LCD puede desplegar. El código almacenado en la DD RAM del carácter hace referencia a una localidad de la CG ROM. Por ejemplo, si en la DD RAM se ha almacenado un código 0x53, el display LCD desplegará la letra “S”, figura 2.38, debido a que el *nibble* (conjunto de 4 bits) superior es DB[7:4]=0101 y el *nibble* inferior es DB[3:0]=0011. La figura 2.38 muestra todos los caracteres predeterminados que puede mostrar el LCD. Se observan que los caracteres se encuentran en código ASCII.

CG RAM

La RAM de generador de carácter (CG RAM: *Character Generator RAM*) proporciona ocho localidades para crear caracteres personalizados. Cada carácter consiste en 5 puntos y 8 líneas.

Figura 2.38. Mapa de caracteres del display LCD.

Para hacer uso del display LCD se tiene que realizar dos procesos importantes: inicializarlo y configurarlo. En el proceso de inicialización, principalmente, se configura el número de bits para interactuar con el display, ya sea 8 bits o 4 bits. En el proceso de configuración se seleccionan las características que tendrá el display, por ejemplo: incrementar o disminuir la localidad de la DD RAM en cada escritura, desplazar el cursor en cada escritura, mostrar u ocultar el cursor, hacer parpadear el cursor, etcétera. En este proceso también se inicializa la localidad de la DD RAM para empezar a escribir los caracteres que se desean mostrar.

Es de gran importancia inicializar la localidad de la DD RAM antes de escribir caracteres en el display, posteriormente se envía el dato a escribir, dependiendo el número de bits que se ha elegido para la comunicación. En el caso de una interface de 4 bits, se tiene que enviar el dato en dos partes, primero se envía el *nibble* superior y después se envía el *nibble* inferior. Para una interface de 8 bits, se envía el dato completo en una sola parte. Un diagrama de tiempos típico de las señales del display LCD se muestra en la figura 2.39.

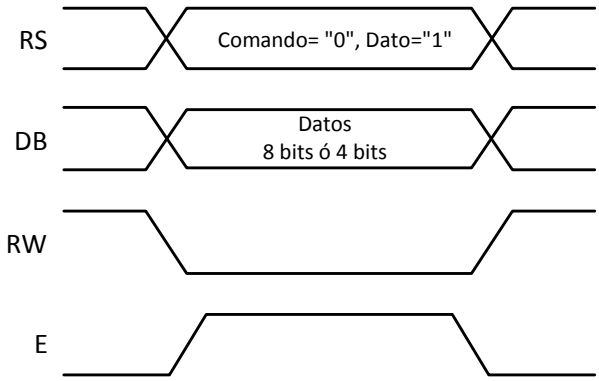


Figura 2.39. Diagrama típico para el display LCD.

Las funciones que destacan en el manejo del display LCD se encuentran las siguientes:

- Limpiar display (*Clear display*): borra los caracteres que se han escrito en la pantalla del LCD.

- Regresar cursos a inicio (*Retur Cursor Home*): coloca el cursor en la posición 0x00 de la DD RAM, la cual es la posición de inicio.

- Establecer modo de entrada (*Entry Mode Set*): con esta función se especifica si el cursor se incrementa o disminuye en cada escritura y si la pantalla LCD se desplaza hacia la izquierda o hacia la derecha para mostrar las localidades ocultas.

- Encender/apagar el display (*Display On/Off*): esta función enciende o apaga el display y configura el cursor (mostrar, ocultar, hacer que parpadee).

- Establecer función: con esta función se configura, principalmente, el número de bits para la comunicación: 8 bits o 4 bits.

- Inicializar localidad de la DD RAM (*Set DD RAM Address*): se da la localidad de inicio de la DD RAM; en donde se quiera empezar a desplegar los caracteres.

- Escribir datos en la DD RAM (*Write Data to DD RAM*): se envían los bits del carácter correspondiente para ser desplegado en la localidad establecida del LCD.

2.10. Diodo emisor de luz e interruptores

El incremento de pantallas digitales en componentes de la vida diaria como calculadoras, relojes, televisiones, etcétera, ha provocado un amplio interés en estructuras que emiten luz cuando se polarizan de forma apropiada. Las dos estructuras emisoras de luz principales son: el diodo emisor

de luz (LED: *Light Emitting Diode*) y la pantalla LCD. Se describirá el LED ya que es de interés particular.

Diodo emisor de luz

El LED es un dispositivo de unión semiconductora *p-n*, capaz de emitir luz visible cuando se polariza adecuadamente. En una unión *p-n* con polarización directa, existe dentro de la estructura, principalmente cerca de la unión, una recombinación de huecos y electrones. Esta recombinación requiere que la energía del electrón sea transferida. Cierta cantidad de ésta energía es transferida en forma de calor y otra cantidad en forma de fotones. En uniones de silicio y germanio mayormente se transfiere energía en forma de calor y minoritariamente en forma de fotones. En uniones de fósforo de arsenurio de galio (GaAsP) o en fosfuro de galio (GaP) mayormente la energía se transfiere en forma de fotones, suficientes para crear una fuente de luz altamente visible.

En la figura 2.40 se muestra el símbolo que representa un LED con polarización directa. Se observan dos terminales, una llamada ánodo (A) y otra llamada cátodo (k), las cuales se pueden polarizar de forma directa o de forma inversa. Para polarizar un LED en forma directa, la terminal ánodo debe tener un voltaje mayor que la terminal cátodo. Para una polarización inversa la terminal cátodo debe tener un voltaje mayor que la terminal ánodo. El valor del voltaje depende de los materiales de la unión *p-n*.

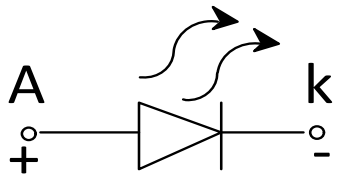


Figura 2.40. Símbolo de un LED.

Los LEDs, generalmente se clasifican en LEDs de baja potencia y LEDs de alta potencia. Los LEDs de alta potencia se utilizan principalmente en aplicaciones de iluminación, mientras que los LEDs de baja potencia se utilizan principalmente como indicadores. Para el presente trabajo se utilizarán LEDs de baja potencia de encapsulado de montaje superficial, mostrado en la figura 2.41.

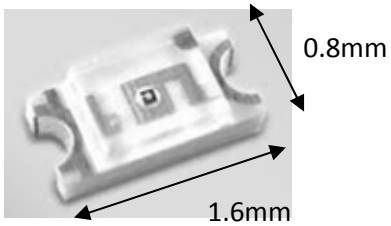


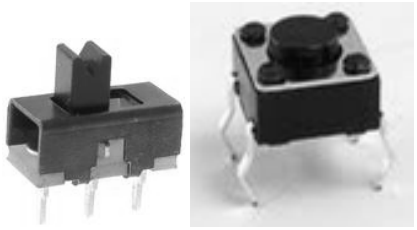
Figura2.41. LED de montaje superficial.

Uno de los componentes básicos y que se encuentran en casi todos los sistemas electrónicos son los interruptores o *switchs*.

Interruptores

Se define a un interruptor como un dispositivo capaz de desviar o interrumpir el paso de corriente en un circuito eléctrico. Los tipos más comunes de interruptores son los mecánicos, aunque también existen los electromecánicos y electrónicos. Cualquier interruptor se puede encontrar en dos posibles estados: abierto, donde la corriente eléctrica está interrumpida, y cerrado, donde la corriente eléctrica fluye a través del interruptor. El mecanismo para cambiar entre estos dos estados puede ser continuo, mediante una palanca que cambie de posición entre abierto (OFF) y cerrado (ON) ó de contacto momentáneo donde, comúnmente, se presiona para cerrar (ON) y se suelta para abrir (OFF).

A los interruptores con mecanismo continuo se les conoce simplemente como interruptores o *switchs*, a los de mecanismo momentáneo se les conoce comúnmente como *push button*. En el inciso *a* de la figura 2.42 se muestra un ejemplo del *switch* y en el inciso *b* se muestra un tipo *push button*.



a) b)

Figura 2.42. Tipos de Interruptores.

De forma esquemática los interruptores se pueden representar como se muestra en la figura 2.43, en el inciso *a* el *switch* y en el inciso *b* el *push button*.

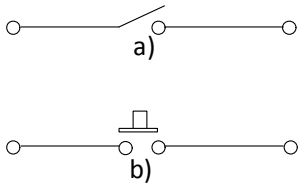


Figura 2.43. Esquema de un interruptor.

Uno de los efectos indeseables en los interruptores es el llamado efecto rebote, figura 2.44. Este efecto se debe a que cuando se acciona el interruptor, sus componentes, regularmente láminas, necesitan un periodo de estabilización. Antes de la estabilización, las láminas presentan una serie de rebotes entre sí, provocando múltiples cierres y aperturas. Cuando se conecta un interruptor a un microprocesador, debido a la gran velocidad del microprocesador, estos rebotes pueden ser detectados, provocando con ello la interpretación de entradas falsas e indeseadas al sistema.

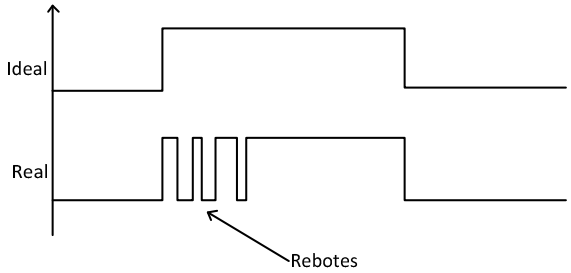


Figura 2.44. Efecto rebote.

Existen principalmente dos métodos para eliminar el efecto rebote indeseado: el primer método es con una aplicación de software, que detecte el primer cambio en la señal de entrada y realizar un retardo; el segundo método es por medio de hardware, donde se incorporan al interruptor componentes extras que van desde resistencias y capacitores, figura 2.45, hasta circuitos integrados. En la mayoría de las aplicaciones se utiliza el método por software, ya que no se necesitan componentes extras.

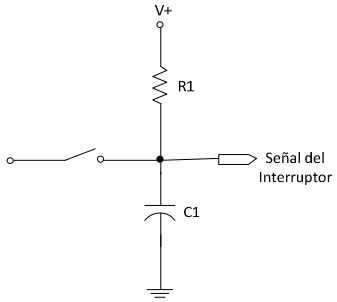


Figura 2.45. Anti-rebote por hardware.

2.11. Memoria SDRAM

La memoria de acceso aleatorio (RAM: *Random Access Memory*) es un dispositivo para el almacenamiento de datos. Su principal característica es que pierde los datos cuando se retira la fuente de alimentación de la memoria, y por ésta característica, se le conoce como memoria volátil. En este tipo de memorias, se accede a la información de manera rápida y directa. Se utiliza para cargar datos temporales y hacer más rápidas las aplicaciones, por ejemplo, en una computadora personal (PC: *Personal Computer*) algunos programas se almacenan en memoria RAM para hacer más rápido el uso de dichos programas.

Existen diversos tipos de memoria RAM, entre las que destacan la RAM dinámica síncrona (SDRAM: *Synchronous Dinamic Random Access Memory*) y la RAM estática, SRAM.

Los datos en la memoria SDRAM necesitan ser refrescado cada determinado tiempo. Su construcción se basa principalmente en capacitores. Dentro de este tipo de memorias se encuentran las memorias de doble tasa de transferencia de datos (DDR: *Double Data Rate*) en sus versiones DDR1, DDR2 y DDR3. La ventaja principal de las memorias DDR es la habilidad de

manejar datos tanto en el flanco de subida como en el flanco de bajada de la señal de reloj, duplicando la frecuencia del reloj. La DDR1 tiene una frecuencia de hasta 400 MHz con un voltaje de alimentación de 2.5V. La DDR2 tiene una frecuencia de 400 MHz a 800 MHz con un voltaje de alimentación de 1.8V. La versión DDR3 tiene una frecuencia de operación de 800 MHz hasta 1 GHz con una polarización de 1.5V.

En una memoria SRAM los datos almacenados no necesitan ser refrescado ya que su construcción se basa en *latches*. Debido a su construcción, es más cara y lenta que la SDRAM. Se utiliza en varias aplicaciones como en microcontroladores, sistemas embebidos, PCs, entre otros dispositivos.

En este capítulo se describieron los conceptos teóricos básicos para el desarrollo de la presente tesis. En el siguiente capítulo se describe ampliamente el desarrollo del sistema de monitoreo y control.

Capítulo III
Diseño del sistema de monitoreo y control

En este capítulo se presentan los objetivos que se pretenden alcanzar con el presente trabajo de tesis, así como la plataforma para el desarrollo de dichos objetivos. Se describen también, las metodologías que se utilizaron y se muestra la integración de la metodología seleccionada que cumple con los objetivos especificados.

3.1. Objetivos

Los objetivos planteados para este trabajo de tesis son los siguientes:

- Diseñar e implementar un sistema de monitoreo y control basado en un FPGA.
- Diseñar e implementar la tarjeta de circuito impreso del sistema de monitoreo y control propuesto.
- Obtener una tarjeta funcional donde se encuentre integrado el sistema de monitoreo y control con la posibilidad de mejorar y expandir las capacidades del mismo.

El sistema de monitoreo y control estará basado alrededor de un FPGA y deberá contener los siguientes elementos, figura 3.1:

- Interfaz de configuración: se utilizará para la configuración del FPGA. Desde una computadora se configurará el FPGA con el circuito digital diseñado.
- Comunicación RS-232: se implementará el protocolo RS-232 que comunicará el FPGA con una computadora para el intercambio de información.
- Convertidor analógico digital (ADC): con este módulo se monitoreará parámetros de interés del sistema como voltajes, corrientes y/o temperaturas.
- Convertidor digital analógico (DAC): realizará acciones de control desde el sistema.
- Display LCD: en este display se mostrarán datos generales sobre los procesos que se estén realizando en el sistema.

- Entradas y salidas de propósito general: estas entradas y salidas servirán para verificar el funcionamiento correcto de los sistemas, así como indicadores del estado de cada proceso, detener procesos, reiniciar procesos, habilitar o deshabilitar procesos entre otros.

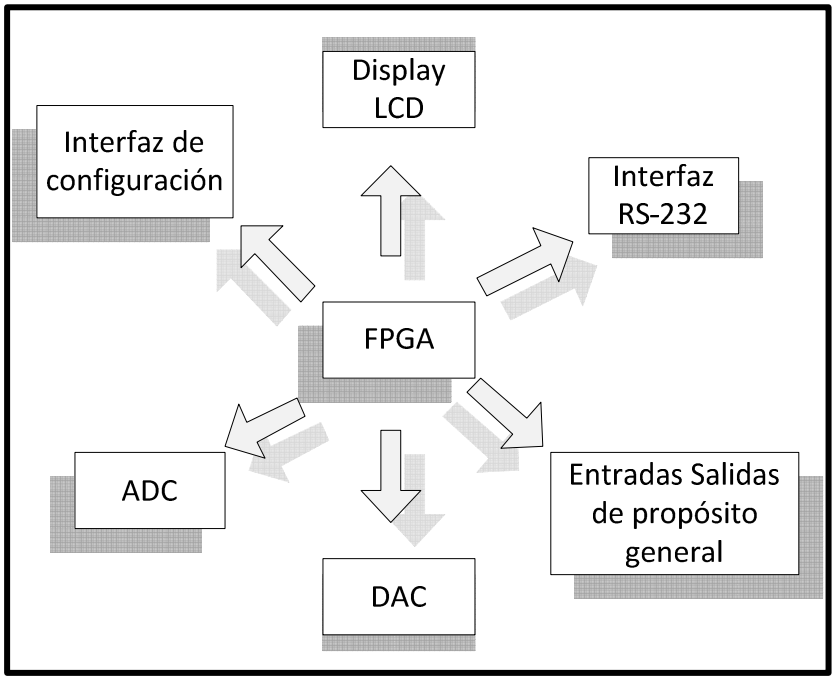


Figura 3.1. Sistema de monitoreo y control.

El sistema de monitoreo y control se basará en la comunicación RS-232. Con dicha comunicación se enviarán comandos desde la computadora hacia el FPGA, para realizar los procesos solicitados, así como enviar datos desde el FPGA hacia la computadora. Los procesos que se podrán solicitar son: adquisición de señales por medio del convertidor analógico digital, envío de voltajes analógicos por medio del convertidor digital analógico, envío de datos digitalizados desde el FPGA hacia la computadora, activación de salidas digitales con base en los datos enviados desde la computadora y finalmente una verificación del correcto funcionamiento del sistema. También contará con indicadores para dar a conocer al usuario qué proceso se está realizando y si se encuentra disponible para recibir algún comando. Estos indicadores se realizarán por medio del display LCD y por medio de salidas de propósito general, en este caso, en un LED.

3.2. Descripción de la tarjeta de desarrollo SPARTAN 3E 1600E MicroBlaze

Para realizar el sistema de monitoreo y control, se seleccionó como base la tarjeta de desarrollo SPARTAN 3E 1600E MicroBlaze, que manufactura la empresa DIGILENT, figura 3.2. Esta tarjeta está

basada en una plataforma muy conveniente para el desarrollo de sistemas embebidos, debido a las características con las que cuenta.

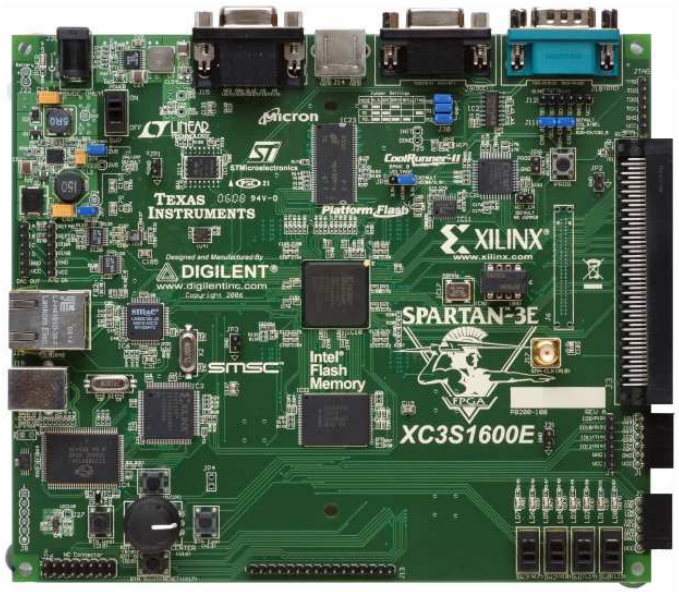


Figura 3.2. Spartan 3E 1600E MicroBlaze.

3.2.1. Características

La tarjeta SPARTAN 3E cuenta con los siguientes componentes:

Dispositivos lógicos programables:

- Un FPGA XC3S1600E de la familia SPARTAN 3E, el de mayor capacidad dentro de esta familia.
- Un CPLD XC2C64A de la marca Xilinx de 64 macroceldas.

Memorias de almacenamiento:

- Dos memorias XC04S Platform Flash PROM (PROM: *Programmable Read Only Memory*) de la marca Xilinx, de 4Mbit.
- Una memoria DDR SDRAM de 512 Mbits
- Una memoria NOR Intel StrataFlash PROM de 128 Mbits
- Una memoria Flash SPI de 16 Mbits
- Una memoria EEPROM SHA-1 para protección del *bitstream*

Periféricos:

- Conector PS/2 para teclado ó ratón
- Puerto VGA
- Interfaz Ethernet 10/100
- Dos puertos para interfaz RS-232 con conector DB9 cada uno
- DAC con interfaz SPI

- ADC y amplificador de ganancia programable (PGA: *Programmable Gain Amplifier*) con interfaz SPI
- Cristal de 50 MHz

Componentes de propósito general:

- Display LCD de 2x16
- Perilla giratoria con *push button*
- Ocho LEDs discretos
- Cuatro interruptores
- Cuatro *push button*

Conectores:

- Tres conectores de 6 terminales cada uno
- Conector Hirose FX2 con 40 terminales de entrada/salida del FPGA disponibles
- Dos conectores para cristales adicionales

Configuración:

- Interfaz USB
- Estándar JTAG

Ya que se conocen las características que ofrece la tarjeta de desarrollo, se seleccionarán los componentes que son de interés particular para el desarrollo del sistema de monitoreo y control.

3.2.2. Selección de periféricos

Ya que se pretende diseñar e implementar un sistema de monitoreo y control, algunos de los componentes de la SPARTAN 3E no se utilizarán. En la tabla 3.1 se enlistan los componentes seleccionados que están orientados al monitoreo y al control.

Al tener definido los componentes que se utilizarán, se procederá a trabajar con ellos. En esta fase se comenzó estudiando el FPGA, familiarizándose con el ambiente de trabajo, es decir, con el IDE de Xilinx. El IDE que se utilizó es el *ISE Design Suit 13.2*.

Una vez adquiridos los conocimientos básicos sobre HDLs, en específico con el lenguaje VHDL, se inició el diseño del sistema de monitoreo y control con este lenguaje. Dando así origen a la primera aproximación del sistema a desarrollar.

Componente	Descripción
FPGA 1600E	Es el componente principal del sistema
Memoria DDR SDRAM de 512 Mbits	Se utilizará para almacenar datos de manera temporal

Tabla 3.1. Selección de componentes. (Continúa)

Componente	Descripción
Puertos de interfaz RS-232	Se utilizarán los dos puertos para enviar y recibir información
DAC con interfaz SPI	Parte fundamental del sistema de control
ADC y PGA con interfaz SPI	Parte fundamental del sistema de monitoreo
Cristal 50 MHz	Se utilizará para sincronizar procesos diseñados en el FPGA
Display LCD de 2x16	Con este dispositivo se mostrarán mensajes de los procesos del sistema
Perilla giratoria con *push button*	Se utilizará como posible entrada de control al sistema
LEDs discretos	Componentes utilizados como indicadores y simuladores de la activación/desactivación de componentes externos
Interruptores	Se utilizarán como posibles entradas para iniciar o cambiar eventos
Push button	Utilizados como posibles entradas para el inicio y re-inicio de procesos
Conectores de 6 terminales	Se utilizarán para conectar cualquier otra salida/entrada adicional que se requiera
Conector Hirose FX2	Se utilizará para conectar salidas/entradas extras
Estándar JTAG	Se utilizará para configurar el FPGA

Tabla 3.1. Selección de componentes.

3.3. Primera aproximación al sistema de monitoreo y control

Con base en los conocimientos adquiridos sobre VHDL, se procedió a diseñar las descripciones que controlarán los componentes seleccionados. Dichos componentes fueron: los interruptores, los LEDs, la perilla giratoria, el amplificador, los convertidores ADC y DAC y el display LCD.

3.3.1. Interruptores, diodo emisor de luz y perilla giratoria

Las conexiones entre los interruptores (tanto *switch* como *push button*), los LEDs y la perilla giratoria con el FPGA se muestran en la figura 3.3. También se muestra el número de la terminal del FPGA en el cual está conectado cada componente. Las flechas indican el tipo de terminal que es: si la flecha sale del FPGA, la terminal es de salida; si la flecha entra al FPGA, la terminal es de entrada.

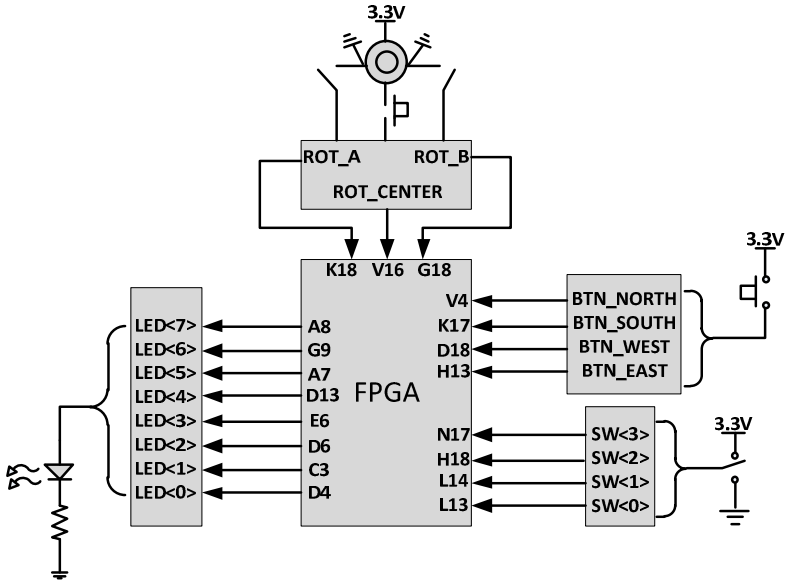


Figura 3.3. Interruptores, LEDs, perilla giratoria y FPGA.

La primera descripción que se realizó para el control de estos dispositivos fue leer el estado tanto de los interruptores como de la perilla giratoria y mostrarlo en los LEDs. Al tener los resultados esperados con esta primera descripción, se procedió a realizar varias pruebas, en donde, por ejemplo, se leía el estado de los interruptores y se realizaba una secuencia de encendido de los LEDs. En todas las pruebas el resultado fue correcto y no hubo mayor complicación.

3.3.2. Amplificador y convertidor analógico digital

La conexión entre el amplificador, el ADC y el FPGA se muestra en la figura 3.4. El amplificador cuenta con las cuatro líneas SPI, dos entradas analógicas y una terminal extra de habilitación por hardware (SHDN_AMP); mientras que el ADC cuenta sólo con 2 líneas SPI (un SPI modificado), una terminal que da inicio a la conversión (CONV) y con dos entradas analógicas.

El circuito integrado del amplificador es el LTC6912-1. Este circuito integrado contiene dos amplificadores, en configuración inversora, con control de ganancia independiente para cada uno. El rango de polarización es de 2.5V a 10.5V. Las ganancias pueden ser de 0, -1, -2, -5, -10, -20, -50 y -100.

El convertidor analógico digital utilizado es el LTC1407A-1, de 14 bits de resolución en complementa a dos. Contiene dos canales diferenciales con rango de entrada de ±1.25V cada uno, que se muestrean simultáneamente cuando se inicia la conversión. El resultado de la conversión se presenta en 32 pulsos de reloj. La frecuencia máxima de muestreo es de 1.5MHz por canal. El voltaje máximo de alimentación es de 4V. Una característica importante de este convertidor es que el resultado de la conversión está disponible hasta que se inicie otra conversión, es decir, el resultado tiene un retardo, ó latencia, de una conversión.

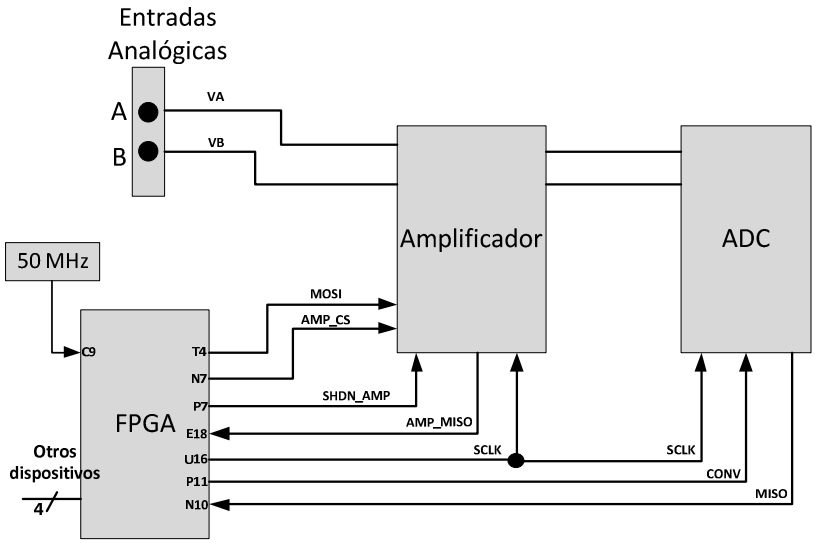


Figura 3.4. Amplificador, ADC y FPGA.

Debido a que algunas líneas del SPI son compartidas con otros dispositivos dentro de la tarjeta de desarrollo, éstos se tienen que desactivar para evitar interferencias en la comunicación cuando se interactúa con el amplificador ó el ADC. En la tabla 3.2 se muestran los dispositivos que comparten las líneas SPI. Dependiendo con qué dispositivo quiere uno comunicarse, son los dispositivos que se tienen que deshabilitar. Por ejemplo, si se quiere comunicar con el amplificador, se tienen que deshabilitar las memorias, el ADC y el DAC.

Señal	Terminal FPGA	Dispositivo	Valor para deshabilitar
SPI_SS_B	U3	Memoria SPI Flash	1
SF_CE0	D16	Memoria StrataFlash PROM	1
FPGA_INIT_B	T3	Memoria Platform PROM	1
AMP_CS	N7	Amplificador programable	1
DAC_CS	N8	Convertidor digital analógico	1
CONV	P11	Convertidor analógico digital	0

Tabla 3.2. Dispositivos conectados al SPI.

La señal de reloj, para sincronizar los procesos, es una señal de 50 MHz con un periodo de 20 ns, proporcionada por el cristal que viene incluido en la tarjeta de desarrollo y se encuentra en la terminal de entrada del FPGA con número C9.

Conversión analógica a digital

La conexión a detalle entre los amplificadores y los convertidores se muestra en la figura 3.5.

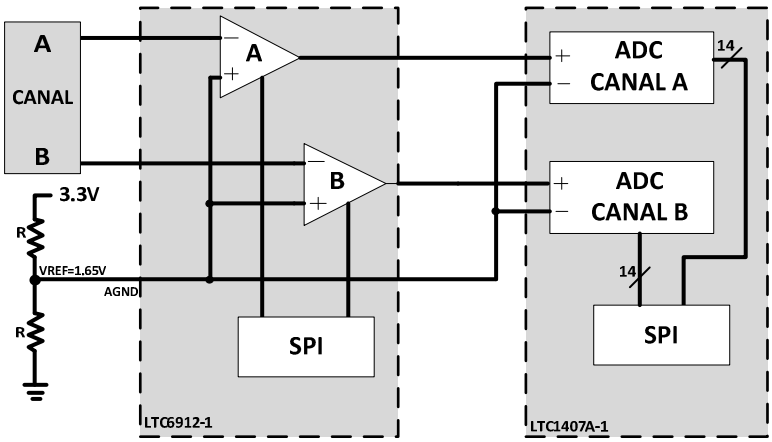


Figura 3.5. Conexión entre los amplificadores y los ADC.

En el caso del amplificador, las señales de entrada se aplican en la terminal inversora, mientras que en la no inversora se tiene como entrada un voltaje constante de 1.65V. Una característica importante de estos amplificadores es que el sistema de referencia para las entradas y las salidas es de 1.65V, por lo que la relación de entrada/salida es:

$$V_{oamp} = GANANCIA\ (V_{-} - V_{+}) + 1.65V \qquad (3.1)$$

donde:

Voamp = voltaje de salida del amplificador
GANANCIA = ganancia del amplificador
V+ = terminal de entrada no inversora
V- = terminal de entrada inversora

En la ecuación 3.1 la ganancia ya se considera como negativa y por lo tanto los voltajes de entrada se invierten. Es importante mencionar que la salida del amplificador, Voamp, varía entre 0V y 3.3V independientemente de la ganancia aplicada. Los valores de la ganancia se muestran en la tabla 3.3. En esta misma tabla se muestra también los voltajes mínimo y máximo de las entradas, VA ó VB, para que sea válida la ganancia seleccionada.

Ganancia	Entrada mínima V	Entrada máxima V	Entrada Vp-p
0	0	0	0
-1	0.15	3.15	3
-2	0.9	2.4	1.5
-5	1.35	1.95	0.6
-10	1.5	1.8	0.3
-20	1.575	1.725	0.15
-50	1.62	1.68	0.06
-100	1.635	1.665	0.03

Tabla 3.3. Ganancias y amplitudes de entrada.

En la tabla 3.4 se muestran los parámetros medidos y calculados de los amplificadores utilizados. La ganancia seleccionada es la unitaria, por facilidad en el manejo del voltaje de entrada. En la primer columna aparecen los voltajes de las terminales no inversora, el cual está fijo a 1.65V, e inversora, que son las entradas VA y VB que están conectadas al mismo voltaje. En la segunda columna se encuentra la resta entre voltajes V- y V+ (voltaje diferencial). En la tercer columna se encuentra Voamp de la ecuación 3.1 para ambos amplificadores, A y B. En la cuarta columna se encuentra la salida del amplificador en donde se observa el efecto de la ganancia. Por último, en la quinta columna se encuentra la ganancia resultado de la salida con ganancia y la entrada diferencial. Cabe señalar que esta tabla se obtuvo midiendo los voltajes de entrada y de salida en las terminales del circuito integrado.

Voltajes de entrada		**Entrada diferencial**	**Voamp**		**Salida con ganancia**		**Ganancia**	
V+=1.65V	**V-=VA=VB**	**V- - V+**	**VoA**	**VoB**	**A**	**B**	**A**	**B**
1.65	0.104	-1.546	3.19	3.19	1.54	1.54	-0.9961	-0.9961
1.65	0.205	-1.445	3.1	3.1	1.45	1.448	-1.0035	-1.0021
1.65	0.307	-1.343	2.998	3	1.348	1.347	-1.0037	-1.003
1.65	0.408	-1.242	2.896	2.9	1.246	1.245	-1.0032	-1.0024
1.65	0.509	-1.141	2.795	2.79	1.145	1.143	-1.0035	-1.0018
1.65	0.605	-1.045	2.7	2.7	1.05	1.048	-1.0048	-1.0029
1.65	0.7	-0.95	2.601	2.6	0.951	0.952	-1.0011	-1.0021
1.65	0.803	-0.847	2.5	2.5	0.85	0.849	-1.0035	-1.0024
1.65	0.904	-0.746	2.399	2.4	0.749	0.748	-1.004	-1.0027
1.65	1.006	-0.644	2.298	2.3	0.648	0.647	-1.0062	-1.0047
1.65	1.101	-0.549	2.2	2.2	0.55	0.55	-1.0018	-1.0018
1.65	1.209	-0.441	2.093	2.09	0.443	0.443	-1.0045	-1.0045
1.65	1.302	-0.348	2	2	0.35	0.35	-1.0057	-1.0057
1.65	1.401	-0.249	1.901	1.9	0.251	0.251	-1.008	-1.008
1.65	1.501	-0.149	1.801	1.8	0.151	0.151	-1.0134	-1.0134
1.65	1.606	-0.044	1.696	1.7	0.046	0.046	-1.0455	-1.0455
1.65	1.705	0.055	1.597	1.6	-0.053	-0.054	-0.9636	-0.9818
1.65	1.808	0.158	1.493	1.49	-0.157	-0.156	-0.9937	-0.9873
1.65	1.906	0.256	1.394	1.4	-0.256	-0.255	-1	-0.9961
1.65	2.009	0.359	1.293	1.29	-0.357	-0.357	-0.9944	-0.9944
1.65	2.106	0.456	1.196	1.2	-0.454	-0.453	-0.9956	-0.9934
1.65	2.203	0.553	1.099	1.1	-0.551	-0.55	-0.9964	-0.9946
1.65	2.302	0.652	0.999	1	-0.651	-0.65	-0.9985	-0.9969
1.65	2.408	0.758	0.893	0.89	-0.757	-0.757	-0.9987	-0.9987
1.65	2.504	0.854	0.797	0.8	-0.853	-0.852	-0.9988	-0.9977
1.65	2.607	0.957	0.693	0.7	-0.957	-0.955	-1	-0.9979
1.65	2.701	1.051	0.6	0.6	-1.05	-1.049	-0.999	-0.9981

Tabla 3.4. Voltajes de entrada/ salida con ganancia unitaria. (Continúa)

Voltajes de entrada		Entrada diferencial	Voamp		Salida con ganancia		Ganancia	
V+=1.65V	V-=VA=VB	V- - V+	VoA	VoB	A	B	A	B
1.65	2.809	1.159	0.492	0.49	-1.158	-1.156	-0.9991	-0.9974
1.65	2.905	1.255	0.397	0.4	-1.253	-1.251	-0.9984	-0.9968
1.65	3.005	1.355	0.296	0.3	-1.354	-1.352	-0.9993	-0.9978
1.65	3.103	1.453	0.198	0.2	-1.452	-1.45	-0.9993	-0.9979
1.65	3.206	1.556	0.095	0.1	-1.555	-1.552	-0.9994	-0.9974
1.65	3.305	1.655	0	0	-1.65	-1.65	-0.997	-0.997

Tabla 3.4. Voltajes de entrada/ salida con ganancia unitaria.

El ADC realiza la conversión de la diferencia de voltajes en las terminales no inversora e inversora de cada canal (canal A y canal B). La terminal inversora de ambos canales, CA- y CB-, están a un voltaje fijo de 1.65V, mientras que las terminales no inversoras, CA+ y CB+, están conectadas a la salida de los amplificadores VoA y VoB respectivamente. Es importante mencionar que la diferencia de estas entradas no debe ser mayor a 1.25V ni menor a -1.25V. La oscilación de voltaje en cada terminal puede ser de 0V a 3.3V, procurando que la diferencia entre terminales esté dentro de los límites ya mencionados. En caso de que la diferencia llegue a ser mayor o menor a estos límites, el valor de la conversión se mantendrá fijo al valor máximo ó al valor mínimo de conversión. El resultado de la conversión se presenta en 14 bits en complemento a dos, donde el bit más significativo representa el signo del número; si el bit más significativo está en un valor cero, el número es positivo, si está en un valor uno, el número es negativo. Los restantes 13 bits son los que representan al número que es proporcional al voltaje diferencial de entrada. El ADC tiene un voltaje de referencia de 2.5V interno, el cual se utiliza para realizar la conversión. El resultado de la conversión está definido por el voltaje diferencial entre la resolución:

$$D[13:0] = \frac{C_+ - C_-}{\frac{V_{ref}}{2^n}} \tag{3.2}$$

donde:

D[13:0] = representación digital en 14 bits de la diferencia de voltajes de entrada
C+ = voltaje en la terminal no inversora
C- = voltaje en la terminal inversora
Vref = voltaje de referencia del ADC
n = número de bits del ADC

sustituyendo valores:

$$D[13:0] = \frac{C_+ - 1.65V}{\frac{2.5V}{2^{14}}} \tag{3.3}$$

donde:

C+ = voltaje de salida del amplificador, ya sea VoA ó VoB.

La ecuación 3.3 define el comportamiento del ADC para ambos canales.

Sustituyendo la salida Voamp de la ecuación 3.1, en la ecuación 3.3, se obtiene:

$$D[13:0] = \frac{GANANCIA\ (V_- - V_+) + 1.65V\ - 1.65V}{\frac{2.5V}{2^{14}}} \quad (3.4)$$

simplificando términos se obtiene:

$$D[13:0] = \frac{GANANCIA\ (V_- - V_+)}{\frac{2.5V}{2^{14}}} \quad (3.5)$$

La ecuación 3.5 describe completamente el comportamiento de la conversión analógica digital involucrando ambos componentes, el amplificador y propiamente el ADC.

Otra representación de la ecuación 3.5 es sustituyendo V+ por 1.65V y descomponiendo el 2^{14} en $2^{13}*2$:

$$D[13:0] = \frac{GANANCIA\ (V_- - 1.65V)}{\frac{1}{2^{13}}\frac{2.5V}{2}} \quad (3.6)$$

realizando las operaciones en el denominador, haciendo 2^{13}=8192 y cambiando V- por Vent:

$$D[13:0] = GANANCIA * \frac{(V_{ent} - 1.65V)}{1.25V} * 8192 \quad (3.7)$$

En la ecuación 3.7 se observa que si la diferencia entre el voltaje de entrada y el voltaje constante de 1.65V es mayor a 1.25V, la conversión se encontrará fuera del rango de conversión. Lo mismo sucede si la diferencia entre el voltaje de entrada y el voltaje constate de 1.65V es menor a -1.25V. Por lo tanto, el rango del voltaje de entrada se reduce. Entonces, para conocer los rangos de voltaje de entrada se toma parte del numerador de la ecuación 3.7 y se iguala a los límites de operación:

$$GANANCIA * \ (V_{ent} - 1.65V)\ \geq \pm 1.25V \quad (3.8)$$

Para el voltaje mínimo de entrada:

$$V_{entmin} \geq \frac{1.25V}{GANANCIA} + 1.65V \tag{3.9}$$

Cabe recordar que la ganancia siempre es negativa.

Para el voltaje máximo de entrada:

$$V_{entmax} \leq \frac{-1.25V}{GANANCIA} + 1.65V \tag{3.10}$$

Cabe aclarar que para un valor mínimo de entrada 0V, la salida del amplificador es máxima, 3.3V, y viceversa, para un voltaje de entrada máximo de 3.3V, la salida del amplificador es mínima, 0V. Con una ganancia de -1, el voltaje Ventmin es igual a 0.4V (por la ecuación 3.9), mientras que el voltaje máximo es de 2.9V (por la ecuación 3.10). En la tabla 3.5 se muestran las ganancias y el rango de voltajes de entrada para cada una de ellas.

Ganancia	Voltaje de entrada	
	Mínimo	Máximo
0	-	-
-1	0.4	2.9
-2	1.025	2.275
-5	1.4	1.9
-10	1.525	1.775
-20	1.5875	1.7125
-50	1.625	1.675
-100	1.6375	1.6625

Tabla 3.5. Ganancias y rango de voltajes.

La ecuación 3.7 y la tabla 3.5 describen completamente el comportamiento del amplificador y del ADC para el proceso de conversión analógica digital.

Comunicación SPI para el amplificador

En la tabla 3.6 se muestran las ganancias y su código en bits para la comunicación SPI. La frecuencia máxima de operación, para la comunicación SPI, es aproximadamente de 10 MHz.

Los datos para la comunicación SPI consisten en ocho bits divididos en dos campos de cuatro bits cada uno. En estos cuatro bits se envía la ganancia deseada de cada amplificador. Los cuatro bits superiores corresponden a la ganancia de amplificador B y los cuatro bits inferiores corresponden a la ganancia del amplificador A. La comunicación SPI, figura 3.6, inicia cuando la señal AMP_CS se

lleva de un estado lógico alto a un estado lógico bajo. En este momento se inicia la señal de reloj y se colocan los datos MOSI en el flanco de subida de la señal de reloj (el bit más significativo es enviado primero). El amplificador va almacenando y desplazando los datos MOSI en un registro interno. Los datos MISO, generados por el amplificador, son mostrados en el flanco de bajada de la señal de reloj para ser leídos por el FPGA. La comunicación termina cuando la señal AMP_CS toma un nivel lógico alto. En este momento el amplificador fija los datos recibidos en el registro interno y procede a configurar los amplificadores con la ganancia especificada. Es importante mencionar que la señal SHDN_AMP se debe encontrar en nivel lógico bajo, debido a que es una habilitación por hardware. Si la señal SHDN_AMP se encuentra en nivel lógico alto, tanto los amplificadores como la interfaz SPI se encontrarán deshabilitados. En la figura 3.6 también se muestran los tiempos mínimos y máximos, en nanosegundos, de la comunicación SPI para el amplificador.

Ganancia	A3	A2	A1	A0
	B3	B2	B1	B0
0	0	0	0	0
-1	0	0	0	1
-2	0	0	1	0
-5	0	0	1	1
-10	0	1	0	0
-20	0	1	0	1
-50	0	1	1	0
-100	0	1	1	1

Tabla 3.6. Ganancia para los amplificadores.

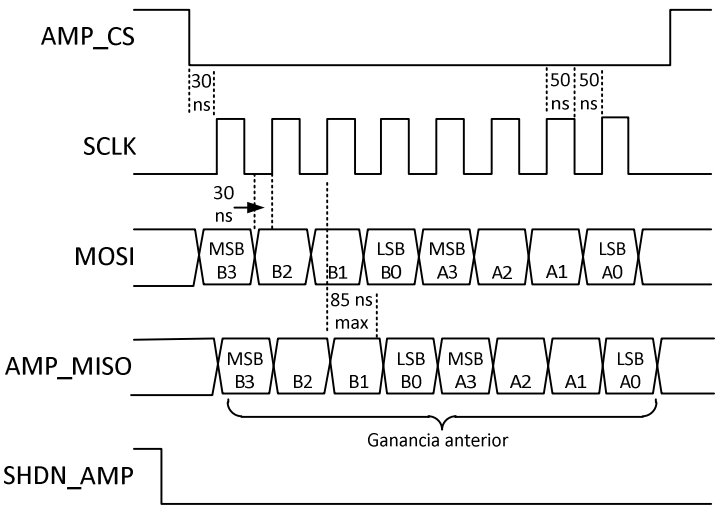


Figura 3.6. SPI para amplificador.

La descripción que se diseñó para controlar el amplificador se muestra, en diagrama de bloques, en la figura 3.7. Las entradas y salidas de la figura 3.7 son con respecto al FPGA, por ejemplo, la señal AMP_MISO es una entrada al FPGA que proviene del amplificador, mientras que la señal

SCLK es una señal de salida del FPGA que llega al amplificador. Todos los procesos se sincronizan con la señal de entrada CLK y se reinician con la señal de entrada RESET. La comunicación inicia con un flanco de subida en la señal de entrada INICIA. El final de la comunicación se observa en la señal de salida FINAL, la cual tendrá un estado lógico alto cuando esto ocurre. La descripción cuenta también con las cuatro líneas SPI: AMP_CS, AMP_MISO, SCLK, MOSI, así como la terminal extra SHDN_AMP, la cual siempre tiene un nivel lógico bajo (siempre está habilitado por hardware). Para comprobar la respuesta del amplificador se tiene la señal SALIDA_MISO, en donde se observa directamente la entrada AMP_MISO, con la cual se puede corroborar el funcionamiento utilizando, por ejemplo, un osciloscopio. Por último se cuentan con las señales de salida necesarias para desactivar los componentes conectados a las líneas SPI: SPI_SS_B, SF_CE0, FPGA_INIT_B, DAC_CS y CONV. Estas señales tienen un valor fijo durante toda la comunicación con el amplificador.

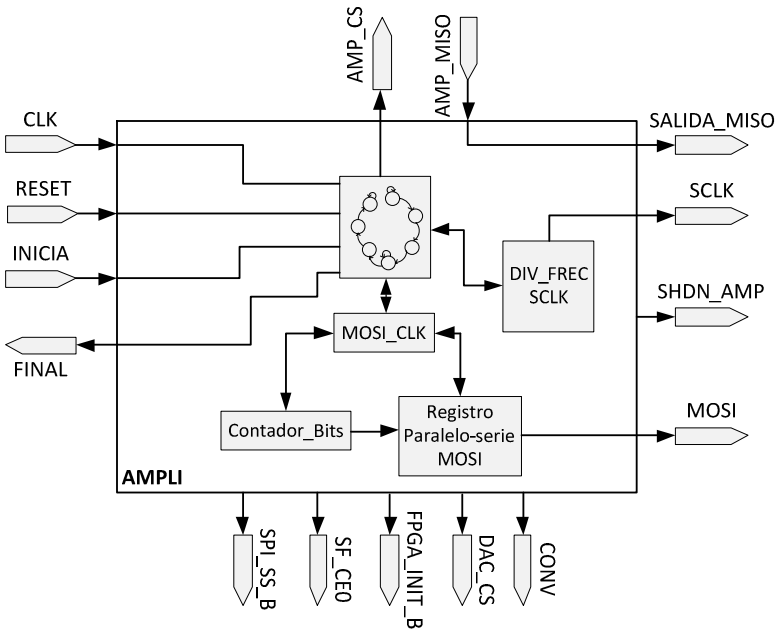


Figura 3.7. Diagrama de la descripción para el amplificador.

Se corroboró el funcionamiento de la descripción mencionada por medio de la simulación realizada en VHDL, con las herramientas proporcionadas por Xilinx. La simulación consiste en ingresar valores de las señales de entrada para obtener valores en las señales de salida que corresponden a la lógica de la descripción realizada. En la simulación mostrada en la figura 3.8 se observan las entradas y salidas que conforman la descripción diseñada. Para dar inicio a la comunicación SPI, se ingresa un valor en la señal INICIA. Se observa que inmediatamente de ingresar un flanco de subida en dicha señal, inicia la comunicación con el cambio en la señal AMP_CS. Los datos en la línea MOSI, los cuales corresponden a una ganancia unitaria (ver tabla 3.6), se encuentran centrados con respecto al flanco de subida de la señal SCLK. Para el caso de la señal de entrada AMP_MISO, se ingresan valores de entrada arbitrarios el cual corresponde exactamente a la señal de salida SALIDA_MISO. La comunicación termina cuando la señal AMP_CS vuelve a cambiar de estado e inmediatamente la señal FINAL toma un valor lógico alto. Las señales

para desactivar los componentes conectados a las líneas SPI se encuentran con un valor fijo durante toda la comunicación, así como también la habilitación por hardware SHDN_AMP.

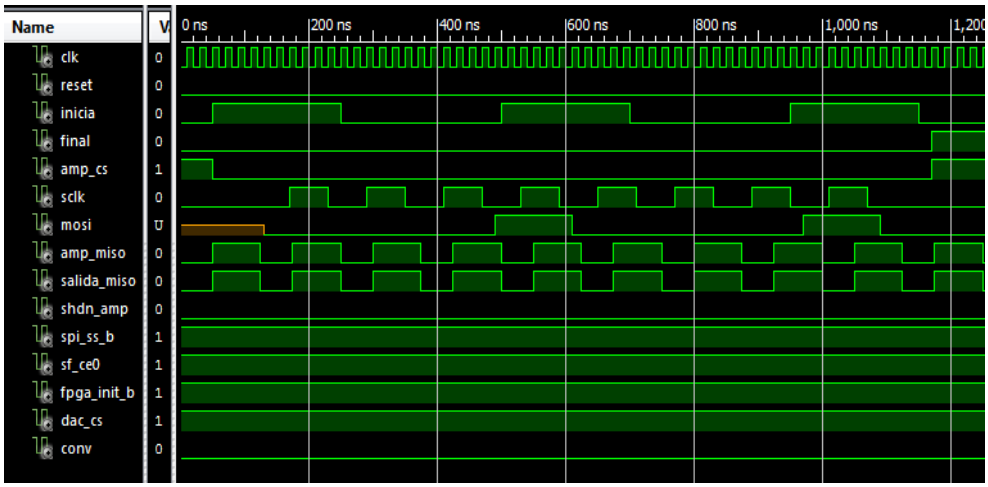


Figura 3.8. Simulación de la descripción diseñada.

Obteniendo los resultados esperados en la simulación, se procedió a configurar el FPGA con la descripción realizada. Al configurar el FPGA y ponerlo en funcionamiento, se obtuvo una comunicación exitosa debido a que el amplificador se configuró correctamente con la ganancia enviada y respondió correctamente, por medio de la terminal AMP_MISO, al FPGA.

En la figura 3.9 se muestra la señal SCLK (en amarillo) y la señal MOSI (en azul). Los datos MOSI se encuentran centrados con respecto al flanco de subida de la señal SCLK.

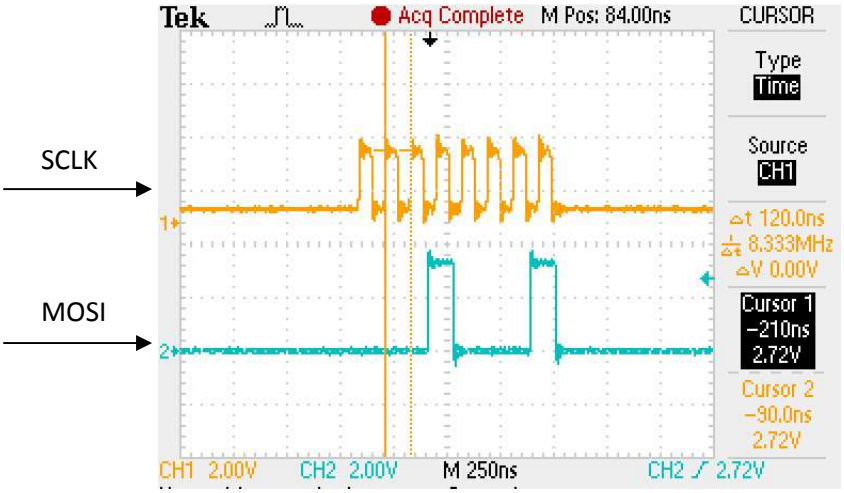


Figura 3.9. Señal SCLK (amarillo) y MOSI (azul).

En la figura 3.10 se muestra la señal SCLK (en amarillo) con la señal SALIDA_MISO (en azul), la cual es la señal AMP_MISO pero convertida en salida para el FPGA. Se observa que los datos de la ganancia almacenada anteriormente en el amplificador se encuentran centrados con respecto al

flanco de bajada de la señal SCLK, corroborando así la correcta comunicación SPI con el amplificador.

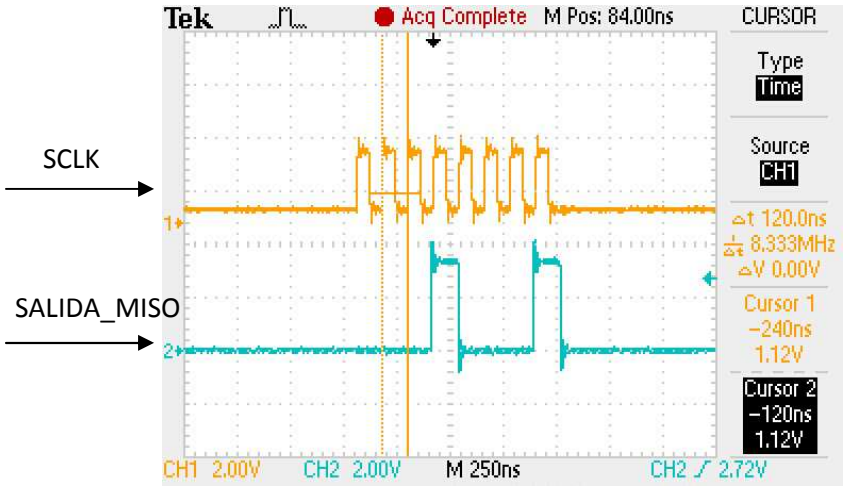


Figura 3.10. Señal SCLK (amarillo) y SALIDA_MISO (azul).

La descripción realizada se comprobó con diferentes valores de ganancia, obteniendo resultados correctos en cada una de ellas. Se corroboró mediante la terminal SALIDA_MISO, en donde se observa la ganancia cargada anteriormente, y en la línea MOSI en donde se observa la ganancia que se quiere cargar al amplificador.

Comunicación SPI para el ADC

La comunicación con el ADC es ligeramente diferente al SPI tradicional, debido a que sólo contiene dos líneas de dicha comunicación, estas líneas son: SCLK y MISO. La transferencia de información, figura 3.11, inicia en el flanco de subida del pulso en la señal CONV. Posteriormente, se genera la señal SCLK para obtener el resultado de la conversión anterior, por medio de la terminal MISO. El resultado de la conversión de ambos canales se presenta en una cadena de 32 bits, separando cada canal por un espacio de dos bits, todos ellos alineados con el flanco de bajada de la señal SCLK. En la figura 3.11 también se muestran los tiempos mínimos que se deben tomar en cuenta para una comunicación adecuada. Es necesario esperar un determinado tiempo entre cada conversión para obtener un valor correcto de la misma. Se recomiendan 2 pulsos de reloj extra en la señal SCLK para cumplir con esta condición, por lo tanto, los pulsos de reloj en la señal SCLK llegan a ser 34. Si no son posibles los 34 pulsos en la señal SCLK se debe mantener por cualquier otro medio el tiempo requerido.

Conociendo cómo se tiene que realizar la transferencia de información con el ADC, se procedió a diseñar una descripción en VHDL para el control del mismo. El diagrama de bloques de la descripción elaborada se muestra en la figura 3.12. Una entrada, por ejemplo la señal MISO, es una entrada al FPGA. Una salida, por ejemplo la señal SCLK, es una salida del FPGA. La señal CLK es la señal de reloj que sincroniza todos los procesos dentro del FPGA, la señal RESET reinicia todos los procesos también dentro del FPGA y la señal INICIO da inicio a toda la comunicación SPI. La

señal SEL_CANAL se utiliza para seleccionar el canal a mostrar, ya sea el canal A o el canal B. La señal de salida CONV es la que le indica al ADC que inicie una conversión. Con la señal SCLK, de 34 pulsos, se realiza la transferencia de información, toda ella con respecto al flanco de bajada de esta señal. En la señal MISO se obtiene el resultado de la conversión que envía el ADC, la cual captura el FPGA y la muestra en la señal SALIDA, esta última de 14 bits. Para que no exista interferencia en la comunicación con el ADC, se desconectan los componentes conectados a las líneas SPI: SPI_SS_B, SF_CE0, FPGA_INIT_B, DAC_CS y AMP_CS.

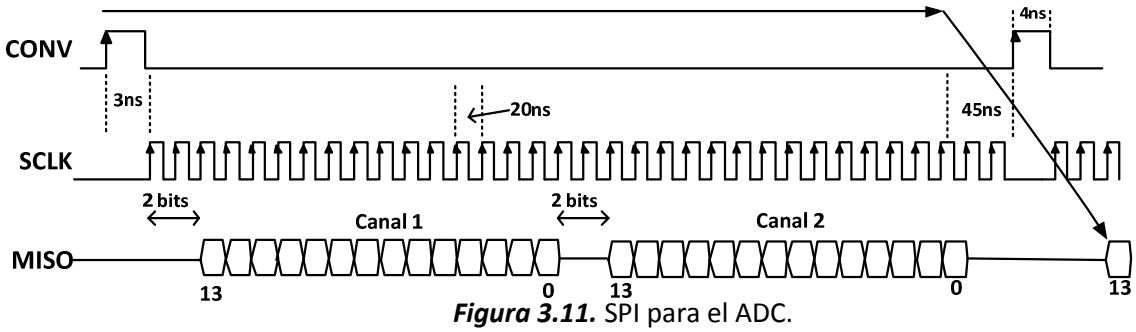


Figura 3.11. SPI para el ADC.

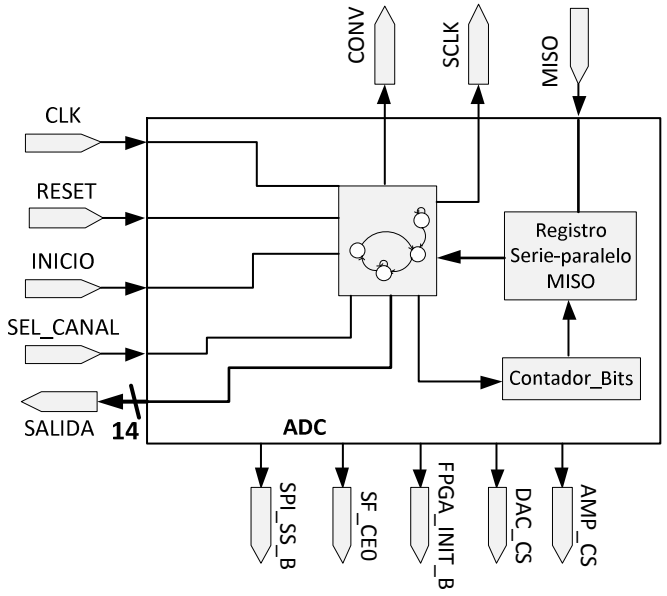


Figura 3.12. Control del ADC.

Para corroborar el buen funcionamiento de la descripción diseñada, se procedió a simularla obteniendo las señales que se muestran en la figura 3.13. Se observa que la comunicación comienza con un flanco de subida de la señal INICIO, sincronizada con CLK, de una duración de 100ns, la cual provoca un pulso en la señal CONV de duración de 20ns. Ésta última, a su vez provoca el inicio de la señal SCLK de 34 pulsos. La señal de entrada MISO se simuló con un tren de pulsos, los cuales se capturan con el flanco de bajada de la señal SCLK. El resultado de esta captura se muestra en la señal SALIDA, ésta presenta el canal A seleccionado con la señal SEL_CANAL,

mediante un nivel lógico bajo. Se observa que la señal SALIDA se va actualizando en cada flanco de bajada de la señal SCLK a partir del tercer flanco.

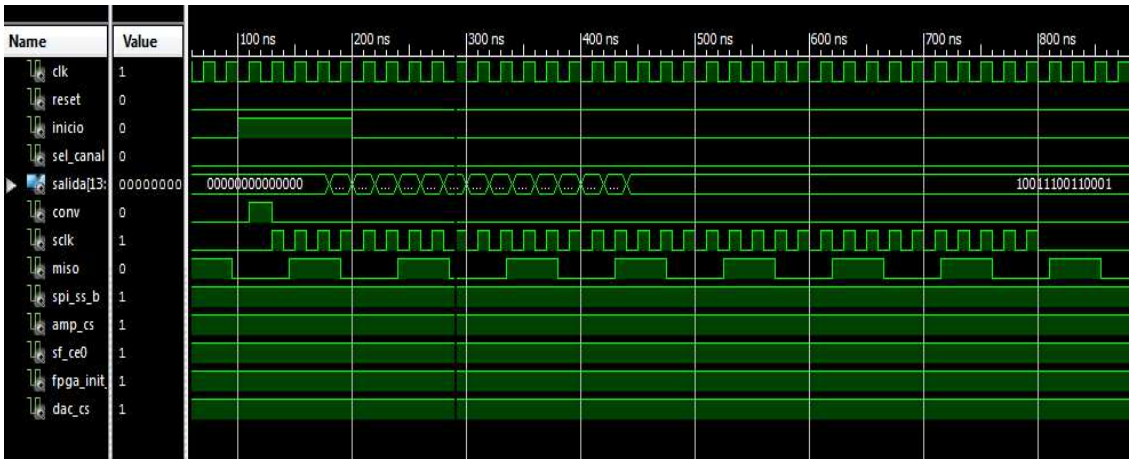


Figura 3.13. Simulación del control para el ADC.

Para poder implementar el control diseñado para el ADC en el FPGA, se tiene que realizar una descripción que incluya el control del ADC y, además, el control del amplificador, esto por la conexión en cascada que se tiene entre estos dos componentes (ver figura 3.4). Si el amplificador no se ha programado antes de que se controle el ADC, la señal analógica de entrada no estará presente en el ADC y por lo tanto, no se obtiene una conversión del voltaje de entrada. La unión de estas descripciones se realizó de manera gráfica con las herramientas de Xilinx. Se crearon los símbolos esquemáticos de cada descripción (del amplificador y ADC) y se conectaron entre sí para obtener la descripción que engloba ambas descripciones (descripción de mayor jerarquía). En la figura 3.14 se muestran los elementos que componen a la descripción de mayor jerarquía. Es importante aclarar que estos elementos son los controles realizados con anterioridad en VHDL y que estarán implementados dentro del FPGA, por lo tanto, la señal CLK, por ejemplo, es la señal de reloj que corresponde a una señal de entrada para el FPGA y que es común a ambas descripciones internas y, por ejemplo, la señal SCLK es una señal de salida del FPGA, común para ambas descripciones internas y que, además, es común para ambos circuitos integrados (ver figura 3.4).

La descripción del amplificador y del ADC en diagrama de bloques se muestra en la figura 3.15. Se muestran todas las señales con las que cuenta dicha descripción. La señal CLK es la señal para sincronizar todos los procesos y la señal RESET reinicia todos los procesos. Un flanco positivo en INICIO provoca el comienzo de la comunicación con los dispositivos, primero con el amplificador y después con el ADC. Las señales SHDN_AMP, MOSI, AMP_MISO, AMP_CS y SALIDA_MISO son propiamente las señales que controlan la configuración del amplificador. Las señales SEL_CANAL, MISO, CONV y SALIDA son las señales que controlan el ADC. La señal SCLK es una señal común a ambos componentes, tanto en el control como en los circuitos integrados; se coloca en una compuerta lógica OR para evitar cualquier problema de compatibilidad en los niveles lógicos. Por último, las señales FPGA_INIT_B, DAC_CS, SPI_SS_B y SF_CE0 se utilizan para desactivar los otros componentes conectados a las líneas SPI.

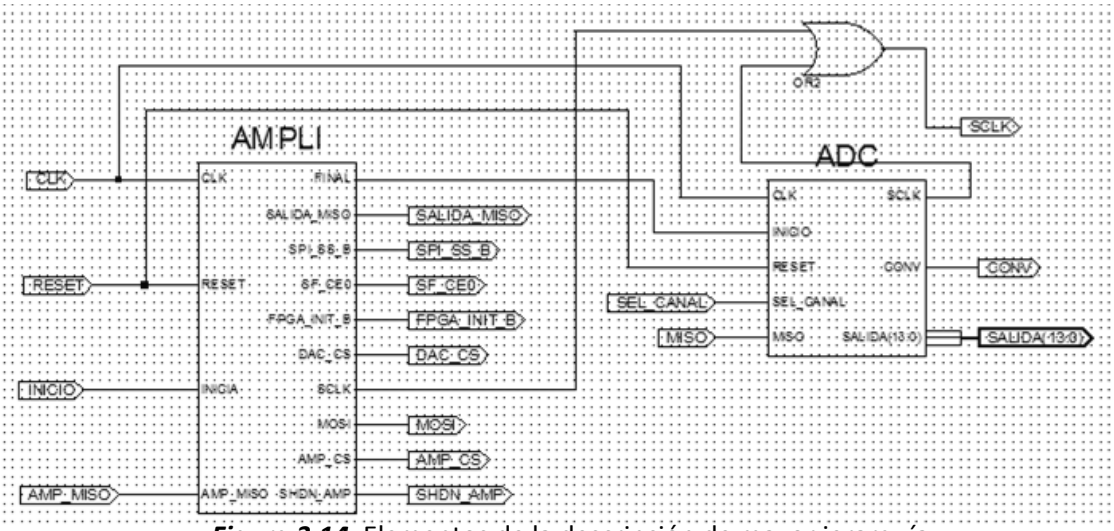


Figura 3.14. Elementos de la descripción de mayor jerarquía.

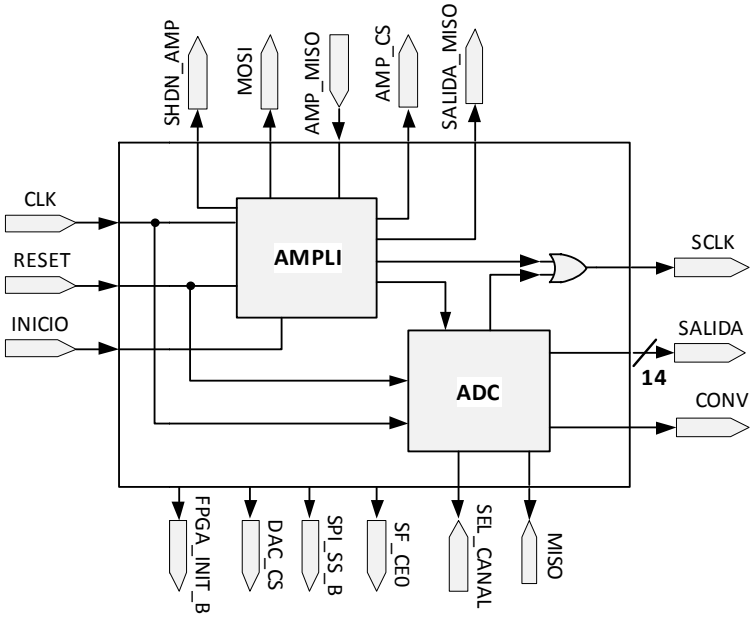


Figura 3.15. Descripción para el control del amplificador y el ADC.

Se corroboró la descripción del amplificador y del ADC realizando una simulación, que se muestra en la figura 3.16. En ella se observan todas las señales involucradas en la descripción. La comunicación inicia con un pulso en la entrada INICIO. Con este pulso se habilita el amplificador mediante la señal AMP_CS, también se genera la señal SCLK y por lo tanto, se colocan los datos en la línea MOSI. La comunicación con el amplificador termina colocando la señal AMP_CS en nivel lógico alto, inmediatamente después se genera el pulso en la señal CONV para iniciar la comunicación con el ADC. Una vez iniciada la comunicación con el ADC, se mantiene en este proceso hasta reiniciarse todo el control. Se capturan los datos de la conversión con el flanco de bajada de la señal SCKL y se muestran en la señal SALIDA. Las entradas AMP_MISO y MISO se simularon con un tren de pulsos arbitrario para cada señal. Las señales para los otros

componentes en la línea SPI y la habilitación por hardware del amplificador siempre tienen un valor fijo en toda la comunicación.

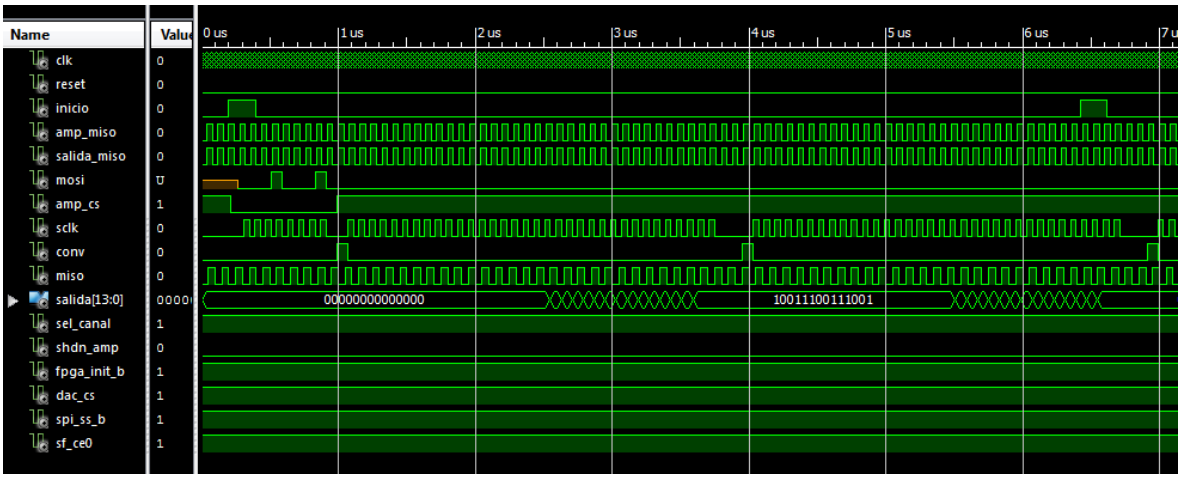


Figura 3.16. Simulación del control del amplificador y del ADC.

Para comprobar el funcionamiento de la descripción realizada, se ingresó un voltaje con una fuente de corriente directa y se observó el resultado de la conversión en unos LEDs, tanto de la tarjeta SPARTAN 3E como externos, estos últimos conectados a unas terminales de propósito general del FPGA. Se establecieron algunos voltajes de entrada, dentro de los rangos establecidos en la tabla 3.5, y con la ecuación 3.7 se obtuvo el valor teórico de la conversión (en base decimal para facilitar el manejo de los números). El valor que se obtuvo en los LEDS se convirtió en un valor decimal, igualmente por facilidad en el manejo de los números y se comparó con el valor teórico obtenido con la ecuación 3.7. Se calculó el voltaje, despejado de la ecuación 3.7, correspondiente al valor obtenido en los LEDs con el propósito de conocer el voltaje que la conversión proporcionaba. En esta implementación no se obtuvieron los resultados esperados ya que los valores mostrados en los LEDs no correspondían, ni eran parecidos, a los valores teóricos obtenidos con la ecuación 3.7. Al medir las señales involucradas, se detectó que la señal MISO estaba muy distorsionada; los cambios de los niveles lógicos no estaban bien definidos y por lo tanto no se detectaban los valores adecuados de la conversión. En la figura 3.17 se muestra en amarillo, la señal SCLK y en azul, la señal MISO, la cual se encuentra muy distorsionada.

Como la frecuencia de la señal SCLK era de 50MHz, la máxima para el dispositivo, se disminuyó como una posible solución al problema. Se decidió, en primer instancia, modificar la frecuencia porque, en la simulación, la captura de datos era correcta y porque en la implementación se logró mostrar datos en los LEDs. La disminución de la frecuencia se realizó modificando la descripción de la figura 3.14, agregando un divisor de frecuencia para el ADC, quedando finalmente como la mostrada en la figura 3.18. En ella solamente se añade el divisor de frecuencia a la entrada de reloj del control del ADC, mientras que las entradas y salidas de toda la descripción no se modificaron. Con esta modificación los resultados fueron los correctos, ya que se obtuvieron valores de conversión acordes con la ecuación 3.7.

Figura 3.17. Señal SCLK (amarillo) y señal MISO (azul).

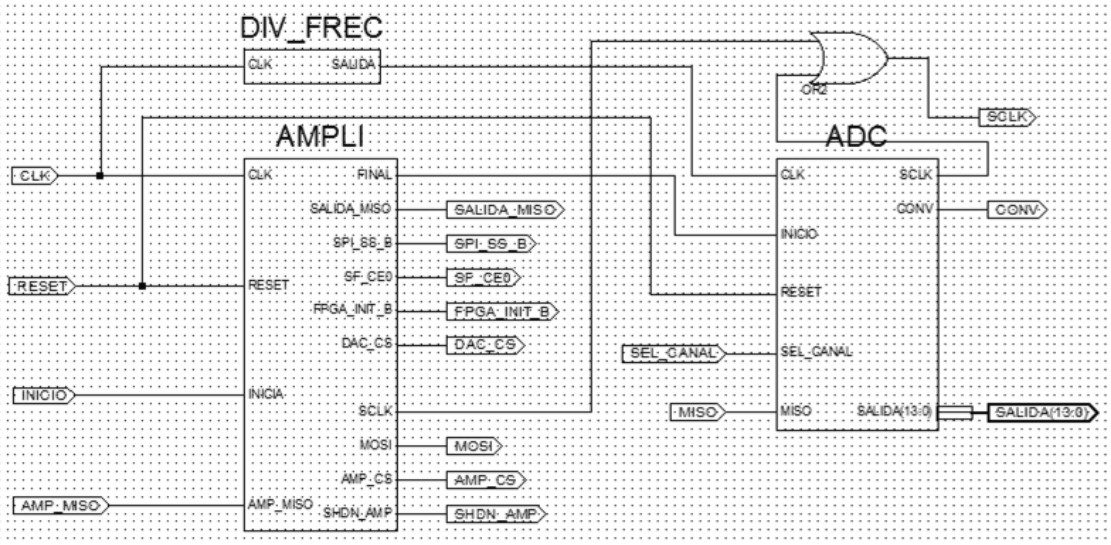


Figura 3.18. Descripción modificada.

Disminuyendo la frecuencia de la señal SCLK del ADC, se obtuvieron formas de onda con mejor definición, tanto para SCLK como para la señal MISO, figura 3.19. La señal MISO, que presentaba una distorsión grande, se convirtió en una señal en donde se perciben bien marcados los cambios de niveles lógicos en los datos a capturar. La señal SCLK se observa como una señal cuadrada bien definida con muy poco ruido en ella. La frecuencia de la señal SCLK se disminuyó de 50MHz a 12.5MHz con un periodo correspondiente de 80ns.

Ya que se observaron los valores adecuados en los LEDS, con respecto a los valores obtenidos con la ecuación 3.7, se procedió a realizar conversiones para corroborar el correcto funcionamiento de la descripción en diversas situaciones. En la tabla 3.7 se muestran las conversiones que se realizaron, para una ganancia unitaria, con la descripción modificada. Los canales A y B, en la

primera columna, se colocaron al mismo voltaje por facilidad y para corroborar que los dos convertidores tuvieran, teóricamente, la misma salida. La segunda columna es la entrada diferencial del amplificador que corresponde al valor de entrada, primer columna, menos 1.65V. La tercera columna es el voltaje de salida del amplificador en donde se observa el efecto de la ganancia. La cuarta columna es el valor teórico, en decimal para un mejor manejo de los números, calculado con la ecuación 3.7. La s*alida*, quinta columna, es el valor mostrado en los LEDs convertido a decimal para una mejor visualización y un mejor manejo en los números, esto, con base en que cualquier número decimal, se puede representar en forma binaria, y en este caso particular, en forma binaria de 14 bits en complemento a dos. La sexta columna, *salida en volts*, es el voltaje analógico que corresponde al valor binario, o decimal, de la salida del convertidor, es decir, de la quinta columna. Este voltaje se calculó a partir de la ecuación 3.7.

Figura 3.19. Señal SCLK (amarillo) y señal MISO (azul) bien definidas.

Entrada	**Entrada diferencial**	**Salida con ganancia**	**Ec 3.7**	**Salida**		**Salida (en Volts)**	
VA=VB	Amplificador	Amplificador	D[13:0]	Canal A	Canal B	Canal A	Canal B
0.104	-1.546	1.546	10131.8656	8191	8191	0.4002	0.4002
0.205	-1.445	1.445	9469.952	8191	8191	0.4002	0.4002
0.307	-1.343	1.343	8801.4848	8191	8191	0.4002	0.4002
0.408	-1.242	1.242	8139.5712	8191	8170	0.4002	0.4034
0.509	-1.141	1.141	7477.6576	7502	7479	0.5053	0.5088
0.605	-1.045	1.045	6848.512	6875	6819	0.601	0.6095
0.7	-0.95	0.95	6225.92	6150	6082	0.7116	0.722
0.803	-0.847	0.847	5550.8992	5556	5544	0.8022	0.8041
0.904	-0.746	0.746	4888.9856	4901	4889	0.9022	0.904
1.006	-0.644	0.644	4220.5184	4228	4170	1.0049	1.0137
1.101	-0.549	0.549	3597.9264	3728	3710	1.0812	1.0839

Tabla 3.7. Conversión con ganancia de -1. (Continúa)

Entrada	Entrada diferencial	Salida con ganancia	Ec 3.7	Salida		Salida (en Volts)	
VA=VB	amplificador	amplificador	D[13:0]	Canal A	Canal B	Canal A	Canal B
1.209	-0.441	0.441	2890.1376	2888	2880	1.2093	1.2105
1.302	-0.348	0.348	2280.6528	2140	2130	1.3235	1.325
1.401	-0.249	0.249	1631.8464	1619	1608	1.403	1.4046
1.501	-0.149	0.149	976.4864	824	818	1.5243	1.5252
1.606	-0.044	0.044	288.3584	281	274	1.6071	1.6082
1.705	0.055	-0.055	-360.448	-432	-477	1.7159	1.7228
1.808	0.158	-0.158	-1035.4688	-1166	-1172	1.8279	1.8288
1.906	0.256	-0.256	-1677.7216	-1514	-1806	1.881	1.9256
2.009	0.359	-0.359	-2352.7424	-2354	-2396	2.0092	2.0156
2.106	0.456	-0.456	-2988.4416	-3010	-3010	2.1093	2.1093
2.203	0.553	-0.553	-3624.1408	-3580	-3582	2.1963	2.1966
2.302	0.652	-0.652	-4272.9472	-4267	-4307	2.3011	2.3072
2.408	0.758	-0.758	-4967.6288	-5000	-5002	2.4129	2.4132
2.504	0.854	-0.854	-5596.7744	-5472	-5508	2.485	2.4905
2.607	0.957	-0.957	-6271.7952	-6332	-6330	2.6162	2.6159
2.701	1.051	-1.051	-6887.8336	-6988	-7022	2.7163	2.7215
2.809	1.159	-1.159	-7595.6224	-7582	-7580	2.8069	2.8066
2.905	1.255	-1.255	-8224.768	-8192	-8192	2.9	2.9
3.005	1.355	-1.355	-8880.128	-8192	-8192	2.9	2.9
3.103	1.453	-1.453	-9522.3808	-8192	-8192	2.9	2.9
3.206	1.556	-1.556	-10197.402	-8192	-8192	2.9	2.9
3.305	1.655	-1.655	-10846.208	-8192	-8192	2.9	2.9

Tabla 3.7. Conversión con ganancia de -1.

Como se observa en la tabla 3.7, los rangos de entrada para la ganancia de -1, corresponden con los mostrados en la tabla 3.5. Para entradas menores a 0.4V la salida se mantiene en 0.4V y para entradas mayores a 2.9V la salida se mantiene en 2.9V. En la figura 3.20 se muestra una gráfica del voltaje de entrada, primer columna de la tabla 3.7, contra el voltaje de salida del convertidor para los tres casos: la conversión teórica (en azul), el canal A y el canal B (ambos en rojo). En el caso teórico, la gráfica debe ser una línea recta porque el voltaje de entrada es igual al voltaje de salida, mientras que para el caso del canal A y B se observan algunas desviaciones con respecto al valor teórico. Las desviaciones (error relativo) entre el voltaje de entrada y el voltaje de salida no pasan el 5%.

En la tabla 3.8 se muestra la conversión ahora con una ganancia de -2. Los rangos de entrada de la tabla 3.8, para una ganancia de -2, corresponden con los mostrados en la tabla 3.5. Para entradas menores a 1.025V la salida se mantiene en 1.025V y para entradas mayores a 2.275V la salida se mantiene en 2.275V. En la figura 3.21 se muestra la gráfica de los valores teóricos (en azul), canal A y canal B (ambos en rojo) con un error relativo menor al 5%.

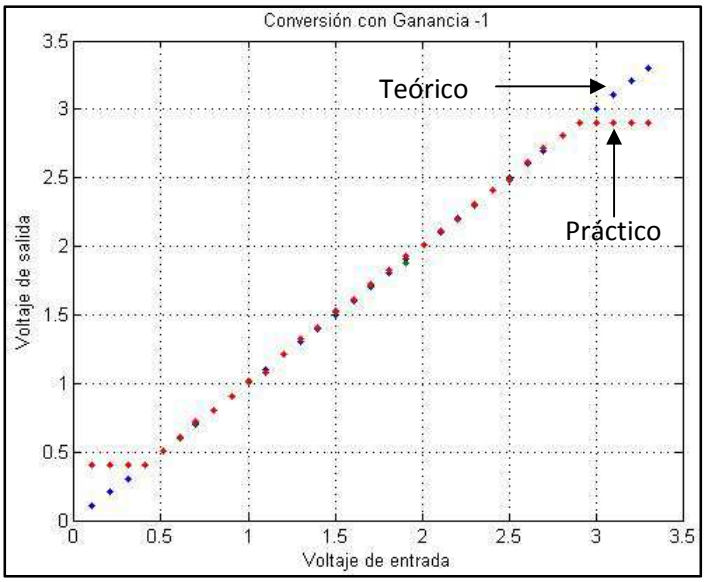


Figura 3.20. Gráfica para una ganancia de -1.

Entrada	**Entrada diferencial**	**Salida con ganancia**	**Ec 3.7**	**Salida**		**Salida (en Volts)**	
VA=VB	amplificador	amplificador	D[13:0]	Canal A	Canal B	Canal A	Canal B
0.905	-0.745	1.49	9764.864	8191	8191	1.0251	1.0251
0.956	-0.694	1.388	9096.3968	8191	8191	1.0251	1.0251
1.006	-0.644	1.288	8441.0368	8191	8191	1.0251	1.0251
1.05	-0.6	1.2	7864.32	7612	7566	1.0693	1.0728
1.107	-0.543	1.086	7117.2096	7109	7103	1.1076	1.1081
1.15	-0.5	1	6553.6	6808	6797	1.1306	1.1314
1.207	-0.443	0.886	5806.4896	5547	5541	1.2268	1.2273
1.25	-0.4	0.8	5242.88	5253	5246	1.2492	1.2498
1.302	-0.348	0.696	4561.3056	4816	4809	1.2826	1.2831
1.352	-0.298	0.596	3905.9456	3891	3843	1.3531	1.3568
1.407	-0.243	0.486	3185.0496	3196	3189	1.4062	1.4067
1.452	-0.198	0.396	2595.2256	2346	2340	1.471	1.4715
1.504	-0.146	0.292	1913.6512	1907	1901	1.5045	1.505
1.553	-0.097	0.194	1271.3984	1272	1267	1.553	1.5533
1.6	-0.05	0.1	655.36	422	377	1.6178	1.6212
1.653	0.003	-0.006	-39.3216	-66	-79	1.655	1.656
1.704	0.054	-0.108	-707.7888	-451	-496	1.6844	1.6878
1.756	0.106	-0.212	-1389.363	-1644	-1692	1.7754	1.7791
1.807	0.157	-0.314	-2057.83	-2121	-2075	1.8118	1.8083
1.849	0.199	-0.398	-2608.333	-2335	-2339	1.8281	1.8285
1.902	0.252	-0.504	-3303.014	-3291	-3298	1.9011	1.9016

Tabla 3.8. Conversión con ganancia de -2. (Continúa)

Entrada	Entrada diferencial	Salida con ganancia	Ec 3.7	Salida		Salida (en Volts)	
VA=VB	amplificador	amplificador	D[13:0]	Canal A	Canal B	Canal A	Canal B
1.951	0.301	-0.602	-3945.267	-3937	-3944	1.9504	1.9509
2.003	0.353	-0.706	-4626.842	-4387	-4430	1.9847	1.988
2.049	0.399	-0.798	-5229.773	-5195	-5196	2.0463	2.0464
2.101	0.451	-0.902	-5911.347	-5905	-5953	2.1005	2.1042
2.153	0.503	-1.006	-6592.922	-6722	-6711	2.1628	2.162
2.2	0.55	-1.1	-7208.96	-7192	-7235	2.1987	2.202
2.253	0.603	-1.206	-7903.642	-7643	-7646	2.2331	2.2333
2.275	0.625	-1.25	-8192	-8160	-8166	2.2726	2.273
2.301	0.651	-1.302	-8532.787	-8192	-8192	2.275	2.275

Tabla 3.8. Conversión con ganancia de -2.

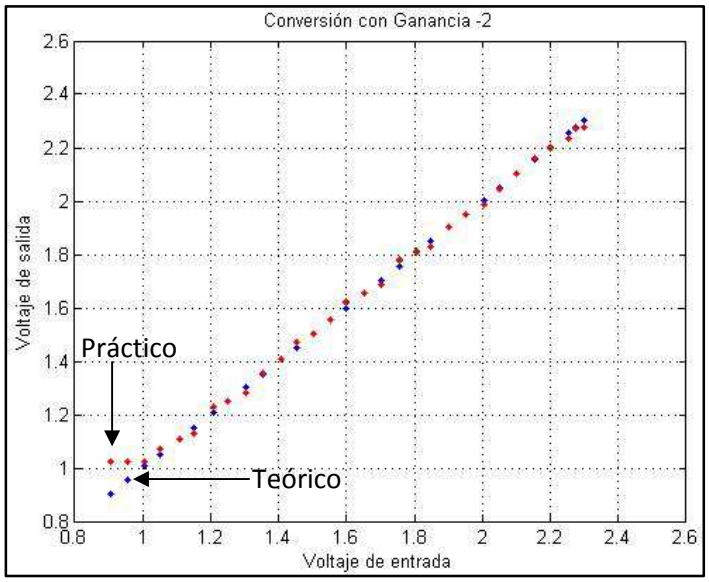


Figura 3.21. Gráfica para una ganancia de -2.

Por último, en la tabla 3.9 se muestra la conversión ahora con una ganancia de -50. Los rangos de entrada de la tabla 3.9, para una ganancia de -50, corresponden con los mostrados en la tabla 3.5. Para entradas menores a 1.625V la salida se mantiene en 1.625V y para entradas mayores a 1.675V la salida se mantiene en 1.675V. En la figura 3.22 se muestra la gráfica de los valores teóricos (en azul), canal A y canal B (ambos en rojo) con un error relativo menor al 5%.Para este tipo de ganancias, la variación de voltaje en la entrada debe ser muy pequeña por lo tanto, las ganancias altas están orientadas a señales con variación muy pequeña.

Entrada	Entrada diferencial	Salida con ganancia	Ec 3.7	Salida		Salida (en Volts)	
VA=VB	amplificador	amplificador	D[13:0]	Canal A	Canal B	Canal A	Canal B
1.61	-0.04	2	13107.2	8191	8191	1.625	1.625
1.627	-0.023	1.15	7536.64	8047	8031	1.6254	1.6255
1.63	-0.02	1	6553.6	6870	6849	1.629	1.6291
1.637	-0.013	0.65	4259.84	4140	4107	1.6374	1.6375
1.64	-0.01	0.5	3276.8	3580	3541	1.6391	1.6392
1.645	-0.005	0.25	1638.4	1916	1883	1.6442	1.6443
1.652	0.002	-0.1	-655.36	-635	-638	1.6519	1.6519
1.654	0.004	-0.2	-1310.72	-868	-861	1.6526	1.6526
1.66	0.01	-0.5	-3276.8	-3113	-3156	1.6595	1.6596
1.665	0.015	-0.75	-4915.2	-4808	-4863	1.6647	1.6648
1.669	0.019	-0.95	-6225.92	-5995	-6002	1.6683	1.6683
1.675	0.025	-1.25	-8192	-7878	-7904	1.674	1.6741
1.678	0.028	-1.4	-9175.04	-8192	-8192	1.675	1.675

Tabla 3.9. Conversión con ganancia de -50.

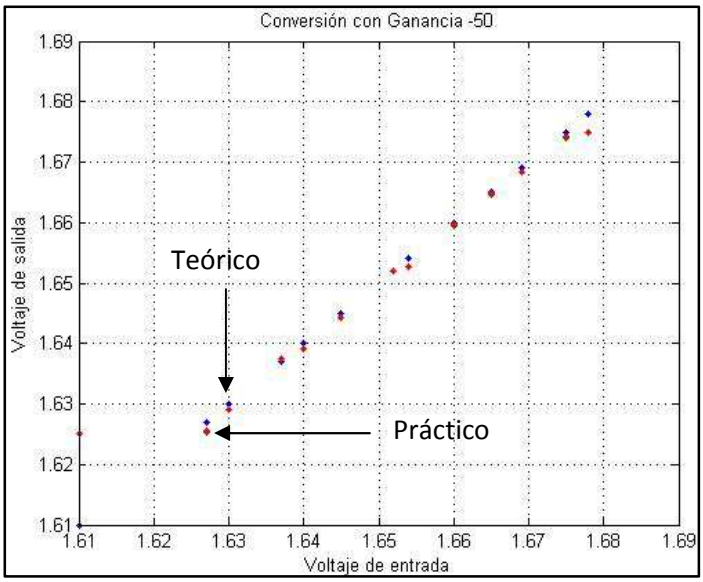


Figura 3.22. Gráfica para una ganancia de -50.

Con las tablas 3.7, 3.8 y 3.9 se comprobó que la descripción realizada para el control del amplificador y del ADC funciona correctamente, ya que la salida de los convertidores concuerda, o es muy similar, a los valores teóricos obtenidos con la ecuación 3.7. Existe una variación en los valores obtenidos de la conversión con respecto a la entrada analógica, la cual, en formato decimal, es grande, en voltaje, no es mayor a unas decenas de milivolts.

3.3.3. Convertidor digital analógico

La conexión entre el DAC y el FPGA se muestra en la figura 3.23. Se observan las cuatro líneas del protocolo SPI y una terminal extra, DAC_CLR, que coloca a 0V las salidas del DAC (reinicio por hardware). Cabe recordar que para trabajar con el DAC se tienen que desactivar los otros componentes conectados a las líneas SPI para no interferir con la comunicación (tabla 3.2).

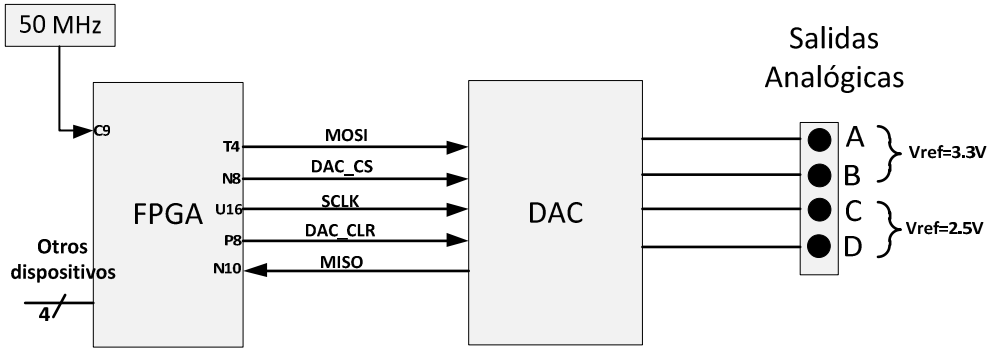


Figura 3.23. Conexión FPGA y DAC.

El circuito integrado del DAC es el LTC2624, que contiene cuatro convertidores de 12 bits cada uno con voltajes de referencias individuales. El rango de polarización es de 2.5V a 5.5V. La frecuencia de operación, para la comunicación SPI, es de hasta 50MHz.

Los DACs pueden configurarse con una comunicación de 24 o 32 bits. Los campos que conforman los 24 bits son los siguientes: 4 bits para comandos, 4 bits para direcciones, 12 bits de datos y 4 bits de relleno. Para una comunicación de 32 bits se agregan al inicio 8 bits de relleno. El valor lógico de los bits de relleno pueden ser altos o bajos. Los bits de comando tienen un valor típico, en binario, de "0011". Los bits de direcciones mostrados en la tabla 3.10, seleccionan en qué convertidor se obtendrá la salida. Los 12 bits de datos es el equivalente, en binario, al valor de voltaje que se desea tener en la salida.

a_3	a_2	a_1	a_0	DAC
0	0	0	0	A
0	0	0	1	B
0	0	1	0	C
0	0	1	1	D
1	1	1	1	Todos

Tabla 3.10. Selección de DAC.

La comunicación SPI de 32 bits con el DAC, figura 3.24, inicia cuando la señal DAC_CS transita de un valor lógico alto a un valor lógico bajo, después, se genera la señal SCLK y se transmiten los bits de datos, en la línea MOSI, enviando el bit más significativo primero. El convertidor captura los datos MOSI en el flanco de subida de la señal de reloj SCLK. Los datos MISO son presentados por el DAC en el flanco de bajada de la señal SCLK. Los datos MISO corresponden a la conversión, el

comando y la dirección que se cargó anteriormente en el DAC. Estos datos pueden servir para corroborar el correcto funcionamiento del DAC y de la comunicación SPI. La comunicación termina cuando la señal DAC_CS transita de un valor lógico bajo a un valor lógico alto y en este momento, el DAC seleccionado, actualiza la salida con el valor de voltaje solicitado. La señal DAC_CLR debe permanecer en nivel alto, ya que es un reinicio por hardware, de lo contrario las salidas del DAC no se actualizarán. Para una comunicación de 24 bits, los 8 bits de inicio se omiten y se comienza directamente con los bits de comando.

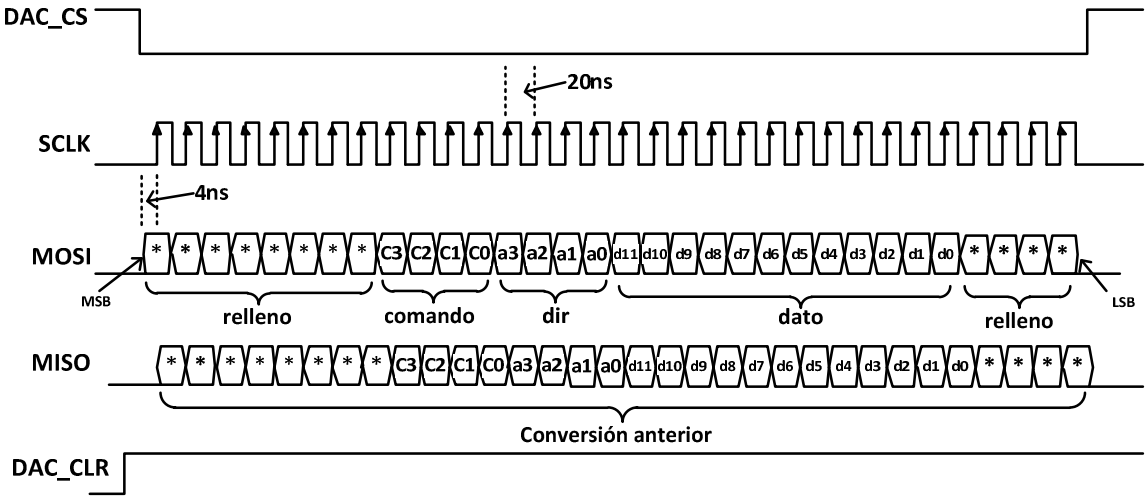


Figura 3.24. Comunicación SPI de 32 bits para el DAC.

Conociendo la comunicación SPI para el DAC, se diseñó el control para dicho dispositivo. El diagrama de bloques del control diseñado se muestra en la figura 3.25. Las señales de entrada CLK y RESET son las que sincronizan y reinician los procesos de control respectivamente. Con un flanco de subida de la señal INICIO, sincronizada con el reloj, se comienza la comunicación con el DAC. La señal de entrada BOTON se utiliza para indicar que los datos para la conversión están listos para ser cargados por la descripción. La señal DATOS, de 12 bits, son los datos a convertir por el DAC, estos datos se cargan en paralelo. La señal FIN_CONV indica el final de la conversión y se coloca en estado lógico alto cuando esto ocurre. La señal de salida DAME_DATO indica que la descripción está esperando los datos a convertir, indicado mediante un estado alto en esta señal. La salida DAC_CS es la habilitación del circuito integrado de la comunicación SPI. La entrada MISO la proporciona el amplificador y contiene los datos del DAC, estos datos son los datos de la conversión que se ha cargado anteriormente. La señal de salida SALIDA_MISO se utiliza para observar la respuesta del DAC y corroborar la correcta comunicación con el mismo. La señal de salida SCLK, de 32 pulsos, es la señal de reloj que sincroniza todo el proceso de la comunicación. La señal de salida DAC_CLR es el reinicio del DAC, regularmente se encuentra en estado alto y se activa con un pulso de nivel lógico bajo de la señal de entrada CLEAR. La salida MOSI es donde se transmiten los datos para configurar el DAC, en ella se envía el comando, la dirección y los datos para la conversión. Por último, se tiene las señales que desactivan los componentes conectados a las líneas SPI, éstas son: SPI_SS_B, SF_CE0, FPGA_INIT_B, AMP_CS y CONV.

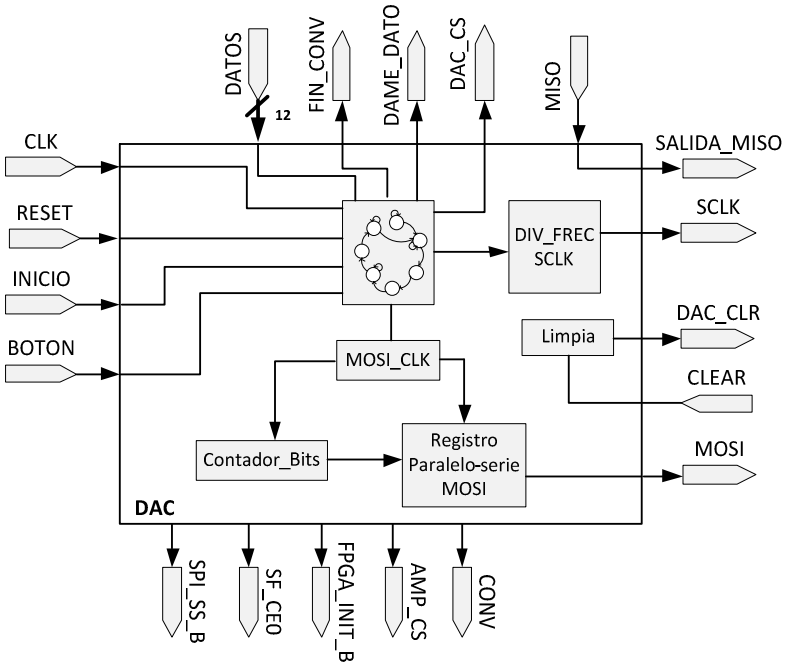


Figura 3.25. Control para el DAC.

Esta descripción se verificó mediante la simulación que se muestra en la figura 3.26. En ella se observan todas las señales involucradas. Con un pulso en la señal INICIO, de mínimo del doble del periodo de la señal CLK, se inicia la comunicación. No se realiza ningún reinicio con la señal RESET. Con la señal DAME_DATO, en estado lógico alto, se solicita el dato para la conversión. Esta señal se mantiene en estado lógico alto hasta que la señal BOTON, la cual indica que los datos están listos para ser ingresados, toma un valor lógico alto. La señal DATOS son los datos a ingresar para realizar la conversión. La señal FIN_CONV indica el final de la conversión mediante un estado lógico alto. La señal DAC_CS toma un valor lógico bajo cuando se inicia la comunicación con el DAC, cuando termina la comunicación la señal DAC_CS toma un valor lógico alto. La señal SCLK de 32 pulsos de reloj, se genera cuando DAC_CS se encuentra en nivel bajo. Los datos en la señal MOSI son: ocho bits de relleno con valor “00000000”, el comando con valor de “0011”, la dirección del DAC con valor “1111” (tabla 3.10), los datos de conversión con valor igual a la señal DATOS, “010011001000” y cuatro bits de relleno con valor de “0000”. Todos los datos de la señal MOSI están definidos con respecto al flanco de subida de la señal SCLK. La señal de entrada MISO se simuló con una serie de pulsos arbitrarios, los cuales coinciden con la señal de salida SALIDA_MISO. El reinicio por hardware, DAC_CLR, permanece a un valor lógico alto durante toda la conversión, ya que no hay cambios en la señal de entrada CLEAR, la cual controla el reinicio por hardware. Las señales para desactivar los otros componentes en las líneas SPI permanecen a un valor fijo durante toda la conversión, las señales SPI_SS_B, SF_CE0, FPGA_INIT_B y AMP_CS permanecen a un nivel lógico alto mientras que la señal CONV permanece en un valor lógico bajo.

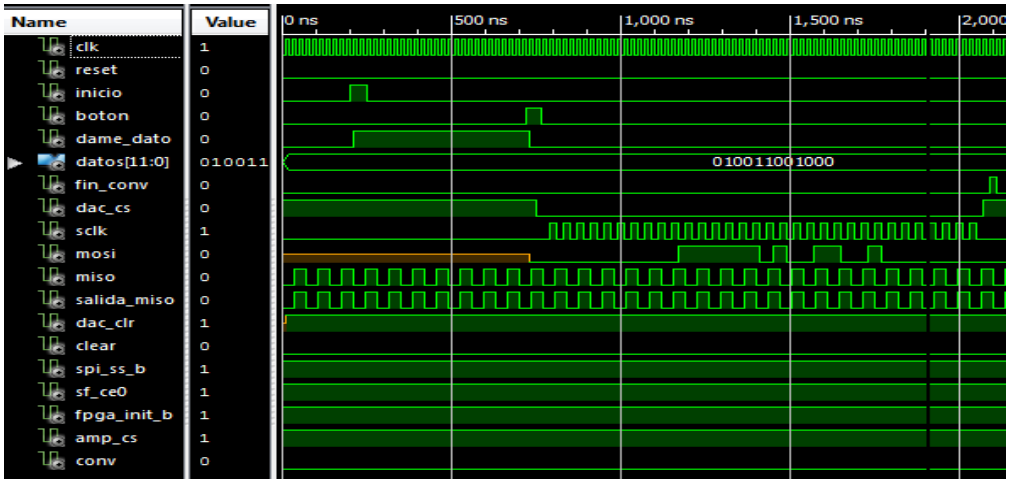


Figura 3.26. Simulación del control del DAC.

Para saber si la descripción diseñada funciona adecuadamente, se tiene que conocer la relación entre la entrada digital y la salida analógica. Esta relación está dada por[1]:

$$V_{salida(ideal)} = \frac{D}{2^n} * \left[V_{ref} - V_{refLO}\right] + V_{refLO} \qquad (3.11)$$

donde:

Vsalida(ideal) = voltaje de salida analógico ideal del DAC
D = valor decimal del dato de entrada para la conversión
n = número de bits del DAC
Vref = voltaje de referencia del DAC
VrefLO = voltaje cuando la entrada está en cero

Con n=12 y como el voltaje VrefLO es cero, la ecuación 3.11 se simplifica a:

$$V_{salida(ideal)} = \frac{D}{2^{12}} * V_{ref} \qquad (3.12)$$

Los DACs A y B tienen un Vref de 3.3V, mientras que los DACs C y D tienen un Vref de 2.5V (figura 3.23), la relación entre la entrada y la salida de los DACs A y B es:

$$V_{salidaAB} = \frac{D}{4096} * 3.3V \qquad (3.13)$$

[1] Especificaciones del circuito integrado LTC2624, Linear Technology, LT 0309 REV D 2004.

y para los DACs C y D:

$$V_{salidaCD} = \frac{D}{4096} * 2.5V \tag{3.14}$$

Al implementar la descripción en el FPGA no se tuvieron problemas, ya que la salida de los DACs se actualizaba al indicar el fin de la conversión. Al medir las señales con el osciloscopio no se encontró problema, ya que se apreciaban sin distorsión alguna. Algunas señales que se midieron, figura 3.27, fueron SCLK en amarillo y MOSI en azul. Se observa que no hay ninguna distorsión y concuerda con la simulación.

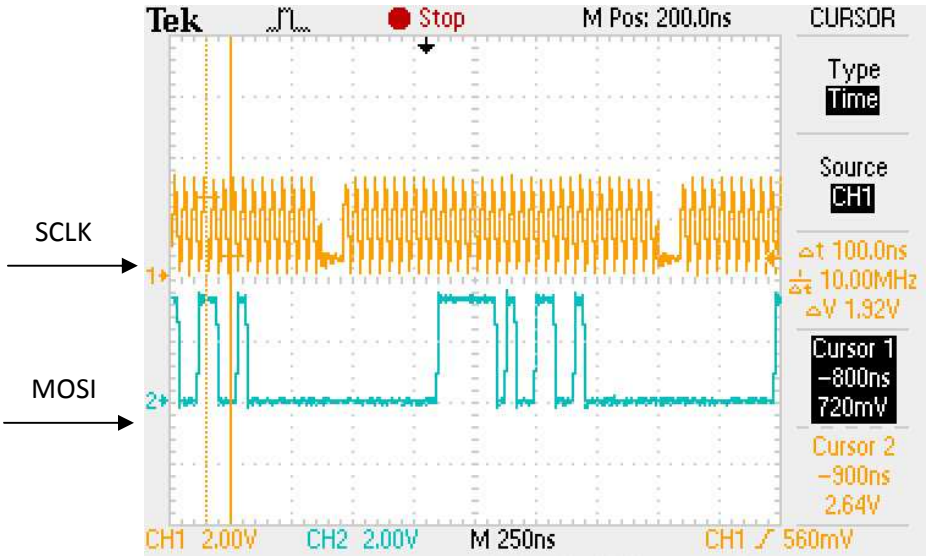


Figura 3.27. SCLK en amarillo y MOSI en Azul.

Al comprobar que la descripción funcionaba correctamente, se procedió a enviar datos y medir el voltaje de salida. La tabla 3.11 muestra los resultados obtenidos para los diferentes valores enviados. El valor decimal, columna uno, es el valor que se ingresó al FPGA mediante las entradas de propósito general, el valor es ingresado en binario pero, por comodidad en la visualización y en las operaciones, se maneja en decimal. El equivalente en binario y que es ingresado al FPGA de forma paralela se muestra en la columna dos. El voltaje teórico, calculado con las ecuaciones 3.13 y 3.14 para los canales A, B y C, D respectivamente, se muestra en la columna 3. Los voltajes de referencia mostrados son los medidos en la tarjeta SPARTAN 3E. La columna cuarta, *Práctico*, es el voltaje medido en cada una de las salidas de los DACs.

Como se observa en la tabla 3.11, los voltajes de salida corresponden con los voltajes teóricos calculados con las ecuaciones 3.13 y 3.14.

Valor decimal	Valor binario	Teórico		Práctico			
		A y B Vref=3.294	C y D Vref=2.575	A	B	C	D
500	0001 1111 0100	0.4021	0.3146	0.4006	0.4005	0.3126	0.3149
1000	0011 1110 1000	0.8042	0.6287	0.8024	0.8025	0.6272	0.6293
1500	0101 1101 1100	1.2063	0.943	1.201	1.201	0.9414	0.944
2000	0111 1101 0000	1.6084	1.2573	1.603	1.604	1.253	1.255
2500	1001 1100 0100	2.0105	1.5717	2.005	2.006	1.567	1.57
3000	1011 1011 1000	2.4126	1.886	2.407	2.408	1.882	1.884
3500	1101 1010 1100	2.8147	2.2003	2.81	2.811	2.197	2.199
4000	1111 1010 0000	3.2168	2.5146	3.213	3.214	2.512	2.514
4095	1111 1111 1111	3.2932	2.5744	3.288	3.289	2.517	2.573

Tabla 3.11. Voltajes de salida en los DACs.

Si bien, existe una diferencia entre los valores teóricos y los prácticos, esta diferencia no es mayor a cinco milivolts. Calculando el error relativo para cada canal, tabla 3.12, se observa que no llega al 1%, concluyendo que la descripción diseñada funciona correctamente.

% error			
A	B	C	D
0.37%	0.4%	0.54%	0.2%
0.22%	0.21%	0.24%	0.1%
0.44%	0.44%	0.17%	0.1%
0.34%	0.27%	0.34%	0.18%
0.27%	0.22%	0.3%	0.11%
0.23%	0.19%	0.21%	0.10%
0.17%	0.13%	0.15%	0.06%
0.12%	0.09%	0.1%	0.02%
0.16%	0.13%	0.13%	0.05%

Tabla 3.12. Error relativo para los DACs.

3.3.4. Comunicación para el display LCD

La tarjeta de desarrollo SPARTAN 3E 1600E cuenta con un display LCD 2x16, de 16 terminales con una interfaz de 4 u 8 bits. La conexión entre el FPGA y el display LCD se muestra en la figura 3.28. Cuando se interactúa con el display, se tiene que desactivar la memoria StrataFlash debido a que comparten líneas de datos. La desactivación de la memoria se realiza con un valor lógico alto en la señal SF_CE0.

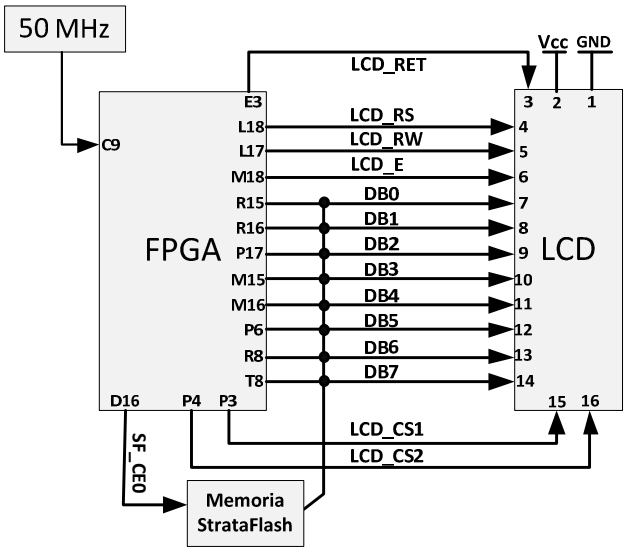


Figura 3.28. Conexión entre el FPGA y el LCD.

El diagrama de tiempos para controlar el dispyal LCD se muestra en la figura 3.29. Se observan los tiempos mínimos para realizar una comunicación exitosa con una interfaz de 4 bits, en donde se envía primero en *nibble* alto y posteriormente el *nibble* bajo del dato.

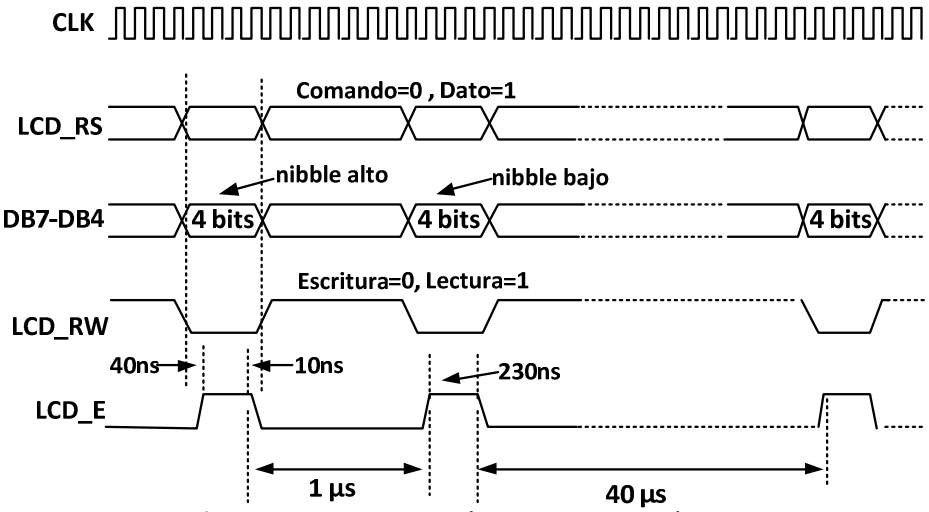


Figura 3.29. Diagrama de tiempos para el LCD.

En la figura 3.30 se muestran los pasos a seguir para inicializar y configurar el display LCD en la tarjeta SPARTAN 3E 1600E con una interfaz de 4 bits.

La descripción implementada para el control del display LCD está basada en descripciones ya verificadas. Estas descripciones se modificaron para obtener la descripción mostrada, a nivel diagrama de bloques, de la figura 3.31. Consiste en cuatro máquinas de estado, donde la PRINCIPAL controla las otras tres. La máquina de estados llamada INICIALIZA controla la inicialización del display. La máquina de estados TRANSMITE controla la transmisión de datos, donde primero se envía el *nibble* alto y después el *nibble* bajo. La máquina de estados llamada

ESCRIBE controla los datos mostrados en el LCD. Las señales CLK y RESET son las que sincronizan y reinician, respectivamente, los procesos de control dentro del FPGA. La señal LED, de 8 bits, es para corroborar el dato que se está enviando al LCD. En la señal DB[7:4], de 4 bits, se transmiten tanto datos como comandos para el display. La señal LCD_E es el pulso de habilitación para los datos y comandos. La señal LCD_RW indica si se escriben o se leen datos del LCD. La señal LCD_RS es la que indica al LCD si el dato a enviar es un comando o un dato a desplegar. Por último está la señal SF_CE0, la cual desactiva la memoria StrataFlash.

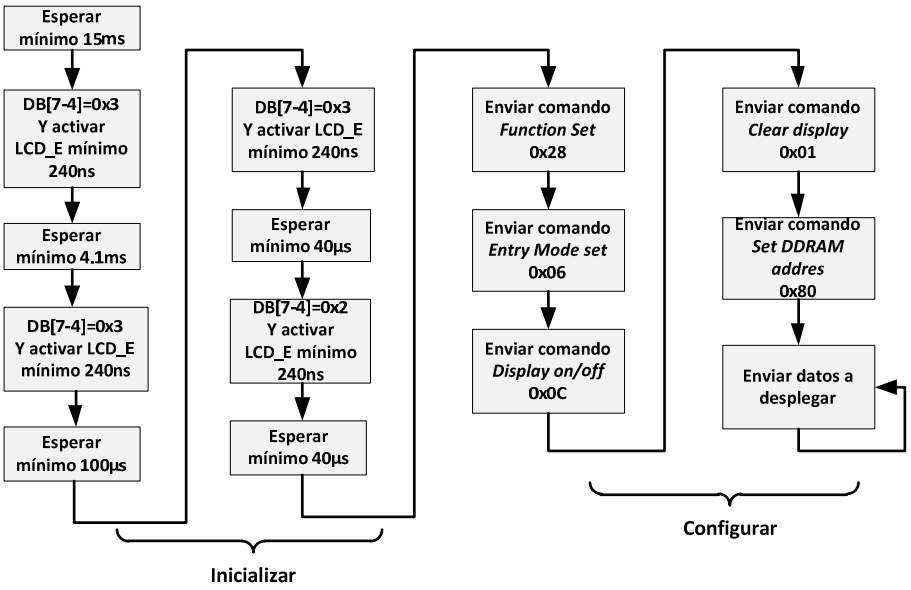


Figura 3.30. Diagrama de flujo para el LCD.

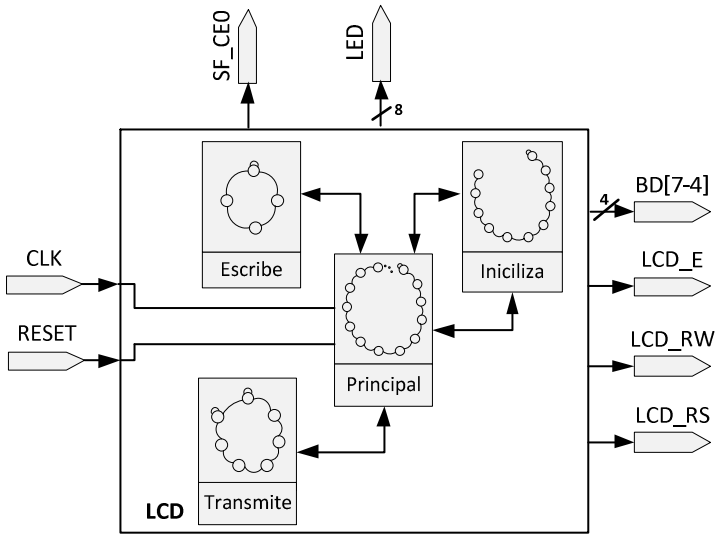


Figura 3.31. Control para el display LCD.

Esta descripción se simuló y se obtuvieron los resultados mostrados en la figura 3.32. Se observan todas las señales involucradas en la descripción. La señal CLK es el reloj que sincroniza todos los procesos. No se hace reinicio durante la transmisión de datos, por lo tanto, la señal RESET está en nivelo bajo. En DB[7:4] se observa, mayoritariamente, el *nibble* bajo de cada dato y comando. En LCD_E se perciben los pulsos de habilitación, de 260ns, para los datos y comandos. En la señal LCD_RS se observa cuando se envía un comando, nivel lógico bajo y cuando un dato, nivel lógico alto. El comando enviado es la posición en donde se empiezan a desplegar los datos. La señal LCD_RW se mantiene a un valor lógico bajo constate, ya que sólo se escribe en el display. Por último, la señal SF_CE0 se mantiene a un valor lógico alto para desactivar la memoria StrataFlash.

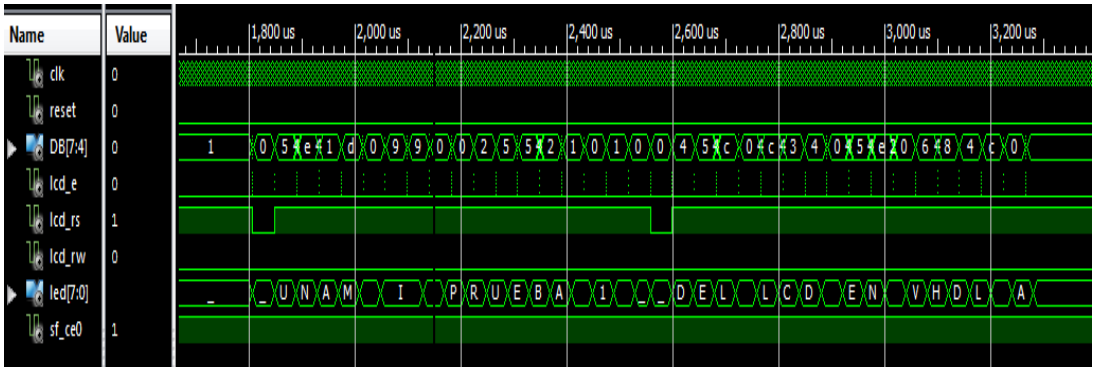


Figura 3.32. Simulación del control para el display LCD.

El resultado de la implementación del control del LCD se muestra en la figura 3.33, donde se muestran los caracteres enviados. Se observa que los datos en la simulación son los mismos que los presentados en el display.

Figura 3.33. Escritura en el display LCD.

3.3.5. Análisis de lo diseñado

Hasta este momento se tiene controlado cuatro de los cinco periféricos propuestos: ADC, DAC, display LCD y entradas/salidas de propósito general. Si bien, los controles diseñados funcionan correctamente, su comportamiento está diseñado para operar de manera individual. Al momento de integrar estos cuatro controles, se presentaron ciertas cuestiones en cuanto a la filosofía de diseño que hasta el momento se estaba utilizando. La primera de estas, fue que para integrar los controles, tenían que modificarse, es decir, adaptarlos entre sí. La segunda cuestión, y la más

importante, fue cómo hacer dicha integración. Por una parte, la integración se puede realizar exclusivamente entre los controles diseñados, todos tienen comunicación con todos. Una representación, en diagrama de bloques, de esta integración se muestra en la figura 3.34, en ella se observan las líneas sólidas que conectan los controles desarrollados entre sí y en líneas punteadas las conexiones que se deben agregar para incorporar un control extra, además de los bloques necesarios, marcados con "¿?", para adaptar los controles entre ellos. La principal desventaja con esta forma de integración es que si se agrega cualquier otro control todo lo diseñado se tiene que adaptar para incorporarlo. La principal ventaja es que es un diseño exactamente a la medida con las interacciones necesarias del diseño.

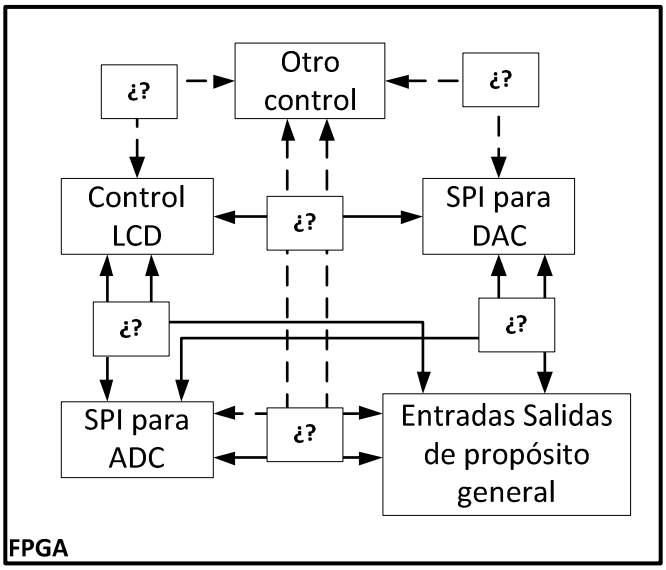


Figura 3.34. Primera forma de integrar los controles diseñados.

Otra manera de realizar la integración es incorporar un control que maneje a todos los controles, tanto a los diseñados como a los adicionales. Una representación, en diagrama de bloques, de esta integración se muestra en la figura 3.35, en donde se observa que el control general interactúa con los controles diseñados, líneas sólidas, y con los controles adicionales, líneas punteadas. La principal desventaja es que sería un control general, y por lo tanto, su diseño e implementación requiere de una lógica más compleja y un tiempo de desarrollo largo. La principal ventaja es que se puede incorporar controles sin necesidad de modificar los demás, expandiendo así las capacidades de conexión del sistema, aumentando los controles sin modificación alguna.

Al plantear estas dos opciones se puede cambiar radicalmente la filosofía de diseño, y el diseño mismo, que hasta el momento se venía manejando. Integrar los controles comunicándose entre ellos, sería seguir con la filosofía de diseño utilizada hasta el momento. Con la integración de los controles mediante un control general es donde se puede cambiar la filosofía de diseño ya que se puede utilizar los sistemas embebidos para cumplir esta integración. La estructura de los sistemas embebidos es muy similar a la segunda forma de integración, en donde un procesador, el control general, gestiona y administra los periféricos que se conectan a él. Utilizando los sistemas embebidos se cambia la filosofía de diseño ya que se utilizan los IP, ó núcleos; los diseños ya no se

construyen desde cero. Los IPs pueden ser desarrollados por cualquier empresa de diseño digital ó en dado caso se pueden desarrollar por uno mismo partiendo de ciertas bases. Utilizar los sistemas embebidos reduce el tiempo de diseño porque existe gran variedad de IPs que se pueden utilizar sin la necesidad de conocer cómo están implementados y conociendo exclusivamente cómo funcionan, esto, sin perder de vista que están construidos con lenguajes de descripción de hardware. La gran ventaja de los sistemas embebidos es el corto tiempo de desarrollo.

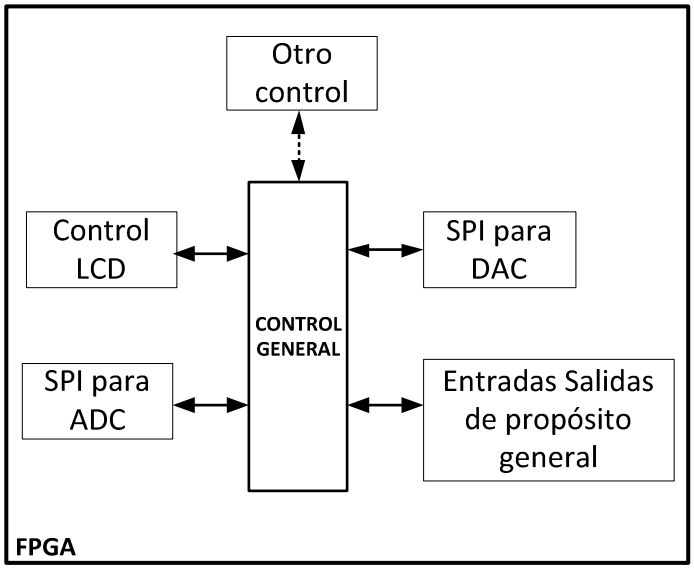


Figura 3.35. Segunda forma de integrar los controles diseñados.

Los IP suponen una gran ventaja en el desarrollo de sistemas embebidos, en especial con la creciente capacidad de recursos integrados en los recientes FPGAs. A continuación se enlistan algunas ventajas de los sistemas embebidos:

- Tiempos de desarrollos cortos
- Gran variedad de IPs con gran cantidad de funcionalidades
- Capacidad de crear IPs propios
- Herramientas de gran capacidad
- Se comienza a diseñar de una base ya elaborada

Con las ventajas ofrecidas por los sistemas embebidos se decidió cambiar la filosofía de diseño que hasta el momento se estaba manejando, dando así, origen a la segunda aproximación para el desarrollo del presente trabajo de tesis.

3.4. Segunda aproximación al sistema de monitoreo y control

Ya con una filosofía de diseño nueva, se tiene que conocer cómo y con qué se realizan los sistemas embebidos en un FPGA, por lo tanto, se dará una breve explicación de las herramientas ofrecidas por Xilinx para dicha filosofía de diseño.

3.4.1. Herramienta para sistemas embebidos

Una herramienta que ofrece Xilinx para los sistemas embebidos, la cual se utilizó, es EDK: *Embedded Development Kit*. EDK es un conjunto de herramientas e IPs que se pueden utilizar para diseñar e implementar un sistema embebido muy completo.

EDK se divide principalmente en dos partes: XPS, *Xilinx Platform Studio* y SDK, *Software Development Kit*. En XPS es donde se desarrolla la parte hardware del sistema embebido, es decir, se especifica el procesador, los periféricos y las conexiones entre ellos. SDK es donde se crea y verifica la aplicación software utilizando C/C++.

Se puede manejar XPS y SDK de dos maneras: una, administrados por ISE Design Suite y otra, independientes de ISE Design Suite. La interacción entre ISE Design Suite, XPS y SDK se ilustra en la figura 3.36.

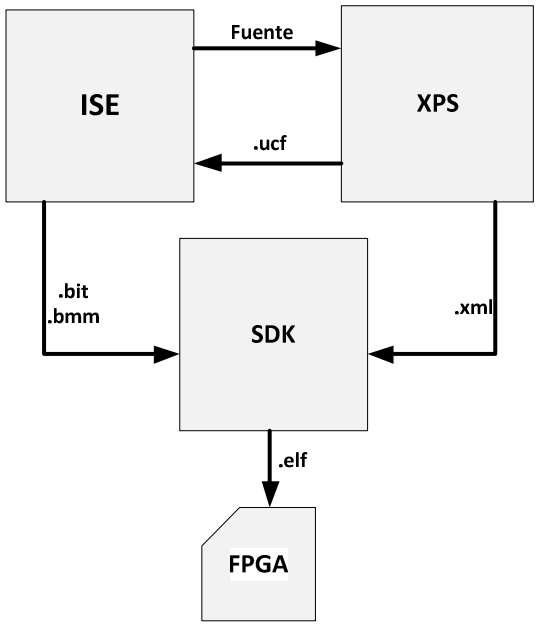


Figura 3.36. ISE, XPS y EDK interactuando.

ISE Design Suit se utiliza como administrador que manda llamar, ya sea a XPS y/o a SDK. Cada software tiene sus archivos de entrada y de salida. En la figura 3.36 se observan algunos de estos archivos, por ejemplo, en ISE Design Suite se crea un archivo fuente, el cual manda llamar automáticamente a XPS. En XPS se crea el hardware, y uno de los archivos de salida es el .ucf donde se enlistan las terminales del FPGA a utilizar. Este archivo es utilizado por ISE Design Suite para generar los archivos .bit y .bmm, el *bitstream* y las especificaciones de memoria, respectivamente, que son utilizados por SDK. Otro archivo de salida de XPS es el .xml, el cual es un archivo de especificaciones del hardware que también utiliza SDK. Por último, SDK genera el archivo .elf, donde se encuentra la aplicación software la cual se descarga al FPGA. También se pueden manejar de manera independiente sin necesidad de administrar con ISE Design Suite. En

este caso, los archivos se cargan manualmente en cada software pero el funcionamiento y los archivos de salida son los mismos. Ya que se conocen las herramientas y la interacción entre ellas, se procedió a trabajar con ellas.

En la mayoría de los proyectos embebidos que se realizaron se utilizó a ISE Design Suite como administrador. En ISE Design Suite se creó un archivo fuente para sistemas embebidos, el cual abre automáticamente XPS para realizar el hardware del sistema. Ya que se ha creado el hardware necesario, se regresó a ISE Design Suite para generar el *bitstream* y las especificaciones de la memoria. Una vez que se han generado el *bitstream* y las especificaciones de memoria, se inició SDK de manera manual (del menú de Windows: inicio→todos los programas→Xilinx ISE Desig Suite 13.2 →EDK→ Xilinx Software Development Kit). En SDK se creó un proyecto y se cargó las especificaciones del hardware con el que se va a trabajar. Por último, en SDK se configuró el FPGA con el *bitstream* y con la aplicación software desarrollada.

3.4.2. MicroBlaze

El primer IP que se estudió fue el del procesador. Debido a los recursos del FPGA el único procesador disponible en la tarjeta SPARTAN 3E 1600E es MicroBlaze. El procesador embebido MicroBlaze es un núcleo *soft* de instrucciones reducidas (RISC: *Reduced Instruction Set Computer*) optimizado para FPGAs de Xilinx. Es altamente configurable permitiendo seleccionar un conjunto específico de características para el diseño requerido. Entre estas características se encuentra una unidad de punto flotante (FPU: *Floatin Point Unit*), *pipeline* de 3 o 5 etapas, unidad de desplazamiento por hardware, multiplicadores y divisores por hardware, entre otras. Algunas de las características fijas de MicroBlaze son: 32 registros de 32 bits cada uno, longitud de palabra de 32 bits y bus de direcciones de 32 bits. En la figura 3.37 se muestra un diagrama de bloques de los componentes de MicroBlaze.

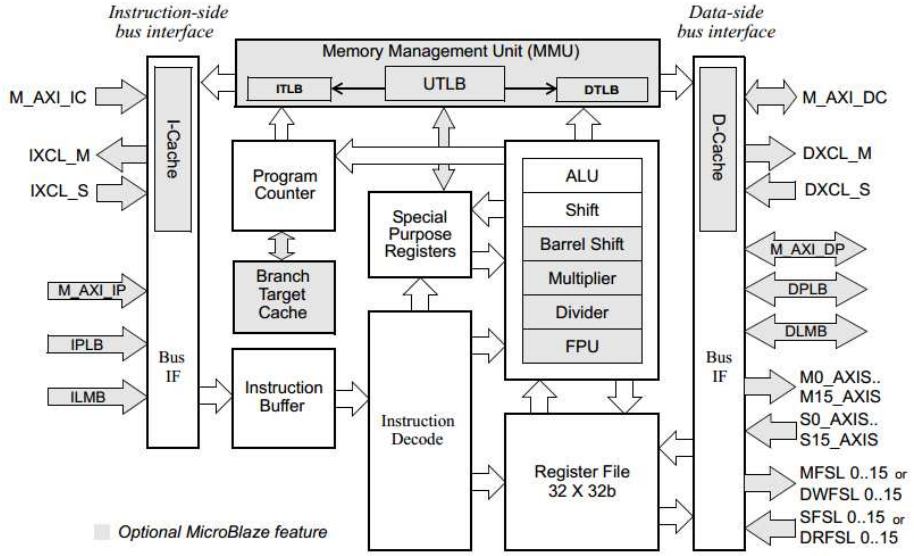


Figura 3.37. Diagrama de bloques de MicroBlaze.

En XPS es donde se realizan las configuraciones necesarias a MicroBlaze. Cuando se inicia XPS se presenta el asistente *Base System Builder,* con el cual se configura de manera rápida y eficiente un diseño hardware, incluyendo MicroBlaze. En la figura 3.38 se muestra la pantalla de inicio de XPS, en donde se selecciona la opción del asistente para iniciar con la construcción del sistema embebido.

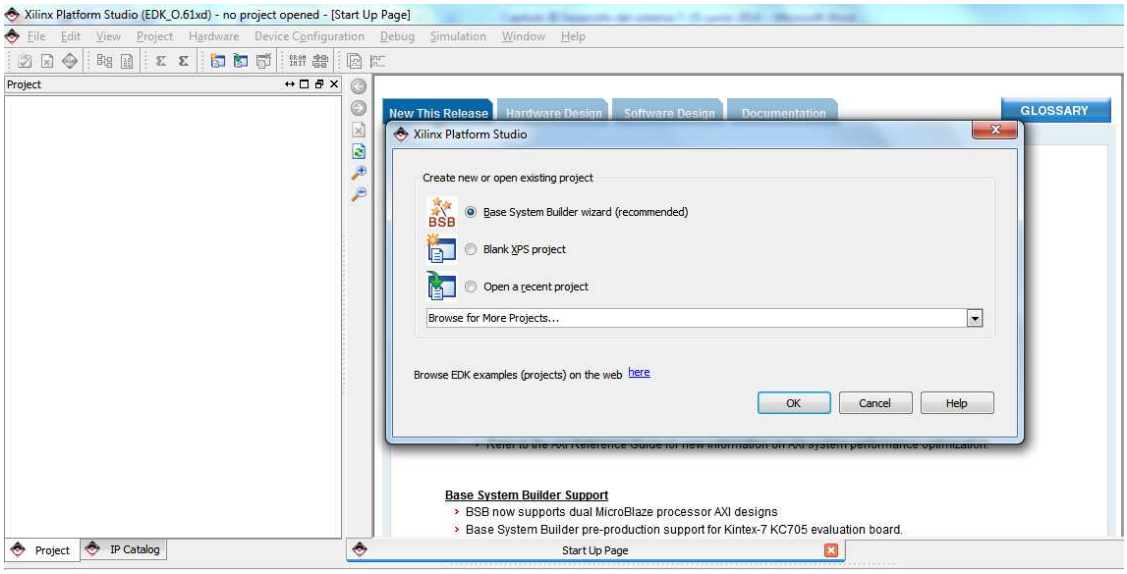


Figura 3.38. Pantalla de inicio de XPS.

Algunas características que se configuran con el asistente son: la tarjeta de desarrollo a utilizar, qué procesador y cuántos se van a utilizar (uno o dos procesadores), el reloj del sistema, el tamaño de la memoria local y algunos periféricos que contenga la tarjeta. En la figura 3.39 se muestra la pantalla en donde se selecciona la tarjeta a utilizar, en la figura 3.40 se muestra en donde se selecciona el procesador a utilizar y en la figura 3.41 se muestra la pantalla donde se seleccionan los periféricos con los que cuenta la tarjeta de desarrollo seleccionada.

Ya que se ha configurado MicroBlaze, XPS crea las conexiones entre los IPs esenciales para el funcionamiento de un sistema embebido básico, figura 3.42. Existen siete IPs básicos: DLMB, ILMB, MB_PLB, MICROBLAZE_0, MDM_0, CLOCK_GENERATOR_0 y PROC_SYS_RESET_0, que a continuación se definen y describen:

DLMB e ILMB: *Data Local Memory Bus* e *Instruction Local Memory Bus*. Son unos buses locales para conectar los puertos de datos y los puertos de instrucciones a Microblaze, regularmente en bloques de memoria RAM del FPGA. Los IPs LMB_BRAM, DLMB_CNTLR e ILMB_CNTLR son auxiliares que hacen posible esta conexión.

MB_PLB: *Processor Local Bus*. El PLB propociona una infraestructura para la conexión de un número opcional de maestros y esclavos, se encarga de gestionar y adminstrar la comunicación entre los IPs conectados.

MICROBLAZE_0. El IP MicroBlaze es propiamente el procesador, el cual puede ser configurado de acuerdo a las necesidades del diseño.

MDM_0: *MicroBlaze Debug Module*. Es el IP que hace posible la depuración del sistema embebido por medio del protocolo JTAG.

CLOCK_GENERATOR_0: *clock generator*. Se utiliza para generar la arquitectura necesaria para las señales de reloj del sistema.

PROC_SYS_RESET_0: *Processor System Reset Module*. Se utiliza para activar y desactivar funciones. Con este IP se genera el reinicio del sistema.

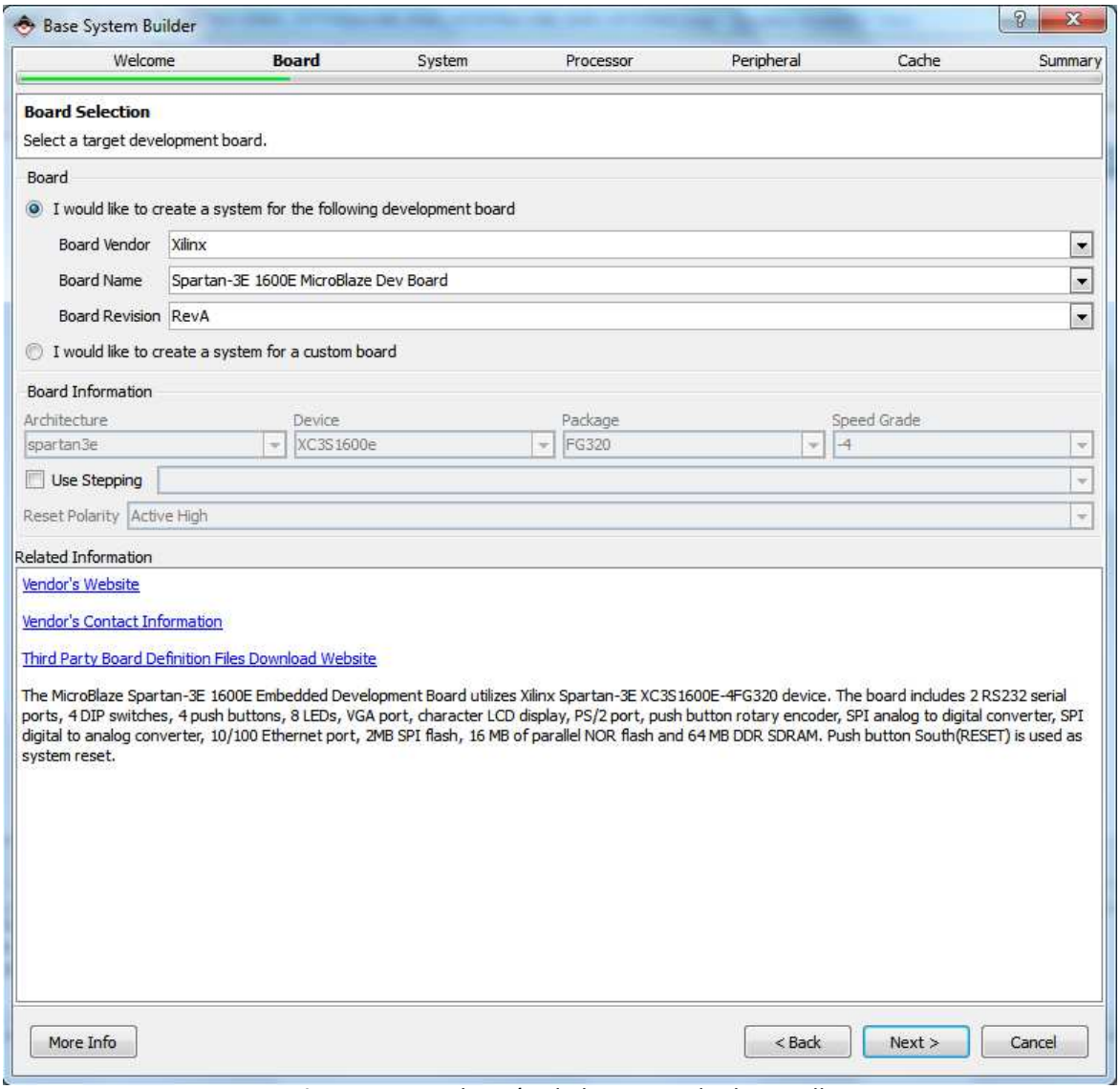


Figura 3.39. Selección de la tarjeta de desarrollo.

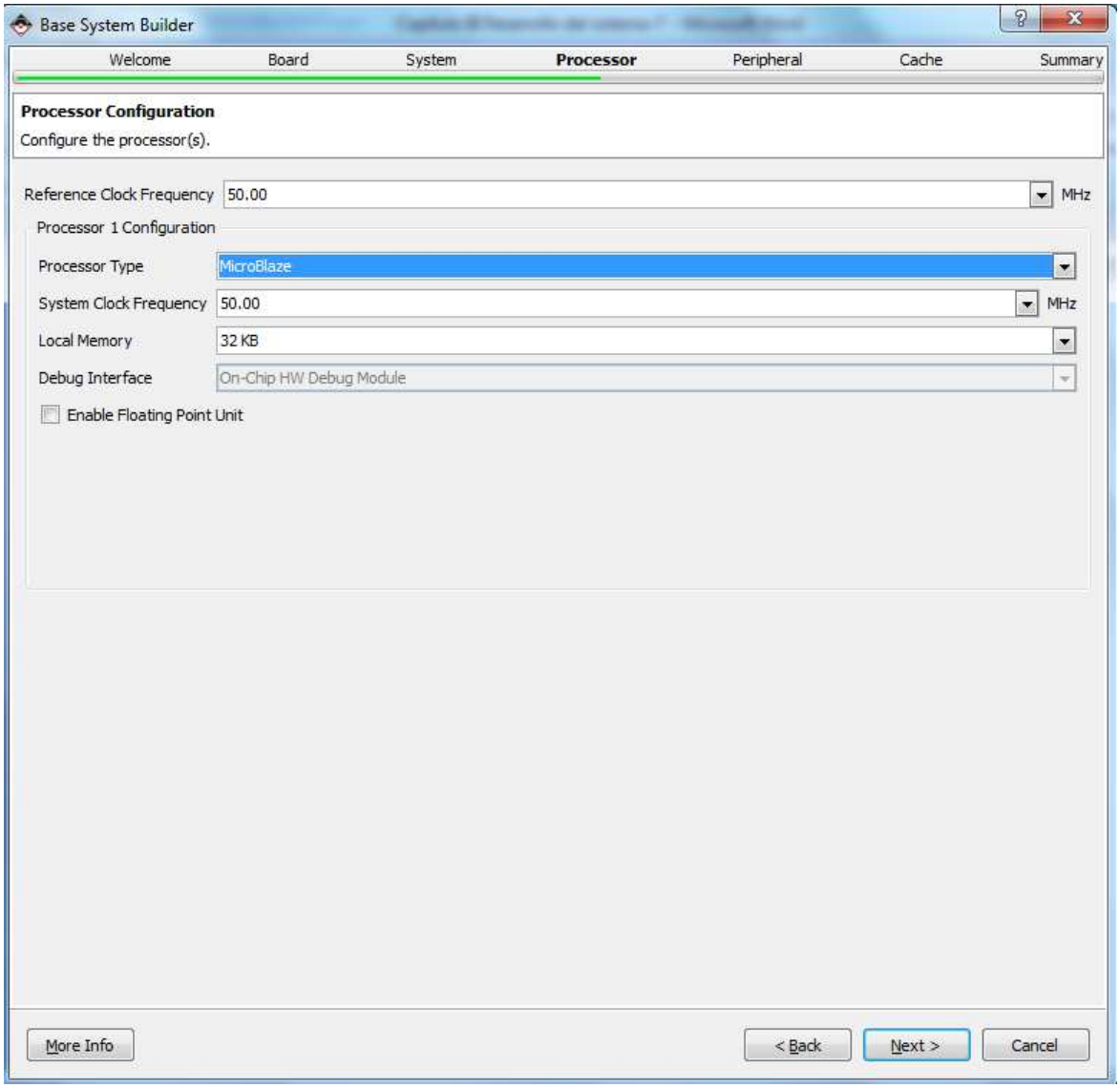


Figura 3.40. Selección del procesador a utilizar.

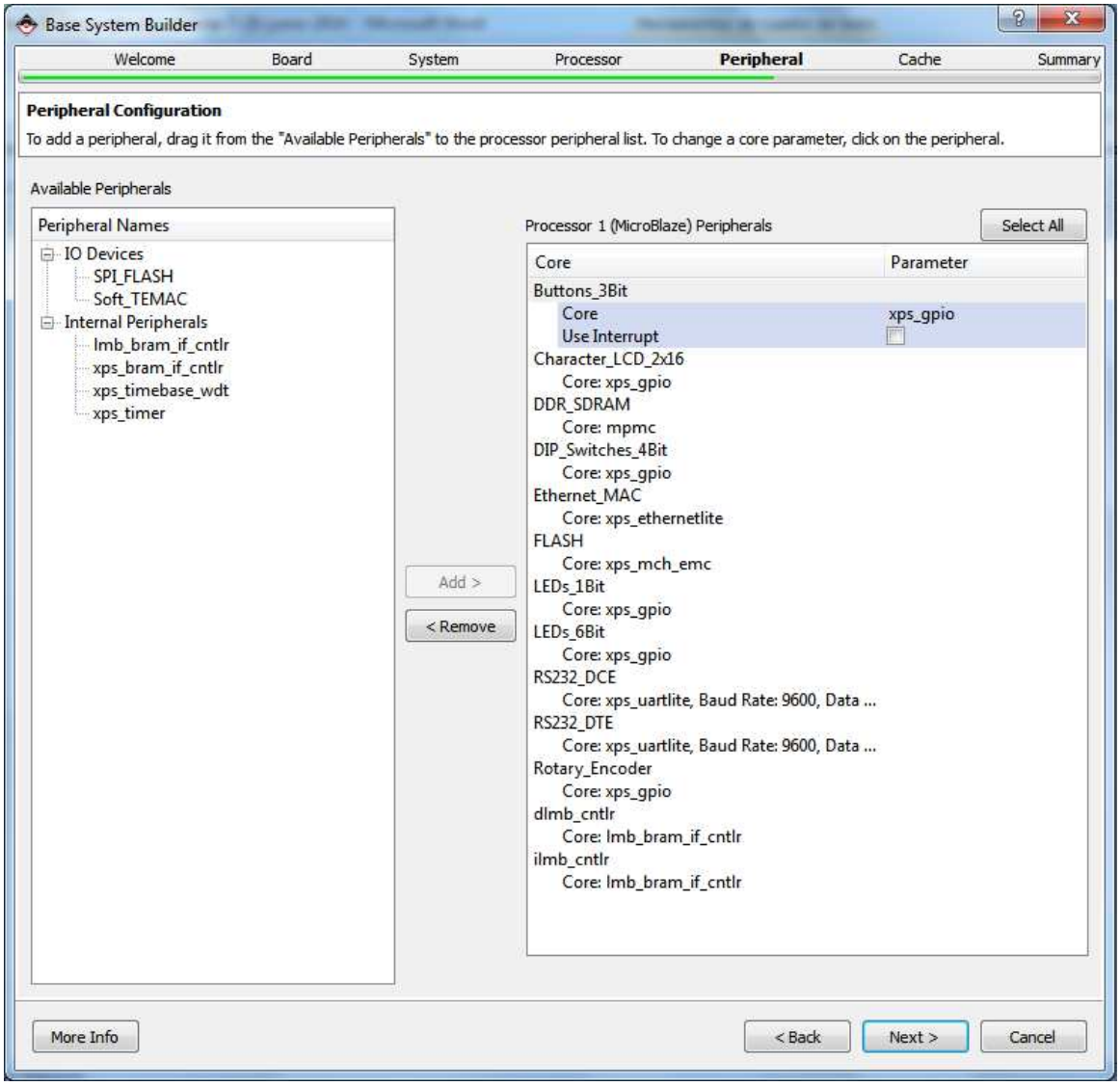


Figura 3.41. Periféricos de la tarjeta de desarrollo.

En la pantalla de la figura 3.42, además de mostrar las conexiones entre IPs, es en donde se agregan los periféricos con los que se va a comunicar MicroBlaze. Como ya se tiene una estructura básica de un sistema embebido, se empezará a conocer y agregar periféricos a este sistema.

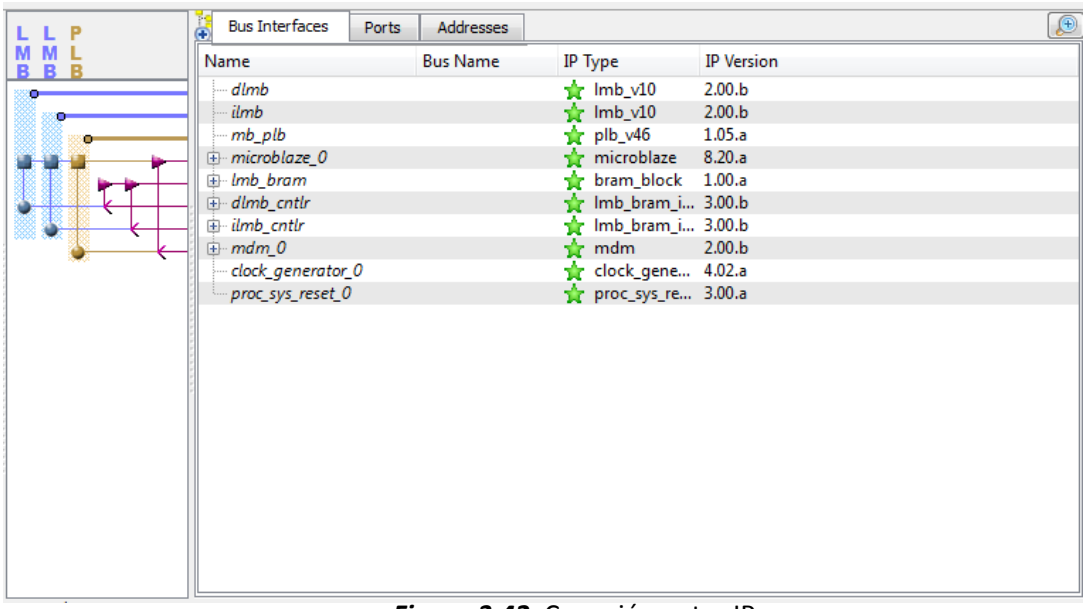


Figura 3.42. Conexión entre IPs.

3.4.3. Entradas y salidas de propósito general

El IP llamado XPS GPIO (*General Purpose Input/Output)* proporciona una interfaz de entradas/salidas de propósito general para la comunicación con el PLB. Algunas de sus características principales del XPS GPIO son: uno o dos canales de entradas/salidas, tamaño configurable de hasta 32 bits, configuración dinámica de entradas/salidas e interrupciones opcionales. El diagrama de bloques del IP se muestra en la figura 3.43, consta de tres partes principales: la interfaz para el PLB, el control de interrupciones y el núcleo GPIO.

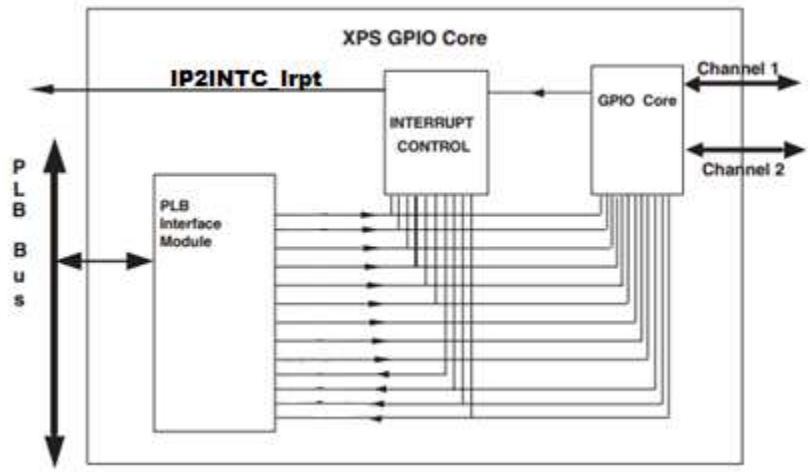


Figura 3.43. Diagrama de bloques del XPS GPIO.

La interfaz PLB proporciona la interacción entre el XPS GPIO y el bus PLB. El control de interrupciones es utilizado para recopilar las interrupciones del núcleo GPIO, con las cuales pide la atención del procesador. El núcleo GPIO consiste en un conjunto de registros y multiplexores para

escribir y leer los registros de los canales, ya sea canal 1, canal 2 ó ambos, también contiene la lógica necesaria para identificar una interrupción.

Existen principalmente tres señales para interactuar con este IP, las cuales son: IP2INTC_Irpt donde se indica que ha ocurrido una interrupción, GPIO_IO que son propiamente las entradas/salidas configurables para el canal 1 y GPIO2_IO que son propiamente las entradas/salidas configurables para el canal 2. Con estas tres señales es con las que interactúa el usuario, dejando un poco de lado las señales de la comunicación con el PLB.

El XPS GPIO tiene, como máximo, siete registros dependiendo de la configuración del mismo, tabla 3.13. En dicha tabla se muestra, en hexadecimal, la dirección de cada registro, partiendo de una dirección base (la dirección base se asigna con la herramienta XPS), y los permisos de cada registro, estos permisos son: escritura (e), lectura (l) o ambas (e/l).

Nombre del registro	Descripción	Dirección	Permisos
GPIO_DATA	Es el registro de datos para el canal 1	BASE+0x00	e/l
GPIO_TRI	Registro para configurar dinámicamente el canal 1 como entrada o salida	BASE+0x04	e/l
GPIO2_DATA	Es el registro de datos para el canal 2	BASE+0x08	e/l
GPIO2_TRI	Registro para configurar dinámicamente el canal 2 como entrada o salida	BASE+0x0C	e/l
GIER	Habilita la salida de interrupción general	BASE+0x11C	e/l
IP IER	Habilita la interrupción para los dos canales	BASE+0x128	e/l
IP ISR	Indica cuando ha ocurrido una interrupción.	BASE+0x120	e/l

Tabla 3.13. Registros del XPS GPIO.

Para configurar un bit como salida, se escribe un 0 en el bit correspondiente del registro GPIO_TRI. Después de configurado como salida, una escritura en el registro GPIO_DATA se refleja en los bits configurados como salida. Para configurar un bit como entrada, se escribe un 1 en el bit correspondiente del registro GPIO_TRI. Al leer el registro GPIO_DATA se reflejan las entradas en los bits configurados como entradas.

Ya que se conoce el funcionamiento y los parámetros importantes del XPS GPIO, se procederá a configurarlo para los diferentes dispositivos que se tienen, éstos son: los interruptores deslizables, los interruptores *push button*, los LEDs, la perilla giratoria y el display LCD.

Interruptores deslizables

Se cuenta con cuatro interruptores deslizables, los cuales siempre serán entradas al FPGA, el XPS GPIO se configura con un tamaño de 4 bits, sólo el canal 1 y únicamente entradas. También se le asigna un nombre, que en este caso es DIP_SWITCHES_4BIT. En la figura 3.44 se muestra una pantalla de la configuración de este IP para los interruptores deslizables.

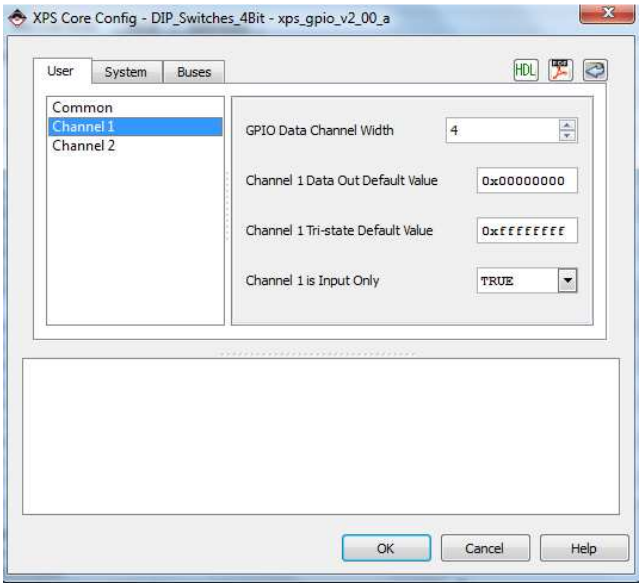


Figura 3.44. XPS GPIO para los interruptores deslizables.

Interruptores push button

La tarjeta SPARTAN 3E cuenta con cuatro interruptores *push button*. Al utilizar MicroBlaze uno de estos *push button* es utilizado como el reinicio (reset) de MicroBlaze, por lo que se cuenta únicamente con tres interruptores para conectar al XPS GPIO. El XPS GPIO se configura con un tamaño de 3 bits, sólo el canal 1 y únicamente entradas. Se le asigna el nombre de BUTTONS_3BIT.

LEDs

Se cuentan con ocho LEDs que se manejan con dos XPS GPIO, uno con 6 bits y el otro con 1 bit, el LED restante no se utiliza. La configuración de ambos XPS GPIO son: exclusivamente salidas y solamente con el canal 1. Los nombres asignados son LEDs_6Bit y LEDs_1Bit.

Perilla giratoria

La perilla giratoria contiene tres entradas, por lo que el XPS GPIO se configura con 3 bits y exclusivamente entradas. El nombre que se le asigna es Rotary_Encoder.

Display LCD

El display LCD cuenta con 3 bits de control y 4 bits para los datos. Aunque se pueden utilizar los 8 bits de datos, se utilizaron únicamente 4 bits. El XPS GPIO se configura con 7 bits, en donde se encuentran los bits para controlar las señales del LCD, estas señales son: LCD_E, LCD_RS, LCD_RW, DB7, DB6, DB5 y DB4. Se utiliza el canal 1 exclusivamente como salidas.

Otros dispositivos

Para desactivar los dispositivos conectados a las líneas SPI se utiliza un XPS GPIO, en donde cada bit desactiva cada componente. En este mismo XPS GPIO se colocan las terminales adicionales del amplificador programable, del ADC y del DAC. Las señales son: DAC_CLR, AMP_SHDN, AD_CONV, FPGA_INIT_B, SF_CE0, SPI_SS_B. El XPS GPIO se configura entonces con 8 bits, únicamente salidas y se le asigna el nombre de OTROS_DISPOS. Se agregan 2 bits más como posibles salidas extras.

Cabe mencionar que estos periféricos, a excepción de los otros dispositivos, se configuraron con el asistente *Base System Builder* y se mostraron las configuraciones que proporciona dicho asistente. Se puede configurar cada uno de estos componentes de forma manual si se quieren modificar los parámetros del asistente. En este caso la configuración que proporciona el asistente es suficiente para los propósitos del presente trabajo.

3.4.4. Comunicación SPI

El IP XPS SPI proporciona una interfaz para dispositivos compatibles con comunicación SPI. Algunas de sus características importantes son: conexión con el bus PLB, proporciona las cuatro señales SPI (MOSI, MISO, SCK y SS), frecuencia de SCK configurable, modo maestro o esclavo, fase y polaridad programable, opción para enviar primero el bit más significativo o el menos significativo, longitud de transferencia de 8 bits, 16 bits o 32 bits, FIFO opcionales para la transmisión y recepción (*First Input First Output*, primera entrada primera salida). En la figura 3.45 se muestra un diagrama de bloques del XPS SPI. Se compone principalmente de cinco partes: interfaz PLB, registros SPI, registros de interrupción, FIFOs de transmisión y recepción y propiamente el módulo SPI.

La interfaz PLB proporciona la interacción entre el XPS SPI y el bus PLB. Los registros SPI controlan al módulo SPI y se comunican con la interfaz PLB. Los registros de interrupción controlan los eventos para las interrupciones del módulo SPI. El módulo SPI consiste en registros de corrimiento, un generador de señales de reloj (BRG: *Baud Rate Generator)* y la unidad de control. Las FIFOs opcionales están presentes en la transmisión y recepción con una longitud de 16 elementos.

Las principales señales del XPS SPI con las que se interactúa son las que se muestran en la tabla 3.14. Se observan las señales típicas: MOSI, MISO, SCK y SS (CS), para cuando el módulo es configurado como maestro o como esclavo. También se muestra la dirección de cada señal, ya sea salida o entrada, dependiendo de la configuración. Por ejemplo la señal SCK_I es la señal de reloj para cuando el XPS SPI es configurado como esclavo, mientras que la señal SCK_O es para cuando es configurado como maestro.

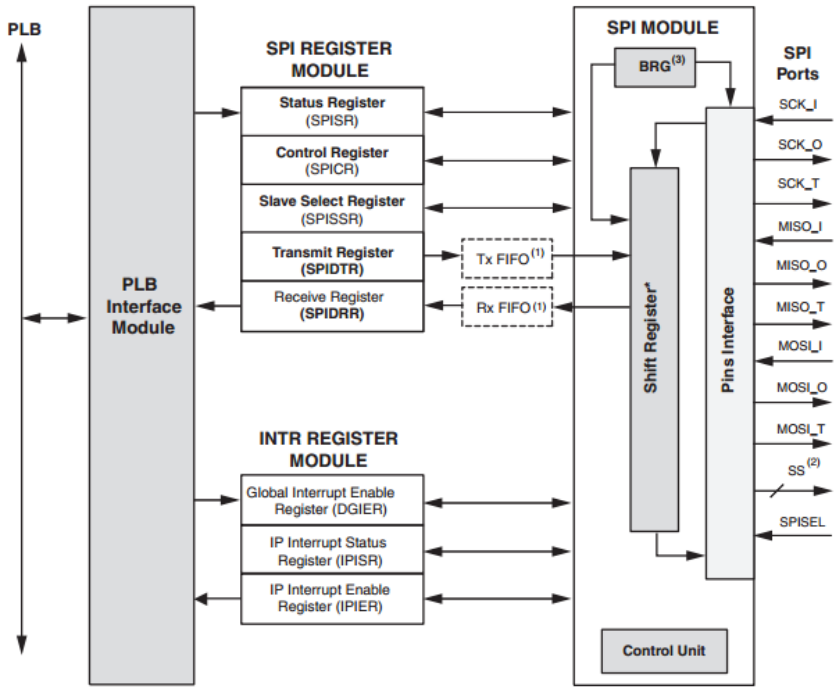


Figura 3.45. Diagrama de bloques de XPS SPI.

Nombre	Entrada/Salida	Descripción
SCK_I	Entrada	Señal de reloj (esclavo)
SCK_O	Salida	Señal de reloj
MOSI_I	Entrada	Señal Master Output Slave Input (esclavo)
MOSI_O	Salida	Señal Master Output Slave Input
MISO_I	Entrada	Señal Master Input Slave Output
MISO_O	Salida	Señal Master Input Slave Output (esclavo)
SPISEL	Entrada	Señal Chip Select cuando XPS SPI está en modo esclavo
SS_O	Salida	Señales Chip Select

Tabla 3.14. Señales principales para el XPS SPI.

El XPS SPI contiene once registros, que se describen en la tabla 3.15. En la tabla se muestra el valor de la dirección en hexadecimal de cada registro, partiendo de una dirección base (la dirección base se asigna con XPS) y se muestran los permisos con los que cuenta cada registro, ya se escritura (E), lectura (L) o ambos (E/L).

Para enviar y recibir datos de forma básica con el XPS SPI en modo maestro, se siguen los siguientes pasos:

- Inicializar el módulo SPI
- Configurar las interrupciones

- Escribir los datos a enviar
- Configurar las características del módulo SPI (modo maestro, polaridad, fase, bit MSB ó LSB para enviar, etcétera)
- Seleccionar el esclavo para comunicarse
- Habilitar la transmisión de datos
- Esperar a que la transmisión se complete
- Deshabilitar la transmisión y escribir los nuevos datos a transmitir
- Repetir los tres últimos pasos anteriores hasta completar los datos requeridos para enviar
- Desactivar el esclavo
- Deshabilitar el módulo SPI

Nombre del registro	Descripción	Dirección	Permisos
SRR	Permite el reinicio del IP XPS SPI por software	Base+40	E
SPICR	Se configura la polaridad, la fase, el bit MSB o LSB, el modo maestro o el esclavo, la habilitación del SPI, entre otros aspectos	Base+60	E/L
SPISR	Indica si la transmisión es completada, si las FIFO están llenas o vacías para la recepción y la transmisión de datos	Base+64	L
SPIDTR	Se escriben los datos que se quieren transmitir	Base+68	E
SPIDRR	Se leen los datos recibidos de la comunicación SPI	Base+6C	L
SPISSR	Contiene el vector de los dispositivos esclavos. El tamaño es configurable y puede ser de hasta 32 dispositivos esclavos	Base+70	E/L
SPI Transmit FIFO	Indica el número de elementos que existen en la FIFO de transmisión	Base+74	L
SPI Receive FIFO	Indica el número de elementos que existen en la FIFO de recepción	Base+78	L
DGIER	Se habilita globalmente la salida de interrupción	Base+1C	E/L
IPISR	Recoge todos los eventos de interrupción del SPI	Base+20	E/L
IPIER	Hablita las interrupciones que recoge el registro IPISR	Base+28	E/L

Tabla 3.15. Registros del XPS SPI.

Para controlar el amplificador, el ADC y el DAC se utilizó únicamente un módulo XPS SPI, llamado XPS_SPI_0. Las características del hardware se configuran de forma gráfica, como se observa en la figura 3.46, éstas son:

- No se incluyen FIFOs de recepción ni transmisión
- Señal de reloj del sistema divido entre 32 para generar la señal SCK
- 32 bits de transmisión
- 3 esclavos para la comunicación
- No se incluyen interrupciones

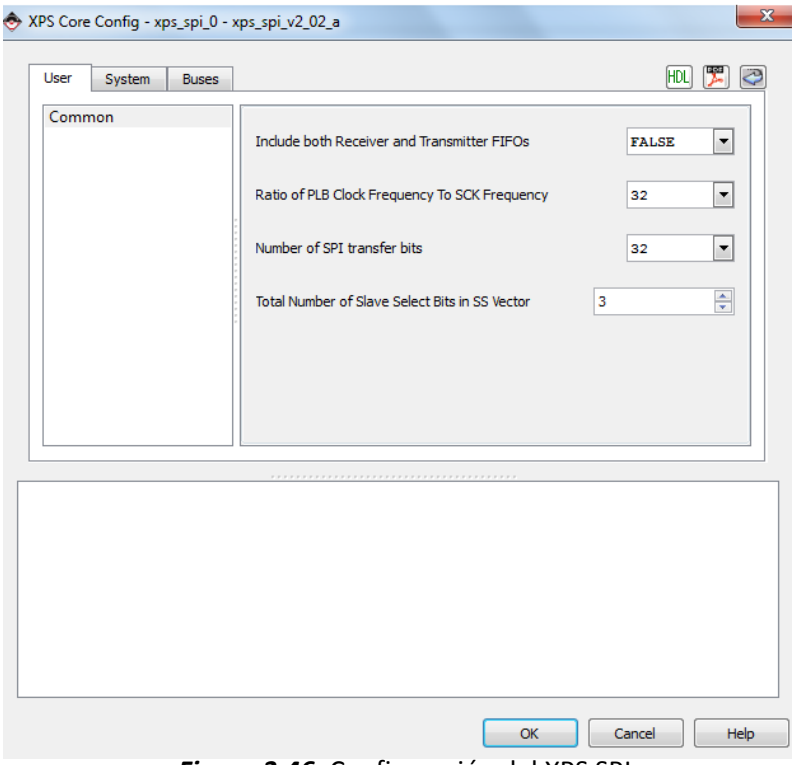


Figura 3.46. Configuración del XPS SPI.

Ya que se tiene configurado el hardware para la comunicación SPI, resta configurar los registros mediante la aplicación software para realizar la comunicación adecuadamente para cada dispositivo. Cabe señalar que las características del hardware, como el número de esclavos o el número de bits a transmitir, no se pueden modificar mediante software.

3.4.5. Comunicación asíncrona

XPS UART Lite es un IP que proporciona una interfaz para una comunicación asíncrona. Está diseñado para interactuar con el bus PLB. Sus principales características son: un canal de transmisión y un canal de recepción, FIFOs de 16 elementos para la recepción y transmisión, bits de datos, bit de paridad y velocidad (*baud rate*) configurables. En la figura 3.47 se muestra el

diagrama de bloques del XPS UART Lite, se compone de la interfaz PLB y de los módulos de registros y de control de la UART.

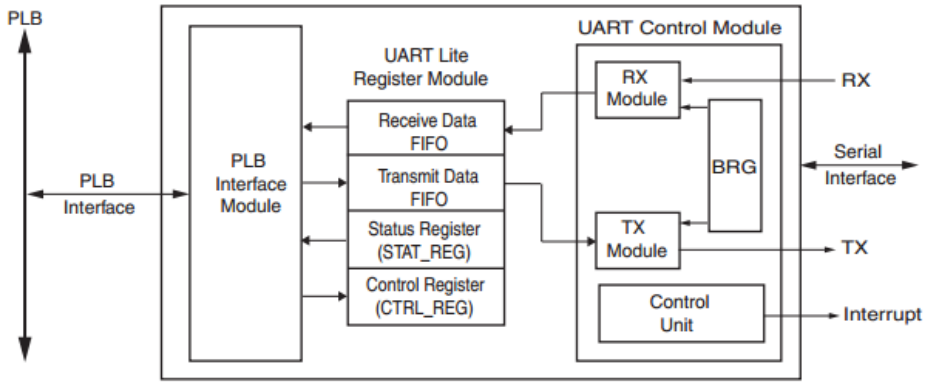


Figura 3.47. Diagrama de bloques UART Lite.

La interfaz PLB proporciona la interacción entre el XPS UART Lite y el bus PLB. El módulo de registros contiene todos los registros que controlan y monitorean al módulo UART, se accede a ellos por medio de la interfaz PLB. El módulo de control de la comunicación asíncrona (UART) consiste en un módulo de recepción, uno de transmisión y un generador de reloj (BRG); existen principalmente tres señales para interactuar con este IP, las cuales son: *Rx,* que es la señal de recepción; *Tx*, que es la señal de transmisión e *Interrupt* que es la señal de interrupciones.

El XPS UART Lite contiene cuatro registros que se describen en la tabla 3.16. En dicha tabla se muestra la dirección, en hexadecimal, de cada registro, partiendo de una dirección base que se asigna con XPS y también se muestran los permisos de escritura o lectura que tiene cada registro.

Nombre del registro	Descripción	Dirección	Permisos
Rx FIFO	Contiene los datos que son recibidos por el XPS UART Lite	BASE+0x00	Lectura
Tx FIFO	Contiene los datos que son transmitidos por el XPS UART Lite	BASE+0x04	Escritura
STAT_REG	Habilita las interrupciones y limpia los registros Rx y Tx	BASE+0x08	Lectura
CTRL_REG	Indica si las interrupciones están habilitadas, si están vacíos o llenos los registros Rx ó Tx, si existe error de paridad, si existe error en el bit de paro, entre otros.	BASE+0x0C	Escritura

Tabla 3.16. Registros del XPS UART Lite.

Para obtener el dato recibido de la comunicación asíncrona se lee el dato del registro *Rx* y para enviar un dato por la comunicación asíncrona se escribe el dato en el registro *Tx*.

La tarjeta SPARTAN 3e cuenta con dos puertos para comunicación UART llamados DCE y DTE. Cuenta también con el circuito integrado ICL3232 para adecuar los niveles de voltajes entre el

FPGA (3.3V) y el puerto RS-232 (±5V). Los parámetros que se configuran del XPS UART Lite para estos dos puertos son: datos de 8 bits, sin ningún tipo de paridad y *baud rate* de 9600. Estos parámetros se configuran de manera gráfica como se muestra en la figura 3.48.

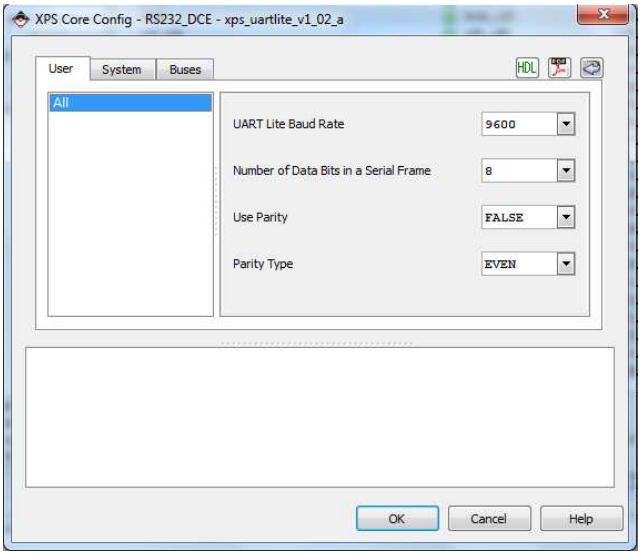


Figura 3.48. Configuración del XPS UART Lite.

3.4.6. Temporizador

El XPS Timer/Counter es un IP que proporciona un módulo de temporización (*timer*) de 32 bits. Sus principales características son: compatible con el bus PLB, captura de eventos, generación de eventos, salida de modulación por ancho de pulso (PWM: *Pulse Width Modulation*) y tamaño del contador configurable. En la figura 3.49 se muestra el diagrama de bloques del XPS Timer/Counter, en donde se aprecian dos componentes principales: la interfaz PLB y el módulo del temporizador.

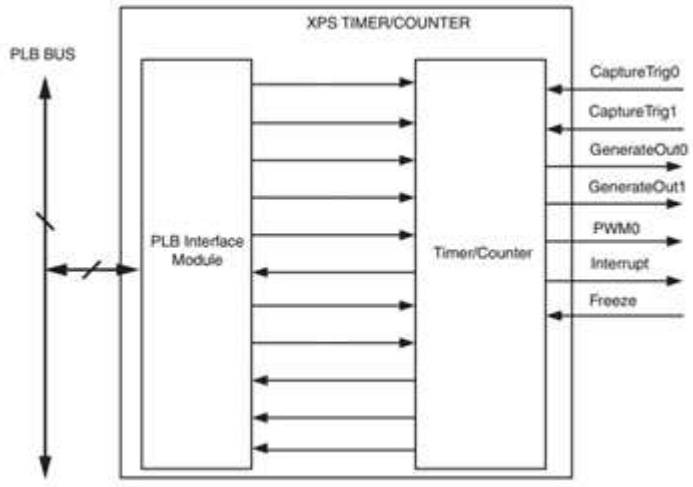


Figura 3.49. Diagrama de bloques XPS Timer/Counter.

La interfaz PLB proporciona la interacción entre el XPS Timer/Counter y el bus PLB. El módulo del temporizador contiene dos contadores idénticos, cada contador tiene asociado sus registros para almacenar los valores iniciales o los valores a capturar.

Existen tres modos de operación del XPS Timer/Counter que a continuación se describen:

Modo de generación: en este modo se carga un valor en el contador y cuando se desborda, se genera una señal de salida. Este modo es útil para generar eventos repetitivos con un intervalo de tiempo específico.

Modo de captura: en este modo el valor del contador se almacena en un registro cuando se detecta un cambio en una señal externa. Este modo es útil para detectar eventos externos.

Modulación por ancho de pulso PWM: en este modo se utilizan los dos contadores para producir la señal de salida PWM. Con un contador se específica el periodo y con otro contador se específica el tiempo en alto.

El XPS Timer/Counter cuenta con siete señales para interactuar con él, las cuales se muestran en la tabla 3.17. También se muestra la dirección de cada señal, ya sea de salida o de entrada.

Nombre	Entrada/Salida	Descripción
CaptureTrig0	Entrada	Señal 0 para el modo captura
CaptureTrig1	Entrada	Señal 1 para el modo captura
Freeze	Entrada	Señal para pausar el o los contadores
GenerateOut0	Salida	Señal 0 para el modo generador
GenerateOut1	Salida	Señal 1 para el modo generador
PWM0	Salida	Señal en el modo PWM
Interrupt	Salida	Señal que indica que ha ocurrido una interrupción

Tabla 3.17. Señales para el XPS Timer/Counter.

El XPS Timer/Counter contiene seis registros que se describen en la tabla 3.18. En ella se muestra el valor de la dirección, en hexadecimal, de cada registro, partiendo de una dirección base, (la dirección base se asigna con XPS) y se muestran los permisos con los que cuenta cada registro, ya sea escritura (E), lectura (L) ó ambos (E/L).

El XPS Timer/Counter se configuró con los dos contadores, de 32 bits, y con las señales de salida GenerateOut0, GenerateOut1 y PWM0. Se le asignó el nombre de XPS_TIMER_0. El XPS Timer/Counter se agregó con la intención de generar retardos de valores conocidos.

3.4.7. Controlador de memoria

Para almacenar datos en alguna memoria externa se utiliza el IP Multi Port Memory Controller (MPMC). Este IP es uno de los más elaborados y complejos, ya que cuenta con todos los

parámetros y las configuraciones para realizar la escritura y lectura en una memoria externa de una manera transparente para el usuario. Soporta principalmente memorias del tipo DDR, DDR2 y DDR3.

Nombre del registro	Descripción	Dirección	Permisos
TCSR0	Se habilitan los contadores, el modo PWM y las interrupciones. Se carga el contador, se configura la cuenta hacia arriba o la cuenta hacia abajo, se selecciona el modo de operación, se habilita la salida o la entrada de los modos de operación, entre otros.	BASE+0x00	E/L
TLR0	Registro de carga para el contador 0	BASE+0x04	E/L
TCR0	Contiene el contador 0	BASE+0x08	L
TCSR1	Se habilitan los contadores, el modo PWM y las interrupciones. Se carga el contador, se configura la cuenta hacia arriba o la cuenta hacia abajo, se selecciona el modo de operación, se habilita la salida o la entrada de los modos de operación, entre otros.	BASE+0x10	E/L
TLR1	Registro de carga para el contador 1	BASE+0x14	E/L
TCR1	Contiene el contador 1	BASE+0x18	L

Tabla 3.18. Registros del XPS Timer/Counter.

La memoria DDR utilizada en la tarjeta SPARTAN 3E es una MT46V32M16 de una interfaz de 16 bits de datos, 13 bits de direcciones, 14 bits de control y una alimentación de 2.5V, con un voltaje de referencia de 1.25V. El controlador que se utilizó para la memoria DDR se configuró con el asistente *Base System Builder,* el cual proporciona las características necesarias para dicha memoria. Al IP MPMC se le asigna el nombre de DDR_SDRAM.

No se profundizará en este IP ya que es muy complejo y muy larga la descripción. Éste es un claro ejemplo de las ventajas que proporcionan los IP en cuanto al funcionamiento sin conocer detalles profundos de su construcción.

3.4.8. Hardware final

Conociendo los IPs de interés, éstos se agregaron a la estructura básica de MicroBlaze, previamente construida (figura 3.42). En la figura 3.50 se muestra la pantalla de XPS, pestaña *BUS INTERFACES*, en donde se observan las conexiones de los IP utilizados con el bus PLB. También se observan los nombres asignados, el tipo de IP y la versión de cada uno de ellos.

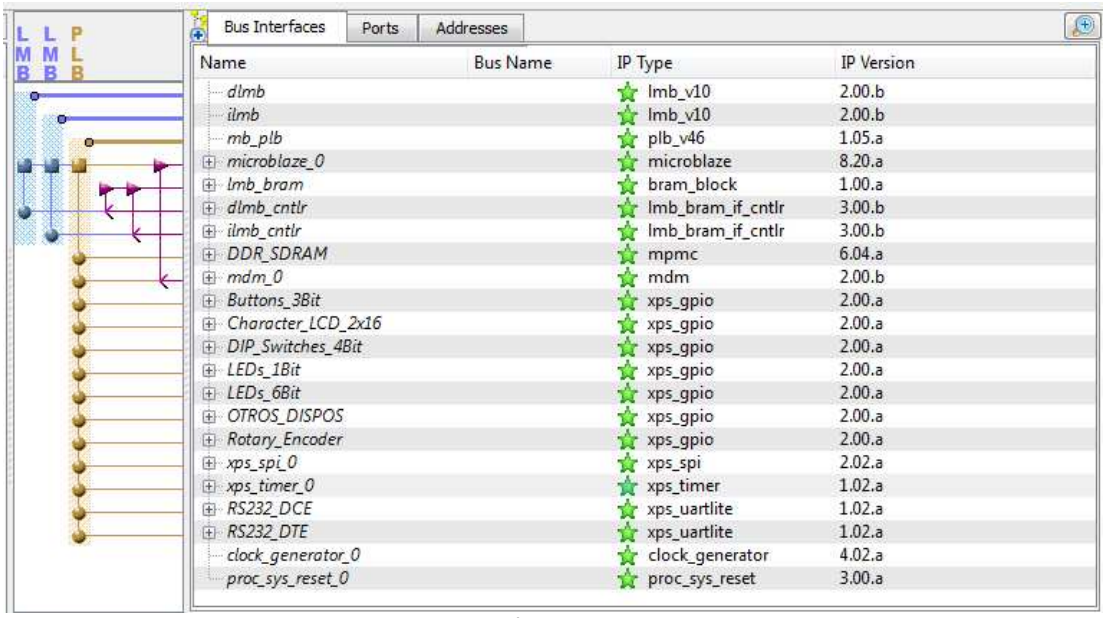


Figura 3.50. Conexión de los IP seleccionados.

En la pestaña *PORTS* es donde se seleccionan las señales de cada IP que serán conectadas al exterior del FPGA. En la figura 3.51 se muestra la pantalla *PORTS* y se observa parte de los puertos externos utilizados. En dicha pantalla se encuentra el nombre de la terminal externa (*EXTERNAL PORTS*), el nombre de la conexión a la que pertenece dicha terminal (*NET*), el tipo de terminal, ya sea entrada (I) o salida (O) y el tamaño en bits de cada terminal (*RANGE*). En el apéndice C se muestran a detalle el nombre y la terminal de cada señal conectada al exterior.

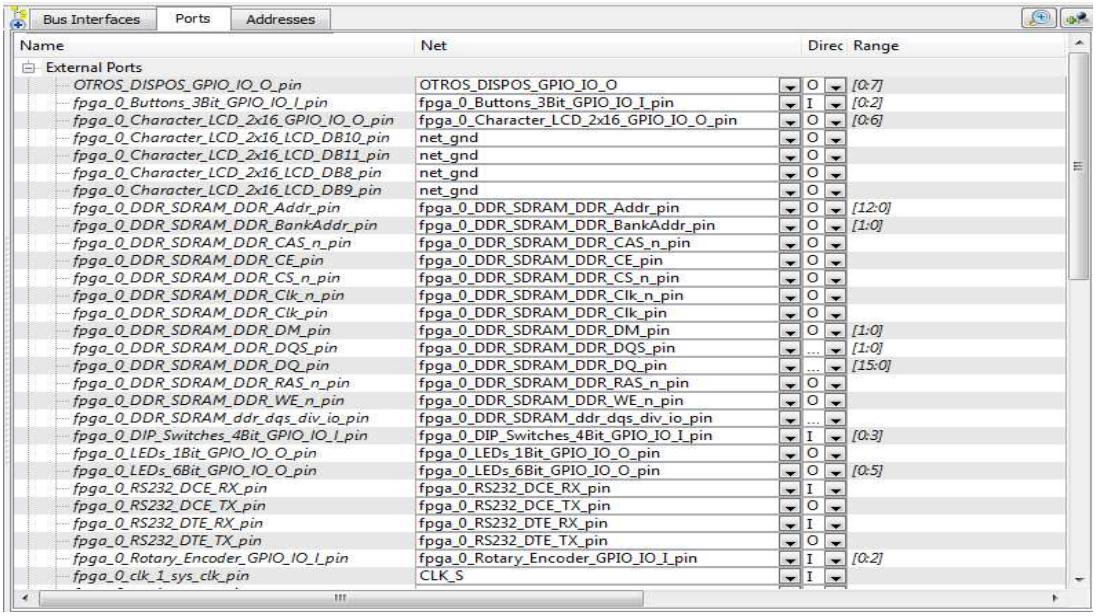
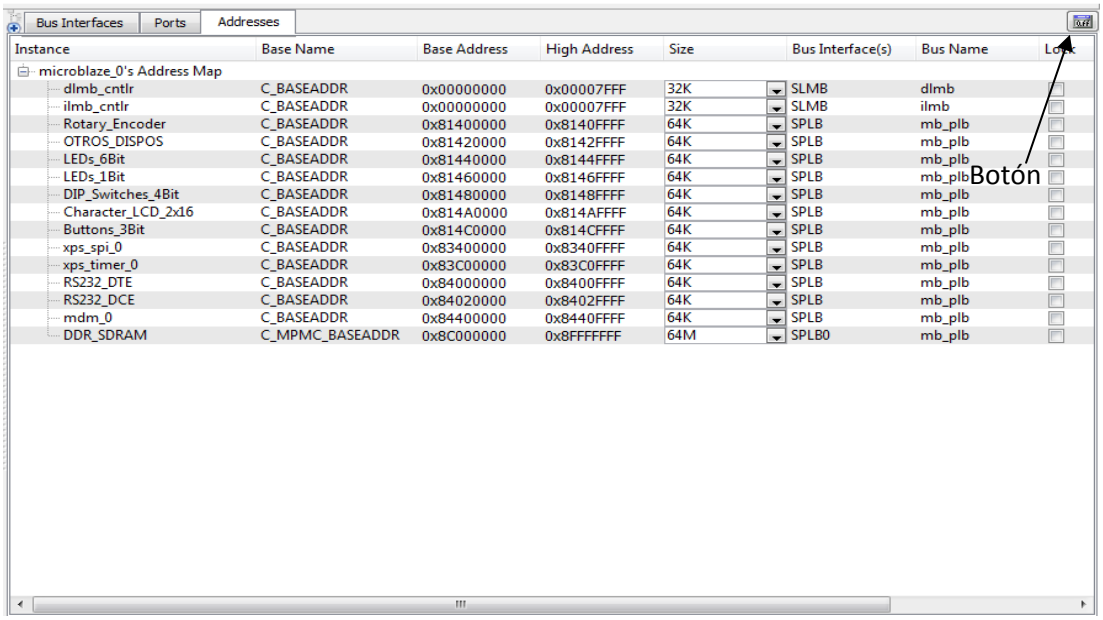


Figura 3.51. Terminales externas.

Por último, en la figura 3.52 se muestra la pestaña *ADDRESSES,* en donde se asigna la dirección base de cada IP con la cual se accede a él. En dicha figura se observan principalmente los IP utilizados, la dirección de inicio (*BASE ADDRESS*), la dirección final (*HIGH ADDRESS*) y el tamaño de dicha dirección. La dirección se puede asignar de manera manual ó se puede asignar de manera automática con el botón que se encuentra en la parte superior derecha de la figura. La asignación de memoria se realizó de manera automática con el botón antes mencionado.



Instance	Base Name	Base Address	High Address	Size	Bus Interface(s)	Bus Name	Lock
microblaze_0's Address Map							
dlmb_cntlr	C_BASEADDR	0x00000000	0x00007FFF	32K	SLMB	dlmb	
ilmb_cntlr	C_BASEADDR	0x00000000	0x00007FFF	32K	SLMB	ilmb	
Rotary_Encoder	C_BASEADDR	0x81400000	0x8140FFFF	64K	SPLB	mb_plb	
OTROS_DISPOS	C_BASEADDR	0x81420000	0x8142FFFF	64K	SPLB	mb_plb	
LEDs_6Bit	C_BASEADDR	0x81440000	0x8144FFFF	64K	SPLB	mb_plb	
LEDs_1Bit	C_BASEADDR	0x81460000	0x8146FFFF	64K	SPLB	mb_plb	
DIP_Switches_4Bit	C_BASEADDR	0x81480000	0x8148FFFF	64K	SPLB	mb_plb	
Character_LCD_2x16	C_BASEADDR	0x814A0000	0x814AFFFF	64K	SPLB	mb_plb	
Buttons_3Bit	C_BASEADDR	0x814C0000	0x814CFFFF	64K	SPLB	mb_plb	
xps_spi_0	C_BASEADDR	0x83400000	0x8340FFFF	64K	SPLB	mb_plb	
xps_timer_0	C_BASEADDR	0x83C00000	0x83C0FFFF	64K	SPLB	mb_plb	
RS232_DTE	C_BASEADDR	0x84000000	0x8400FFFF	64K	SPLB	mb_plb	
RS232_DCE	C_BASEADDR	0x84020000	0x8402FFFF	64K	SPLB	mb_plb	
mdm_0	C_BASEADDR	0x84400000	0x8440FFFF	64K	SPLB	mb_plb	
DDR_SDRAM	C_MPMC_BASEADDR	0x8C000000	0x8FFFFFFF	64M	SPLB0	mb_plb	

Figura 3.52. Direcciones de cada IP.

Una representación del hardware final a nivel de diagrama de bloques se muestra en la figura 3.53. Dentro del bloque FPGA se encuentra todo el sistema embebido con los IP seleccionados. Cada IP tiene su número de entradas y/o salidas así como el periférico al que corresponde dicho IP.

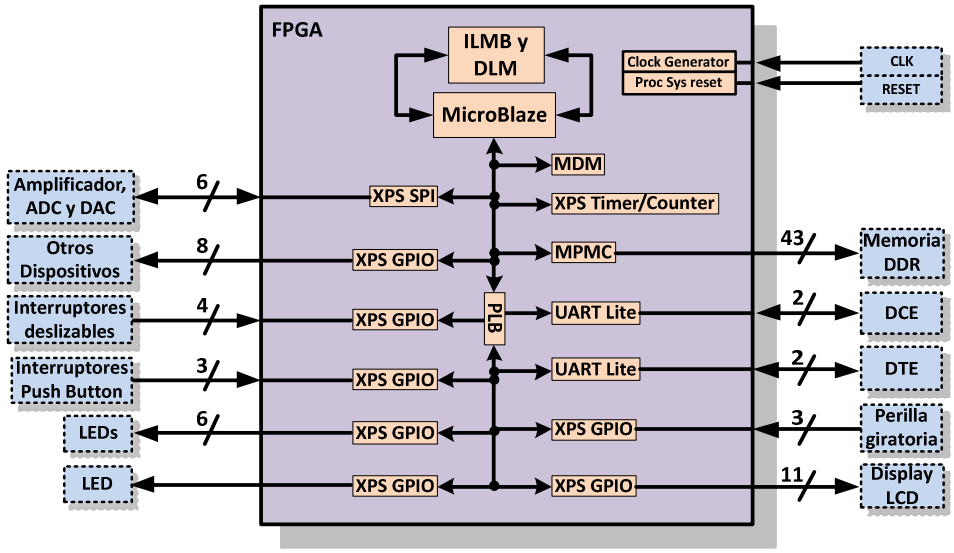


Figura 3.53. Diagrama de bloques hardware final.

Teniendo el hardware final en XPS, se crea el *bitstream* y las especificaciones de memoria en ISE Design Suite. Una vez creados se exportan a SDK, donde se crea el software de la aplicación.

3.4.9. Creación del software

Cuando se inicia SDK, lo primero que se tiene que crear es un espacio de trabajo (*workspace)*, figura 3.54. El espacio de trabajo es una carpeta en donde SDK almacena los archivos del ó de los proyectos a realizar. Es importante que la ruta de ese directorio no contenga espacios.

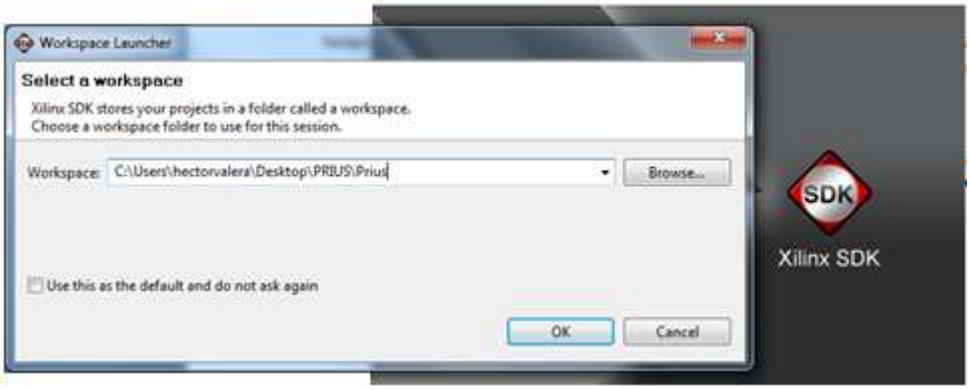


Figura 3.54. Selección del espacio de trabajo.

Una vez seleccionado el espacio de trabajo aparecerá la pantalla de inicio de SDK, figura 3.55.

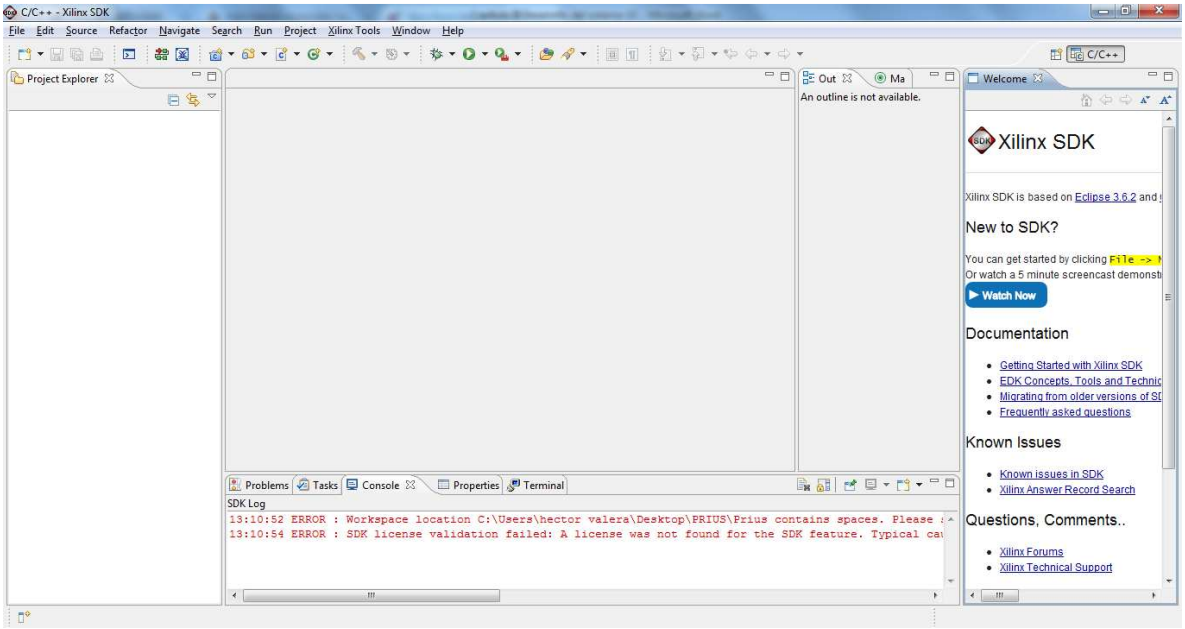


Figura 3.55. Pantalla de inicio de SDK.

En esta pantalla se elije la opción File → New → Xilinx C Project para crear un proyecto. Debido a que es el primer proyecto que se crea, se tienen que importar las especificaciones del hardware

que se va a utilizar y que se crearon anteriormente con XPS e ISE Design Suite, es decir, el *bitstream* y las especificaciones de memoria (BMM: *Block Memory Map*), figura 3.56.

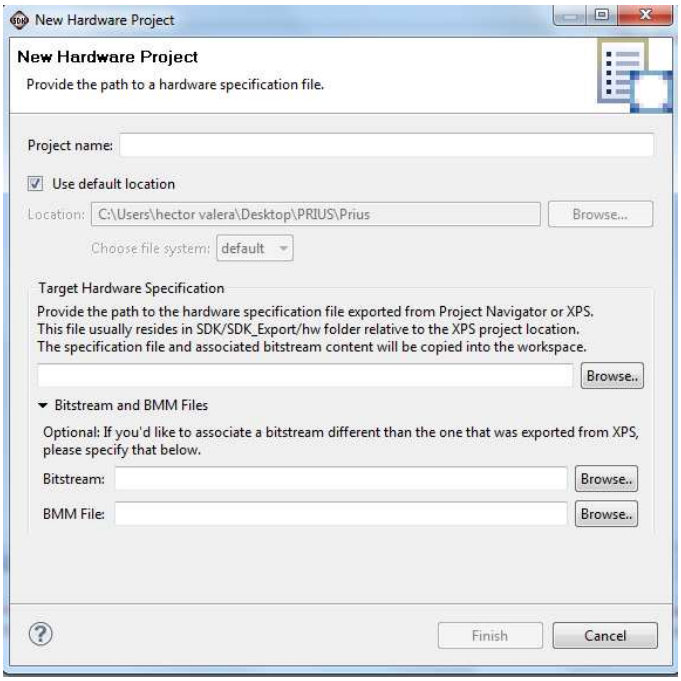


Figura 3.56. Especificación del hardware a utilizar.

Ya especificado el hardware, se selecciona el nombre del proyecto y el tipo de aplicación que se requiera, ya sea una aplicación en blanco ó ejemplos que proporciona SDK, figura 3.57.

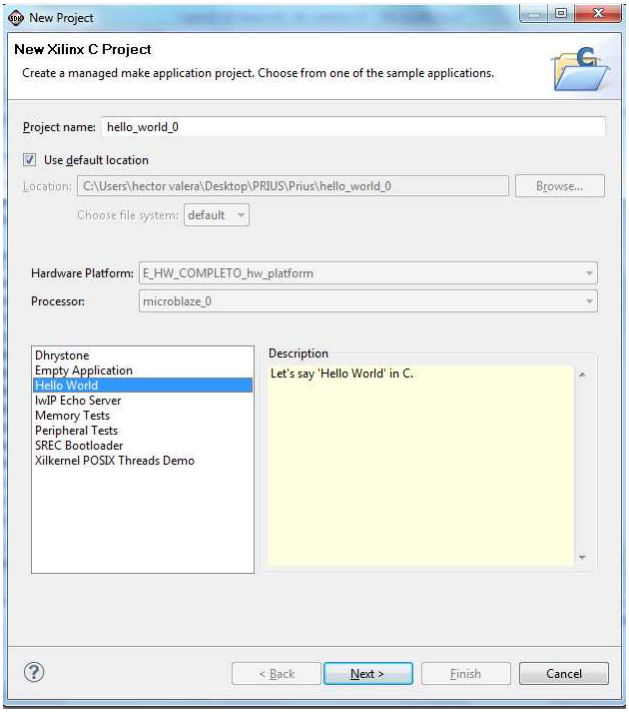


Figura 3.57. Nombre y tipo de proyecto.

Por último, se crea el *Board Support Package* ó BSP, que es una colección de bibliotecas y controladores que forman la capa más baja de la aplicación software, figura 3.58.

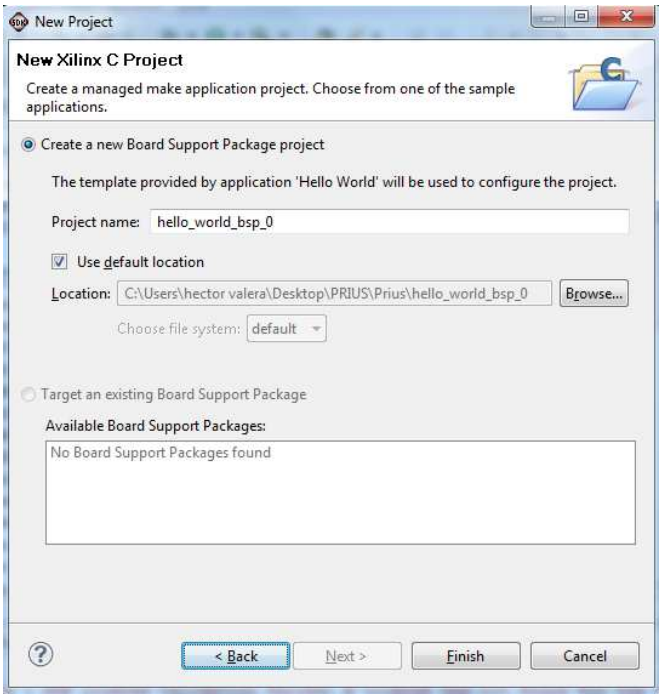


Figura 3.58. Creación del BSP.

En la figura 3.59 se muestra la pantalla de SDK para el desarrollo del software. Tiene principalmente tres secciones: PROJECT EXPLORER, donde se muestran los proyectos que se han creado; la sección PRIUS.c, que es el archivo fuente en donde se realiza el software y la sección OUTLINE, donde se muestran todas las variables, archivos de cabecera, bibliotecas, entre otras cosas, que se utilizan en la creación del software. En la parte inferior se encuentran unas pestañas en donde las más importantes son tres: CONSOLE, PROBLEMS y DEBUG. CONSOLE es donde se muestra información del tamaño del software, si existe algún error en el código y los procesos que se están ejecutando o se ejecutaron. La pestaña PROBLEMS es donde se observan los errores que se tienen en el código, así como advertencias (*warnings)* que pudiera tener el código desarrollado. La pestaña DEBUG indica si algún proyecto está ejecutándose en el FPGA. La pestaña TASKS muestra los marcadores que se tienen, por ejemplo, un punto de paro en el software. La pestaña PROPERTIES muestra las características del archivo fuente como el tamaño, la localización y el nombre. La pestaña XMD CONSOLE se utiliza para línea de comandos.

Una vez creado el proyecto para el desarrollo del software, se empezó a configurar y controlar cada IP. Cabe señalar que SDK proporciona la información necesaria de cada IP para hacer uso de ellos. Dentro de esta información se encuentra una lista de las funciones para escribir y leer los registros, la descripción de dichas funciones y ejemplos de cómo utilizarlas.

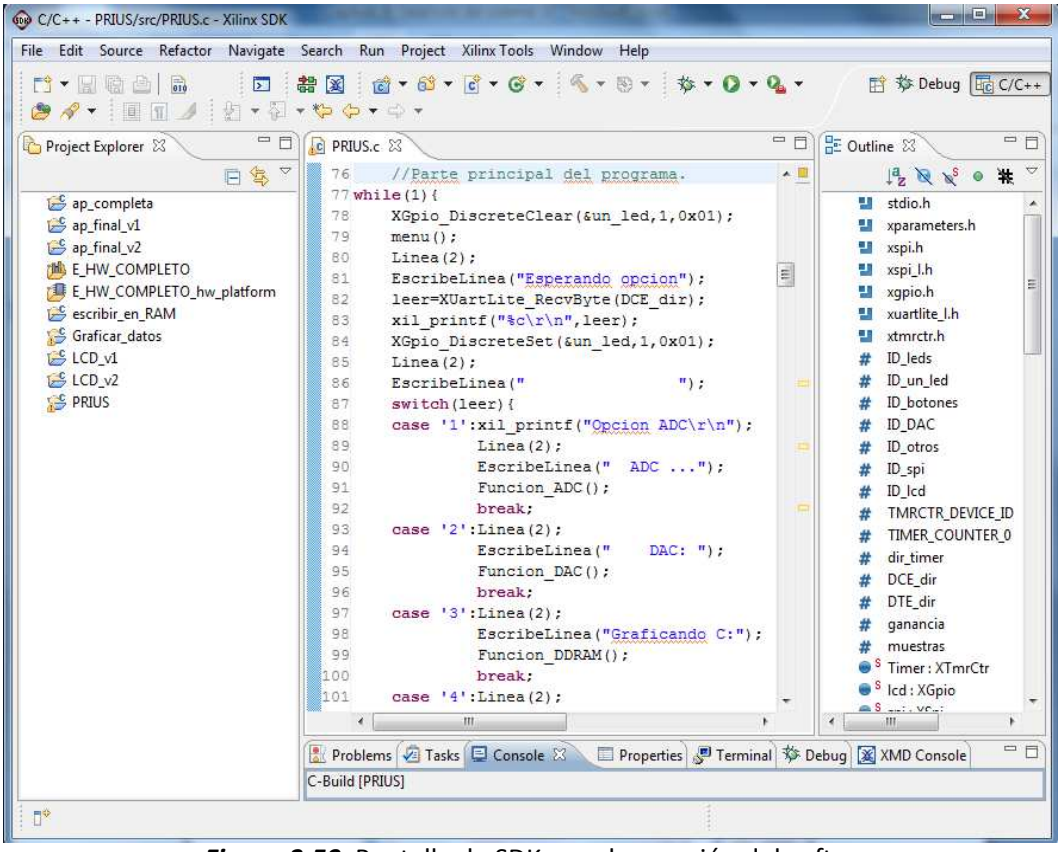


Figura 3.59. Pantalla de SDK para la creación del software.

3.5. Integración del sistema de monitoreo y control

Para realizar la integración de los IP y que interactúen entre ellos, se realiza una aplicación software. En este mismo software se tienen que configurar los registros de cada uno de los IP para el funcionamiento requerido, es decir, se realiza una configuración software. Para acceder al dispositivo deseado, se crea una variable (instancia) del tipo de IP que controla dicho dispositivo, por ejemplo, si se accede a los LEDs, interruptores o al display LCD, se crea una variable del tipo XGpio, ya que el IP que maneja estos componentes es el XPS GPIO y, por ejemplo, si se accede al SPI se crea una variable del tipo XSpi. A continuación se describe la configuración software que se utilizó en cada IP para controlar cada periférico:

Interruptores deslizables y push button

La configuración del XPS GPIO para estos dispositivos consiste únicamente en indicar que se utilizarán como entradas. Esto se realiza escribiendo el valor 0x0F, hexadecimal, en el registro GPIO_TRI, para los interruptores deslizables, y 0x03, también en hexadecimal, para los *push buttons*. Para leer el estado de los interruptores se lee el registro GPIO_DATA y se asigna a una variable. La configuración para determinar si es entrada o salida se realiza mediante una instrucción llamada *XGpio_SetDataDirection*. Para realizar una lectura del XPS GPIO se utiliza la

instrucción *XGpio_DiscreteRead* y para realizar una escritura se utiliza la instrucción *XGpio_DiscreteSet*.

LEDs

La configuración del XPS GPIO para este dispositivo consiste en indicar que se utilizará como salida. Esto se realiza escribiendo en hexadecimal 0x00 en el registro GPIO_TRI. Para activar los LEDs, se escribe en el registro GPIO_DATA el dato que se requiera mostrar.

Perilla giratoria

La configuración del XPS GPIO para este dispositivo consta de indicar que se utilizará como entrada. Esto se realiza escribiendo 0x0F, en hexadecimal, en el registro GPIO_TRI. Para leer el estado de la perilla giratoria se lee el registro GPIO_DATA y se le asigna a una variable.

Display LCD

La configuración del XPS GPIO para este dispositivo consta en indicar que se utilizará como salida. Esto se realiza escribiendo en hexadecimal 0x00 en el registro GPIO_TRI. Para poder desplegar caracteres en el LCD se realizaron las funciones necesarias para poder iniciarlo y configurarlo. Estas funciones se realizaron mediante la escritura de los datos a enviar en el registro GPIO_DATA.

Otros dispositivos

Para otros dispositivos, el XPS GPIO se configura como salidas, escribiendo un 0x00 en el registro GPIO_TRI. Cabe recordar que en este XPS GPIO se encuentra, principalmente, la señal para iniciar la conversión analógica digital, AD_CONV, la señal de habilitación del amplificador, AMP_SHDN y la señal de reinicio por hardware del DAC, DAC_CLR. También se encuentran las señales FPGA_INIT_B, SF_CE0 y SPI_SS_B, las cuales desactivan los componentes conectados a la comunicación SPI. El valor que se escribe en el registro GPIO_DATA es 0x9C. En la tabla 3.19 se muestra el orden de los bits y el valor de salida de cada uno. El bit más significativo es DAC_CLR y el menos significativo es SPI_SSB.

DAC_CLR	AMP_SHDN	AD_CONV	FPGA_INIT_B	SF_CE0	SPI_SS_B	-	-
1	0	0	1	1	1	0	0

Tabla 3.19. Valores de salida para otros dispositivos.

Es importante observar que para activar la conversión analógica digital se tiene que modificar el valor del bit AD_CONV, al igual que si se desea reiniciar o deshabilitar el DAC o el amplificador, se tienen que modificar los bits DAC_CLR y AMP_SHDN respectivamente.

Amplificador y convertidores

Para controlar al amplificador y los convertidores ADC y DAC se utilizó únicamente un XPS SPI en modo maestro para los tres dispositivos. La polaridad del reloj CPOL (el estado inicial de la señal

SCK) siempre se colocó en nivel lógico bajo, mientras que la fase CPHA (el flanco de operación) se colocó en flanco de subida para el amplificador y el DAC, mientras que para el ADC se colocó en flanco de bajada. Para configurar el XPS SPI y controlar de manera correcta el amplificador y el DAC se escribe un 0x106, en hexadecimal, en el registro SPICR y para manejar el ADC se escribe un 0x116 en el mismo registro. Debido a la comunicación SPI modificada con la que cuenta el ADC, el XPS SPI no es suficiente para controlar el proceso de conversión. Para realizar una conversión, se utiliza primero el XPS GPIO denominado "otros dispositivos" para obtener un pulso en la señal AD_CONV y después se activa el SPI para generar la señal de reloj y obtener los datos de conversión. Para poder transmitir información y poder configurar el IP según el dispositivo, se realizaron funciones que controlan dichas características. La instrucción que se utiliza para configurar el XPS SPI es *XSpi_SetControlReg*. Algunas opciones para configurar el XPS SPI son *XPS_CR_CLK_PHASE*, que controla el flanco de operación, y *XPS_CR_MASTER_MODE_MASK* que lo configura en modo maestro.

DCE y DTE

Para el caso del XPS UART Lite no hay configuración de software, sólo se escriben y leen los datos que se envían y se transmiten. La instrucción para recibir un dato de 8 bits es XUartlite_RecvByte y para enviar un dato de 8 bits es XUartlite_SendByte.

Temporizador/Contador

El XPS Timer/Counter se utiliza para retardos, por lo que se configuró en modo de generación. Los retardos se realizan cargando el valor del retardo necesario en el contador 0 y se espera a que el contador se desborde, por lo que se configura para que realice la cuenta hacia abajo. Para configurar el contador con estas características se escribe un 0x22, hexadecimal, en el registro TCSR0. Para configura el XPS Timer/Counter se utiliza la instrucción XTmrCtr_SetOptions. Algunas opciones para configurar son XTC_DOWN_COUNT_OPTION que configura la cuenta hacia abajo y TIMER _COUNTER_0 que selecciona el contador 0 del XPS Timer/Counter.

Memoria DDR

Para el MPMC no se requiere configuración de software y solamente se escriben y leen los datos de la memoria DDR.

Es importante mencionar que el lenguaje que se utilizó para realizar la aplicación software es lenguaje C, por lo que la estructura, el manejo de variables, el manejo de funciones y las sentencias como IF, ELSE, SWITCH son las estándares del lenguaje C. En el apéndice *C* se muestra el código fuente desarrollado, donde se observan las configuraciones de software de cada IP. También se observa la estructura general del software en el lenguaje C. La estructura general se compone de las siguientes secciones: declaración de bibliotecas, declaración de variables globales, declaración de funciones, función principal *main()* y definición de las funciones declaradas.

Ya que se configuró cada IP, se procedió a diseñar lo que es propiamente el sistema de monitoreo y control.

3.5.1. Sistema de monitoreo y control

El software para el sistema de monitoreo y control está basado principalmente en la comunicación UART con la cual se envían y reciben datos. El software consiste en un menú con cinco opciones, figura 3.60.

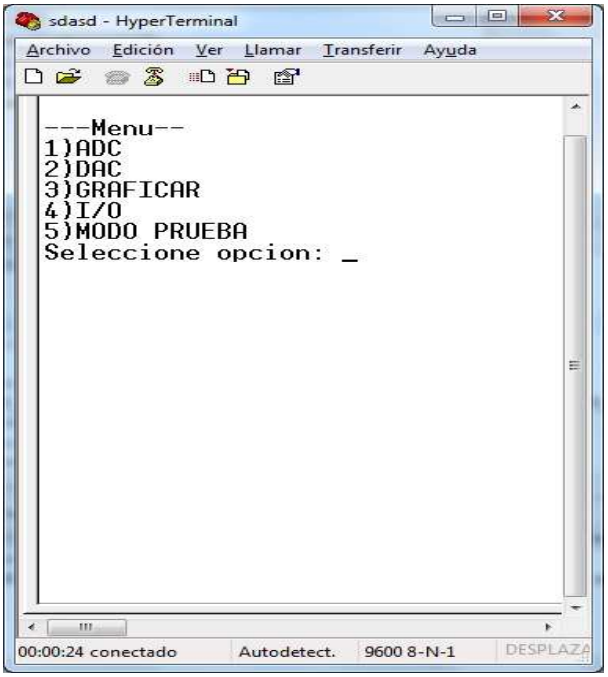


Figura 3.60. Menú principal.

A continuación se describe la función de cada opción:

Opción 1) ADC: en esta opción se adquieren señales analógicas por medio del ADC. Los datos adquiridos se almacenan en la memoria DDR.

Opción 2) DAC: en esta opción se envía al DAC el voltaje ingresado mediante la comunicación UART, también se selecciona el DAC con el que se quiere comunicar.

Opción 3) GRAFICAR: esta opción es complementaria a la opción uno, ya que se envían los datos almacenados en la memoria DDR de la conversión analógica digital por la comunicación UART. Se selecciona qué canal se quiere enviar, ya sea el canal 1 ó el canal 2.

Opción 4) I/O: con esta opción se activan seis LEDs y se leen cuatro interruptores deslizables, simulando la activación y simulando la lectura del estado de sistemas externos. Para el caso de los LEDs se ingresa el valor en base decimal entre 0 y 63 y se muestra en base binario. Para el caso de los interruptores se lee el estado de los interruptores y se despliega en base decimal.

Opción 5) MODO PRUEBA: con esta opción se comprueba que la comunicación SPI es correcta, ya que se activa la opción uno y la opción dos, es decir, se adquieren datos con el ADC, se almacenan en la memoria DDR, se leen los datos de la memoria DDR y se envían por los DACs.

Junto a estas cinco opciones se encuentra el display LCD y un LED como indicadores. El display LCD va mostrando los procesos y las acciones que se están ejecutando en el FPGA. Por ejemplo, si se selecciona la opción 1, en el display LCD se muestra el mensaje "ADC …" ó en la opción 4 se muestra el mensaje "I/O …". El LED indica cuándo el sistema está ocupando, mediante un nivel lógico alto, y cuándo está desocupado, mediante un nivel lógico bajo.

Con el software creado para la segunda aproximación se da por concluido el diseño del sistema de monitoreo y control. Cabe recordar que la gran ventaja de este sistema de monitoreo y control es la capacidad de reconfigurarse, ya sea aumentando o disminuyendo sus capacidades tanto en software como en hardware.

En este capítulo se ha presentado el método utilizado para el diseño del sistema de monitoreo y control. En el siguiente capítulo se describirá el diseño y la construcción del circuito impreso para el sistema de monitoreo y control.

Capítulo IV
Implementación de prototipos de hardware

En este capítulo se presenta el diseño y la implementación de los circuitos impresos de cada componente que conforma el sistema de monitoreo y control, así como las consideraciones generales y particulares que se tomaron en cuenta para el diseño de cada uno de ellos. También se abordan las dificultades que se presentaron en el desarrollo de algunos de ellos.

4.1. Introducción

Parte del sistema de monitoreo y control es diseñar y construir una tarjeta de circuito impreso (PCB: *Printed Circuit Board*) con los componentes del mismo, principalmente por dos razones: la primera, para obtener una plataforma de desarrollo FPGA propia; y la segunda, para desarrollar circuitos impresos de más de dos capas de cobre.

El incremento de capacidades en los dispositivos FPGA actuales y el tamaño reducido de su encapsulado hace que los circuitos impresos sean de mayor complejidad tanto en el diseño como en la construcción. Con encapsulados de entre 256 y 1156 terminales, con separación entre ellas del orden de 1mm, provoca que se diseñen circuitos impresos de, mínimo, cuatro capas de cobre.

Junto a las características del encapsulado del FPGA se encuentran las características de los componentes que incluirá el circuito impreso, por ejemplo, capacitores, resistencias, inductores, reguladores, interruptores, circuitos integrados, entre otros. El tamaño de estos componentes define en gran medida el tamaño del circuito impreso. Un aspecto importante en la selección del tamaño de estos componentes es la manera de ensamblar el circuito impreso, ya que si no se tiene el equipo adecuado es casi imposible ensamblarlos.

En el diseño de los circuitos impresos presentados se consideraron varios aspectos, entre los que destacan: el encapsulado del FPGA, el tamaño de los componentes y el ensamblado de los mismos. Con respecto al FPGA se seleccionó un encapsulado con relativamente pocas terminales, 320 terminales. En cuanto al tamaño de componentes, mayoritariamente capacitores y resistencias, se seleccionó un tamaño de entre los más pequeños, alrededor de 1mmx1mm, igualmente para los circuitos integrados se seleccionaron encapsulados de los más pequeños. En cuanto al ensamblado, se pretende soldar a mano y con el equipo disponible en el instituto.

4.2. Consideraciones generales para el diseño de circuitos impresos

Las consideraciones que se hicieron en todos los diseños de los circuitos impresos se describen a continuación:

1. Evitar pistas con ángulos de 90°, figura 4.1.

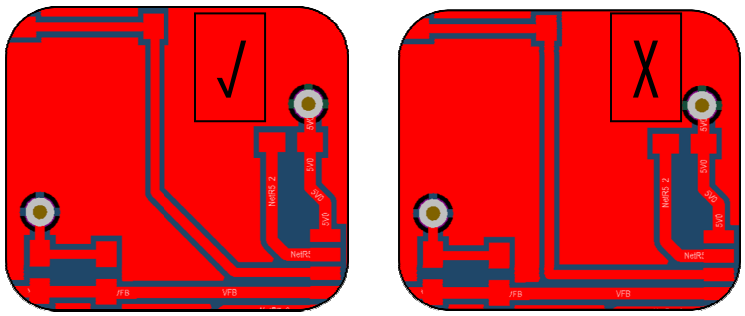


Figura 4.1. Evitar pistas de 90°.

2. Separación uniforme entre pistas paralelas, figura 4.2.

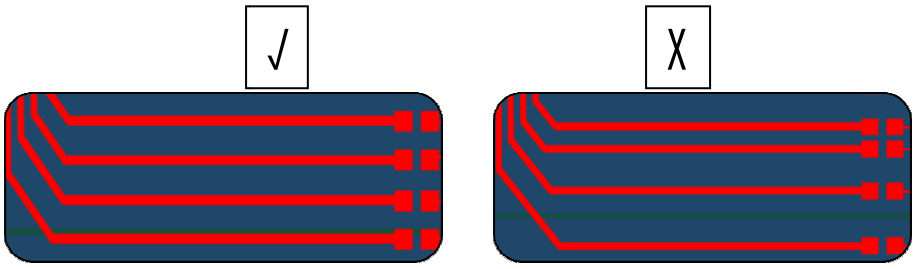


Figura 4.2. Separación uniforme entre pistas.

3. Unión radial entre *pad* y pista, figura 4.3.

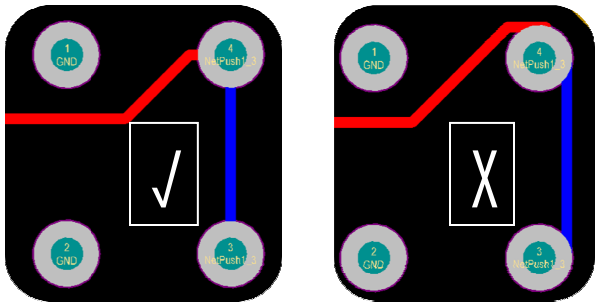


Figura 4.3. Unión radial entre *pad* y pista.

4. Pistas lo más cortas posibles, figura 4.4.

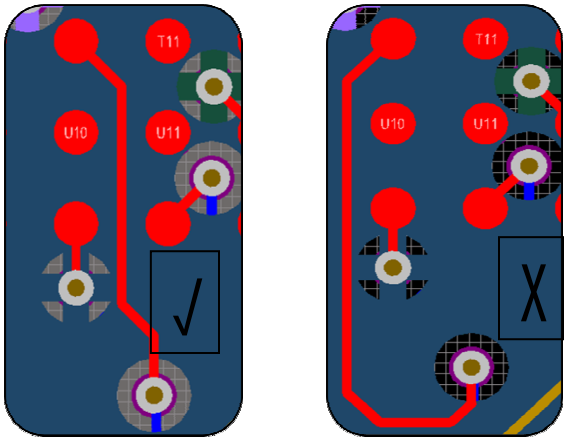


Figura 4.4. Pistas lo más cortas posible.

5. Separación mínima de 0.2mm entre pistas, componentes y planos de cobre, figura 4.5.

Figura 4.5. Separación entre pistas.

6. Separación del borde de la placa a las pistas de mínimo 2mm, figura 4.6.

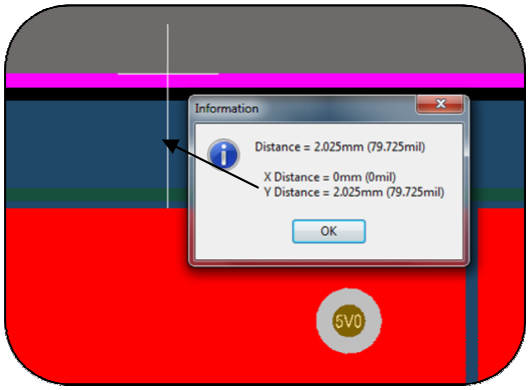


Figura 4.6. Separación entre borde y pista.

7. Planos de tierra y de alimentación sólidos y continuos (no letras, no figuras), figura 4.7.

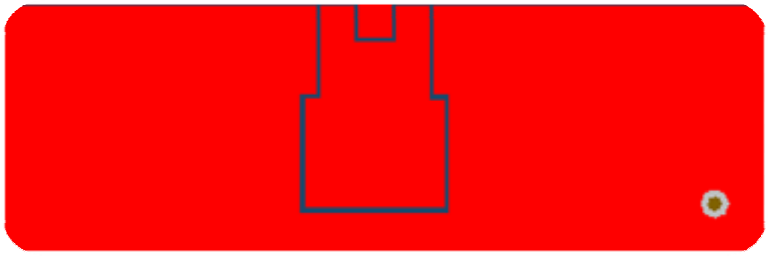


Figura 4.7. Planos de tierra y alimentación sólidos y continuos.

8. Separación de la parte analógica de la parte digital.

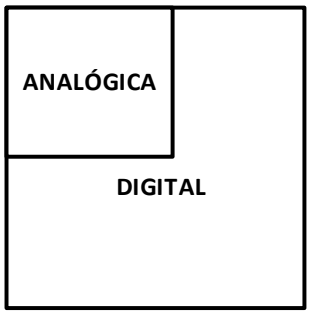


Figura 4.8. Parte analógica y parte digital separadas.

9. Colocación de capacitores de desacoplo lo más cercano posible al circuito integrado.

10. Trazado de las líneas perpendiculares entre capas, figura 4.9, es decir, en una capa de forma horizontal y en la otra capa de forma vertical.

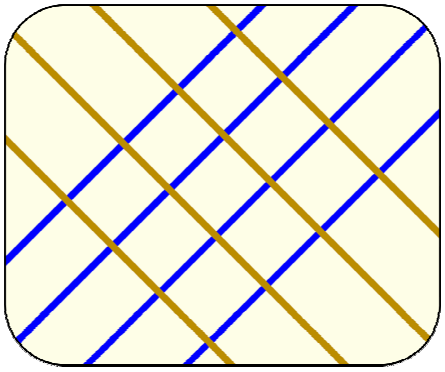


Figura 4.9. Tirado de las líneas perpendicular.

4.3. Circuito impreso de los convertidores

En la figura 4.10 se muestra el diagrama esquemático del amplificador, del ADC y del DAC. La mayoría de las bibliotecas para los circuitos integrados fueron desarrolladas manualmente, ya que

el software de diseño de PCBs no contaba especificamente con el componente a utilizar. Como se observa, el amplificador cuenta con 16 terminales al igual que el DAC, mientras que el ADC cuenta con 10 terminales y una terminal extra, la cual se encuentra debajo del circuito integrado y debe conectarse a tierra.

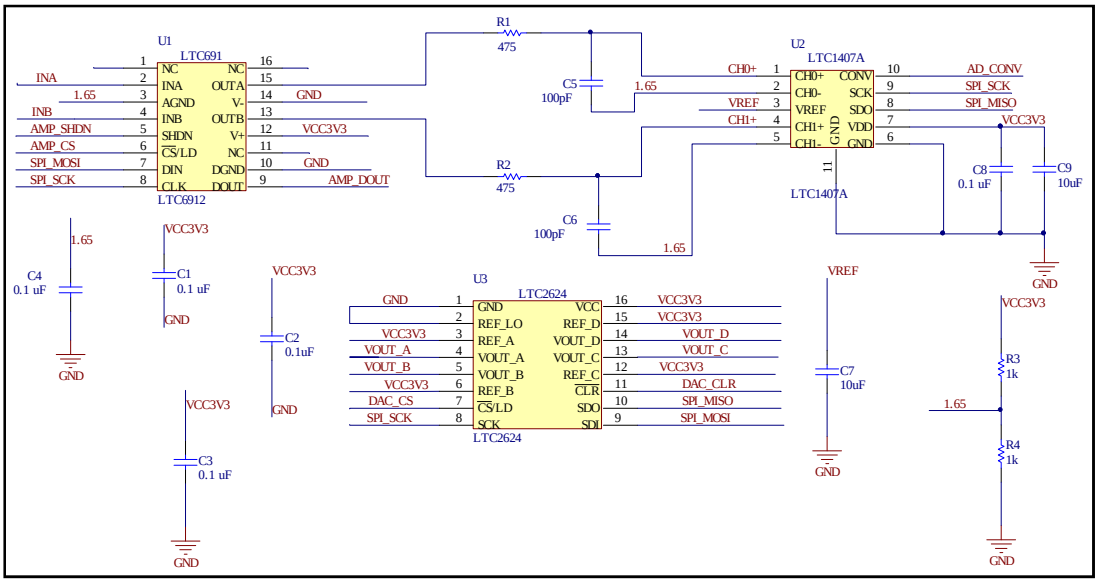
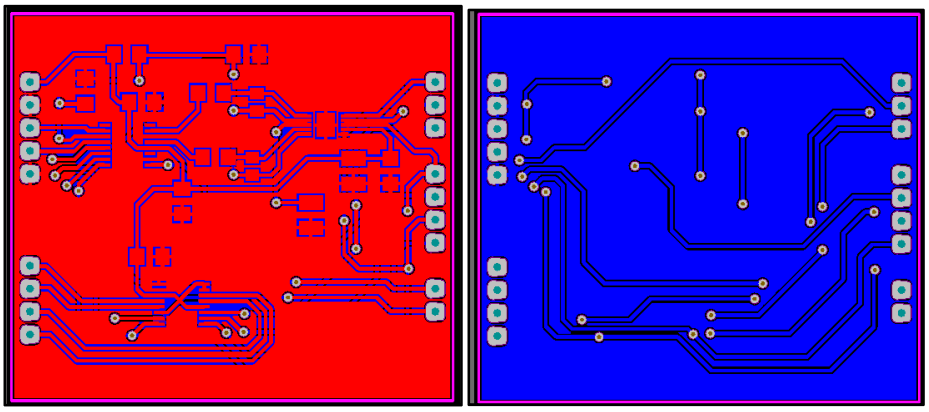


Figura 4.10. Diagrama esquemático del amplificador, ADC y DAC.

Se decidó colocar los tres componentes en un solo diseño esquemático, ya que los tres cuentan con comunicación SPI y algunas líneas son compartidas entre ellos.

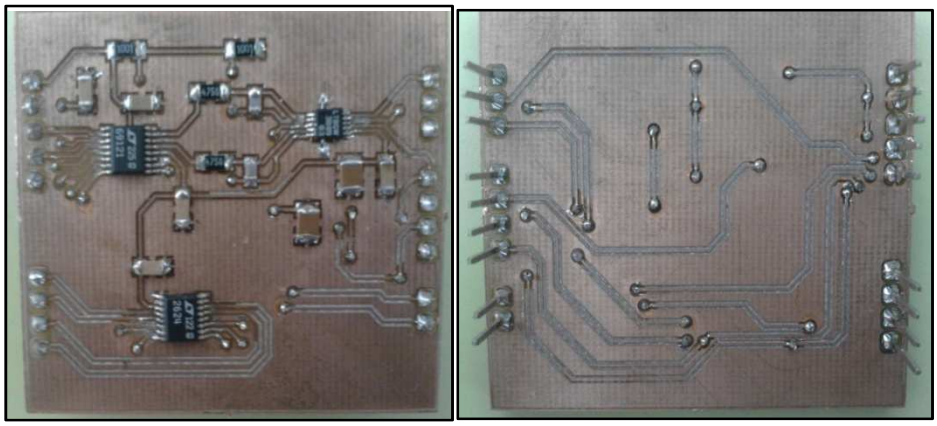
En la figura 4.11 se muestra el diseño de la PCB para el amplificador, el ADC y el DAC. En el inciso *a* se muestra la capa superior (TOP) y en el inciso *b* se muestra la capa inferior (BOTTOM).



a) Superior b) Inferior

Figura 4.11. PCB para el amplificador, el ADC y el DAC.

En la figura 4.12 se muestra la implementación de la PCB, en el inciso *a* la capa superior (TOP) y en el inciso *b* la capa inferior (BOTTOM). La PCB se construyó con el propósito de corroborar y validar el diseño realizado, para que posteriormente se incluya en la PCB final, sabiendo de antemano que el funcionamiento es el correcto. A la PCB final se le nombró ***PRIUS***.



a) Superior b) Inferior

Figura 4.12. Implementación de la PCB.

La PCB de los convertidores funcionó adecuadamente y se validó el diseño realizado, por lo que se utilizó en la tarjeta PRIUS.

4.4. Circuito impreso para la comunicación serie

El diagrama esquemático para la comunicación asíncrona se muestra en la figura 4.13, se observan, principalmente, los conectores DB9 para realizar la conexión con el puerto RS-232.

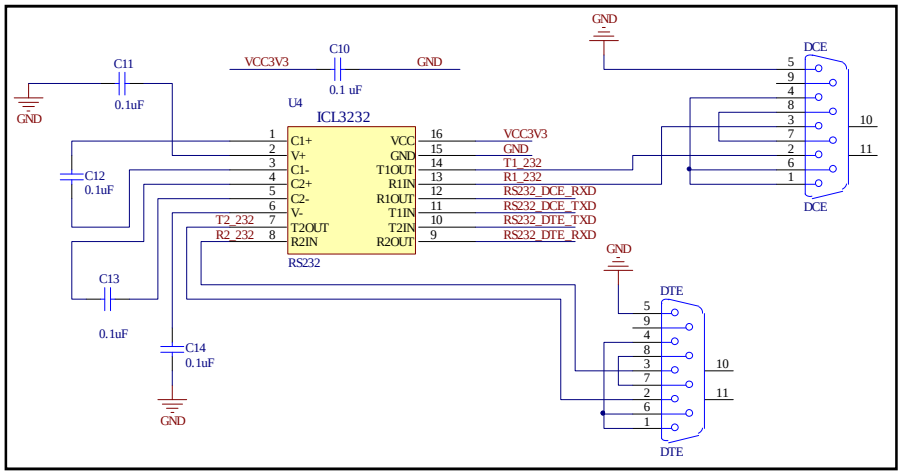
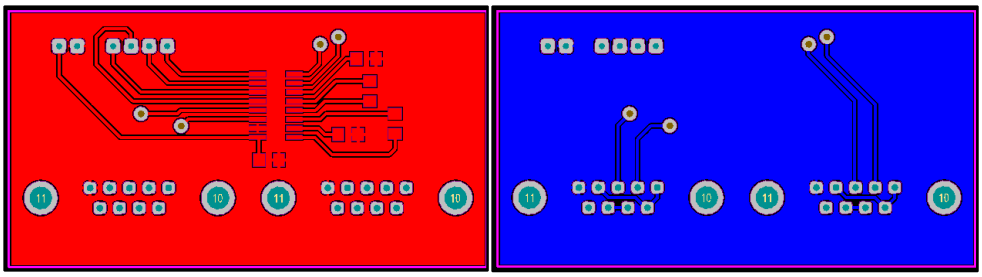


Figura 4.13. Diagrama esquemático comunicación asíncrona.

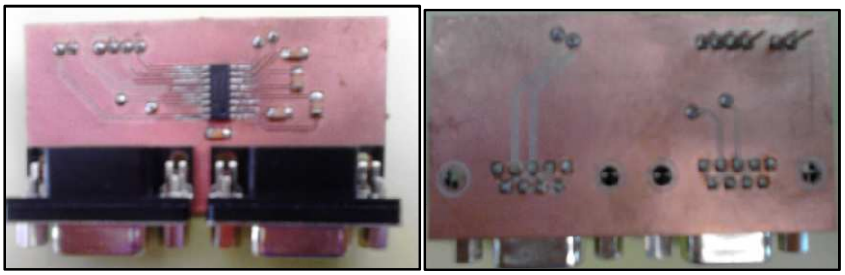
En la figura 4.14, inciso *a*, se muestra la capa superior (TOP) de la PCB diseñada para la comunicación asíncrona, mientras que en el inciso *b* se muestra la capa inferior (BOTTOM) de la PCB.



a) Superior b) Inferior

Figura 4.14. PCB para comunicación asíncrona.

En la figura 4.15 se muestra la implementación de la PCB, en el inciso *a* la capa superior (TOP) y en el inciso *b* la capa inferior (BOTTOM). La PCB se construyó con el propósito de corroborar y validar el diseño realizado, para que posteriormente se incluya en la tarjeta PRIUS, sabiendo de antemano que el funcionamiento es el correcto.



a) Superior b) Inferior

Figura 4.15. Implementación de la PCB.

La PCB de la comunicación asíncrona funcionó adecuadamente y se validó el diseño realizado, por lo que se utilizó en la tarjeta PRIUS.

4.5. Display LCD, LEDs, interruptores, DDR RAM y fuentes de alimentación

Para estos dispositivos no se construyó un circuito impreso debido principalmente a dos razones: la primera, para el caso de los interruptores, los LEDs y el display LCD, no fue necesaria, y la segunda, para el caso de la memoria y las fuentes de alimentación, los componentes no estaban disponibles para realizar pruebas porque se compraron en el extranjero y tardarían tres meses en llegar. Lo que se desarrolló fue el diagrama esquemático y los *footprints* para utilizarlos en el diseño de la tarjeta PRIUS. A continuación se describen los diagramas esquemáticos y los *footprints* que se realizaron para estos componentes.

4.5.1. Display LCD, LEDs e interruptores

Se desarrollaron los *footprints* del LCD y de los interruptores *push button* y deslizables. Para los LEDs se utilizaron empaquetados disponibles en el software de diseño. En la figura 4.16 se muestra

los *footprints* elaborados manualmente y que se utilizaron en la tarjeta PRIUS para estos dispositivos.

Figura 4.16. *Footprint* para el display LCD e interruptores.

Para poder realizar pruebas a estos dispositivos solamente se necesitó una placa de pruebas ó *protoboard*. En la *protoboard* se colocó fácilmente el display LCD, los interruptores *push button* y los LEDs sin mayor problema. En la figura 4.17 se muestran las conexiones del display LCD, los LEDs y los interruptores en la *protoboard*.

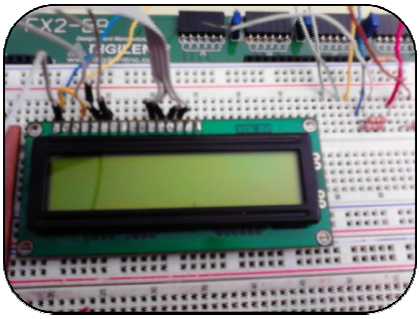


Figura 4.17. Conexión con la *protoboard*.

4.5.2. Memoria DDR RAM

El diagrama esquemático para la memoria DDR se muestra en la figura 4.18. Cabe mencionar que el encapsulado de este circuito integrado cuenta con 66 terminales y se realizó de manera manual con la ayuda de los asistentes del software de diseño de PCBs.

4.5.3. Fuentes de alimentación

Los voltajes necesarios para alimentar todos los dispositivos del sistema de monitoreo y control son: 1.2V, 1.25V, 2.5V y 3.3V. Estos voltajes se obtienen de dos circuitos integrados, los cuales son: el TPS75003 y el LTC3412. El TPS75003 proporciona los voltajes de 1.2V, 2.5V y 3.3V mientras que el LTC3412 proporciona un voltaje de 2.5V y de éste, se obtiene 1.25V. A continuación se describe brevemente cada regulador.

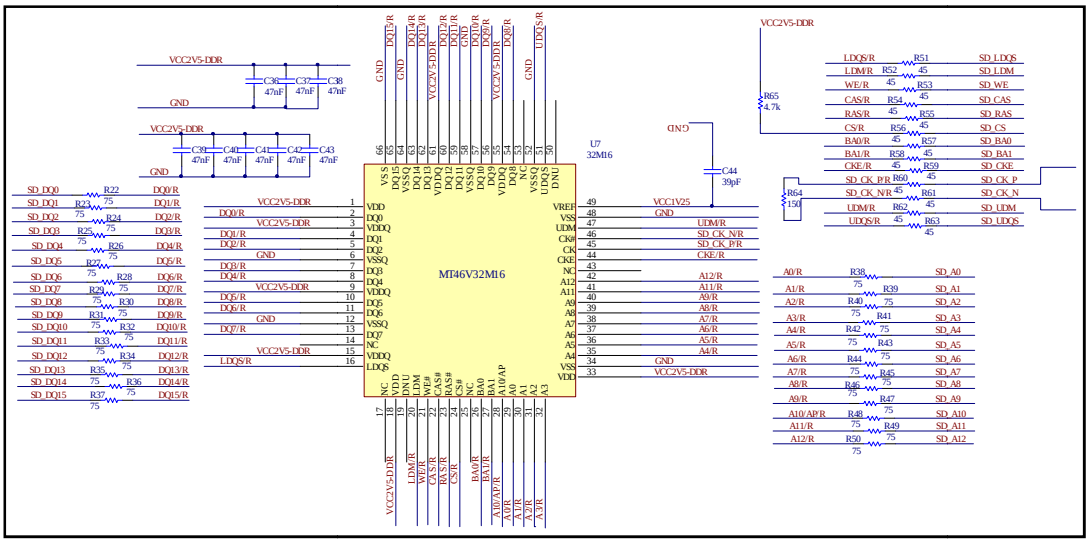


Figura 4.18. Diagrama esquemático memoria DDR.

Fuente de alimentación basada en el circuito integrado TPS75003

El circuito integrado (CI) TPS75003 está validado para alimentar los FPGA de Xilinx, especialmente de la familia SPARTAN 3E. El rango de voltaje de entrada que acepta es de 2.2V a 6.5V. Contiene un total de tres reguladores, dos fuentes conmutadas reductoras (*buck*) de 3A y una fuente lineal LDO (LDO: *Low Drop Out*) de 300mA. Con un voltaje de entrada de 5V se obtienen 3.3V y 1.2V de las fuentes conmutadas y 2.5V de la fuente lineal.

El voltaje de 1.2V se utiliza para la lógica interna del FPGA y se le asigna el nombre de VCCINT. El voltaje de 2.5V alimenta las terminales de programación (el JTAG) y se le asigna el nombre de VCCAUX. El voltaje de 3.3V alimenta los bancos de entradas/salidas del FPGA, se le asigna el nombre de VCCO y además alimenta a los demás circuitos integrados como al amplificador, el ADC, el DAC, el ICL3232 y también es el voltaje de entrada para los interruptores, tanto *push button* como deslizables.

El circuito integrado TPS75003 tiene 20 terminales y una conexión extra que se encuentra debajo del mismo, ésta última terminal se debe conectar a la tierra del sistema, para un mejor rendimiento del circuito. Una característica importante de este componente es el empaquetado llamado QFN (*Quad Flatpack No lead*), figura 4.19, ya que las terminales se encuentran por debajo del encapsulado y no sobresalen del mismo, esto lo hace complicado de soldar.

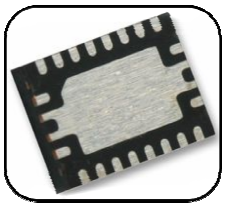


Figura 4.19. Encapsulado del circuito integrado TPS75003.

El diagrama esquemático de la fuente de alimentación para la tarjeta PRIUS se muestra en la figura 4.20. Cabe comentar que dicha fuente está basada en el circuito integrado TPS75003. En el diagrama esquemático se observan los componentes necesarios para obtener los voltajes deseados, así como los conectores tanto para la entrada como para las salidas.

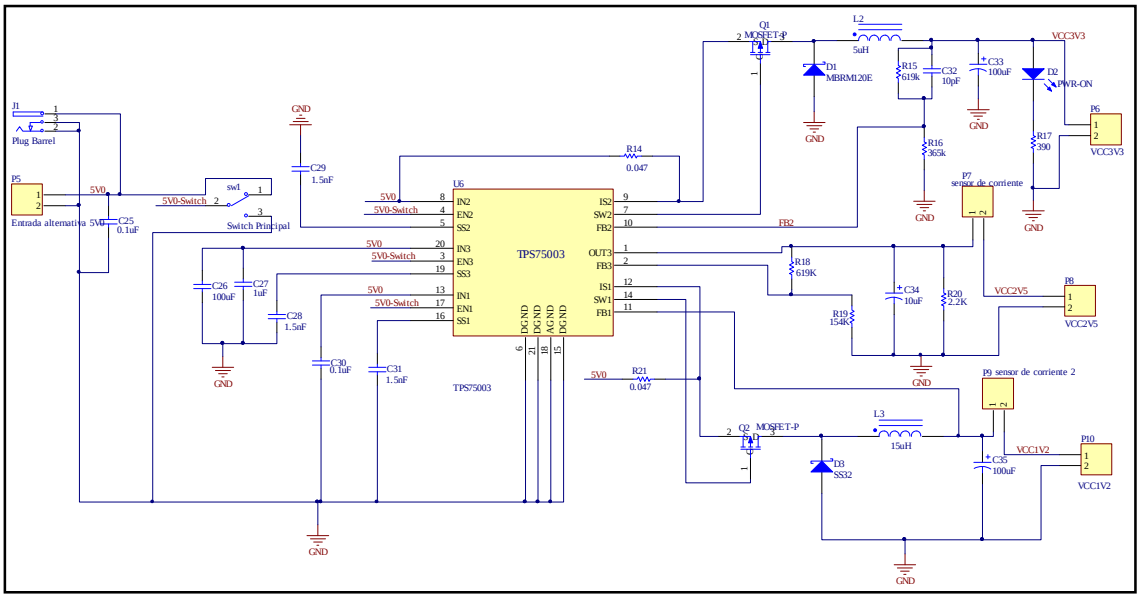


Figura 4.20. Diagrama esquemático de la fuente de alimentación.

Fuente de alimentación basada en el circuito integrado LTC3412

El circuito integrado LTC3412 es un convertidor DC/DC *step down* (*buck*) de alta eficiencia. El rango de voltaje de entrada es de 2.625V a 5.5V y proporciona un voltaje de salida entre 0.8V y 5V con una corriente de hasta 2.5A. Con un voltaje de entrada de 5V se obtiene un voltaje de salida de 2.5V y con un divisor de voltaje se obtiene el voltaje de 1.25V. Este regulador se utiliza para alimentar la memoria DDR y el banco de entradas del FPGA que controlan dicha memoria. También se utiliza, junto con el voltaje de 3.3V, como posible alimentación de un banco de entradas/salidas del FPGA.

El diagrama esquemático de la fuente de alimentación basada en el LTC3412 se muestra en la figura 4.21, donde se aprecia el divisor para generar el voltaje de 1.25V. El LTC3412 cuenta con 16 terminales y una terminal extra que se encuentra debajo del circuito integrado, esta terminal debe conectarse a la tierra del sistema para un mejor rendimiento del circuito integrado.

4.6. Configuración de la tarjeta PRIUS

Para poder configurar el FPGA con el *bitstream* generado, en la tarjeta de desarrollo SPARTAN 3E, se utiliza la interfaz USB-JTAG; para ello simplemente se conecta esta tarjeta a la computadora mediante un puerto USB, la computadora detecta la tarjeta mediante un controlador software y está lista para configurarse.

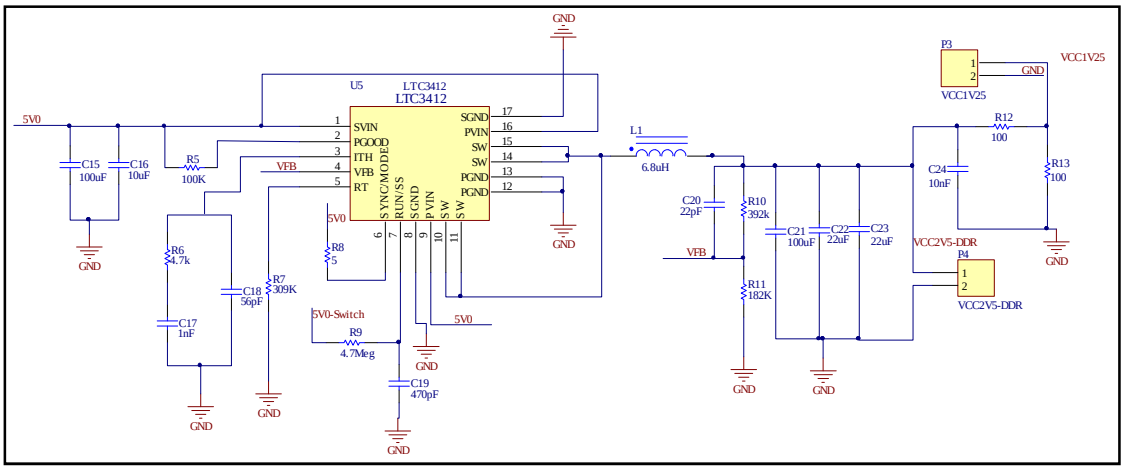


Figura 4.21. Diagrama esquemático de la fuente de alimentación.

La interfaz USB-JTAG es uno de los diseños más restringidos que se tienen en la tarjeta de desarrollo SPARTAN 3E, ya que en ella se realiza toda la lógica para realizar la conversión de protocolos de USB a JTAG. Por este motivo no se encontró información de cómo realizar esta lógica. Se buscó información de los fabricantes, tanto de Xilinx como de Digilent, pero no se encontró información de cómo desarrollar dicha conversión. Por este motivo no se pudo realizar el desarrollo de esta interfaz para la tarjeta PRIUS y se decidió realizar la configuración del FPGA directamente por el puerto JTAG.

La manera de configurar la tarjeta PRIUS se ilustra en la figura 4.22, donde se hace uso del módulo USB-JTAG de la tarjeta SPARTAN 3E. La SPARTAN 3E tiene disponible las terminales JTAG y es con estas terminales como se configura a la tarjeta PRIUS.

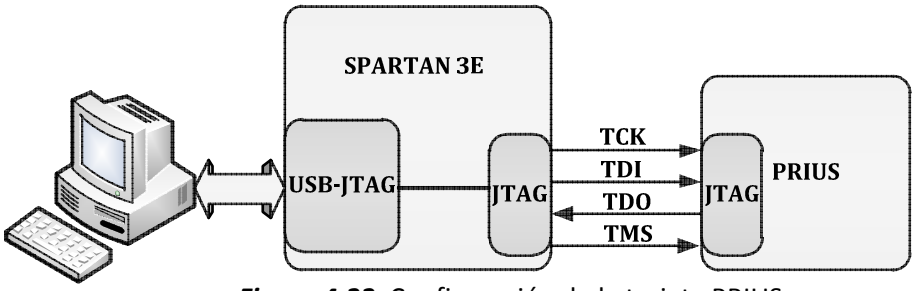


Figura 4.22. Configuración de la tarjeta PRIUS.

4.7. Circuito impreso final: Tarjeta PRIUS

La tarjeta PRIUS contiene todos los componentes del sistema de monitoreo y control al igual que los diseños realizados de los circuitos impresos para cada componente. Para diseñar la tarjeta se hicieron algunas consideraciones especiales que a continuación se describen.

4.7.1. Consideraciones especiales

Se hicieron, principalmente, tres consideraciones importantes: el número de capas de cobre, los tipos de vías y la distancia entre los componentes.

Selección del número de capas

Debido al número de terminales del FPGA y a los diferentes voltajes que maneja, se seleccionaron seis capas de cobre. La elección de seis capas fue el resultado de un convenio entre el número de terminales a utilizar y el precio de la fabricación de la PCB. Entre mayor número de capas, mayor cantidad de terminales disponibles y mayor el costo de fabricación. A las capas de una PCB se le conoce como *stack* y el orden del *stack* utilizado se muestra en la tabla 4.1, en ella se observa el número de capa, el nombre asignado, el orden en que se encuentran y la utilidad (el uso) de cada una de ellas.

Número de capa	Nombre	Orden	Utilidad
1	TOP	Superior (TOP)	Señales
2	Señal	Intermedia-1	Señales
3	3V3	Intermedia-2	3.3V,2.5V-DDR
4	GND	Intermedia-3	GND
5	5V0	Intermedia-4	1.2V,1.25V,2.5V,5V
6	BOTTOM	Inferior (BOTTOM)	Señales

Tabla 4.1. *Stack* de la PCB.

Cabe señalar que debido al número de capas, la tarjeta PRIUS se mandó fabricar, ya que no se cuenta con el equipo necesario para fabricar dichas PCBs. Una vez seleccionado el número de capas y el uso de cada una de ellas, el ruteo se realizó siguiendo las consideraciones generales para diseñar circuitos impresos.

Tipo de vías

Se conoce como vía a los orificios conductores que conectan dos o más capas de cobre en un circuito impreso. Existen tres tipos de vías, figura 4.23, las cuales son: *through hole*, que atraviesan toda la PCB; *blind* que son visibles sólo por un lado de la PCB ya sea el superior o el inferior; y *buried*, que se encuentran en capas intermedias de la PCB y no son visibles por ningún lado. La principal ventaja de utilizar las vías *buried* o *blind* es que se hace eficiente el uso de las capas ya que se puede rutear por debajo o por encima de éstas, mientras que en las vías *through hole* es imposible ya que atraviesan todas las capas. La desventaja de las vías *buried* y *blind* es el costo de fabricación ya que son más difíciles de fabricar y son más costosas. En el diseño realizado para la tarjeta PRIUS se utilizaron exclusivamente vías *through hole* debido a que el costo de fabricación era menor y eran suficientes para el ruteo de la PCB.

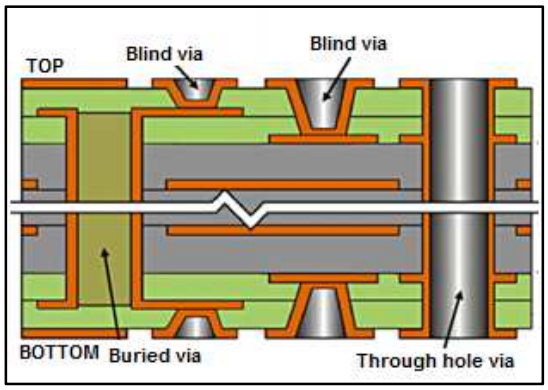


Figura 4.23. Tipos de vías en una PCB.

Una regla general del tamaño de las vías es que el diámetro total sea como mínimo el doble del diámetro del orificio, figura 4.24. Se utilizaron dos tamaños de vías, una de 0.2mm de diámetro del orificio con 0.4mm de diámetro total y 0.5mm de diámetro del orificio con 1mm de diámetro total. La limitante en el tamaño de las vías es la capacidad que tenga el fabricante de circuitos impresos para construirlas.

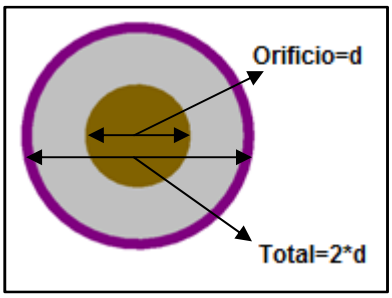


Figura 4.24. Tamaño de la vía.

Distancia entre componentes

Como se pretende soldar a mano, prácticamente todos los componentes de la PCB, la distancia entre componentes no debía ser muy pequeña ya que esto dificultaría el soldado de los mismos. A pesar de que se seleccionaron componentes pequeños, se procuró realizar la PCB con suficiente espacio para no tener problemas en el ensamblado; con base en estas consideraciones al final se limitó a un tamaño de 16cmx16cm. En este caso no se siguió ninguna regla en específico sobre la distancia.

Una vez hecho las consideraciones especiales, se describirán las características del diseño de la PCB.

4.7.2. Características del FPGA en la tarjeta PRIUS

Las características con las que cuenta el FPGA en la tarjeta PRIUS son básicamente tres: los voltajes en cada banco de entradas/salidas, las terminales de configuración y las terminales de entradas/salidas de propósito general disponibles.

Voltajes de bancos de entradas/salidas

El FPGA xc3s1600e cuenta con cuatro bancos de entradas/salidas, figura 4.25, los cuales se alimentan con los siguientes voltajes: el banco 0 se alimenta con 3.3V ó 2.5V, dependiendo de la posición del *jumper* que se incorporó para dicha selección, los bancos 1 y 2 se alimentan con 3.3V y el banco 3 con 2.5V, éste ultimo maneja la memoria DDR. Cabe comentar que la definición de estos voltajes le da mayor versatilidad al sistema.

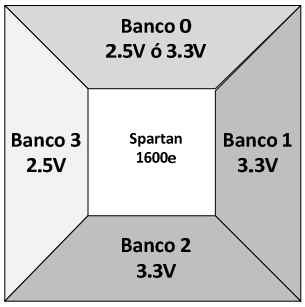


Figura 4.25. Voltajes en los bancos del FPGA.

Terminales de configuración

Como el método de configuración seleccionado es el JTAG, el diseño de la PCB se reduce, ya que solamente se necesitan once terminales para ello. Dichas terminales se muestran en la tabla 4.2, están incluidas las terminales del JTAG y las terminales de control del FPGA. La desventaja principal es que no se cuenta con una memoria externa para almacenar la configuración del FPGA y la principal ventaja es la reducción del diseño de la PCB.

Nombre	Descripción
TDI	Datos de entrada del JTAG
TDO	Datos de salida del JTAG
TMS	Datos de control del JTAG
TCK	Señal de reloj del JTAG
PROG_B	Se utiliza para borrar la configuración del FPGA
DONE	Indica cuando el FPGA está configurado correctamente
HSWAP	Se utiliza para colocar resistores de PULL UP a las terminales durante la configuración

Tabla 4.2. Terminales para la configuración del FPGA. (Continúa)

Nombre	Descripción
INIT_B	Indica cuando se está limpiando la memoria interna de configuración del FPGA
M0	Selecciona el modo de configuración, para JTAG debe tener un valor de 1
M1	Selecciona el modo de configuración, para JTAG debe tener un valor de 0
M2	Selecciona el modo de configuración, para JTAG debe tener un valor de 1

Tabla 4.2. Terminales para la configuración del FPGA.

Terminales de entradas/salidas

En la tarjeta PRIUS se utilizaron 207 terminales de las 320 que tiene el FPGA, 87 son del sistema de monitoreo y control, 40 son terminales disponibles para el usuario, 11 terminales son para la configuración, 41 terminales son para todos los voltajes de alimentación necesarios y 28 terminales de GND. No se utilizaron 113 terminales. En el apéndice C se muestra la lista de todas las terminales del FPGA con una breve descripción.

4.7.3. Circuito impreso

Para realizar el diagrama esquemático del FPGA, el software de diseño lo divide en ocho partes: cuatro bloques de los bancos de entradas/salidas, un bloque extra para las terminales que son únicamente entradas, un bloque del puerto JTAG, un bloque para los voltajes y un bloque para las terminales de GND. En la figura 4.26 se muestra una parte del diagrama esquemático realizado para el FPGA (bloque JTAG y bloque de terminales que son únicamente entradas).

En la figura 4.27, inciso *a*, se muestra la capa *TOP* de la tarjeta PRIUS y en el inciso *b* se muestra la capa *señal*. En la figura 4.28, inciso *a*, se muestra la capa *3V3* y en el inciso *b* se muestra la capa *GND*. En la figura 4.29 se muestran las capas *5V0* y *BOTTOM*, incisos *a* y *b* respectivamente. Por último en los incisos *a* y *b* de la figura 4.30 se muestra la tarjeta PRIUS completa vista desde la capa *TOP* y vista desde la capa *BOTTOM* respectivamente.

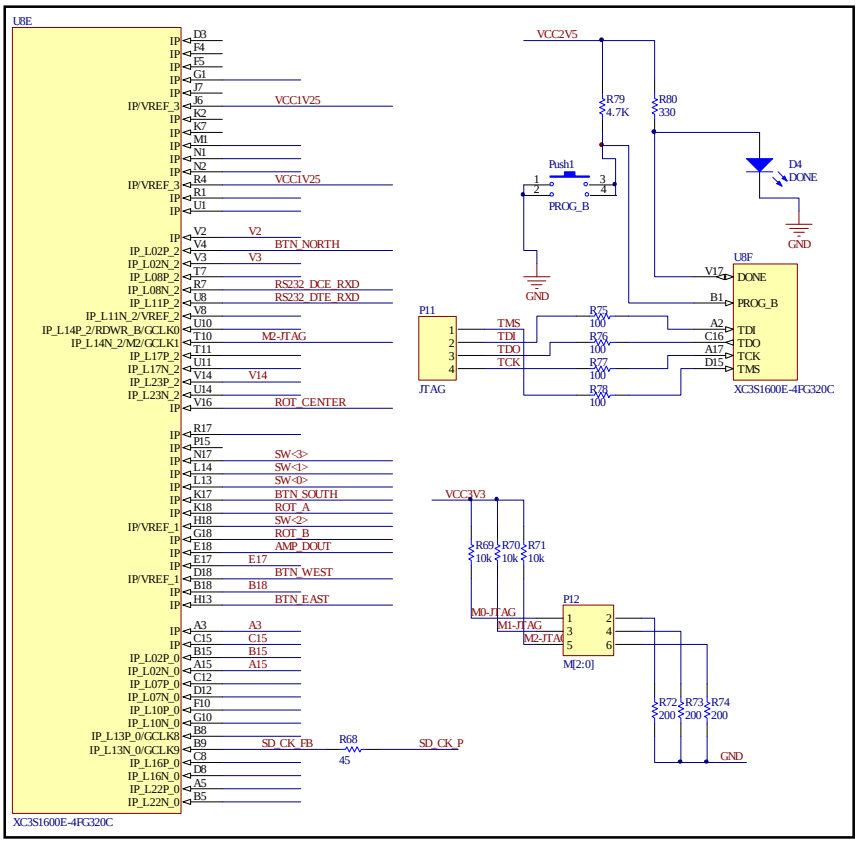
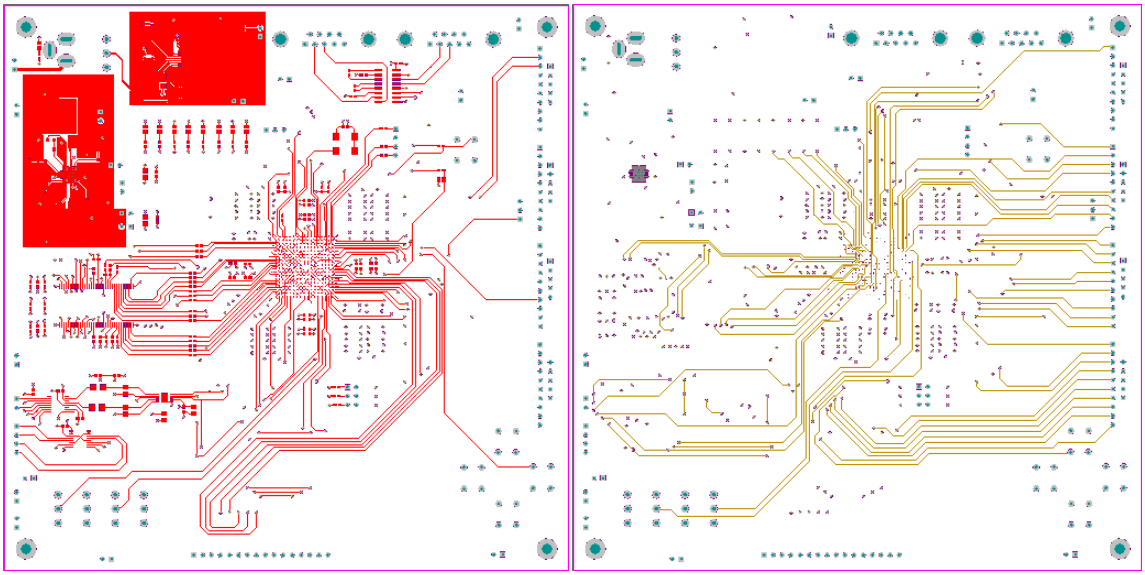
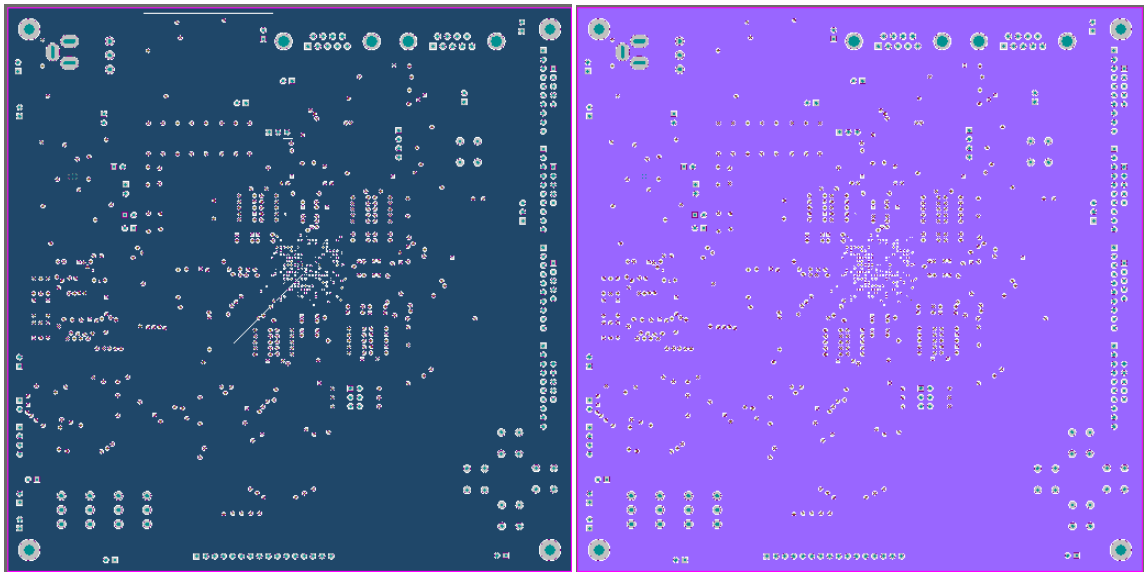


Figura 4.26. Diagrama esquemático del FPGA.



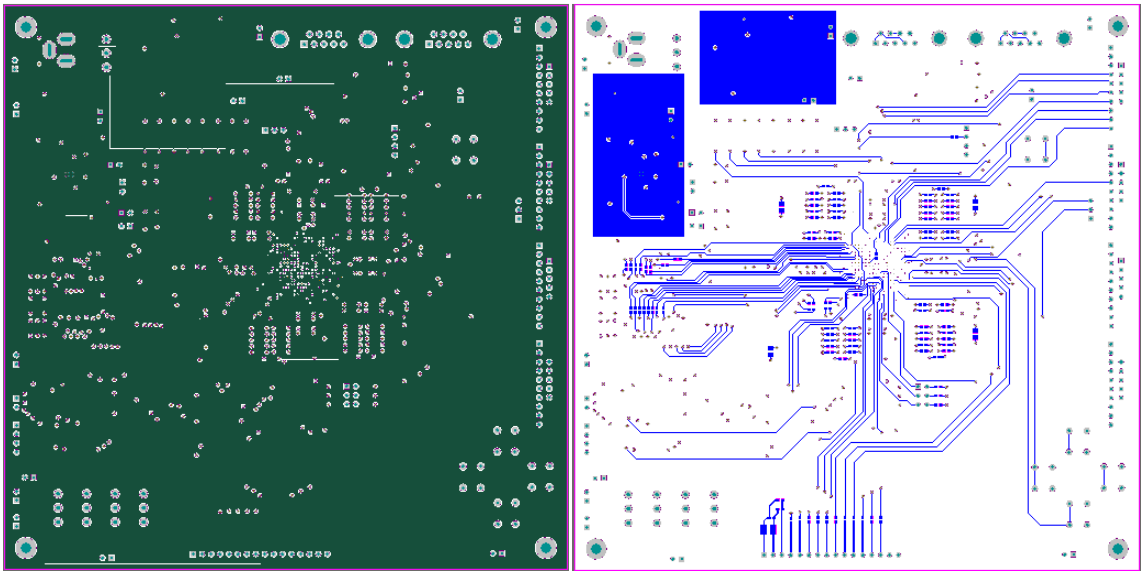
a) *TOP* b) *señal*

Figura 4.27. Capas *TOP* y *señal* de la tarjeta PRIUS.



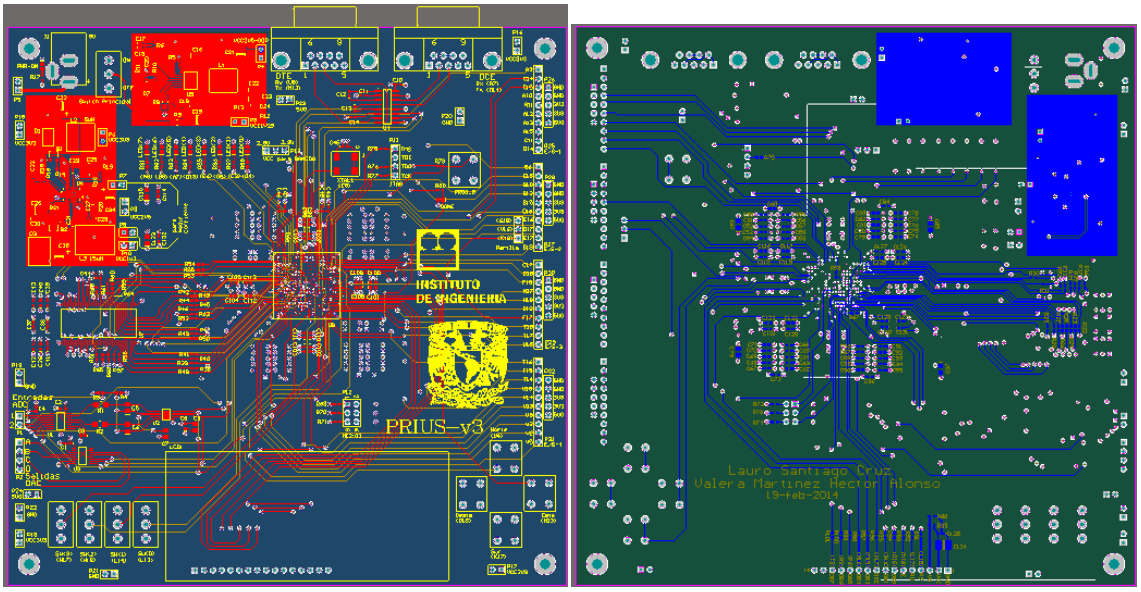
a) *3V3* b) *GND*

Figura 4.28. Capas *3V3* y *GND* de la tarjeta PRIUS.



a) *5V0* b) *BOTTOM*

Figura 4.29. Capas *5V0* y *BOTTOM* la tarjeta PRIUS.



a) Vista *TOP* b) Vista *BOTTOM*

Figura 4.30. Vistas de las capas de la tarjeta PRIUS.

La tarjeta PRIUS ya fabricada se muestra en las figuras 4.31 y 4.32, la cara *TOP* y la cara *BOTTOM* respectivamente.

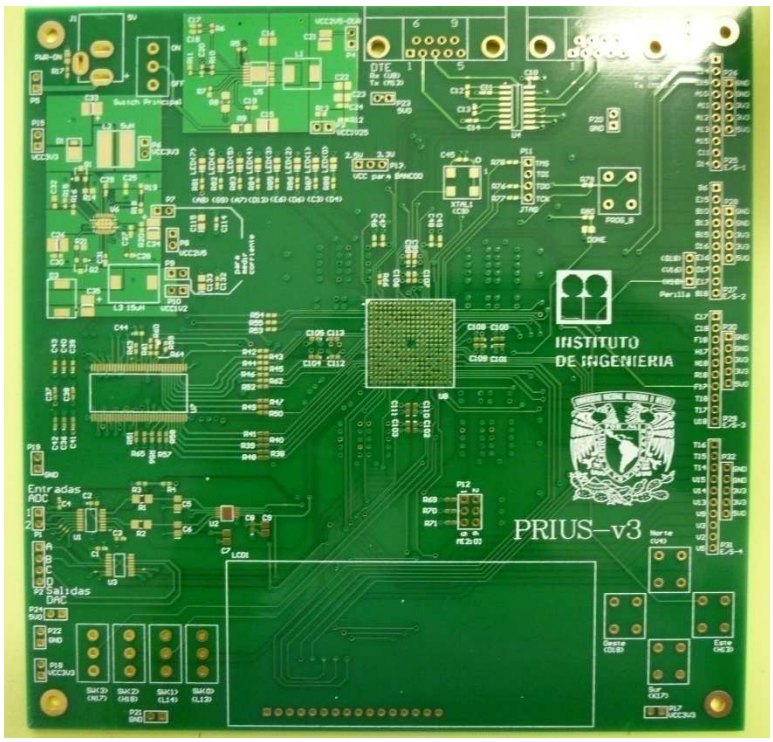


Figura 4.31. Cara *TOP* de la tarjeta PRIUS fabricada.

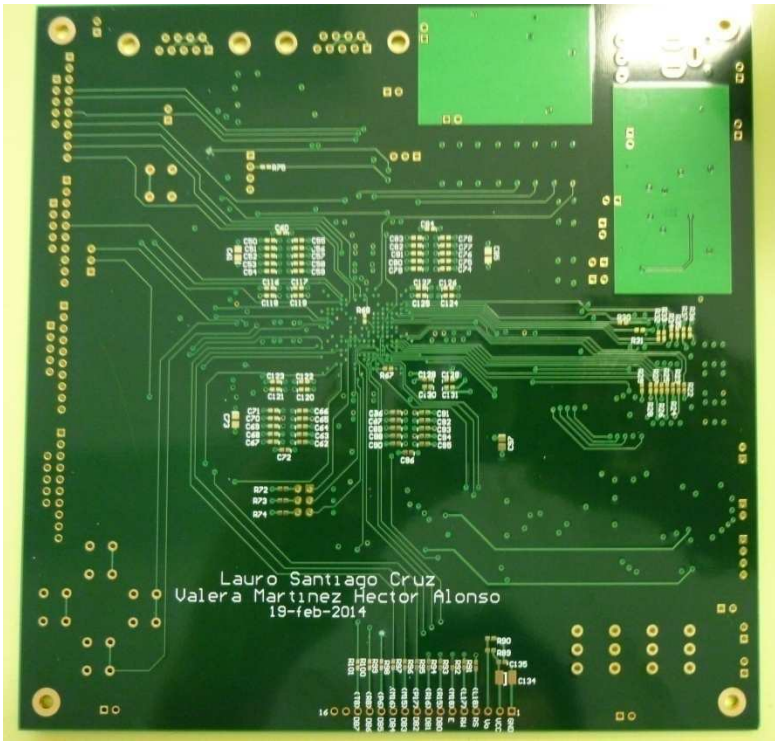


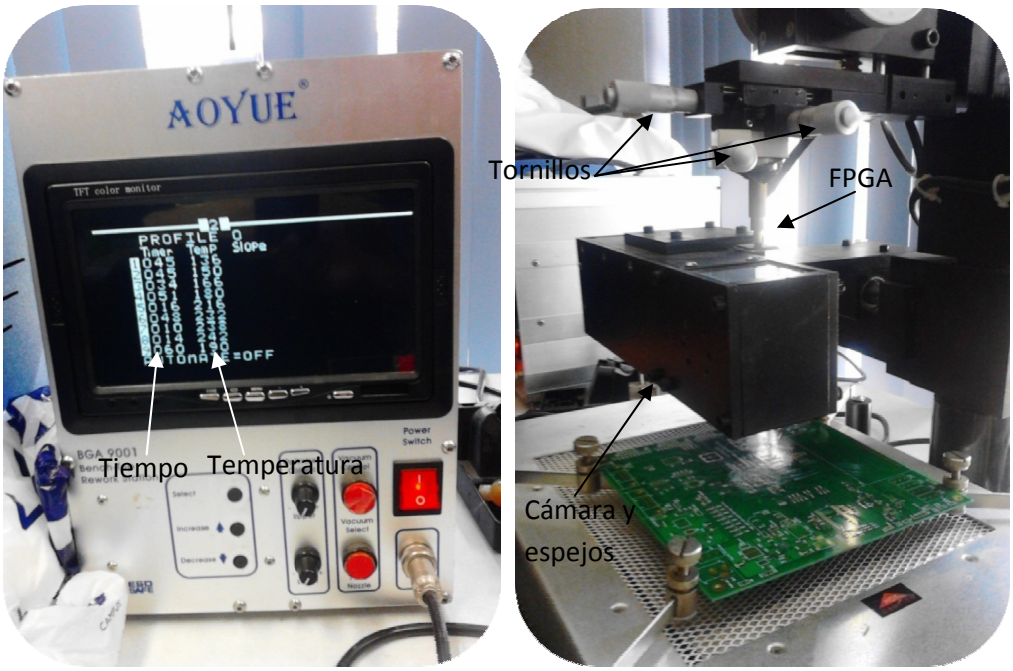
Figura 4.32. Cara *BOTTOM* de la tarjeta PRIUS fabricada.

4.8. Ensamblado de la tarjeta PRIUS

Ya que se tuvo la tarjeta fabricada, se probó la continuidad en cada una de las pistas para corroborar la correcta fabricación. Como no existió problema alguno, se procedió a soldar los componentes. Los primeros componentes en soldar fueron el FPGA y el TPS75003, quienes representaban mayor dificultad. Los otros componentes no representaban mayor dificultad y no hubo mayor problema en ensamblarlos.

4.8.1. Ensamblado del FPGA

Para soldar el FPGA se recurrió a una máquina para soldar componentes BGA marca AOYUE. Para llevar a cabo el proceso de soldado se requiere de tres aspectos principales: el perfil de temperaturas para soldar el componente, la alineación del componte y el control de temperatura. El perfil de temperatura se encuentra en las especificaciones del circuito integrado y se tiene que ingresar a la máquina. La alineación del componente se realiza manualmente con la ayuda de espejos, una cámara y tornillos. El control de temperatura se realiza mediante un sensor de temperatura que se puede colocar en diversas posiciones. En la figura 4.33, inciso *a*, se muestra el perfil de temperatura ingresado en la máquina y en el inciso *b* se muestra la alineación del componente.



a) Perfil de temperatura b) Alineación

Figura 4.33. Perfil de temperatura y alineación de la máquina para soldar.

En la figura 4.34 se muestra el sensor de temperatura y la fuente de calor de la máquina.

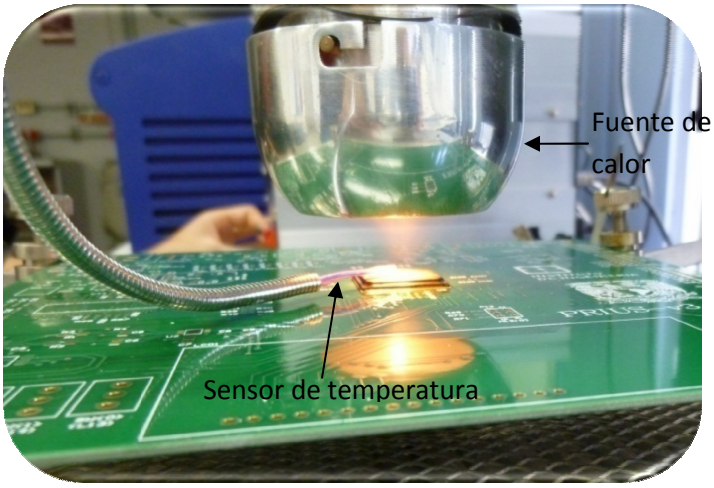
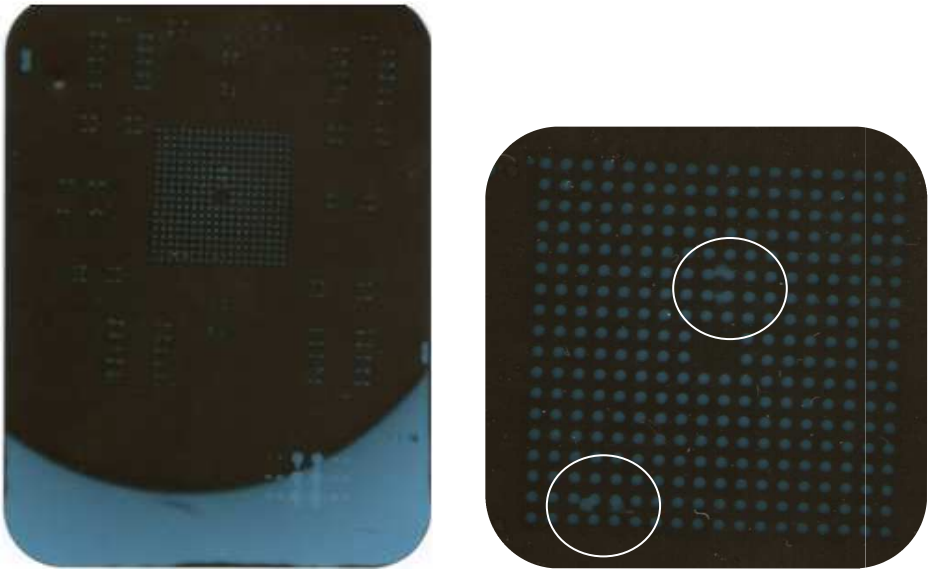


Figura 4.34. Sensor de temperatura de la máquina para soldar.

Al utilizar por primera vez la máquina para soldar se obtuvo un resultado malo, ya que se dañó por completo el circuito integrado, figura 4.35. Las dos posibles causas por las que se dañó el circuito integrado fueron: una, no precalentar el circuito integrado ni la PCB y dos, exceso de calor aplicado. Al ingresar el perfil de temperatura en la máquina, el aumento o disminución de la misma se realiza de manera automática con base en el sensor, lo que implica que la colocación del sensor es muy importante.

Figura 4.35. FPGA dañado.

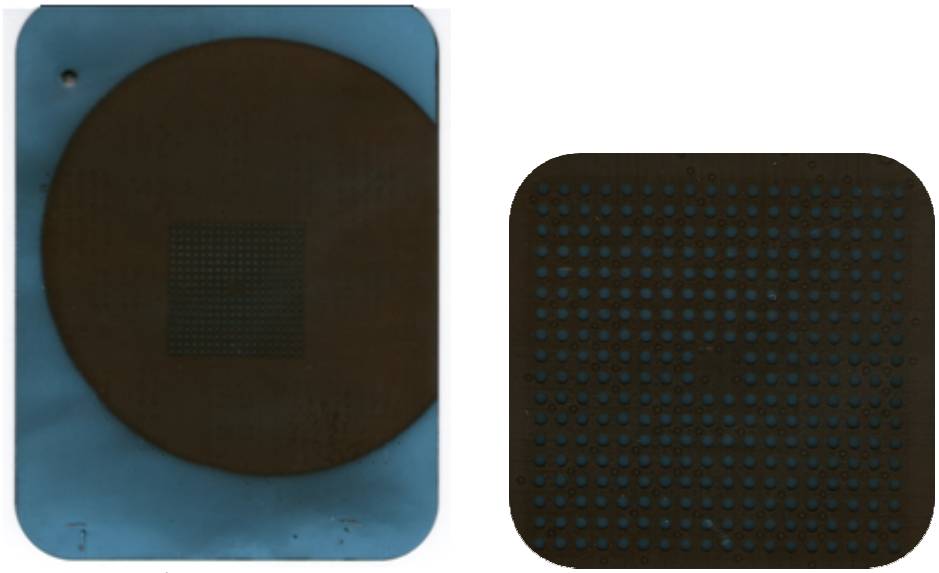
La segunda vez que se utilizó la máquina, precalentando el integrado y la PCB y controlando manualmente la temperatura (sin utilizar el perfil de temperatura en la máquina), se obtuvo un resultado relativamente bueno, ya que se logró colocar el componente y sin ningún tipo de daño, al menos a simple vista. Para corroborar que el soldado era correcto, se soldaron los componentes necesarios (fuentes de voltaje, capacitores, resistencias, *headers*, entre otros) para comprobar su funcionamiento. Una vez ensamblada la tarjeta PRIUS se realizaron pruebas de configuración con la ayuda de la tarjeta SPARTAN 3E, todas ellas sin resultados satisfactorios, ya que no era detectada por la computadora, por lo que no se podía realizar la configuración requerida. Como no se sabía si el soldado era correcto, se optó por tomar una radiografía del FPGA, la cual se muestra en la figura 4.36. En el inciso *a* se muestra la radiografía completa y en el inciso *b* se muestra la parte del FPGA. En la parte del FPGA se observan ciertas manchas entre las terminales que indican que está mal soldado, por lo que el proceso utilizado en la máquina no era el adecuado.



a) Completa b) FPGA

Figura 4.36. Radiografía del FPGA mal soldado.

Como el proceso que se utilizaba en la máquina no era el correcto, se decidió soldar el FPGA en una empresa dedicada a soldar este tipo de encapsulados. Esta decisión fue tomada con base en que se quería corrobora el funcionamiento de la PRIUS y no aprender a utilizar la máquina para soldar. Una vez soldado el componente, lo primero que se realizó esta vez, fue tomar una radiografía, figura 4.37. En el inciso *a* se muestra la radiografía completa mientras que en el inciso *b* se muestra la parte del FPGA, en ella, las terminales se observan uniformes y sin ningún tipo de alteración. Al realizar pruebas de configuración, soldando los componentes necesarios para este propósito, se obtuvieron resultados satisfactorios (ya que fue detectada la tarjeta PRIUS por la computadora y se logró configurarla), concluyendo así el ensamblado del FPGA y validando el diagrama esquemático diseñado.



a) Completa b) FPGA

Figura 4.37. Radiografía del FPGA bien soldado.

4.8.2. Ensamblado del circuito integrado TPS75003

Los factores importantes para soldar este dispositivo son la alineación del componente y el tipo de cautín y soldadura que se utilizarán. Se utilizó un microscopio para el manejo de este componente, ya que es muy pequeño y, a simple vista, es complicado manejarlo para soldarlo. Al alinearlo se tuvo cuidado en que todas las terminales coincidieran con el cobre de la PCB, tanto horizontal como verticalmente, si no sucede así, el componente queda desplazado aun teniendo algunas terminales alineadas por uno de sus lados. Cuando se colocó la soldadura, se tuvo cuidado que el cautín, con la punta más fina disponible, tuviera muy poca soldadura ya que si tiene demasiada, las terminales se unen por el exceso de la misma y se realizará más trabajo en quitar el exceso.

La primera vez que se soldó este componente se obtuvieron resultados poco satisfactorios, ya que dos de las tres fuentes que proporciona este circuito integrado funcionaban correctamente, mientras que la restante no daba el voltaje solicitado. Los voltajes correctos eran 3.3V y 1.2V,

mientras que el voltaje incorrecto era el de 2.5V. Después de revisar exhaustivamente el circuito y no encontrar falla alguna, se decidió soldar otro circuito integrado. La segunda vez que se soldó, se presentó el mismo problema, dos de las fuentes funcionaban correctamente mientras que una no proporcionaba el voltaje solicitado. En este caso los voltajes correctos fueron 2.5V y 1.2V mientras que el voltaje incorrecto fue el de 3.3V. En la figura 4.38 se muestran las terminales del componente, visto con el microscopio, aparentemente bien soldadas.

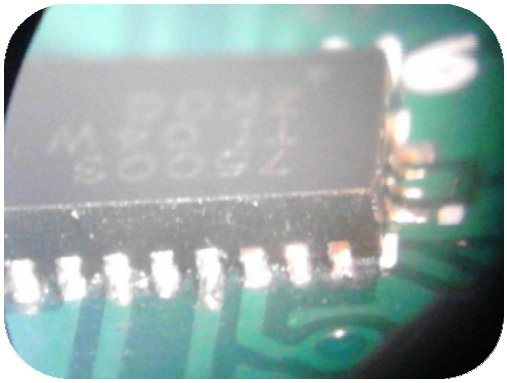


Figura 4.38. Terminales del circuito integrado TPS75003.

Debido a que el método que se estaba utilizando para soldar el componente no daba resultados satisfactorios, se decidió soldar el componente en una empresa dedicada al soldado de componentes. Al soldar el componente en una empresa se aseguraba, al menos en teoría, la unión correcta del componente en la PCB. Al realizar pruebas al componente los voltajes fueron los correctos ya que proporcionaban los voltajes solicitados: 3.3V, 2.5V y 1.2V. Con estos resultados se validó el diagrama esquemático diseñado y se concluyó el ensamblado de la fuente de alimentación principal.

4.8.3. Componentes restantes

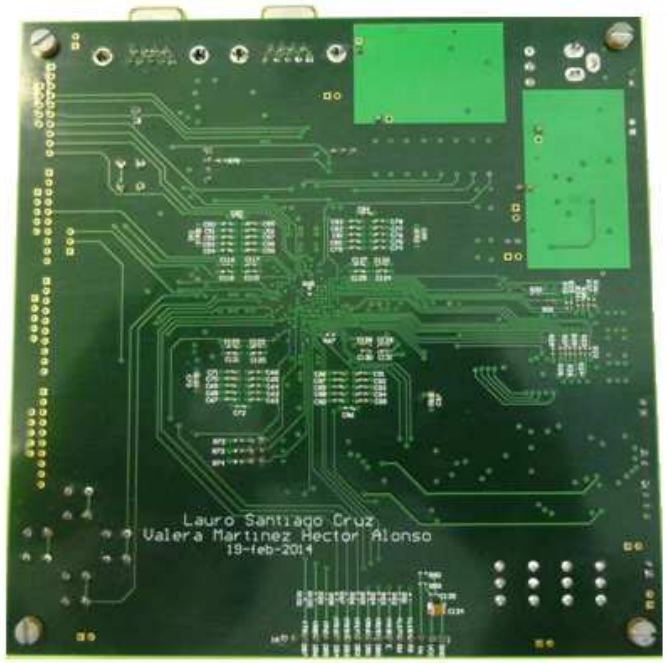
Los circuitos integrados LTC3412 e ICL3232, el amplificador, el ADC, el DAC, los LEDs, los capacitores, los resistores, la memoria DDR y el cristal de 50MHz no presentaron dificultad en el soldado, dando así, fin al ensamblado de la tarjeta PRIUS.

4.9. Tarjeta PRIUS

Una vez ensamblados todos los componentes se obtuvo la tarjeta PRIUS mostrada en la figura 4.39, en el inciso *a* la cara TOP y en el inciso *b* la cara BOTTOM.

La tarjeta PRIUS cuenta con un FPGA SPARTAN xc3s1600e, un ADC de dos canales, un DAC de 4 canales, dos puertos para comunicación RS-232, un cristal de 50MHz, 40 terminales de entrada/salida, 8 LEDs, 4 interruptores deslizables, 4 interruptores *push button*, un display LCD de 2x16, voltajes disponibles de GND, 3.3V y 5V, una memoria DDR RAM de 512 Mbits y un puerto JTAG.

a) Cara *TOP*



b) Cara *BOTTOM*

Figura 4.39. Tarjeta PRIUS.

En este capítulo se describió de manera detallada los aspectos fundamentales para la construcción de la tarjeta PRIUS. En el siguiente capítulo se presentarán las pruebas realizadas a los sistemas que se elaboraron, entre ellos la tarjeta PRIUS.

Capítulo V
Pruebas realizadas

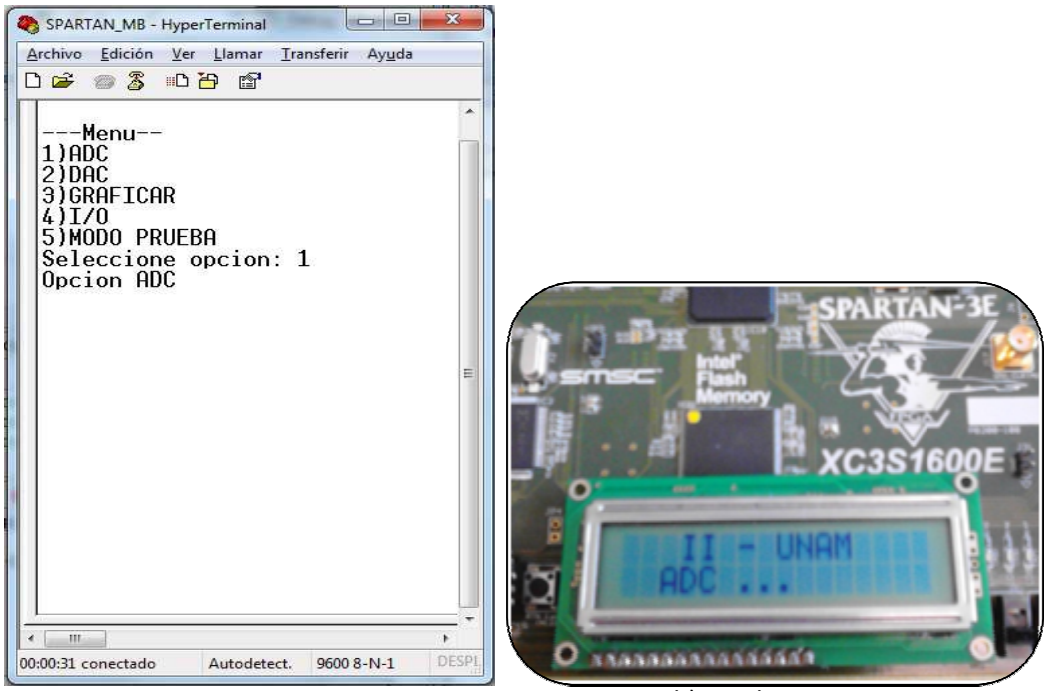
En este capítulo se presentan las pruebas que se realizaron a las tarjetas de desarrollo SPARTAN 3E y PRIUS. Las pruebas realizadas fueron con el software diseñado para MicroBlaze.

5.1. Tarjeta SPARTAN 3E

Las pruebas que se realizaron en la tarjeta de desarrollo SPARTAN 3E consisten en corroborar el funcionamiento de las opciones que ofrece el software diseñado, éstas son las correspondientes a: 1) ADC, 2) DAC, 3) GRAFICAR, 4) I/O y 5) MODO PRUEBA.

1) ADC

En la opción correspondiente al ADC se activa el convertidor analógico digital y además se almacenan los datos de la conversión en la memoria DDR para que posteriormente sean leídos. En la comunicación asíncrona se utiliza *Hyperterminal*. En la figura 5.1 inciso *a* se muestra un mensaje donde se indica que se ha seleccionado la opción 1 y en el display LCD, figura 5.1 inciso *b*, se muestra en la segunda línea del display LCD el mensaje "ADC …".



a) *Hyperterminal* b) Display LCD

Figura 5.1. Mensajes de la opción 1.

La primera prueba que se realizó para comprobar que la conversión era correcta fue con un voltaje de corriente directa que se ingresó en ambos canales de conversión. Para observar los datos almacenados en la memoria DDR se utilizó una pestaña llamada MEMORY en el modo DEBUG de SDK, figura 5.2. Con el modo DEBUG el software se puede ejecutar paso a paso, ejecutar de manera continua, pausar, iniciar y parar la ejecución. Por lo que se ejecutó el software, se ingresó la opción 1 mediante *Hyperterminal* y se detuvo la ejecución. Después se buscó la dirección de la memoria y se observaron los datos almacenados.

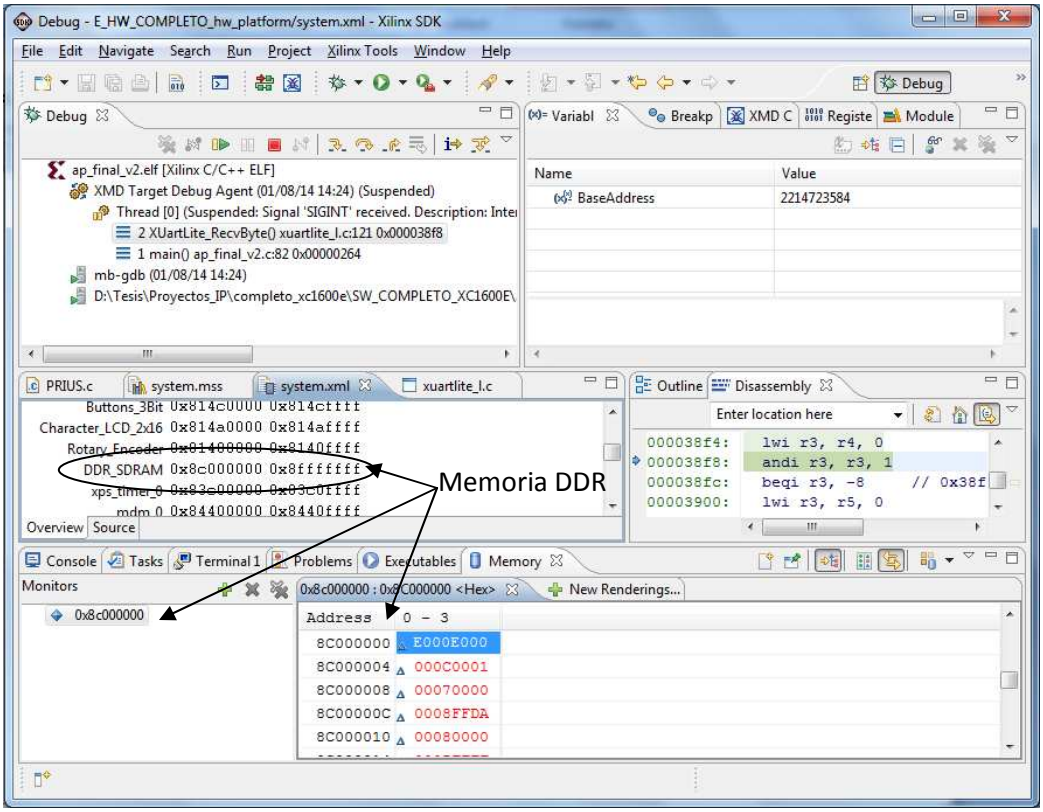


Figura 5.2. Modo DEBUG en SDK.

Con los datos observados en la memoria DDR se obtuvo la tabla 5.1. En dicha tabla se observa el voltaje ingresado a cada canal, la entrada diferencial y la salida con ganancia del amplificador, el valor teórico de la conversión, mediante la aplicación de la ecuación 3.7, y las salidas de ambos convertidores en formato decimal y en formato de voltaje (el análisis de la conversión se describe en el capítulo 3 apartado 3.3.2). Estos valores se obtuvieron con una ganancia unitaria y el rango de voltaje de entrada para esta ganancia es de 0.4V a 2.9V, como se muestra en dicha tabla.

En la figura 5.3 se muestra una gráfica del voltaje de entrada contra el voltaje de salida del convertidor para los tres casos: la conversión teórica (en azul), el canal A y el canal B (ambos en rojo). En el caso teórico, la gráfica debe ser una línea recta porque el voltaje de entrada es igual al voltaje de salida, mientras que para el caso real, es decir, el canal A y B, se observan algunas

desviaciones con respecto al valor teórico. Las desviaciones (error relativo) entre el voltaje de entrada y el voltaje de salida no son mayores al 1.15%.

Entrada	**Entrada diferencial**	**Salida con ganancia**	**Ec 3.7**	**Salida**		**Salida (en Volts)**	
VA=VB	amplificador	amplificador	D[13:0]	Canal A	Canal B	Canal A	Canal B
0.1634	-1.4866	1.4866	9742.5818	8191	8191	0.4002	0.4002
0.2424	-1.4076	1.4076	9224.8474	8191	8191	0.4002	0.4002
0.3191	-1.3309	1.3309	8722.1862	8191	8191	0.4002	0.4002
0.4214	-1.2286	1.2286	8051.753	8191	8191	0.4002	0.4002
0.5229	-1.1271	1.1271	7386.5626	7426	7410	0.5169	0.5193
0.6115	-1.0385	1.0385	6805.9136	6845	6827	0.6055	0.6083
0.7195	-0.9305	0.9305	6098.1248	6138	6119	0.7134	0.7163
0.8065	-0.8435	0.8435	5527.9616	5565	5547	0.8008	0.8036
0.9052	-0.7448	0.7448	4881.1213	4910	4861	0.9008	0.9083
1.041	-0.609	0.609	3991.1424	4015	3963	1.0374	1.0453
1.107	-0.543	0.543	3558.6048	3551	3499	1.1082	1.1161
1.207	-0.443	0.443	2903.2448	2915	2863	1.2052	1.2131
1.313	-0.337	0.337	2208.5632	2215	2206	1.312	1.3134
1.427	-0.223	0.223	1461.4528	1465	1454	1.4265	1.4281
1.511	-0.139	0.139	910.9504	916	866	1.5102	1.5179
1.615	-0.035	0.035	229.376	231	222	1.6148	1.6161
1.714	0.064	-0.064	-419.4304	-417	-464	1.7136	1.7208
1.816	0.166	-0.166	-1087.898	-1080	-1086	1.8148	1.8157
1.917	0.267	-0.267	-1749.811	-1741	-1788	1.9157	1.9228
2.023	0.373	-0.373	-2444.493	-2438	-2441	2.022	2.0225
2.123	0.473	-0.473	-3099.853	-3092	-3093	2.1218	2.122
2.203	0.553	-0.553	-3624.141	-3624	-3633	2.203	2.2044
2.314	0.664	-0.664	-4351.59	-4350	-4350	2.3138	2.3138
2.413	0.763	-0.763	-5000.397	-4997	-5034	2.4125	2.4181
2.516	0.866	-0.866	-5675.418	-5674	-5714	2.5158	2.5219
2.601	0.951	-0.951	-6232.474	-6231	-6224	2.6008	2.5997
2.707	1.057	-1.057	-6927.155	-6918	-6914	2.7056	2.705
2.803	1.153	-1.153	-7556.301	-7552	-7553	2.8023	2.8025
2.909	1.259	-1.259	-8250.982	-8192	-8192	2.9	2.9
3.011	1.361	-1.361	-8919.45	-8192	-8192	2.9	2.9
3.116	1.466	-1.466	-9607.578	-8192	-8192	2.9	2.9
3.201	1.551	-1.551	-10164.63	-8192	-8192	2.9	2.9
3.304	1.654	-1.654	-10839.65	-8192	-8192	2.9	2.9

Tabla 5.1. Conversión con ganancia de -1.

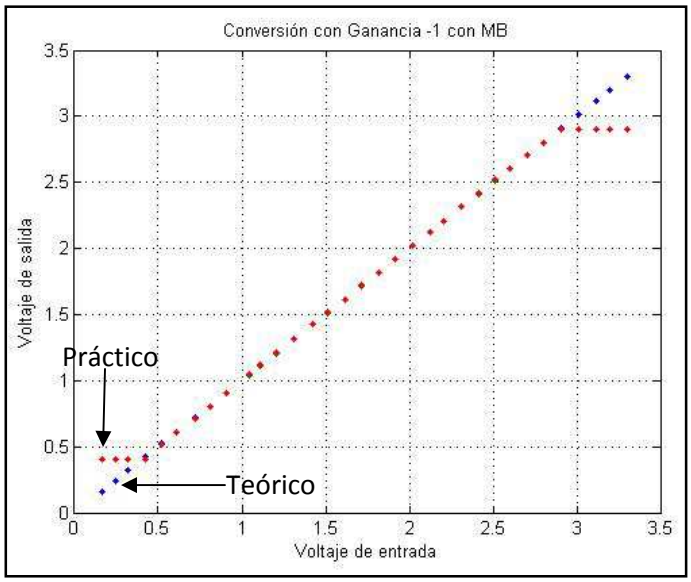


Figura 5.3. Gráfica para una ganancia de -1.

En la tabla 5.2 se muestra la conversión ahora con una ganancia de -2, el rango de voltaje de entrada para esta ganancia es de 1.025V a 2.275V. En la figura 5.4 se muestra una gráfica del voltaje de entrada contra el voltaje de salida del convertidor para los tres casos: la conversión teórica (en azul), el canal A y el canal B (ambos en rojo). Las desviaciones (error relativo) entre el voltaje de entrada y el voltaje de salida no son mayores al 1%.

Entrada	**Entrada diferencial**	**Salida con ganancia**	**Ec 3.7**	**Salida**		**Salida (en Volts)**	
VA=VB	amplificador	amplificador	D[13:0]	Canal A	Canal B	Canal A	Canal B
0.905	-0.745	1.49	9764.864	8191	8191	1.0251	1.0251
0.957	-0.693	1.386	9083.2896	8191	8191	1.0251	1.0251
1.006	-0.644	1.288	8441.0368	8191	8191	1.0251	1.0251
1.051	-0.599	1.198	7851.2128	7888	7841	1.0482	1.0518
1.1	-0.55	1.1	7208.96	7257	7210	1.0963	1.0999
1.149	-0.501	1.002	6566.7072	6623	6611	1.1447	1.1456
1.202	-0.448	0.896	5872.0256	5931	5921	1.1975	1.1983
1.251	-0.399	0.798	5229.7728	5287	5241	1.2466	1.2501
1.3	-0.35	0.7	4587.52	4652	4646	1.2951	1.2955
1.35	-0.3	0.6	3932.16	3987	3942	1.3458	1.3492
1.403	-0.247	0.494	3237.4784	3289	3240	1.3991	1.4028
1.453	-0.197	0.394	2582.1184	2654	2605	1.4475	1.4513
1.501	-0.149	0.298	1952.9728	2028	2021	1.4953	1.4958
1.55	-0.1	0.2	1310.72	1386	1378	1.5443	1.5449
1.6	-0.05	0.1	655.36	725	719	1.5947	1.5951
1.649	-0.001	0.002	13.1072	79	33	1.644	1.6475
1.7	0.05	-0.1	-655.36	-601	-607	1.6959	1.6963

Tabla 5.2. Conversión con ganancia de -2. (Continúa)

Entrada	Entrada diferencial	Salida con ganancia	Ec 3.7	Salida		Salida (en Volts)	
VA=VB	amplificador	amplificador	D[13:0]	Canal A	Canal B	Canal A	Canal B
1.751	0.101	-0.202	-1323.827	-1264	-1311	1.7464	1.75
1.8	0.15	-0.3	-1966.08	-1912	-1921	1.7959	1.7966
1.85	0.2	-0.4	-2621.44	-2565	-2571	1.8457	1.8462
1.9	0.25	-0.5	-3276.8	-3198	-3204	1.894	1.8944
1.95	0.3	-0.6	-3932.16	-3849	-3855	1.9437	1.9441
2	0.35	-0.7	-4587.52	-4501	-4548	1.9934	1.997
2.052	0.402	-0.804	-5269.094	-5194	-5241	2.0463	2.0499
2.101	0.451	-0.902	-5911.347	-5835	-5843	2.0952	2.0958
2.152	0.502	-1.004	-6579.814	-6502	-6548	2.1461	2.1496
2.201	0.551	-1.102	-7222.067	-7129	-7135	2.1939	2.1944
2.252	0.602	-1.204	-7890.534	-7799	-7808	2.245	2.2457
2.272	0.622	-1.244	-8152.678	-8055	-8063	2.2645	2.2652
2.303	0.653	-1.306	-8559.002	-8192	-8192	2.275	2.275
2.351	0.701	-1.402	-9188.147	-8192	-8192	2.275	2.275
2.402	0.752	-1.504	-9856.614	-8192	-8192	2.275	2.275

Tabla 5.2. Conversión con ganancia de -2.

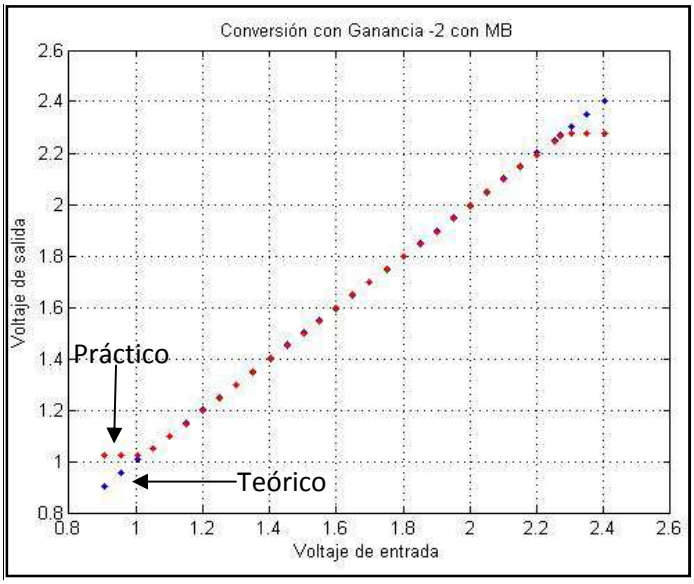


Figura 5.4. Gráfica para una ganancia de -2.

Por último, en la tabla 5.3 se muestra la conversión ahora con una ganancia de -50, el rango de voltaje de entrada para esta ganancia es de 1.625V a 1.675V. En la figura 5.5 se muestra una gráfica del voltaje de entrada contra el voltaje de salida del convertidor para los tres casos: la conversión teórica (en azul), el canal A y el canal B (ambos en rojo). Las desviaciones (error relativo) entre el voltaje de entrada y el voltaje de salida no son mayores al 1%.

Entrada	Entrada diferencial	Salida con ganancia	Ec 3.7	Salida		Salida (en Volts)	
VA=VB	amplificador	amplificador	D[13:0]	Canal A	Canal B	Canal A	Canal B
1.61	-0.04	2	13107.2	8191	8191	1.625	1.625
1.627	-0.023	1.15	7536.64	8191	8191	1.625	1.625
1.631	-0.019	0.95	6225.92	7555	7529	1.6269	1.627
1.634	-0.016	0.8	5242.88	6879	6861	1.629	1.6291
1.641	-0.009	0.45	2949.12	4309	4321	1.6368	1.6368
1.644	-0.006	0.3	1966.08	3343	3349	1.6398	1.6398
1.648	-0.002	0.1	655.36	2125	2096	1.6435	1.6436
1.652	0.002	-0.1	-655.36	979	943	1.647	1.6471
1.661	0.011	-0.55	-3604.5	-1949	-1959	1.6559	1.656
1.664	0.014	-0.7	-4587.5	-2976	-3018	1.6591	1.6592
1.669	0.019	-0.95	-6225.9	-5174	-5185	1.6658	1.6658
1.675	0.025	-1.25	-8192	-6884	-6903	1.671	1.6711
1.681	0.031	-1.55	-10158	-8192	-8192	1.675	1.675

Tabla 5.3. Conversión con ganancia de -50.

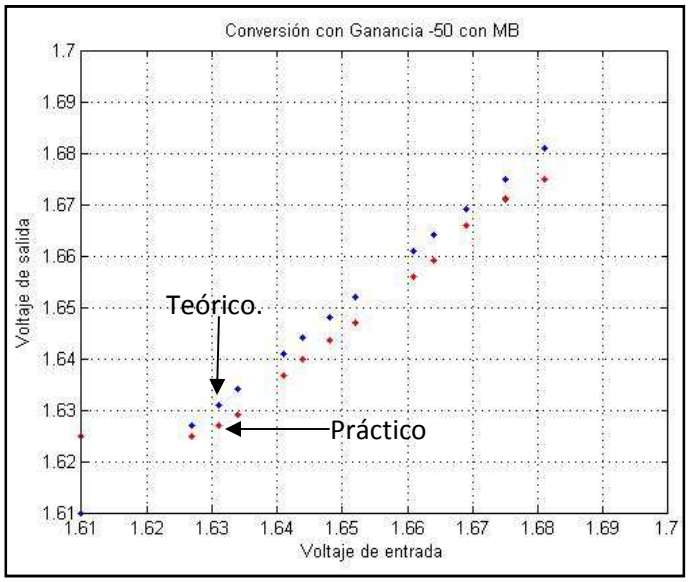


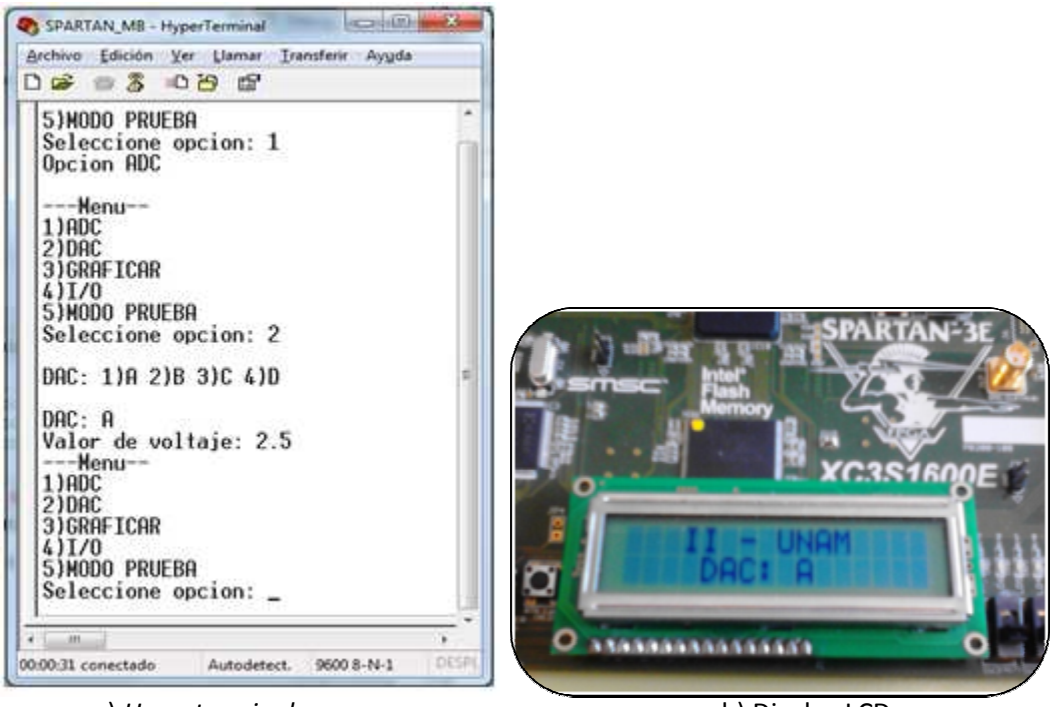
Figura 5.5. Gráfica para una ganancia de -50.

Observando las tablas 5.1, 5.2 y 5.3 se concluye que la conversión analógica digital es correcta y que además el almacenamiento de información en la memoria DDR también es correcto.

2) DAC

En esta opción se activan los cuatro canales, denominados A, B, C y D, del convertidor digital analógico. Mediante *Hyperterminal* se ingresa el canal del DAC que realizará la conversión, y

también se ingresa el valor de voltaje que se requiera en formato de un entero y un decimal. En la figura 5.6 inciso *a* se observa la pantalla de *Hyperterminal* en donde se selecciona el DAC y se ingresa el voltaje deseado. En la figura 5.6 inciso *b* se observa el display LCD que muestra el mensaje "DAC: A" que es el DAC seleccionado.



a) *Hyperterminal* b) Display LCD

Figura 5.6. Mensajes de la opción 2.

En la tabla 5.4 se muestran los valores ingresados mediante la opción 2 y los valores medidos en los cuatro canales del DAC: A, B, C y D, también se observa el porcentaje de error que existe entre cada medición, el cual no es mayor al 3%, y el voltaje de referencia de cada DAC. El tiempo que se tarda en actualizarse la salida de los DACs es de 20.4µs.

Voltaje de entrada		**Voltaje Medido**				**%Error**			
A y B Vref=3.294	C y D Vref=2.575	A	B	C	D	A	B	C	D
0.4	0.3	0.3988	0.3988	0.3042	0.3064	0.3%	0.3%	1.4%	2.13%
0.8	0.6	0.797	0.797	0.609	0.612	0.37%	0.38%	1.5%	2%
1.2	0.9	1.197	1.198	0.915	0.918	0.25%	0.17%	1.67%	2%
1.6	1.2	1.597	1.597	1.22	1.223	0.18%	0.19%	1.67%	1.92%
2	1.5	1.997	1.998	1.526	1.529	0.15%	0.1%	1.73%	1.93%
2.4	1.8	2.397	2.398	1.833	1.835	0.13%	0.08%	1.83%	1.94%
2.8	2.1	2.798	2.798	2.138	2.14	0.07%	0.07%	1.81%	1.90%
3.2	2.4	3.198	3.198	2.444	2.447	0.06%	0.06%	1.83%	1.96%

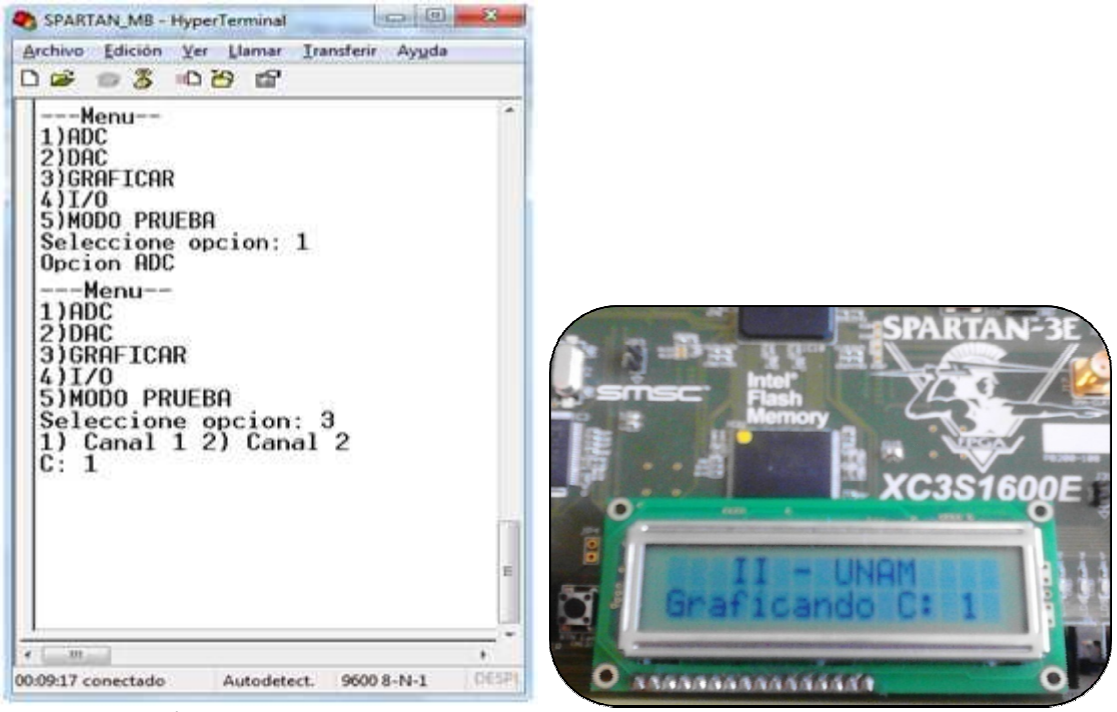
Tabla 5.4. Voltajes de salida y porcentaje de error en el DAC.

Con los resultados de la tabla 5.4 se concluye que la comunicación con los DACs es correcta, además de corroborar el funcionamiento de la comunicación asíncrona.

3) GRAFICAR

En esta opción se leen los datos almacenados en la memoria DDR y se envían por la comunicación UART, esta opción es complemento de la opción uno ya que se envían los datos adquiridos con el ADC. Para poder observar los datos enviados, se desarrolló un programa en MATLAB que adquiere los datos y los grafica. El programa consiste básicamente en leer cierto número de datos del puerto serie y colocarlos en una gráfica.

Se selecciona la opción tres mediante *Hyperterminal* y también se selecciona el canal que se quiera transmitir, ya sea el canal A (Canal 1) ó el canal B (canal 2). Inmediatamente de que se selecciona el canal, se envían los datos a un puerto UART. En la figura 5.7 inciso *a* se muestra la pantalla de *Hyperterminal* y en el inciso *b* se muestra el display LCD con el mensajes "Graficando C: 1" que es el canal seleccionado.



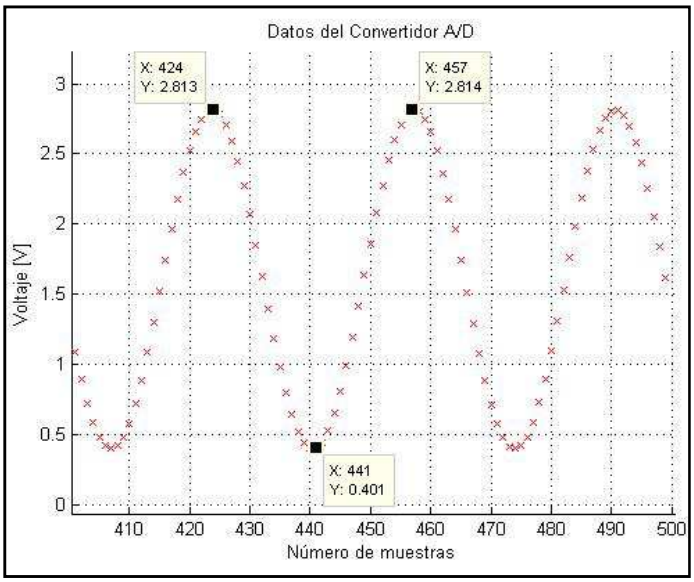
a) *Hyperterminal* b) Display LCD

Figura 5.7. Mensajes de la opción 3.

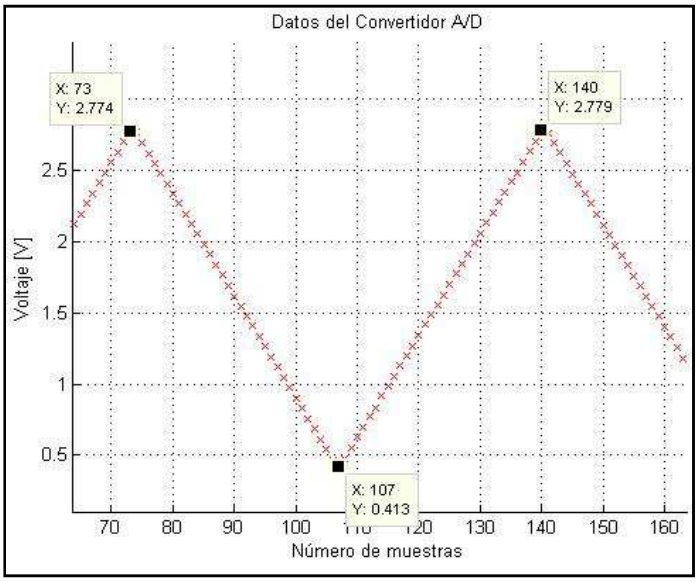
Se utilizan los dos puertos UART de la tarjeta SPARTAN 3E, ya que uno se utiliza con *Hyperterminal* y el otro se utiliza con MATLAB.

Para comprobar el funcionamiento de la opción tres, se ingresó una señal senoidal en el canal A (canal 1) y una señal triangular en el canal B (canal 2), se adquirieron los datos y después se graficaron dichos canales, todo esto con una ganancia unitaria en el amplificador. En la figura 5.8 se muestran las dos gráficas que se obtuvieron en MATLAB de ambos canales, en el inciso *a* del

canal A (canal 1) y en el inciso *b* del canal B (canal 2). El ADC adquiere un dato cada 29.8µs correspondiente a una frecuencia de 34.013kHz. El tiempo de 29.8µs incluye el almacenamiento y el procesamiento de los datos (ajustarlos a un número de 16 bits) en la memoria DDR.



a) Señal senoidal



b) Señal triangular

Figura 5.8. Gráficas en MATLAB de la opción 3.

En la tabla 5.5 se muestran los parámetros de las señales de entrada y las señales de salida que se obtuvieron con MATLAB. Para obtener la frecuencia de las señales graficadas en MATLAB se multiplica el número de muestras por el tiempo de adquisición del ADC y se obtiene el inverso de dicho producto. Para el caso de la onda senoidal las muestras son: 457-424=33 (de los marcadores de la figura 5.8), estas 33 muestra se multiplican por 29.8µs: 33* 29.8µs = 983.4µs y el inverso es

1.0169 kHz. Para el caso de la onda triangular es 140-73=67 (de los marcadores de la figura 5.8), multiplicadas por el tiempo de adquisición: 67*29.8µs=1.9966ms y el inverso es 500.8514Hz.

Parámetro/ Valor	**Entrada**		**Salida**		**% Error**	
	Senoidal	Triangular	Senoidal	Triangular	Senoidal	Triangular
Vmax	2.88V	2.88V	2.813V	2.779V	2.3264%	3.5069%
Vmin	0.4V	0.4V	0.401V	0.413V	0.25%	3.25%
Vpp	2.48V	2.48V	2.412V	2.366V	2.7419	4.5968%
Frecuencia	1 kHz	500 Hz	1.0169kHz	500.8514Hz	2%	0.17%

Tabla 5.5. Parámetros de las señales en MATLAB.

Como los porcentajes de error son menores al 5% se corrobora el buen funcionamiento de la opción 3, tanto la comunicación UART como el proceso de conversión analógica digital.

4) I/O

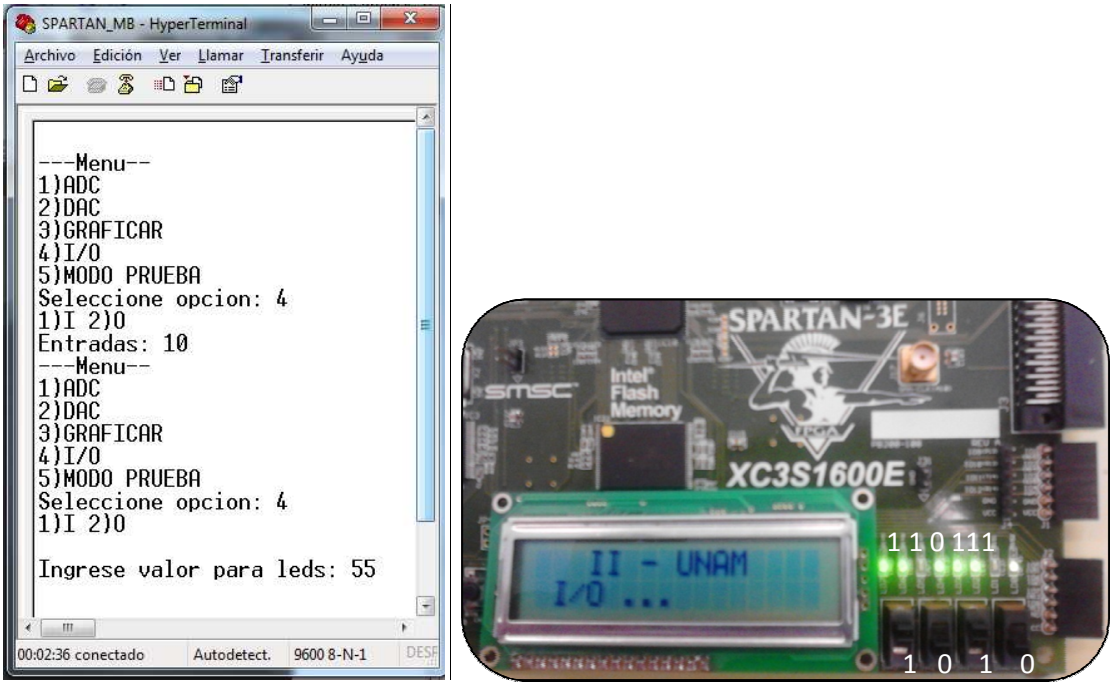
En esta opción se lee el estado de los interruptores deslizables y se controla el encendido/apagado de los LEDs. Mediante *Hyperterminal* se selecciona entre leer el estado de los interruptores ó activar los LEDs. Para el caso de los LEDs se ingresa el valor en base decimal y se despliega en base binario, mientras que para los interruptores se lee el estado en binario y se despliega en *Hyperterminal* en base decimal. En el inciso *a* de la figura 5.9 se muestra la pantalla de *Hyperterminal* en donde se observan la lectura de los interruptores y también se observa el valor ingresado en los LEDs. En el inciso *b* de la figura 5.9 se muestran los interruptores, el display LCD y los LEDs en la tarjeta SPARTAN 3E. Los interruptores tienen una un valor binario de "1010" que en decimal es 10, mientras que para los LEDs el valor decimal ingresado es 55 que en binario es "110111". El display LCD muestra en la segunda línea el mensaje "I/O …" el cual indica que se ha seleccionado dicha opción.

En la opción 4 se obtuvieron los resultados esperados ya que se activaron los LEDs y se leyó el estado de los interruptores de manera correcta y sin mayor problema, por lo que el funcionamiento es el correcto.

5) MODO PRUEBA

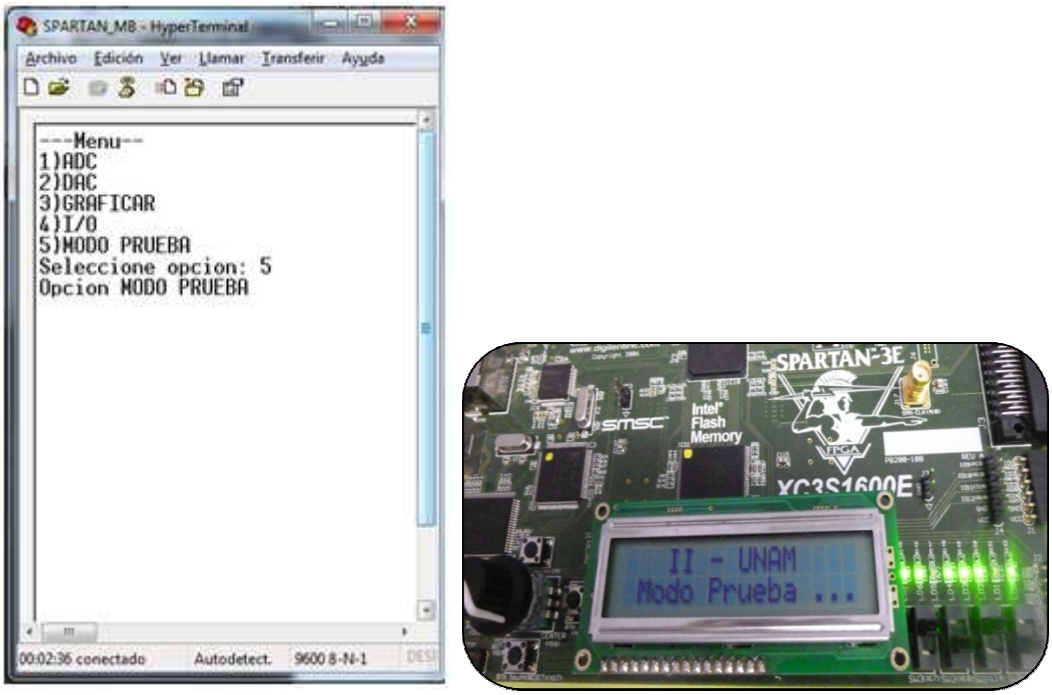
En esta opción se pretende corroborar el funcionamiento del sistema de monitoreo y control, principalmente de la comunicación SPI, por lo que los dispositivos que se utilizan son el ADC y los DACs. Lo que se realiza es conectar la opción 1 y la opción 2, es decir, adquirir datos mediante el ADC, almacenarlos en memoria DDR, leer los datos de la memoria, ajustarlos para los DACs y enviarlos a los cuatro DACs.

Cuando se selecciona la opción 5, en *Hyperterminal* se muestra el mensaje "MODO PRUEBA" y en el display LCD se muestra el mensaje "Modo Prueba …", así como se muestra en la figura 5.10 incisos *a* y *b* respectivamente.



a) *Hyperterminal* b) Interruptores, display LCD y LEDs

Figura 5.9. Mensajes de la opción 4.



a) *Hyperterminal* b) Display LCD

Figura 5.10. Mensajes de la opción 5.

En la figura 5.11 se muestra la salida de los DACs A y C. En ella se observan las señales ingresadas en las entradas, estas son una onda senoidal y una onda triangular. Las señales se envían continuas, es decir, primero se envía la senoidal y después la triangular.

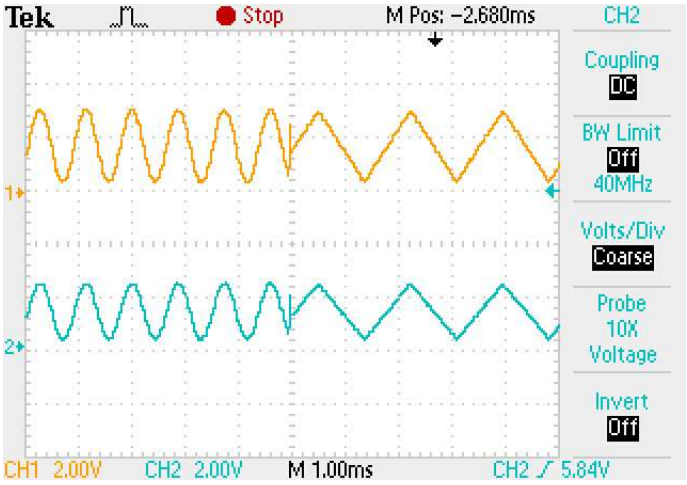


Figura 5.11. Salida del DAC en el modo prueba.

En la figura 5.12 se muestran las conexiones realizadas con la tarjeta SPARTAN 3E donde se realizaron las pruebas del sistema de monitoreo y control.

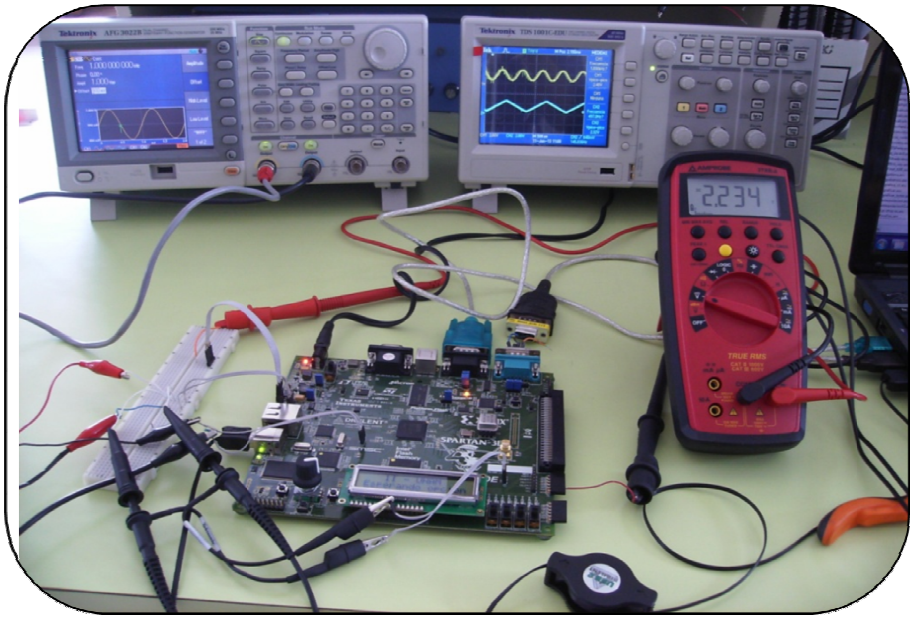


Figura 5.12. Conexiones con la tarjeta SPARTAN 3E.

Con estos resultados se corroboró el funcionamiento del sistema de monitoreo y control, basado en un sistema embebido dentro de un FPGA, con la tarjeta de desarrollo SPARTAN 3E.

Una vez validado el sistema de monitoreo y control se procedió a validar el hardware de los prototipos construidos, utilizando para ello la tarjeta de desarrollo SPARTAN 3E.

5.2. Evaluación de prototipos

Los circuitos impresos que se validaron fueron el del ADC, el del DAC y el de la comunicación asíncrona. También se pudo comprobar el funcionamiento del display LCD externo a la tarjeta de desarrollo SPARTAN 3E sin necesidad de circuito impreso. La memoria DDR y el propio FPGA, con todo lo necesario para su funcionamiento, se utilizaron de la tarjeta de desarrollo, al igual que los LEDs y los interruptores, por lo que no se realizaron pruebas a estos dispositivos.

Lo que se realizó para esta prueba fue cambiar las terminales que manejan el ADC, el DAC, el RS-232 y el dispyal LCD dentro de la tarjeta SPARTAN 3E, a terminales de propósito general para manejar los componentes que se construyeron. Para ello se utilizó una tarjeta de expansión en donde se encuentran disponibles dichas terminales. La conexión que se realizó para esta prueba se muestra en la figura 5.13, donde se observa la tarjeta de desarrollo, la tarjeta de expansión, los circuitos impresos y el display LCD.

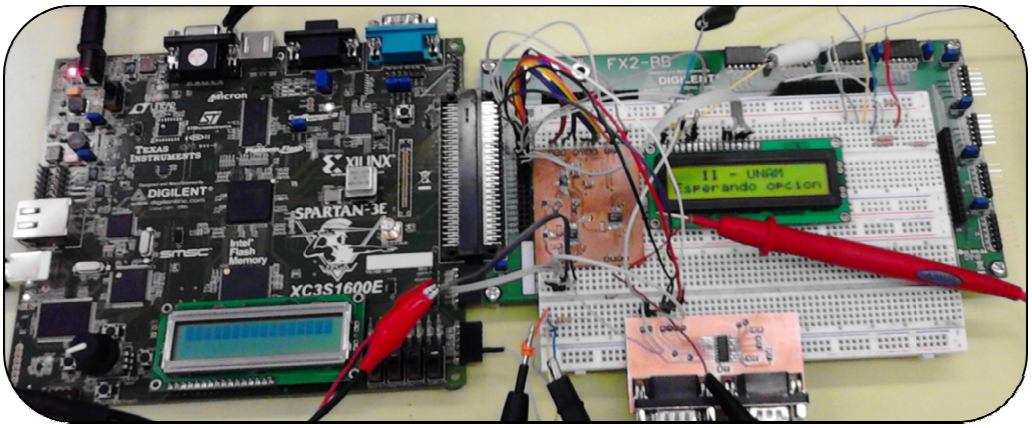


Figura 5.13. Conexión de los circuitos impresos construidos.

El primer circuito impreso que se validó fue el de la comunicación asíncrona, ya que sin ella no se puede acceder al menú presentado por el sistema de monitoreo y control. Al realizar las pruebas no se presentó ningún tipo de problema y funcionó adecuadamente.

El segundo componente que se comprobó su funcionamiento fue el display LCD, el cual desplegaba el mensaje correspondiente a cada opción solicitada. En la opción 1 el mensaje “ADC …”; en la opción 2 “DAC: A”; en la opción 3 “Graficando C: 1”; en la opción 4 “I/O …” y en la opción 5 “Modo Prueba …”.

Para corroborar el funcionamiento del ADC se realizaron mediciones de voltaje de corriente directa con diferentes ganancias, con la ayuda de la opción 1 del sistema de monitoreo y control. En la tabla 5.6 se muestra los valores obtenidos con una ganancia unitaria con el ADC construido, se observa el voltaje para ambas entradas, el voltaje de entrada y salida diferencial del amplificador, el valor teórico de la conversión y las salidas en formato decimal y en formato de voltaje del convertidor.

Entrada	Entrada diferencial	Salida con ganancia	Ec 3.7	Salida		Salida (en Volts)	
VA=VB	amplificador	amplificador	D[13:0]	Canal A	Canal B	Canal A	Canal B
0.1634	-1.4866	1.4866	9742.5818	8191	8191	0.4002	0.4002
0.2424	-1.4076	1.4076	9224.8474	8191	8191	0.4002	0.4002
0.308	-1.342	1.342	8794.9312	8191	8191	0.4002	0.4002
0.404	-1.246	1.246	8165.7856	8127	8079	0.4099	0.4172
0.503	-1.147	1.147	7516.9792	7492	7404	0.5068	0.5202
0.605	-1.045	1.045	6848.512	6811	6815	0.6107	0.6101
0.702	-0.948	0.948	6212.8128	6209	6214	0.7026	0.7018
0.804	-0.846	0.846	5544.3456	5574	5533	0.7995	0.8057
0.906	-0.744	0.744	4875.8784	4845	4800	0.9107	0.9176
1.003	-0.647	0.647	4240.1792	4201	4194	1.009	1.01
1.1	-0.55	0.55	3604.48	3594	3558	1.1016	1.1071
1.207	-0.443	0.443	2903.2448	2904	2887	1.2069	1.2095
1.301	-0.349	0.349	2287.2064	2251	2248	1.3065	1.307
1.403	-0.247	0.247	1618.7392	1558	1526	1.4123	1.4172
1.503	-0.147	0.147	963.3792	923	919	1.5092	1.5098
1.603	-0.047	0.047	308.0192	262	255	1.61	1.6111
1.709	0.059	-0.059	-386.6624	-427	-441	1.7152	1.7173
1.809	0.159	-0.159	-1042.0224	-1061	-1064	1.8119	1.8124
1.903	0.253	-0.253	-1658.0608	-1671	-1678	1.905	1.906
2.002	0.352	-0.352	-2306.8672	-2343	-2380	2.0075	2.0132
2.103	0.453	-0.453	-2968.7808	-3051	-3062	2.1155	2.1172
2.205	0.555	-0.555	-3637.248	-3746	-3781	2.2216	2.2269
2.307	0.657	-0.657	-4305.7152	-4368	-4371	2.3165	2.317
2.402	0.752	-0.752	-4928.3072	-4983	-4982	2.4103	2.4102
2.504	0.854	-0.854	-5596.7744	-5607	-5609	2.5056	2.5059
2.601	0.951	-0.951	-6232.4736	-6278	-6286	2.6079	2.6092
2.702	1.052	-1.052	-6894.3872	-7054	-7119	2.7264	2.7363
2.804	1.154	-1.154	-7562.8544	-7581	-7628	2.8068	2.8139
2.9	1.25	-1.25	-8192	-8192	-8192	2.9	2.9
3.011	1.361	-1.361	-8919.4496	-8192	-8192	2.9	2.9
3.116	1.466	-1.466	-9607.5776	-8192	-8192	2.9	2.9
3.201	1.551	-1.551	-10164.634	-8192	-8192	2.9	2.9
3.304	1.654	-1.654	-10839.654	-8192	-8192	2.9	2.9

Tabla 5.6. Conversión con ganancia de -1 del ADC construido.

En la figura 5.14 se muestra una gráfica de los voltajes de conversión con una ganancia unitaria. En rojo se muestra la conversión de los canales A y B y en azul se muestra el valor de conversión teórico. La desviación entre los voltajes teóricos y prácticos no es mayor al 1%.

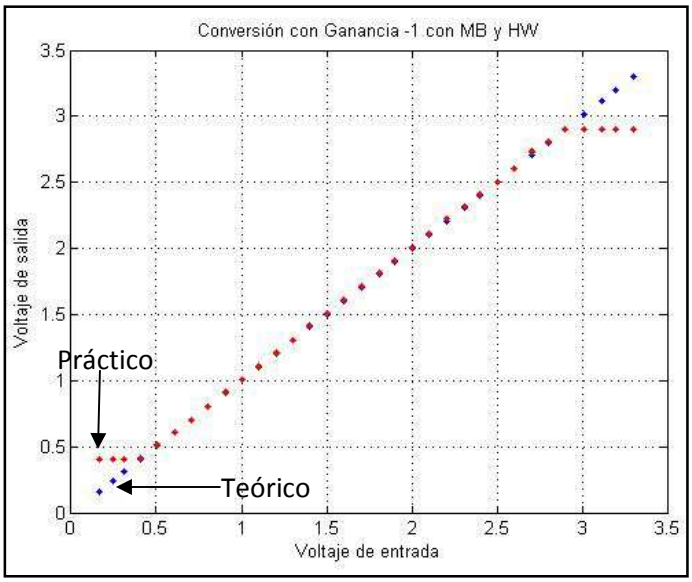


Figura 5.14. Gráfica para una ganancia de -1 del ADC construido.

En la tabla 5.7 se muestra la conversión ahora con una ganancia de -2.

Entrada	**Entrada diferencial**	**Salida con ganancia**	**Ec 3.7**	**Salida**		**Salida (en Volts)**	
VA=VB	amplificador	amplificador	D[13:0]	Canal A	Canal B	Canal A	Canal B
0.907	-0.743	1.486	9738.6496	8191	8191	1.0251	1.0251
0.95	-0.7	1.4	9175.04	8191	8191	1.0251	1.0251
1.001	-0.649	1.298	8506.5728	8191	8191	1.0251	1.0251
1.051	-0.599	1.198	7851.2128	7715	7678	1.0614	1.0642
1.1	-0.55	1.1	7208.96	7101	7101	1.1082	1.1082
1.155	-0.495	0.99	6488.064	6185	6214	1.1781	1.1759
1.202	-0.448	0.896	5872.0256	5800	5835	1.2075	1.2048
1.251	-0.399	0.798	5229.7728	5136	5124	1.2582	1.2591
1.303	-0.347	0.694	4548.1984	4491	4446	1.3074	1.3108
1.353	-0.297	0.594	3892.8384	3831	3832	1.3577	1.3576
1.403	-0.247	0.494	3237.4784	3158	3124	1.4091	1.4117
1.45	-0.2	0.4	2621.44	2598	2596	1.4518	1.4519
1.506	-0.144	0.288	1887.4368	1842	1838	1.5095	1.5098
1.55	-0.1	0.2	1310.72	1264	1208	1.5536	1.5578
1.6	-0.05	0.1	655.36	430	373	1.6172	1.6215
1.651	0.001	-0.002	-13.1072	-99	-111	1.6576	1.6585
1.706	0.056	-0.112	-734.0032	-1064	-1133	1.7312	1.7364
1.751	0.101	-0.202	-1323.8272	-1402	-1441	1.757	1.7599
1.806	0.156	-0.312	-2044.7232	-2115	-2128	1.8114	1.8124
1.854	0.204	-0.408	-2673.8688	-2727	-2741	1.8581	1.8591
1.904	0.254	-0.508	-3329.2288	-3381	-3383	1.9079	1.9081

Tabla 5.7. Conversión con ganancia de -2 del ADC construido. (Continúa)

Entrada	Entrada diferencial	Salida con ganancia	Ec 3.7	Salida		Salida (en Volts)	
VA=VB	amplificador	amplificador	D[13:0]	Canal A	Canal B	Canal A	Canal B
1.95	0.3	-0.6	-3932.16	-4008	-4056	1.9558	1.9594
2.001	0.351	-0.702	-4600.6272	-4681	-4698	2.0071	2.0084
2.057	0.407	-0.814	-5334.6304	-5367	-5391	2.0595	2.0613
2.105	0.455	-0.91	-5963.776	-6020	-6029	2.1093	2.11
2.152	0.502	-1.004	-6579.8144	-6646	-6662	2.157	2.1583
2.201	0.551	-1.102	-7222.0672	-7262	-7279	2.204	2.2053
2.252	0.602	-1.204	-7890.5344	-7890	-7915	2.252	2.2539
2.273	0.623	-1.246	-8165.7856	-8186	-8192	2.2745	2.275
2.303	0.653	-1.306	-8559.0016	-8192	-8192	2.275	2.275
2.351	0.701	-1.402	-9188.1472	-8192	-8192	2.275	2.275
2.402	0.752	-1.504	-9856.6144	-8192	-8192	2.275	2.275

Tabla 5.7. Conversión con ganancia de -2 del ADC construido.

En la figura 5.15 se muestra la gráfica para la conversión con ganancia de -2. En azul se muestra la conversión teórica y en rojo se muestra la conversión de ambos canales del A y del B. El porcentaje de error de conversión es menor al 2%.

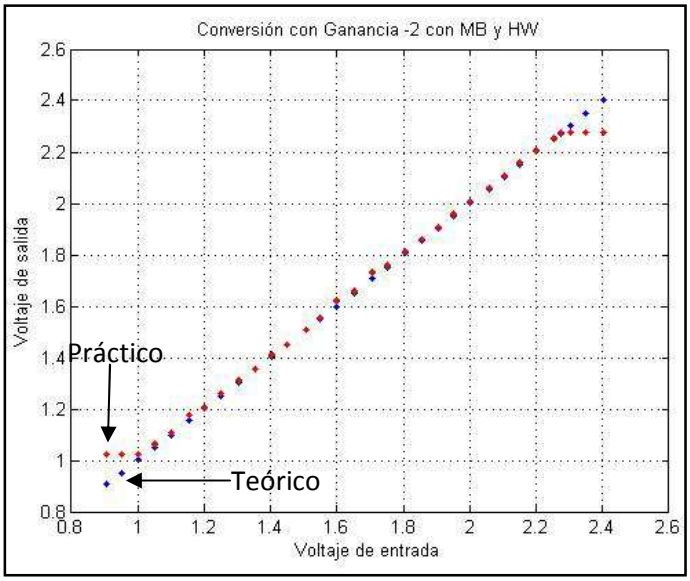


Figura 5.15. Gráfica para una ganancia de -2 del ADC construido.

En la tabla 5.8 se muestra la conversión ahora con una ganancia de -50. En la figura 5.16 se muestra la gráfica para la conversión con ganancia de -50. En azul se muestra la conversión teórica y en rojo se muestra la conversión de ambos canales del A y del B. El porcentaje de error de conversión es menor al 1%.

Entrada	Entrada diferencial	Salida con ganancia	Ec 3.7	Salida		Salida (en Volts)	
VA=VB	amplificador	amplificador	D[13:0]	Canal A	Canal B	Canal A	Canal B
1.611	-0.039	1.95	12779.52	8191	8191	1.625	1.625
1.628	-0.022	1.1	7208.96	7267	7219	1.6278	1.628
1.633	-0.017	0.85	5570.56	6415	6343	1.6304	1.6306
1.636	-0.014	0.7	4587.52	3959	3984	1.6379	1.6378
1.641	-0.009	0.45	2949.12	1967	1941	1.644	1.6441
1.646	-0.004	0.2	1310.72	427	212	1.6487	1.6494
1.647	-0.003	0.15	983.04	381	372	1.6488	1.6489
1.652	0.002	-0.1	-655.36	-674	-593	1.6521	1.6518
1.658	0.008	-0.4	-2621.44	-2954	-3044	1.659	1.6593
1.665	0.015	-0.75	-4915.2	-5789	-5983	1.6677	1.6683
1.671	0.021	-1.05	-6881.28	-7635	-7840	1.6733	1.6739
1.676	0.026	-1.3	-8519.68	-8192	-8192	1.675	1.675
1.681	0.031	-1.55	-10158.1	-8192	-8192	1.675	1.675

Tabla 5.8. Conversión con ganancia de -50 del ADC construido.

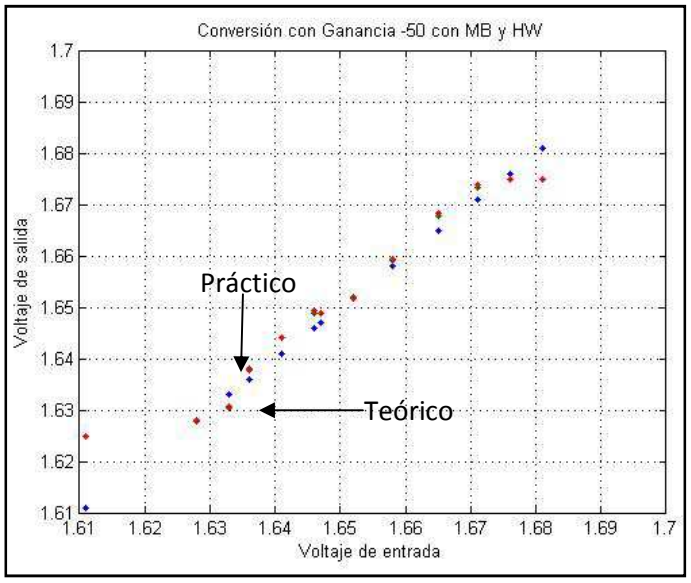


Figura 5.16. Gráfica para una ganancia de -50 del ADC construido.

Para corroborar el funcionamiento del DAC se realizaron mediciones de voltaje en los cuatro canales con la ayuda de la opción 2 del sistema de monitoreo y control. Una característica importante en el circuito impreso construido para el DAC es que los voltajes de referencia para los cuatro canales es el mismo, el cual tiene un valor de 3.3V, esto por la facilidad que se tenía al conectar únicamente un voltaje al circuito impreso. En la tabla 5.9 se muestran los voltajes que se ingresaron mediante *Hyperterminal*, también se muestra el voltaje medido en cada canal y el error

que se obtuvo en cada medición de cada canal. Como se observa el porcentaje de error es menor al 1%.

Voltaje de entrada	**Voltaje Medido**				**%Error**			
Vref=3.294	A	B	C	D	A	B	C	D
0.4	0.397	0.4	0.398	0.397	0.75%	0%	0.5%	0.75%
0.8	0.795	0.798	0.797	0.795	0.63%	0.25%	0.38%	0.63%
1.2	1.195	1.198	1.196	1.194	0.42%	0.17%	0.33%	0.5%
1.6	1.593	1.597	1.595	1.593	0.44%	0.19%	0.31%	0.44%
2	1.993	1.996	1.995	1.992	0.35%	0.2%	0.25%	0.4%
2.4	2.391	2.395	2.394	2.39	0.38%	0.21%	0.25%	0.42%
2.8	2.79	2.794	2.794	2.789	0.36%	0.21%	0.21%	0.39%
3.2	3.189	3.192	3.192	3.186	0.34%	0.25%	0.25%	0.44%

Tabla 5.9. Voltajes de salida del DAC construido.

Como parte de la opción 3, ofrecida en el sistema de monitoreo y control, se realizó la captura de una señal senoidal y una señal triangular con el circuito impreso realizado. Una vez adquiridos los datos se realizó su graficación utilizando MATLAB. En la figura 5.17 se muestra la señal senoidal del canal 1, mientras que en la figura 5.18 se muestra la onda triangular del canal 2.

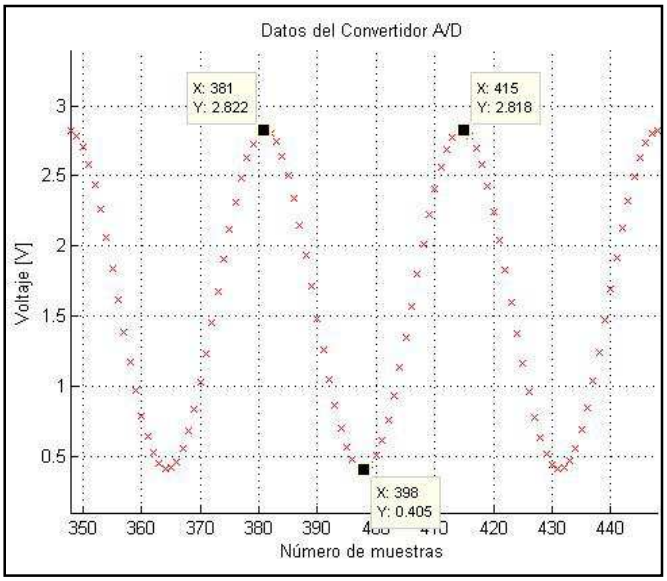


Figura 5.17. Señal senoidal en MATLAB del ADC construido.

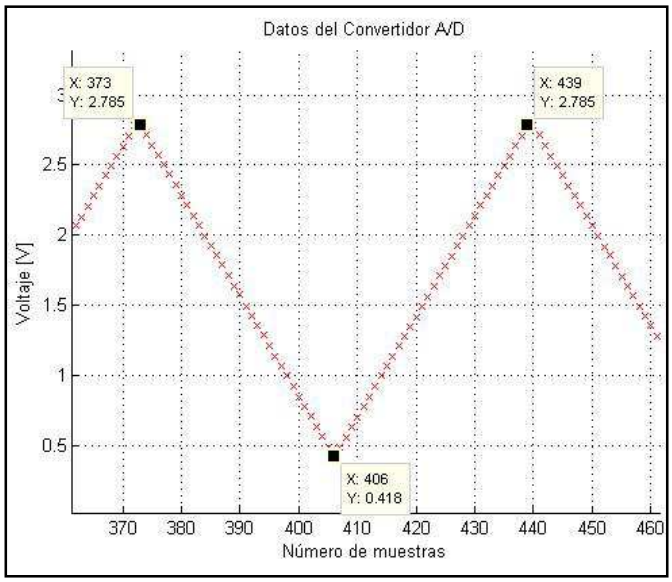


Figura 5.18. Señal triangular en MATLAB del ADC construido.

En la tabla 5.10 se muestran los parámetros de las señales de entrada y de salida que se utilizaron para comprobar el funcionamiento del ADC en la opción 3, GRAFICAR. Los resultados son muy similares a los obtenidos con el ADC de la tarjeta SPARTAN 3E.

Parámetro /Valor	**Entrada**		**Salida**		**%Error**	
	Senoidal	Triangular	Senoidal	Triangular	Senodal	Triangular
Vmax	2.88V	2.88V	2.822V	2.785V	2.01%	3.3%
Vmin	0.4V	0.4V	0.405V	0.418V	1.25%	4.5%
Vpp	2.48V	2.48V	2.417V	2.367V	2.54%	4.56%
Frecuencia	1 kHz	500 Hz	986.972Hz	508.44Hz	1.30%	1.69%

Tabla 5.10. Parámetros de las señales en MATLAB.

Por último se comprobó el funcionamiento de la opción 5, en donde se conecta el ADC y el DAC, obteniendo las señales mostradas en la figura 5.19. Se observa que la amplitud de las señales es la misma debido a que el voltaje de referencia es el mismo para los cuatro canales.

Con estas pruebas se concluye la validación de los circuitos impresos construidos. Los diseños validados se utilizaron en la tarjeta PRIUS asegurando así su funcionamiento. Los diseños para el FPGA, los reguladores y la memoria DDR se validaron propiamente en la tarjeta PRIUS.

5.3. Tarjeta PRIUS

La primera prueba realizada a la tarjeta PRIUS fue a los reguladores de voltaje, ya que son de suma importancia para el funcionamiento de todos los componentes del sistema de monitoreo y control. El circuito integrado TPS75003, figura 5.20 inciso *a*, funcionó correctamente, proporcionando los siguientes valores de voltaje: 1.225V, 2.537V y 3.33V. El circuito integrado

LTC3412, figura 5.20, inciso *b*, funcionó correctamente, proporcionando un voltaje de 2.55V y el divisor de voltaje proporcionando un voltaje de 1.277V.

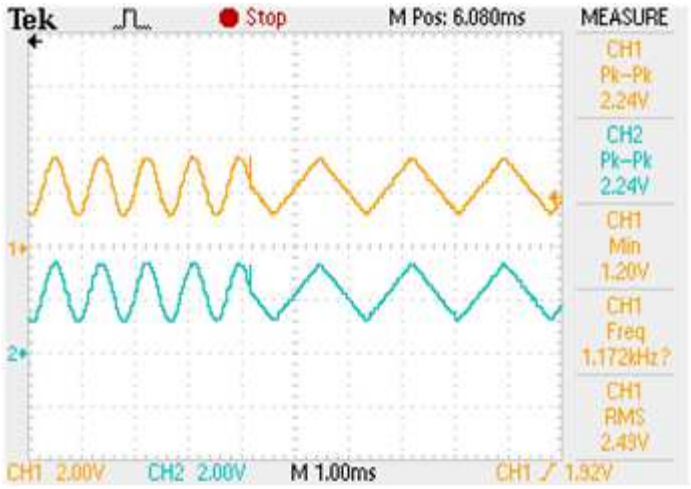
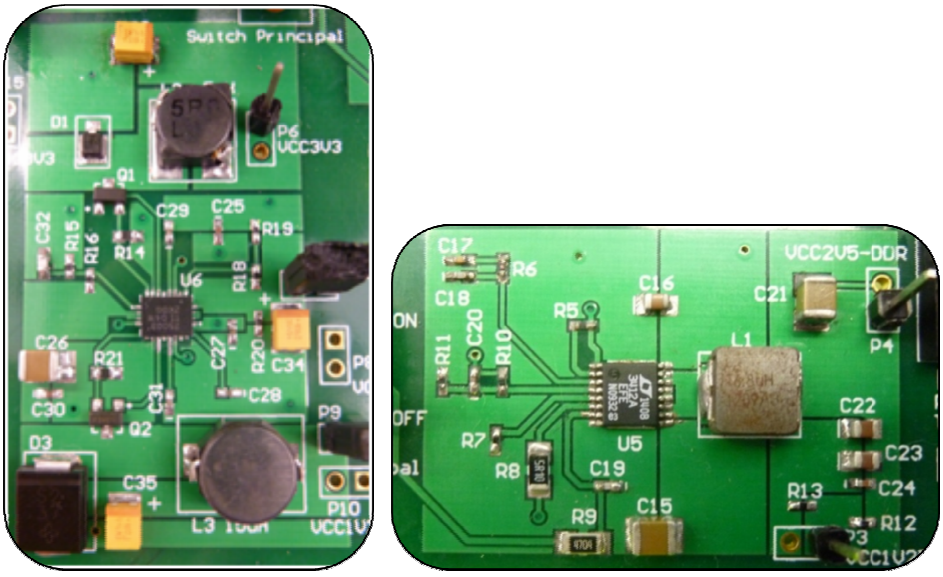


Figura 5.19. Salida del DAC construido en modo prueba.



a) TPS75003 b) LTC3412

Figura 5.20. Reguladores de voltaje en la tarjeta PRIUS.

Ya que funcionaron correctamente los reguladores, se procedió a corroborar el funcionamiento del FPGA. Para ello se soldaron los componentes necesarios para poderlo configurar, los componentes fueron: resistencias, capacitores, LEDs, *push button*, conectores e interruptores. Lo primero que se comprobó fue que el software de Xilinx detectara el FPGA, para ello se utilizó la tarjeta de desarrollo SPARTAN 3E. La conexión real para configurar la tarjeta PRIUS con la tarjeta SPARTAN 3E se muestra en la figura 5.21, en ella se observan cuatro aspectos: la conexión JTAG entre ambas tarjetas, la unión de las señales de tierra de las tarjetas, la desactivación de la tarjeta SPARTAN 3E y el módulo USB de la tarjeta SPARTAN 3E. Lo que se realiza con la conexión entre las

tarjetas es desconectar los componentes JTAG de la tarjeta SPARTAN 3E y conectar el único componente JTAG de la tarjeta PRIUS. Una vez realizada la conexión entre las tarjetas, se utilizó el software llamado iMPACT (que es parte de ISE Design Suite 13.2) para detectar los componentes JTAG. La detección se realizó de manera correcta ya que iMPACT mostró el FPGA de la tarjeta PRIUS. En la figura 5.22 se muestra la pantalla de iMPACT en donde se muestra el FPGA de la tarjeta PRIUS.

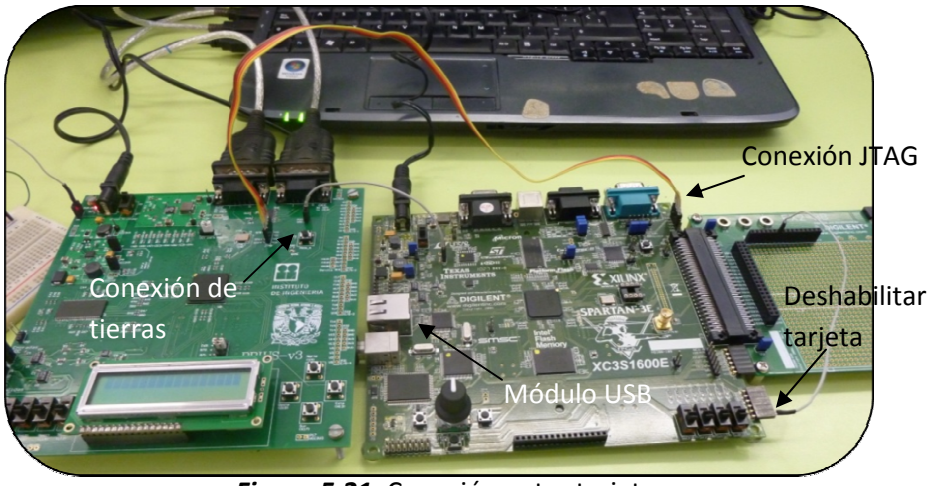


Figura 5.21. Conexión entre tarjetas.

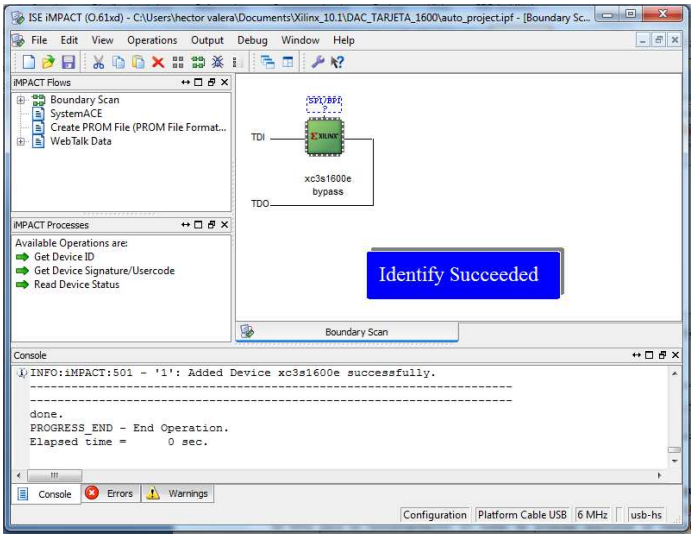


Figura 5.22. FPGA de la tarjeta PRIUS en iMPACT.

Ya que se logró detectar el FPGA, la segunda prueba que se realizó fue configurar el FPGA mediante iMPACT, ya que con éste se observan a detalle todos los dispositivos conectados en la cadena JTAG. Se configuró el FPGA descargando el *bitstream* de algunos diseños realizados en VHDL, los cuales utilizaban los interruptores, los LEDs y el cristal de 50 MHz para su funcionamiento. En todas las pruebas realizadas se obtuvieron resultados satisfactorios porque se configuró de manera correcta el FPGA.

La tercera y última prueba que se realizó fue configurar la tarjeta PRIUS mediante SDK, en donde la configuración del FPGA se realiza de manera transparente, sin mostrar detalles de los dispositivos JTAG, y además, es en donde se descarga el software realizado para el procesador embebido. En SDK se selecciona la opción *Program FPGA* del menú *Xilinx Tools*, figura 5.23, y se configura el FPGA. Como la configuración del FPGA con SDK fue correcta, se procedió a corroborar el funcionamiento del sistema de monitoreo y control diseñado.

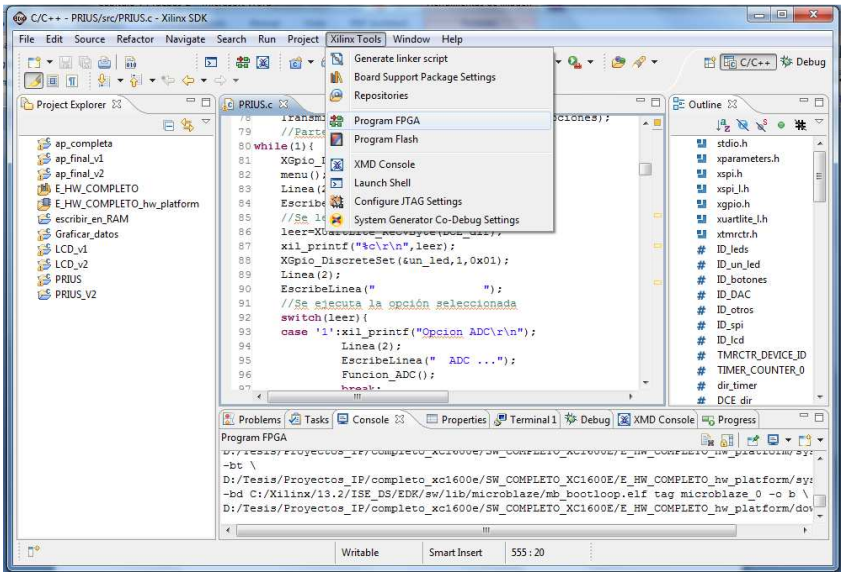


Figura 5.23. Configuración del FPGA con SDK.

La comunicación UART fue lo primero que se corroboró, ya que sin ella el menú del sistema de monitoreo y control no se desplegaría. La UART Funcionó correctamente y desplegó el menú de la figura 5.24.

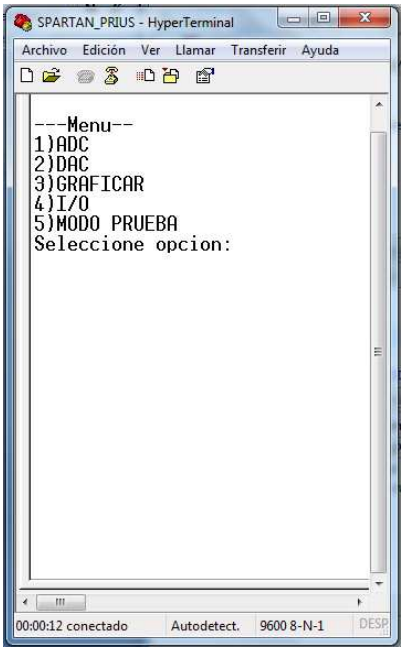
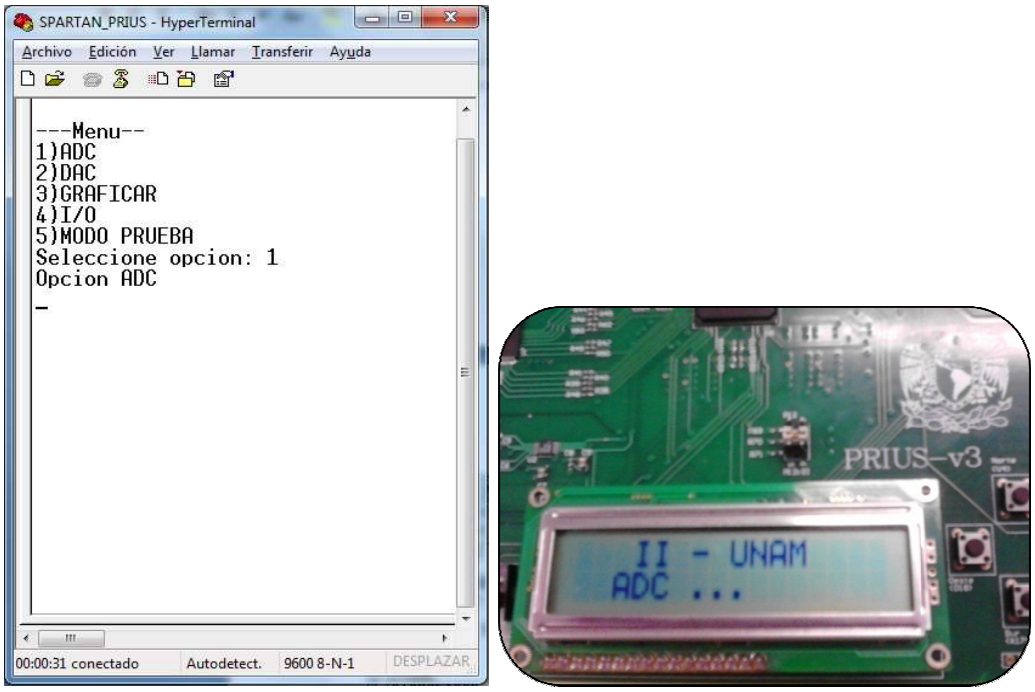


Figura 5.24. Menú en *Hyperterminal*.

1) ADC

En esta opción se activa el ADC, se realizan conversiones y los datos adquiridos se almacenan en la memoria DDR, por lo que los componentes que se validaron fueron el amplificador, el ADC y la memoria DDR. En la figura 5.25, inciso *a*, se muestra el mensaje "Opcion ADC" en *Hyperterminal*, mientras que en el inciso *b* se muestra el mensaje "ADC …" en el display LCD, los cuales son los mismos que se obtuvieron con la tarjeta de desarrollo SPARTAN 3E verificando el correcto funcionamiento de los componentes.



a) *Hyperterminal* b) Display LCD

Figura 5.25. Mensajes de la opción 1 en la tarjeta PRIUS.

En la figura 5.26 se muestra la memoria DDR y todos los componentes para su funcionamiento en la tarjeta PRIUS. Con los resultados obtenidos se validó el funcionamiento de la memoria DDR.

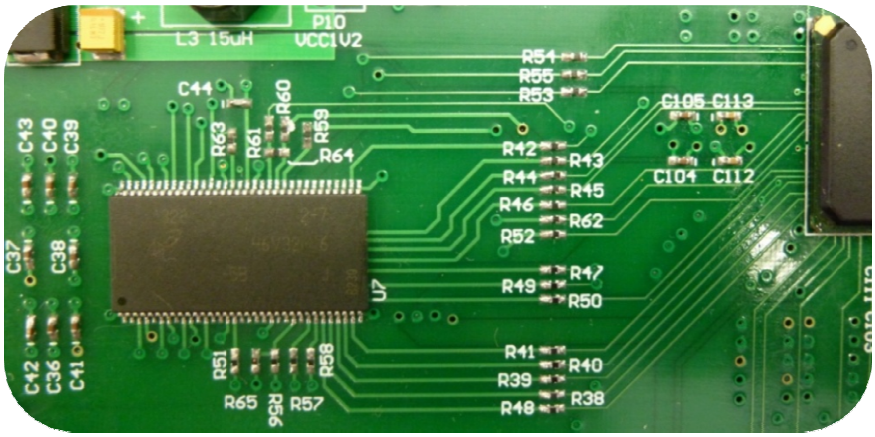


Figura 5.26. Memoria DDR en la tarjeta PRIUS.

Al realizar algunas mediciones de voltaje en la tarjeta PRIUS, se detectó una diferencia en el voltaje de referencia del amplificador con respecto a la tarjeta SPARTAN 3E. El valor de referencia en la tarjeta PRIUS es de 1.665V en lugar de 1.648V de la SPARTAN 3E, por lo que la ecuación de conversión se ve afectada ligeramente a:

$$D[13:0] = GANANCIA * \frac{(V_{ent} - 1.665V)}{1.25V} * 8192 \qquad (5.1)$$

La variación en el voltaje de referencia es debido a que el voltaje de alimentación es de 3.33V en la tarjeta PRIUS y el voltaje utilizado en el amplificador es la mitad de este voltaje, es decir, 1.665V.

Con esta modificación en el voltaje de referencia, los voltajes de entrada para cada ganancia se ven afectados, así como lo muestra la tabla 5.11.

Ganancia	SPARTAN 3E		PRIUS	
	Mínimo	Máximo	Mínimo	Máximo
0	**-**	-	-	-
-1	0.4	2.9	0.415	2.915
-2	1.025	2.275	1.04	2.29
-5	1.4	1.9	1.415	1.915
-10	1.525	1.775	1.54	1.79
-20	1.5875	1.7125	1.6025	1.7275
-50	1.625	1.675	1.64	1.69
-100	1.6375	1.6625	1.6525	1.6775

Tabla 5.11. Valores nuevos de entrada.

En la tabla 5.12 se observa el voltaje ingresado a cada canal, la entrada y salida diferencial del amplificador, el valor teórico de la conversión y las salidas de ambos convertidores en formato decimal y en formato de voltaje. Estos valores se obtuvieron con una ganancia unitaria y con la ecuación 5.1.

Entrada	Entrada diferencial	Salida con ganancia	Ec 3.7	Salida		Salida (en Volts)	
VA=VB	amplificador	amplificador	D[13:0]	Canal A	Canal B	Canal A	Canal B
0.183	-1.481	1.481	9614.1312	8191	8191	0.4152	0.4152
0.2424	-1.4216	1.4216	9224.8474	8191	8191	0.4152	0.4152
0.308	-1.356	1.356	8794.9312	8191	8191	0.4152	0.4152
0.404	-1.26	1.26	8165.7856	8191	8191	0.4152	0.4152
0.503	-1.161	1.161	7516.9792	7622	7622	0.502	0.502
0.602	-1.062	1.062	6868.1728	6955	6953	0.6038	0.6041
0.701	-0.963	0.963	6219.3664	6328	6325	0.6994	0.6999
0.8	-0.864	0.864	5570.56	5672	5673	0.7995	0.7994

Tabla 5.12. Conversión con ganancia de -1 tarjeta PRIUS. (Continúa)

Entrada	**Entrada diferencial**	**Salida con ganancia**	**Ec 3.7**	**Salida**		**Salida (en Volts)**	
VA=VB	amplificador	amplificador	D[13:0]	Canal A	Canal B	Canal A	Canal B
0.903	-0.761	0.761	4895.5392	4991	5017	0.9034	0.8995
1.001	-0.663	0.663	4253.2864	4362	4358	0.9994	1
1.105	-0.559	0.559	3571.712	3687	3679	1.1024	1.1036
1.203	-0.461	0.461	2929.4592	3044	3043	1.2005	1.2007
1.3	-0.364	0.364	2293.76	2422	2443	1.2954	1.2922
1.4	-0.264	0.264	1638.4	1724	1714	1.4019	1.4035
1.504	-0.16	0.16	956.8256	1058	1081	1.5036	1.5001
1.605	-0.059	0.059	294.912	412	433	1.6021	1.5989
1.703	0.039	-0.039	-347.3408	-245	-252	1.7024	1.7035
1.8	0.136	-0.136	-983.04	-885	-859	1.8	1.7961
1.905	0.241	-0.241	-1671.168	-1598	-1602	1.9088	1.9094
2.003	0.339	-0.339	-2313.4208	-2225	-2202	2.0045	2.001
2.1	0.436	-0.436	-2949.12	-2850	-2827	2.0999	2.0964
2.203	0.539	-0.539	-3624.1408	-3496	-3508	2.1984	2.2003
2.3	0.636	-0.636	-4259.84	-4140	-4148	2.2967	2.2979
2.405	0.741	-0.741	-4947.968	-4824	-4833	2.4011	2.4025
2.501	0.837	-0.837	-5577.1136	-5475	-5480	2.5004	2.5012
2.604	0.94	-0.94	-6252.1344	-6137	-6112	2.6014	2.5976
2.705	1.041	-1.041	-6914.048	-6790	-6772	2.7011	2.6983
2.804	1.14	-1.14	-7562.8544	-7465	-7469	2.8041	2.8047
2.905	1.241	-1.241	-8224.768	-8124	-8106	2.9046	2.9019
3.006	1.342	-1.342	-8886.6816	-8192	-8192	2.915	2.915
3.116	1.452	-1.452	-9607.5776	-8192	-8192	2.915	2.915
3.201	1.537	-1.537	-10164.6336	-8192	-8192	2.915	2.915
3.304	1.64	-1.64	-10839.6544	-8192	-8192	2.915	2.915

Tabla 5.12. Conversión con ganancia de -1 tarjeta PRIUS.

En la figura 5.27 se muestra una gráfica de los voltajes de conversión con una ganancia unitaria. En rojo se muestra la conversión de los canales A y B y en azul se muestra el valor de conversión teórico. La desviación entre los voltajes teóricos y prácticos no es mayor al 1%.

En la tabla 5.13 se muestra la conversión ahora con una ganancia de -2. En la figura 5.28 se muestra una gráfica de los voltajes de conversión con una ganancia de -2. En rojo se muestra la conversión de los canales A y B y en azul se muestra el valor de conversión teórico. La desviación entre los voltajes teóricos y prácticos no es mayor al 1%.

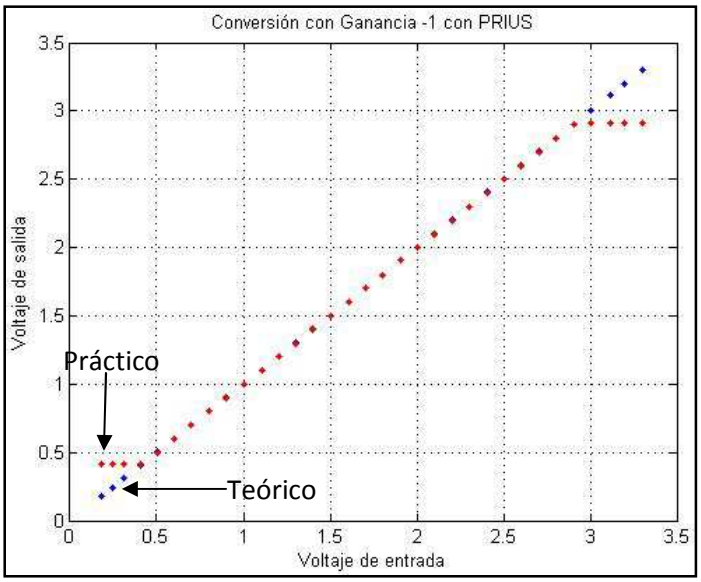


Figura 5.27. Gráfica para una ganancia de -1 en la tarjeta PRIUS.

Entrada	**Entrada diferencial**	**Salida con ganancia**	**Ec 3.7**	**Salida**		**Salida (en Volts)**	
VA=VB	amplificador	amplificador	D[13:0]	Canal A	Canal B	Canal A	Canal B
0.908	-0.756	1.512	9909.0432	8191	8191	1.0391	1.0391
0.958	-0.706	1.412	9253.6832	8191	8191	1.0391	1.0391
1.001	-0.663	1.326	8690.0736	8191	8191	1.0391	1.0391
1.052	-0.612	1.224	8021.6064	8041	8063	1.0505	1.0488
1.102	-0.562	1.124	7366.2464	7394	7417	1.0999	1.0981
1.152	-0.512	1.024	6710.8864	6713	6708	1.1518	1.1522
1.203	-0.461	0.922	6042.4192	6041	6065	1.2031	1.2013
1.253	-0.411	0.822	5387.0592	5400	5391	1.252	1.2527
1.302	-0.362	0.724	4744.8064	4736	4746	1.3027	1.3019
1.353	-0.311	0.622	4076.3392	4115	4146	1.3501	1.3477
1.405	-0.259	0.518	3394.7648	3438	3462	1.4017	1.3999
1.45	-0.214	0.428	2804.9408	2828	2849	1.4482	1.4466
1.503	-0.161	0.322	2110.2592	2106	2127	1.5033	1.5017
1.552	-0.112	0.224	1468.0064	1467	1483	1.5521	1.5509
1.604	-0.06	0.12	786.432	794	802	1.6034	1.6028
1.652	-0.012	0.024	157.2864	161	184	1.6517	1.65
1.703	0.039	-0.078	-511.1808	-534	-517	1.7047	1.7034
1.75	0.086	-0.172	-1127.219	-1120	-1130	1.7494	1.7502
1.802	0.138	-0.276	-1808.794	-1821	-1792	1.8029	1.8007
1.851	0.187	-0.374	-2451.046	-2411	-2416	1.8479	1.8483
1.901	0.237	-0.474	-3106.406	-3066	-3045	1.8979	1.8963
1.95	0.286	-0.572	-3748.659	-3740	-3751	1.9493	1.9502
2.007	0.343	-0.686	-4495.77	-4501	-4480	2.0074	2.0058

Tabla 5.13. Conversión con ganancia de -2 tarjeta PRIUS. (Continúa)

Entrada	Entrada diferencial	Salida con ganancia	Ec 3.7	Salida		Salida (en Volts)	
VA=VB	amplificador	amplificador	D[13:0]	Canal A	Canal B	Canal A	Canal B
2.052	0.388	-0.776	-5085.594	-5025	-5007	2.0474	2.046
2.105	0.441	-0.882	-5780.275	-5745	-5756	2.1023	2.1031
2.152	0.488	-0.976	-6396.314	-6386	-6364	2.1512	2.1495
2.201	0.537	-1.074	-7038.566	-7034	-7011	2.2007	2.1989
2.252	0.588	-1.176	-7707.034	-7709	-7693	2.2522	2.2509
2.275	0.611	-1.222	-8008.499	-7938	-8008	2.2696	2.275
2.3	0.636	-1.272	-8336.179	-8192	-8192	2.289	2.289
2.351	0.687	-1.374	-9004.646	-8192	-8192	2.289	2.289
2.402	0.738	-1.476	-9673.114	-8192	-8192	2.289	2.289

Tabla 5.13. Conversión con ganancia de -2 tarjeta PRIUS.

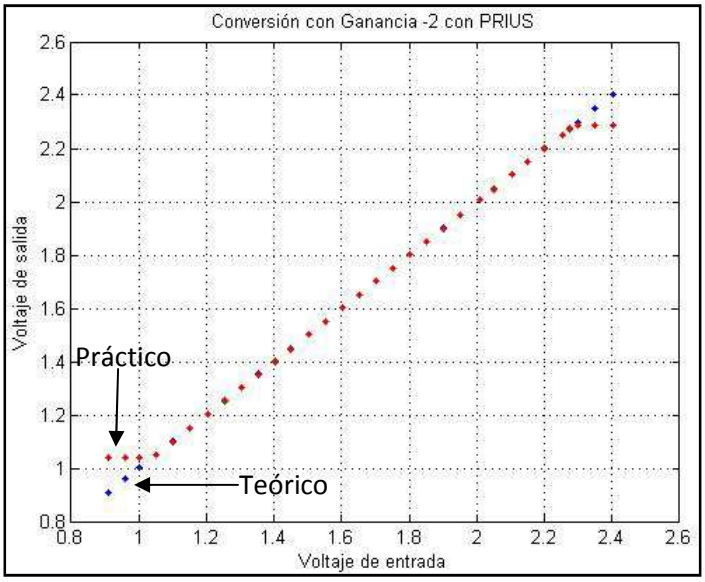


Figura 5.28. Gráfica para una ganancia de -2 en la tarjeta PRIUS.

Por último, en la tabla 5.14 se muestra la conversión ahora con una ganancia de -50. En la figura 5.29 se muestra una gráfica de los voltajes de conversión con una ganancia de -50. En rojo se muestra la conversión de los canales A y B y en azul se muestra el valor de conversión teórico. La desviación entre los voltajes teóricos y prácticos no es mayor al 1%.

Entrada	Entrada diferencial	Salida con ganancia	Ec 3.7	Salida		Salida (en Volts)	
VA=VB	amplificador	amplificador	D[13:0]	Canal A	Canal B	Canal A	Canal B
1.611	-0.053	2.65	17367.04	8191	8191	1.639	1.639
1.627	-0.037	1.85	12124.16	8191	8191	1.639	1.639
1.632	-0.032	1.6	10485.76	8191	8191	1.639	1.639

Tabla 5.14. Conversión con ganancia de -50 tarjeta PRIUS. (Continúa)

Entrada	Entrada diferencial	Salida con ganancia	Ec 3.7	Salida		Salida (en Volts)	
VA=VB	amplificador	amplificador	D[13:0]	Canal A	Canal B	Canal A	Canal B
1.635	-0.029	1.45	9502.72	8191	8191	1.639	1.639
1.64	-0.024	1.2	7864.32	8048	7927	1.6394	1.6398
1.645	-0.019	0.95	6225.92	6190	6074	1.6451	1.6455
1.647	-0.017	0.85	5570.56	6344	6223	1.6446	1.645
1.652	-0.012	0.6	3932.16	4375	4228	1.6506	1.6511
1.658	-0.006	0.3	1966.08	1297	1181	1.66	1.6604
1.665	0.001	-0.05	-327.68	-284	-429	1.6649	1.6653
1.672	0.008	-0.4	-2621.44	-1861	-2002	1.6697	1.6701
1.677	0.013	-0.65	-4259.84	-3276	-3428	1.674	1.6745
1.681	0.017	-0.85	-5570.56	-4737	-4864	1.6785	1.6788
1.685	0.021	-1.05	-6881.28	-6370	-6521	1.6834	1.6839
1.69	0.026	-1.3	-8519.68	-8166	-8192	1.6889	1.689
1.696	0.032	-1.6	-10485.8	-8192	-8192	1.689	1.689

Tabla 5.14. Conversión con ganancia de -50 tarjeta PRIUS.

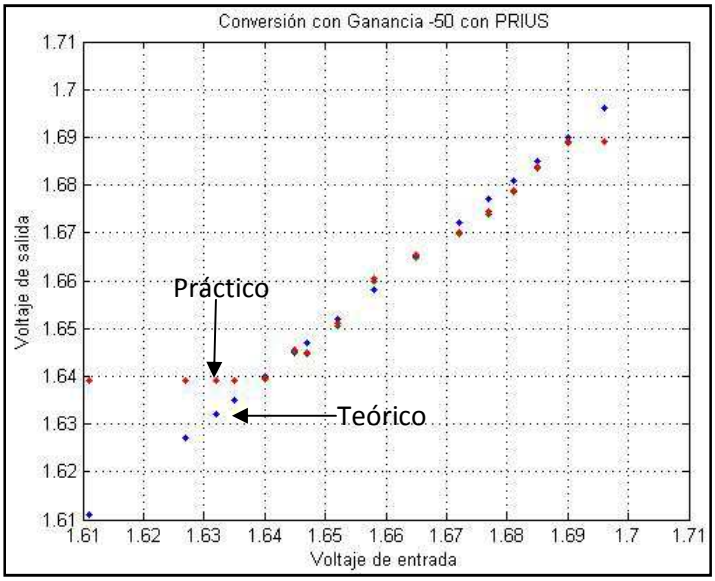
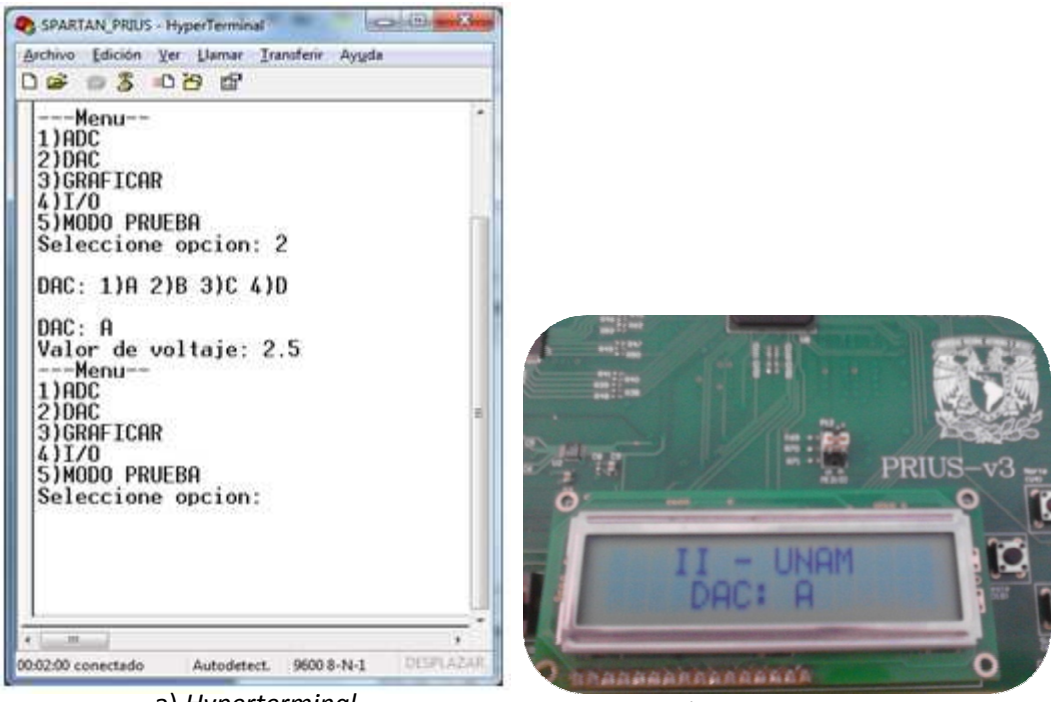


Figura 5.29. Gráfica para una ganancia de -50 en la tarjeta PRIUS.

Con estos resultados se corrobora el correcto funcionamiento de la tarjeta PRIUS con la opción 1 del sistema de monitoreo y control.

2) DAC

Con esta opción se activa el DAC, el mensaje en *Hyperterminal* se muestra en el inciso *a* de la figura 5.30 y el mensaje en el display LCD se muestra en el inciso *b* de la misma figura.



a) *Hyperterminal* b) Display LCD

Figura 5.30. Mensajes de la opción 2 en la tarjeta PRIUS.

Una característica importante en la tarjeta PRIUS es que el voltaje de referencia de los cuatro canales es de 3.3V, por lo que la ecuación de salida para cualquier canal es:

$$V_{salidaABCD} = \frac{D}{4096} * 3.3V \tag{5.2}$$

Conociendo la relación que existe entre la entrada y la salida se realizaron mediciones a los cuatro canales obteniendo la tabla 5.15. En dicha tabla se muestran los voltajes que se ingresaron mediante *Hyperterminal*, el voltaje medido en cada canal del DAC y el error que se obtuvo en cada medición de cada canal. Como se observa en dicha tabla, el porcentaje de error es menor al 1.25% en todos los casos.

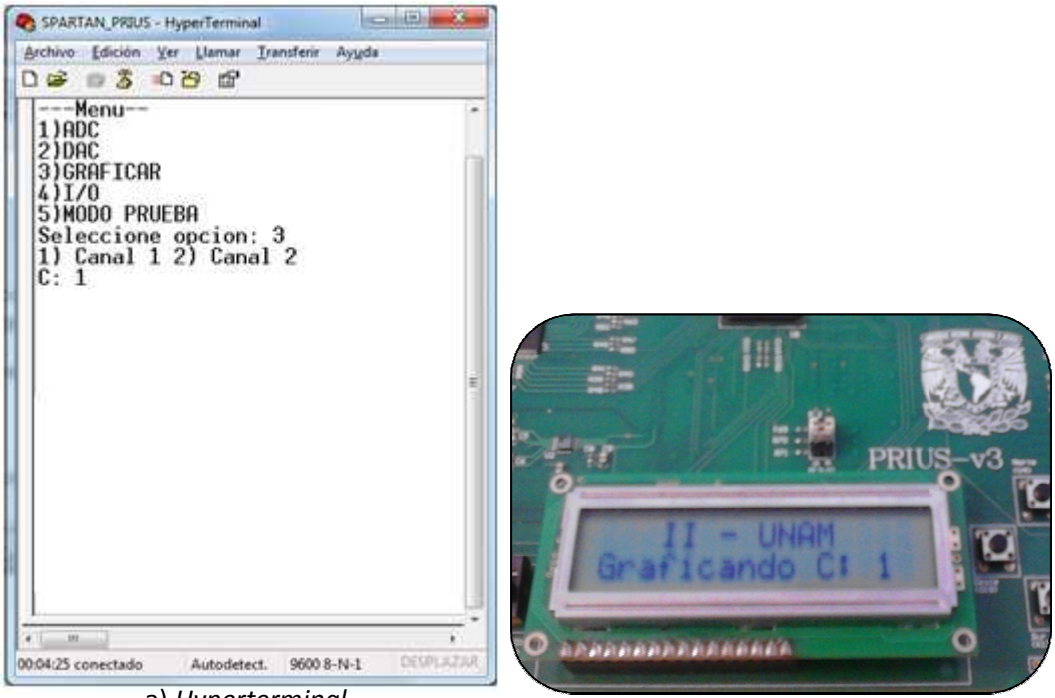
Voltaje de entrada	**Voltaje Medido**				**%Error**			
Vref=3.33	A	B	C	D	A	B	C	D
0.4	0.404	0.405	0.403	0.404	1%	1.25%	0.75%	1%
0.8	0.807	0.808	0.806	0.808	0.88%	1%	0.75%	1%
1.2	1.211	1.213	1.212	1.211	0.92%	1.08%	1%	0.92%
1.6	1.614	1.616	1.615	1.615	0.88%	1%	0.94%	0.94%
2	2.018	2.021	2.02	2.019	0.9%	1.05%	1%	0.95%
2.4	2.422	2.425	2.424	2.422	0.92%	1.04%	1%	0.92%
2.8	2.826	2.83	2.828	2.827	0.93%	1.07%	1%	0.96%
3.2	3.229	3.23	3.232	3.229	0.91%	0.94%	1%	0.91%

Tabla 5.15. Voltajes de salida del DAC en la tarjeta PRIUS.

Con estos resultados se concluye que la comunicación con los DACs es correcta, además de corroborar el funcionamiento de la comunicación UART en la tarjeta PRIUS.

3) GRAFICAR

En esta opción se leen los datos almacenados en la memoria DDR y se envían por la comunicación UART. En la figura 5.31 inciso *a* se muestra la pantalla de *Hyperterminal* con el mensaje "C: 1" y en el inciso *b* se muestra el display LCD con el mensajes "Graficando C: 1" que es el canal seleccionado.



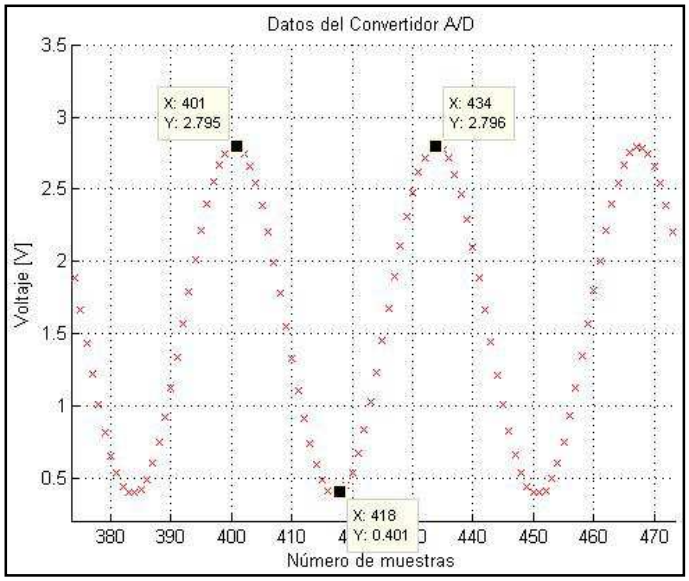
a) *Hyperterminal* b) Display LCD

Figura 5.31. Mensajes de la opción 3 en la tarjeta PRIUS.

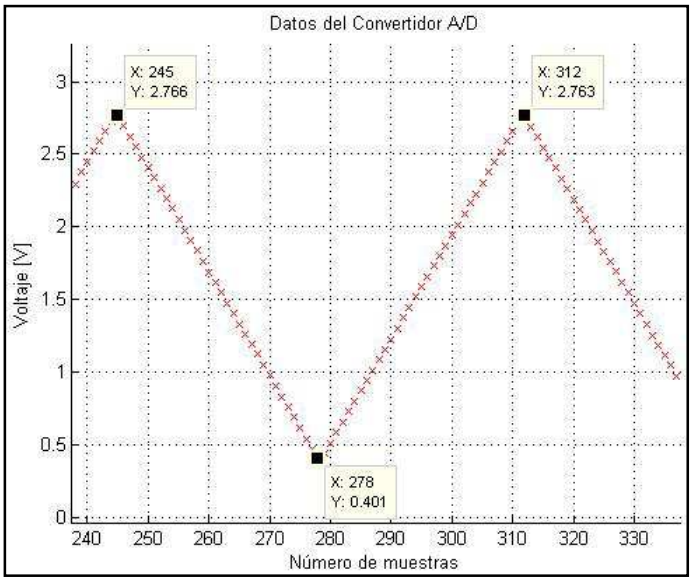
Para comprobar el funcionamiento de la opción 3, se ingresó una señal senoidal en el canal A (canal 1) y una señal triangular en el canal B (canal 2), se adquirieron los datos y después se graficaron dichos canales, todo esto con una ganancia unitaria en el amplificador. En la figura 5.32 se muestran las dos gráficas que se obtuvieron en MATLAB de ambos canales, en el inciso *a* del canal A (canal 1) y en el inciso *b* del canal B (canal 2). El ADC adquiere un dato cada 29.8µs correspondiente a una frecuencia de 34.013kHz. El tiempo de 29.8µs incluye el almacenamiento y el procesamiento de los datos (ajustarlos a un número de 16 bits) en la memoria DDR.

En la tabla 5.16 se muestran los parámetros de las señales de entrada/salida que se obtuvieron con MATLAB. Para obtener la frecuencia de las señales graficadas en MATLAB se multiplica el número de muestras por el tiempo de adquisición del ADC y se obtiene el inverso de dicho producto. Para el caso de la onda senoidal, las muestras son: 434-401=33 (de los marcadores de la figura 5.32), estas 33 muestra se multiplican por 29.8µs: 33* 29.8µs = 983.4µs y el inverso es

1.0169 kHz. Para el caso de la onda triangular es 312-245=67 (de los marcadores de la figura 5.32), multiplicadas por el tiempo de adquisición: 67*29.8µs=1.9966ms y el inverso es 500.8514Hz.



a) Señal senoidal



b) Señal triangular

Figura 5.32. Gráficas en MATLAB de la opción 3 en la tarjeta PRIUS.

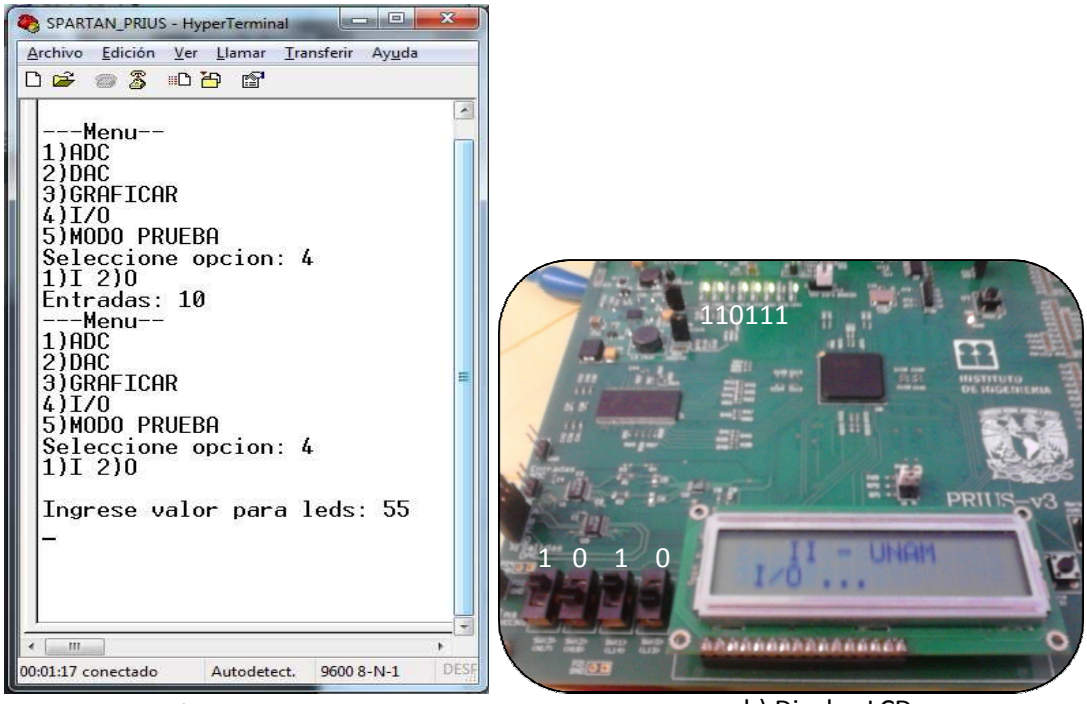
Parámetro /Valor	Entrada		Salida		%Error	
	Senoidal	Triangular	Senoidal	Triangular	Senodal	Triangular
Vmax	2.88V	2.88V	2.796V	2.766V	2.92%	3.96%
Vmin	0.4V	0.4V	0.401V	0.401V	0.25%	0.25%
Vpp	2.48V	2.48V	2.395V	2.365V	3.43%	4.64%
Frecuencia	1 kHz	500 Hz	1.0168kHz	500.8514Hz	1.68%	0.17%

Tabla 5.16. Parámetros de las señales de MATLAB.

Con estos resultados se valida el funcionamiento de la opción 3 en la tarjeta PRIUS, validando también el buen funcionamiento de los componentes involucrados en dicha opción.

4) I/O

En esta opción se lee el estado de los interruptores deslizables y se controlan los LEDs. Mediante *Hyperterminal* se selecciona entre leer el estado de los interruptores ó activar los LEDs. En el inciso *a* de la figura 5.33 se muestra la pantalla de *Hyperterminal* en donde se observan la lectura de los interruptores y también se observa el valor ingresado en los LEDs. En el inciso *b* de la figura 5.33 se muestran los interruptores, el display LCD y los LEDs en la tarjeta PRIUS. Los interruptores tienen una un valor binario de "1010" que en decimal es 10, mientras que para los LEDs el valor decimal ingresado es 55 que en binario es "110111". El display LCD muestra en la segunda línea el mensaje "I/O …" el cual indica que se ha seleccionado dicha opción.



a) *Hyperterminal* b) Display LCD

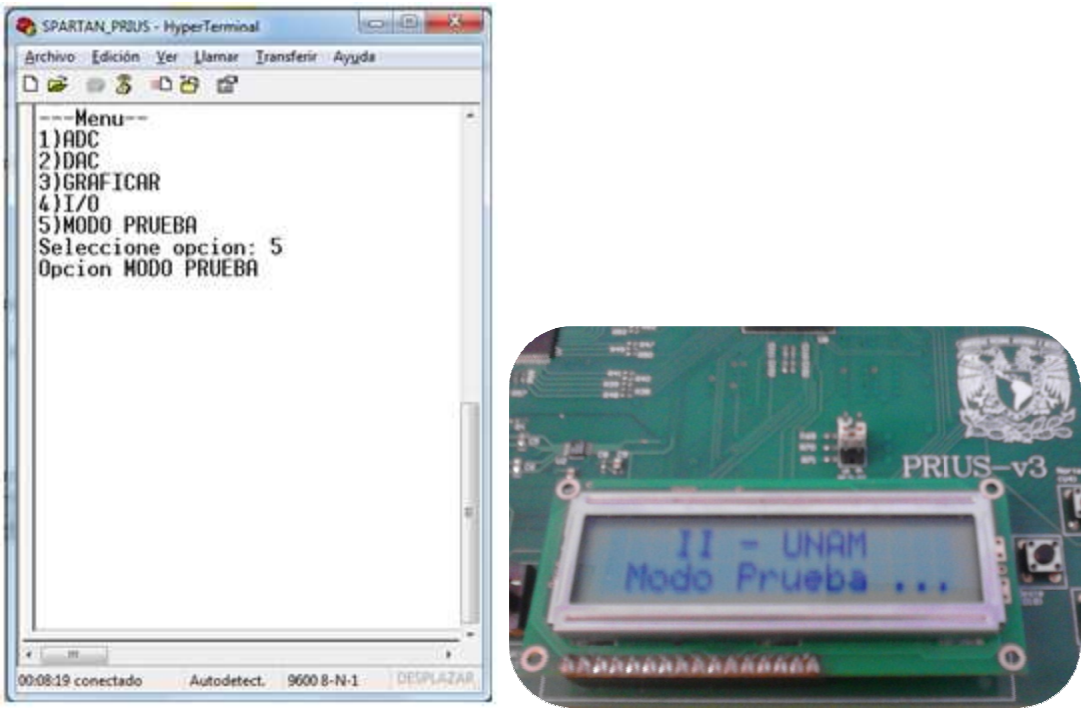
Figura 5.33. Mensajes de la opción 4 en la tarjeta PRIUS.

Con estos resultados se corrobora el correcto funcionamiento de los interruptores y de los LEDs en la tarjeta PRIUS.

5) MODO PRUEBA

En esta opción se pretende corroborar el funcionamiento del sistema de monitoreo y control, principalmente de la comunicación SPI, por lo que los dispositivos que se utilizan son el ADC, la memoria DDR y el DAC. Lo que se realiza es conectar la opción 1 y la opción 2, es decir, adquirir datos mediante el ADC, almacenarlos en memoria DDR, leer los datos de la memoria, ajustarlos para los DACs y enviarlos a los cuatro canales del DAC.

En esta opción se muestra el mensaje “MODO PRUEBA” en *Hyperterminal* y en el display LCD se muestra el mensaje “Modo Prueba …”, así como se muestra en la figura 5.34 incisos *a* y *b* respectivamente.



a) *Hyperterminal* b) Display LCD

Figura 5.34. Mensajes de la opción 5 en la tarjeta PRIUS.

En la figura 5.35 se muestran las salidas del DAC en donde se observan las señales de entrada del ADC. Cabe señalar que la amplitud de la señales es la misma debido a que el voltaje de referencia es el mismo para los cuatro canales del DAC.

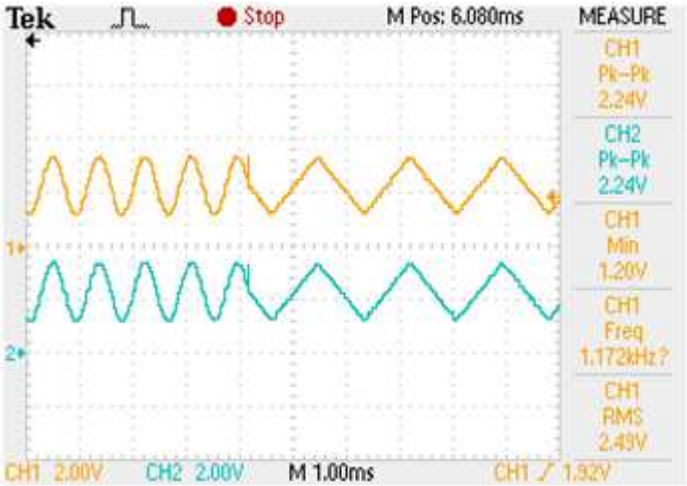


Figura 5.35. Salida del DAC en el modo prueba tarjeta PRIUS.

En la figura 5.36 se muestra la conexión de la tarjeta PRIUS al realizar las pruebas mostradas. Cabe señalar que una vez configurado el FPGA de la tarjeta PRIUS, la tarjeta SPARTAN 3E ya no es necesaria.

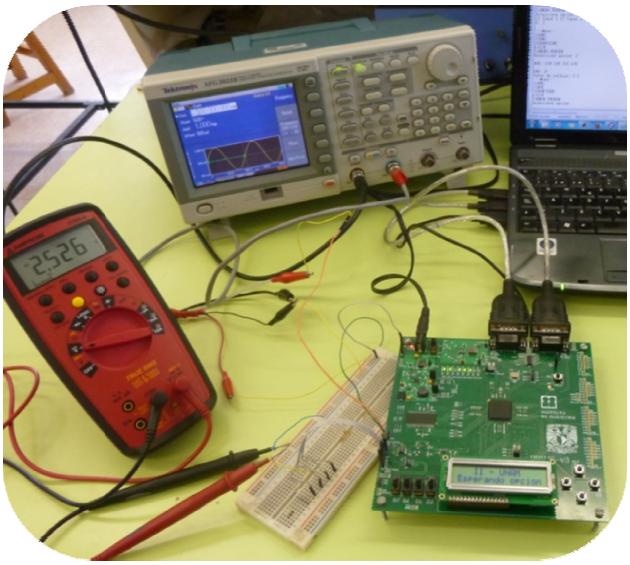


Figura 5.36. Conexión de la tarjeta PRIUS.

Con los resultados obtenidos en las pruebas realizadas, se validó el funcionamiento de la tarjeta PRIUS, al igual que se validó el funcionamiento del sistema de monitoreo y control diseñado.

Una vez comprobada la correcta operación del sistema desarrollado, en el siguiente capítulo se abordarán las conclusiones y los resultados obtenidos con el presente trabajo de tesis.

Capítulo VI
Resultados y conclusiones

En este capítulo se presentan los resultados que se obtuvieron en la elaboración del presente trabajo de tesis, así como también las conclusiones generales y algunas recomendaciones.

6.1. Resultados

Los resultados más importantes que se obtuvieron en el desarrollo de este trabajo fueron:

- Comprensión de los sistemas embebidos en un FPGA.

- Diseño e implementación de un sistema de monitoreo y control basado en un FPGA. Se utilizó un procesador dentro del FPGA para realizar el sistema. Dicho sistema está basado en la comunicación UART y cuenta con cinco posibles opciones:

  - 1) ADC: se realiza la captura de señales analógicas y se almacenan en la memoria externa DDR RAM.

  - 2) DAC: a través de *Hyperterminal* se realiza la captura del dato a convertir, y que posteriormente es enviado al sistema para su conversión digital analógica.

  - 3) GRAFICAR: se leen los datos de la memoria DDR RAM adquiridos con la opción 1 y se envían a la PC, vía UART, para su graficación.

  - 4) I/O: se leen los estados de los interruptores, simulando sistemas externos, y se activan los LEDs, simulando la activación de sistemas externos.

  - 5) MODO PRUEBA: se corrobora que la comunicación con los convertidores ADC y DAC sea correcta.

- Desarrollo del sistema PRIUS. Consiste en el diseño y construcción de una PCB multicapa. Se obtuvo una PCB de 6 capas completamente funcional, la cual contiene los componentes del sistema de monitoreo y control, estos componentes son: un FPGA xc3s1600e de Xilinx, un ADC de dos canales, un DAC de cuatro canales, dos puertos RS-232, un cristal de 50MHz, 40 terminales de entrada/salida, ocho LEDs, cuatro interruptores deslizables, cuatro interruptores *push button*, un *display* LCD de 2x16, terminales disponibles de GND, 3.3V y 5V, una memoria DDR RAM de 512 Mbits y un puerto JTAG.

6.2. Conclusiones

Con base en los resultados obtenidos del presente trabajo de tesis, se concluye que se cumplió satisfactoriamente el objetivo planteado, ya que se obtuvo un sistema de monitoreo y control basado en un FPGA.

El sistema de monitoreo y control se desarrolló utilizando como base una arquitectura de procesamiento previamente elaborada por Xilinx, que en este caso fue MicroBlaze, agilizando con ello el desarrollo del sistema.

El sistema funcionó de manera correcta en la PCB diseñada y ensamblada. Dos factores fundamentales para el correcto funcionamiento del sistema fueron: contar con los equipos adecuados para el ensamblado y saber utilizar dichos equipos, ya que si no se cuenta con ellos ó si no se saben utilizar, es muy difícil obtener resultados satisfactorio.

Actualmente el sistema está listo para ser utilizado para monitorear parámetros en el almacenamiento de energía mediante superconductores. Otra aplicación detectada es el monitoreo de parámetros vehiculares utilizando el bus CAN.

Para finalizar, en el presente trabajo de tesis se aplicaron los conocimientos adquiridos en la carrera de Ingeniería Eléctrica Electrónica para el diseño y construcción de sistemas digitales, dando como resultado un sistema de gran potencial tanto en software como en hardware.

6.3. Recomendaciones

Algunas recomendaciones que pueden mejorar el sistema de monitoreo y control son:

Para la parte del software:

- Hacer uso de las interrupciones de MicroBlaze.

- Realizar la detección de teclas como *enter*, barra espaciadora y retroceso de *Hyperterminal*.

Para la parte de la PCB

- Realizar el módulo USB-JTAG. Debido a la poca información con respecto a este módulo, se partiría prácticamente desde cero y se puede considerar como un tema de tesis.

- Colocar memorias no volátiles para el almacenamiento de la configuración del FPGA. Existen memorias no volátiles de Xilinx, con interfaz JTAG, compatibles con el FPGA SPARTAN 3E para realizar la configuración desde dicha memoria.

- Tener un mayor número de terminales de entrada/salida.

Finalmente, es deseable que el presente trabajo de tesis sea la base para seguir desarrollando proyectos con FPGAs. Con el incremento en las capacidades de integración de los circuitos, existen FPGAs con una gran variedad de recursos y además, con más de 1100 terminales, lo cual constituye un reto tanto para manejar la gran cantidad de recursos, así como para la realización de una PCB para dichos FPGAs.

Bibliografía

Bibliografía

- Brown Stephen, Vranesic Zvonko, *Fundamentals of Digital Logic with VHDL Design*, Second edition, Mc Graw Hill, Mexico city, 2005.

- Hamblen James O., Hall Tyson S., Furman Michael D., *Rapid prototyping of digital systems Quartus II®*, Springer, United States of America, 2006.

- Hauck Scott, DeHon André, *Reconfigurable Computing the Theory and Practice of FPGA Based Computation*, Morgan Kaufmann, Amsterdam, 2008.

- Hsiung Pao-Ann, Santambrogio Marco D., Huang Chun-Hsian, *Reconfigurable System Design and Verification*, CRC Press, Boca Raton, 2009.

- Kuon Ian, Tessier Russell, Rose Jonathan, *FPGA Architecture: Survey and Challenges*, Now publishers, Boston, 2008.

- Nelson Vitor P., Nagle H. Troy, Carroll Bill D., Irwin J. David, *Digital Logic Circuit Analysis and Design*, Prentice-Hall, New Jersey, 1995.

- Pardo Carpio Fernando, Boluda Grau José A., *VHDL Lenguaje para síntesis y modelado de circuitos*, RA-MA, Madrid, 1999.

- Pellerin David, Taylor Douglas, *VHDL made easy!*, Prentice Hall PTR, New Jersey, 1997.

- Pellerin David, Thibault Scott, *Practical FPGA Programming in C*, Prentice Hall, Mexico city, 2005.

- Roth Charles H. Jr, *Digital Systems Design Using VHDL*, PWS, Mexico city, 1998.

- Wolf Wayne, *FPGA-Based System Design*, Prentice Hall PTR, New Jersey, 2004.

Notas de aplicación y manuales

- *Device Package User Guide*, UG112 (v3.7), September 2012, Xilinx.

- *EDK Concepts, Tools, and Techniques A Hands-On Guide to Effective Embedded System Design*, UG683 (v13.2), July 2011, Xilinx.

- *Four-and Six-Layer, High-Speed PCB Design for the Spartan-3E FT256 BGA Package*, XAPP489, 2006, Xilinx.

- *ISE In-Depth Tutorial*, UG695 (v13.1), March 2011, Xilinx.

- *MicroBlaze Development Kit Spartan-3E 1600E Edition User Guide*, UG257 (v1.1), December 2007, Xilinx.

- *QFN/SON PCB Attachment*, SLUA271A, Revised september 2007, Steve Kummerl, Bernhard Lange, Dominic Nguyen , Texas Instruments.

Hojas de especificaciones

- *LTC1407/LTC1407A Serial 12-Bit/14-Bit, 3Msps Simultaneous Sampling ADCs with Shutdown*, LT 0109 REV B, 2003, Linear Technology.

- *LTC2604/LTC2614/LTC2624 Quad 16-Bit Rail-to-Rail DACs in 16-Lead SSOP*, LT 0309 REV D, 2004, Linear Technology.

- *LTC6912 Dual Programmable Gain Amplifiers with Serial Digital Interface*, LT/LT 1005 REV A, 2004, Linear Technology.

- *MicroBlaze Processor Reference Guide Embedded Development Kit EDK 13.2*, UG081 (v13.2), June 2011, Xilinx.

- *Spartan-3E FPGA Family*, DS312, october 2012, Xilinx.

- *TPS75003 Triple-Supply Power Management IC for Powering FPGAs and DSPs*, SBVS052I, Revised august 2010, Texas Instruments.

Apéndices

A. Glosario de términos

B. Terminales del FPGA

C. Software principal

Apéndice A: Glosario de términos

Bitstream. Archivo de configuración del FPGA que contiene tanto el diseño realizado como las conexiones programables para implementarlo.

BSR. *Boundary Scan Register*. Registro principal del estándar JTAG, es utilizado para mover datos desde y para el dispositivo.

CLB. *Configurable Logic Block. S*on los componentes principales de un FPGA, en ellos se implementan las funciones lógicas del diseño a realizado.

DDR. *Double Data Rate*. Memorias volátiles que pueden manejar datos tanto en el flanco de subida como en el flanco de bajada de la señal de reloj.

EDK. *Embedded Development Kit*. Conjunto de software que ofrece Xilinx para el desarrollo de sistemas embebidos dentro de un FPGA.

FPGA. *Field Programmable Gate Array*. Circuito integrado destinado a la implementación de hardware personalizado con la capacidad de ser reconfigurado una infinidad de veces

HDL. *Hardware Description Language*. Conjunto de software que permiten diseñar y depurar sistemas digitales.

IOB. *Input Output Block*. Son los encargados de controlar el flujo de datos tanto a la entrada como a la salida del FPGA, así como también de los bloques lógicos internos.

IP. *Intellectual property*. Son especificaciones de hardware que realiza una tarea muy particular. También se les conoce como núcleos, módulos o *cores*.

JTAG. *Joint Test Action Group*. Nombre por el cual se le conoce al estándar IEEE 1149.1 que lleva por nombre: *Standard Test Access Port and Boundary-Scan Architecture* y es utilizado para pruebas de circuitos integrados y como herramienta para depuración de aplicaciones embebidas.

LUT. *Lookup Table*. Se encuentran en los CLBs y en ellas se implementan las funciones lógicas a realizar por el FPGA.

PLB. *Processor Local Bus*. Es una infraestructura para la conexión de IPs maestros y/o esclavos y se encarga de gestionar y adminstrar la comunicación entre los ellos.

SDK. *Software Development Kit*. Herramienta de Xilinx para realizar la aplicación software para el procesador embebido en el FPGA.

SDRAM. *Synchronous Dinamic Random Access Memory*. Tipo de memoria RAM en donde los datos necesitan ser refrescado cada determinado tiempo.

SKEW. Fenómeno en donde una señal, generalmente de reloj, llega a diferente tiempo a dos o más dispositivos conectados a ella.

SPI. *Serial Peripheral Interface*. Protocolo síncrono entre micro-controladores y periféricos con una estructura maestro-esclavo que utiliza cuatro líneas para el intercambio de información.

STACK. Nombre con el que se hace referencia a las capas de cobre que tiene una tarjeta de circuito impreso.

UART. *Universal Asynchronous Receiver Transmitter*. Interfaz serie para una comunicación asíncrona.

VHDL. *Very High Speed Integrated Circuit Hardware Description Language*. Es el lenguaje de descripción de hardware más utilizado en el diseño y depuración de circuitos digitales.

XPS. *Xilinx Platform Studio*. Herramienta de Xilinx para realizar el diseño de hardware para el procesador embebido en el FPGA.

Apéndice B: Terminales del FPGA

En este apéndice se muestran las terminales utilizadas en el sistema de monitoreo y control que se diseñó con MicroBlaze y también se muestran todas las terminales con las que cuenta el FPGA.

B.1. Terminales del sistema de monitoreo y control

Las terminales que se utilizaron en el sistema de monitoreo y control son parte del archivo creado por XPS, mediante éste se indican las terminales que se conectan con el exterior del FPGA. La estructura básica para definir terminales inicia con la palabra reservada *NET*, seguido del nombre asignado a la señal; después se tiene la palabra reservada *LOC*, con ésta se coloca el nombre de la terminal del FPGA. Separado con una barra vertical "|" se coloca el tipo de estándar que debe manejar la terminal del FPGA, esto se realiza mediante la palabra reservada *IOSTANDARD*, también se le indica si la terminal debe tener resistores *PULLDOWN* o *PULLUP* separado también con una barra vertical "|". El orden del estándar y de los resistores es indiferente. Otras palabras reservadas, como *PERIOD*, se utiliza en las señales de reloj para indicar el periodo de la misma ó *TIG*, que se utiliza para no realizar análisis y optimizaciones en la señal.

A manera de ejemplo se presentan los siguientes listados:

```
#  Spartan-3E 1600E MicroBlaze Dev Board
Net fpga_0_RS232_DTE_RX_pin LOC=U8  |  IOSTANDARD = LVCMOS33;
Net fpga_0_RS232_DTE_TX_pin LOC=M13  |  IOSTANDARD = LVCMOS33;
Net fpga_0_RS232_DCE_RX_pin LOC=R7  |  IOSTANDARD = LVCMOS33;
Net fpga_0_RS232_DCE_TX_pin LOC=M14  |  IOSTANDARD = LVCMOS33;

Net xps_spi_0_SCK_O_pin LOC=U16  |  IOSTANDARD = LVCMOS33;
Net xps_spi_0_MISO_I_pin LOC=N10  |  IOSTANDARD = LVCMOS33;
Net xps_spi_0_MOSI_O_pin LOC=T4  |  IOSTANDARD = LVCMOS33;
#ADC
Net xps_spi_0_SS_O_pin<0> LOC=N15  |  IOSTANDARD = LVCMOS33;
#AMPLI
Net xps_spi_0_SS_O_pin<1> LOC=N7  |  IOSTANDARD = LVCMOS33;
#DAC
Net xps_spi_0_SS_O_pin<2> LOC=N8  |  IOSTANDARD = LVCMOS33;

#DAC_CLR
Net OTROS_DISPOS_GPIO_IO_O_pin<0> LOC=P8 |  IOSTANDARD = LVCMOS33;
#SHDN_AMPLI
Net OTROS_DISPOS_GPIO_IO_O_pin<1> LOC=P7 |  IOSTANDARD = LVCMOS33;
#AD_CONV
Net OTROS_DISPOS_GPIO_IO_O_pin<2> LOC=P11 |  IOSTANDARD = LVCMOS33;
```

```
#FPGA_INIT_B
Net OTROS_DISPOS_GPIO_IO_O_pin<3> LOC=T3 |  IOSTANDARD = LVCMOS33;
#SF_CE0
Net OTROS_DISPOS_GPIO_IO_O_pin<4> LOC= D16 |  IOSTANDARD = LVCMOS33;
#SS_B
Net OTROS_DISPOS_GPIO_IO_O_pin<5> LOC=U3 |  IOSTANDARD = LVCMOS33;
Net OTROS_DISPOS_GPIO_IO_O_pin<6> LOC=N14 |  IOSTANDARD = LVCMOS33;
Net OTROS_DISPOS_GPIO_IO_O_pin<7> LOC=E15 |  IOSTANDARD = LVCMOS33;

Net xps_timer_0_GenerateOut0_pin LOC=V5 | IOSTANDARD = LVCMOS33;
Net xps_timer_0_GenerateOut1_pin LOC=V6 | IOSTANDARD = LVCMOS33;
Net xps_timer_0_PWM0_pin LOC=N12 | IOSTANDARD = LVCMOS33;

Net fpga_0_LEDs_6Bit_GPIO_IO_O_pin<0> LOC=A8  |  IOSTANDARD = LVCMOS33;
Net fpga_0_LEDs_6Bit_GPIO_IO_O_pin<1> LOC=G9  |  IOSTANDARD = LVCMOS33;
Net fpga_0_LEDs_6Bit_GPIO_IO_O_pin<2> LOC=A7  |  IOSTANDARD = LVCMOS33;
Net fpga_0_LEDs_6Bit_GPIO_IO_O_pin<3> LOC=D13  |  IOSTANDARD = LVCMOS33;
Net fpga_0_LEDs_6Bit_GPIO_IO_O_pin<4> LOC=E6  |  IOSTANDARD = LVCMOS33;
Net fpga_0_LEDs_6Bit_GPIO_IO_O_pin<5> LOC=D6  |  IOSTANDARD = LVCMOS33;
Net fpga_0_LEDs_1Bit_GPIO_IO_O_pin LOC=D4  |  IOSTANDARD = SSTL2_I;

Netfpga_0_DIP_Switches_4Bit_GPIO_IO_I_pin<0>LOC=N17|PULLDOWN|IOSTANDARD=LVCMOS3
3;
Netfpga_0_DIP_Switches_4Bit_GPIO_IO_I_pin<1>LOC=H18|PULLDOWN|IOSTANDARD=LVCMOS3
3;
Netfpga_0_DIP_Switches_4Bit_GPIO_IO_I_pin<2>LOC=L14|PULLDOWN|IOSTANDARD=LVCMOS33
;
Netfpga_0_DIP_Switches_4Bit_GPIO_IO_I_pin<3>LOC=L13|PULLDOWN|IOSTANDARD=LVCMOS33
;

Net fpga_0_Buttons_3Bit_GPIO_IO_I_pin<0> LOC=K17 | PULLDOWN  |IOSTANDARD = LVCMOS33;
Net fpga_0_Buttons_3Bit_GPIO_IO_I_pin<1> LOC=H13 | PULLDOWN  |IOSTANDARD = LVCMOS33;
Net fpga_0_Buttons_3Bit_GPIO_IO_I_pin<2> LOC=V4 | PULLDOWN  |  IOSTANDARD = LVCMOS33;

Netfpga_0_Character_LCD_2x16_GPIO_IO_O_pin<0>LOC=M18|PULLDOWN|IOSTANDARD=LVCM
OS33;
Net fpga_0_Character_LCD_2x16_GPIO_IO_O_pin<1> LOC=L18  |  IOSTANDARD = LVCMOS33;
Net fpga_0_Character_LCD_2x16_GPIO_IO_O_pin<2> LOC=L17  |  IOSTANDARD = LVCMOS33;
Net fpga_0_Character_LCD_2x16_GPIO_IO_O_pin<3> LOC=t8  |  IOSTANDARD = LVCMOS33;
Net fpga_0_Character_LCD_2x16_GPIO_IO_O_pin<4> LOC=r8  |  IOSTANDARD = LVCMOS33;
Net fpga_0_Character_LCD_2x16_GPIO_IO_O_pin<5> LOC=p6  |  IOSTANDARD = LVCMOS33;
Net fpga_0_Character_LCD_2x16_GPIO_IO_O_pin<6> LOC=m16  |  IOSTANDARD = LVCMOS33;
```

```
Net fpga_0_Character_LCD_2x16_LCD_DB11_pin LOC=M15 |  IOSTANDARD = LVCMOS33;
Net fpga_0_Character_LCD_2x16_LCD_DB10_pin LOC=P17 |  IOSTANDARD = LVCMOS33;
Net fpga_0_Character_LCD_2x16_LCD_DB9_pin LOC=R16  |  IOSTANDARD = LVCMOS33;
Net fpga_0_Character_LCD_2x16_LCD_DB8_pin LOC=R15  |  IOSTANDARD = LVCMOS33;

Net fpga_0_Rotary_Encoder_GPIO_IO_I_pin<0> LOC=K18| PULLUP  |  IOSTANDARD = LVCMOS33;
Net fpga_0_Rotary_Encoder_GPIO_IO_I_pin<1> LOC=G18| PULLUP  |  IOSTANDARD = LVCMOS33;
Net fpga_0_Rotary_Encoder_GPIO_IO_I_pin<2> LOC=V16|PULLDOWN|IOSTANDARD =LVCMOS33;

Net fpga_0_DDR_SDRAM_DDR_Clk_pin LOC=J5  |  IOSTANDARD = DIFF_SSTL2_I;
Net fpga_0_DDR_SDRAM_DDR_Clk_n_pin LOC=J4  |  IOSTANDARD = DIFF_SSTL2_I;
Net fpga_0_DDR_SDRAM_DDR_CE_pin LOC=K3  |  IOSTANDARD = SSTL2_I;
Net fpga_0_DDR_SDRAM_DDR_CS_n_pin LOC=K4  |  IOSTANDARD = SSTL2_I;
Net fpga_0_DDR_SDRAM_DDR_RAS_n_pin LOC=C1  |  IOSTANDARD = SSTL2_I;
Net fpga_0_DDR_SDRAM_DDR_CAS_n_pin LOC=C2  |  IOSTANDARD = SSTL2_I;
Net fpga_0_DDR_SDRAM_DDR_WE_n_pin LOC=D1  |  IOSTANDARD = SSTL2_I;
Net fpga_0_DDR_SDRAM_DDR_BankAddr_pin<0> LOC=K5  |  IOSTANDARD = SSTL2_I;
Net fpga_0_DDR_SDRAM_DDR_BankAddr_pin<1> LOC=K6  |  IOSTANDARD = SSTL2_I;
Net fpga_0_DDR_SDRAM_DDR_Addr_pin<0> LOC=T1  |  IOSTANDARD = SSTL2_I;
Net fpga_0_DDR_SDRAM_DDR_Addr_pin<1> LOC=R3  |  IOSTANDARD = SSTL2_I;
Net fpga_0_DDR_SDRAM_DDR_Addr_pin<2> LOC=R2  |  IOSTANDARD = SSTL2_I;
Net fpga_0_DDR_SDRAM_DDR_Addr_pin<3> LOC=P1  |  IOSTANDARD = SSTL2_I;
Net fpga_0_DDR_SDRAM_DDR_Addr_pin<4> LOC=E4  |  IOSTANDARD = SSTL2_I;
Net fpga_0_DDR_SDRAM_DDR_Addr_pin<5> LOC=H4  |  IOSTANDARD = SSTL2_I;
Net fpga_0_DDR_SDRAM_DDR_Addr_pin<6> LOC=H3  |  IOSTANDARD = SSTL2_I;
Net fpga_0_DDR_SDRAM_DDR_Addr_pin<7> LOC=H1  |  IOSTANDARD = SSTL2_I;
Net fpga_0_DDR_SDRAM_DDR_Addr_pin<8> LOC=H2  |  IOSTANDARD = SSTL2_I;
Net fpga_0_DDR_SDRAM_DDR_Addr_pin<9> LOC=N4  |  IOSTANDARD = SSTL2_I;
Net fpga_0_DDR_SDRAM_DDR_Addr_pin<10> LOC=T2  |  IOSTANDARD = SSTL2_I;
Net fpga_0_DDR_SDRAM_DDR_Addr_pin<11> LOC=N5  |  IOSTANDARD = SSTL2_I;
Net fpga_0_DDR_SDRAM_DDR_Addr_pin<12> LOC=P2  |  IOSTANDARD = SSTL2_I;
Net fpga_0_DDR_SDRAM_DDR_DQ_pin<0> LOC=L2  |  IOSTANDARD = SSTL2_I  |  PULLUP;
Net fpga_0_DDR_SDRAM_DDR_DQ_pin<1> LOC=L1  |  IOSTANDARD = SSTL2_I  |  PULLUP;
Net fpga_0_DDR_SDRAM_DDR_DQ_pin<2> LOC=L3  |  IOSTANDARD = SSTL2_I  |  PULLUP;
Net fpga_0_DDR_SDRAM_DDR_DQ_pin<3> LOC=L4  |  IOSTANDARD = SSTL2_I  |  PULLUP;
Net fpga_0_DDR_SDRAM_DDR_DQ_pin<4> LOC=M3  |  IOSTANDARD = SSTL2_I  |  PULLUP;
Net fpga_0_DDR_SDRAM_DDR_DQ_pin<5> LOC=M4  |  IOSTANDARD = SSTL2_I  |  PULLUP;
Net fpga_0_DDR_SDRAM_DDR_DQ_pin<6> LOC=M5  |  IOSTANDARD = SSTL2_I  |  PULLUP;
Net fpga_0_DDR_SDRAM_DDR_DQ_pin<7> LOC=M6  |  IOSTANDARD = SSTL2_I  |  PULLUP;
Net fpga_0_DDR_SDRAM_DDR_DQ_pin<8> LOC=E2  |  IOSTANDARD = SSTL2_I  |  PULLUP;
Net fpga_0_DDR_SDRAM_DDR_DQ_pin<9> LOC=E1  |  IOSTANDARD = SSTL2_I  |  PULLUP;
Net fpga_0_DDR_SDRAM_DDR_DQ_pin<10> LOC=F1  |  IOSTANDARD = SSTL2_I  |  PULLUP;
```

```
Net fpga_0_DDR_SDRAM_DDR_DQ_pin<11> LOC=F2  |  IOSTANDARD = SSTL2_I  |  PULLUP;
Net fpga_0_DDR_SDRAM_DDR_DQ_pin<12> LOC=G6  |  IOSTANDARD = SSTL2_I  |  PULLUP;
Net fpga_0_DDR_SDRAM_DDR_DQ_pin<13> LOC=G5  |  IOSTANDARD = SSTL2_I  |  PULLUP;
Net fpga_0_DDR_SDRAM_DDR_DQ_pin<14> LOC=H6  |  IOSTANDARD = SSTL2_I  |  PULLUP;
Net fpga_0_DDR_SDRAM_DDR_DQ_pin<15> LOC=H5  |  IOSTANDARD = SSTL2_I  |  PULLUP;
Net fpga_0_DDR_SDRAM_DDR_DM_pin<0> LOC=J2  |  IOSTANDARD = SSTL2_I;
Net fpga_0_DDR_SDRAM_DDR_DM_pin<1> LOC=J1  |  IOSTANDARD = SSTL2_I;
Net fpga_0_DDR_SDRAM_DDR_DQS_pin<0> LOC=L6  |  IOSTANDARD = SSTL2_I;
Net fpga_0_DDR_SDRAM_DDR_DQS_pin<1> LOC=G3  |  IOSTANDARD = SSTL2_I  |  PULLUP;
Net fpga_0_DDR_SDRAM_ddr_dqs_div_io_pin LOC=P13  |  IOSTANDARD = LVCMOS33;

Net fpga_0_clk_1_sys_clk_pin TNM_NET = sys_clk_pin;
TIMESPEC TS_sys_clk_pin = PERIOD sys_clk_pin 50000 kHz;
Net fpga_0_clk_1_sys_clk_pin LOC=C9  |  IOSTANDARD = LVCMOS33  |
CLOCK_DEDICATED_ROUTE = FALSE;
Net fpga_0_rst_1_sys_rst_pin TIG;
Net fpga_0_rst_1_sys_rst_pin LOC=D18  |  IOSTANDARD = LVCMOS33  |  PULLDOWN;
```

B.2. Terminales del FPGA

En la tabla B.1 se muestra la lista de terminales del FPGA, cabe señalar que las terminales están agrupadas en módulos, es decir: SPI, RS232, memoria DDR SDRAM, GPIOs, entrada/salida, configuración del FPGA, voltaje de alimentación y las no utilizadas. Se observa el número de terminal del FPGA, el nombre asignado, la descripción, el tipo de terminal y el banco al que pertenece. Si el tipo de terminal es GCLK es una terminal que puede manejar un reloj global, si la terminal es del tipo RHCLK o LHCLK es una terminal que maneja señales de reloj del lado derecho e izquierdo del componente respectivamente. Si es DUAL significa que tiene dos funciones: como entrada/salida y también se utiliza para configurar el FPGA. Si es I/O significa que se puede utilizar como entrada ó como salida. Si el tipo es INPUT significa que esa terminal únicamente puede ser manejada como entrada. Si el tipo es VREF significa que se utiliza para el voltaje de referencia del banco al que pertenece (se utiliza para estándares que necesitan voltaje de referencia).

#	Terminal FPGA	Nombre	Descripción	Pin	Tipo	Banco
1	C9	CLK_50MHz	Reloj principal de 50MHz	IO_L14P_0/GCLK10	GCLK	0
2	T4	SPI_MOSI	Señal MOSI del SPI	IO_L03N_2/MOSI/CSI_B	DUAL	2
3	N7	AMP_CS	CS amplificador	IO_L07P_2	I/O	2
4	P7	AMP_SHDN	Habilitación amplificador	IO_L07N_2	I/O	2
5	P8	DAC_CLR	CLEAR del DAC	IO_L09P_2	I/O	2
6	N8	DAC_CS	CS DAC	IO_L09N_2	I/O	2
7	N10	SPI_MISO	Señal MISO del SPI	IO_L16N_2/DIN/D0	DUAL	2
8	P11	AD_CONV	Señal de inicio de conversión	IO_L18P_2	I/O	2
9	U16	SPI_SCK	Señal de reloj para el SPI	IO_L26N_2/CCLK	DUAL	2
10	E18	AMP_DOUT	Señal MISO del amplificador	IP	INPUT	1
11	M13	RS232_DTE_TXD	Señal Tx del DTE	IO_L05N_1/VREF_1	VREF	1
12	M14	RS232_DCE_TXD	Señal Tx del DCE	IO_L05P_1	I/O	1
13	R7	RS232_DCE_RXD	Señal Rx del DCE	IP_L08N_2	INPUT	2
14	U8	RS232_DTE_RXD	Señal Rx del DTE	IP_L11P_2	INPUT	2
15	T1	SD_A0	Línea de dirección A0 para DDR	IO_L24P_3	I/O	3
16	R3	SD_A1	Línea de dirección A1 para DDR	IO_L23P_3	I/O	3
17	R2	SD_A2	Línea de dirección A2 para DDR	IO_L23N_3	I/O	3
18	P1	SD_A3	Línea de dirección A3 para DDR	IO_L21N_3	I/O	3
19	E4	SD_A4	Línea de dirección A4 para DDR	IO_L04P_3	I/O	3

Tabla B.1. Terminales del FPGA. (Continúa)

#	Terminal FPGA	Nombre	Descripción	Pin	Tipo	Banco
20	H4	SD_A5	Línea de dirección A5 para DDR	IO_L09P_3	I/O	3
21	H3	SD_A6	Línea de dirección A6 para DDR	IO_L09N_3	I/O	3
22	H1	SD_A7	Línea de dirección A7 para DDR	IO_L10N_3	I/O	3
23	H2	SD_A8	Línea de dirección A8 para DDR	IO_L10P_3	I/O	3
24	N4	SD_A9	Línea de dirección A9 para DDR	IO_L20P_3	I/O	3
25	T2	SD_A10	Línea de dirección A10 para DDR	IO_L24P_3	I/O	3
26	N5	SD_A11	Línea de dirección A11 para DDR	IO_L20N_3	I/O	3
27	P2	SD_A12	Línea de dirección A12 para DDR	IO_L21P_3	I/O	3
28	L2	SD_DQ0	Línea de datos DQ0 para DDR	IO_L15N_3	I/O	3
29	L1	SD_DQ1	Línea de datos DQ1 para DDR	IO_L15P_3	I/O	3
30	L3	SD_DQ2	Línea de datos DQ2 para DDR	IO_L16P_3	I/O	3
31	L4	SD_DQ3	Línea de datos DQ3 para DDR	IO_L16N_3	I/O	3
32	M3	SD_DQ4	Línea de datos DQ4 para DDR	IO_L18N_3	I/O	3
33	M4	SD_DQ5	Línea de datos DQ5 para DDR	IO_L18P_3	I/O	3
34	M5	SD_DQ6	Línea de datos DQ6 para DDR	IO_L19P_3	I/O	3
35	M6	SD_DQ7	Línea de datos DQ7 para DDR	IO_L19N_3	I/O	3
36	E2	SD_DQ8	Línea de datos DQ8 para DDR	IO_L03P_3	I/O	3
37	E1	SD_DQ9	Línea de datos DQ9 para DDR	IO_L03N_3	I/O	3
38	F1	SD_DQ10	Línea de datos DQ10 para DDR	IO_L05P_3	I/O	3

Tabla B.1. Terminales del FPGA. (Continúa)

#	Terminal FPGA	Nombre	Descripción	Pin	Tipo	Banco
39	F2	SD_DQ11	Línea de datos DQ11 para DDR	IO_L05N_3	I/O	3
40	G6	SD_DQ12	Línea de datos DQ12 para DDR	IO_L07P_3	I/O	3
41	G5	SD_DQ13	Línea de datos DQ13 para DDR	IO_L07N_3	I/O	3
42	H6	SD_DQ14	Línea de datos DQ14 para DDR	IO_L08P_3	I/O	3
43	H5	SD_DQ15	Línea de datos DQ15 para DDR	IO_L08N_3	I/O	3
44	K6	SD_BA1	Dirección de banco 1	IO_L14P_3/LHCLK6	LHCLK	3
45	K5	SD_BA0	Dirección de banco 2	IO_L14N_3/LHCLK7	LHCLK	3
46	C1	SD_RAS	Línea de comando RAS	IO_L01P_3	I/O	3
47	C2	SD_CAS	Línea de comando CAS	IO_L01N_3	I/O	3
48	D1	SD_WE	Línea de comando WE	IO_L02P_3	I/O	3
49	J4	SD_CK_N	Reloj diferencial negativo	IO_L11N_3/LHCLK1	LHCLK	3
50	J5	SD_CK_P	Reloj diferencial positivo	IO_L11P_3/LHCLK0	LHCLK	3
51	K3	SD_CKE	Línea de habilitación del reloj	IO_L13P_3/LHCLK4/TRDY2	LHCLK	3
52	K4	SD_CS	CS memoria DDR	IO_L13N_3/LHCLK5	LHCLK	3
53	J1	SD_UDM	Datos UDM	IO_L12P_3/LHCLK2	LHCLK	3
54	J2	SD_LDM	Datos LDM	IO_L12N_3/LHCLK3/IRDY2	LHCLK	3
55	G3	SD_UDQS	Dato UDQS	IO_L06P_3	I/O	3
56	L6	SD_LDQS	Data LDQS	IO_L17P_3	I/O	3
57	B9	SD_CK_FB	Realimentación del reloj	IP_L13N_0/GCLK9	GCLK	0

Tabla B.1. Terminales del FPGA. (Continúa)

#	Terminal FPGA	Nombre	Descripción	Pin	Tipo	Banco
58	D4	LED<0>	LED 0	IO	I/O	3
59	C3	LED<1>	LED 1	IO_L25P_0	I/O	0
60	D6	LED<2>	LED 2	IO_L21P_0	I/O	0
61	E6	LED<3>	LED 3	IO_L21N_0	I/O	0
62	D13	LED<4>	LED 4	IO	I/O	0
63	A7	LED<5>	LED 5	IO	I/O	0
64	G9	LED<6>	LED 6	IO	I/O	0
65	A8	LED<7>	LED 7	IO	I/O	0
66	L13	SW<0>	Interruptor deslizable 0	IP	INPUT	1
67	L14	SW<1>	Interruptor deslizable 1	IP	INPUT	1
68	H18	SW<2>	Interruptor deslizable 2	IP/VREF_1	VREF	1
69	N17	SW<3>	Interruptor deslizable 3	IP	INPUT	1
70	D18	BTN_WEST	*Push button* oeste	IP/VREF_1	VREF	1
71	H13	BTN_EAST	*Push button* este	IP	INPUT	1
72	K17	BTN_SOUTH	*Push button* sur	IP	INPUT	1
73	V4	BTN_NORTH	*Push button* norte	IP_L02P_2	INPUT	2
74	K18	ROT_A	Perilla giratoria A	IP	INPUT	1
75	V16	ROT_CENTER	Perilla giratoria centro	IP	INPUT	2
76	G18	ROT_B	Perilla giratoria B	IP	INPUT	1

Tabla B.1. Terminales del FPGA. (Continúa)

#	Terminal FPGA	Nombre	Descripción	Pin	Tipo	Banco
77	L18	LCD_RS	Señal RS del display LCD	IO_L10P_1	I/O	1
78	L17	LCD_RW	Señal RW del display LCD	IO_L10N_1/VREF_1	VREF	1
79	M18	LCD_E	Señal ENABLE del display LCD	IO_L08N_1	I/O	1
80	R15	LCD_DB0	Señal de dato 0 del LCD	IO_L03P_1	I/O	1
81	R16	LCD_DB1	Señal de dato 1 del LCD	IO_L03N_1/VREF_1	VREF	1
82	P17	LCD_DB2	Señal de dato 2 del LCD	IO_L06P_1	I/O	1
83	M15	LCD_DB3	Señal de dato 3 del LCD	IO_L07P_1	I/O	1
84	M16	LCD_DB4	Señal de dato 4 del LCD	IO_L07N_1	I/O	1
85	P6	LCD_DB5	Señal de dato 5 del LCD	IO_L05N_2	I/O	2
86	R8	LCD_DB6	Señal de dato 6 del LCD	IO_L10P_2	I/O	2
87	T8	LCD_DB7	Señal de dato 7 del LCD	IO_L10N_2	I/O	2
88	A3	A3	Terminal de E	IP	INPUT	0
89	C14	C14	Terminal de E/S	IO_L03N_0/VREF_0	VREF	0
90	C15	C15	Terminal de E	IP	INPUT	0
91	A10	A10	Terminal de E/S	IO_L12N_0/GCLK7	GCLK	0
92	A11	A11	Terminal de E/S	IO	I/O	0
93	A12	A12	Terminal de E/S	IO	I/O	0
94	A13	A13	Terminal de E/S	IO_L05P_0	I/O	0
95	A15	A15	Terminal de E/S	IP_L02N_0	INPUT	0

Tabla B.1. Terminales del FPGA. (Continúa)

#	Terminal FPGA	Nombre	Descripción	Pin	Tipo	Banco
96	C11	C11	Terminal de E/S	IO_L09P_0	I/O	0
97	D14	D14	Terminal de E/S	IO_L03P_0	I/O	0
98	B6	B6	Terminal de E/S	IO_L20P_0	I/O	0
99	E15	E15	Terminal de E/S	IO_L22P_1	I/O	1
100	B10	B10	Terminal de E/S	IO_L12P_0/GCLK6	GCLK	0
101	B13	B13	Terminal de E/S	IO_L05N_0/VREF_0	VREF	0
102	B15	B15	Terminal de E	IP_L02P_0	INPUT	0
103	D16	D16	Terminal de E/S	IO_L23N_1/LDC0	DUAL	1
104	E16	E16	Terminal de E/S	IO_L22N_1	I/O	1
105	D17	D17	Terminal de E/S	IO_L23P_1/HDC	DUAL	1
106	E17	E17	Terminal de E	IP	INPUT	1
107	B18	B18	Terminal de E	IP	INPUT	1
108	C17	C17	Terminal de E/S	IO_L24N_1/LDC2	DUAL	1
109	C18	C18	Terminal de E/S	IO_L24P_1/LDC1	DUAL	1
110	F18	F18	Terminal de E/S	IO_L19P_1	I/O	1
111	H17	H17	Terminal de E/S	IO_L16N_1/A0	DUAL	1
112	N18	N18	Terminal de E/S	IO_L08P_1	I/O	1
113	R18	R18	Terminal de E/S	IO_L02P_1/A14	DUAL	1
114	F17	F17	Terminal de E/S	IO_L19N_1	I/O	1

Tabla B.1. Terminales del FPGA. (Continúa)

#	Terminal FPGA	Nombre	Descripción	Pin	Tipo	Banco
115	T18	T18	Terminal de E/S	IO_L02N_1/A13	DUAL	1
116	T17	T17	Terminal de E/S	IO_L01N_1/A15	DUAL	1
117	U18	U18	Terminal de E/S	IO_L01P_1/A16	DUAL	1
118	T16	T16	Terminal de E/S	IO_L26P_2/VS0/A17	DUAL	2
119	T15	T15	Terminal de E/S	IO/VREF_2	VREF	2
120	T14	T14	Terminal de E/S	IO_L24P_2/A21	DUAL	2
121	V15	V15	Terminal de E/S	IO_L25P_2/VS2/A19	DUAL	2
122	V14	V14	Terminal de E	IP_L23P_2	INPUT	2
123	V13	V13	Terminal de E/S	IO_L19N_2/VREF_2	VREF	2
124	V9	V9	Terminal de E/S	IO_L13N_2/D3/GCLK15	DUAL/GCLK	2
125	V3	V3	Terminal de E	IP_L02N_2	INPUT	2
126	V2	V2	Terminal de E	IP	INPUT	2
127	V5	V5	Terminal de E/S	IO_L06P_2	I/O	2
128	B3	HSWAP	Control de las resistencias de PULLUP	IO_L25N_0/HSWAP	DUAL	0
129	T3	INIT_B	Terminal INIT_B	IO_L01N_2/INIT_B	DUAL	2
130	V11	M1	Modo de configuración 1	IO/M1	DUAL	2
131	M10	M0	Modo de configuración 0	IO_L16P_2/M0	DUAL	2
132	T10	M2	Modo de configuración 2	IP_L14N_2/M2/GCLK1	DUAL/GCLK	2

Tabla B.1. Terminales del FPGA. (Continúa)

#	Terminal FPGA	Nombre	Descripción	Pin	Tipo	Banco
133	V17	DONE	Fin de configuración	DONE	CONFIG	VCCAUX
134	B1	PROG_B	Borrar configuración	PROG_B	CONFIG	VCCAUX
135	A2	TDI	Test Data In para JTAG	TDI	JTAG	VCCAUX
136	C16	TDO	Test Data Out para JTAG	TDO	JTAG	VCCAUX
137	A17	TCK	Test Clock para JTAG	TCK	JTAG	VCCAUX
138	D15	TMS	Test Mode para JTAG	TMS	JTAG	VCCAUX
139	D2	VCC1V25	Voltaje de referencia banco 3	IO_L02N_3/VREF_3	VREF	3
140	G4	VCC1V25	Voltaje de referencia banco 3	IO_L06N_3/VREF_3	VREF	3
141	L5	VCC1V25	Voltaje de referencia banco 3	IO_L17N_3/VREF_3	VREF	3
142	J6	VCC1V25	Voltaje de referencia banco 3	IP/VREF_3	VREF	3
143	R4	VCC1V25	Voltaje de referencia banco 3	IP/VREF_3	VREF	3
144	T13	VCC3V3	Voltaje para banco 2 de E/S	VCCO_2	VCCO	2
145	T6	VCC3V3	Voltaje para banco 2 de E/S	VCCO_2	VCCO	2
146	M11	VCC3V3	Voltaje para banco 2 de E/S	VCCO_2	VCCO	2
147	V10	VCC3V3	Voltaje para banco 2 de E/S	VCCO_2	VCCO	2
148	M8	VCC3V3	Voltaje para banco 2 de E/S	VCCO_2	VCCO	2
149	N3	VCC2V5-DDR	Voltaje para banco 3 de E/S	VCCO_3	VCCO	3
150	L7	VCC2V5-DDR	Voltaje para banco 3 de E/S	VCCO_3	VCCO	3
151	H7	VCC2V5-DDR	Voltaje para banco 3 de E/S	VCCO_3	VCCO	3

Tabla B.1. Terminales del FPGA. (Continúa)

#	Terminal FPGA	Nombre	Descripción	Pin	Tipo	Banco
152	F3	VCC2V5-DDR	Voltaje para banco 3 de E/S	VCCO_3	VCCO	3
153	K1	VCC2V5-DDR	Voltaje para banco 3 de E/S	VCCO_3	VCCO	3
154	L12	VCC3V3	Voltaje para banco 1 de E/S	VCCO_1	VCCO	1
155	N16	VCC3V3	Voltaje para banco 1 de E/S	VCCO_1	VCCO	1
156	F16	VCC3V3	Voltaje para banco 1 de E/S	VCCO_1	VCCO	1
157	H12	VCC3V3	Voltaje para banco 1 de E/S	VCCO_1	VCCO	1
158	J18	VCC3V3	Voltaje para banco 1 de E/S	VCCO_1	VCCO	1
159	A9	VCC-BANK0	Voltaje para banco 0 de E/S	VCCO_0	VCCO	0
160	G8	VCC-BANK0	Voltaje para banco 0 de E/S	VCCO_0	VCCO	0
161	C13	VCC-BANK0	Voltaje para banco 0 de E/S	VCCO_0	VCCO	0
162	C6	VCC-BANK0	Voltaje para banco 0 de E/S	VCCO_0	VCCO	0
163	G11	VCC-BANK0	Voltaje para banco 0 de E/S	VCCO_0	VCCO	0
164	P14	VCC1V2	Voltaje interno	VCCINT	VCCINT	VCCINT
165	N6	VCC1V2	Voltaje interno	VCCINT	VCCINT	VCCINT
166	F13	VCC1V2	Voltaje interno	VCCINT	VCCINT	VCCINT
167	E14	VCC1V2	Voltaje interno	VCCINT	VCCINT	VCCINT
168	F6	VCC1V2	Voltaje interno	VCCINT	VCCINT	VCCINT
169	E5	VCC1V2	Voltaje interno	VCCINT	VCCINT	VCCINT
170	N13	VCC1V2	Voltaje interno	VCCINT	VCCINT	VCCINT

Tabla B.1. Terminales del FPGA. (Continúa)

#	Terminal FPGA	Nombre	Descripción	Pin	Tipo	Banco
171	P5	VCC1V2	Voltaje interno	VCCINT	VCCINT	VCCINT
172	B7	VCC2V5	Voltaje auxiliar	VCCAUX	VCCAUX	VCCAUX
173	B12	VCC2V5	Voltaje auxiliar	VCCAUX	VCCAUX	VCCAUX
174	G2	VCC2V5	Voltaje auxiliar	VCCAUX	VCCAUX	VCCAUX
175	U12	VCC2V5	Voltaje auxiliar	VCCAUX	VCCAUX	VCCAUX
176	M17	VCC2V5	Voltaje auxiliar	VCCAUX	VCCAUX	VCCAUX
177	U7	VCC2V5	Voltaje auxiliar	VCCAUX	VCCAUX	VCCAUX
178	M2	VCC2V5	Voltaje auxiliar	VCCAUX	VCCAUX	VCCAUX
179	G17	VCC2V5	Voltaje auxiliar	VCCAUX	VCCAUX	VCCAUX
180	H10	GND	Tierra (GND)	GND	GND	GND
181	K11	GND	Tierra (GND)	GND	GND	GND
182	L11	GND	Tierra (GND)	GND	GND	GND
183	K16	GND	Tierra (GND)	GND	GND	GND
184	A1	GND	Tierra (GND)	GND	GND	GND
185	K8	GND	Tierra (GND)	GND	GND	GND
186	B2	GND	Tierra (GND)	GND	GND	GND
187	L9	GND	Tierra (GND)	GND	GND	GND
188	A18	GND	Tierra (GND)	GND	GND	GND
189	M12	GND	Tierra (GND)	GND	GND	GND

Tabla B.1. Terminales del FPGA. (Continúa)

#	Terminal FPGA	Nombre	Descripción	Pin	Tipo	Banco
190	B17	GND	Tierra (GND)	GND	GND	GND
191	H9	GND	Tierra (GND)	GND	GND	GND
192	V1	GND	Tierra (GND)	GND	GND	GND
193	J8	GND	Tierra (GND)	GND	GND	GND
194	U2	GND	Tierra (GND)	GND	GND	GND
195	J11	GND	Tierra (GND)	GND	GND	GND
196	V18	GND	Tierra (GND)	GND	GND	GND
197	L8	GND	Tierra (GND)	GND	GND	GND
198	U17	GND	Tierra (GND)	GND	GND	GND
199	L10	GND	Tierra (GND)	GND	GND	GND
200	C10	GND	Tierra (GND)	GND	GND	GND
201	M7	GND	Tierra (GND)	GND	GND	GND
202	G7	GND	Tierra (GND)	GND	GND	GND
203	J3	GND	Tierra (GND)	GND	GND	GND
204	G12	GND	Tierra (GND)	GND	GND	GND
205	T9	GND	Tierra (GND)	GND	GND	GND
206	H8	GND	Tierra (GND)	GND	GND	GND
207	H11	GND	Tierra (GND)	GND	GND	GND
208	A4	-	No conectado	IO_L24P_0	I/O	0

Tabla B.1. Terminales del FPGA. (Continúa)

#	Terminal FPGA	Nombre	Descripción	Pin	Tipo	Banco
209	A5	-	No conectado	IP_L22P_0	INPUT	0
210	A6	-	No conectado	IO_L20N_0	I/O	0
211	A14	-	No conectado	IO_L04N_0	I/O	0
212	A16	-	No conectado	IO_L01N_0	I/O	0
213	B4	-	No conectado	IO_L24N_0	I/O	0
214	B5	-	No conectado	IP_L22N_0	INPUT	0
215	B8	-	No conectado	IP_L13P_0/GCLK8	GCLK	0
216	B11	-	No conectado	IO/VREF_0	VREF	0
217	B14	-	No conectado	IO_L04P_0	I/O	0
218	B16	-	No conectado	IO_L01P_0	I/O	0
219	C4	-	No conectado	IO	I/O	0
220	C5	-	No conectado	IO_L23P_0	I/O	0
221	C7	-	No conectado	IO_L18P_0	I/O	0
222	C8	-	No conectado	IP_L16P_0	INPUT	0
223	C12	-	No conectado	IP_L07P_0	INPUT	0
224	D3	-	No conectado	IP	INPUT	3
225	D5	-	No conectado	IO_L23N_0/VREF_0	VREF	0
226	D7	-	No conectado	IO_L18N_0/VREF_0	VREF	0
227	D8		No conectado	IP_L16N_0	INPUT	0

Tabla B.1. Terminales del FPGA. (Continúa)

#	Terminal FPGA	Nombre	Descripción	Pin	Tipo	Banco
228	D9	-	No conectado	IO_L14N_0/GCLK11	GCLK	0
229	D10	-	No conectado	IO_L11P_0/GCLK4	GCLK	0
230	D11	-	No conectado	IO_L09N_0	I/O	0
231	D12	-	No conectado	IP_L07N_0	INPUT	0
232	E3	-	No conectado	IO_L04N_3	I/O	3
233	E7	-	No conectado	IO_L19N_0/VREF_0	VREF	0
234	E8	-	No conectado	IO_L17P_0	I/O	0
235	E9	-	No conectado	IO_L15P_0	I/O	0
236	E10	-	No conectado	IO_L11N_0/GCLK5	GCLK	0
237	E11	-	No conectado	IO_L08P_0	I/O	0
238	E12	-	No conectado	IO_L06N_0	I/O	0
239	E13	-	No conectado	IO	I/O	0
240	F4	-	No conectado	IP	INPUT	3
241	F5	-	No conectado	IP	INPUT	3
242	F7	-	No conectado	IO_L19P_0	I/O	0
243	F8	-	No conectado	IO_L17N_0	I/O	0
244	F9	-	No conectado	IO_L15N_0	I/O	0
245	F10	-	No conectado	IP_L10P_0	INPUT	0
246	F11	-	No conectado	IO_L08N_0	I/O	0

Tabla B.1. Terminales del FPGA. (Continúa)

#	Terminal FPGA	Nombre	Descripción	Pin	Tipo	Banco
247	F12	-	No conectado	IO_L06P_0	I/O	0
248	F14	-	No conectado	IO_L21N_1	I/O	1
249	F15	-	No conectado	IO_L21P_1	I/O	1
250	G1	-	No conectado	IP	INPUT	3
251	G10	-	No conectado	IP_L10N_0	INPUT	0
252	G13	-	No conectado	IO_L20N_1	I/O	1
253	G14	-	No conectado	IO_L20P_1	I/O	1
254	G15	-	No conectado	IO_L18P_1	I/O	1
255	G16	-	No conectado	IO_L18N_1	I/O	1
256	H14	-	No conectado	IO_L17P_1	I/O	1
257	H15	-	No conectado	IO_L17N_1	I/O	1
258	H16	-	No conectado	IO_L16P_1	I/O	1
259	J7	-	No conectado	IP	INPUT	3
260	J12	-	No conectado	IO_L15P_1/A2	DUAL	1
261	J13	-	No conectado	IO_L15N_1/A1	DUAL	1
262	J14	-	No conectado	IO_L14N_1/A3/RHCLK7	RHCLK/DUAL	1
263	J15	-	No conectado	IO_L14P_1/A4/RHCLK6	RHCLK/DUAL	1
264	J16	-	No conectado	IO_L13N_1/A5/RHCLK5	RHCLK/DUAL	1
265	J17	-	No conectado	IO_L13P_1/A6/RHCLK4/IRDY1	RHCLK/DUAL	1

Tabla B.1. Terminales del FPGA. (Continúa)

#	Terminal FPGA	Nombre	Descripción	Pin	Tipo	Banco
266	K2	-	No conectado	IP	INPUT	3
267	K7	-	No conectado	IP	INPUT	3
268	K12	-	No conectado	IO_L11N_1/A9/RHCLK1	RHCLK/DUAL	1
269	K13	-	No conectado	IO_L11P_1/A10/RHCLK0	RHCLK/DUAL	1
270	K14	-	No conectado	IO_L12N_1/A7/RHCLK3/TRDY1	RHCLK/DUAL	1
271	K15	-	No conectado	IO_L12P_1/A8/RHCLK2	RHCLK/DUAL	1
272	L15	-	No conectado	IO_L09N_1/A11	DUAL	1
273	L16	-	No conectado	IO_L09P_1/A12	DUAL	1
274	M1	-	No conectado	IP	INPUT	3
275	M9	-	No conectado	IO_L12N_2/D6/GCLK13	DUAL/GCLK	2
276	N1	-	No conectado	IP	INPUT	3
277	N2	-	No conectado	IP	INPUT	3
278	N9	-	No conectado	IO_L12P_2/D7/GCLK12	DUAL/GCLK	2
279	N11	-	No conectado	IO_L18N_2	I/O	2
280	N12	-	No conectado	IO_L21P_2	I/O	2
281	N14	-	No conectado	IO_L04N_1	I/O	1
282	N15	-	No conectado	IO_L04P_1	I/O	1
283	P3	-	No conectado	IO_L22P_3	I/O	3
284	P4	-	No conectado	IO_L22N_3	I/O	3

Tabla B.1. Terminales del FPGA. (Continúa)

#	Terminal FPGA	Nombre	Descripción	Pin	Tipo	Banco
285	P9	-	No conectado	IO	I/O	2
286	P10	-	No conectado	IO_L15N_2/D1/GCLK3	DUAL/GCLK	2
287	P12	-	No conectado	IO_L21N_2	I/O	2
288	P13	-	No conectado	IO_L22P_2/A23	DUAL	2
289	P15	-	No conectado	IP	INPUT	1
290	P16	-	No conectado	IO	I/O	1
291	P18	-	No conectado	IO_L06N_1	I/O	1
292	R1	-	No conectado	IP	INPUT	3
293	R5	-	No conectado	IO_L04P_2	I/O	2
294	R6	-	No conectado	IO_L05P_2	I/O	2
295	R9	-	No conectado	IO/D5	DUAL	2
296	R10	-	No conectado	IO_L15P_2/D2/GCLK2	DUAL/GCLK	2
297	R11	-	No conectado	IO	I/O	2
298	R12	-	No conectado	IO_L20N_2	I/O	2
299	R13	-	No conectado	IO_L22N_2/A22	DUAL	2
300	R14	-	No conectado	IO_L24N_2/A20	DUAL	2
301	R17	-	No conectado	IP	INPUT	1
302	T5	-	No conectado	IO_L04N_2	I/O	2
303	T7	-	No conectado	IP_L08P_2	INPUT	2

Tabla B.1. Terminales del FPGA. (Continúa)

#	Terminal FPGA	Nombre	Descripción	Pin	Tipo	Banco
304	T11	-	No conectado	IP_L17P_2	INPUT	2
305	T12	-	No conectado	IO_L20P_2	I/O	2
306	U1	-	No conectado	IP	INPUT	3
307	U3	-	No conectado	IO_L01P_2/CSO_B	DUAL	2
308	U4	-	No conectado	IO_L03P_2/DOUT/BUSY	DUAL	2
309	U5	-	No conectado	IO/VREF_2	VREF	2
310	U6	-	No conectado	IO	I/O	2
311	U9	-	No conectado	IO_L13P_2/D4/GCLK14	DUAL/GCLK	2
312	U10	-	No conectado	IP_L14P_2/RDWR_B/GCLK	DUAL/GCLK	2
313	U11	-	No conectado	IP_L17N_2	INPUT	2
314	U13	-	No conectado	IO	I/O	2
315	U14	-	No conectado	IP_L23N_2	INPUT	2
316	U15	-	No conectado	IO_L25N_2/VS1/A18	DUAL	2
317	V6	-	No conectado	IO_L06N_2/VREF_2	VREF	2
318	V7	-	No conectado	IO	I/O	2
319	V8	-	No conectado	IP_L11N_2/VREF_2	VREF	2
320	V12	-	No conectado	IO_L19P_2	I/O	2

Tabla B.1. Terminales del FPGA.

Apéndice C: Software desarrollado

En este apéndice se muestra el software desarrollado del sistema de monitoreo y control para el procesador MicroBlaze.

```
/*
 * Aplicación final: PRIUS.c
 * Author: Héctor Alonso Valera Martínez
 */
//Bibliotecas utilizadas
#include <stdio.h>
#include "xparameters.h"
#include "xspi.h"
#include "xspi_l.h"
#include "xgpio.h"
#include "xuartlite_l.h"
#include "xtmrctr.h"
//Identificadores de IP
#define ID_leds XPAR_LEDS_6BIT_DEVICE_ID
#define ID_un_led XPAR_LEDS_1BIT_DEVICE_ID
#define ID_botones XPAR_DIP_SWITCHES_4BIT_DEVICE_ID
#define ID_DAC XPAR_XPS_SPI_0_DEVICE_ID
#define ID_otros XPAR_OTROS_DISPOS_DEVICE_ID
#define ID_spi XPAR_XPS_SPI_0_DEVICE_ID
#define ID_lcd XPAR_CHARACTER_LCD_2X16_DEVICE_ID
#define TMRCTR_DEVICE_ID  XPAR_XPS_TIMER_0_DEVICE_ID
#define TIMER_COUNTER_0 0
#define dir_timer XPAR_XPS_TIMER_0_BASEADDR
#define DCE_dir XPAR_RS232_DCE_BASEADDR
#define DTE_dir XPAR_RS232_DTE_BASEADDR
#define ganancia 0x00000011
#define muestras 10000
//Instancias de cada IP
static XTmrCtr Timer;
static XGpio lcd;
static XSpi  spi;
static XGpio leds;
static XGpio un_led;
static XGpio botones;
static XGpio otros;
//Variables globales
u32 opciones;
u8 buffer[3];
//Funciones para llamar
void menu();
int Configurar_Gpio();
int Inicializa_SPI(XSpi *SPI, u32 ID, u32 opciones);
long Transmite_SPI(XSpi *SPI, u32 enviar, u32 esclavo, u32 opciones);
void Funcion_ADC();
void Funcion_DAC();
void Funcion_DDRAM();
void Funcion_IO();
void Modo_Prueba();
s16 procesar_datos_canal_1(u32 dato);
```

```
s16 procesar_datos_canal_2(u32 dato);
int milivolts(int dato);
int ConfiguraLcd();
void InicializaLcd();
void EscribeEnLcd(char *c);
void EnviarComando(u8 dato);
void Linea(int linea);
void LimpiarLcd();
void EnviarDato(char c);
void Posicion(int linea,u8 cuadrito);
void EscribeLinea(char *c);
void retardo_ms(int a);
u32 dato_dac(u8 buffer1, u8 buffer2, float ref);
//Función principal
int main(){
 int leer;
 ConfiguraLcd();
 InicializaLcd();
 EscribeEnLcd("   II - UNAM  ");
 Configurar_Gpio();
 //Se configura el SPI para el amplificador
 opciones=XSP_CR_MASTER_MODE_MASK | XSP_CR_ENABLE_MASK
 |XSP_CR_TRANS_INHIBIT_MASK;
 Inicializa_SPI(&spi, ID_spi, opciones);
 //Se envia la ganancia al amplificador
 Transmite_SPI(&spi,ganancia,0x00000005,opciones);
 while(1){                                  //Parte principal del programa
      XGpio_DiscreteClear(&un_led,1,0x01);
      menu();
      Linea(2);
      EscribeLinea("Esperando opcion");
      leer=XUartLite_RecvByte(DCE_dir);   //Se lee la opción seleccionada
      xil_printf("%c\r\n",leer);
      XGpio_DiscreteSet(&un_led,1,0x01);
      Linea(2);
      EscribeLinea("                ");
      switch(leer){                 //Se ejecuta la opción seleccionada
            case '1':xil_printf("Opcion ADC\r\n");
                     Linea(2);
                     EscribeLinea("  ADC ...");
                     Funcion_ADC();
                     break;
            case '2':Linea(2);
                     EscribeLinea("    DAC: ");
                     Funcion_DAC();
                     break;
            case '3':Linea(2);
                     EscribeLinea("Graficando C:");
                     Funcion_DDRAM();
                     break;
            case '4':Linea(2);
                     EscribeLinea(" LEDs ...");
                     Funcion_IO();
                     break;
            case '5':Linea(2);
                     EscribeLinea(" Modo Prueba ...");
                     xil_printf("Opcion MODO PRUEBA\r\n");
```

```
                    Modo_Prueba();
                    break;
            default: xil_printf("Opcion no valida\r\n");
                    break;
                       }
      Linea(2);
      EscribeEnLcd("                ");
            }
      return XST_SUCCESS;
                  }
void menu(){                                  //Menú mostrado en hyperterminal
 xil_printf("\r\n---Menu--\r\n");
 xil_printf("1)ADC\r\n2)DAC\r\n3)GRAFICAR\n\r4)I/O\r\n5)MODO
                 PRUEBA\r\n");
 xil_printf("Seleccione opcion: ");
            }
int Configurar_Gpio(){
 XGpio_Initialize(&leds, ID_leds);        //Se inicializan los LEDs
 XGpio_Initialize(&un_led, ID_un_led);    //Se inicializa el LED
 //Se inicializan los interruptores deslizables
 XGpio_Initialize(&botones, ID_botones );
 XGpio_Initialize(&otros, ID_otros );     //Se inicilizan otros dispos
 XGpio_SetDataDirection(&leds,1,0x00);    //LEDs como salida
 XGpio_SetDataDirection(&un_led,1,0x00); //LED como salida
 XGpio_SetDataDirection(&botones,1,0xF); //Botones como entrada
 XGpio_SetDataDirection(&otros,1,0x00);  //Otros dispos como salidas
 /*Desactivar otros componentes
  * [DAC_CLR   SHDN_Amp   ADC_conv  init_B  SF_CE0  SS_B  0 0]
  *     1         0          0        1        1      1   0 0
  * */
 //Desactivar otros componentes y habilitar amplificador y DAC
 XGpio_DiscreteSet(&otros,1,0x9C);
 return XST_SUCCESS;
                       }
int Inicializa_SPI(XSpi *SPI, u32 ID, u32 opciones){
 XSpi_Initialize(SPI, ID);          //Se inicializa el SPI
 XSpi_Start(SPI);                   //Se habilita el SPI
 XSpi_IntrGlobalDisable(SPI);       //Se desactivan las interrupciones
 //Se configura el módulo SPI con "opciones"
 XSpi_SetControlReg(SPI,opciones );
      return XST_SUCCESS;
                                                    }
long Transmite_SPI(XSpi *SPI, u32 enviar, u32 esclavo, u32 opciones){
 long recibir;
 int lee;
 //Escribir los datos a enviar, el Maestro está deshabilitado
 XSpi_WriteReg(XPAR_XPS_SPI_0_BASEADDR,XSP_DTR_OFFSET,enviar);
 XSpi_WriteReg(XPAR_XPS_SPI_0_BASEADDR,XSP_SSR_OFFSET,esclavo);
 XSpi_Enable(SPI);
      do{                                   //Se espera la transferencia
         lee=XSpi_IntrGetStatus(SPI);
         } while(lee != 0x14);
 XSpi_IntrClear(SPI,XSP_INTR_TX_EMPTY_MASK|XSP_INTR_RX_FULL_MASK);
 XSpi_SetControlReg(SPI,opciones );            //Inhibir la transmisión
 //Se lee el dato recibido
 recibir=XSpi_ReadReg(XPAR_XPS_SPI_0_BASEADDR,XSP_DRR_OFFSET );
      return recibir;
```

```
    }
void Funcion_ADC(){
 int lectura_ADC,k;
 s16 dato_c1,dato_c2;
 u16 *MemRAM;
 //Se apunta a la memoria DDR
 MemRAM = XPAR_DDR_SDRAM_MPMC_BASEADDR;
 //Se configura el SPI para el ADC
 opciones=XSP_CR_MASTER_MODE_MASK |
      XSP_CR_ENABLE_MASK|XSP_CR_TRANS_INHIBIT_MASK|XSP_CR_CLK_PHASE_MASK;
 XSpi_SetControlReg(&spi,opciones );
 for(k=0;k<muestras;k++){
      XGpio_DiscreteSet(&otros,1,0xBC);              //Activa el ADC
      XGpio_DiscreteSet(&otros,1,0x9C);
      //Lee los datos del ADC
      lectura_ADC=Transmite_SPI(&spi,0x00000000,0x00000006,opciones);
      //Ajusta los datos a 16 bits
      dato_c1=procesar_datos_canal_1(lectura_ADC);
      dato_c2=procesar_datos_canal_2(lectura_ADC);
      MemRAM[2*k]=dato_c1;          //Almacena los datos en la memoria DDR
      MemRAM[(2*k)+1]=dato_c2;
                        }
                   }
void Funcion_DAC(){
 int dir,i;
 u32 Escritura_DAC,dir_h,dato_h;
 xil_printf("\r\nDAC: 1)A 2)B 3)C 4)D\n\r");
 dir=XUartLite_RecvByte(DCE_dir);         //Captura la opción seleccionada
 if((dir>=49)&&(dir<=52)){                //Se seleccionan solo números
      Posicion(2,9);
      EnviarDato(dir+16);
      xil_printf("\r\nDAC: %c",dir+16);
      xil_printf("\r\nValor de voltaje: ");
      for(i=0;i<3;i++){       //Se captura el valor de voltaje ingresado
            buffer[i]=XUartLite_RecvByte(DCE_dir);
            xil_printf("%c",buffer[i]);
                        }
            switch(dir){                        //Se selecciona el DAC
                  case '1': dir_h=0x00000000;
                            dato_h=dato_dac(buffer[0], buffer[2],3.3);
                            break;
                  case '2': dir_h=0x00010000;
                            dato_h=dato_dac(buffer[0], buffer[2],3.3);
                            break;
                  case '3': dir_h=0x00020000;
                            dato_h=dato_dac(buffer[0], buffer[2],3.3);
                            break;
                  case '4': dir_h=0x00030000;
                            dato_h=dato_dac(buffer[0], buffer[2],3.3);
                            break;
                  default:  dir_h=0x00000000;
                            break;
                        }
             //Se construye el dato de 32 bits para el DAC
             Escritura_DAC=0x00300000|dir_h|dato_h;
             //Se configura el SPI para el DAC
             opciones=XSP_CR_MASTER_MODE_MASK  |
```

```
            XSP_CR_ENABLE_MASK|XSP_CR_TRANS_INHIBIT_MASK;
            XSpi_SetControlReg(&spi,opciones );
            Transmite_SPI(&spi, Escritura_DAC,0x00000003,opciones);
                        }
 else{
      xil_printf("No valido\r\n");
      }
                  }
void Funcion_DDRAM(){
 int leer,k,c_mV;
 s16 dato_lectura;
 u16 enviar;
 u16 *MemRAM;
 MemRAM = XPAR_DDR_SDRAM_MPMC_BASEADDR;  //Se apunta a la memoria DDR
 xil_printf("1) Canal 1 2) Canal 2\n\r");
 leer=XUartLite_RecvByte(DCE_dir);       //Se lee el canal seleccionado
 Posicion(2,14);
 EnviarDato(leer);
 switch(leer){
      case '1': xil_printf("C: %c\n\r",leer);
                for(k=0;k<muestras;k++){
                //Se lee el canal 1 de la memoria DDR
                dato_lectura=MemRAM[2*k];
                c_mV=milivolts(dato_lectura); //Se convierte en milivolts
                enviar=0x00FF&c_mV; //Se envía la parte baja del dato
                XUartLite_SendByte(DTE_dir,enviar);
                enviar=0xFF00&c_mV; //Se envía la parte alta del dato
                enviar>>=8;
                XUartLite_SendByte(DTE_dir,enviar);
                                    }
                break;
      case '2': xil_printf("C: %c\n\r",leer);
                for(k=0;k<muestras;k++){
                //Se lee el canal 2 de la memoria DDR
                dato_lectura=MemRAM[(2*k)+1];
                c_mV=milivolts(dato_lectura); //Se convierte en milivolts
                enviar=0x00FF&c_mV; //Se envía la parte baja del dato
                XUartLite_SendByte(DTE_dir,enviar);
                enviar=0xFF00&c_mV; //Se envía la parte alta del dato
                enviar>>=8;
                XUartLite_SendByte(DTE_dir,enviar);
                                    }
                break;
      default: xil_printf("No valido\r\n");
               break;
               }
                    }
void Funcion_IO(){
 int leer,dato1,dato2,i,interruptor;
 xil_printf("1)I 2)O\r\n");
 leer=XUartLite_RecvByte(DCE_dir);       //Se lee la opción seleccionada
 switch(leer){
      case '1': //Se lee el valor de los interruptores
                interruptor=XGpio_DiscreteRead(&botones,1);
                //Se despliega en hyperterminal
                xil_printf("Entradas: %d",interruptor);
                break;
```

```
      case '2': xil_printf("\r\nIngrese valor para leds: ");
                //Se captura el primer valor
                dato1=XUartLite_RecvByte(DCE_dir);
                xil_printf("%c",dato1); //Se despliega en hyperterminal
                //Se captura el segundo valor
                dato2=XUartLite_RecvByte(DCE_dir);
                //Se despliega en hyperterminal
                xil_printf("%c\n\r",dato2);
                //Se comprueban que sean números
                if((dato1>=0x30) && (dato1<=0x39)){
                        if((dato2>=0x30) && (dato2<=0x39)){
                                //Se envían a los LEDs
                                i=10*(dato1&0x0F)+(dato2&0x0F);
                                XGpio_DiscreteSet(&leds,1,i);
                                                        }
                          else{
                               xil_printf("\r\nNo valido\n\r");
                               }
                                                    }
                        else{
                             xil_printf("\r\nNo valido\n\r");
                          }
                       break;
      default: xil_printf("No valido\r\n");
               break;
               }
                    }
void Modo_Prueba(){
 u16 *MemRAM;
 u16 dato_lectura;
 int k;
 u32 dato_dac;
 MemRAM = XPAR_DDR_SDRAM_MPMC_BASEADDR;  //Se apunta a la memoria DDR
 Funcion_ADC();               //Se manda llamar a la función para el ADC
 //Se configura el SPI para el DAC
opciones=XSP_CR_MASTER_MODE_MASK|XSP_CR_ENABLE_MASK|XSP_CR_TRANS_INHIBIT_
MASK;
 for(k=0;k<muestras;k++){
      dato_lectura=MemRAM[2*k];          //Se lee el canal 1
      dato_lectura <<= 3;                //Se ajusta a 12 bits
      //Se construye el dato de 32 bits para los 4 DACs
      dato_dac=0x003F0000|dato_lectura;
      Transmite_SPI(&spi, dato_dac,0x00000003,opciones);
                        }
 for(k=0;k<muestras;k++){
      dato_lectura=MemRAM[(2*k)+1];      //Se lee el canal 2
      dato_lectura <<= 3;                //Se ajusta a 12 bits
      //Se construye el dato de 32 bits para los 4 DACs
      dato_dac=0x003F0000|dato_lectura;
      Transmite_SPI(&spi, dato_dac,0x00000003,opciones);
                        }
 //Se limpian las salidas de los 4 DACs
 XGpio_DiscreteClear(&otros,1,0x80);
 XGpio_DiscreteSet(&otros,1,0x9C);
                    }
s16 procesar_datos_canal_1(u32 dato){
 s16 canal_1=0x0000;
```

```
 u32 canal_1_aux=0x0000;
 canal_1_aux=0x3FFF0000 & dato;          //Se selecciona el canal 1
 canal_1 =canal_1_aux>>16;
 if(canal_1 & 0x2000){                   //Se detecta el signo
       canal_1=canal_1 | 0xE000;
                      }
         return canal_1;
                                    }
s16 procesar_datos_canal_2(u32 dato){
 s16 canal_2=0x0000;
 canal_2=0x00003FFF & dato;              //Se selecciona el canal 1
 if(canal_2 & 0x2000){                   //Se detecta el signo
      canal_2=canal_2 | 0xE000;
                      }
 return canal_2;
                                    }
int milivolts(int dato){
 int mV;
 mV=(1650+((dato*1250)/(8192*(-1)))); //El dato se convierte a voltaje
 return mV;
                         }
u32 dato_dac(u8 buffer1, u8 buffer2, float ref){
 float d_h_f;
 u32 d_h;
 //El dato ingresado por la UART se convierte a decimal
 d_h_f=( 4096 * ( (buffer1-48) + 0.1*(buffer2-48) ) )/ref;
 d_h=(int)(d_h_f);
 d_h<<=4;
 return d_h;
                                                 }
int ConfiguraLcd(){
 /* El registro para el LCD es:
  * [ E D/I R/W DB7 BD6 DB5 DB4 ]
  *  MSB                   LSB
  */
 int Status;
 //Se comprueba que el GPIO esté correcto
 Status = XGpio_Initialize(&lcd, ID_lcd);
 if (Status != XST_SUCCESS) {
      return XST_FAILURE;
                            }
 XGpio_SetDataDirection(&lcd,1,0x00); //XPS GPIO como salida
 //Se inicializa el XPS Timer/Counter
 Status = XTmrCtr_Initialize(&Timer, TMRCTR_DEVICE_ID);
 if (Status != XST_SUCCESS){
      return XST_FAILURE;
                           }
 //Se comprueba el funcionamiento del XPS Timer/Counter
 Status = XTmrCtr_SelfTest(&Timer, TIMER_COUNTER_0);
 if (Status != XST_SUCCESS){
      return XST_FAILURE;
                           }
 //Se configura el XPS Timer/Counter
 XTmrCtr_SetOptions(&Timer, TIMER_COUNTER_0, XTC_EXT_COMPARE_OPTION|
 XTC_DOWN_COUNT_OPTION);
 return XST_SUCCESS;
                    }
```

```
void retardo_ms (int a){
 //Se coloca el valor del retardo en milisegundos en el contador
 XTmrCtr_SetResetValue(&Timer, TIMER_COUNTER_0,a*49998);
 XTmrCtr_Start(&Timer, TIMER_COUNTER_0); //Se inicia el contador
 //Se espera a que termine la cuenta
 while(!(XTmrCtr_IsExpired(&Timer, TIMER_COUNTER_0))){}
 XTmrCtr_Stop(&Timer,TIMER_COUNTER_0); //Se detiene el contador
                        }
void InicializaLcd(){
 //Se inicializa el LCD con los parámetros requeridos
 XGpio_DiscreteClear(&lcd,1,0xFF);
 retardo_ms(15);
 XGpio_DiscreteSet(&lcd,1,0x43);    //Se envía el dato 0x03
 XGpio_DiscreteClear(&lcd,1,0x43);
 retardo_ms(5);
 XGpio_DiscreteSet(&lcd,1,0x43);    //Se envía el dato 0x03
 XGpio_DiscreteClear(&lcd,1,0x43);
 retardo_ms(1);
 XGpio_DiscreteSet(&lcd,1,0x43);    //Se envía el dato 0x03
 XGpio_DiscreteClear(&lcd,1,0x43);
 retardo_ms(1);
 XGpio_DiscreteSet(&lcd,1,0x42);    //Se envía el dato 0x02
 XGpio_DiscreteClear(&lcd,1,0x42);
 retardo_ms(1);
 EnviarComando(0x28);   //Function Set 0x28
 EnviarComando(0x06);   //Entry Mode Set 0x06
 EnviarComando(0x0C);   //Display On OFF 0x0C
 EnviarComando(0x01);   //Limpiar Display 0x01
 Linea(1);              //SET DD RAM L1 0x80
                    }
void EscribeEnLcd(char *c){
 int i;
 for(i=0;i<32;i++){//Se comprueba que existen datos a mostrar
      if(c[i]==0){
                  return;
                 }
      EnviarDato(c[i]);       //Se escribe el dato en el LCD
      if(i==15){  //Se cambia a la línea 2 cuando se termina la línea 1
            Linea(2);
               }
                  }
                           }
void EnviarComando(u8 dato){
 u8 aux;
 aux=0xf0&dato;              //Se obtiene el nibble alto del dato
 aux >>= 4;
 XGpio_DiscreteSet(&lcd,1,0x40|aux); //Se envía al LCD; bit D/I en '0'
 XGpio_DiscreteClear(&lcd,1,0x40|aux);
 retardo_ms(5);                      //Retardo necesario entre nibbles
 aux=0x0F&dato;                      //Se obtiene el nibble bajo del dato
 XGpio_DiscreteSet(&lcd,1,0x40|aux); //Se envía al LCD
 XGpio_DiscreteClear(&lcd,1,0x40|aux);
 retardo_ms(5);              //Retardo necesario entre nibbles
                          }
void Linea(int linea){
 if(linea==2){          //Se envía el comando para la línea 2 del LCD
      EnviarComando(0xC0);
```

```
            }
 else EnviarComando(0x80); //Se envía el comando para la línea 1 del LCD
                      }
void LimpiarLcd(){
 EnviarComando(0x01);      //Se envía el comando para limpiar el LCD
      Linea(1);
                  }
void EnviarDato(char c){
 u8 aux;
 aux=(0xF0)&c; //Se obtiene el nibble alto del dato
 aux>>=4;
 XGpio_DiscreteSet(&lcd,1,0x60|aux); //Se envía al LCD; bit D/I en '1'
 XGpio_DiscreteClear(&lcd,1,0x60|aux);
 retardo_ms(5); //Retardo necesario entre nibbles
 aux=(0x0F)&c; //Se obtiene el nibble bajo del dato
 XGpio_DiscreteSet(&lcd,1,0x60|aux); //Se envía al LCD; bit D/I en '1'
 XGpio_DiscreteClear(&lcd,1,0x60|aux);
 retardo_ms(5); //Retardo necesario entre nibbles
                       }
void Posicion(int linea,u8 cuadrito){
 if(linea==2){
      EnviarComando(0xC0|cuadrito); //Línea 2 posición "cuadrito"
           }
 else                               //Línea 1 posición "cuadrito"
      EnviarComando(0x80|cuadrito);
                                     }
void EscribeLinea(char *c){
 int i;
 for(i=0;i<16;i++){
 if(c[i]==0){           //Se comprueba que existan datos
      return;
           }
EnviarDato(c[i]);       //Se envían los datos
                  }
                        }
```