INSTITUTO POLITÉCNICO NACIONAL

CENTRO DE INVESTIGACIÓN EN COMPUTACIÓN

ANALIZADOR DE ESPECTRO UTILIZANDO UNA TARJETA PCI BASADA EN DSP'S

T E S I S

QUE PARA OBTENER EL GRADO DE:
MAESTRO EN CIENCIAS EN INGENIERÍA DE CÓMPUTO
CON ESPECIALIDAD EN SISTEMAS DIGITALES

PRESENTA

JOSÉ MANUEL ROMERO XIMIL

DIRECTOR: M. EN C. SERGIO SANDOVAL REYES

MÉXICO, D.F. FEBRERO DEL 2002

A mis papás, Gregoria y Gerardo, y a mis hermanas, Natalia y Alejandra, por el apoyo incondicional que siempre me han dado y estar conmigo en todo momento.

A mis amigos, por su invaluable compañía.

A Gloria ;)

AGRADECIMIENTOS

A mi director de tesis M. en C. Sergio Sandoval Reyes por su tiempo y buena disposición para asesorarme en la elaboración de esta tesis.

A todos los profesores de la Maestría en Ingeniería de Cómputo, en especial al Dr. Sergio Suárez Guerra con quien estuve trabajando en el desarrollo de un proyecto PIFI, en el que adquirí conocimientos útiles para la realización de la tesis y al M. en C. Osvaldo Espinosa Sosa por las enseñanzas en procesamiento digital de señales.

Al Centro de Investigación en Computación por permitirme cursar los estudios de maestría y utilizar la infraestructura necesaria para llevar a cabo el tema de tesis.

Al Instituto Politécnico Nacional, en el cual he cursado mis estudios de nivel medio superior, superior y maestría, por brindarme el apoyo económico para terminar mis estudios.

ANALIZADOR DE ESPECTRO UTILIZANDO UNA TARJETA PCI BASADA EN DSP'S

José Manuel Romero Ximil

RESUMEN

El campo de la instrumentación virtual es cada vez más amplio, donde podemos emplear diversos instrumentos como voltímetros, osciloscopios, amperímetros, entre otros. Uno de estos instrumentos es el analizador de espectro, que muestra una señal en el dominio de la frecuencia, usado en muchos campos de la ciencia como la medicina, la acústica, la sismología, y otros. El desarrollo de un analizador de espectro utilizando una tarjeta PCI basada en DSP's es descrito en el presente trabajo. Dicha tarjeta realiza el análisis espectral de una señal eléctrica utilizando un convertidor ADC, una memoria, un DSP y una interfaz PCI, en combinación con los recursos de una computadora personal. La tarjeta se conecta al bus PCI de la computadora y a través de una interfaz gráfica despliega el espectro en la pantalla. La tarjeta PCI utiliza el convertidor de señal analógica a señal digital para obtener muestras de la señal eléctrica. El procesamiento de las muestras para obtener la estimación de la potencia del espectro de la señal, se realiza con la Transformada Rápida de Fourier (FFT) a través de un DSP y una memoria que se encuentran en la tarjeta. Los datos que contienen el espectro de la señal son almacenados temporalmente en dicha memoria y transferidos al CPU de la computadora con la ayuda de un controlador para el bus PCI. En la computadora se desarrolló un programa que se encarga de recibir los datos que contienen el espectro de la señal y los despliega en la pantalla de la computadora. De esta manera, se proporciona un instrumento de medición que utiliza los recursos de una computadora.

SPECTRUM ANALYZER USING A DSP-BASED PCI BOARD

Jose Manuel Romero Ximil

ABSTRACT

As the virtual instrument field becomes broader, we can use many instruments such as voltmeters, oscilloscopes, ammeters, and others. One of these is the spectrum analyzer, which shows an input signal at frequency domain. It is used in many science fields like Medicine, Acoustics, Seismology, Speech Recognition and others. A spectrum analyzer using a DSP-based PCI board is described in this thesis. This one performs spectral analysis of an input signal using a board containing an ADC, memory, a DSP and a PCI interface that goes inside a PC. The board plugs into a computer PCI bus and through a graphic interface displays the spectrum on screen. The PCI board uses an ADC to get electric signal samples. Then the power spectrum estimation is obtained processing the samples through the Fast Fourier Transform (FFT) using the board's DSP and memory. Spectrum data are provisionally stored in memory and transferred to the CPU using a PCI bus controller. A spectrum analyzer program reads the spectrum data from PCI board and displays it on screen. In this way, a measurement instrument that uses PC resources is accomplished.

ÍNDICE

LISTA DE FIGURAS Y TABLAS

Figuras

Tablas

INTRODUCCIÓN

La digitalización de señales es un área muy grande que se incluye en los campos del control, instrumentación y otros. La digitalización tiene muchas aplicaciones y las necesidades cada vez son mayores. Una de esas necesidades es la adquisición, almacenamiento y presentación de señales eléctricas en una computadora.

El procesamiento digital de señales hace posible desarrollar algunos instrumentos de medición que pueden interactuar con las computadoras, tal es el caso de los instrumentos basados en PC o instrumentos virtuales. Estos dispositivos toman ventaja de la capacidad de almacenamiento, procesamiento y presentación de éstas, reduciendo los costos y el tiempo de diseño. Aunque cabe destacar que también se limita el potencial de los mismos.

El siguiente trabajo propone el desarrollo de un analizador de espectro basado en una PC mediante una tarjeta PCI que utiliza un procesador de señales digitales. Con este fin hemos organizado la tesis en siete capítulos, los cuales se describen a continuación.

En el capítulo 1, **Analizador de espectro basado en PC**, se describe qué hace un analizador de espectro, así como la importancia que tiene en el desarrollo de proyectos de investigación y de aplicaciones. El desarrollo de instrumentos virtuales es una alternativa para las personas que no requieren un instrumento muy costoso y que además cuentan con una PC. Por esta razón se propone una tarjeta para computadora, que trabaja de manera conjunta con un programa de aplicación para funcionar como un analizador de espectro. Esta tarjeta se conecta al bus PCI de una PC y contiene un DSP que realiza el procesamiento de las señales. También se señalan los objetivos y se resalta la necesidad de contar con instrumentos en las áreas de diseño electrónico.

En el capítulo 2, **Analizadores de espectro comerciales y solución propuesta**, se mencionan algunas de las soluciones que existen en el mercado de los analizadores de espectro basados en PC. Entre estos productos se encuentran los fabricados por National Instruments y por Hewlett Packard, que son las compañías líderes tanto en instrumentos virtuales como en equipos de medición. También se menciona la solución propuesta, que consiste de una tarjeta PCI que utiliza un DSP para obtener el espectro en frecuencia de una señal muestreada, junto con un programa de aplicación que despliega el espectro en la pantalla de la computadora.

En el capítulo 3, **Tarjeta que funciona como analizador de espectro**, se describe el funcionamiento de un analizador de espectro, que consta básicamente de una etapa de acondicionamiento de la señal, una etapa de digitalización, una de almacenamiento de muestras, otra de procesamiento de los datos para encontrar el espectro en frecuencia de la señal y finalmente la transferencia del espectro a la computadora a través del bus PCI. El analizador de espectro consiste de una tarjeta con un DSP y un software que permite visualizar el espectro de la señal de entrada y controlar algunos recursos de la tarjeta.

En el capítulo 4, **Diseño e implementación de la tarjeta PCI**, se muestra el diseño de la tarjeta PCI basada en un DSP que va a ser utilizada como un analizador de espectro. Se describe cada uno de los bloques que conforman la tarjeta y qué función realizan en el procesamiento de la señal para obtener el espectro en frecuencia de la misma.

En el capítulo 5, **Diseño del software de análisis espectral**, se describe el desarrollo del programa que obtiene la estimación de la potencia del espectro de la señal, así como la interfaz de la tarjeta con el sistema operativo Windows, que presenta el espectro de la señal en la pantalla de la computadora.

En el capítulo 6, **Pruebas y resultados**, se mencionan las diferentes pruebas que se realizaron a la tarjeta para comprobar su funcionamiento adecuado y los resultados de las mediciones a un conjunto de señales eléctricas.

En el capítulo 7, **Conclusiones**, se describen los objetivos alcanzados en el desarrollo del trabajo y se proponen trabajos a futuro.

En la **Bibliografía** aparecen las fuentes consultadas para la elaboración de la tesis.

Finalmente, se integran el diagrama esquemático de la tarjeta, información del bus PCI y los archivos fuente de los programas en los **Anexos**.

CAPÍTULO 1

ANALIZADOR DE ESPECTRO BASADO EN PC

En el presente capítulo se describe la necesidad del uso de un analizador de espectro, un instrumento poco común en los laboratorios, debido a su elevado costo, pero que es fundamental en muchos campos de investigación aplicada.

1.1 Antecedentes

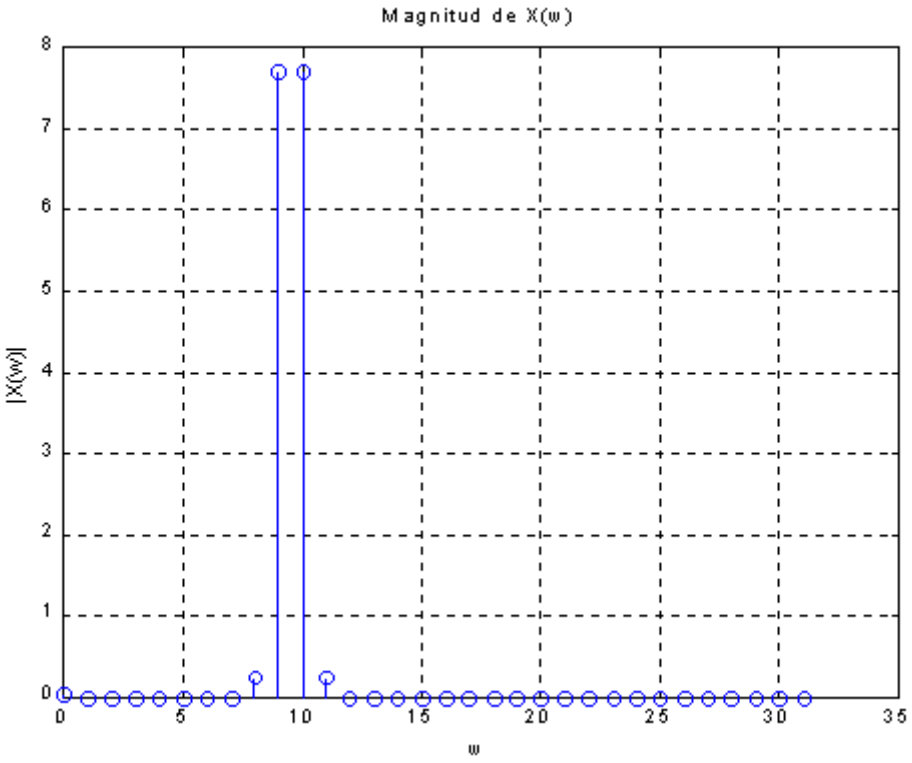

Figura 1.1. Espectro en frecuencia de una señal senoidal.

Un analizador de espectro [BG: 1988] puede ser pensado como un filtro con un ancho de banda estrecho que es barrido eléctricamente a través de un intervalo fijo de frecuencias mientras proporciona un despliegue visual de la amplitud de salida contra la frecuencia (figura 1.1).

El analizador de espectro nos permite observar una señal en el dominio de la frecuencia, es decir, podemos ver las magnitudes de las distintas frecuencias que conforman una señal dada.

El uso de los analizadores de espectro basados en PC no es muy común en México, algunas causas son el costo elevado de las unidades, se utilizan en casos muy particulares y son pocas las empresas que los venden.

Se han desarrollado varios proyectos que enfocan sus intereses a la construcción de instrumentos virtuales basados en una PC. La ventaja de los instrumentos virtuales radica en que se utiliza una infraestructura muy común en los centros de trabajo, las computadoras. Estos instrumentos son más económicos que los convencionales y por lo tanto son una buena opción para la gente que necesita un instrumento de medición y que cuenta con una computadora.

1.2 Descripción del problema

El análisis de señales en el dominio de la frecuencia es de suma importancia en muchas áreas del conocimiento [ST: 1997]. En el monitoreo de vibración, el espectro de una señal muestreada nos da información sobre las características mecánicas de las partes bajo estudio. En el análisis de voz, nos ayuda al reconocimiento. En sistemas de radar y sonar, proporciona información sobre la localización de la fuente. En medicina, el análisis espectral de electrocardiogramas y electroencefalogramas proporciona material útil para diagnóstico. En sismología, puede ayudar a predecir los movimientos de la tierra. Aunque se observa que el uso del análisis espectral es muy útil en muchas áreas, no siempre se cuenta con un analizador de espectro, ya sea porque no existe en el centro de trabajo, o porque no está disponible para un gran número de personas. Y debe destacarse que aunque los analizadores de espectro se utilizan en muchas áreas, su uso es muy especializado.

Por esta razón, se propone un analizador de espectro basado en PC, que represente una opción para aquellos investigadores, estudiantes o profesionistas, que necesiten realizar análisis de señales en el dominio de la frecuencia.

Este instrumento consiste de una tarjeta que se conecta al bus PCI de una PC y de un software que permite interactuar al usuario con el hardware. La tarjeta se encarga de adquirir los datos, de procesarlos y de enviarlos, con la ayuda del software, a la computadora para su presentación en una pantalla. El tratamiento de los datos adquiridos lo realiza un procesador de señales digitales que, por medio de algoritmos matemáticos, encuentra el espectro en frecuencia de una señal dada.

El uso del bus PCI tiene muchas ventajas, como es la transferencia de datos de 32 bits a 33 MHz. Además, las computadoras actuales dan soporte a PCI más que a ningún otro tipo de bus y ya está quedando en desuso el estándar ISA, que se ha vuelto lento y poco efectivo en aplicaciones de tiempo real.

1.3 Definición del problema

Construcción de un analizador de espectro de bajo costo que utiliza procesamiento digital de señales basado en una computadora personal con una interfaz PCI.

1.4 Objetivos

1.4.1 Objetivo general

Diseñar y construir una tarjeta PCI que utilice un procesador de señales digitales para que funcione como un analizador de espectro.

1.4.2 Objetivos específicos

- Diseño y construcción de la tarjeta.
- Desarrollo del software de procesamiento para el DSP.
- Desarrollo de una interfaz gráfica para visualizar el espectro en una pantalla.

1.5 Justificación

El desarrollo de una tarjeta para computadora que puede ser utilizada como un analizador de espectro, pondría al alcance de muchas personas la posibilidad de trabajar en sus proyectos, sin tener que recurrir a los pocos instrumentos de medición que se encuentran en los laboratorios o centros de trabajo, además de que se aprovecharían las características propias de las computadoras, como son su capacidad gráfica y de almacenamiento de datos y la posibilidad de compartir información con otras computadoras.

A pesar de que los instrumentos de este tipo ya se encuentran en el mercado, aún tienen un precio muy elevado, por encima del costo de una computadora personal. Entonces, el desarrollo de un instrumento a un precio más accesible, sería una buena alternativa para las personas que necesitan un analizador de espectro y que cuentan con una computadora personal.

1.6 Resumen

En este capítulo se describió qué hace un analizador de espectro, así como la importancia que tiene en el desarrollo de proyectos de investigación y de aplicaciones. El desarrollo de instrumentos virtuales es una alternativa para las personas que no requieren un instrumento muy costoso y que además cuentan con una PC. Por esta razón se propone una tarjeta para computadora, que trabaja de manera conjunta con un programa de aplicación para funcionar como un analizador de espectro. Esta tarjeta se conecta al bus PCI de una PC y contiene un DSP que realiza el procesamiento de las señales. También se señalan los objetivos y se resalta la necesidad de contar con este instrumento en varias áreas del conocimiento. En el siguiente capítulo se menciona la solución que se hace al problema junto con una revisión de algunos analizadores de espectro que se encuentran en el mercado.

CAPÍTULO 2

ANALIZADORES DE ESPECTRO COMERCIALES Y SOLUCIÓN PROPUESTA

Existen en el mercado varias compañías que ofrecen productos de instrumentación virtual, entre las que destaca National Instruments (NI), además se encuentran aquellas compañías que venden instrumentos de medición convencionales, como Hewlett Packard (HP). A continuación se describen algunos productos de análisis de señales en el dominio de la frecuencia que ofrecen estas compañías y también se introduce la solución propuesta al problema planteado en el capítulo anterior.

2.1 Soluciones afines

Existen varias compañías en el mercado que ofrecen instrumentos de medición, algunos son tarjetas que se insertan en la ranura de una computadora y las funciones que ofrecen son manipuladas por medio de software, otros son instrumentos que no dependen de una computadora, pero que pueden comunicarse con ella, facilitando ambos tipos el manejo de la información.

Sea de una forma u otra, en el caso de los analizadores de espectro, es utilizado cada vez más el procesamiento digital de señales como herramienta para desarrollar estos instrumentos. De esta manera vemos que compañías como HP y NI ofrecen algunos de sus productos basados en algoritmos matemáticos como la Transformada Rápida de Fourier (Fast Fourier Transform: FFT).

A continuación se describen algunos instrumentos de estas compañías.

2.1.1 Solución National Instruments

NI fabrica tarjetas de adquisición de datos que se insertan en las ranuras de expansión de las computadoras. Entre estas tarjetas se encuentran aquellas que son especiales para utilizarse como instrumentos de adquisición de señal dinámica. Los modelos 4551 y 4552 [NI: 1998] son tarjetas PCI similares, que se utilizan como analizadores de señal dinámica (figura 2.1).

El NI 4551 tiene dos canales de entrada que hacen un muestreo simultáneo a una tasa de 204,800 muestras por segundo y dos canales de salida con actualizaciones simultáneas a una tasa de 51,200 muestras por segundo. Los convertidores, tanto de entrada como de salida, tienen una resolución de 16 bits y son del tipo delta-sigma. El NI 4552 tiene las mismas características que el NI 4551 en cuanto a sus componentes, pero tiene 4 canales de entrada y no tiene canales de salida. En la figura 2.2 se muestra el diagrama de la tarjeta.

Figura 2.1. Tarjetas NI 4551 y NI 4552.

Las aplicaciones que soporta son el procesamiento y análisis de señales de audio, investigación de acústica y voz, sonar, prueba y medición de frecuencias de audio, análisis de vibración, y cualquier aplicación que requiera alta fidelidad en la adquisición de señales con un ancho de banda de hasta 95 KHz.

El software que acompaña a la tarjeta es el VirtualBench-DSA que permite utilizar la tarjeta como un analizador dinámico, para observar señales en el dominio de la frecuencia. También se puede utilizar toda una serie de productos de la misma compañía como LabVIEW, LabWindows, ComponentWorks, etc., estos programas son de propósito general y algunos tienen interfaces con lenguajes de programación como C.

El VirtualBench-DSA es una aplicación para el sistema operativo Windows que puede realizar varias funciones con la tarjeta NI 4551/4552 como medir distorsión armónica total, contenido armónico, respuesta en frecuencia, respuesta al impulso, espectro de potencia, espectro de amplitud, además de funciones de ventaneo. La ventana principal muestra una pantalla donde se puede visualizar la señal que se va a analizar y la señal después del análisis. Las distintas opciones y sus respectivos parámetros se introducen por medio de botones y cajas de diálogo (figura 2.3).

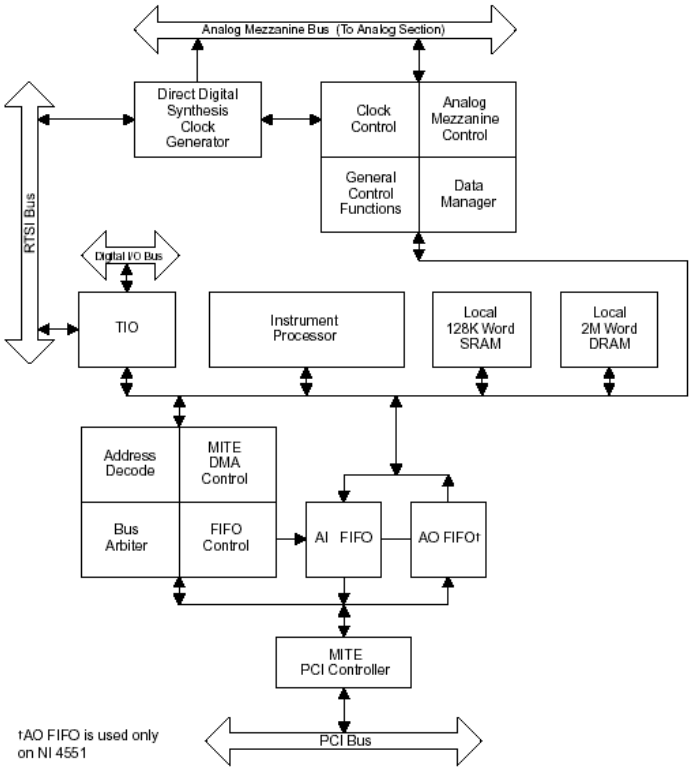

Figura 2.2. Diagrama de la tarjeta NI4551.

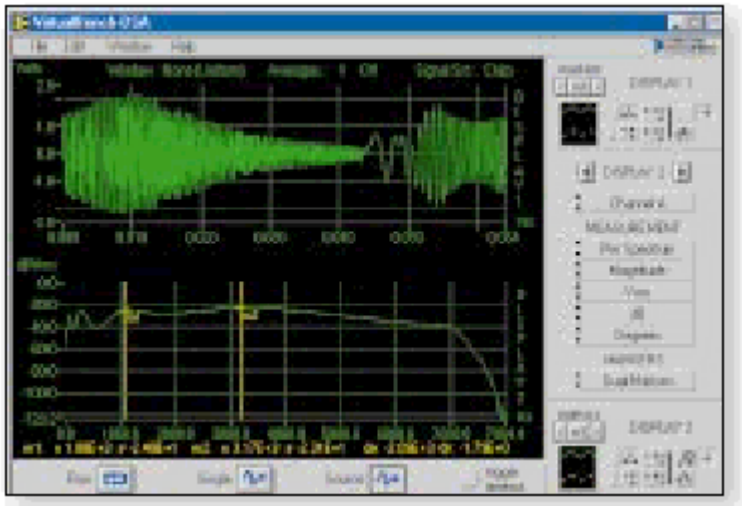

Figura 2.3. Analizador de señal dinámica de VirtualBench.

2.1.2 Solución Hewlett Packard

HP se ha distinguido por la fabricación de instrumentos de medición, y entre sus productos se encuentra un analizador de señal dinámica, el HP 35670A [HP: 1997]. Es un equipo portátil que realiza análisis en el dominio de la frecuencia, del tiempo y de la amplitud (figura 2.4).

Los analizadores de espectro basados en la FFT, como el HP 35670A son ideales para la medición de espectro de señales de baja frecuencia como la voz o la vibración mecánica.

Este instrumento tiene dos canales de entrada, los canales comparten un ADC de 16 bits de resolución con un ancho de banda de 102,400 Hz.

Figura 2.4. Analizador de señal dinámica HP 35670A.

Este instrumento tiene una pantalla donde se puede visualizar el espectro de una señal, la señal misma (función en modo osciloscopio) y el análisis en el dominio de la amplitud. Tiene varios botones y perillas que controlan las funciones que se pueden utilizar y los parámetros que necesitan cada una de ellas. Además de ser portátil, puede almacenar las muestras en un disco de 3.5 pulgadas.

2.2 Solución propuesta

Tomando como base las anteriores soluciones, la solución que se propone es una tarjeta PCI con funciones de un analizador de espectro. Esta tarjeta hace uso de la FFT para obtener el espectro en frecuencia de una señal muestreada. El algoritmo de la FFT es implementado en lenguaje ensamblador para el DSP TMS320C31 que es de la tercera generación de procesadores de señales digitales de Texas Instruments (TI). Este DSP puede realizar operaciones con números de punto flotante (reales). La tarjeta cuenta con un canal de entrada de 16 bits con una tasa máxima de dos y medio millones de muestras por segundo.

Los datos procesados en la tarjeta, por medio del DSP, se transfieren a la computadora, donde una aplicación para el sistema operativo Windows muestra el espectro de la señal. Esta aplicación puede manipular los parámetros necesarios para configurar el instrumento, como frecuencia de muestreo y número de muestras. Básicamente consiste de una pantalla que muestra las señales y un conjunto de botones y cajas de texto que permiten variar los parámetros.

Este analizador de espectro, a diferencia de los fabricados por HP y NI, utiliza un procesador de señales digitales de uso común, el TMS320C31 de TI. Esta compañía tiene mucho soporte para el desarrollo de aplicaciones y sus productos son un estándar en la industria. Además este instrumento está enfocado a aplicaciones que no necesiten el uso de varios canales de entrada. Esto permite que el costo de la tarjeta sea inferior en comparación a las otras soluciones. Algunas similitudes con las soluciones de HP y NI son que utiliza la FFT para realizar el análisis de señales, la vinculación con las computadoras personales, que es indispensable en los desarrollos de los últimos tiempos y el uso de un bus de alta velocidad como es el PCI, que es de gran utilidad para transferencias de bloques de datos.

2.3 Resumen

En este capítulo se describieron algunas de las soluciones que existen en el mercado de los analizadores de espectro basados en PC. Entre estos productos se encuentran los fabricados por National Instruments y por Hewlett Packard, que son las compañías líderes tanto en instrumentos virtuales como en equipos de medición. También se menciona la solución propuesta, que consiste de una tarjeta PCI que utiliza un DSP para obtener el espectro en frecuencia de una señal muestreada, junto con un programa de aplicación que despliega el espectro en la pantalla de la computadora. En el siguiente capítulo se muestra una descripción funcional de la solución propuesta.

CAPÍTULO 3

TARJETA QUE FUNCIONA COMO ANALIZADOR DE ESPECTRO

En este capítulo se describe la forma como funciona un analizador de espectro basado en una PC y los bloques funcionales en que se divide. Además se menciona el hardware y el software que conforman al instrumento y el hardware y software necesario para el diseño.

3.1 Descripción del funcionamiento

En la figura 3.1 se muestran los bloques que componen a un analizador de espectro basado en una PC con interfaz PCI. Primero, se necesita una etapa de acondicionamiento para manejar la señal que se desea digitalizar, esta etapa depende de las características de la etapa de conversión analógico-digital. Después, un ADC digitaliza la señal en forma de muestras que posteriormente se almacenan en memoria, para ser procesadas por el DSP en la siguiente etapa. Finalmente, los datos son transferidos a la computadora por medio de la interfaz PCI. Las muestras son transmitidas por el puerto PCI a la memoria del sistema en forma de bloque. La lógica de control permite manejar las distintas etapas.

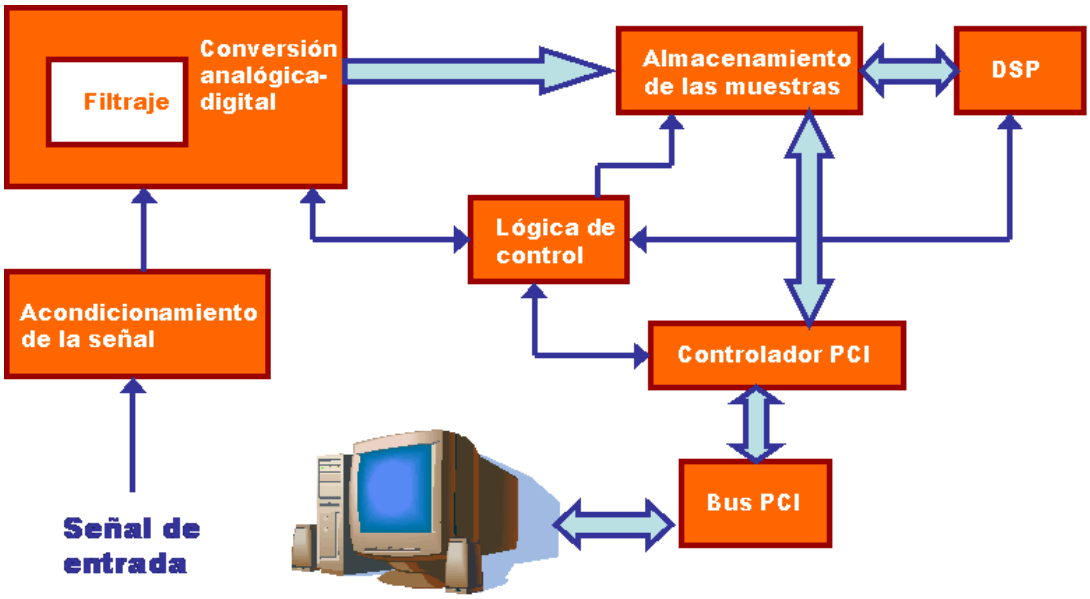

Figura 3.1. Diagrama a bloques del analizador de espectro.

Cuando se desea observar el espectro de una señal en un analizador, se debe tomar en cuenta el voltaje máximo que acepta a la entrada el instrumento, el ancho de banda tanto de la señal como del instrumento, además de otras características. Para el caso de esta tarjeta PCI el voltaje máximo es de 4Vp-p. El ADC que utiliza la tarjeta requiere de un circuito que maneje la señal de entrada y que convierta la señal de entrada en una señal diferencial. Por esta razón se utiliza un circuito que convierte la señal referenciada a tierra a una señal diferencial y que puede manejar un ancho de banda suficiente para el ADC.

La etapa de conversión analógico-digital es la que se encarga de digitalizar la señal, asigna un número binario a un valor determinado de la amplitud de la señal, generando un conjunto de datos o muestras. Dentro de la etapa de digitalización se contempla un bloque de filtraje para eliminar componentes de frecuencia por encima del rango de entrada. El ADC debe proporcionar un ancho de banda amplio y una tasa de muestreo alta, ya que normalmente se desea observar señales que van desde unos cuantos hertz hasta más de 1 MHz. Normalmente los convertidores que se utilizan para este caso son del tipo delta-sigma, porque tienen buena resolución y su conversión es lineal.

El almacenamiento de los datos en la memoria es un punto muy importante, porque la velocidad en que la memoria puede guardar los datos, determina en cierta forma el rendimiento del sistema. Por eso, la memoria necesita ser de alta velocidad, para que pueda funcionar de manera conjunta con la interfaz PCI y lograr una alta tasa de transferencia de datos entre la tarjeta y la computadora. También es importante que esta memoria sea de fácil acceso para el DSP para que éste pueda realizar los cálculos de una forma más eficiente.

Un procesador de señales digitales realiza los cálculos necesarios para encontrar el espectro de la señal digitalizada. El DSP utiliza algoritmos matemáticos, entre los que se encuentra la FFT, para transportar la señal, del dominio del tiempo al dominio de la frecuencia. Estos procesadores están optimizados para realizar tareas en las que el uso exhaustivo de operaciones matemáticas es lo principal. Esta característica los hace ideales para esta aplicación.

La interfaz PCI es esencial en aplicaciones que requieren la transferencia de datos en tiempo real, en esta aplicación el uso de un bus de alta velocidad permite observar en la pantalla de la computadora el espectro de la señal de una forma más rápida y continua. El bus PCI es ya un estándar para las tarjetas que se insertan en una ranura de expansión de una PC, entonces su utilización en esta aplicación es necesaria.

La lógica de control se encarga de manipular los distintos dispositivos electrónicos, para asignar recursos, sincronizar la transferencia de datos y para decodificar comandos. Los dispositivos que realizan estas funciones necesitan ser de una velocidad que cumpla con los requerimientos de tiempo de los recursos de la tarjeta.

El espectro obtenido en la tarjeta se muestra en la pantalla de la computadora, pero también existe la posibilidad de observar la señal original, es decir, la tarjeta puede funcionar como un osciloscopio. La pantalla de la computadora simula el panel frontal de un instrumento convencional, es decir, se pueden observar perillas y botones, además de cajas de texto para introducir datos. Estos controles funcionarán de la misma manera que lo hacen sus contrapartes

en el otro tipo de instrumento, modifican los parámetros de digitalización y de esta forma controlan el hardware.

3.2 Hardware y Software a desarrollar

El analizador de espectro consiste de una parte de hardware y otra de software. La parte de hardware la constituye la tarjeta PCI que utiliza un DSP para realizar los cálculos de la FFT. El software consiste de un programa de procesamiento para el DSP y de un programa de aplicación que muestra el espectro en frecuencia de la señal que se introduce en la tarjeta.

3.2.1 Hardware de la tarjeta PCI

La tarjeta cuenta con un ADC delta-sigma para digitalizar la señal de entrada. Este ADC tiene incorporado un filtro que limita la frecuencia de la señal. Normalmente este tipo de convertidor es el que se utiliza para digitalizar señales en los analizadores de espectro. Las muestras son almacenadas en memoria mientras se digitaliza un número determinado de ellas. Posteriormente, las muestras son utilizadas en los cálculos que realiza el DSP TMS320C31 para encontrar el espectro de la señal. Cuando el DSP encuentra el espectro, los datos son enviados a la computadora, a través del bus PCI para que sean desplegados en la pantalla (un ejemplo de un espectro en frecuencia se muestra en la figura 3.2). La tarjeta cuenta con un circuito integrado que funciona como interfaz entre la tarjeta misma y el bus PCI en la computadora. Este circuito se hace cargo del protocolo de transferencia de datos.

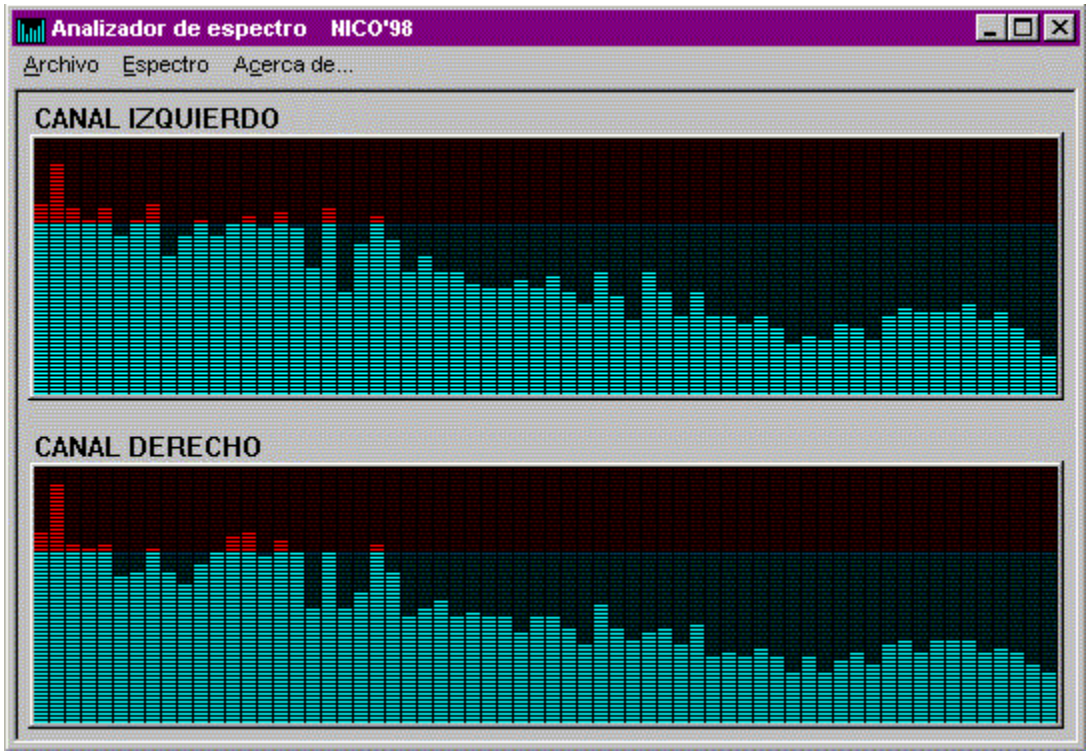

Figura 3.2. Ventana que muestra una interfaz gráfica para un analizador de espectro.

3.2.2 Software para obtener el espectro en frecuencia de una señal

El software que utiliza el analizador de espectro consiste de un programa para el DSP y otro para la computadora. El programa de procesamiento se ejecuta en el DSP, y obtiene la estimación de la potencia del espectro de una señal eléctrica. El programa de aplicación está formado por un manejador de dispositivo que permite el acceso al hardware de la tarjeta desde el sistema operativo Windows, para manipular algunas variables de la lógica de control y poder dar flexibilidad a la forma en que se van a muestrear las señales. Es importante el uso de una biblioteca de funciones que pueda tener acceso a los recursos del hardware, como es la suministrada por el programa WinDriver. Estas funciones ahorran una gran cantidad de tiempo en el desarrollo de manejadores de dispositivos. El programa de aplicación también consiste de una pantalla que representa el panel frontal de un instrumento convencional. Tiene un área para desplegar el espectro de la señal y cuenta con botones y cajas de texto para manipular los recursos de la tarjeta. El desarrollo del software involucra el uso de un entorno de programación visual como C++ Builder o Visual C++.

3.3 Recursos de cómputo a emplear

- Computadora Pentium o superior con una ranura PCI disponible.
- Convertidor analógico-digital tipo delta-sigma de 16 bits.
- Chip controlador de bus PCI S5920.
- Memorias estáticas de alta velocidad.
- DSP TMS320C31 de Texas Instruments.
- Componentes de lógica programable.
- Programa WinDriver.
- C++ Builder para la programación de la aplicación.
- Kit de desarrollo TMS320C3X DSK.

3.4 Pruebas de funcionamiento

Las pruebas que se deben realizar son las siguientes:

- Digitalización de la señal de entrada.
- Almacenamiento de las muestras en la memoria estática.
- Verificar que el algoritmo de la FFT funcione correctamente.
- Funcionamiento de la interfaz PCI para transferir y recibir datos entre la tarjeta y la computadora.
- Verificar que la lógica de control actualice las variables que se necesitan en la etapa de acondicionamiento y en la etapa de conversión.
- Comunicación de la tarjeta con el sistema operativo por medio del manejador.

- Verificar que el espectro mostrado en la pantalla sea el correcto, dado un conjunto de señales de entrada conocidas, como la función seno.
- Contrastar el espectro mostrado en la pantalla con el de otro instrumento, dada una misma señal de entrada.

3.5 Resumen

En este capítulo se describió el funcionamiento de un analizador de espectro, que consta básicamente de una etapa de acondicionamiento de la señal, una etapa de digitalización, una de almacenamiento de muestras, otra de procesamiento de los datos para encontrar el espectro en frecuencia de la señal y finalmente la transferencia del espectro a la computadora a través del bus PCI. El analizador de espectro consiste de una tarjeta con un DSP y un software que permite visualizar el espectro de la señal de entrada y controlar algunos recursos de la tarjeta. En el siguiente capítulo se describe de forma detallada el funcionamiento de cada parte de la tarjeta.

CAPÍTULO 4

DISEÑO E IMPLEMENTACIÓN DE LA TARJETA PCI

En este capítulo se describe el funcionamiento de la tarjeta PCI que funciona como un analizador de espectro. Cada una de las partes que la componen se describe de forma detallada.

4.1 Diseño del hardware de análisis de espectro

En esta sección se explica a detalle la configuración de los dispositivos que conforman la tarjeta PCI. El diagrama a bloques de la tarjeta se puede ver en la figura 4.1. Se resaltan los bloques que conforman la tarjeta, referente al hardware descrito en la sección 3.2.1.

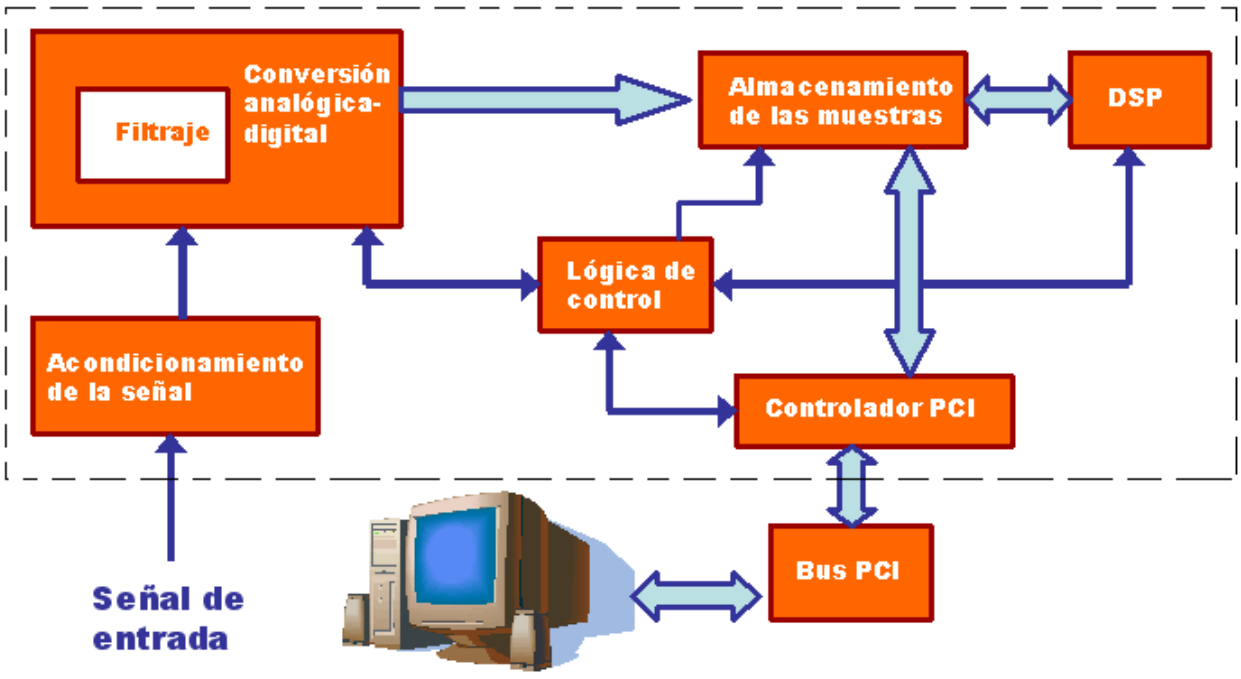

Figura 4.1. Diagrama a bloques de la tarjeta PCI.

La señal de entrada pasa por una etapa de acondicionamiento, la cual produce una señal diferencial que es requerida por el ADC. Las muestras adquiridas por el ADC son almacenadas en memoria RAM. El DSP lee estas muestras y las procesa, aplicando, entre otras cosas, la FFT. Una vez que se ha procesado la señal en el DSP, se transfieren los datos almacenados en RAM hacia la computadora, por medio del bus PCI. Todos estos pasos son administrados por la lógica de control. Una vez que los datos procesados están en la computadora, el software de aplicación muestra el espectro de la señal. A continuación se describen más a detalle los bloques de acondicionamiento y de conversión.

4.1.1 Etapa de digitalización de la señal de entrada

En la etapa de entrada de la señal se tiene por un lado un conector BNC por donde una señal es introducida, y por el otro lado se tiene un bus de 16 bits que transporta las muestras de la señal en dirección a la memoria RAM de la tarjeta. En la figura 4.2 se muestra un diagrama a bloques de esta parte de la tarjeta.

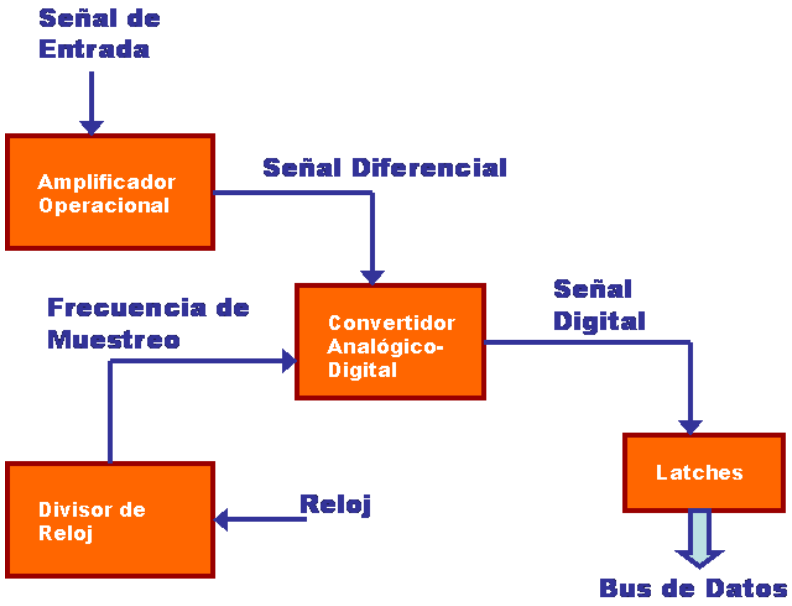

Figura 4.2. Diagrama a bloques de la etapa de digitalización.

Primero, la señal de entrada que es introducida por medio del conector BNC pasa por una etapa de acondicionamiento, conformada por el amplificador operacional AD8138 que convierte la señal referenciada a tierra en una señal diferencial, que va hacia el convertidor analógico/digital AD9260. La configuración del amplificador operacional se muestra en la figura 4.3. En esta figura se observa que tanto el convertidor como el amplificador operacional utilizan la misma alimentación de 5V y las terminales VOCM y VREF de los dispositivos se conectan entre sí para definir el voltaje de referencia de la señal, que es igual a 2.5V, la mitad del voltaje de alimentación.

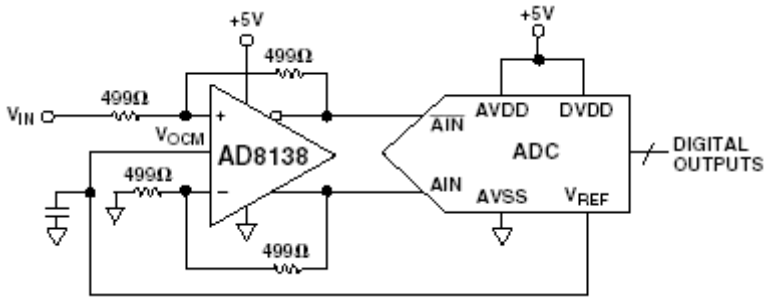

Figura 4.3. Configuración típica del amplificador AD8138.

Se utiliza el convertidor analógico/digital AD9260 en la etapa de digitalización de la señal de entrada. El convertidor tiene una resolución de 16 bits y puede manejar una tasa de palabra de salida de 2.5MHz. En la figura 4.4 se muestra el diagrama a bloques del convertidor. Este convertidor puede trabajar sin un circuito de retención de señal porque hace un muestreo a una frecuencia de 20MHz. Además, cuenta con tres etapas de filtraje, que implementan tres filtros FIR pasabajos que se ajustan automáticamente a la frecuencia de muestreo y para una tasa de salida de 2.5MHz la frecuencia de corte es de 1.1MHz. De esta manera, se tiene todo un sistema de muestreo en un solo chip.

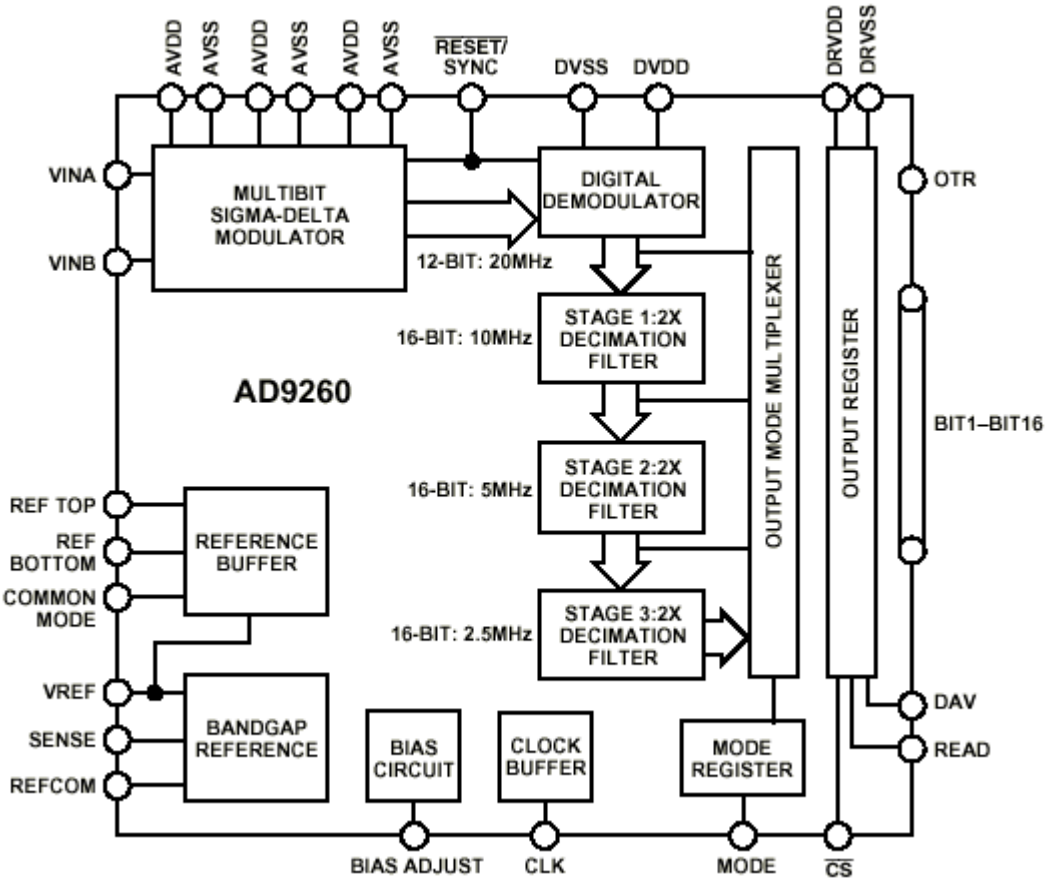

Figura 4.4. Diagrama a bloques del convertidor AD9260.

Se utiliza un circuito TC74VHC4040 para obtener, a partir de una frecuencia de 20MHz, un conjunto de frecuencias que sirven como reloj para el ADC. Este dispositivo es un contador de 12 bits y al ir incrementando su valor, se obtienen en sus terminales un conjunto de frecuencias menores a la introducida al reloj de este dispositivo. Para seleccionar una frecuencia específica de esas terminales, se utiliza un circuito multiplexor 74FCT151 que habilitar sólo una a la vez, este circuito utiliza tres bits de control que definen ocho diferentes frecuencias. Estos tres bits se pueden manipular por medio de software, escribiendo en los tres bits más significativos del byte 3 del buzón de salida PCI. Con estos dos circuitos, y con ayuda del software, podemos tener 8 diferentes tasas de muestreo para el convertidor.

El convertidor está configurado para que use una entrada diferencial de 4V pico a pico, esto se logra conectando sus terminales SENSE y REFCOM a tierra. El convertidor es capaz de operar en cuatro modos diferentes, proporcionando varias tasas de salida de datos. Colocando la terminal MODE en tierra se obtiene el mejor desempeño del dispositivo, proporcionando 2.5 millones de muestras por segundo. Para más detalles observar el diagrama en el Anexo A y ver la hoja de especificaciones del dispositivo.

Las terminales READ, OTR, DAV, /CS y /RESET del convertidor se utilizan para controlar la parte digital. READ y /CS controlan la salida de las muestras, estas terminales siempre están habilitadas, así, la información en los latches (que se observan en la figura 4.2) siempre está actualizada. OTR es la señal que indica un desbordamiento, la tarjeta no toma en cuenta los desbordamientos, por lo que no es utilizada esta terminal. DAV indica cuando hay una muestra disponible en la salida, esta señal se conecta a la lógica de control implementada en los CPLD's; y /RESET sirve para inicializar el convertidor, también se conecta a la lógica de control.

Finalmente, a la salida de los datos (en las terminales BIT1-16) se tienen latches que sirven como interfaz con el bus de datos de la tarjeta, que conecta al ADC con la memoria RAM en el bloque de almacenamiento de muestras. La lógica de control los habilita e inhabilita con la señal ADCOE.

4.1.2 Almacenamiento de muestras

Para almacenar las muestras que proporciona el convertidor analógico-digital se utilizan dos memorias IDT71016 que tienen un bus de datos de 16 bits cada una y un bus de direcciones de 16 bits también. Una memoria representa los 16 bits menos significativos de la palabra de 32 bits y la otra representa los 16 bits más significativos.

Puesto que el controlador PCI utiliza ciclos de acceso de ráfaga para realizar las transacciones de lectura y escritura, se requiere un circuito que genere las direcciones. El ADC tampoco genera direcciones. Por esta razón un circuito TC74VHC4040 se encarga de generar los 12 bits menos significativos del bus de direcciones para utilizarse con el controlador PCI y el ADC. Los cuatro bits restantes son generados por la lógica de control, para un total de 16 bits en el bus de direcciones. El reset, el reloj y la habilitación de la salida de este dispositivo son controlados por la lógica programada en los CPLDs. La figura 4.5 muestra los bloques que comprenden el almacenamiento de las muestras.

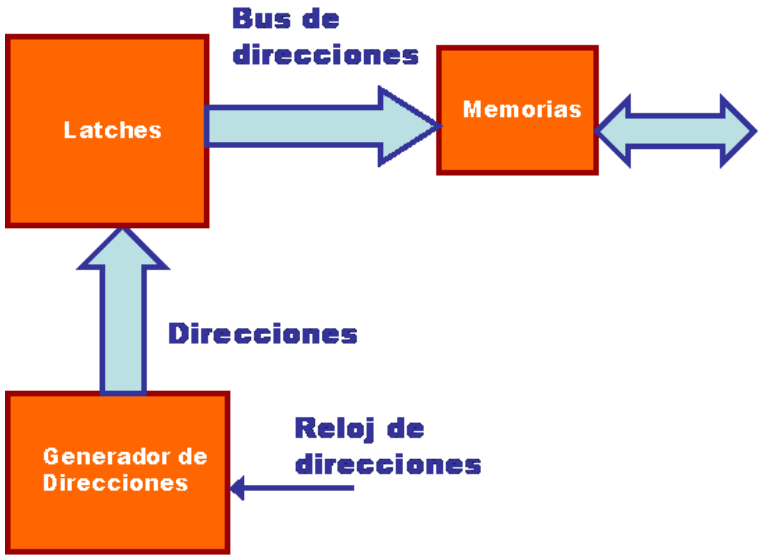

Figura 4.5. Bloques del almacenamiento de muestras.

Una vez que se tienen las muestras en la memoria RAM, el DSP comienza a procesar esos datos y el resultado del procesamiento lo guarda también en memoria RAM. La siguiente sección muestra las características del DSP y su importancia en el procesamiento de la señal.

4.1.3 Procesador Digital de Señales

El dispositivo que se utiliza para realizar el procesamiento digital de las señales, en este caso la Transformada Rápida de Fourier, es el DSP TMS320C31 de la compañía Texas Instruments. A continuación se presenta una breve descripción del dispositivo [TI: 1997].

La arquitectura del DSP C31 responde a la demanda de sistemas basados en algoritmos aritméticos que requieren soluciones tanto de hardware como de software. Se obtiene un alto rendimiento por medio de la unidad de punto flotante, la memoria interna del dispositivo, el alto grado de paralelismo y el controlador de DMA. Estas características hacen al C31 una valiosa herramienta en el análisis del espectro de una señal.

El C31 tiene una arquitectura de CPU basada en registros. La unidad central de procesamiento consiste de los componentes siguientes (ver figura 4.6):

- Multiplicador de punto flotante/enteros.
- Unidad aritmética lógica (ALU).
- Registro de corrimiento de 32 bits.
- Buses internos (CPU1/CPU2 y REG1/REG2).
- Unidades auxiliares de registros aritméticos (ARAUs).
- Otros registros.

El multiplicador realiza multiplicaciones de enteros de 24 bits y valores de punto flotante de 32 bits. La ALU ejecuta operaciones en un solo ciclo, de datos enteros de 24 bits, lógicos de 32 bits

y de punto flotante de 40 bits, incluyendo conversiones de enteros y de punto flotante. El registro de corrimiento es utilizado para desplazar hasta 32 bits hacia la izquierda o la derecha en un solo ciclo. Cuatro buses internos, CPU1, CPU2, REG1 y REG2 traen dos operandos de la memoria y dos operandos del archivo de registro, permitiendo multiplicaciones y sumas/restas en paralelo sobre cuatro operandos enteros o de punto flotante en un ciclo. Dos unidades auxiliares de registros aritméticos (ARAU0 y ARAU1) pueden generar dos direcciones en un ciclo. Las ARAUs operan en paralelo con el multiplicador y la ALU.

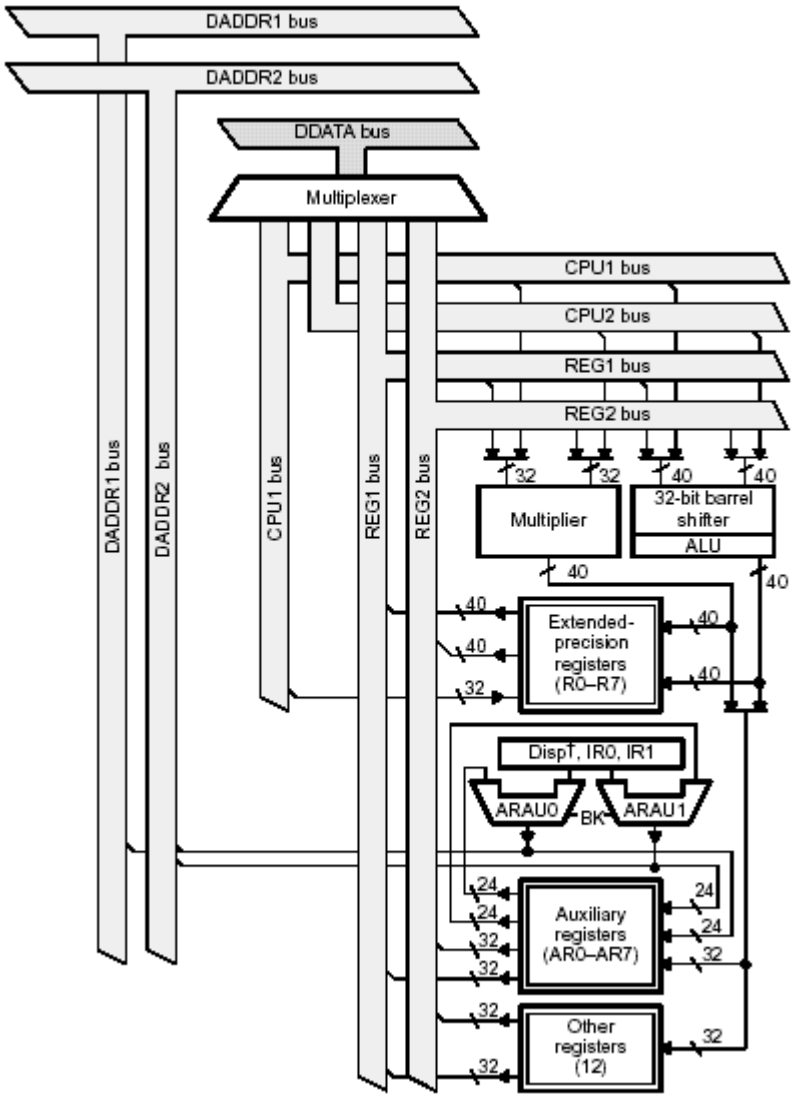

Figura 4.6. Unidad central de procesamiento del C31.

Los registros de precisión extendida (R7-R0) pueden almacenar y soportar operaciones de enteros de 24 bits y números de punto flotante de 32 bits. Los registros auxiliares de 32 bits (AR7-AR0) son accedidos por la CPU y modificados por las dos ARAUs. La función principal de los registros auxiliares es la generación de direcciones de 24 bits, también pueden ser usados como contadores o registros de propósito general de 32 bits que pueden ser modificados por el multiplicador y la ALU. El apuntador a la página de datos (DP) es un registro de 32 bits, los 8

bits menos significativos de este registro son utilizados por el modo de direccionamiento directo. Los registros índices de 32 bits (IR0 e IR1) contienen el valor usado por la ARAU para generar una dirección indexada. El apuntador a la pila del sistema (SP) es un registro de 32 bits que contiene la dirección del tope de la pila del sistema. El registro de estado (ST) contiene información global relacionada con el estado de la CPU. El registro de banderas de interrupción de la CPU (IF) también es de 32 bits. El registro de banderas de entrada/salida (IOF) controla el funcionamiento de las terminales externas XF0 y XF1. El contador de repeticiones (RC) es un registro de 32 bits que especifica el número de veces que se repite un bloque de código cuando se realiza una repetición de bloque. Cuando el procesador está operando en modo de repetición, el registro dirección de inicio de repetición (RS) de 32 bits contiene la dirección de inicio del bloque de memoria de programa a repetirse, y el registro dirección final de repetición (RE) de 32 bits contiene la dirección final del bloque a repetir.

El contador de programa (PC) es un registro de 32 bits que contiene la dirección de la siguiente instrucción a ser buscada. Aunque el contador de programa no es parte del archivo de registro de la CPU, es un registro que puede ser modificado por las instrucciones que modifican el flujo del programa.

El algoritmo de la FFT utiliza esencialmente operaciones aritméticas como multiplicaciones y sumas (ver capítulo 5 para más detalles), por esta razón, la arquitectura que se ha descrito es muy adecuada para este propósito. El programa que implementa el algoritmo es cargado en la memoria interna del DSP, con esto se logra un alto rendimiento, debido a la misma arquitectura. A continuación se explica el funcionamiento de la memoria interna del DSP, que se utiliza casi exclusivamente para almacenar el programa que implementa la FFT.

El espacio de memoria total del C31 es de 16 millones de palabras de 32 bits. El espacio de programa, datos y entrada/salida está contenido dentro de este espacio de direcciones de 16 millones de palabras, permitiendo el almacenamiento de tablas, coeficientes, código de programa, o datos. El C31 proporciona 2K X 32 bits de memoria interna, además de un cargador de arranque que permite cargar programa y datos al momento del reset por medio de memorias de 8, 16 ó 32 bits o por el puerto serial. Los buses separados de programa y datos permiten realizar operaciones de lectura/escritura en paralelo. Un caché de instrucciones de 64 X 32 bits es proporcionado para almacenar secciones de código que se repiten frecuentemente, lo cual reduce en gran medida el número de accesos fuera del chip.

El C31 proporciona una interfaz externa: el bus primario (figura 4.7) [TI: 1999]. Esta interfaz consiste de un bus de datos de 32 bits, un bus de direcciones de 24 bits y un conjunto de señales de control. Direcciona memoria de programa/datos externa o espacio de entrada/salida. La tarjeta PCI la utiliza para acceder a los datos que se encuentran en la memoria RAM.

El DSP soporta 4 interrupciones externas (INT3 – INT0), un número de interrupciones internas y una señal de reset externa no mascarable. Estas interrupciones se utilizan también para configurar el cargador de arranque. La tarjeta PCI utiliza la interrupción 1 para cargar el programa en la memoria interna del DSP.

Dos banderas externas de entrada/salida, XF0 y XF1, pueden ser configuradas como terminales de entrada o salida bajo el control del software. La bandera XF0 es utilizada para señalar al DSP que debe iniciar el procesamiento de la señal.

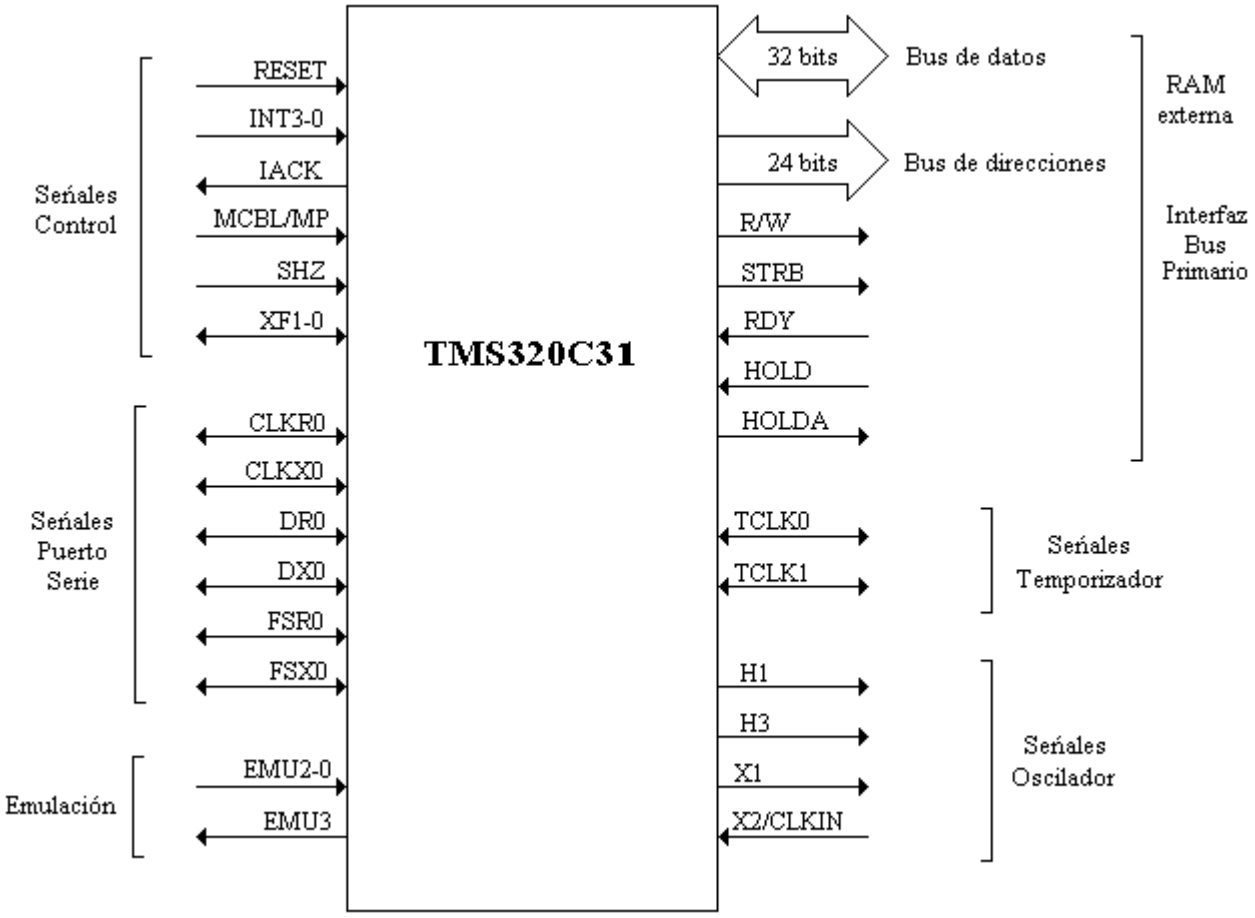

Figura 4.7. Señales del TMS320C31.

Para mayor información consultar el manual de usuario del DSP TMS320C3X que se puede obtener en la página de Internet de Texas Instruments www.ti.com.

4.1.3.1 Configuración del DSP en la tarjeta

El DSP TMS320C31 utiliza un bus de datos de 32 bits y un bus de direcciones de 24 bits. Para evitar problemas de colisiones en los buses de la tarjeta que llegan a la memoria RAM, las terminales del bus de datos del C31 se conectan a dispositivos FCT245 que controlan el acceso al bus de datos de la tarjeta y las terminales del bus de direcciones del C31 se conectan a dispositivos FCT573 que controlan el acceso al bus de direcciones de la tarjeta. La terminal R/W controla la dirección de los datos en los dispositivos FCT245. Los ocho bits más significativos del bus de direcciones A24-A31 se conectan a un CPLD para utilizar las señales en la lógica de control.

El C31 puede ser configurado para que trabaje en modo microprocesador o en modo microcomputadora, en este sentido se ha conectado la terminal MCBL/MP a VCC para que

trabaje en modo microcomputadora, que hace uso del cargador de arranque para cargar un programa desde un dispositivo externo, en este caso, la memoria RAM de la tarjeta.

La terminal RDY se conecta a tierra para indicar que no se van a introducir estados de espera por hardware. La terminal HOLD se usa para bloquear al C31, en la tarjeta, esta terminal se conecta a VCC para que el dispositivo no quede bloqueado y la terminal HOLDA queda desconectada, pues indica cuando se ha bloqueado el sistema. La terminal SHZ se conecta a VCC para que las terminales del DSP siempre estén habilitadas, esta terminal en estado bajo pone todas las señales del DSP en alta impedancia. Las terminales INT1 y RESET son controladas por la lógica programada en los CPLD's para cargar un programa en la memoria interna del DSP. Las terminales INT0, INT2 e INT3 son conectadas a VCC para evitar que se produzca una interrupción por medio de ellas. La terminal XF0 es controlada por un CPLD para indicar el inicio del procesamiento y la terminal XF1 se conecta a VCC para tener un estado conocido en esa entrada. La terminal IACK queda desconectada.

Todas las terminales del puerto serie (CLKR0, CLKX0, DR0, DX0, FSR0 y FSX0) y las de los temporizadores (TCLK0 y TCLK1) son conectadas a VCC para tener un estado conocido. Estos periféricos no son utilizados por la tarjeta. La terminal H1 se conecta a la terminal ADCLK del dispositivo PCI S5920 para servir como reloj del bus Add-On (ver funcionamiento del dispositivo S5920 más adelante). La terminal H3 se conecta al circuito divisor de frecuencia para generar la señal de reloj que se utiliza en la etapa de digitalización. Estas dos terminales, H1 y H3, proporcionan una señal de reloj de 20MHz. Las terminales X1 y X2/CLKIN están configuradas para trabajar con un cristal de 40MHz y generan la señal de reloj para el DSP.

Las terminales R/W, STRB y H1 se conectan a los CPLDs para ser utilizadas en la lógica de control. R/W indica si se desea hacer una lectura o una escritura por medio del bus primario. STRB es la señal que valida los datos en el bus primario, si esta señal esta habilitada, significa que los datos en el bus de datos y en el de direcciones son válidos. H1 es una señal de reloj de 20MHz con la que están sincronizadas las señales del bus primario.

Finalmente, las terminales EMU0-EMU2 se conectan a VCC para tener un estado conocido y la terminal EMU3 queda desconectada. Estas terminales no son utilizadas por la tarjeta.

4.1.4 Interfaz PCI

Después de que el DSP ha terminado de procesar la señal y ha obtenido el espectro en frecuencia de la misma, los datos quedan almacenados en la memoria RAM de la tarjeta, listos para ser transferidos a la computadora. En este momento se hace uso del controlador PCI de la tarjeta, que nos ayuda a enviar los datos que se encuentran en la memoria RAM a un espacio en la memoria de la computadora, para ser utilizados por el software de aplicación.

El bus PCI fue definido por Intel para crear un estándar en la comunicación de dispositivos de una computadora [SA: 1995], [SW: 1998]. A continuación se mencionan algunas características del bus PCI, que muestran la importancia de esa interfaz en la actualidad, ya que es el estándar en las tarjetas de expansión para computadoras personales.

- Independiente del procesador. Los componentes diseñados para el bus PCI son específicos para PCI, no para el procesador, de esta manera, se aísla el diseño del dispositivo de los cambios en las actualizaciones de los procesadores.
- Soporte hasta para 256 dispositivos PCI funcionales por bus PCI. Aunque una implementación típica del bus PCI soporta aproximadamente 10 cargas eléctricas, cada paquete de dispositivo PCI puede contener hasta 8 funciones PCI diferentes. El bus PCI soporta lógicamente hasta 32 paquetes de dispositivo PCI físicos, para un total de 256 funciones PCI posibles por bus PCI.
- Soporte hasta para 256 buses PCI. La especificación proporciona soporte hasta para 256 buses PCI.
- Bajo consumo de poder. Una de las ventajas principales del diseño de la especificación PCI es la creación de un diseño del sistema que maneja tan poca corriente como es posible.
- Ráfagas utilizadas para todas las transferencias de escritura y lectura. Soporta una tasa de transferencia pico de 132 Mbytes por segundo para transferencias tanto de escritura como de lectura, una tasa de transferencia pico de 264 Mbytes por segundo para transferencias de 64 bits y una tasa de transferencia de hasta 528 Mbytes por segundo en un bus PCI de 66MHz.
- Velocidad del bus. La revisión 2.0 soporta una velocidad del bus de hasta 33MHz. La revisión 2.1 añade soporte para una operación del bus a 66MHz.
- Ancho de bus de 64 bits. Definición completa de una extensión para 64 bits.
- Acceso rápido. Tan rápido como 60ns.
- Soporte para maestro del bus. Un maestro PCI puede acceder a un objetivo que se encuentra en otro bus PCI abajo en la jerarquía del bus.
- Revisión de la integridad de la transacción. Revisión de la paridad en la dirección, comandos y datos.
- Tres espacios de direcciones. Espacio de direcciones para memoria, entrada/salida y configuración.
- Auto configuración. Especificación al nivel de bits de los registros de configuración necesarios para soportar la detección y configuración automática de periféricos.
- Tarjetas de expansión. La especificación incluye una definición de los conectores y de las tarjetas de expansión PCI, que pueden ser de tres tamaños.

Estas características hacen al bus PCI la arquitectura principal en el diseño de tarjetas de expansión para una computadora. Ver el Anexo B para una breve descripción de los ciclos de lectura y escritura del bus PCI.

4.1.4.1 Operación del dispositivo PCI S5920

El dispositivo que permite a la tarjeta comunicarse con el bus PCI es el S5920 de la compañía AMCC. Este circuito integrado permite una tasa de transferencia de 132 Mbytes por segundo, ya que maneja el reloj a 33MHz y un bus de datos de 32 bits [AM: 1998]. También, transforma las señales del bus PCI en un bus llamado Add-On que es más fácil de utilizar y que permite realizar transferencias sencillas y de ráfaga. Este bus puede tener un ancho de datos de 8/16/32 bits. La figura 4.8 muestra el diagrama a bloques del dispositivo.

Este dispositivo se encuentra en la tarjeta PCI, lo que nos permite instalar la tarjeta en una ranura de expansión de una tarjeta madre de computadora que sea de tipo PCI. Este dispositivo también

nos permite enviar y recibir mensajes, con los cuales podemos manipular el funcionamiento del hardware de la tarjeta, para decirle que inicie la conversión, o que inicie el procesamiento o que se va a realizar una transferencia de datos entre la tarjeta y la computadora. En este último caso, la tarjeta envía los valores obtenidos después de realizar la FFT (espectro en frecuencia) a la computadora y ésta los requiere para graficar el espectro en la pantalla del monitor.

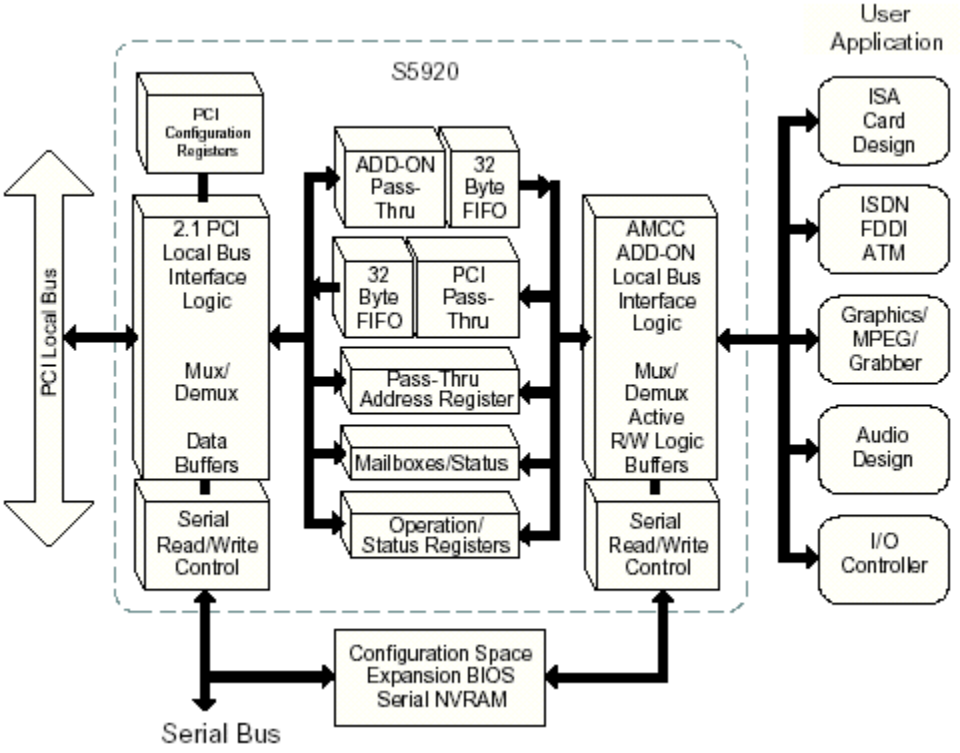

Figura 4.8. Diagrama a bloques del S5920.

El S5920 proporciona hasta 4 bloques de memoria o de entrada/salida a los cuales se les puede definir el tamaño, estos bloques son llamados regiones Pass-Thru. Las transferencias de datos a través de estas regiones pueden ser realizadas directamente al bus Add-On o por medio de dos FIFOs de 32 bytes.

Este controlador también cuenta con dos registros buzón de 32 bits para transferencias de datos adicionales o de comandos definidos por el usuario. Se puede examinar cada buzón para saber si está lleno o vacío, a un nivel de bytes, a través del registro de estado del buzón. El S5920 proporciona varios pines para los datos y para el control de los buzones que permiten realizar escrituras y lecturas directamente por hardware. Además, existe un pin dedicado para una interrupción del bus Add-On al bus PCI.

Finalmente, este circuito integrado da soporte para una memoria nvRAM (RAM no volátil) serial. Esto permite que el diseñador configure al dispositivo durante la inicialización, es decir, la nvRAM contiene los datos necesarios que se cargarán en los registros de configuración PCI. Esta memoria también puede servir como un BIOS de expansión.

Todos los dispositivos PCI deben proporcionar un conjunto de registros llamados registros de configuración PCI. Éstos son leídos por el BIOS del sistema durante la inicialización y contienen información como fabricante, tipo de dispositivo, requerimientos de memoria y otros. Estos registros son cargados con valores por defecto o con los datos que se encuentran en la memoria serial.

Los accesos a las regiones Pass-Thru pueden ejecutar ciclos de bus PCI en tiempo real o a través de una FIFO interna. Las operaciones en tiempo real permiten al bus PCI escribir o leer directamente a los recursos del bus Add-On. El S5920 permite declarar hasta cuatro regiones Pass-Thru. Cada región puede ser definida como de 8, 16 ó 32 bits para espacio en memoria y de 32 bits para el espacio de entrada/salida. Cuando se requiere una escritura o una lectura del bus PCI, el dispositivo activa unos pines para señalarlo. La lógica de control del usuario decodifica estas señales para saber si es una lectura o una escritura, cuáles bytes están habilitados, qué región Pass-Thru es la indicada y si es un ciclo de acceso sencillo o de ráfaga. El canal Pass-Thru incorpora dos FIFOs de 32 bytes, una está dedicada para lecturas y otra para escrituras. Habilitando las FIFOs se pueden aceptar transferencias con ciclos de espera. El bus Add-On puede funcionar en dos modos: activo o pasivo. En el modo pasivo el usuario debe controlar los pines del circuito integrado para escribir o leer datos. En el modo activo, una máquina de estados interna controla al bus Add-On.

4.1.4.2 Funcionamiento de la interfaz PCI de la tarjeta

La tarjeta está configurada para utilizar un ancho de bus de usuario de 32 bits. Se pueden realizar lecturas y escrituras con un ciclo de acceso de ráfaga a través de una región Pass-Thru, también se pueden enviar y recibir comandos hacia y desde la tarjeta por medio de un buzón. El bus Add-On queda configurado para ser utilizado en modo activo, dejando así que el S5920 controle las transferencias de datos en las regiones Pass-Thru.

Durante la inicialización, los registros de configuración PCI de la tarjeta se cargan con los valores que se encuentran en la memoria serial AT24C04 conectada a las terminales SCL y SDA del S5920. A continuación se describe la configuración, de los registros PCI (figura 4.9), que se requiere para que funcione adecuadamente el dispositivo en la tarjeta (los valores de todos los registros de configuración en la memoria serial se encuentran en el Anexo C):

- El registro de identificación del vendedor (VID) tiene un tamaño de 2 bytes. Este registro sirve para identificar al fabricante del dispositivo PCI, en este caso se carga con el valor 10E8h, que es el número que identifica a la compañía AMCC. Este número es asignado a cada compañía por el PCI Special Interest Group.
- El registro de identificación de dispositivo (DID) tiene un tamaño de 2 bytes. Asigna un número de referencia para el dispositivo, con este número se identifica al dispositivo dentro del sistema. El dispositivo utiliza el número de referencia 5920h.
- El registro de dirección base 0 (BADR0) tiene un tamaño de 4 bytes. Este registro se utiliza para tener acceso a los registros internos del S5920, como los buzones y los registros de estado. Este registro se carga con un valor de 10E8FF81h que define un espacio de entrada/salida de 128 bytes.

- El registro de dirección base 1 (BADR1) tiene un tamaño de 4 bytes. Este registro define la dirección de memoria base que se va a utilizar para acceder a la memoria de la tarjeta. Se carga con un valor de FFFFFF01h, que define un espacio de entrada/salida de 256 bytes.

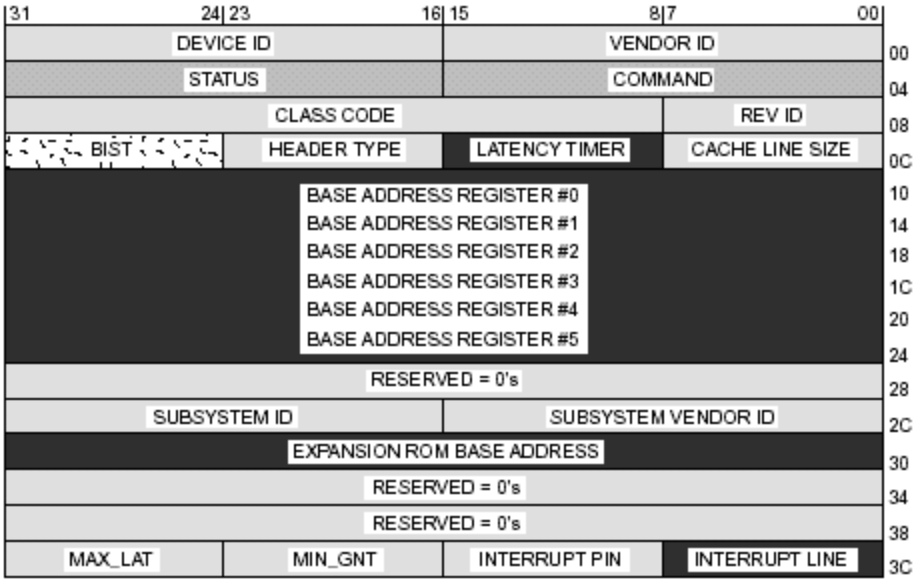

Figura 4.9. Registros de configuración PCI en el S5920.

A continuación se describe la configuración de las terminales (figura 4.10) necesarias para el funcionamiento adecuado del dispositivo S5920:

- La terminal MDMODE (modo de los datos del buzón) se conecta a GND para permitir que las terminales MD[7: 0] sean tanto de entrada como de salida.
- La terminal LOAD# se conecta a VCC por medio de un resistor para asegurar que las terminales MD[7: 0] siempre sean salidas, salvo cuando la lógica de control conduzca esta entrada a un nivel bajo para permitir la escritura de comandos en el buzón. Al conectar la terminal MDMODE a un estado bajo y a la terminal LOAD# a un estado alto, las terminales MD[7: 0] representan los datos del byte 3 del buzón de salida PCI, esto permite enviar comandos desde el bus PCI hacia el bus Add-On. Cuando la terminal LOAD# pasa a un estado bajo, las terminales MD[7: 0] representan los datos del byte 3 del buzón de entrada PCI, esto permite a la computadora recibir comandos desde el bus Add-On.
- La terminal PTMODE (modo del Pass-Thru) configura la operación del canal de datos Pass-Thru. Esta terminal se conecta a GND para que la región opere en modo activo, esto es, que el dispositivo controle las señales y los datos en el bus.
- La terminal PTRDY# se conecta a un estado alto para indicar que no se introducirán estados de espera externos en la transferencia de datos.
- La terminal DQMODE define el ancho del bus DQ cuando se opera en modo pasivo o cuando se desea escribir o leer los registros del dispositivo por el bus Add-On. En este caso no se realiza ninguna de las operaciones anteriores, pero se conecta la terminal a un estado bajo para que el ancho del bus sea de 32 bits.

- Las terminales ADR[6: 2] sirven para seleccionar a qué registro interno se desea tener acceso. En este caso no se realiza esta operación, pero se conectan estas terminales a un estado bajo.
- Las terminales BE[3: 0]# proporcionan la habilitación de bytes individuales en las operaciones de escritura y de lectura de registros. Estas terminales se conectan a un estado bajo para habilitar los cuatro bytes.

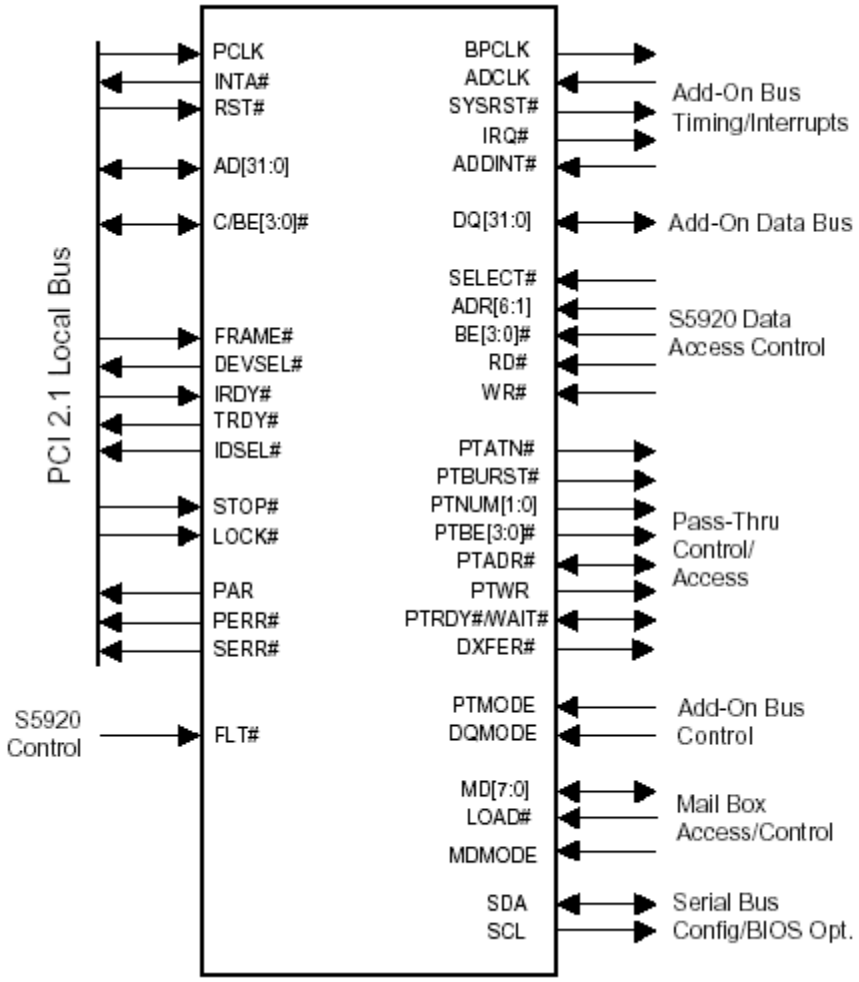

Figura 4.10. Terminales del dispositivo S5920.

- Las terminales SELECT#, WR# y RD# se utilizan para escribir o leer los registros desde el bus Add-On o para transacciones en el modo pasivo. La tarjeta no realiza las operaciones anteriores, por eso, las terminales se conectan a un estado alto, que inhabilita a estas señales.
- La terminal BPCLK queda desconectada y la terminal ADCLK se conecta a la terminal H1 del DSP para asegurar que el bus Add-On trabaje a una frecuencia de 20MHz.
- La terminal ADDINT# se conecta a un estado alto para asegurar que el bus Add-On no realizará una petición de interrupción al bus PCI. La terminal IRQ# queda desconectada.
- La terminal FLT# se conecta a un estado alto para asegurar que todas las terminales del dispositivo no serán flotadas.
- Las terminales DQ[31: 0] forman el bus de datos de 32 bits que se utiliza en las transferencias de datos por medio de la región Pass-Thru. Estas terminales se conectan a dispositivos

74FCT245 que controlan el bus de datos de la tarjeta, la señal PTWR controla la dirección de los datos en estos dispositivos.

- Las terminales PTBURST#, PTNUM[1: 0], PTBE[3: 0]# y PTADR# quedan desconectadas.
- Las terminales SYSRST#, PTATN#, PTWR y DXFR# se conectan a los CPLDs para ser utilizadas por la lógica de control.
- Las terminales MD[7: 0] son bidireccionales, por lo que se conectan a dos latches, uno de escritura y otro de lectura. Estos latches son controlados por una máquina de estados programada en un CPLD.

4.2 Diseño de la lógica de control

Como dijimos en la sección 4.1, la lógica de control se encarga de administrar la secuencia de operaciones que debe realizar la tarjeta PCI. Tales operaciones pueden ser la inicialización del DSP o del ADC, la señalización de inicio de la conversión o del procesamiento, o que se va a realizar una transferencia por medio del bus PCI.

4.2.1 Ecuaciones de control

En esta sección se describe la lógica de control implementada en dos CPLDs XC9536. En el primer CPLD se tienen las siguientes señales:

Entradas

- MMD0, MMD1, MMD2, MMD3, MMD4.
- DSPA16, DSPA17, DSPA18, DSPA19, DSPA20, DSPA21, DSPA22, DSPA23.
- STRB, RW.
- PTATN, PTWR, DXFR.

Salidas

- XF0, RESET, DSPOE, INT1.
- GDMR, WE
- COND1, COND2, COND3.
- ADCRES
- GDB12, GDB13, GDB14, GDB15.
- PCIOE.

Tabla 4.1. Comandos que envía la computadora a la tarjeta.

Valor binario	Comando
00000	No hay comando
00011	Transfiere programa desde la computadora a la RAM de la tarjeta
00101	Reset al DSP
00110	Habilita INT1

01001	Reset al ADC
01010	Digitalizar señal de entrada
01100	Transfiere datos desde la RAM de la tarjeta a la computadora
10001	Reset al generador de direcciones
10010	Procesar la señal en el DSP

Las señales MMD0-4 pertenecen a los 5 bits menos significativos del byte 3 del buzón de salida PCI (los tres bits restantes se usan para escoger una frecuencia de muestreo), que se utilizan para enviar mensajes de la computadora a la tarjeta, la tabla 4.1 muestra el significado de los comandos que pueden utilizarse.

Las señales DSPA(n), STRB y RW provienen del DSP (ver figura 4.7) y las señales PTATN, PTWR y DXFR provienen del circuito controlador del bus PCI (ver figura 4.10).

La señal XF0 se utiliza para indicarle al DSP que debe comenzar a procesar la señal digitalizada. Después de que el convertidor ha terminado de digitalizar la señal de entrada, la máquina de estados (implementada en el segundo CPLD) avisa a la computadora que es el momento de iniciar el procesamiento de la señal en el DSP. Mientras tanto el DSP está revisando de manera constante la terminal XF0 que le va a indicar el inicio del procesamiento. La terminal XF0 va a estar en un estado bajo si no hay procesamiento, y debe estar en estado alto durante el procesamiento. Ya que la computadora sabe que es el momento de procesar los datos, envía el comando correspondiente, en este caso 12h (ver tabla 4.1). A continuación se muestra la ecuación que describe la habilitación de la terminal XF0:

XF0 = MMD1 * MMD4 * /MMD2 * /MMD0 * /MMD3

La señal de RESET se utiliza para inicializar el DSP y además se utiliza en la secuencia del cargador de programa. Para cargar un programa en la memoria del DSP y que lo empiece a ejecutar, se requiere habilitar la señal de reset del DSP y posteriormente habilitar una de las interrupciones del DSP. Después de esto, el DSP comienza a leer una región de la memoria externa, donde debe encontrarse el programa a ser cargado. La señal de reset del DSP es activa en un nivel bajo, por lo que su ecuación resultante dado el comando 05h es:

/RESET = /MMD1 * /MMD4 * MMD2 * MMD0 * /MMD3

INT1 es la interrupción que se utiliza en la secuencia para cargar el programa de procesamiento en la memoria del DSP. Después de que se habilita la señal de reset, el DSP espera la habilitación de una señal de interrupción. Pueden ser cuatro las posibilidades de arranque, dependiendo qué número de interrupción se habilite. Con la INT1 se comienza a buscar el programa a partir de la dirección 400000h. Después de que se termina de cargar el programa, el DSP comienza a ejecutarlo automáticamente. INT1 es activa en bajo y el comando que envía la computadora es el 06h, entonces, su ecuación es:

/INT1 = MMD1 * /MMD4 * MMD2 * /MMD0 * /MMD3

DSPOE es la señal que controla los buses externos del C31, cuando está en estado bajo, el DSP tiene acceso a los datos y direcciones de la memoria RAM de la tarjeta, en estado alto, los buses

del DSP no pueden tener comunicación con la memoria de la tarjeta. Esta señal puede permitir al C31 el uso de la memoria RAM de la tarjeta cuando requiera un dato que se encuentre en el rango de 400000h a 40FFFFh. Este rango representa 64Kb de memoria RAM que tiene la tarjeta. Una dirección que se encuentre fuera de este rango no tendrá acceso a la memoria RAM. La señal STRB del DSP valida los datos en el bus de direcciones, la ecuación para DSPOE es:

/DSPOE = /STRB * /DSPA18 * /DSPA17 * /DSPA16 * DSPA22 * /DSPA23 * /DSPA20 * /DSPA21 * /DSPA19

La señal GDMR inicializa el generador de direcciones. Al inicio de una transferencia de datos entre la tarjeta y la computadora, o al inicio de la digitalización, el generador de direcciones TC74VHC4040 debe tener sus terminales en cero, esto se logra poniendo en estado alto su terminal MR, el comando de la computadora es 11h:

GDMR = /MMD1 * MMD4 * /MMD2 * MMD0 * /MMD3

WE es la señal que controla a la terminal del mismo nombre de las memorias. Esta terminal en estado bajo, indica una escritura en las memorias, en estado alto indica una lectura de las memorias. Esta terminal, normalmente va a estar en un estado bajo, es decir, de escritura, pero va a pasar a un estado alto cuando se cumpla cualquiera de las dos opciones siguientes. La primera, cuando el DSP desee leer la memoria RAM, en este caso su terminal STRB pasa a un estado bajo y su otra terminal RW queda en estado alto. La segunda, cuando el controlador PCI lea de las memorias, entonces sus terminales PTATN y PTWR pasan a estado bajo. Esta es la ecuación que controla la terminal WE:

WE = /STRB * RW + /PTATN * /PTWR

La señal COND1 es muy similar a la señal DSPOE, indica que se está direccionando una parte de la memoria RAM de la tarjeta. Cuando ocurre esto, COND1 pasa a un estado alto. Esta señal se ocupa en la lógica implementada en el segundo CPLD.

COND1 = /STRB * /DSPA18 * /DSPA17 * /DSPA16 * DSPA22 * /DSPA23 * /DSPA20 * /DSPA21 * /DSPA19

Cuando el DSP termina de cargar el programa de procesamiento en su memoria interna, se lo debe indicar a la máquina de estados, para que esta a su vez se lo indique a la computadora. Una vez que se ha terminado de cargar el programa en el DSP comienza su ejecución, el programa hace una escritura a la dirección 500000h. Aunque el programa cree que escribe a esta localidad de memoria, en realidad no lo hace, simplemente la lógica de control detecta que se pretendió escribir a esa dirección y la interpretación que le da es que el programa se ha cargado exitosamente en la memoria interna del DSP. Puesto que la máquina de estados se implementa en el segundo CPLD, esta señal es una entrada para el segundo CPLD.

COND2 = /STRB * DSPA22 * /DSPA23 * DSPA20 * /DSPA21

Una vez que el DSP ha terminado de procesar la señal, debe informar a la computadora que los datos ya están listos para ser leídos. El DSP escribe un dato en la dirección 600000h para indicar

que ha terminado el procesamiento. La señal COND3 pasa a un estado alto cuando la lógica de control detecta la dirección 600000h y entiende que ha finalizado el procesamiento.

COND3 = /STRB * DSPA22 * /DSPA23 * /DSPA20 * DSPA21

ADCRES es la señal de reset del convertidor. Algunas veces es conveniente inicializar el convertidor, sobre todo cuando se cambia de frecuencia de muestreo. La computadora envía el comando 09h para habilitar la señal de reset del ADC que es activa en bajo.

/ADCRES = /MMD1 * /MMD4 * /MMD2 * MMD0 * MMD3

GDB12, GDB13, GDB14 y GDB15 son los 4 bits más significativos del generador de direcciones para el controlador PCI y el convertidor. La lógica pone en cero las señales GDB13, GDB14 y GDB15, porque no se requiere direccionar más memoria de la que se puede direccionar con 13 bits. GDB12 normalmente va a estar en cero, excepto cuando se esté digitalizando una señal. Los datos que se van obteniendo de la digitalización se van guardando a partir de la dirección 1000h de las memorias, que para el mapa de memoria del DSP sería la dirección 401000h. Entonces GDB12 pasa a estado alto cuando la computadora envíe el comando 0Ah.

GDB12 = MMD1 * /MMD4 * /MMD2 * /MMD0 * MMD3

La señal PCIOE habilita los dispositivos 74FCT245 que controlan el flujo de datos entre el controlador PCI y las memorias de la tarjeta. En estado bajo, esta señal permite la comunicación entre las terminales de datos de las memorias y el dispositivo S5920. Cuando el controlador PCI realiza la escritura o la lectura de un dato, su terminal DXFR pasa a estado bajo, entonces debe existir un acceso a las memorias y debe habilitarse la señal PCIOE.

PCIOE = DXFR

A continuación se describen las ecuaciones implementadas en el segundo CPLD. En este CPLD se tienen las siguientes señales:

Entradas

- MMD0, MMD1, MMD2, MMD3, MMD4.
- GDCB11, H1, ADCDAV.
- SYSRST, PTATN, DXFR.
- COND1, COND2, COND3.

Salidas

- CSL, CSH.
- ADCOE.
- GDCP, GDOE.
- COM0, COM1, COM2.
- COMOE, LLE, LOAD.

GDCB11 es el bit 11 del generador de direcciones y se utiliza para parar la digitalización de la señal de entrada. SYSRST proviene del controlador PCI y se utiliza para inicializar la máquina de estados que contiene este CPLD. H1 proviene del DSP. ADCDAV proviene del ADC y se utiliza para saber cuándo hay datos disponibles a la salida del convertidor.
La señal CSH controla la terminal CS de la memoria RAM que contiene los datos de la parte alta de cada palabra de 32 bits. Esta terminal es activa en bajo. CS valida los datos en el bus de direcciones y en el bus de datos de la memoria. Para que esta señal sea habilitada, se necesita que se cumpla cualquiera de dos condiciones. La primera, que la terminal DXFR del S5920 pase a estado bajo, indicando la transferencia de un dato por el bus PCI. La segunda, que el DSP requiera leer o escribir datos en la memoria, con esto, la terminal COND1 pasa a estado alto y además debe ocurrir sólo en el nivel alto de un ciclo de reloj de H1, porque en el nivel bajo se actualizan las direcciones y con esto aseguramos que la dirección ya está en el bus cuando se habilitan los datos.

CSH = /H1 * DXFR + DXFR * /COND1

CSL controla la terminal CS de la memoria RAM con los datos de la parte baja de cada palabra de 32 bits. Esta señal es muy similar a la señal CSH, pero añade una tercera condición para su habilitación: se puede activar cuando el convertidor escriba en la memoria. Como el convertidor proporciona un dato de 16 bits, sólo se utiliza una memoria para guardar ese dato, la memoria con los datos de la parte baja. Entonces esta opción requiere que la computadora envíe el comando 0Ah, que la señal ADCDAV esté en estado alto y que la terminal GDCB11 no tenga el valor '1'. 0Ah es el comando de digitalización de datos, cuando ADCDAV está en alto, significa que los datos a la salida del convertidor son válidos y GDCB11 debe estar en cero porque sólo se requieren 2048 muestras, con el valor de '1' significa que ya se ha obtenido ese número de muestras.

/CSL = /DXFR + H1 * COND1
+ MMD3 * ADCDAV * MMD1 * /MMD4 * /MMD0 * /GDCB11 * /MMD2

La señal ADCOE controla los dispositivos 74FCT573 que están conectados a la salida del convertidor para el acceso al bus de datos de la memoria. Esta señal es activa en bajo y requiere que la computadora envíe el comando 0Ah.

/ADCOE = MMD3 * MMD1 * /MMD4 * /MMD0 * /MMD2

GDCP es la señal de reloj del generador de direcciones. El generador de direcciones se va a incrementar cuando ocurra un flanco de bajada en su terminal CP. Este flanco se va a generar cuando la terminal DXFR se inhabilite, es decir, cuando haya terminado de hacer la transferencia del dato actual y pase de un estado bajo a un estado alto. Otra posibilidad para generar este flanco es cuando se esté digitalizando una señal y la terminal ADCDAV pase de estado alto a estado bajo, durante el estado alto se guardan los datos en la memoria y en el estado bajo se actualizan los datos y es cuando el generador de direcciones debe incrementarse.

GDCP = /DXFR + MMD3 * ADCDAV * MMD1 * /MMD4 * /MMD0 * /GDCB11 * /MMD2

La señal GDOE controla el acceso del generador de direcciones al bus de direcciones de la memoria, esta señal controla la terminal OE de los dispositivos 74FCT573 que se encuentran entre el generador y la memoria. Normalmente esta señal está en estado bajo, es decir, activa y con esto se tiene siempre en el bus de direcciones de la memoria los datos del generador de direcciones. Sólo cuando el DSP requiera el uso de la memoria, esta señal va a pasar a un estado alto, aislando los datos del generador.

GDOE = COND1

Las señales COMOE, LLE y LOAD se utilizan para configurar la lectura y escritura de las terminales MD[7: 0] del controlador PCI. COM0-COM2 son los tres bits que se utilizan para enviar mensajes desde la tarjeta hacia la computadora. La tabla 4.2 muestra los comandos que la tarjeta le puede enviar a la computadora.

Tabla 4.2. Mensajes que envía la tarjeta a la computadora.

Valor binario	Comando
000	No hay comando
001	Se terminó la transferencia del programa desde la computadora hasta la memoria RAM de la tarjeta
010	Se ha cargado el programa en el DSP
011	Se ha terminado de procesar la señal en el DSP
100	Se ha terminado de digitalizar la señal de entrada

COM0, COM1, COM2, COMOE, LLE y LOAD son controladas por una máquina de estados programada en el CPLD. Básicamente, las señales COM0, COM1, y COM2 representan los bits de los mensajes que la tarjeta envía a la computadora. COMOE habilita el latch que permite conectar los bits COM(n) a las terminales MD[7: 0] del S5920 para escribir en el buzón. LLE habilita el latch que representa las señales MD[7: 0] del byte 3 del buzón de salida PCI, estas señales necesitan estar fijas mientras se realiza un ciclo de escritura en las terminales MD[7: 0]. La señal LOAD indica que se tienen que cambiar de dirección las señales MD[7: 0] para realizar una escritura al byte 3 del buzón de entrada PCI. A continuación se describe cómo la máquina de estados controla estas señales.

4.2.2 Máquina de estados

A continuación se presenta la máquina de estados que controla los bits COM0, COM1, COM2, COMOE, LLE y LOAD del CPLD dos. La figura 4.11 muestra un esquema de conexiones entre el controlador PCI y el CPLD que implementa la máquina de estados. Cuando la computadora envía un comando hacia la tarjeta, las líneas MD[0: 7] deben comportarse como salidas, entonces la entrada LOAD debe estar en estado alto, para esta configuración. La terminal LLE debe estar en estado alto para permitir el paso de los datos del bus MD hacia el bus MMD que va hacia el CPLD, mientras que la señal COMOE debe estar en estado alto para no permitir que las señales COM[0: 2] pasen al bus MD.

Cuando la máquina de estados desea enviar un comando hacia la computadora, las líneas del bus MMD deben quedar fijas; esto se logra poniendo en estado bajo la señal LLE. Las señales

COM[0: 2] deben pasar al bus MD, entonces COMOE pasa a un estado bajo para habilitar las señales y LOAD debe pasar a un estado bajo para indicar al controlador PCI que las terminales MD[0: 7] se deben comportar como entradas.

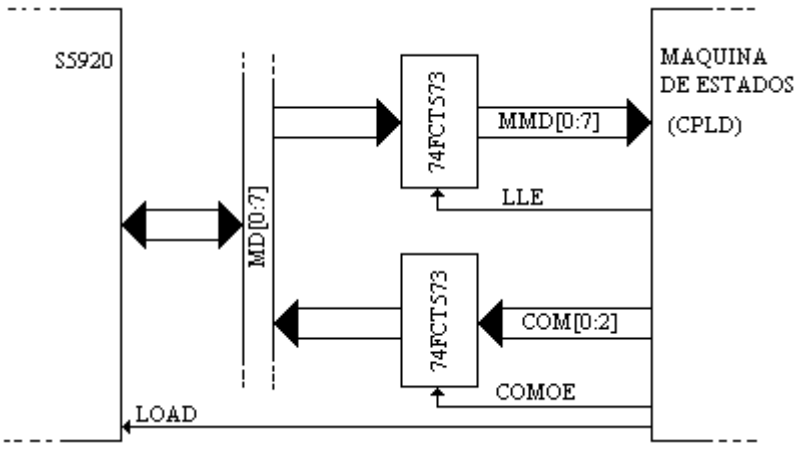

Figura 4.11. Envío y recepción de comandos entre la tarjeta y la computadora.

La figura 4.12 muestra las etapas de la máquina de estados y una breve descripción de cada una de ellas.

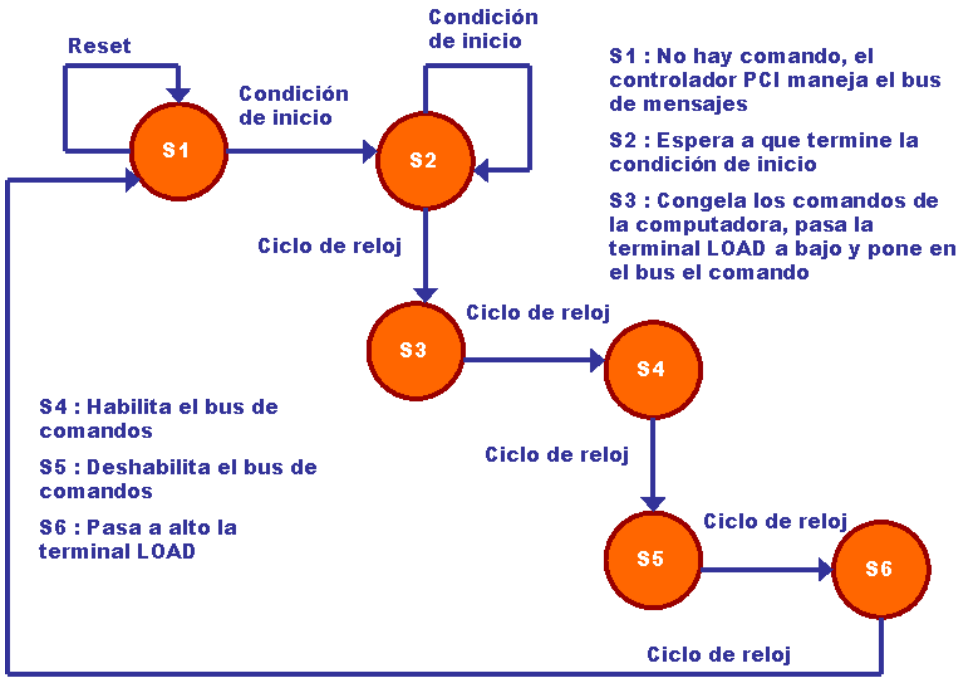

Figura 4.12. Máquina de estados.

La figura 4.13 muestra las señales que genera la máquina de estados para escribir un comando en el byte 3 del buzón de entrada PCI del controlador.

En el momento en que la máquina de estados desea enviar un comando hacia la computadora por medio del controlador PCI, las señales LLE y LOAD pasan de un estado alto a uno bajo para dejar fijas las señales MMD y avisar al S5920 que las terminales MD[0: 7] deben cambiarse de dirección para que ahora sean entradas. Las terminales COM[0: 2] contienen la información del comando deseado. En el siguiente ciclo de reloj, la terminal COMOE pasa de un estado alto a uno bajo para permitir que el comando esté listo en las terminales MD[0: 7] cuando llegue el siguiente ciclo de reloj, que es cuando el S5920 realiza la lectura del bus. Después de que el controlador ha leído el comando, la señal COMOE pasa a un estado alto para bloquear las señales COM[0: 2]. En el siguiente ciclo de reloj, la señal LOAD pasa a un estado alto, para volver a cambiar el sentido de las terminales MD[0: 7], y se comporten como salidas. Finalmente, en el siguiente ciclo de reloj, la señal LLE pasa a un estado alto, para que las señales MMD[0: 7] tengan el valor de las salidas MD[0: 7] y lleguen los comandos de la computadora a la tarjeta.

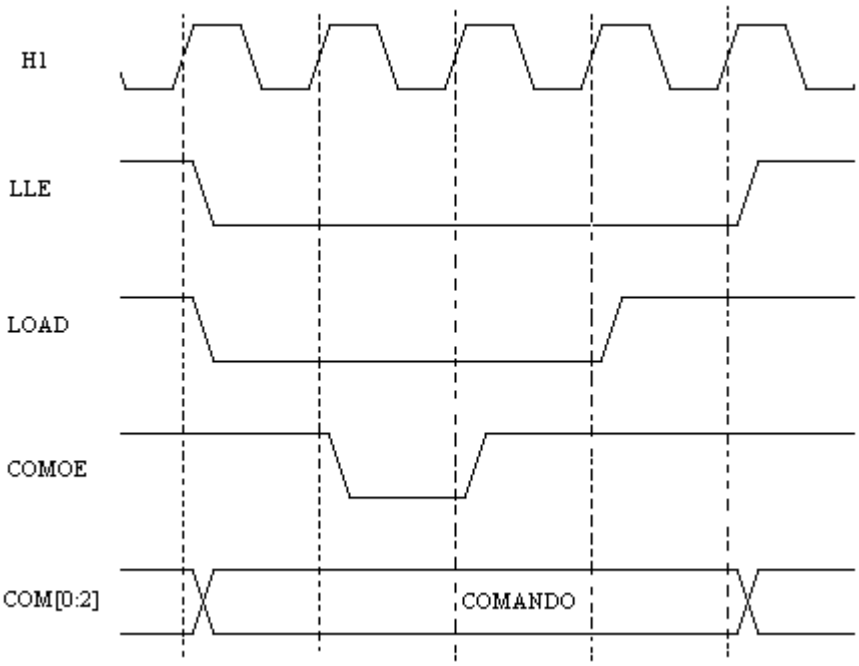

Figura 4.13. Señales que genera la máquina de estados.

Antes de que la máquina de estados genere estas señales, debe ocurrir algún evento que signifique el paso de un comando hacia la computadora (condición de inicio). En el caso del comando 01h que es para indicar que se ha terminado de transferir el programa desde la computadora hasta la memoria RAM de la tarjeta, el evento que indica la finalización de la transferencia es el paso de la señal PTATN a un estado alto. Para enviar el programa hacia la tarjeta, la computadora primero envía el comando de transferencia de programa con código 03h y después hace una transferencia PCI. El chip controlador manda la señal PTATN a un estado bajo cuando se hace la transferencia PCI, y cuando termina la transferencia, envía a PTATN a un estado alto. Entonces, cuando se tiene el comando de transferencia de programa y la señal PTATN pasa de un estado bajo a uno alto, la máquina de estados genera las señales necesarias para escribir el código 01h en el buzón, que indica que el programa se encuentra en la memoria de la tarjeta.

Una vez que el DSP se ha inicializado y ha recibido el programa desde la memoria RAM de la tarjeta, el programa es ejecutado y lo primero que hace es escribir en la localidad 500000h para que la máquina de estado genere las señales que escriben el comando 02h en el buzón por medio de las terminales COM[0: 2] y avisar a la computadora que ya se tiene el programa en la memoria interna del DSP.

El evento que genera el comando 03h es la finalización del procesamiento de la señal. Después de que el DSP terminó de procesar la señal, escribe en la dirección 600000h para que la máquina de estados genere la escritura del comando en el buzón. La máquina de estados también requiere que la computadora haya enviado el comando 12h de procesamiento de señal.

El último comando que genera la máquina de estados es el 04h, para la finalización de la digitalización de los datos. El generador de direcciones se incrementa 2048 veces para obtener el mismo número de muestras y después el bit 11 del contador TC74VHC4040 pasa de un valor de 0 a un valor de 1. Cuando la máquina de estados detecta que se está digitalizando una señal y el bit 11 se pone en 1, escribe el valor 04h en el buzón por medio de las señales COM[0: 2].

4.3 Implementación de la tarjeta

El circuito impreso de la tarjeta consta de 6 capas, la primera recibe el nombre de TOP, esta capa es superficial y es en la que se montan los dispositivos; algunos circuitos de la tarjeta, en especial los de montaje superficial, se sueldan en esta capa. Esta capa es esencialmente de pistas y pads para los circuitos. La capa llamada BOTTOM también es superficial, aquí se sueldan los circuitos con terminales Thru Hole y también se tienen pistas. Existen dos capas intermedias llamadas INNER1 e INNER2, estas también contienen pistas y se conectan con las demás capas por medio de vías (Thru Holes). Finalmente existen dos capas intermedias que se utilizan para las señales de potencia, PWR y GND, estas capas son planos de cobre que distribuyen estas señales por toda la superficie de la tarjeta.

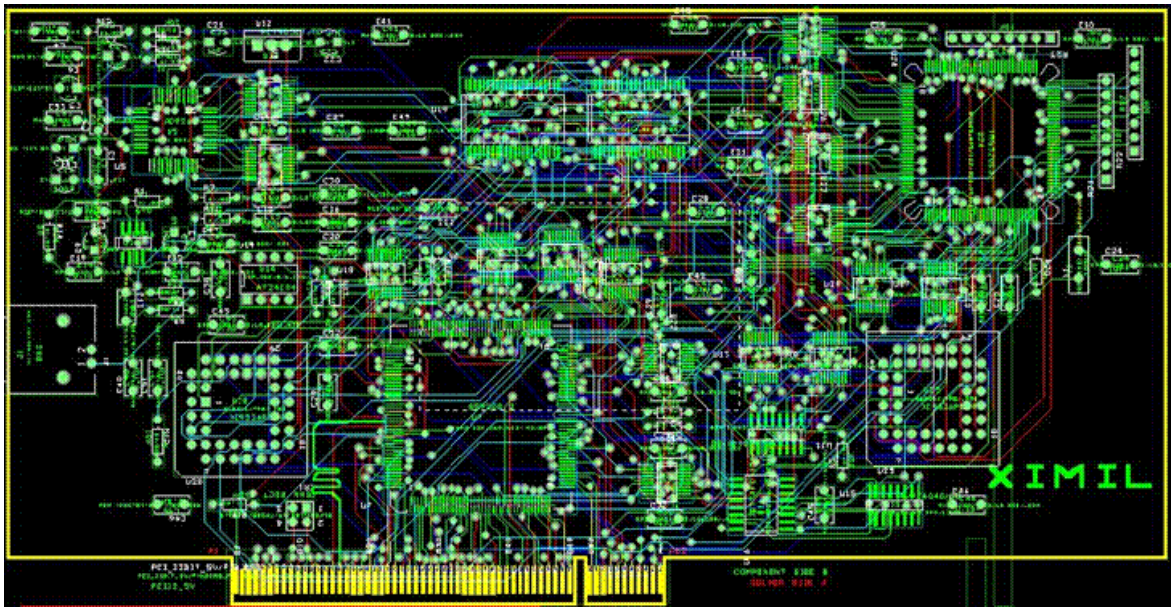

Figura 4.14. Diseño del circuito impreso de la tarjeta PCI.

La pista de la señal de reloj CLK del bus PCI debe tener una distancia de 2.5 pulgadas de longitud y las demás señales que conectan al peine del bus PCI con el controlador del bus PCI deben tener una longitud máxima de 1.5 pulgadas.

El ancho de las pistas es de 8 mils y el de la señal de reloj es de 20 mils. Las dimensiones de la tarjeta son aproximadamente de 8.3 X 4.2 pulgadas. El diseño del circuito impreso se hizo en OrCAD Layout Plus (Editor del Circuito Impreso) a partir del Netlist generado por OrCAD Capture (Editor del Diagrama Esquemático).

La figura 4.14 muestra una imagen del diseño de la tarjeta en OrCAD Layout.

La etapa de acondicionamiento consta de un amplificador operacional que convierte una señal referenciada a tierra a una señal diferencial. La figura 4.15 muestra el diseño del circuito impreso en OrCAD Layout, que consta de una sola superficie.

Figura 4.15. Diseño del circuito impreso de la etapa de acondicionamiento.

La tarjeta fue fabricada por la compañía Advanced Circuits, con material FR4 de 0.062 pulgadas de grosor y acabados SMOBC con máscara antisoldante. La figura 4.16 muestra la tarjeta con sus componentes.

Figura 4.16. Circuito impreso y componentes.

4.4 Resumen

En este capítulo se mencionó el diseño e implementación de la tarjeta PCI, la forma en que se utilizan los distintos dispositivos para obtener el espectro en frecuencia de una señal analógica. La secuencia de las señales en la tarjeta y el circuito que controla el acceso a los buses. También se habla de la programación de los CPLDs que se utilizan en la tarjeta y el diseño del circuito impreso de la misma. En el capítulo siguiente se habla sobre el software que se utiliza en la tarjeta y la computadora.

CAPÍTULO 5

DISEÑO DEL SOFTWARE DE ANÁLISIS ESPECTRAL

En el presente capítulo se describe el funcionamiento del software con que cuenta el analizador de espectro, éste incluye el programa de procesamiento para el DSP, el programa para el acceso al hardware y el programa que presenta el espectro en la pantalla de la computadora. La figura 5.1 muestra el diagrama a bloques del analizador de espectro y los bloques que aparecen resaltados indican las partes donde se realiza el procesamiento y la presentación del espectro, de acuerdo a lo descrito en la sección 3.2.2.

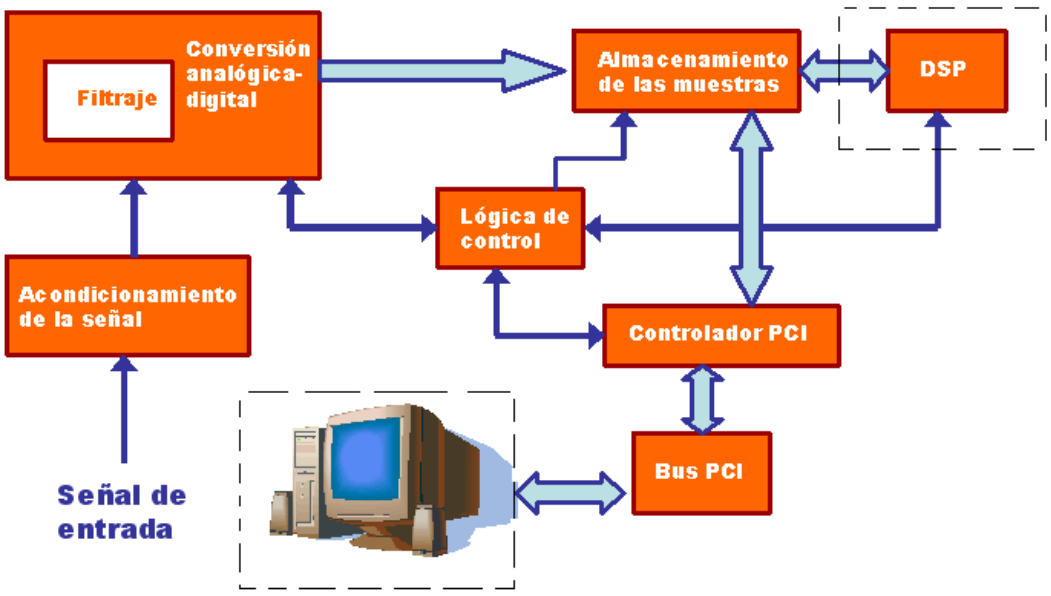

Figura 5.1. Bloques del software de análisis espectral

5.1 Programa de procesamiento para el DSP

La tarjeta PCI cuenta entre sus componentes con el DSP TMS320C31 de Texas Instruments, este procesador puede manejar números enteros y de punto flotante. Este dispositivo es el que va a realizar el procesamiento digital de la señal, es decir, dado un conjunto de muestras previamente almacenadas en la memoria de la tarjeta obtiene la estimación de la potencia del espectro de la señal. Para obtener la estimación de la potencia del espectro, se usa el método propuesto por Welch [WE: 1967] que se describe a continuación:

Dado un conjunto de muestras se procede a dividirlo en segmentos de igual tamaño, estos segmentos pueden estar o no traslapados, pero se recomienda que exista un traslape del 50% entre ellos. En la figura 5.2 se muestra un ejemplo de cómo se obtienen segmentos de 4 muestras cada uno, dado un conjunto de 16 muestras. Se obtienen 7 segmentos de 4 muestras cada uno traslapados entre sí en un 50%.

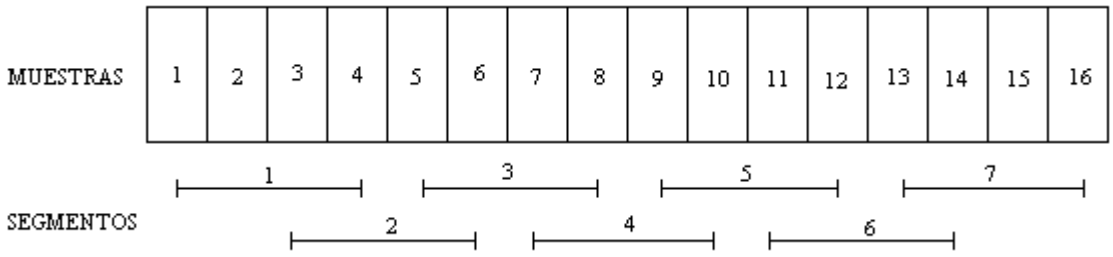

Figura 5.2. División del conjunto de muestras en segmentos.

Después de obtener los segmentos, cada segmento es multiplicado por una ventana de datos dada por:

$$W(j) = 1 - \left(\frac{j - \frac{L-1}{2}}{\frac{L+1}{2}} \right)^2$$ con j = 0, 1, ..., L - 1. L: tamaño de segmento

El siguiente paso es aplicar la FFT a cada uno de los segmentos y elevar al cuadrado cada uno de los coeficientes del resultado. En la siguiente etapa, cada coeficiente se divide entre el tamaño del segmento y entre el coeficiente U de la ventana:

$$U = \frac{1}{L} \sum_{j=0}^{L-1} W^2(j)$$ L: tamaño de ventana

Finalmente, la estimación de la potencia del espectro es el promedio de cada coeficiente de todos los segmentos.

El programa que se ejecuta en el DSP, considera el uso de 8 segmentos de 256 muestras cada uno con un traslape del 50% entre los segmentos. El algoritmo que se utiliza para implementar la FFT en el DSP es el de Decimación en Tiempo Radix-2. El programa completo se puede ver en el anexo D.

5.2 Controlador del dispositivo

Para tener acceso a los recursos que proporciona la tarjeta PCI que funciona como un analizador de espectro, tomando en cuenta que funcionará en computadoras que utilizan el

sistema operativo Windows, se requiere de un controlador de dispositivo que se comunique con el sistema operativo y éste a su vez con el hardware con rutinas de bajo nivel.

El diseño de controladores de dispositivo para el sistema operativo Windows plantea dos opciones. La primera consiste en el desarrollo del controlador utilizando el Kit de desarrollo de controladores que proporciona Microsoft, por este camino el desarrollo de controladores requiere de bastante experiencia en la programación de manejadores y además un gran conocimiento de la operación del núcleo del sistema. Sin embargo, los controladores desarrollados bajo este esquema tienen un alto rendimiento, ya que funcionan en el modo kernel.

La segunda opción consiste en utilizar un paquete como WinDriver que simplifica el desarrollo de los controladores, proporcionando una biblioteca de funciones que realizan el acceso al hardware en el modo usuario. La figura 5.3 muestra la arquitectura de WinDriver, donde se puede apreciar que la biblioteca que utiliza el usuario se comunica con el núcleo de Windriver en el modo kernel, para tener acceso al hardware desde este modo. También se observa que WinDriver cuenta con una conexión para que el usuario pueda acceder al núcleo de WinDriver por medio de un controlador de dispositivo en modo kernel.

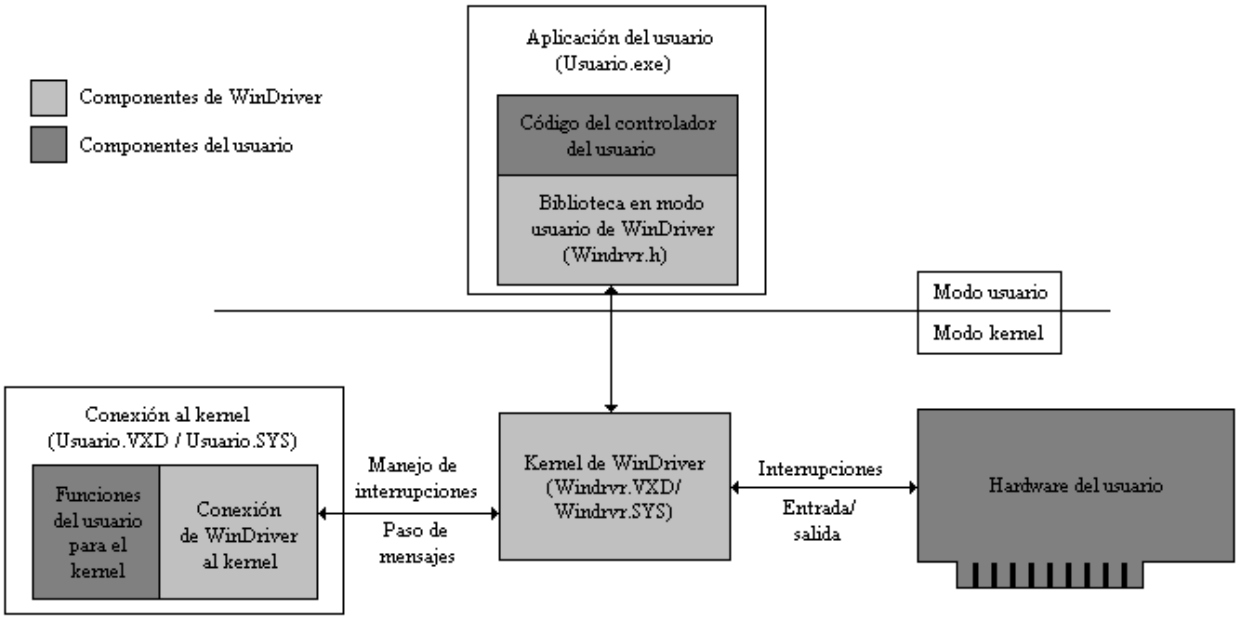

Figura 5.3. Arquitectura de WinDriver.

El software que acompaña a la tarjeta hace uso de la biblioteca proporcionada por WinDriver para lograr el acceso a los recursos del hardware. Existe un conjunto de funciones que dan soporte al dispositivo de AMCC S5933, este circuito tiene las mismas funciones que el S5920, pero añade otras. Sin embargo, para las funciones que son comunes para ambos controladores, la biblioteca de WinDriver para AMCC es compatible. A continuación se describen algunas funciones para el S5933 (se puede obtener más información en [KR: 2000]):

- AMCC_CountCards(). Cuenta el número de tarjetas en el bus PCI que tienen los identificadores de vendedor y de dispositivo dados en los parámetros.

- AMCC_Open(). Abre un manejador para la tarjeta AMCC.
- AMCC_Close(). Libera el manejador de la tarjeta.
- AMCC_IsAddrSpaceActive(). Verifica si existe espacio de direcciones en la tarjeta.
- AMCC_ReadRegDWord(). Lee un byte de un registro del S5933.
- AMCC_WriteRegDWord(). Escribe un byte a un registro del S5933.
- AMCC_ReadDWord(). Lee un byte desde el espacio de direcciones de la tarjeta.
- AMCC_WriteDWord(). Escribe un byte en el espacio de direcciones de la tarjeta.
- AMCC_ReadSpaceBlock(). Lee un bloque de datos de la tarjeta y lo almacena en un buffer.
- AMCC_WriteSpaceBlock(). Escribe un bloque de datos en la tarjeta desde un buffer.
- AMCC_IntIsEnable(). Verifica si están habilitadas las interrupciones.
- AMCC_IntEnable(). Habilita las rutinas de manejo de interrupciones.
- AMCC_IntDisable(). Deshabilita las rutinas de manejo de interrupciones.
- AMCC_ReadPCIReg(). Lee de los registros de configuración PCI.
- AMCC_WritePCIReg(). Escribe en los registros de configuración PCI.

La función AMCC_Open se utiliza para abrir un manejador de dispositivo para la tarjeta, tomando como parámetros un identificador donde se devolverá el manejador, el identificador del dispositivo PCI, el identificador del vendedor, el número de tarjeta y las opciones de configuración. Si la función logra con éxito abrir un manejador para la tarjeta, entonces devuelve TRUE, en caso contrario devuelve FALSE.

El identificador para el manejador es una variable de tipo AMCCHANDLE, este valor es pasado por referencia para que pueda ser modificado dentro de la función. El identificador del vendedor es de tipo DWORD y toma el valor de 0x10E8 que es el número que identifica a la compañía AMCC. El identificador del dispositivo es de tipo DWORD y tiene un valor de 0x5920, que identifica al dispositivo PCI. En el caso de que existan varias tarjetas con los mismos valores de vendedor y dispositivo, el número de tarjeta hace la diferencia; para este caso el valor es 0x0000 que indica que vamos a utilizar la primera tarjeta. El parámetro de opciones es de tipo DWORD y sirve para definir si la tarjeta utiliza interrupciones del sistema, en este caso no las usamos, por lo que el valor es 0x0000.

Al terminar de utilizar la tarjeta se debe cerrar el manejador de dispositivo obtenido con la función AMCC_Open, por eso, antes de terminar el programa se debe llamar a la función AMCC_Close pasándole como parámetro el identificador del manejador.

Las funciones AMCC_WriteRegDWord y AMCC_ReadRegDWord se utilizan para escribir y leer de los buzones de entrada y salida. Principalmente se utilizan para leer y escribir el byte 3 de los buzones, cuyo valor se ve reflejado en las terminales MD0-MD7 del S5920.

La función AMCC_WriteRegDWord toma como parámetro el manejador, el offset del registro y el dato que se quiere escribir. El offset es de tipo DWORD y toma el valor de 0x0C que identifica al registro OMB del S5920. El dato es de tipo DWORD y puede tomar los valores que se describen en la tabla 4.1, tomando en consideración que son para el byte 3, como ejemplo se muestra el valor para el comando "Reset al DSP", 0x05FFFFFF.

La función AMCC_ReadRegDWord toma como parámetros al manejador y el offset del registro. El offset es de tipo DWORD y su valor es 0x1C que identifica al registro IMB. Esta función retorna un dato de tipo DWORD que representa el valor del buzón de entrada. Este dato puede tomar los valores que se presentan en la tabla 4.2.

Las funciones AMCC_WriteSpaceBlock y AMCC_ReadSpaceBlock se utilizan para transferir bloques de memoria desde y hacia la tarjeta. Ambas toman como parámetro al manejador, el offset de la región Pass-Thru, el buffer de entrada o de salida, el número de bytes a transferir y la región Pass-Thru.

El offset es de tipo DWORD y por lo general toma el valor de 0x0000, ya que las transferencias siempre son desde la primera localidad de memoria. El buffer es de tipo PVOID y en el se transfieren los datos. El número de bytes es de tipo DWORD y como su nombre lo indica representa el número total de bytes que se van a transferir. La región Pass-Thru representa el espacio de direcciones de la tarjeta, en este caso toma el valor AMCC_ADDR_SPACE0 de tipo AMCC_ADDR.

Estas son las funciones que se utilizan principalmente para establecer la comunicación entre la tarjeta y la computadora.

5.3 Presentación del Espectro de la señal

El software que muestra el espectro de la señal en la pantalla de la computadora está programado en Borland C++ Builder. Proporciona una vista tanto de la señal en tiempo, como en frecuencia. Utiliza las funciones descritas en la sección anterior para controlar los recursos de la tarjeta y sincronizar los procedimientos. A continuación se describe la secuencia para obtener en la pantalla de la computadora el espectro de una señal muestreada:

1. Abrir un manejador de dispositivo para la tarjeta utilizando la función AMCC_Open.
2. Leer el archivo que contiene el programa que se va a ejecutar en el DSP.
3. Transferir el programa a la memoria de la tarjeta.
4. Inicializar el DSP para que lea el programa y lo ejecute, esto se hace mandando consecutivamente los comandos de reset e interrupción al DSP.
5. Habilitar el convertidor para obtener un número necesario de muestras.
6. Avisarle al DSP que las muestras están listas para ser procesadas.
7. Recibir en la computadora el resultado del procesamiento.
8. Desplegar en la pantalla la señal en el dominio del tiempo.
9. Desplegar en la pantalla la estimación de la potencia del espectro.
10. Cerrar el manejador del dispositivo con la función AMCC_Close.

Para abrir un manejador de dispositivo para la tarjeta se utiliza la función AMCC_Open, que necesita los números de identificación del vendedor y del dispositivo. El controlador PCI con que cuenta la tarjeta está configurado con un ID de vendedor de 10E8h que es el código de la compañía AMCC y con un ID de dispositivo de 5920h que es el código del controlador PCI. La

función AMCC_Open verifica si existe un dispositivo con estas características y devuelve un handle en caso afirmativo para ser usado en las demás funciones.

Una vez que sabemos que se encuentra la tarjeta en el sistema procedemos a transferir el programa que se debe ejecutar en el DSP. Abrimos el archivo que contiene el programa para el DSP y enviamos la información a la tarjeta utilizando la función AMCC_WriteSpaceBlock.

El siguiente paso es avisarle al DSP que ya puede leer y comenzar a ejecutar el programa. Primero se genera la señal de reset para DSP escribiendo el comando en el buzón del controlador PCI. Después se genera la señal de la interrupción 1 para el DSP de la misma manera. En este momento el DSP lee el programa que se encuentra en la memoria de la tarjeta, lo copia en su memoria interna y lo comienza a ejecutar.

El programa que se ejecuta en el DSP genera una condición de inicio para la máquina de estados, que sirve para indicarle a la computadora que ya se está ejecutando el programa. Después espera a recibir la autorización para comenzar a procesar las muestras.

La computadora sabe que ya está ejecutándose el programa en el DSP y habilita al convertidor para que obtenga un total de 2048 muestras. Cuando el ADC termina de obtener las muestras genera una condición de inicio para la máquina de estados, para que genere el comando que indica que ya se ha terminado el muestreo.

Después la computadora da la autorización al DSP para que comience a procesar las muestras, entonces el DSP aplica el método de Welch para obtener la estimación de la potencia del espectro. Antes de dar por terminado el procesamiento, el DSP convierte todos los datos de punto flotante que están en un formato propio del DSP al formato IEEE que se ocupa en la computadora. Cuando termina genera una condición de inicio para que la máquina de estados avise a la computadora que ha terminado el procesamiento.

Ahora es cuando la computadora lee la información de la memoria de la tarjeta con la función AMCC_ReadSpaceBlock para presentar tanto la señal en el dominio del tiempo como en el dominio de la frecuencia.

Se utilizan dos cajas de gráficos, una para la señal en tiempo y otra en frecuencia. La señal en tiempo muestra el voltaje por división y el tiempo por división. La señal en frecuencia muestra el rango de frecuencias del analizador y la potencia del espectro en decibeles.

Finalmente, se cierra el manejador de dispositivo de la tarjeta utilizando la función AMCC_Close.

El listado del programa que presenta las muestras en la pantalla de la computadora se puede ver en el anexo E. La figura 5.4 muestra un ejemplo de la salida del programa.

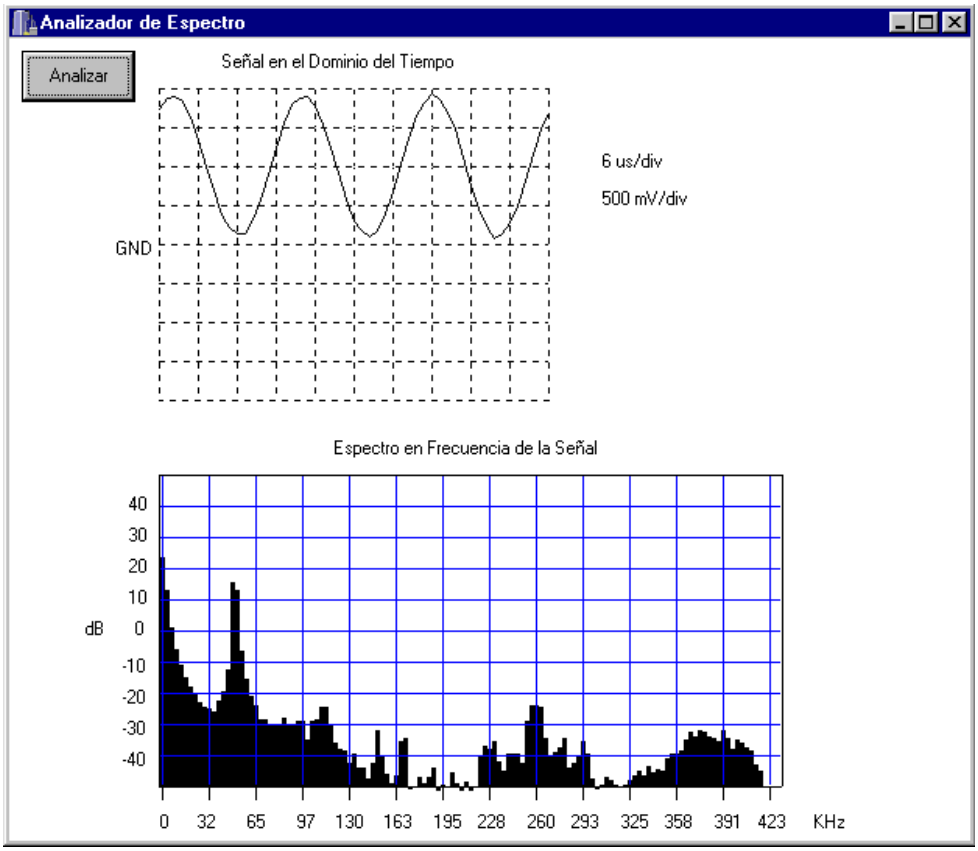

Figura 5.4. Pantalla que muestra el software de análisis espectral

5.4 Resumen

En el presente capítulo se describió la forma en que se realiza el procesamiento en el DSP para obtener la estimación de la potencia del espectro de una señal, por medio del método de Welch. También se mencionaron las funciones que proporciona la biblioteca de WinDriver para controlar los recursos de la tarjeta desde el sistema operativo Windows. Finalmente, se mostró la secuencia que se utiliza para mostrar el espectro de una señal en la pantalla de la computadora. En el siguiente capítulo se analizan algunas pruebas que se realizaron al analizador de espectro.

CAPÍTULO 6

PRUEBAS Y RESULTADOS

En este capítulo se describen las pruebas que se hicieron para comprobar el buen funcionamiento de la tarjeta PCI. Entre estas pruebas se encuentran aquellas que se refieren al diseño de la tarjeta y los resultados de las mediciones.

6.1 Pruebas de la tarjeta para PC

El diseño de la tarjeta PCI que funciona como analizador de espectro comenzó con el desarrollo de una pequeña tarjeta PCI que servía como interfaz entre la computadora y el bus Add-On del controlador PCI. Del lado del bus Add-On se consideró un conjunto de LED's para observar el envío de comandos de la computadora a la tarjeta. Por otro lado, se simulaba la entrada de datos en la región Pass-Thru por medio de jumpers. En conjunto, la tarjeta está integrada por el controlador PCI, un conjunto de LED's que sirven para visualizar los comandos enviados a la tarjeta por medio del buzón y jumpers para simular la entrada de datos por medio de una región Pass-Thru. Esta tarjeta está configurada para que trabaje con un bus de datos de 8 bits.

Con esta tarjeta también se hicieron las pruebas con la biblioteca de funciones de WinDriver, para probar el soporte que daba al chip S5920. La biblioteca de funciones de funciones de WinDriver está diseñada para trabajar con el chip S5933 de AMCC, que es un chip controlador que puede funcionar como maestro del bus, en cambio el S5920 sólo puede ser usado como dispositivo objetivo (target en inglés).

Después del ensamblado de la tarjeta se comenzaron las pruebas con la comunicación entre la computadora y la tarjeta. Y de este conjunto de pruebas se obtuvo la comunicación exitosa entre la computadora y el bus Add-On de la tarjeta. Las funciones que proporciona WinDriver funcionaron de manera correcta al utilizar el S5920, haciendo uso de aquellas que realmente el dispositivo esté capacitado para realizar, como la transferencia de datos por la región Pass-Thru y por los buzones.

El siguiente paso fue el diseño de la tarjeta que ya integrara el controlador PCI, el DSP y el convertidor.

La tarjeta para el analizador de espectro se probó en varias etapas. La primera de ellas fue la configuración de la memoria serial que es leída por el controlador PCI, y que contiene la información de los registros PCI que definen el funcionamiento del controlador. Uno de los registros define el tamaño de la región Pass-Thru y en qué espacio de memoria la va a situar el sistema. Por el tamaño de la región Pass-Thru y por la compatibilidad con sistemas basados en MS-DOS, se definió que la región estuviera definida dentro del espacio de Entrada/Salida. Para una PC el tamaño máximo que se puede definir en este espacio es de 256 bytes, de este modo la trasferencia de datos se realiza en una secuencia sucesiva de paquetes de 256 bytes.

Después se comenzó a transferir programas a la memoria de la tarjeta para que el DSP los pudiera leer y ejecutar. De esta manera, se obtenía el código por medio del compilador para el ensamblador del C31 y se añadían las palabras que requiere el DSP para leer un programa, como son: el ancho del bus, la dirección de inicio donde se coloca el programa en la memoria interna del DSP, el tamaño del programa y los estados de espera para la lectura de cada palabra. Estos programas consistían, básicamente, en la escritura de datos en localidades de memoria que necesitaran utilizar la interfaz externa del DSP, para escribir en la memoria de la tarjeta. Estos datos podían ser leídos por la computadora y corroborar que el DSP estuviera funcionando adecuadamente de forma conjunta con la memoria de la tarjeta. Y no sólo eso, sino que también estuviera funcionando adecuadamente el controlador PCI, ya que se usa tanto para enviar el programa como para leer el resultado de los programas.

El siguiente paso fue verificar que el convertidor estuviera funcionando bien. La prueba que se hizo fue guardar un conjunto de muestras en la memoria de la tarjeta y después leerla por medio del controlador PCI, de tal manera que un programa dibujara las muestras en la pantalla, para que se pudiera comparar la señal de entrada con la señal vista en la computadora.

En este punto se tuvo que modificar una parte del circuito. El convertidor necesita un circuito que convierte una señal referenciada a tierra por una señal diferencial, por esta razón se estaba utilizando un par de amplificadores operacionales que realizaran este cambio, pero la señal que proporcionaban estos amplificadores no daba el voltaje deseado en una de las terminales diferenciales. Esta es la razón por la que se hizo un cambio en el diseño para corregir esta situación. Se decidió utilizar un amplificador diferencial que trabajara con el mismo voltaje de alimentación, de 0 a 5 volts. Ahora sí, el voltaje diferencial es el deseado y se pudo observar la señal de entrada en la pantalla de la computadora.

Posteriormente, se desarrollo un programa en la computadora que integrara todas las partes, es decir, que obtuviera el espectro de la señal. Primero se obtenían las muestras, después se transfería el programa al DSP, luego se leía el resultado del procesamiento y finalmente se mostraba el espectro en la pantalla de la computadora. Pero aún faltaba algo, verificar el funcionamiento de la máquina de estados.

Sin la máquina de estados, es muy difícil saber en qué momento ha terminado una etapa del procesamiento y es tiempo de comenzar otra. La máquina nos da la información sobre el inicio y fin de algunos pasos del procesamiento, por ello es muy importante. Entonces, el programa que se ejecuta en la computadora, se modifica nuevamente para recibir los comandos provenientes de la máquina de estados; no comienza una nueva etapa del procesamiento, si no ha finalizado una

anterior y la máquina de estados lo ha indicado. Así, el usuario sólo tiene que presionar un botón para que la computadora muestre el espectro de una señal.

6.2 Resultados de mediciones

Se realizaron varias pruebas de medición, las primeras fueron sobre la señal de entrada, para observar en la computadora la señal en el dominio del tiempo. De esta manera se muestrearon señales senoidales, triangulares y cuadradas. En la pantalla se observó que tanto la amplitud, como el voltaje de referencia y la forma de la señal eran correctas. La figura 6.1 muestra la tarjeta PCI al momento de estar funcionando en la PC, junto con un generador de funciones y un osciloscopio.

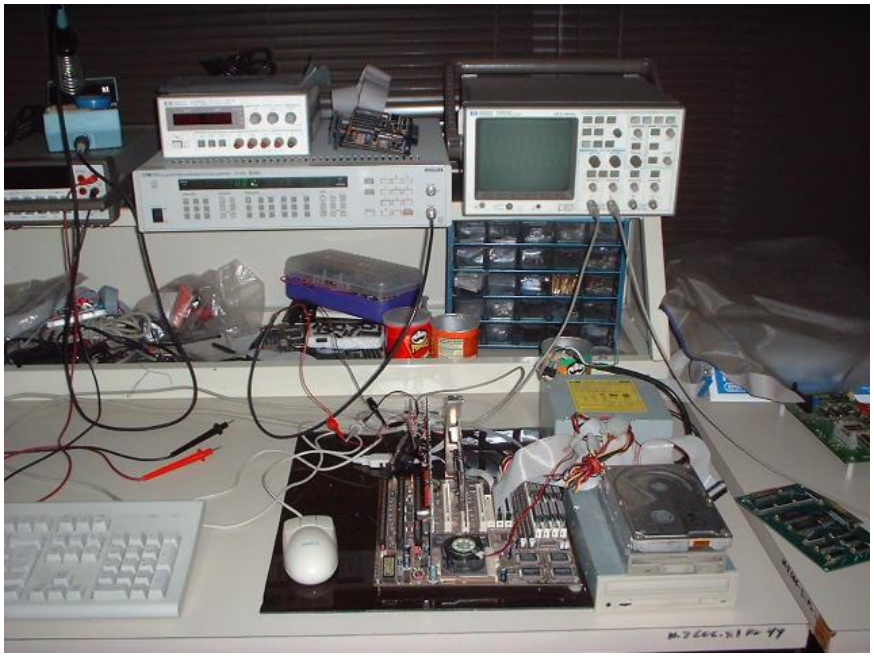

Figura 6.1. Equipo de pruebas.

A continuación se muestra el ejemplo de una señal en el dominio del tiempo (osciloscopio) y en el dominio de la frecuencia (analizador de espectro). La figura 6.2 muestra la salida del programa del analizador de espectro, donde se puede observar una señal de 50KHz y su espectro en frecuencia, que es un pulso que se observa en la parte baja de la pantalla, las frecuencias aumentan de izquierda a derecha. La figura 6.3 muestra la misma señal pero en el osciloscopio.

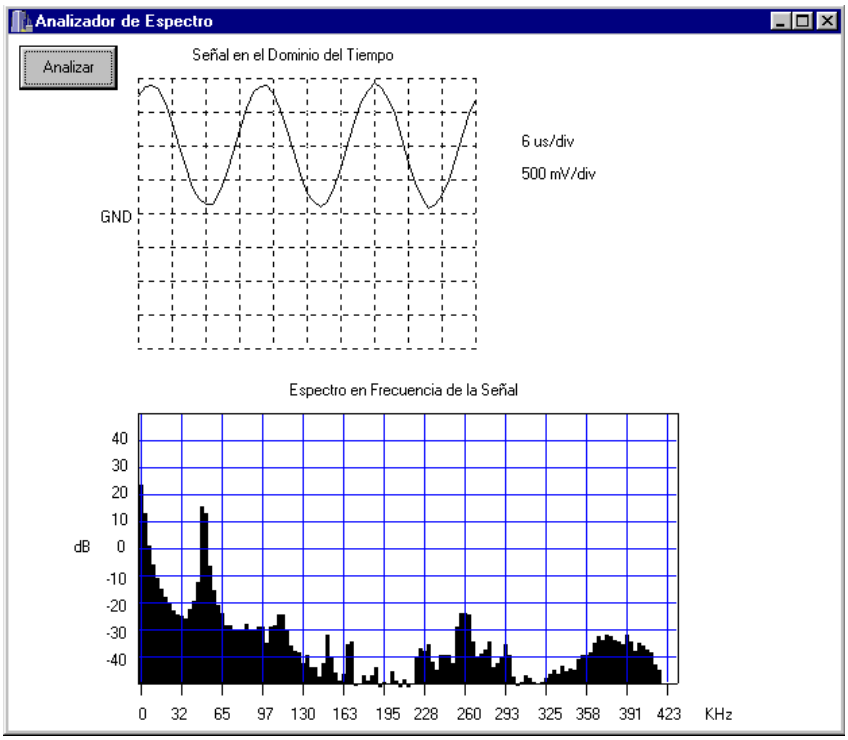

Figura 6.2. Pantalla que muestra una señal de 50KHz en tiempo y frecuencia.

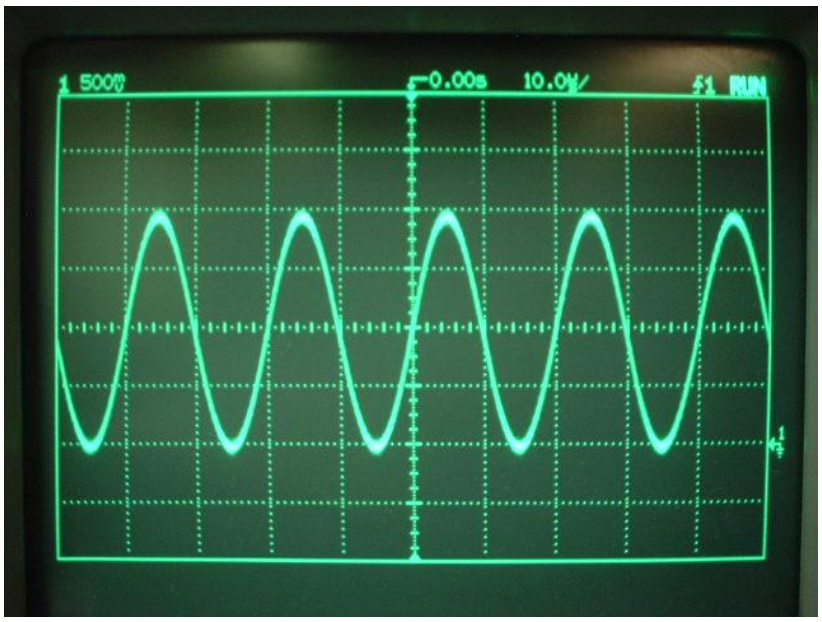

Figura 6.3. Señal de 50KHz vista en el osciloscopio.

Las siguientes figuras muestran una señal senoidal de 100KHz en la pantalla de la computadora y en el osciloscopio. La figura 6.4 muestra la pantalla de la computadora donde se observa la señal tanto en el dominio del tiempo como de la frecuencia. La figura 6.5 muestra la señal en el osciloscopio.

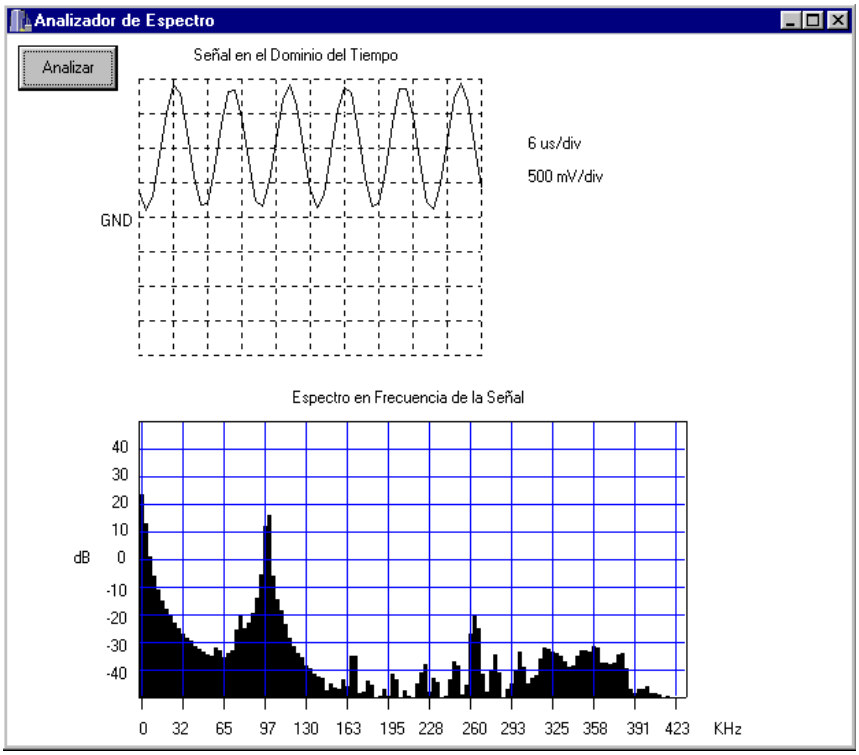

Figura 6.4. Pantalla que muestra una señal de 100KHz en tiempo y frecuencia.

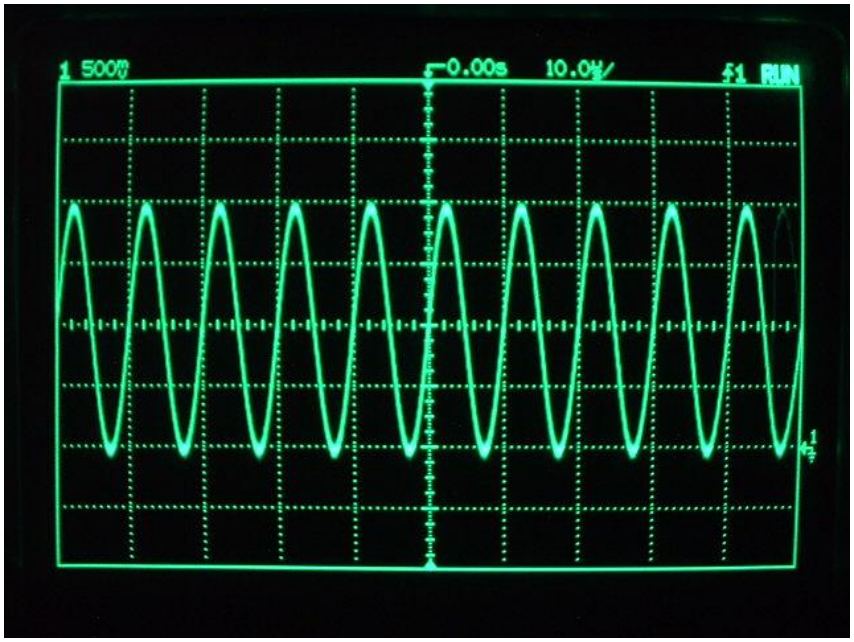

Figura 6.5. Señal de 100KHz vista en el osciloscopio.

Como se puede apreciar en la salida que despliega el programa del analizador de espectro, cuando se incrementa la frecuencia de la señal, el pulso que indica la frecuencia en el análisis espectral se desplaza hacia la derecha, lo cual quiere decir que se ha aumentado la frecuencia.

Otras mediciones que se hicieron fueron con señales de voz. La figura 6.6 muestra una señal de voz correspondiente a la vocal A de una persona. La figura 6.7 muestra una señal de voz correspondiente a la vocal E.

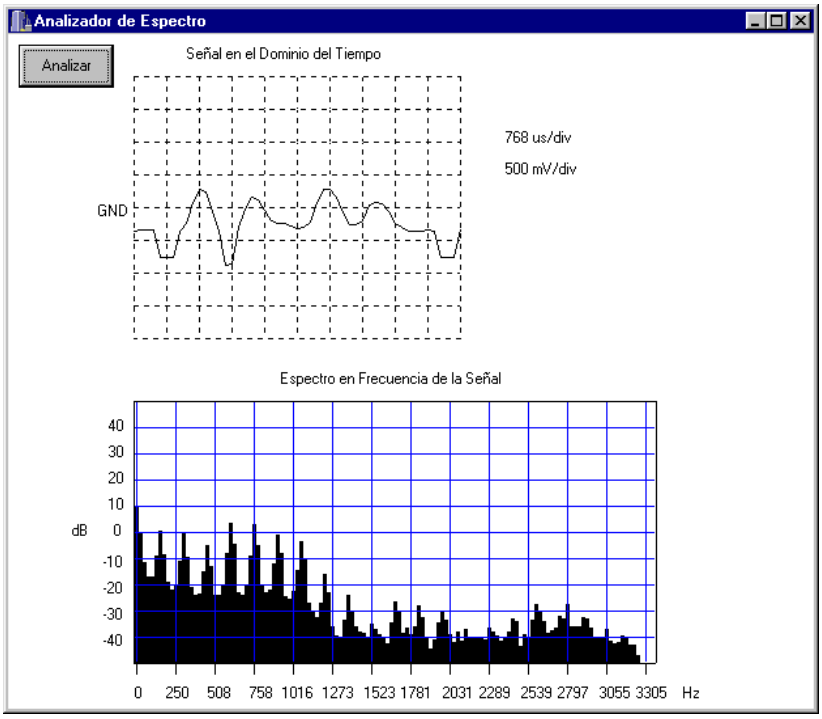

Figura 6.6. Señal de voz, vocal A.

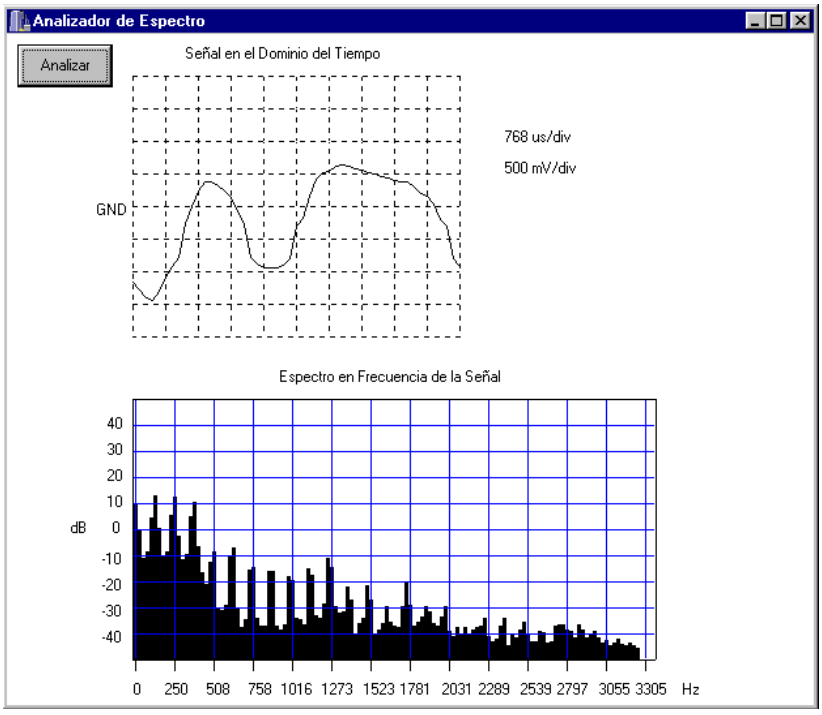

Figura 6.7. Señal de voz, vocal E.

6.3 Resumen

En este capítulo se mencionaron las pruebas de funcionamiento que se hicieron a la tarjeta, desde el diseño y prueba de los circuitos de la tarjeta, hasta pruebas de medición del instrumento. Entre las mediciones que se hicieron se encuentran señales determinísticas como la función seno y señales no determinísticas como la señal de voz. En el siguiente capítulo se mencionan las conclusiones.

CAPÍTULO 7

CONCLUSIONES

En el presente capítulo se hacen algunos comentarios sobre las metas que se cumplieron en el desarrollo del analizador de espectro, en cuanto a diseño y funcionalidad. Además se proponen trabajos a futuro para incrementar las capacidades de la tarjeta PCI y del analizador.

7.1 Logros alcanzados

Una de las metas que se deseaba alcanzar en el desarrollo de la tarjeta PCI era la implementación de instrumentos virtuales para el área de medición. En esta época en la que la computadora se ha vuelto una herramienta imprescindible en los estudios de ingeniería, es muy conveniente utilizar la infraestructura con que se cuenta en las universidades o en el hogar para ofrecer nuevas capacidades a los equipos, en este caso, la posibilidad de tener un analizador de espectro en una computadora. La tabla 7.1 muestra las especificaciones del analizador de espectro. El costo de desarrollo de este equipo de medición es mucho menor al costo de un equipo de medición que funciona sin la computadora y es también menor al de los instrumentos virtuales que se encuentran en el mercado. Esto cumple con la meta más importante que se debe tener: proporcionar una herramienta que utilice la infraestructura ya existente y que tenga un costo menor a las que se encuentran en el mercado. En la tabla 7.2 se puede ver el costo de los componentes y de la mano de obra, no se toman en cuenta costos indirectos como uso de las instalaciones o tiempo de asesoría del Director de Tesis. Este costo queda abajo del precio de las tarjetas NI4551 y NI4552 de 5500 dólares.

La tarjeta PCI no sirve exclusivamente para obtener el espectro de una señal, también se puede utilizar como un osciloscopio virtual, ya que nos permite observar las señales en el dominio del tiempo. El programa que muestra el espectro de la señal en la pantalla de la computadora, también despliega la señal en tiempo. Esto agrega más funcionalidad a una tarjeta que puede ser programada para que realice más de una tarea.

El diseño de la tarjeta permite que pueda ser utilizada para programar diversas herramientas de procesamiento digital, ya que los programas son cargados en la memoria de la tarjeta y el DSP se encarga de ejecutarlos por medio de los comandos que envía la computadora. De esta manera, la tarjeta puede funcionar como herramienta de desarrollo de otras aplicaciones. En algunas ocasiones, la memoria con que cuenta el DSP es muy poca para algunos propósitos, pero con la memoria de 256KB de la tarjeta, el rango de aplicaciones se eleva considerablemente.

Tabla 7.1 Especificaciones del analizador de espectro

Características

Número de canales	1
Resolución del convertidor	16 bits
Linealidad	±0.75 LSB
Tasas de muestreo	2.5MHZ, 1.25MHz, 312.5Khz, 78KHz, 19.5KHz, 4.8KHz, 1.2KHz, 610Hz
Voltaje de entrada	4Vp-p
Filtro anti-alias	0.45 Tasa de muestreo
Número de muestras (máx.)	32K muestras
Tamaño FFT (máx.)	4K muestras

Tabla 7.2 Costo de la tarjeta

Concepto	**Costo (dólares)**
Controlador PCI	25
DSP	30
ADC	45
CPLD	8
Memoria	25
Otros componentes	20
Tarjeta	250
Mano de obra (beca)	3000
Total	3403

Uno de los objetivos en el diseño de la tarjeta era contribuir en el desarrollo de aplicaciones para el bus PCI. Aunque el bus PCI tiene varios años de haber entrado al mercado de las computadoras, realmente hay pocas aplicaciones hechas por los estudiantes del CIC. Los diseños, casi, estaban orientados exclusivamente al bus ISA, que en la actualidad está quedando al margen en las computadoras. Por esta razón existía la inquietud de llevar a cabo un proyecto que se beneficiara de las características del bus PCI, estándar para tarjetas de expansión, compatibilidad con muchos sistemas y configuración automática (plug and play).

7.2 Trabajos a futuro

Como se mencionó en la sección anterior, la tarjeta puede utilizarse para realizar más de una tarea. Puede funcionar como analizador de espectro, como osciloscopio, como herramienta de desarrollo, etc.

El software que muestra el espectro de la señal puede extenderse para que el usuario pueda configurar el analizador, como las escalas, la frecuencia de muestreo, etc.

Yendo más allá de un analizador de espectro, el instrumento puede utilizarse para una aplicación que requiera el análisis espectral de un conjunto de muestras, como puede ser una aplicación en el área médica, de reconocimiento de voz, de análisis de vibración, etc.

También, puede extenderse el programa de aplicación de análisis de espectro para que obtenga todo un conjunto de herramientas de medición de señal dinámica, como los ofrecidos por algunos fabricantes de instrumentos virtuales.

Como herramienta de desarrollo puede extenderse para ofrecer un software que permita cargar diversos programas desde una interfaz gráfica y que pueda seguirse en la pantalla la secuencia de instrucciones que se ejecutan en cada programa como lo hacen los depuradores comerciales.

En cuanto al diseño de la tarjeta, puede incrementarse el número de canales de entrada de señal. Para muestrear más señales al mismo tiempo, puede añadirse otro convertidor o dividir la tasa de muestreo del convertidor entre más señales, utilizando una conmutación en la entrada. También se pueden agregar etapas de ganancia y atenuación de señal para que el usuario pueda utilizar un rango más amplio de voltajes de entrada.

7.3 Resumen

En este capítulo se mencionaron los logros alcanzados en el desarrollo del analizador de espectro como instrumento virtual y como base para otras aplicaciones. Además, se propusieron algunas alternativas para incrementar las capacidades de la tarjeta PCI y del programa de aplicación.

BIBLIOGRAFÍA

[AM: 1998] AMCC, Libro de datos de los productos PCI, EUA, Applied Micro Circuits Corporation, 1998.

[BG: 1998] Berlin, H. M. y Getz, F. C., Principles of electronic instrumentation and measurement, EUA, Macmillan Publishing Company, 1988.

[HP: 1997] Hewlett Packard, Analizador de señal dinámica HP 35670A, Manual de usuario, EUA, Hewlett Packard, 1997.

[KR: 2000] KRF Tech, WinDriver Developer's Guide, EUA, KRF Tech, 2000.

[NI: 1998] National Instruments, Analizador de señal dinámica NI 4551/4552, Manual del Usuario, EUA, National Instruments, 1998.

[SA: 1995] Shanley, T. y Anderson, D., PCI system architecture, EUA, MindShare, Inc., 1995.

[ST: 1997] Stoica, Petre y Moses, R. L., Introduction to Spectral Analysis, EUA, Prentice Hall, 1997.

[SW: 1998] Solari, E. y Willse, G., PCI hardware and software architecture and design, EUA, Annabooks, 1998.

[TI: 1997] Texas Instruments, Guía del usuario del TMS320C31, EUA, Texas Instruments, 1997.

[TI: 1999] Texas Instruments, Hoja de especificación del TMS320C31, EUA, Texas Instruments, 1999.

[WE: 1967] Welch, Peter D., The use of Fast Fourier Transform for the Estimation of Power Spectra, EUA, IEEE Transactions on Audio and Electroacoustics, Junio 1967.

ANEXOS

A Diagrama eléctrico de la tarjeta PCI
B Operación del bus PCI
C Registros de configuración PCI
D Programa de procesamiento para el DSP
E Programa que presenta el espectro en la computadora

ANEXO A

DIAGRAMA ELÉCTRICO DE LA TARJETA PCI

A continuación se muestra el diagrama eléctrico de la tarjeta PCI.

ANEXO B

OPERACIÓN DEL BUS PCI

La especificación del bus PCI define la interacción entre dos recursos PCI: el maestro y el objetivo del bus PCI. El maestro puede ser un procesador o un puente que representa a un procesador en otro bus. El objetivo puede ser un recurso de memoria o de entrada/salida junto con un puente que proporciona la interacción con recursos en otros buses. También se define un protocolo para que el maestro lea y escriba datos desde y hacia un objetivo, además de un espacio de direcciones para memoria y entrada/salida.

La especificación también define ciclos de acceso sencillos y de ráfaga. Un ciclo de acceso sencillo permite una lectura o escritura en cada ciclo, mientras que un ciclo de acceso de ráfaga permite múltiples lecturas o escrituras en un solo ciclo de acceso. A continuación se describen los ciclos de acceso sencillo y de ráfaga.

- Ciclo de acceso sencillo. Los espacios de direcciones de memoria, de entrada/salida y de configuración son seleccionados por el ciclo de acceso (figura B.1). Un ciclo de acceso comienza con la fase de dirección, es decir, cuando el maestro lanza la señal FRAME# (ciclo de marco) para indicar que existen una dirección y un comando válidos en las señales AD (direcciones y datos) y C/BE# (comandos y habilitación de bytes), y maneja la paridad válida para estas señales en la señal PAR un ciclo de reloj después. El tipo de comando indica el espacio de direcciones y si el ciclo de acceso es de lectura o de escritura. El ciclo de acceso continúa con la fase de datos. Todos los objetivos PCI decodifican la dirección y el indicado pide el ciclo de acceso lanzando la señal DEVSEL# (selección de dispositivo). Para un ciclo de acceso de lectura, el maestro maneja los datos que se van a escribir, la información de bytes válidos y la paridad de estas señales en las líneas AD, C/BE# y PAR, respectivamente. Para un ciclo de acceso de lectura, el maestro maneja la información de bytes válidos en las líneas C/BE#. El objetivo maneja los datos que se van a leer en las líneas AD y la paridad de las señales AD y C/BE# en la línea PAR. Debido a la múltiplexión de la dirección y los datos, el maestro pone las líneas AD en tercer estado durante un ciclo de reloj, antes de que el objetivo envíe los datos por estas líneas. El acceso a los datos ocurre en el flanco de subida de la señal CLK (reloj) cuando las señales IRDY# (iniciador listo) y TRDY# (objetivo listo) están activadas. La fase ociosa indica que no se están ejecutando ciclos de bus durante los respectivos periodos de la señal de reloj CLK. La fase ociosa ocurre cuando no están activadas las señales FRAME# e IRDY#.

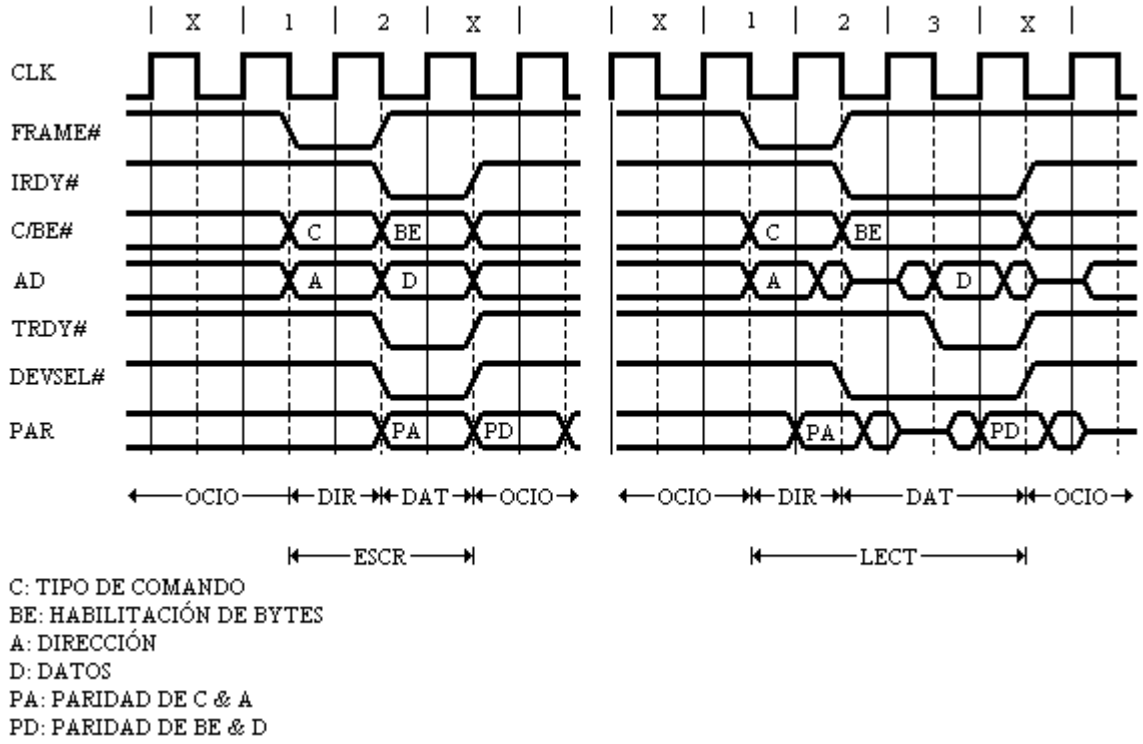

Figura B.1. Ciclos de acceso sencillos de escritura y de lectura del bus PCI.

- Ciclo de acceso de ráfaga. La activación y la no activación de las señales FRAME# e IRDY# transforman un ciclo de acceso sencillo en un ciclo de acceso de ráfaga (figuras B.2 y B.3). Durante la fase de dirección, el protocolo de las señales es el mismo para ciclos de acceso sencillos y de ráfaga. Durante la fase de datos, la señal IRDY# es activada cuando el maestro está listo para acceder a los datos. Si la señal FRAME# es desactivada simultáneamente con la activación de la señal IRDY# para el primer acceso, el acceso es completado como sencillo (ver figura B.2). Si la señal FRAME# se mantiene activa después de la activación de la señal IRDY# para el primer acceso, el ciclo continúa como una ráfaga. El ciclo de acceso de ráfaga (ver figura B.3) continúa hasta que la señal FRAME# es desactivada cuando la señal IRDY# es activada. La fase de datos de una ráfaga consiste de una serie de micro-accesos. Cada micro-acceso consiste de acceso a datos cuando las señales TRDY# e IRDY# están activas. El primer micro-acceso está definido como micro-acceso inicial (INIT) y los micro-accesos posteriores están definidos como micro-accesos subsecuentes (SUB). La dirección base del micro-acceso inicial es establecida durante la fase de dirección. Las direcciones de los micro-accesos subsecuentes son relativas al incremento de la dirección base.

Existen otros ciclos de acceso en la especificación PCI. Una descripción más detallada de cada uno de ellos se puede encontrar en [SW: 1998] y en [SA: 1995]. En este anexo sólo se describen los ciclos que muestran el funcionamiento básico de la transferencia de datos en el bus PCI.

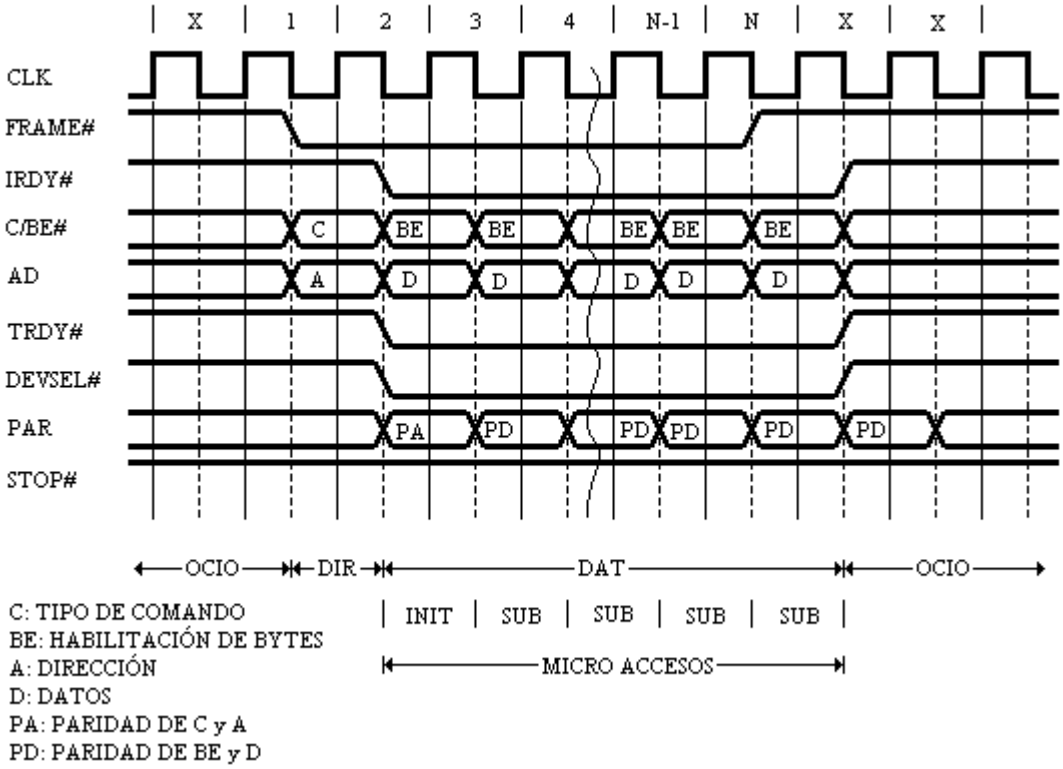

Figura B.2. Ciclo de acceso de ráfaga de escritura del bus PCI.

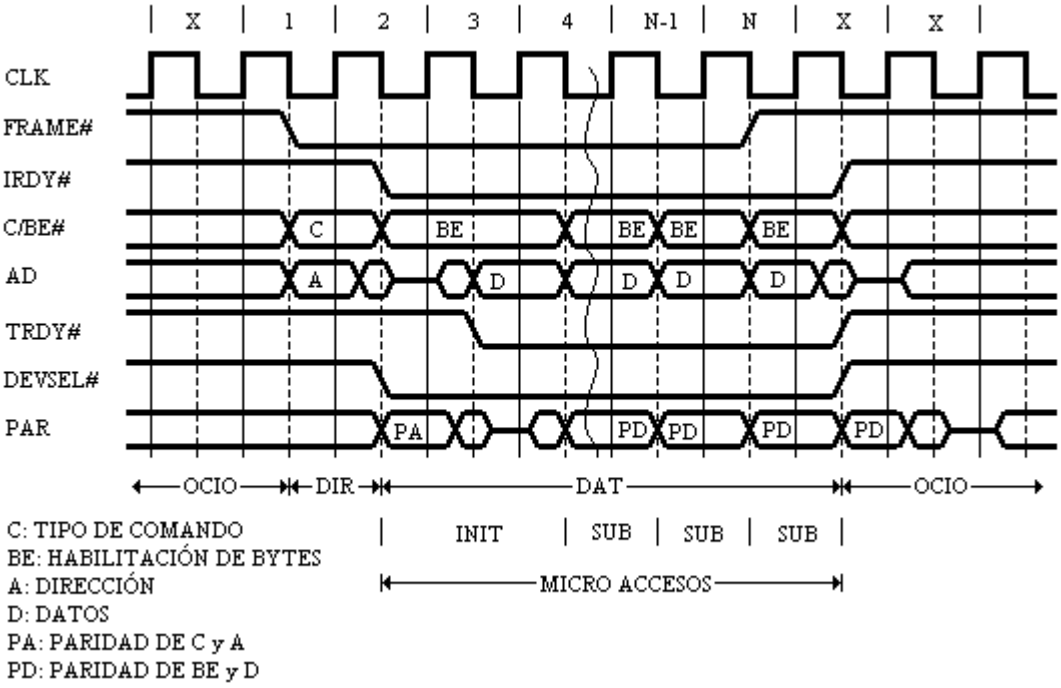

Figura B.3. Ciclo de acceso de ráfaga de lectura del bus PCI.

ANEXO C

REGISTROS DE CONFIGURACIÓN PCI

A continuación se muestran los valores de los registros de configuración PCI que se guardan en la memoria serial de la tarjeta.

Tabla C.1. Registros de configuración PCI.

Registro	Valor
VID	10E8h
DID	5920h
PCICMD	0000h
PCISTS	0000h
RID	00h
CLCD	FF0000h
CALN	00h
LAT	00h
HDR	00h
BIST	00h
BADR0	10E8FF81h
BADR1	FFFFFF01h
SVID	0000h
SID	0000h
XROM	00000000h
INTLN	FFh
INTPIN	01h
MINGNT	00h
MAXLAT	00h

ANEXO D

PROGRAMA DE PROCESAMIENTO PARA EL DSP

A continuación se muestra el listado del programa que se ejecuta en el DSP, para obtener el espectro en frecuencia de una señal eléctrica.

```
                .start    "CODE",0x809802   ;DIRECCIÓN DE INICIO
                .start  "DATOS",0x809A00    ;DIRECCIÓN DE DATOS
                .sect     "CODE"            ;SEGMENTO DE CÓDIGO
COEF            .set      809Ah
VEN             .set      809Bh
ENT             .set      809Ch
REAL            .set      809Dh
IMAG            .set      809Eh
PRUEBAD         .set      4001h
RESUD           .set      4008h
RES             .set      4000h
CAMBIO          .set      4013h
COND2           .set      5000h
COND3           .set      6000h
                .entry  PRINC
PRINC           ldp     @STACK        ;DEFINE PILA
                ldi     @STACK,SP
                ldi       COND2,AR0
                ash       8h,AR0
                sti       R0,*AR0
                ldi       0,IOF
PEQU            ldi       IOF,R0
                tstb      8,R0
                bz        PEQU
                ldi       CAMBIO,AR0
                ash       8h,AR0
                ldi       PRUEBAD,AR1
                ash       8h,AR1
                ldi       480h,R7
CFLOT           ldi       *AR0++,R0
                lsh       16,R0
                ash       -16,R0
                float     R0,R0
                mpyf      @DIVI,R0
                stf       R0,*AR1++
                subi      1,R7
                bnz       CFLOT
```

```
        ldi     0,R0        ;primer segmento
        sti     R0,@SEGM
        ldi     RESUD,AR1
        ash     8h,AR1
        ldi     AR1,R1
        sti     R1,@RESU
        ldi     PRUEBAD,AR0
        ash     8h,AR0
        ldi     AR0,R0
        sti     R0,@PRUEBA
OTROC           ldi     @PRUEBA,AR0
        ldi     100h,R7
        ldi     ENT,AR1
        ash     8h,AR1
DFG     ldf     *AR0++,R0
        stf     R0,*AR1++
        subi    1,R7
        bnz     DFG
        ldi     @PRUEBA,R0
        addi    80h,R0
        sti     R0,@PRUEBA
        ldi     100h,R7     ;256 CICLOS
        ldi     ENT,AR0     ;DEFINE ENTRADA DE DATOS
        ash     8h,AR0
        ldi     VEN,AR1     ;DIRECCION DE VENTANA
        ash     8h,AR1
VENT    ldf     *AR0,R0     ; DATO ENTRADA
        ldf     *AR1++,R1   ;VALOR VENTANA
        mpyf3   R0,R1,R2    ;VENTANEO
        stf     R2,*AR0++   ;GUARDA DATO
        subi    1,R7
        bnz     VENT        ; MAS DATOS?
        ldi     IMAG,AR0    ;RESULTADO REAL
        ash     8h,AR0
        ldf     0.0,R0      ; LIMPIA ARREGLO
        rpts    255
        stf     R0,*AR0++
        ldi     REAL,AR0    ;RESULTADO REAL
        ash     8h,AR0
        ldi     ENT,AR1     ;DATOS ENTRADA
        ash     8h,AR1
        ldi     128,IR0
        ldi     100h,R7
BITR    ldf     *AR1++(IR0)B,R0 ; carga arreglo
        stf     R0,*AR0++       ; bit reversi
        subi    1,R7
        bnz     BITR        ;MAS DATOS?
        ldi     0,AR0       ; iteracion
INIL5   ldi     1,AR1       ; mari
        ldi     1,AR2       ; grupo
        ldi     7,AR3
        subi    AR0,AR3
        lsh     AR0,AR1     ; mari
        lsh     AR3,AR2     ; grupo
        ldi     AR0,IR0
```

```
        ldi     0,AR3       ; cada grupo
INIL4   ldi     0,AR4         ; cada mariposa
INIL3   ldi     2,AR5         ; primera mariposa
        mpyi    AR1,AR5
        mpyi    AR3,AR5
        addi    AR4,AR5
        ldi     AR5,AR6     ; segunda mariposa
        addi    AR1,AR6
        ldi     2,AR0       ; separacion
        mpyi    AR2,AR0
        mpyi    AR4,AR0
        ldi     REAL,AR7    ;RESULTADO REAL
        ash     8h,AR7
        addi    AR5,AR7
        ldf     *AR7,R0             ; primer sumando
        ldi     REAL,AR7    ;RESULTADO REAL
        ash     8h,AR7
        addi    AR6,AR7
        ldf     *AR7,R1             ; primer operando
        ldi     COEF,AR7     ;COEFICIENTES
        ash     8h,AR7
        addi    AR0,AR7
        ldf     *AR7,R2             ; segundo operando
        mpyf    R2,R1       ; segundo sumando
        ldi     IMAG,AR7    ;RESULTADO IMAG
        ash     8h,AR7
        addi    AR6,AR7
        ldf     *AR7,R2             ; primer operando
        ldi     COEF,AR7     ;COEFICIENTES
        ash     8h,AR7
        addi    AR0,AR7
        addi    1,AR7
        ldf     *AR7,R3             ; segundo operando
        mpyf    R3,R2       ; tercer sumando
        addf    R1,R0
        subf    R2,R0
        ldf     R0,R4       ; primer valor
        ldi     IMAG,AR7    ;RESULTADO IMAG
        ash     8h,AR7
        addi    AR5,AR7
        ldf     *AR7,R0             ; primer sumando
        ldi     IMAG,AR7    ;RESULTADO IMAG
        ash     8h,AR7
        addi    AR6,AR7
        ldf     *AR7,R1             ; primer operando
        ldi     COEF,AR7     ;COEFICIENTES
        ash     8h,AR7
        addi    AR0,AR7
        ldf     *AR7,R2             ; segundo operando
        mpyf    R2,R1       ; segundo sumando
        ldi     REAL,AR7    ;RESULTADO REAL
        ash     8h,AR7
        addi    AR6,AR7
        ldf     *AR7,R2             ; primer operando
        ldi     COEF,AR7     ;COEFICIENTES
        ash     8h,AR7
```

```
addi    AR0,AR7
addi    1,AR7
ldf     *AR7,R3                 ; segundo operando
mpyf    R3,R2           ; tercer sumando
addf    R1,R0
addf    R2,R0
ldf     R0,R5           ; segundo valor
ldi     REAL,AR7        ;RESULTADO REAL
ash     8h,AR7
addi    AR5,AR7
ldf     *AR7,R0                 ; primer sumando
ldi     REAL,AR7        ;RESULTADO REAL
ash     8h,AR7
addi    AR6,AR7
ldf     *AR7,R1                 ; primer operando
ldi     COEF,AR7        ;COEFICIENTES
ash     8h,AR7
addi    AR0,AR7
ldf     *AR7,R2                 ; segundo operando
mpyf    R2,R1           ; segundo sumando
ldi     IMAG,AR7        ;RESULTADO IMAG
ash     8h,AR7
addi    AR6,AR7
ldf     *AR7,R2                 ; primer operando
ldi     COEF,AR7        ;COEFICIENTES
ash     8h,AR7
addi    AR0,AR7
addi    1,AR7
ldf     *AR7,R3                 ; segundo operando
mpyf    R3,R2           ; tercer sumando
subf    R1,R0
addf    R2,R0
ldf     R0,R6           ; tercer valor
ldi     IMAG,AR7        ;RESULTADO IMAG
ash     8h,AR7
addi    AR5,AR7
ldf     *AR7,R0                 ; primer sumando
ldi     IMAG,AR7        ;RESULTADO IMAG
ash     8h,AR7
addi    AR6,AR7
ldf     *AR7,R1                 ; primer operando
ldi     COEF,AR7        ;COEFICIENTES
ash     8h,AR7
addi    AR0,AR7
ldf     *AR7,R2                 ; segundo operando
mpyf    R2,R1           ; segundo sumando
ldi     REAL,AR7        ;RESULTADO REAL
ash     8h,AR7
addi    AR6,AR7
ldf     *AR7,R2                 ; primer operando
ldi     COEF,AR7        ;COEFICIENTES
ash     8h,AR7
addi    AR0,AR7
addi    1,AR7
ldf     *AR7,R3                 ; segundo operando
mpyf    R3,R2           ; tercer sumando
```

```
        subf    R1,R0
        subf    R2,R0
        ldf     R0,R7          ; cuarto valor
        ldi     REAL,AR7       ;RESULTADO REAL
        ash     8h,AR7
        addi    AR5,AR7
        stf     R4,*AR7
        ldi     IMAG,AR7       ;RESULTADO IMAG
        ash     8h,AR7
        addi    AR5,AR7
        stf     R5,*AR7
        ldi     REAL,AR7       ;RESULTADO REAL
        ash     8h,AR7
        addi    AR6,AR7
        stf     R6,*AR7
        ldi     IMAG,AR7       ;RESULTADO IMAG
        ash     8h,AR7
        addi    AR6,AR7
        stf     R7,*AR7
        addi    1,AR4
        cmpi    AR4,AR1
        bnz     INIL3
        addi    1,AR3
        cmpi    AR3,AR2
        bnz     INIL4
        ldi     IR0,AR0
        addi    1,AR0
        ldi     8,AR5
        cmpi    AR0,AR5
        bnz     INIL5
        ldi     @RESU,AR0
        ldi     100h,R7
        ldi     REAL,AR1
        ash     8h,AR1
        ldi     IMAG,AR2
        ash     8h,AR2
FGD     ldf     *AR1++,R0
        mpyf    R0,R0
        ldf     *AR2++,R1
        mpyf    R1,R1
        addf    R1,R0
        mpyf    @NXQ,R0
        stf     R0,*AR0++
        subi    1,R7
        bnz     FGD
        ldi     @RESU,R0
        addi    100h,R0
        sti     R0,@RESU
        ldi     @SEGM,R0
        addi    1,R0
        sti     R0,@SEGM
        cmpi    8,R0
        bnz     OTROC
        ldi     7,R6
        ldi     RESUD,AR1
        ash     8h,AR1
```

```
        ldi     RES,AR0
        ash     8h,AR0
        ldi     100h,R7
UNCH    ldf     *AR1++,R0
        stf     R0,*AR0++
        subi    1,R7
        bnz     UNCH
UCH     ldi     RES,AR0
        ash     8h,AR0
        ldi     100h,R7
OCH     ldf     *AR1++,R0
        ldf     *AR0,R1
        addf    R0,R1
        stf     R1,*AR0++
        subi    1,R7
        bnz     OCH
        subi    1,R6
        bnz     UCH
        ldi     RES,AR1
        ash     8h,AR1
        ldi     100h,R7
YNM     ldf     *AR1,R0
        mpyf    @XSEG,R0
        stf     R0,*AR1++
        subi    1,R7
        bnz     YNM
        ldi     @TABLA,AR1      ;DIRECCION DE VENTANA
        ldi     RES,AR3
        ash     8h,AR3
        ldi     580h,R7
CHAN    ldf     *AR3,R0
        LDF     R0,R0 ; Determine the sign of the number
        LDFZ    *+AR1(4),R0 ; If 0, load appropriate number
        BND     NEG ; Branch to NEG if negative (delayed)
        ABSF    R0,R0 ; Take the absolute value of the number
        LSH     1,R0 ; Eliminate the sign bit in R0
        PUSHF   R0
        POP     R0 ; Place number in lower 32 bits of R0
        ADDI    *+AR1(2),R0 ; Add exponent bias (127)
        LSH     -1,R0 ; Add the positive sign
CONT    TSTB    *+AR1(5),R0
        bNZ     XIMIL           ; If e > 0, return
        TSTB    *+AR1(7),R0
        BZ      XIMIL   ; If e = 0 & f = 0, return
        PUSH    R0
        POPF    R0
        LSH     -1,R0 ; Shift f right by one bit
        PUSHF   R0
        POP     R0
        ADDI    *+AR1(6),R0 ; Add 1 to the MSB of f
        b       XIMIL
NEG     POP     R0 ; Place number in lower 32 bits of R0
        BRD     CONT
        ADDI    *+AR1(2),R0 ; Add exponent bias (127)
        LSH     -1,R0 ; Make space for the sign
        ADDI    *+AR1(3),R0 ; Add the negative sign
```

```
XIMIL       sti     R0,*AR3++
            subi    1,R7
            bnz     CHAN
            ldi     COND3,AR0
            ash     8h,AR0
            sti     R0,*AR0
PEQUE       ldi     IOF,R0
            tstb    8,R0
            bnz     PEQUE
XIMIL2      b       XIMIL2
DIVI        .float  6.103515625E-5
NXQ         .float  0.007295719846
XSEG        .float  0.125
SEGM        .word   0
CICLO       .word   0
PRUEBA      .word   0
RESU        .word   0
TABLA       .word   $+1
            .word   0xFF800000
            .word   0xFF000000
            .word   0x7F000000
            .word   0x80000000
            .word   0x81000000
            .word   0x7F800000
            .word   0x00400000
            .word   0x007FFFFF
            .word   0x7F7FFFFF
STACK       .word   $+1
            .sect   "DATOS"
COEFI       .float  1.0,0.0
            .float  0.999698818696204,-0.0245412285229123
            .float  0.998795456205172,-0.049067674327418
            .float  0.99729045667869,-0.0735645635996674
            .float  0.995184726672197,-0.0980171403295606
            .float  0.99247953459871,-0.122410675199216
            .float  0.989176509964781,-0.146730474455362
            .float  0.985277642388941,-0.170961888760301
            .float  0.98078528040323,-0.195090322016128
            .float  0.975702130038529,-0.21910124015687
            .float  0.970031253194544,-0.242980179903264
            .float  0.96377606579544,-0.266712757474898
            .float  0.956940335732209,-0.290284677254462
            .float  0.949528180593037,-0.313681740398892
            .float  0.941544065183021,-0.33688985339222
            .float  0.932992798834739,-0.359895036534988
            .float  0.923879532511287,-0.38268343236509
            .float  0.914209755703531,-0.40524131400499
            .float  0.903989293123443,-0.427555093430282
            .float  0.893224301195515,-0.449611329654607
            .float  0.881921264348355,-0.471396736825998
            .float  0.870086991108711,-0.492898192229784
            .float  0.857728610000272,-0.514102744193222
            .float  0.844853565249707,-0.534997619887097
            .float  0.831469612302545,-0.555570233019602
            .float  0.817584813151584,-0.575808191417845
            .float  0.803207531480645,-0.595699304492433
```

```
.float  0.788346427626606,-0.615231590580627
.float  0.773010453362737,-0.634393284163645
.float  0.757208846506485,-0.653172842953777
.float  0.740951125354959,-0.671558954847018
.float  0.724247082951467,-0.689540544737067
.float  0.707106781186548,-0.707106781186547
.float  0.689540544737067,-0.724247082951467
.float  0.671558954847018,-0.740951125354959
.float  0.653172842953777,-0.757208846506484
.float  0.634393284163645,-0.773010453362737
.float  0.615231590580627,-0.788346427626606
.float  0.595699304492433,-0.803207531480645
.float  0.575808191417845,-0.817584813151584
.float  0.555570233019602,-0.831469612302545
.float  0.534997619887097,-0.844853565249707
.float  0.514102744193222,-0.857728610000272
.float  0.492898192229784,-0.870086991108711
.float  0.471396736825998,-0.881921264348355
.float  0.449611329654607,-0.893224301195515
.float  0.427555093430282,-0.903989293123443
.float  0.40524131400499,-0.914209755703531
.float  0.38268343236509,-0.923879532511287
.float  0.359895036534988,-0.932992798834739
.float  0.33688985339222,-0.941544065183021
.float  0.313681740398892,-0.949528180593037
.float  0.290284677254462,-0.956940335732209
.float  0.266712757474898,-0.96377606579544
.float  0.242980179903264,-0.970031253194544
.float  0.21910124015687,-0.975702130038529
.float  0.195090322016128,-0.98078528040323
.float  0.170961888760301,-0.985277642388941
.float  0.146730474455362,-0.989176509964781
.float  0.122410675199216,-0.99247953459871
.float  0.0980171403295608,-0.995184726672197
.float  0.0735645635996675,-0.99729045667869
.float  0.0490676743274181,-0.998795456205172
.float  0.0245412285229123,-0.999698818696204
.float  6.12303176911189e-17,-1.0
.float  -0.0245412285229121,-0.999698818696204
.float  -0.049067674327418,-0.998795456205172
.float  -0.0735645635996673,-0.99729045667869
.float  -0.0980171403295606,-0.995184726672197
.float  -0.122410675199216,-0.99247953459871
.float  -0.146730474455362,-0.989176509964781
.float  -0.170961888760301,-0.985277642388941
.float  -0.195090322016128,-0.98078528040323
.float  -0.21910124015687,-0.975702130038529
.float  -0.242980179903264,-0.970031253194544
.float  -0.266712757474898,-0.96377606579544
.float  -0.290284677254462,-0.956940335732209
.float  -0.313681740398891,-0.949528180593037
.float  -0.33688985339222,-0.941544065183021
.float  -0.359895036534988,-0.932992798834739
.float  -0.38268343236509,-0.923879532511287
.float  -0.40524131400499,-0.914209755703531
.float  -0.427555093430282,-0.903989293123443
```

```
        .float  -0.449611329654607,-0.893224301195515
        .float  -0.471396736825998,-0.881921264348355
        .float  -0.492898192229784,-0.870086991108711
        .float  -0.514102744193222,-0.857728610000272
        .float  -0.534997619887097,-0.844853565249707
        .float  -0.555570233019602,-0.831469612302545
        .float  -0.575808191417845,-0.817584813151584
        .float  -0.595699304492433,-0.803207531480645
        .float  -0.615231590580627,-0.788346427626606
        .float  -0.634393284163645,-0.773010453362737
        .float  -0.653172842953777,-0.757208846506485
        .float  -0.671558954847018,-0.740951125354959
        .float  -0.689540544737067,-0.724247082951467
        .float  -0.707106781186547,-0.707106781186548
        .float  -0.724247082951467,-0.689540544737067
        .float  -0.740951125354959,-0.671558954847019
        .float  -0.757208846506485,-0.653172842953777
        .float  -0.773010453362737,-0.634393284163645
        .float  -0.788346427626606,-0.615231590580627
        .float  -0.803207531480645,-0.595699304492433
        .float  -0.817584813151584,-0.575808191417845
        .float  -0.831469612302545,-0.555570233019602
        .float  -0.844853565249707,-0.534997619887097
        .float  -0.857728610000272,-0.514102744193222
        .float  -0.870086991108711,-0.492898192229784
        .float  -0.881921264348355,-0.471396736825998
        .float  -0.893224301195515,-0.449611329654607
        .float  -0.903989293123443,-0.427555093430282
        .float  -0.914209755703531,-0.40524131400499
        .float  -0.923879532511287,-0.38268343236509
        .float  -0.932992798834739,-0.359895036534988
        .float  -0.941544065183021,-0.33688985339222
        .float  -0.949528180593037,-0.313681740398891
        .float  -0.956940335732209,-0.290284677254462
        .float  -0.96377606579544,-0.266712757474898
        .float  -0.970031253194544,-0.242980179903264
        .float  -0.975702130038528,-0.21910124015687
        .float  -0.98078528040323,-0.195090322016129
        .float  -0.985277642388941,-0.170961888760301
        .float  -0.989176509964781,-0.146730474455362
        .float  -0.99247953459871,-0.122410675199216
        .float  -0.995184726672197,-0.0980171403295608
        .float  -0.99729045667869,-0.0735645635996677
        .float  -0.998795456205172,-0.049067674327418
        .float  -0.999698818696204,-0.0245412285229123
VENTA:          .float  0.015503641236052,0.0308861602749473
        .float  0.0461475571166862,0.0612878317612682
        .float  0.0763069842086935,0.0912050144589622
        .float  0.105981922512074,0.12063770836803
        .float  0.135172372026829,0.149585913488471
        .float  0.163878332752956,0.178049629820285
        .float  0.192099804690457,0.206028857363472
        .float  0.219836787839331,0.233523596118034
        .float  0.247089282199579,0.260533846083968
        .float  0.2738572877712,0.287059607261276
        .float  0.300140804554195,0.313100879649957
```

```
.float  0.325939832548562,0.338657663250011
.float  0.351254371754304,0.363729958061439
.float  0.376084422171418,0.388317764084241
.float  0.400429983799906,0.412421081318415
.float  0.424291056639768,0.436039909763963
.float  0.447667640691002,0.459174249420884
.float  0.47055973595361,0.481824100289179
.float  0.492967342427592,0.503989462368847
.float  0.514890460112946,0.525670335659889
.float  0.536329089009675,0.546866720162304
.float  0.557283229117776,0.567578615876092
.float  0.577752880437251,0.587806022801254
.float  0.5977380429681,0.607548940937789
.float  0.617238716710321,0.626807370285697
.float  0.636254901663916,0.645581310844979
.float  0.654786597828885,0.663870762615634
.float  0.672833805205227,0.681675725597662
.float  0.690396523792942,0.698996199791064
.float  0.70747475359203,0.715832185195839
.float  0.724068494602492,0.732183681811988
.float  0.740177746824327,0.74805068963951
.float  0.755802510257536,0.763433208678405
.float  0.770942784902118,0.778331238928674
.float  0.785598570758073,0.792744780390316
.float  0.799769867825402,0.806673833063332
.float  0.813456676104105,0.820118396947721
.float  0.82665899559418,0.833078472043483
.float  0.839376826295629,0.845554058350618
.float  0.851610168208451,0.857545155869127
.float  0.863359021332647,0.86905176459901
.float  0.874623385668216,0.880073884540266
.float  0.885403261215158,0.890611515692895
.float  0.895698647973474,0.900664658056897
.float  0.905509545943163,0.910233311632273
.float  0.914835955124226,0.919317476419022
.float  0.923677875516662,0.927917152417145
.float  0.932035307120471,0.936032339626641
.float  0.939908249935654,0.94366303804751
.float  0.94729670396221,0.950809247679753
.float  0.954200669200139,0.957470968523369
.float  0.960620145649442,0.963648200578359
.float  0.966555133310118,0.969340943844721
.float  0.972005632182168,0.974549198322458
.float  0.976971642265591,0.979272964011567
.float  0.981453163560387,0.98351224091205
.float  0.985450196066557,0.987267029023907
.float  0.9889627397841,0.990537328347136
.float  0.991990794713016,0.993323138881739
.float  0.994534360853306,0.995624460627716
.float  0.996593438204969,0.997441293585066
.float  0.998168026768006,0.998773637753789
.float  0.999258126542416,0.999621493133885
.float  0.999863737528199,0.999984859725355
.float  0.999984859725355,0.999863737528199
.float  0.999621493133885,0.999258126542416
.float  0.998773637753789,0.998168026768006
```

```
.float  0.997441293585066,0.996593438204969
.float  0.995624460627716,0.994534360853306
.float  0.993323138881739,0.991990794713016
.float  0.990537328347136,0.9889627397841
.float  0.987267029023907,0.985450196066557
.float  0.98351224091205,0.981453163560387
.float  0.979272964011567,0.976971642265591
.float  0.974549198322458,0.972005632182168
.float  0.969340943844721,0.966555133310118
.float  0.963648200578359,0.960620145649442
.float  0.957470968523369,0.954200669200139
.float  0.950809247679753,0.94729670396221
.float  0.94366303804751,0.939908249935654
.float  0.936032339626641,0.932035307120471
.float  0.927917152417145,0.923677875516662
.float  0.919317476419022,0.914835955124226
.float  0.910233311632273,0.905509545943163
.float  0.900664658056897,0.895698647973474
.float  0.890611515692895,0.885403261215158
.float  0.880073884540266,0.874623385668216
.float  0.86905176459901,0.863359021332647
.float  0.857545155869127,0.851610168208451
.float  0.845554058350618,0.839376826295629
.float  0.833078472043483,0.82665899559418
.float  0.820118396947721,0.813456676104105
.float  0.806673833063332,0.799769867825402
.float  0.792744780390316,0.785598570758073
.float  0.778331238928674,0.770942784902118
.float  0.763433208678405,0.755802510257536
.float  0.74805068963951,0.740177746824327
.float  0.732183681811988,0.724068494602492
.float  0.715832185195839,0.70747475359203
.float  0.698996199791064,0.690396523792942
.float  0.681675725597662,0.672833805205227
.float  0.663870762615634,0.654786597828885
.float  0.645581310844979,0.636254901663916
.float  0.626807370285697,0.617238716710321
.float  0.607548940937789,0.5977380429681
.float  0.587806022801254,0.577752880437251
.float  0.567578615876092,0.557283229117776
.float  0.546866720162304,0.536329089009675
.float  0.525670335659889,0.514890460112946
.float  0.503989462368847,0.492967342427592
.float  0.481824100289179,0.47055973595361
.float  0.459174249420884,0.447667640691002
.float  0.436039909763963,0.424291056639768
.float  0.412421081318415,0.400429983799906
.float  0.388317764084241,0.376084422171418
.float  0.363729958061439,0.351254371754304
.float  0.338657663250011,0.325939832548562
.float  0.313100879649957,0.300140804554195
.float  0.287059607261276,0.2738572877712
.float  0.260533846083968,0.247089282199579
.float  0.233523596118034,0.219836787839331
.float  0.206028857363472,0.192099804690457
.float  0.178049629820285,0.163878332752956
```

```
            .float   0.149585913488471,0.135172372026829
            .float   0.12063770836803,0.105981922512074
            .float   0.0912050144589622,0.0763069842086935
            .float   0.0612878317612682,0.0461475571166862
            .float   0.0308861602749473,0.015503641236052
ENTRADA     .space   512
            .end
```

ANEXO E

PROGRAMA QUE PRESENTA EL ESPECTRO EN LA COMPUTADORA

Se presenta el listado del programa que presenta el espectro de una señal en la pantalla de la computadora. El programa está basado en la biblioteca de Borland C++ Builder.

```
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop
#include <stdio.h>
#include <math.h>
#include "Unit1.h"
#include "amcclib.c"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
AMCCHANDLE hAmcc=NULL;
DWORD bufw[32][65];
DWORD bufr[32][65];

typedef union {
    DWORD ent;
    float sal;
    } flot;

//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
      : TForm(Owner)
{
}
//---------------------------------------------------------------------------

void __fastcall TForm1::Button1Click(TObject *Sender)
{
  DWORD comand,estado;
  char cad[100];
  int i,j,k;
  FILE *Archivo;
  float temp, prom;
  flot flotante[1500];
  if(AMCC_Open(&hAmcc,0x10E8,0x5920,0,0)){
    Label1->Caption="Abierto";
```

```
    Archivo=fopen("ffttar.txt","r");
    for(i=0;i<13;i++)
      for(j=0;j<64;j++){
        fgets(cad,99,Archivo);
          for(k=0;k<10;k++)
            cad[k]=cad[k+11];
        cad[10]=NULL;
        bufw[i][j]=StrToInt(cad);
      }
    i=0;
    while(fgets(cad,99,Archivo)!=NULL){
      for(k=0;k<10;k++)
        cad[k]=cad[k+11];
      cad[10]=NULL;
      bufw[13][i]=StrToInt(cad);
      i++;
    }
    fclose(Archivo);
    AMCC_WriteRegDWord(hAmcc,0x60,0x01010101); /*conf.*/
    AMCC_WriteRegDWord(hAmcc,0x0C,0x00FFFFFF); /*nada*/
    AMCC_WriteRegDWord(hAmcc,0x0C,0x09FFFFFF); /*resADC*/
    AMCC_WriteRegDWord(hAmcc,0x0C,0x00FFFFFF); /*nada*/
    AMCC_WriteRegDWord(hAmcc,0x3C,0x01000000); /*SYSRST#*/
    AMCC_WriteRegDWord(hAmcc,0x3C,0x00000000); /*SYSRST*/
    AMCC_WriteRegDWord(hAmcc,0x0C,0x00FFFFFF); /*nada*/
    AMCC_WriteRegDWord(hAmcc,0x0C,0x11FFFFFF); /*reset al gen.*/
    AMCC_WriteRegDWord(hAmcc,0x0C,0x00FFFFFF); /*nada*/
    AMCC_WriteRegDWord(hAmcc,0x0C,0x03FFFFFF); /*enviar prog*/
    for(j=0;j<14;j++){
      comand=AMCC_ReadRegDWord(hAmcc,0x1C);
      AMCC_WriteSpaceBlock(hAmcc,0,bufw[j],0x100,AMCC_ADDR_SPACE0); /*datos*/
    }
    while((comand=AMCC_ReadRegDWord(hAmcc,0x1C))!=0x01000000);
    AMCC_WriteRegDWord(hAmcc,0x0C,0x00FFFFFF); /*nada*/
    AMCC_WriteRegDWord(hAmcc,0x0C,0x05FFFFFF); /* reset DSP */
    AMCC_WriteRegDWord(hAmcc,0x0C,0x00FFFFFF); /* nada */
    AMCC_WriteRegDWord(hAmcc,0x0C,0x06FFFFFF); /* int1 */
    AMCC_WriteRegDWord(hAmcc,0x0C,0x00FFFFFF); /* nada */
    while(1){
      comand=AMCC_ReadRegDWord(hAmcc,0x1C);
      if(comand==0x02000000) break;
    }
    AMCC_WriteRegDWord(hAmcc,0x0C,0x00FFFFFF); /* nada */
    AMCC_WriteRegDWord(hAmcc,0x0C,0x11FFFFFF); /*reset al gen.*/
    AMCC_WriteRegDWord(hAmcc,0x0C,0x00FFFFFF); /*nada*/
    AMCC_WriteRegDWord(hAmcc,0x0C,0x0AFFFFFF); /*digital*/
    while(1){
      comand=AMCC_ReadRegDWord(hAmcc,0x1C);
      if(comand==0x04000000) break;
    }
    AMCC_WriteRegDWord(hAmcc,0x0C,0x00FFFFFF); /* nada */
    AMCC_WriteRegDWord(hAmcc,0x0C,0x12FFFFFF); /*iniciar proce*/
//    AMCC_WriteRegDWord(hAmcc,0x0C,0x00FFFFFF); /*nada*/
    while(1){
      comand=AMCC_ReadRegDWord(hAmcc,0x1C);
      if(comand==0x03000000) break;
```

```
    }
    AMCC_WriteRegDWord(hAmcc,0x0C,0x00FFFFFF); /*nada*/
//    Label1->Caption="FIN";
//  AMCC_WriteRegDWord(hAmcc,0x0C,0x00FFFFFF); /*nada*/
  AMCC_WriteRegDWord(hAmcc,0x0C,0x11FFFFFF); /*reset al gen.*/
  AMCC_WriteRegDWord(hAmcc,0x0C,0x00FFFFFF); /*nada*/
  for(int j=0;j<22;j++)
    AMCC_ReadSpaceBlock(hAmcc,0,bufr[j],0x100,AMCC_ADDR_SPACE0); /*datos*/
  for(int j=0;j<22;j++){
    for(int i=0;i<64;i++){
      flotante[64*j+i].ent=bufr[j][i];
    }
  }
  PaintBox1->Repaint();
  PaintBox2->Canvas->Pen->Color=clBlack;
  PaintBox2->Repaint();
  PaintBox1->Canvas->Pen->Style=psDot;
  PaintBox1->Canvas->MoveTo(0,0);
  PaintBox1->Canvas->LineTo(250,0);
  PaintBox1->Canvas->LineTo(250,200);
  PaintBox1->Canvas->LineTo(0,200);
  PaintBox1->Canvas->LineTo(0,0);
  for(int i=0;i<10;i++){
    PaintBox1->Canvas->MoveTo(i*25,0);
    PaintBox1->Canvas->LineTo(i*25,200);
  }
  for(int i=0;i<8;i++){
    PaintBox1->Canvas->MoveTo(0,i*25);
    PaintBox1->Canvas->LineTo(250,i*25);
  }
  PaintBox1->Canvas->Pen->Style=psSolid;
  temp=-flotante[256].sal*50.0 + 100.0;
  PaintBox1->Canvas->MoveTo(0,temp);
  for(int i=0;i<51;i++){
    temp=-flotante[i+256].sal*50.0 + 100.0;
    PaintBox1->Canvas->LineTo(i*5,temp);
    }
  PaintBox2->Canvas->MoveTo(0,0);
  PaintBox2->Canvas->LineTo(400,0);
  PaintBox2->Canvas->LineTo(400,200);
  PaintBox2->Canvas->LineTo(0,200);
  PaintBox2->Canvas->LineTo(0,0);
  for(int i=0;i<14;i++){
    PaintBox2->Canvas->MoveTo(i*30+2,200);
    PaintBox2->Canvas->LineTo(i*30+2,209);
    }
  prom=0;
  for(int i = 0;i<129;i++){
    prom+=flotante[i].sal;
  }
  prom/=129.0;
  for(int i=0;i<129;i++){
    temp=-20*log10(flotante[i].sal) + 100.0;
    PaintBox2->Canvas->MoveTo(i*3+1,200);
    PaintBox2->Canvas->LineTo(i*3+1,temp);
    PaintBox2->Canvas->MoveTo(i*3+2,200);
```

```
    PaintBox2->Canvas->LineTo(i*3+2,temp);
    PaintBox2->Canvas->MoveTo(i*3+3,200);
    PaintBox2->Canvas->LineTo(i*3+3,temp);
    }
  PaintBox2->Canvas->Pen->Color=clBlue;
  for(int i = 0;i<9;i++){
    PaintBox2->Canvas->MoveTo(1,(i+1)*20);
    PaintBox2->Canvas->LineTo(399,(i+1)*20);
  }
  for(int i=0;i<14;i++){
    PaintBox2->Canvas->MoveTo(i*30+2,1);
    PaintBox2->Canvas->LineTo(i*30+2,199);
    }
  AMCC_Close(hAmcc);
   }
   else
     Label1->Caption="Cerrado";
}
//---------------------------------------------------------------------------
```