ENTORNO DE TRABAJO DE PRÁCTICAS CON CPLDS

Francisco Javier Quiles Latorre[1], Carlos Diego Moreno Moreno[1], Manuel Agustín Ortiz López[1], José Ignacio Benavides Benítez[1], Miguel Ángel Montijano Vizcaino[1]

[1]Universidad de Córdoba. el1qulaf@uco.es

RESUMEN

En esta comunicación se describe el entorno de trabajo que se utiliza en el diseño de sistemas digitales en la asignatura Diseño de Sistemas Microcomputadores: síntesis mediante PLDs, del 3º curso de ingeniería Técnica en Electrónica Industrial. El entorno consta de las herramientas software ispLEVER (diseño y simulación) e ispVM System (programación) de Lattice Semiconductor, y la tarjeta UCOisp1032, diseñada y desarrollada por el grupo de Arquitecturas Avanzadas de Computadores de la Universidad de Córdoba. La tarjeta incluye el CPLD ispLSI1032E–100LJ84 de Lattice Semiconductor. Además de analizar las herramientas y la tarjeta, se describe las prácticas realizadas y la metodología, que debe seguir el alumnado, para desarrollarlas.

1. INTRODUCCIÓN

Actualmente es imprescindible usar Dispositivos Lógicos Programables en el diseño e implementación de los Sistemas Electrónicos Digitales. Por ello, se ha incluido la asignatura optativa Diseño de Sistemas Microcomputadores: Síntesis mediante PLDs en la titulación de Ingeniero Técnico en Electrónica Industrial, que tiene por objetivo estudiar la metodología de diseño y desarrollo de Sistemas Digitales mediante PLDs.

Se realizan prácticas de simulación y laboratorio. Las de simulación se desarrollan mediante la herramienta que posteriormente se indicará, y se emplean PLDs Simples (GAL22V10 y GAL16V8) y el CPLD idóneo para cada caso de la familia IspLSI1K de Lattice Semiconductor. Respecto a las prácticas de laboratorio, se comenzó realizando prácticas solamente para los PLDs Simples, ya que éstos son muy fáciles de montar en protoboard, a diferencia de los CPLDs por tener encapsulados SMD. Para realizar prácticas con CPLDs evidentemente se debe emplear una placa de desarrollo. Existen diversas alternativas en el mercado para los distintos fabricantes de CPLDs, pero ninguna ofrece un interface al bus PCI a un bajo coste. Éste ultimo requisito se consideró imprescindible, ya que una de las Líneas de Trabajo de nuestro Grupo de Investigación, implica el diseño y desarrollo de placas de Adquisición de Datos o de cualquier otro tipo para PC. Por tanto, se decidió que la mejor opción era diseñar y desarrollar una placa que sirviese para la docencia, y que a su vez, nos permitiese realizar un interface básico al bus PCI.

A continuación se describirá el Entorno de Trabajo que se emplea actualmente, centrándonos tanto en la herramienta software como en la placa desarrollada, y su aplicación docente, indicando las practicas realizadas y el proceso de trabajo.

2. DESCRIPCIÓN DEL ENTORNO DE TRABAJO

El entorno de trabajo consta de los medios necesarios para, por una parte describir el diseño y simularlo, y por otra, programar el CPLD usando tecnología ISP (In-System Programmable) y comprobar su funcionamiento real. Por tanto, el entorno de trabajo consta de dos herramientas software de lattice Semiconductor (ispLEVER y ispVM System) para el diseño, simulación y programación de los CPLDs, y la tarjeta UCOisp1032, que ha sido diseñada y desarrollada

por el Grupo de Investigación de Arquitecturas Avanzadas de Computadores de la Universidad de Córdoba.

2.1. Herramienta software ispLEVER v2.0

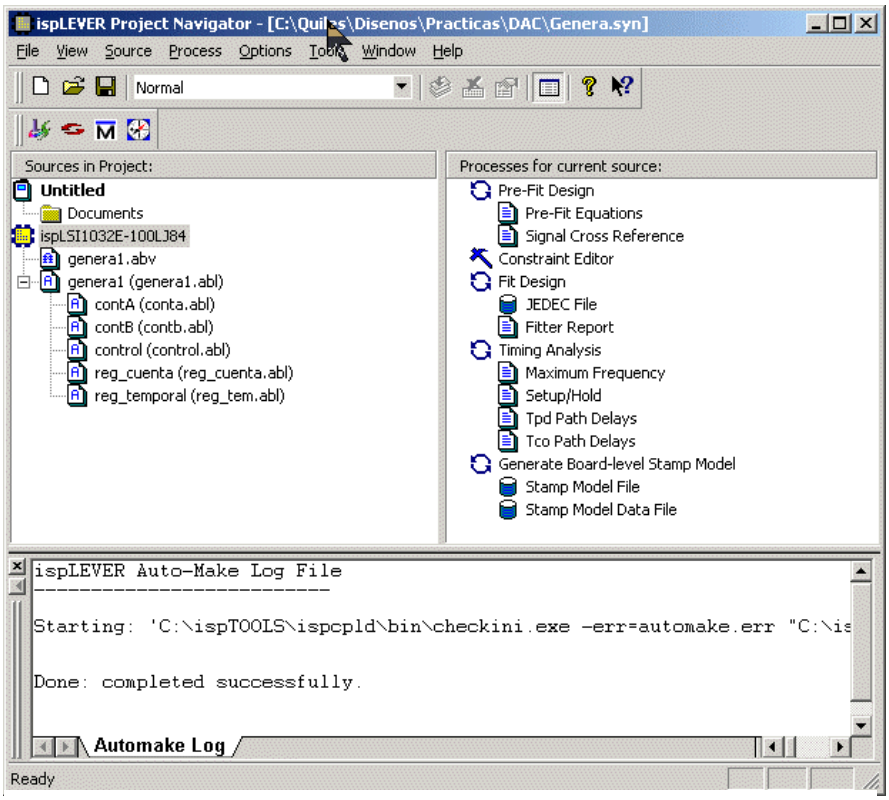

Figura 1. Ventana principal de la herramienta ispLEVER 2.0

IspLEVER v2.0 es la herramienta que se emplea para describir el diseño, simularlo y generar el fichero JEDEC de programación [1][2][3]. Existen en el mercado diversas herramientas, pero se seleccionó ispLEVER 2.0, ya que es la que mejor cumplía los objetivos que debía reunir el entorno de trabajo:

- Entrada del diseño mediante ABEL-HDL
- Entrada del diseño mediante VHDL
- Descripción de diseños jerárquicos
- Diseño de PLD Simples como las GAL22V10 y GAL16V8
- Diseño de CPLD de 128 macrocélulas como mínimo
- Simulación JEDEC y temporal
- Generación del fichero de simulación VHDL

La versión Starter 2.0 de ispLEVER tiene también la ventaja de que es gratuita, por lo que se puede instalar en el Centro de Cálculo sin ningún coste, y a su vez los alumnos la pueden instalar en sus ordenadores personales. En la figura 1 se muestra la pantalla principal de ispLEVER 2.0, denominada Navegador de Proyectos. Se distinguen tres ventanas: fuentes del proyecto, procesos para la fuente actual y visualización de ficheros. También incluye un Editor de texto ASCII, que se emplea para editar los ficheros fuente del diseño.

El entorno de prácticas se está usando actualmente en la asignatura Diseño de Sistemas Microcomputadores: Síntesis mediante PLDs, que corresponde a la titulación de Ingeniero Técnico en Electrónica Industrial. Dado que en la titulación de 2º ciclo de Ingeniero en Automática y Electrónica Industrial, se imparte el lenguaje VHDL en la asignatura de

Sistemas Electrónicos Digitales, se ha decido usar el lenguaje ABEL-HDL en la asignatura de 1er ciclo [3][4][5]. Por otra parte ABEL-HDL es un lenguaje más sencillo de aprender que VHDL, y además permite describir directamente las características de las señales del diseño sin tener que usar atributos, como ocurre en el lenguaje VHDL.

Tal y como se puede observar en la figura 1, las fuentes del proyecto son el CPLD (ispLSI1032E-100LJ84), el fichero de vectores de test (*.abv) y los ficheros fuente (*.abl) descritos en lenguaje ABEL-HDL. En la ventana de procesos se indica los que se pueden ejecutar según la fuente que esté seleccionada. En la figura 1 está seleccionado el dispositivo. En este caso los procesos posibles serían preimplementación del diseño, implementación del diseño (generación de los fichero JEDEC y report de información sobre la implementación), análisis temporal (frecuencia máxima y tiempos de retardo, establecimiento y mantenimiento), y generación del modelo para análisis temporal (Fichero de modelo stamp).

Si se selecciona el fichero de vectores de test la ventana de procesos cambia, de forma que se visualizan los procesos correspondientes a la simulación funcional y temporal. Si elegimos un fichero fuente los procesos permitidos serán los relativos a la comprobación de sintaxis y generación de ecuaciones simplificadas.

La ventana de visualización muestra el contenido de los ficheros que se genera en función del proceso que se esté ejecutando. El fichero Automake.log indica lo que hace el proceso y los warning o errores que se generen, como puede ser los fallos de sintaxis del fichero fuente.

2.2. Herramienta software ispVM System

Es una herramienta de programación de PLD a través del conector JTAG [6]. Al igual que ispLEVER la proporciona Lattice Semiconductor sin coste alguno. Por tanto, para programar el CPLD solamente hay que conectar el conector JTAG de la placa mediante el cable ispDOWNLOAD modelo pDS4102-DL2 al puerto paralelo de un computador personal, y descargar el fichero JEDEC de programación mediante la herramienta ispVM System.

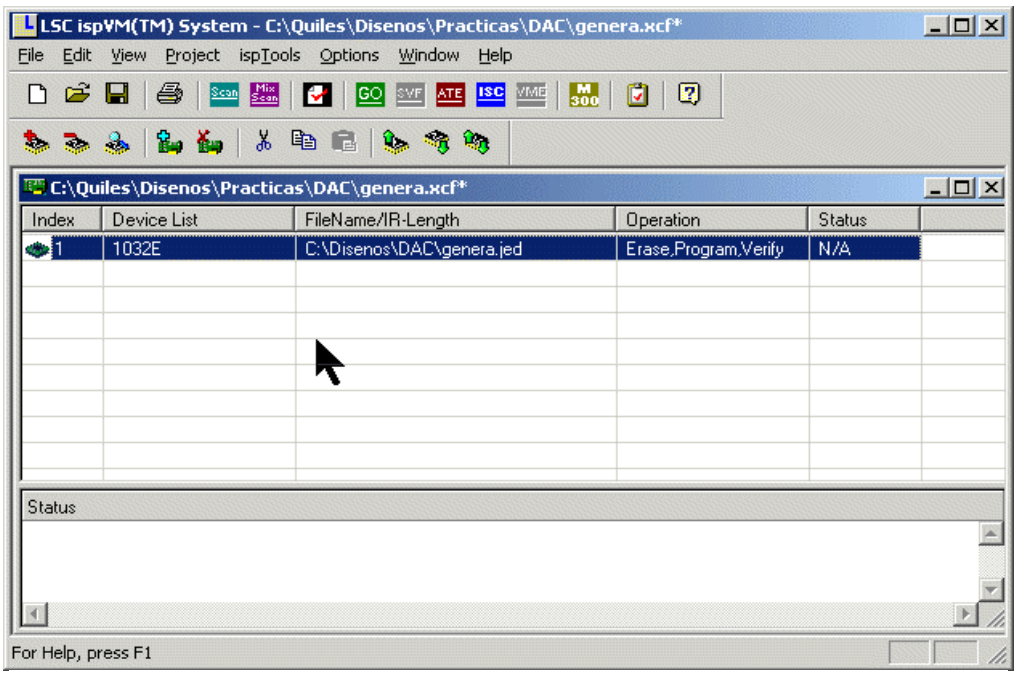

Figura 2. Ventana principal de la herramienta de programación ispVM System

El uso de la tecnología ISP tiene múltiples ventajas, entre las que cabe destacar:

- Reducción del coste, ya que no es necesario un equipo de programación.
- Se facilita el proceso de programación, ya que no se debe extraer el CPLD de la placa, y el proceso, como posteriormente se verá, es mucho más sencillo que en un equipo de programación.
- Se alarga la vida del CPLD o del zócalo en el que esté insertado, al no tener que extraerlo de la placa.

En la figura 2 se muestra la ventana principal de ispVM System. Posteriormente se indicará el proceso de programación.

2.3. Tarjeta UCOisp1032

Esta tarjeta incluye el CPLD ispLSI1032E-100LJ84 de Lattice Semiconductor [7][8]. Entre sus características más importantes cabe destacar:

- Conexión al bus PCI de 32 bits, 33Mhz y 5V
- Alimentación externa o interna desde el bus PCI
- Interfaz TTL-RS232 y conector DB9
- Microinterruptor de 4 circuitos para configuración de señales de entrada
- 8 leds para monitorizar señales
- Un visualizador de 7 segmentos de cátodo común
- 2 conectores de expansión para conectar placas de periféricos
- Programación del CPLD a través del conector JTAG
- Funcionamiento autónomo o en un slot PCI de un ordenador personal

Se ha seleccionado este CPLD, ya que es el único con tecnología ISP que contiene 128 macrocélulas en un encapsulado PLCC de 84 patillas. Los restantes CPLD tienen un encapsulado del tipo QFP, por lo que es más difícil soldarlo que el anterior, ya que en éste simplemente hay que usar un zócalo PLCC-PGA para convertir las patillas de SMD a Through-hole.

En la figura 3 se muestra el diagrama de bloques de la placa. La mayoría de los bloques ya se han comentado al indicar las características de la placa. Sobre el resto, cabe comentar que se ha incluido una GAL22V10 para generar el reset al CPLD. Éste se activa por tres causas:

- Señal de reset del bus PCI
- Reset de encendido. Se genera mediante un red RC
- Pulsador de reset

La GAL controla los LED y el visualizador de 7 segmentos. Por tanto, puede funcionar como buffer o decodificador BCD a 7 segmentos. En el modo buffer se usa para monitorizar el valor de una serie de señales de E_S por medio de los diodos LED. El modo de funcionamiento lo controla el CPLD mediante una señal. La selección entre diodos LED y visualizador se realiza mediante un puente.

Se ha incluido un MAX232 y un conector DB9 para implementar un puerto serie a 3 hilos (Transmisión, recepción y masa). La fuente de alimentación genera una tensión continua estabilizada de +5V a partir de una fuente continua externa comprendida en el rango de +7V a +12V. La placa también se puede alimentar a través del bus PCI. La configuración de la tensión de alimentación se realiza mediante un puente.

El CPLD ispLSI1032E tiene 4 entradas de señal de reloj. Una se conecta a la señal de reloj de 33MHz del bus PCI, otra a la de un oscilador de cuarzo y las otras dos al conector externo DIN de 26 contactos.

Las señales del bus PCI están accesibles en un conector de expansión de 64 contactos. De esta manera se cumplen dos objetivos:

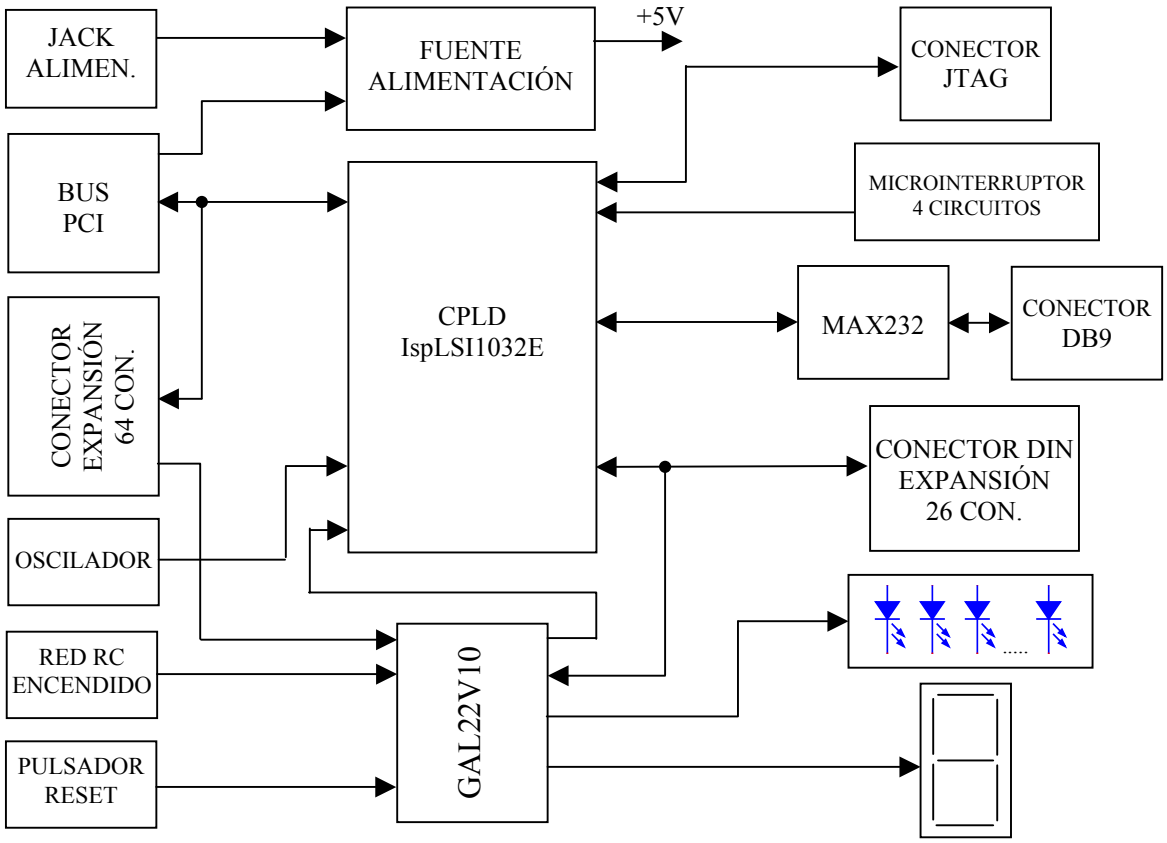

Figura 3. Diagrama de bloques de la placa UCOisp1032

- Conectar los POD de un Analizador Lógico para estudiar los cronogramas del bus PCI.
- Conectar una placa de expansión.

Actualmente se están desarrollando varias placas de expansión, que incluyen SRAM, Flash EPROM, conversores A/D y D/A, relés, sensores y triacs, de forma que se pueda realizar prácticas relativas a la materia de Electrónica Industrial, tales como, implementar un generador digital de señal, un termómetro digital, temporizador programable de encendido o apagado, control de iluminación, control de motores paso a paso, etc.

3. APLICACIÓN DOCENTE

El entorno se está empleando en las prácticas de la asignatura de Diseño de Sistemas Microcomputadores: Síntesis mediante PLDs, que se imparte en el 3º curso de la Titulación de Ingeniero Técnico en Electrónica Industrial. Esta asignatura tiene asignados 4.5 créditos de los cuales 1.5 son de prácticas. A continuación se describe algunas de las prácticas realizadas por el alumnado usando el entorno de trabajo objeto de esta comunicación.

3.1. Contador BCD con salida de display 7 segmentos

Se propone diseñar un contador BCD pero que realice la cuenta en un display de 7 segmentos. Para ello habría dos opciones: implementar en un módulo un contador BCD, y en otro módulo un decodificador de BCD a salida de 7 segmentos; o bien, y más apropiada, implementar en un solo módulo un contador mediante tablas de verdad que vaya contando directamente en el formato del display de 7 segmentos.

El sistema tendrá tres entradas de control síncronas (RESET, ENABLE y TEST) y una salida de acarreo (CARRY). RESET es la entrada de puesta a cero, ENABLE la de habilitación de la cuenta y TEST una entrada de comprobación del display que al activarse se activarán todos los segmentos del display.

3.2. Circuito de control de un ascensor

Se plantea el diseño del circuito de control de un ascensor de 3 pisos. La selección del piso se hace por medio de tres botones, cuyo estado se indica por las señales P0, P1 y P2. P0 corresponde al piso bajo, P1 al piso 1º y P2 al piso 2º. Si se pulsan al mismo tiempo varios botones se selecciona el piso más próximo, teniendo en cuenta que si el ascensor está parado en el piso 1 (intermedio) se seleccionará el piso bajo.

El sistema secuencial controlará el motor del ascensor generando dos señales: SUBIR y BAJAR, según el sentido del movimiento, que debe experimentar.

La señal de entrada PULSO, cuando se pone a nivel alto, indica que el ascensor pasa por algún piso, por lo que se debe tener en cuenta para determinar la secuencia de subida o bajada. Esta señal es síncrona y se activa sólo durante un pulso de reloj.

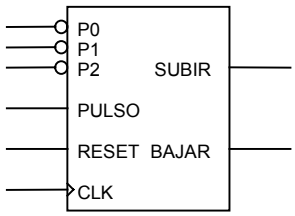

Figura 4

3.3. Temporizador

El objetivo de esta práctica es el diseño e implementación del sistema de la figura 5, que corresponde a un circuito temporizador semejante al de control de una luz de escalera.

El funcionamiento del sistema es el siguiente: al activarse la señal INICIO debe encenderse el LED y permanecer encendido 10 ó 20 segundos según la señal S1020, contados a partir de que se desactive INICIO. Una vez encendido si se vuelve a activar INICIO se recarga la cuenta. La señal RESET, activa a nivel bajo, es el pulsador de RESET de la tarjeta.

El sistema está formado por tres bloques: DIVISOR8M es un divisor de frecuencia, al cual le llega una señal de reloj de 8 MHz y genera una señal de 1 Hz, para que la cuenta se realice en segundos; CONTADOR es un contador descendente que realiza la cuenta de 20 o de 10 segundos según la señal S1020; y CONTROLADOR es un autómata que controla el funcionamiento de todo el sistema.

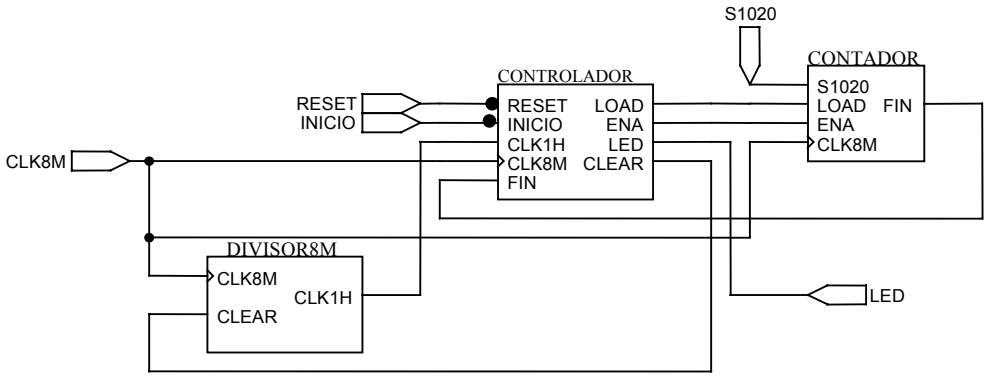

Figura 5. Diagrama de bloques del temporizador.

3.4. Medida de la anchura de un pulso

La función del sistema es medir la anchura del pulso a nivel alto aplicado en la entrada PULSO. La frecuencia de reloj es de 1 MHz y la del pulso varía entre 10 KHz y 100 KHz. La relación de simetría del pulso es del 50%, es decir, está el mismo tiempo a nivel alto que a nivel bajo. El visualizador representa la anchura del pulso en μs.

El sistema consta, además del bloque de control, de un contador BCD con entrada de habilitación (ENA) y puesta a cero (CLEAR) y el número necesario de decodificadores BCD a 7 segmentos y visualizadores. El contador BCD es el encargado de medir la anchura del pulso y el registro se utiliza para almacenar el valor entre cada actualización.

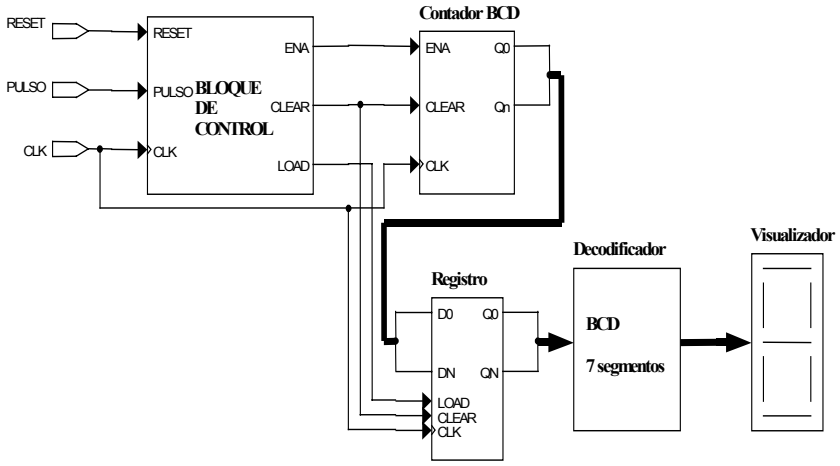

Figura 6. Diagrama de bloques del medidor de anchura de un pulso

El bloque de control realiza la siguiente secuencia: En el instante inicial y mientras se active RESET se visualiza un 0 en el display, una vez desactivo RESET analiza el nivel lógico de PULSO. Si PULSO está a nivel bajo debe mantener el valor 0 en el visualizador, y si está a nivel alto, debe medir su anchura mediante el contador. Cuando PULSO pasa a nivel bajo transfiere la cuenta que determina la anchura al registro, y la mantiene hasta que se reciba otro pulso y se determine su anchura.

3.5. Transmisor serie asíncrono

Se plantea el diseño del transmisor de una UART. Los caracteres son de 11 bits, de los cuales 8 son de datos, uno de start (inicio), uno de paridad par y uno de stop (parada). La velocidad de transmisión es de 9600 bits por segundo (bps). Las señales del transmisor se indican en la figura 7:

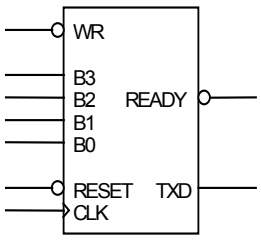

Figura 7

RESET inicializa todos los componentes del transmisor a un estado conocido, que posteriormente se indicará. Es asíncrona y activa a nivel bajo. CLK es la señal de reloj de 40 MHz que sincroniza el funcionamiento de todos los bloques. B3, B2, B1 y B0 son las entradas de datos en las que se representan los números en código BCD. WR es la señal de escritura, es asíncrona y activa a nivel bajo, almacena el número en el transmisor e inicia la transmisión. TXD es la salida de la transmisión serie. READY indica el estado del transmisor, si se activa (nivel bajo) indica que se puede escribir un dato nuevo.

Se pretende conectar el transmisor con el puerto serie de un PC, y visualizar los números que se envíen en el monitor mediante el hiperterminal de Windows. Debido a que éste usa el

código ASCII, se debe convertir los números decimales del código BCD al código ASCII de 8 bits.

El diagrama de bloques se indica en la figura 8:

El registro buffer solamente realiza la operación de carga paralela, su función es la de almacenar el número a transmitir. El registro serializador tiene un tamaño de 10 bits y realiza las operaciones de carga paralela y desplazamiento a la derecha, realiza la conversión de paralelo a serie. El generador de baudios genera la señal de reloj que sincroniza la transmisión del dato a la velocidad de 9600 bps. El bloque de control está compuesto de una máquina de estados, que genera las señales de control a los tres componentes de forma que se realice la transmisión adecuadamente y genera la señal de estado Ready.

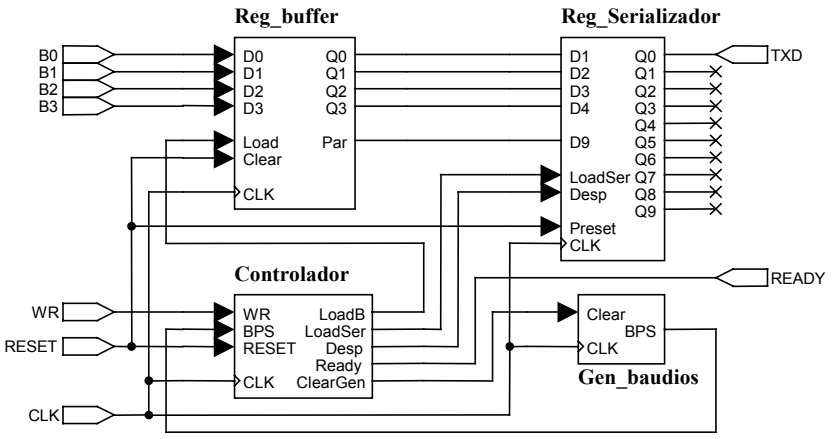

Figura 8. Diagrama de bloques del transmisor serie asíncrono

3.6. Generador digital de señal

En esta práctica se le propone al alumnado diseñar y describir en ABEL como un diseño jerárquico el bloque de control de un generador digital de funciones, que generará una señal triangular de frecuencia programable. Consta de un DAC (conversor digital a analógico) de 8 bits de resolución y un bloque de control, que es el objetivo de este diseño. El conversor D/A realiza lógicamente la conversión digital a analógico, y el bloque de control genera la secuencia de señales necesaria para escribir sucesivamente los valores discretos en el DAC, y obtener la señal analógica de salida.

La estructura interna del bloque de control se representa en la siguiente figura 9.

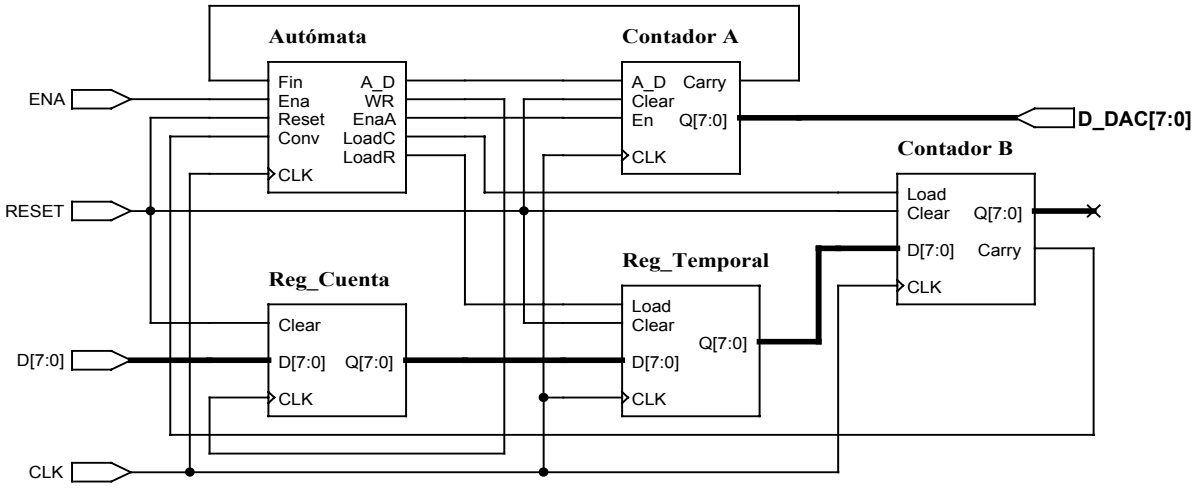

Figura 9. Diagrama de bloques del generador de señal triangular

El contadorA es un contador binario de 8 bits reversible. Se utiliza para generar los valores discretos de la señal triangular, que deben escribirse sucesivamente en el DAC. El contadorB es un contador binario de 8 bits descendente, usado para generar la frecuencia de conversión. El registro de cuenta almacena el valor de cuenta inicial del contadorB escrito por el microprocesador. El registro temporal se emplea para almacenar la cuenta inicial de forma que se pueda cambiar el contenido del registro de cuenta durante el proceso de conversión. Finalmente, el autómata genera la secuencia de señales para controlar el funcionamiento del resto de componentes.

Las señales de entrada y salida del sistema son las siguientes: RESET inicializa todo el sistema y pone a cero los dos contadores y registros; CLK frecuencia de reloj de 8 MHz; Ena habilita el proceso de conversión de un periodo completo; WR_R carga del registro de cuenta, con el flanco de subida de WR_R se almacena el dato de las entradas D[7..0] en el registro; D[7..0] entrada de datos del registro de cuenta; D_DAC[7..0] son las entradas de datos del DAC, que corresponde a las salidas del contadorA; WR es la señal de escritura del DAC.

3.7. Otras prácticas

Otras prácticas que se le propone al alumnado como optativas, son las siguientes: receptor serie asíncrono (como continuación de la práctica 3.5, transmisor serie asíncrono), termómetro digital, control de un motor paso a paso, control de potencia mediante triacs.

3.8. Observaciones

Para poder comprobar adecuadamente el funcionamiento del diseño en la prácticas de los apartados 3.5 y 3.6, se ha diseñado una placa que contiene un DAC, un monoestable, un microinterruptor de 8 circuitos y un pulsador. El monoestable genera un pulso de escritura, libre de rebotes, cada vez que se pulse el pulsador. El microinterruptor se usa para fijar el número BCD que se va a transmitir (transmisor serie), o para determinar la frecuencia de conversión (generador digital de señal).

El transmisor serie también se puede comprobar, insertando otra placa UCOisp1032 en un slot PCI de un computador personal, y conectándola a la placa que implementa el transmisor a través del conector DIN. La placa PCI implementa un bus de salida de 4 bits y su correspondiente señal de escritura, activa a nivel bajo. Para escribir el número BCD simplemente hay que ejecutar el comando O (output) del Debuger de MS-DOS.

4. PROCESO PARA REALIZAR UNA PRÁCTICA

En la figura 10 se indica la secuencia de procesos para realizar un diseño mediante un CPLD con la herramienta ispLEVER 2.0. Evidentemente, ésta es semejante a la de la mayoría de las herramientas.

4.1. Entrada del diseño

Una vez especificado el problema se realiza su diseño. Si es un sistema complejo, que puede dividirse en varios componentes, conviene diseñarlo como un sistema jerárquico. En este caso, si se usa la metodología descendente, se parte de una descripción RTL del sistema, y posteriormente se describe los componentes de cada uno de los niveles. Antes de realizar la descripción del diseño, debe crearse un proyecto del tipo ABEL/Esquemáticos, y seleccionar el dispositivo ispLSI1032E-100LJ84.

La descripción se realiza mediante el lenguaje ABEL-HDL. Cada componente se describe mediante un módulo y cada módulo se describe en un fichero fuente diferente, aunque se podrían incluir todos los módulos de un nivel en el mismo fichero fuente. En la figura 1, en la ventana de fuentes del proyecto, se muestra la estructura jerárquica de módulos y ficheros para la práctica del generador digital de señal.

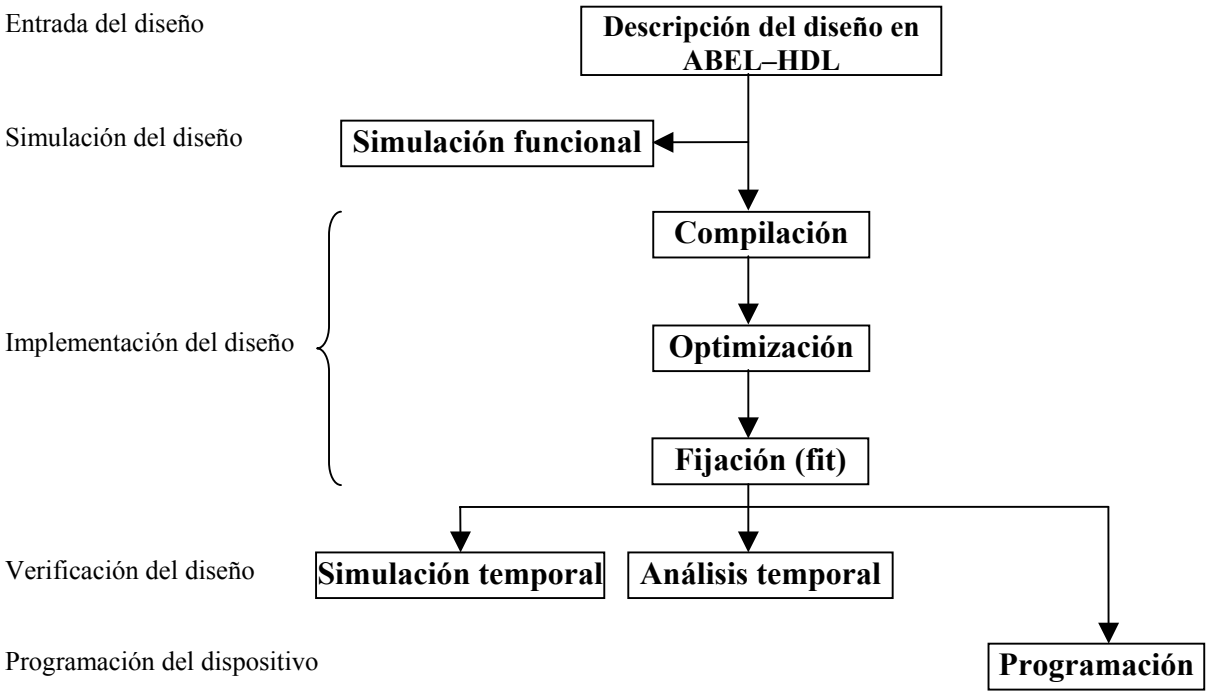

Figura 10. Diagrama de flujo del desarrollo de una práctica

4.2. Simulación del diseño

IspLEVER incluye un simulador que permite realizar dos tipos de simulación: funcional y temporal. En esta fase se usa la primera. Permite comprobar si el diseño es funcionalmente correcto, es decir si el comportamiento lógico del diseño descrito corresponde al que se especificó. Como se realiza antes que la implementación del diseño, no tiene en cuenta los retardos de propagación. No obstante conviene realizarla, ya que de esta forma se tiene la certeza de que se va a implementar un diseño bastante depurado desde el punto de vista de los errores lógicos de diseño.

Para realizar la simulación se debe indicar los valores de las señales de entrada. ABEL-HDL permite indicarlos mediante los vectores de test. Estos se han descrito en un fichero de vectores (genera1.abv) en vez de en el fichero fuente raíz. Posteriormente se justificará.

4.3. Implementación del diseño

Consta de 3 procesos: Compilación, Optimización y Fijación (Fit) del diseño en el CPLD. Para los diseños descritos en ABEL-HDL los tres procesos se integran en uno, por lo que se reduce el número de pasos. Así, en la ventana de procesos solamente aparece el de fijación del diseño (Fit design). El compilador realiza varios pasos de los que los más importantes son dos: comprobación de la sintaxis de los ficheros fuente y generación de las ecuaciones compiladas, que son las que se usan en los pasos siguientes. El optimizador se puede configurar. Por defecto, intenta conseguir las máximas prestaciones en el dispositivo más

pequeño posible. El fitter es el último proceso y es el que implementa finalmente el diseño en el CPLD seleccionado. Por tanto, crea el fichero JEDEC de programación. Consta de 4 pasos de los cuales los más importantes son el de partición, asignación y rutado (place and route).

En este proceso también se generan los ficheros de simulación temporal, tanto para el simulador integrador, como para simuladores VHDL (*.vho y *.sdf).

4.4. Verificación del diseño

La simulación temporal permite comprobar el funcionamiento real del sistema, ya que se realiza la simulación final del CPLD, una vez que se ha implementado el diseño. En este caso la simulación, no sólo tiene en cuenta el comportamiento lógico sino también los distintos parámetros temporales, como retardos de propagación, tiempos de establecimiento y mantenimiento, frecuencia máxima, etc. Por tanto, puede ocurrir que un diseño funcionalmente correcto genere errores en la simulación temporal.

Al describir los vectores de test en el fichero de vectores de test, no tiene que repetirse el proceso de implementación del diseño, cada vez que se modifiquen. De esta manera se agiliza el proceso de simulación.

Dado que ispLEVER ejecuta todos los procesos previos necesarios al solicitado, se puede reducir el proceso a un único paso. Así, una vez descrito el diseño basta con realizar la simulación temporal, para que ispLEVER realice previamente la implementación del diseño, y por tanto, genere el fichero de programación.

4.2. Programación del dispositivo

Tal y como ya indicó anteriormente, la programación del CPLD se realiza mediante la herramienta ispVM System de Lattice. El procedimiento de programación es bastante sencillo, ya que solamente tiene dos pasos.

Primero se escanea la cadena de dispositivos JTAG, haciendo clic en el botón SCAN. El resultado se muestra en una ventana (genera1.xcf) que permite caracterizar los dispositivos de la cadena. En cada línea se indica un dispositivo. Consta de varios campos, de los cuales, solamente hay que configurar el correspondiente al fichero de programación.

A continuación se realiza la programación del CPLD. Para ello, basta con hacer clic en el botón GO. La herramienta realiza automáticamente las operaciones indicadas en el campo correspondiente. Por tanto, primero borra el CPLD, a continuación lo programa y finalmente lo verifica.

En la figura 11 se muestra una fotografía del montaje y comprobación del funcionamiento del diseño correspondiente a la práctica del generador digital de señal.

5. CONCLUSIONES

El entorno desarrollado permite realizar prácticas con CPLDs de una forma bastante simple, al reducirse el número de pasos necesarios para desarrollar un práctica. Así, una vez que se ha descrito el diseño mediante el lenguaje ABEL-HDL, simplemente hay que ejecutar el proceso correspondiente a la simulación temporal para crear el fichero de programación y comprobar su funcionamiento real mediante el simulador que integra la herramienta ispLEVER. Si el funcionamiento simulado es correcto, gracias a la tecnología ISP, se programa fácilmente el CPLD mediante la herramienta ISPVM System, tal y como se vio anteriormente.

Al incorporar un interface RS232 y el conector PCI, junto con las placas de periféricos desarrolladas, se puede usar también el entorno en otras asignaturas de la titulación de

Ingeniero Técnico en Electrónica Industrial, como puede ser Informática Industrial, Interfaces y Periféricos, y Arquitecturas Basadas en Microprocesador.

Dado el número de macrocélulas del CPLD, solamente se ha podido implementar un interface al bus PCI básico, que implementa los comandos de configuración y E/S. Por ello, se va a desarrollar otra placa con un CPLD que integre 256 o 512 macrocélulas.

Al usar herramientas gratuitas, el coste del entorno es reducido, y a su vez facilita la realización de las prácticas al alumnado, ya que las pueden desarrollar con sus propios ordenadores.

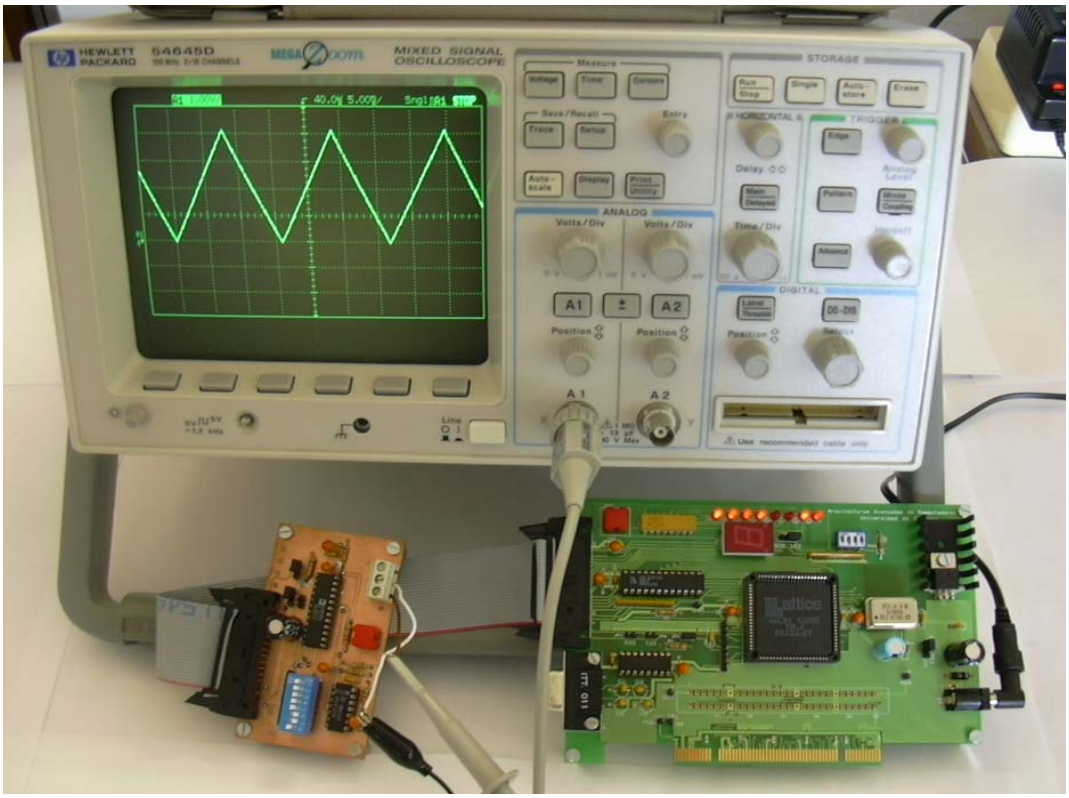

Figura 11. Montaje correspondiente a la práctica del generador digital de señal

6. BIBLIOGRAFÍA

[1] F.J. Quiles Latorre, C.D. Moreno Moreno,*"Guía básica de la herramienta ispLEVER v 2.0. Diseño de Sistemas Digitales mediante Dispositivos lógicos programables"*, Universidad de Córdoba, 2003.

[2] *"ispLEVER v2.0 Concepts Manual",* Lattice Semiconductor Corporation, 2003.

[3] http://www.latticesemi.com, Lattice Semiconductor Corporation.

[4] F.J. Quiles Latorre, "*Guía básica de diseño con el lenguaje ABEL–HDL*", Universidad de Córdoba, 2003.

[5] "*ABEL–HDL Reference Manual*" versión 8.0, Lattice Semiconductor Corporation, Marzo 2003.

[6] *"ispVM System User Manual Versión 9.0.1"*, Lattice Semiconductor Corporation, 2003.

[7] F.J. Quiles Latorre,*"Descripción de la Familia de CPLDs ispLSI1000 de Lattice"*, Universidad de Córdoba, 2003..

[8] *"ispLSI 1000EA Family Architectural Description",* Lattice Semiconductor Corporation, 2002.