Transferencia de datos por puerto USB de una tarjeta FPGA Nexys 2 empleando LabWindows CVI v9.0

Juan Raúl Rodríguez Suárez [1]

[1] *Universidad de Pinar del Rio. Grupo de Diagnóstico Avanzado de Maquinarias GIDAM. Departamento de Telecomunicaciones y Electrónica*
jotar@tele.upr.edu.cu

RESUMEN / ***ABSTRACT***

En el trabajo se presenta la implementación de la comunicación por puerto USB entre una computadora con sistema operativo Windows y una tarjeta FPGA Nexys2. Dicha tarjeta posee un microcontrolador Cypress CY7C68013A que maneja un puerto USB2 de alta velocidad que ha sido programado para emular un puerto paralelo EPP.
Se expone el diseño y evaluación de una interfaz gráfica de usuario con programación LabWindows que maneja la biblioteca DPCUTIL y que posibilita la transferencia y visualización de datos desde la Nexys2. La aplicación permite la transferencia en tiempo real de un procesador digital de 12bits, de dos canales implementado con frecuencia de muestreo de 1MHz en la FPGA. En el trabajo se evalúa los resultados alcanzados en cuanto a la velocidad de transferencia de datos efectiva.

Palabras claves: Comunicaciones, FPGA, Puerto USB, transferencia de datos.

This paper deals with the implementation of USB communication between a PC with Windows OS and a Nexys2 FPGA Board .This board has a Cypress Microcontroller that handles a high speed USB port and it is configured to emulate a EPP port. This work presents the design and evaluation of GUI that is supported on LabWindows CVI 9, and use the driver DPCUTIL.dll to control the data transfers for Nexys2. An FPGA application has been designed that responds to the request of PC via USB and provides data transfer with an ADC converter of 2 channels, 12 bits, with a 1 MHz sampling frequency. The LabWindows Application provides a graphical monitoring of data transfer and post processing of data, taking advantage of the library functions that are available in LW. Also data transfer speed is evaluated.

Key words: communications, data transfers, FPGA, USB Port.

INTRODUCCIÓN

La interfaz USB se ha convertido en un estándar para la comunicación de una PC con diversos periféricos como cámaras, impresoras, amplificadores de audio y memorias externas así como con sistemas digitales basados en microcontroladores, procesadores digitales (DSP) y FPGA. Tal es el caso de la tarjeta FPGA Nexys2 [1] que posee un puerto USB que permite la alimentación de la tarjeta, la programación de la misma para implementar una aplicación determinada y la transferencia de datos entre una aplicación y la PC. El fabricante suministra una aplicación para el sistema operativo Windows denominada Adept[2] -que es usada fundamentalmente para programar el circuito FPGA, aunque también posibilita el diagnóstico de la misma y la transferencia de datos. Sin embargo no brinda el código de dicha aplicación por lo que no puede ser aplicada o expandida para procesar transferencias especificas que sean necesarias tales como un sistema de adquisición de datos o simple procesador digital que estén implementados en la FPGA.

En este trabajo se presenta y analiza el diseño de una aplicación de interfaz gráfica en LabWindows/CVI v9 que permite el control, visualización y post procesamiento de la transferencia de datos con un circuito sintetizado en la FPGA. La aplicación en la FPGA es un controlador de conversión análogo- digital ADC que permite la digitalización de hasta dos señales analógicas a una frecuencia de muestreo máxima de 1 MHz y resolución de 12 bits. Las muestras generadas por los conversores son enviadas por el puerto USB hacia la PC. Aunque se trata de una aplicación específica como se analiza en detalle la metodología empleada puede ser fácilmente aplicada o modificada para nuevas aplicaciones.

MATERIALES Y MÉTODOS.

FUNDAMENTOS DE LA COMUNICACIÓN USB.

Para analizar la comunicación USB entre una PC y un dispositivo USB se deben de considerar los siguientes elementos de hardware y software.

El bus USB es del tipo “maestro/esclavo”, donde el maestro es la PC (también llamado ¨host ¨) que es el que puede iniciar la comunicación y el tráfico de datos con el esclavo ó dispositivo USB. Así las interrupciones no son posibles.

El dispositivo USB que se conecta a la PC debe tener un circuito controlador USB que responde a los requerimientos de la PC para su identificación y para recibir y enviar señales. Todo dispositivo USB es un dispositivo ¨inteligente¨ pues mediante el circuito controlador USB es capaz de responder a los eventos que se presenten en el bus USB. En nuestro caso la Nexys 2 posee un circuito integrado Controlador USB-CYPRESS 68001A y un circuito FPGA Xilinx Spartan3E xcs3e1200 entre otros. El circuito controlador es el que recibe las señales de la PC por el conector USB y se conecta con determinados pines de la FPGA, el mismo ha sido programado para emular un puerto paralelo de virtual de alta velocidad EPP[3]. Por lo tanto cualquier aplicación que se desarrolle en la FPGA que pretenda transferir datos mediante el puerto USB, lo hace como si el puerto USB fuera en realidad un puerto EPP. Para implementar la comunicación USB en la Nexys2, primeramente en la FPGA hay que implementar un bloque controlador EPP que interprete y genere las señales EPP que recibe o envía el controlador CYPRESS, para la FPGA es como si estuviera conectada a un puerto EPP.

Por otra parte en la propia PC se halla disponible un controlador USB pero configurado para iniciar la comunicación como “maestro”. Cuando un dispositivo USB se conecta con la PC si es un dispositivo estándar en el sistema operativo se encuentra un programa driver que identifica el dispositivo automáticamente y le asigna un identificador, y permite que dentro del sistema operativo se ejecute la operación del dispositivo. Si es un dispositivo USB no estándar se necesita de un programa driver específico que provea las funciones necesarias previstas para el dispositivo. Este driver por lo general es suministrado por el fabricante del dispositivo USB. El driver para la Nexys2 es una biblioteca denominada “dpcutil.lib” que permite transferencias de datos[4] que ha sido escrita y compilada con Microsoft C++. Esta biblioteca de funciones genera y procesa los datos transferidos e interpretados en este caso según el protocolo EPP. Para poder controlar la comunicación USB con LabWindows/CVI, esta biblioteca debe previamente ser cargada en dicho programa para poder emplear las funciones que ella dispone y que les permite entenderse con el controlador CYPRESS de la Nexys2.

EL PROTOCOLO EPP.

El puerto EPP permite la transferencia de datos con un conjunto de registros definidos en el periférico en cuestión. Este puerto se encuentra definido en una PC con puerto paralelo. Pero en nuestro caso el puerto EPP es virtual y se encuentra definido en el controlador CYPRESS que se conecta con el circuito FPGA que entonces actúa como periférico. Un puerto EPP dispone de dos puertos: un puerto de datos y un puerto de direcciones. El puerto de datos permite definir la dirección asociada al registro que se pretende transferir (leer o escribir) y el puerto de datos recibe el valor asociado a la transferencia en cuestión (el dato que se escribe o se lee). Para ello el puerto EPP emplea físicamente un único bus bidireccional de 8 bits, llamado bus de datos D [7:0] que permite la transferencia de datos y direcciones. Para identificar si el valor en el bus de D se corresponde con una dirección o con un dato, se emplean dos señales de validación: DSTB para los datos y ASTB para las direcciones, las cuales se activan con el nivel bajo y que además en todo momento solo una pueda estar activa. Por otra parte se emplea otra señal de control WR que indica el sentido de la transferencia. Si WR esta a nivel alto, indica que se efectúa una lectura y si esta a nivel bajo se una escritura. Las señales DSTB, ADST y WR son señales de entrada se suministran al periférico (del controlador CYPRESS hacia la FPGA) pero existe una señal WAIT que es generada por el periférico como salida (del FGPA hacia el controlador CYPRESS), que es una señal de reconocimiento de acceso. El dispositivo (el circuito FPGA) pone esta señal a nivel bajo para indicar que no se están realizando transferencias hacia el por el puerto (DSTB o ASTB a nivel alto). Pero cuando el puerto inicia un transferencia determinada y activa alguna de las señales a nivel bajo, se encuesta la señal WAIT que se recibe del FPGA, si esta señal permanece a nivel bajo, se insertan estados de espera en la transferencia por el puerto hasta que el dispositivo responde con nivel alto en WAIT que le indica al puerto que puede terminar la transferencia y subir la señal de validación correspondiente.

Existen por lo tanto cuatro transferencias posibles: escritura de direcciones, lectura de direcciones, escritura de datos y lectura de datos. Sin embargo cuando se va a escribir o leer en un registro del periférico en realidad se realizan las siguientes transferencias:

La lectura de un registro implica primero escribir la dirección asociada al registro en cuestión y después leer el dato que almacena este registro, por la tanto se realizan consecutivamente las transferencias de escritura y lectura. Primeramente en el bus de D se carga el valor de la dirección del registro de interés y se activa ASTB para indicar al periférico que se trata de una dirección, la línea WR se baja para indicar que se está efectuando una escritura, cuando finaliza esta transferencia se activa la señal DSTB indicando que se va a procesar un dato, la línea WR se pone en alto para indicar que se va a leer del registro cuando la transferencia de lectura se completa el dato leído se carga en bus. La secuencia de señales se presenta en la figura 1.

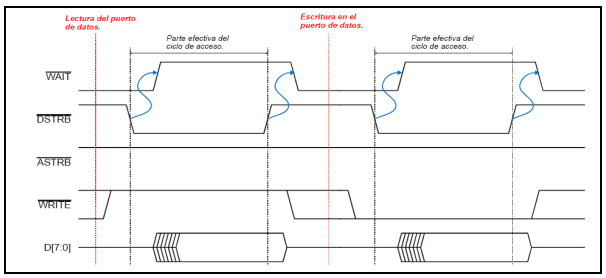

Figura 1. Lectura de un registro EPP.

La escritura de un registro por otra parte implica primero escribir la dirección asociada al registro y después escribir el dato del registro, por lo tanto se realizan las transferencias dos transferencias de escritura. La secuencia de señales se presenta en la figura 2.

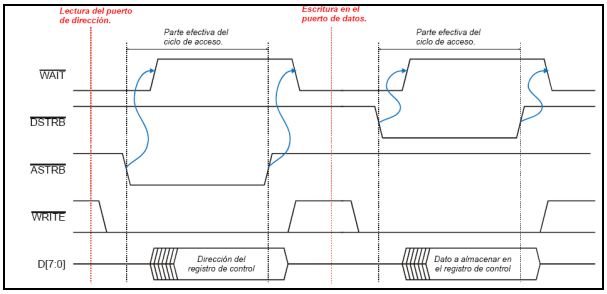

Figura 2. Escritura de un registro EPP.

REQUERIMIENTOS DEL DISEÑO.

En la PC se necesita implementar un programa de alto nivel que trabaje en el sistema operativo Windows, que pueda carga la biblioteca dpcutil.dll (el driver USB de la Nexys2) y que permita identificar la identificación de la misma cuando se conecta al puerto USB y el control de las transferencias que solicite, así como el procesamiento y visualización gráfica de los datos recibidos.

En la Nexys2 hay tres tipos de circuitos integrados que intervienen en esta aplicación: el controlador USB CYPRESS ya descrito anteriormente , dos circuitos conversores análogo digital ADC para digitalizar dos señales analógicas y obtener así las muestras que serán transmitidas hacia la PC por el puerto USB y la FPGA propiamente dicha. En la figura 3 se presenta un diagrama de bloques de dichas conexiones. Por lo tanto en la FPGA hay que sintetizar primeramente dos bloques de control asociados a cada uno de estos circuitos: un controlador ADC que genere las señales de control para el conversor ADC y reciba las muestras tomadas de las señales analógicas y un controlador EPP que se comunique con el puerto virtual EPP implementado el controlador USB.
Puesto que el protocolo EPP se basa en la lectura y escritura de registros, a las muestras generadas por el conversor ADC hay que asociarles un registro con una determinada dirección para que pueda ser leído en la PC.

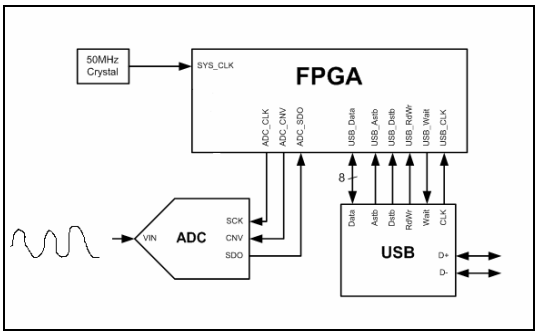

Figura 3. Disposición de circuitos en la Nexys2.

DISEÑO DEL CIRCUITO FPGA.

a) Conversor ADC

Se ha seleccionado un módulo PMOD AD1[5] que dispone de dos circuitos de conversión análogo digital ADS7476[6] en formato serie de 12 bits, unipolar, con una frecuencia de muestreo de hasta 1 MHz, así como de los filtros antialiassing correspondientes. Por tratarse de conversores de formato serie, solamente son necesarias dos líneas de control CS – selección del circuito a nivel bajo - y SCLK – reloj serie - que además son comunes a ambos circuito. Cada uno de los conversores responde a las señales de control generando un flujo de datos serie con cada una de las transiciones del reloj serie cuando esta activado, a través del terminal DIN. Así solamente se necesitan 4 líneas de conexión hacia el circuito FPGA donde se debe instrumentar el controlador correspondiente. El reloj CS es el que define la frecuencia de muestreo, y trabaja a la máxima frecuencia del conversor. Si se desean frecuencias de muestreo menores entonces se emplea un bloque diezmador a fin de obtener la frecuencia de interés.

b) Controlador del conversor ADC

El primer elemento a considerar es el controlador de los conversores ADC. Este bloque genera las señales de control CS, SCLK, recibe las salidas series DIN1, DIN2, y produce las salidas de las muestras DATA1, DATA2 con formato punto fijo de 12,0.

El diseño de este circuito se realizó con la herramienta XSG que permite la síntesis, la simulación e implementación de circuitos digitales en componentes FPGA de Xilinx sobre el ambiente de Matlab/Simulink y que es especialmente favorable para trabajar con procesamiento digital de señales. Para lograr el trabajo del circuito ADS7476 es preciso primero suministrar la señal de reloj serie SCLK y la de inicio de conversión CS. Ambas señales se generan en base a bloques contadores. La señal SCLk se implementa con un contador de 4 niveles de la señal de reloj del sistema de 50 MHz, obteniéndose un divisor por 4 resultando finalmente en un reloj SCLK de 12.5 MHz. La señal CS que habilita la conversión en nivel bajo se implementa también con un contador pero ahora de 20 niveles, y la frecuencia de reloj se escoge de 12.5 MHz para así

generar un pulso periódico con un ciclo útil del 80 %. Esta señal define la frecuencia de muestreo de 20 periodos de SCLK e igual a 625kHz y también es usada para inhabilitar la generación del reloj serie cuando CS este a nivel alto. El conversor ADS7476 responde generando 16 pulsos de datos series, por lo tanto se incluye un bloque ¨black box¨ que enlaza el sistema con un fichero vhdl que permite la conversión serie paralelo, y que actualiza un registro de 12 bits cada 625 kHz, tomando el dato de entrada cada transición negativa del reloj serie SCLK. En la figura 4 se muestra el diseño del controlador ADC en el ambiente XSG.

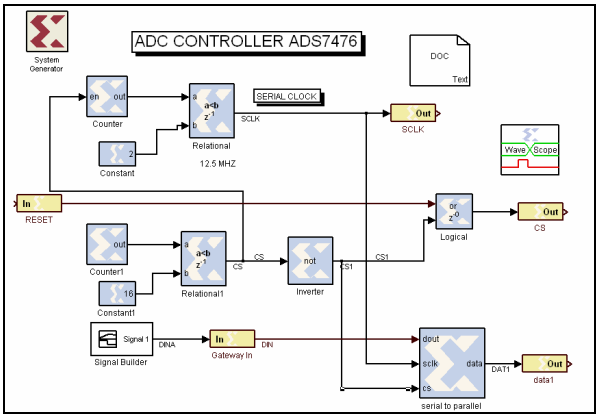

Figura 4. Diseño del controlador ADC con XSG.

Un aspecto particularmente importante en la síntesis de circuitos FPGA es la simulación, pues solamente con ella podemos tener una cierta certeza de que el circuito funcione adecuadamente. En este caso la simulación se puede efectuar también en el ambiente de Simulink con el bloque WaveScope, detalles de la misma se muestran en la figura 5. En este caso se generó la señal serie de prueba simulando la respuesta del conversor con un bloque signal builder disponible en Simulink

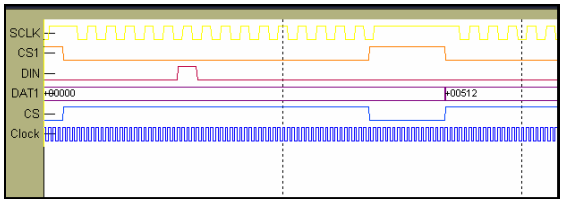

Figura 5. Detalles de la simulación del controlador ADC.

c) Transferencia de datos en bloques

En muchas aplicaciones típicas de procesamiento digital de señales se emplea el procesamiento en tiempo real mediante un flujo de datos con un tiempo de muestreo determinado, y donde se requiere que el tiempo de procesamiento de una muestra por determinado sistema sea menor que el tiempo de muestreo. Esto asegura que la muestra a la salida de un sistema se procese antes del arribo de la siguiente muestra a la entrada.

En la comunicación USB los datos se transmiten en paquetes. Un paquete está formado por una serie de campos o conjuntos de bits, entre los cuales se encuentran: el campo sincronismo SYNC que define el inicio del paquete, el campo de identificación ID, el campo de dirección, el campo ENDPOINT ,el campo del número de FRAME, el campo de datos que puede ser de hasta 1024 bytes para USB2 y los campos para detección de errores NRC.

El tamaño del campo de datos depende del tipo de transferencia, de la velocidad del bus y de la cantidad de datos de la transferencia. El protocolo USB2 soporta tres tipos de velocidad de transferencia: baja - de 800 Bytes/s, total - de 1.2 MBytes/s y alta de 53 MBytes/s. Sin embargo la velocidad de transferencia de la aplicación es siempre menor que la velocidad de transferencia del bus.

Los circuitos de conversión análogo digital ADS7476 generan un flujo de datos (muestras) en la FPGA de 12 bits con un velocidad máxima de 1MHz que queremos transmitir mediante el puerto USB, que da un equivalente de 2MB/s .El controlador USB de la tarjeta emula un puerto paralelo EPP, que realiza las operaciones de transferencias mediante escrituras y lecturas sobre registros. Por lo tanto de alguna forma debemos asociar un registro con el flujo de muestras generadas. Puesto que la transmisión USB se efectúa mediante paquetes de datos, la forma más eficiente de operación es asociar o agrupar un grupo de muestras en un bloque que se transmita como un todo y no transmitir las muestras una por una a medida que se obtenga. Por ello se emplea un bloque de memoria RAM donde se almacena un conjunto de muestras recibidas del conversor y se asocia a esta memoria RAM un registro en la FPGA que pueda entonces ser leído por el puerto EPP.

Para formar un bloque de datos, las muestras tomadas del conversor ADC se envían hacia la entrada de una memoria RAM. Como se almacenan datos de un tamaño de un byte se incluye un bloque que divide los datos de 12 bits en dos datos de 8 bits, empleando un multiplexor de dos entradas controlado por un reloj de selección de 1.25 MHz que permite obtener 1 byte al doble de la frecuencia de muestreo. Como se van a almacenar 2048 datos en la RAM se incluye también un bloque que genera las direcciones para la misma, en base a un contador de 11 bits con un reloj también de 1.25 MHz.

Finalmente se realiza la compilación de este bloque dentro de Matlab empleando la compilación tipo ngc. La misma genera un fichero único en formato ngc que comprime toda la información del diseño del sistema que ha sido realizado con XSG y que debe de ser adicionado a un proyecto ISE como fuente para que pueda ser incluido en un diseño a un nivel superior como un componente del sistema. Esta compilación brinda además un fichero vho que comprende una plantilla en formato vhdl con brinda detalles de cómo crear el componente y como inicializarlo en un diseño final.

d) Controladores RAM y EPP

En este trabajo se ha empleado como punto de partida para el diseño del sistema la aplicación BRamCfg[7] que fue desarrollada por Digilentic. Esta aplicación crea una memoria

RAM interna de doble puerto en la FPGA (BRam.vhd) , el controlador de la memoria (BRamComCtl.vhd) y un bloque de interface EPP (EppCtrlAsync.vhd) y la misma permite efectuar transferencias de escrituras y lecturas por el puerto USB. En la aplicación original, el puerto A de la memoria RAM se emplea en lecturas y escrituras y el puerto B se deja libre. En nuestra aplicación empleamos el puerto B de la memoria para escribir los datos tomados por los conversores ADC. De esta forma la memoria se escribe continuamente por el conversor ADC mediante el puerto B y se lee por el puerto A mediante el puerto EPP. Los bloques empleados se describen brevemente a continuación.

Memoria RAM

El bloque de memoria RAM permite almacenar 2048 datos de 1 byte con dos puertos. El puerto B se emplea para almacenar las muestras del conversor (escritura), el puerto A se emplea para lectura por el puerto EPP.

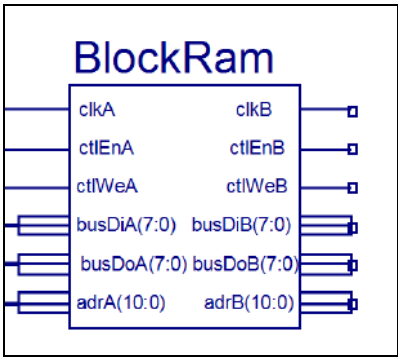

Figura 6. Bloque de Memoria RAM

Este bloque se implementa directamente con una función primitiva de Xilinx Logicore disponible en la herramienta de diseño ISE. Dispone de dos puertos A y B que pueden ser escritos o leidos independientemente pero que comparten las mismas localizaciones de memoria. Cada puerto posee las entradas de control correspondientes: reloj clk habilitación En, escritura We, entrada Di , salida Do y direcciones adr. Este bloque se ha configurado para emplear parte de la memoria de bloque disponible en la RAM con un tamaño de 2048 unidades de 1 Byte. El puerto B se usa sólo para escribir en la memoria las muestras tomadas del conversor y el puerto A se usa sólo para leer los datos. Por lo tanto las direcciones y la entrada de datos del puerto B se toman del bloque ADC descrito anteriormente. El puerto A se maneja con el bloque controlador de memoria que se explica a continuación. Este bloque tiene dos clientes en cuestión el controlador ADC (puerto B) y el bloque controlador RAM (puerto A).

Controlador de RAM

Este bloque realiza el control de la memoria RAM, para ello asocia el registro de 8 bits con la salida de datos del puerto A con el fin de efectuar la lectura del mismo, en cada operación de lectura incrementa automáticamente hacia la proxima dirección, y habilita la señal DSTB del puerto EPP como reloj para el puerto A de la RAM y habilita los puertos A y B convenientemente

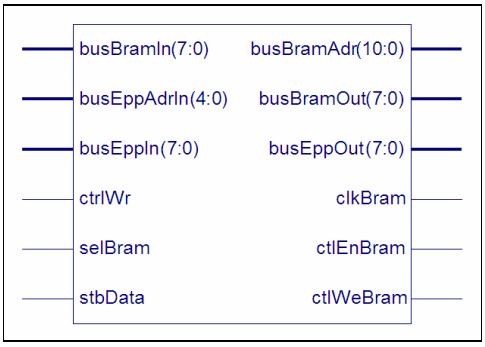

Figura 7 .Controlador RAM

El controlador RAM es cliente del bloque final que es el controlador EPP, por ello recibe una copia de las señales de validación ASTB y de WR del puerto EPP en sus entradas stbData y ctrlData , procesa el bus D bidireccional de EPP en forma de dos puertos independientes busEppIn que recibe datos desde EPP(escritura) y busEppOut que envía datos desde el FPGA (lectura) y recibe una señal para seleccionar la memoria RAM-A. Ademas en el bloque se generan las señales para el manejo del puerto A del bloque de memoria RAM (clk, En, We, Adr) y recibe por la entrada BusBramin la salida del puerto A de la RAM como respuesta a una petición de lectura. De esta forma se lee una muestra tomada del conversor ADC que ha sido almacenada por una escritura del bloque conversor por la entrada del puerto B de la RAM. Esta lectura se envía por la salida busEppOut .

En el bloque se implementa un registro interno de 8 bits denominado regBramAdr que es un puntero a las direcciones del puerto A de la memoria RAM para la próxima dirección a leer. Si la dirección de este registro es 1 se está indicando el byte menos significativo de una dirección, si es 2 se está indicando el byte más significativo y si es cero se está indicando el valor del dato leído o escrito en la dirección. Como la transferencia de un dato por el puerto EPP (escritura o lectura) se activa con el nivel bajo de la señal DSTB, la misma se complementa a fin de generar el reloj del puerto A de la memoria. Además el flanco positivo de DSTB que indica el fin de una transacción de dato se emplea para incrementar la dirección de la próxima celda a leer.

CONTROLADOR EPP

Este controlador es el que maneja directamente la transferencia de datos con el circuito CYPRES 68001A por lo tanto recibe las señales de control correspondientes al puerto EPP (ASTB,DSTB y WR) y el puerto bidireccional D, y genera la señal Wait. En este bloque se crea el registro interno de direcciones regEppAdr de 8 bits que puede ser leído o escrito cuando se valida un dato como dirección (ASTB a nivel bajo), se escribe con WR bajo y se lee con nivel alto. Los 3 bits mas significativos de este registro pueden ser usados

como selección de diferentes clientes hasta un total de 15. Los 5 bits restantes definen hasta 32 posibles direcciones de parejas de registros de escritura y lectura en un cliente. En esta aplicación podemos usar dos clientes, correspondientes a cada uno de los canales de los dos conversores ADC disponibles.

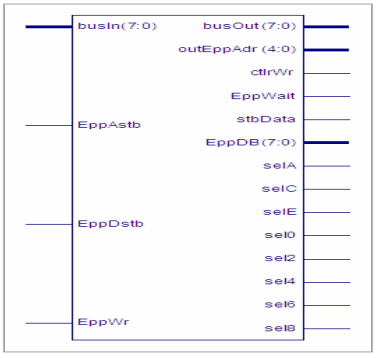

Figura 8. Controlador EPP

Los clientes emplean las señales derivadas de DSTB y WR para el control de su aplicación y se hace una copia de las señales de validación para el controlador de memoria, y reemplaza el bus bidireccional en dos buses independientes de entrada y salida que son usados por los clientes.

DISEÑO DE LA INTERFAZ GRAFICA DE USUARIO.

Se escogió el programa LabWindows porque es un programa en lenguaje compatible ansi C , por lo que permite aplicar la biblioteca dpcutil.dll . Este programa implementado en la PC es el que controla toda la operación de la transferencia de datos con la Nexys2 y brinda la interfaz grafica de usuario que facilita su empleo. Para ello primeramente es necesario cargar en el proyecto previamente la biblioteca dcptutil.lib, así como los proyectos cabecera asociados a la misma: dcputil.h, gendefs.h y dpcdefs.h , ello permite emplear libremente el conjunto de funciones de que dispone esta biblioteca.

El primer paso antes de emplear cualquier función de la biblioteca, es verificar que la misma se ha inicializado correctamente, para ello se llama a la función *DpcInit* que debe retornar el valor 1 . Igualmente cuando la aplicación finaliza se debe llamar a *DpcTerm* para cerrar.

Las principales funciones de la biblioteca dpcutil.dll que se ha usado son:

DvmgGetDefaultDev. Permite obtener el índice del dispositivo.
DvmgGetDevName. Permite obtener el nombre del dispositivo e identificador
DpcOpenData. Con el nombre del dispositivo, se abre la comunicación y devuelve un manipulador.
DpcGetRegRepeat . Con el manipulador de la comunicación y la dirección del registro (en nuestro caso asociado a la memoria RAM) se puede efectuar la lectura de dicho registro un número de veces (en nuestro caso dado por el tamaño de la RAM) y almacenar los datos en una variable tipo arreglo.
DpcCloseData. Terminada la transferencia se cierra la comunicación USB y se cierra posteriormente la biblioteca

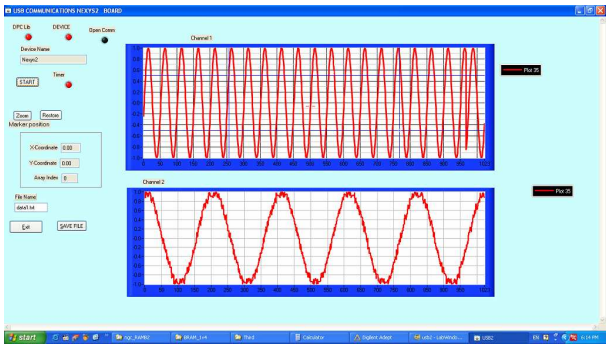

Figura 9. Interfaz gráfica

En la figura 9 se presenta el diseño de la interfaz grafica. Se destacan las dos graficas asociadas a cada uno de los conversores, que permiten visualizar los datos. Estas graficas se actualizan cada un segundo. Cada grafica pose tres cursores, dos de ellos para proveer zoom y el tercero permite leer datos desde la grafica. Asociado a cada grafico hay una función de llamada que permite el zoom o el restablecimiento del formato original de la grafica.

Se pueden salvar los datos en un fichero texto para un análisis posterior, especificando además el nombre del fichero. Cuando se realiza click izquierdo con el mouse sobre el botón START, se hace una llamada a la biblioteca dcputil.dll, se obtiene el nombre del dispositivo y se iluminan los leds indicadores de la comunicación. También se habilita el temporizador programado a un segundo. A partir de ese momento cada un segundo se toman los datos del puerto y se muestran en las graficas. Con click derecho en el botón start, se inhabilita el temporizador, y se pueden procesar datos con los cursores sobre las graficas o se puede finalizar la operación finalmente accionando sobre el botón EXIT.

DISCUSIÓN DE RESULTADOS.

El diagrama de bloques completo del circuito sintetizado en la FPGA se muestra en la figura 10 y comprende los controladores para los circuitos USB y ADC , así como dos bloques de RAM y sus respectivos controladores y un bloque EPP. La síntesis final del circuito se realizó en lenguaje vhdl con la herramienta de Xilinx ISE 8.3

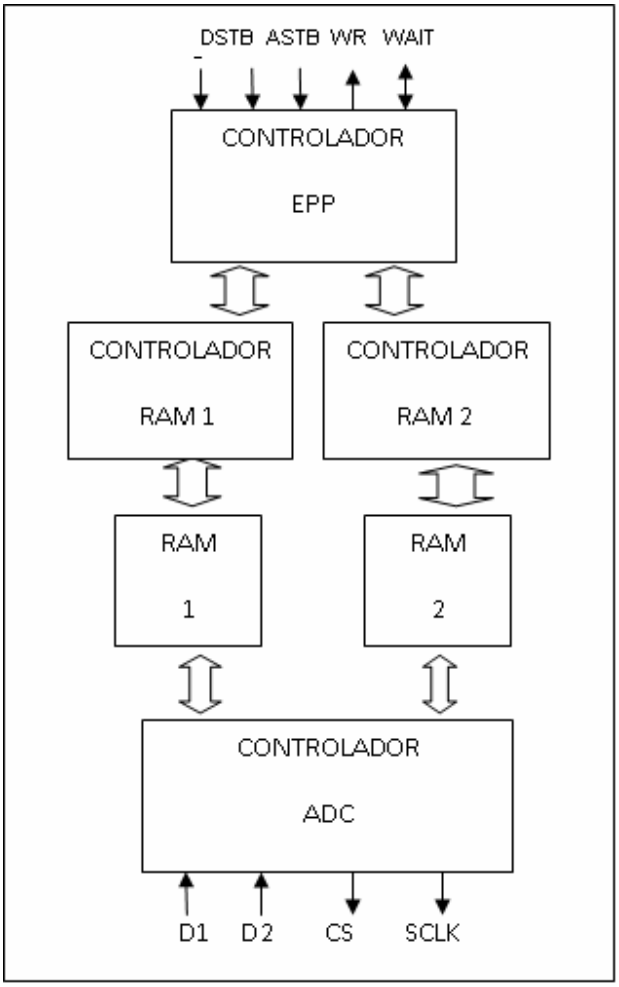

Figura 10. Diagrama de bloques general

Para el ajuste toda la aplicación, se procedió a comprobar cada una de sus partes por separado y luego en conjunto. El programa LabWindows se ajustó cargando el circuito FPGA con la aplicación dpimref.vhd[3] que permite comprobar la operación de la Nexys2 y que se basa en la transferencia de datos entre distintos registros asociados a periféricos de la tarjeta como los interruptores, leds, lámparas de siete segmentos. Normalmente se usa con la aplicación ADEPT, en este caso se usó LabWindows lo cual nos permitió familiarizarnos con el uso de las diversas funciones de la biblioteca dpcutil.dll que se emplearon.

Posteriormente se procedió al ajuste del circuito en la FPGA. Para ello primeramente se ajustó el bloque de control del conversor ADC, comprobando la generación de las señales de control CS y SCLK y la correcta generación del código del conversor en la conversión serie paralelo. Se diseñó entonces una aplicación de prueba donde las muestras digitales se generan a partir de un bloque sintetizador DDS en una memoria ROM interna que es direccionada por un circuito acumulador. La memoria se muestrea a 1 MHz y la constante del acumulador permite variar la frecuencia de la señal sinosoidal que se desea generar. Esta aplicación comprende también la generación de la direcciones para la memoria RAM. Con esta aplicación se realizó la compilación en formato ngc que permitió la generación de un fichero vhd que es usado como componente en la aplicación final.

Se realizó una primera variante del circuito FPGA a partir de la aplicación descrita BramCfg, pero cargando el puerto B de la RAM con los datos generados por la aplicación DDS y la generación de las direcciones. Esta circuito se ajustó con la aplicación en LabWidows. Finalmente se sustituyó el bloque sintetizador DDS por el control del conversor ADC

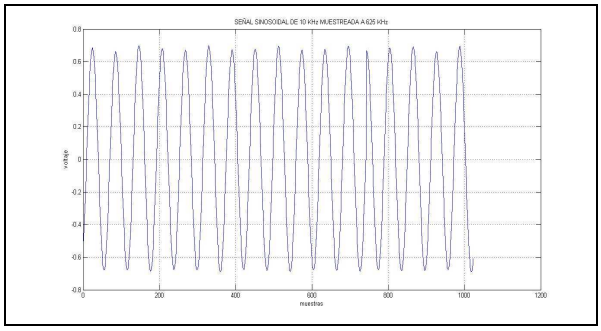

Figura 11. Señal en el tiempo capturada por la interfaz USB y analizada en Matlab.

En la figura 11 se muestra un fichero de datos capturado por la aplicación LabWindows y procesado en Matlab. La señal original es generada por un generador de funciones a 10.3 kHz, y es muestreada a 625 kHz por el conversor ADS7476 acoplado a la nexys2. En la figura 12 se muestra el espectro de la misma señal calculado con la transformada rápida de Fourier con 8192 puntos. Se observa que la frecuencia del pico coincide apreciablemente con la del generador en 10.38KHz. La frecuencia máxima de muestreo obtenida es de 1MHz y que está determinada por el tipo de conversor usado, determina la velocidad de transferencia mínima de la aplicación de 2MB/s, con dos conversores en paralelo resulta entonces de 4MB/s.

En la aplicación inicial se realizó una transferencia práctica de 4096 Bytes/s resultado de transferir 1024 muestras de 12 bits en dos canales cada segundo. Por otra parte la RAM de la FPGA puede configurarse hasta una dirección de 16 bits, lo que resulta en 131072 B/s.

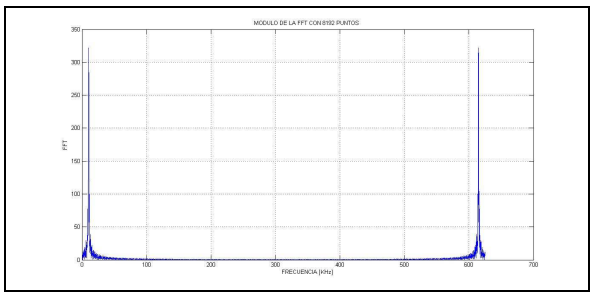

Figura 12. Análisis en frecuencia de la señal de la figura 11.

CONCLUSIONES

Se ha implementado un circuito sobre la FPGA de la tarjeta Nexys2 que permite la transferencia de datos tomados de dos conversores ADC de; tipo ADS7476 de 12 bits, con una frecuencia muestreo de 1 MHz hacia una PC mediante el puerto USB.

La transferencia de datos es controlada por una aplicación en lenguaje LabWindows, que permite la visualización de los datos, su procesamiento y almacenamiento de los mismos en ficheros. En el programa LabWindows se ha cargado la biblioteca driver de la aplicación dpcutil.dll y se han empleado diversas funciones de la misma.

El circuito pose entre otros bloques un controlador EPP que se comunica con el circuito controlador USB disponible en la Nexys2 que emula un puerto EPP y un controlador para el manejo de los conversores ADS7476 especialmente diseñado al efecto. Se han implementado bloques de memoria y controladores para las mismas para facilitar la transferencia en el protocolo USB.

REFERENCIAS

1. **DIGILENT**. (Jan 21, 2008). Digilent Nexys2 Board Reference Manual. [on line]. Disponible: http://www.digilentic.com.

2. **DIGILENT.** (Aug 27,2007). Digilent Adept Quick Start Guide. [on line]. Disponible: http://www.digilentic.com.

3. **APPERSON G.** (08/10/2004) . Digilent Parallel Interface Model Reference Manual [on line]. Available: http://www.digilentic.com.

4. **DIGILENT**. (06/03/05). Digilent Port Communications Programmers Reference Manual. [on line]. Disponible en: http://www.digilentic.com.

5. **DABACAN,** I. (July 7, 2008). *PmodAD1 Reference Component*. [on line]. Disponible: http://www.digilentic.com.

6. **NATIONAL SEMICONDUCTOR.** (Sep 2007). ADCS7476/ADCS7477/ADCS7478 1MSPS, 12-/10-/8-Bit A/D Converters in SOT-23 & LLP. [on line]. Disponible: www.national.com.

7. **DIGILENT ROMANIA.** (July 18, 2008). BramCfg *Reference Project*. [on line]. Disponible: http://www.digilentic.com.

AUTOR

Juan Raúl Rodríguez Suárez. Licenciado en Física. 1974 UH, Doctor en Ciencias Técnicas 1989. ISPJAE. Actualmente es Profesor Auxiliar del Departamento de Telecomunicaciones y Electrónica de la UPR y miembro del Grupo de Investigación de Diagnóstico Avanzado de Maquinarias GIDAM. Imparte las asignaturas de PDS, e Instrumentación Electrónica. Experiencia en Microelectrónica de 1973-1995, también en sensores, simulación, Instrumentación virtual, procesamiento digital de señales, Microcontroladores y Sistemas FPGA. Profesor Invitado en Etiopía 2003 y 2009- 2010.