UNIVERSIDAD CENTRAL "MARTA ABREU" DE LAS VILLAS
VERITATE SOLA NOBIS IMPONETUR VIRILIS TOGA. 1948

Facultad de Ingeniería Eléctrica

Departamento de Electrónica y Telecomunicaciones

TRABAJO DE DIPLOMA

Desarrollo de aplicaciones utilizando PicoBlaze en System Generator y su implementación en una FPGA de Xilinx

Autor: Ariel Rodríguez Cepero

Tutor: Ing. Yakdiel Rodríguez-Gallo Guerra

Santa Clara

2012

"Año 54 de la Revolución"

Universidad Central "Marta Abreu" de Las Villas

Facultad de Ingeniería Eléctrica

Departamento de Electrónica y Telecomunicaciones

TRABAJO DE DIPLOMA

Desarrollo de aplicaciones utilizando PicoBlaze en System Generator y su implementación en una FPGA de Xilinx

Autor: Ariel Rodríguez Cepero

E-mail: arcepero@gmail.com

Tutor: Ing. Yakdiel Rodríguez-Gallo Guerra

E-mail: yrodriguez-gallo@uclv.edu.cu

Santa Clara

2012

"Año 54 de la Revolución"

Hago constar que el presente trabajo de diploma fue realizado en la Universidad Central "Marta Abreu" de Las Villas como parte de la culminación de estudios de la especialidad de Ingeniería en Telecomunicaciones y Electrónica, autorizando a que el mismo sea utilizado por la Institución, para los fines que estime conveniente, tanto de forma parcial como total y que además no podrá ser presentado en eventos, ni publicados sin autorización de la Universidad.

Firma del Autor

Los abajo firmantes certificamos que el presente trabajo ha sido realizado según acuerdo de la dirección de nuestro centro y el mismo cumple con los requisitos que debe tener un trabajo de esta envergadura referido a la temática señalada.

Firma del Tutor

Firma del Jefe de Departamento
donde se defiende el trabajo

Firma del Responsable de
Información Científico-Técnica

PENSAMIENTO

"El conocimiento es la mejor inversión que se puede hacer"

Abraham Lincoln

DEDICATORIA

A mis padres, por apoyarme en todo momento y por creer en mí

AGRADECIMIENTOS

A mi Dios que ha sido mi socorro en los momentos más difíciles

A mi tutor por confiarme la responsabilidad de trabajar en este proyecto

A toda mi familia que siempre estuvo brindándome apoyo y cariño y por sus repetidas muestras de preocupación

Al claustro de profesores, que durante estos cinco años se ha esforzado para hacer de mí un profesional

A mis amigos del cuarto, pues sin su apoyo no lo hubiera logrado

A todos ustedes, gracias.

TAREA TÉCNICA

Para dar cumplimiento a los objetivos trazados y alcanzar los resultados esperados en este trabajo, se elaboraron las siguientes tareas técnicas.

- Caracterización de los microcontroladores empotrados y de las herramientas Matlab/Simulink, Xilinx ISE, System Generator y las FPGAs de Xilinx.
- Descripción de la arquitectura del microcontrolador PicoBlaze.
- Investigación de las prestaciones que brinda el microcontrolador PicoBlaze en System Generator.
- Elaboración de aplicaciones utilizando PicoBlaze en System Generator para su implementación en FPGAs.
- Comprobación del funcionamiento de las aplicaciones desarrolladas en una FPGA de Xilinx.
- Elaboración del informe final del Trabajo de Diploma.

Firma del Autor

Firma del Tutor

RESUMEN

En el presente trabajo de diploma se desarrollan tres aplicaciones utilizando el microcontrolador PicoBlaze en el software de modelación System Generator. Esta investigación surge debido a la necesidad de profundizar en las temáticas relacionadas con los microcontroladores empotrados a partir de las tendencias actuales del uso de estos dispositivos en una gran cantidad de aplicaciones. Para la realización de este trabajo se hizo una investigación de las principales características de los microcontroladores empotrados, presentándose algunos ejemplos, y de las prestaciones que brinda PicoBlaze en el ambiente de System Generator. También se describen las características fundamentales del Kit Nexys2 de Xilinx que posee la Facultad de Ingeniería Eléctrica y las herramientas de software Matlab/Simulink, Xilinx ISE y System Generator, elementos fundamentales en el diseño de las aplicaciones. El trabajo constituye la base para el desarrollo de futuras investigaciones sobre el tema.

ÍNDICE

PENSAMIENTO i

DEDICATORIA ii

AGRADECIMIENTOS iii

TAREA TÉCNICA iv

RESUMEN v

INTRODUCCIÓN 1

CAPÍTULO 1. CARACTERIZACIÓN DE LAS HERRAMIENTAS DE HARDWARE Y SOFTWARE PARA LA ELABORACIÓN DE APLICACIONES 4

1.1 Antecedentes y prestaciones del microcontrolador PicoBlaze 4

1.2 Características de las FPGAs de Xilinx 6

1.2.1 Características principales del Kit de desarrollo Nexys2 8

1.3 Características de los softwares Matlab/Simulink, Xilinx ISE y System Generator 13

1.4 Conclusiones del capítulo 14

CAPÍTULO 2. ARQUITECTURA DEL MICROCONTROLADOR PICOBLAZE. FUNDAMENTO TEÓRICO DE LAS APLICACIONES DESARROLLADAS 15

2.1 Características de la arquitectura del microcontrolador PicoBlaze 15

2.2 Bloques funcionales de PicoBlaze 16

2.3 PicoBlaze en el ambiente de System Generator 17

2.3.1 Ensamblado del código KCPSM3 en Matlab 19
2.4 Análisis teórico de las aplicaciones confeccionadas 20
2.4.1 Función multiplicación con microcontroladores 20
2.4.2 Modulación BPSK 21
2.4.3 Comunicación Serie 22
2.5 Conclusiones del capítulo 23
CAPÍTULO 3. IMPLEMENTACIÓN Y COMPROBACIÓN DE LAS APLICACIONES DESARROLLADAS 24
3.1 Configuración de la tarjeta Nexys2 24
3.2 Simulación e implementación de la función multiplicación 26
3.3 Simulación e implementación del modulador BPSK 30
3.4 Simulación e implementación de una comunicación serie 33
3.5 Conclusiones del capítulo 36
CONCLUSIONES Y RECOMENDACIONES 37
Conclusiones 37
Recomendaciones 38
REFERENCIAS BIBLIOGRÁFICAS 39
ANEXOS 43

INTRODUCCIÓN

Las FPGAs (*Field Programmable Gate Array*) o arreglo de puertas programables por campo son circuitos integrados de propósito general, que tienen como principal ventaja la flexibilidad, pues no son circuitos cerrados, es decir, se pueden modificar, cambiando fácilmente su utilidad y adaptándose de esa manera a los continuos cambios que se producen. Esto puede traducirse en una reducción de costos de implementación.

Las FPGAs fueron inventadas en el año 1984 por Ross Freeman y Bernard Vonderschmitt, co-fundadores de Xilinx, y surgen como una evolución de los dispositivos lógicos programables CPLDs (*Complex Programmable Logic Device*).

Estos dispositivos representan una solución tentativa en cuanto a rendimiento, prestaciones y consumo de recursos se refiere, permitiendo realizar operaciones matemáticas a gran velocidad, facilidad de diseño y reprogramación de los circuitos, en comparación con otros dispositivos como los DSPs (*Digital Signal Processing*), y las ASIC (*Application-Specifc Integrated Circuit*).

Estas ventajas, unidas a las continuas mejoras que introducen los fabricantes en las FPGAs, hacen que cada vez sean más usadas y que su tecnología esté ganando terreno y siendo impulsada en el mundo.

Actualmente es común encontrar microcontroladores empotrados en las FPGAs de Xilinx como por ejemplo PicoBlaze, con bajo costo de fabricación y optimizados para realizar tareas específicas, a diferencia de los de propósito general.

Los microcontroladores empotrados pueden ser usados en una variedad de aplicaciones como sistemas para teléfonos celulares, seguridad para autos, equipos para oficina y el hogar, entre otras, dependiendo de sus prestaciones particulares.

PicoBlaze no está concebido como un microcontrolador de altas prestaciones, siendo compacto y flexible. Es por ello que puede ser usado para el procesamiento de datos simples y control, particularmente en aplicaciones de rutina (*house-keeping*) que no sean críticas en cuanto al tiempo de ejecución y en operaciones de entrada/salida. Puede ser fácilmente integrado en un sistema más complejo agregando nuevas dimensiones en la flexibilidad de un diseño basado en FPGAs.

Tradicionalmente para el diseño en las FPGAs usando PicoBlaze, y de manera general, se recurría a lenguajes de descripción hardware (HDL), tales como VHDL y Verilog. No obstante en los últimos años han aparecido herramientas de modelación como System Generator que brindan a los desarrolladores un mayor nivel de abstracción y una mayor interacción. Generalmente estas herramientas son más fáciles de aprender pues se separan más del nivel de hardware con el objetivo de facilitar el diseño de los sistemas.

La Facultad de Ingeniería Eléctrica de la Universidad Central "Marta Abreu" de Las Villas posee el Kit Nexys2 de Digilent el cual posee una FPGA Spartan-3E de Xilinx. En ella se encuentra empotrado el microcontrolador PicoBlaze. Además posee todas las herramientas de software necesarias para el desarrollo de aplicaciones con este microcontrolador, sin embargo solo se han realizado trabajos previos enfocados en lenguajes de descripción de hardware, y no utilizando las prestaciones que brinda PicoBlaze en System Generator, lo que conlleva a que no se hayan desarrollado aplicaciones utilizando herramientas de modelación; de ahí la importancia que tiene este trabajo, pues brindará un análisis teórico y práctico sobre System Generator enfocado en PicoBlaze, el cual servirá como base para las futuras investigaciones que se realicen sobre este tema en la Facultad.

A partir de lo expuesto anteriormente se plantea el siguiente problema de investigación: ¿Cómo elaborar aplicaciones utilizando PicoBlaze en System Generator para implementarlas en FPGAs?

Por tanto, el objetivo general de este trabajo de diploma es desarrollar aplicaciones usando el microcontrolador PicoBlaze en el software System Generator para su implementación en FPGAs de Xilinx. Con el fin de darle cumplimiento al objetivo general se han trazado los siguientes objetivos específicos:

- Caracterizar el microcontrolador PicoBlaze y las herramientas Matlab/Simulink, Xilinx ISE y System Generator, así como las FPGAs de Xilinx.
- Describir la arquitectura del microcontrolador PicoBlaze.
- Determinar las prestaciones que brinda el microcontrolador PicoBlaze en el entorno de System Generator para la elaboración de aplicaciones en FPGAs.
- Elaborar aplicaciones con PicoBlaze en System Generator para implementarlas en las FPGAs.
- Comprobar el funcionamiento de las aplicaciones desarrolladas en una FPGA.

A partir de los objetivos específicos surgen las siguientes interrogantes científicas:

- ¿Cómo interrelacionar los software Xilinx ISE, Matlab/Simulink y System Generator para la implementación de aplicaciones en FPGAs?
- ¿Cuáles son las prestaciones que brinda el microprocesador PicoBlaze en System Generator para la elaboración de aplicaciones?
- ¿Qué aplicaciones elaborar utilizando PicoBlaze en System Generator para implementarlas en las FPGAs?
- ¿Cómo comprobar el funcionamiento de las aplicaciones desarrolladas con PicoBlaze en System Generator en una FPGA?

Para lograr satisfacer cada uno de los objetivos planteados el trabajo se ha dividido en Introducción, tres capítulos, Conclusiones y Recomendaciones. El primer capítulo aborda las principales ventajas de las herramientas usadas para la elaboración de las aplicaciones con el microcontrolador PicoBlaze. En el segundo capítulo se analiza la arquitectura del microcontrolador PicoBlaze y se revisan los fundamentos teóricos referentes al funcionamiento de las aplicaciones desarrolladas. Por último, en el tercer capítulo, se explica el funcionamiento de las tres aplicaciones desarrolladas desde el ambiente de simulación y se exponen los resultados prácticos obtenidos con el montaje real de los diseños.

CAPÍTULO 1. CARACTERIZACIÓN DE LAS HERRAMIENTAS DE HARDWARE Y SOFTWARE PARA LA ELABORACIÓN DE APLICACIONES

En este capítulo se abordan las principales ventajas de los sistemas empotrados, específicamente del microcontrolador PicoBlaze, presentándose ejemplos de algunas aplicaciones. También se caracterizan las herramientas de hardware y software para el desarrollo de aplicaciones usando PicoBlaze.

1.1 Antecedentes y prestaciones del microcontrolador PicoBlaze

Un sistema empotrado es un sistema informático (hardware y software) de tiempo real integrado en un sistema o dispositivo mayor, en el que realiza funciones de monitoreo, control o procesamiento. Una de sus ventajas es su bajo costo, puesto que es posible fabricarlos por miles o por millones de unidades. Los sistemas empotrados suelen ser pequeños, de bajo consumo de energía, usan un procesador y una memoria para almacenar el programa para el cual fueron diseñados. Ejecutan un único programa repetidamente y reaccionan continuamente a los cambios en el ambiente o sistema en que están inmersos (Huerta, 2010).

Según Pablo Huerta (Huerta, 2010), el aumento progresivo de las necesidades computacionales de los sistemas empotrados trae como consecuencia que los procesadores de propósito general no cumplan con los requerimientos necesarios, por lo cual se buscan soluciones alternativas. Una posible solución es mejorar el rendimiento del procesador que se está utilizando, introduciendo mejoras estructurales para aumentar la frecuencia del reloj. Esta solución presenta como inconveniente el aumento del consumo de energía.

Otra solución consiste en utilizar múltiples procesadores en un solo chip (MPSoC, *Multiprocesor System on Chip*), bien de propósito general o específico (Peña, 2008).

Las FPGAs ofrecen gran cantidad de recursos, millones de puertas lógicas, bloques de memoria, lo cual las hace una solución viable para la implementación de múltiples procesadores. Por ejemplo, la compañía Xilinx ofrece FPGAs de las familias Virtex-II Pro, Virtex-4, Spartan 3 y Spartan 3E. En ellas se implementan procesadores en hardware o HCP (*Hard-Core Processors*) y procesadores desarrollados en software o SCP (*Soft-Core Processors*). Estos últimos se pueden implementar utilizando recursos lógicos del FPGA. Xilinx distribuye dos procesadores SCP: PicoBlaze y MicroBlaze (Huerta, 2010).

El rendimiento de los SCP es menor que el que pueden ofrecer los HCP, pero tienen la ventaja de que su número no está fijado de antemano, es decir, se pueden utilizar todos los que sean necesarios y sólo se está limitado por la capacidad de la FPGA. Otra ventaja es la facilidad de configuración, lo que permite crear sistemas heterogéneos a pesar de utilizar procesadores de la misma familia (Huerta, 2010).

PicoBlaze es un microcontrolador SCP, compacto y de distribución libre, optimizado para las familias de FPGA Spartan-3, Virtex-II y Virtex-II Pro. Útil para control y procesamiento de datos simples. Ocupa solo 91 *Slices* de una FPGA Spartan 3E. Este microcontrolador es flexible y no requiere recursos externos. Su funcionalidad básica puede ser extendida conectándole puertos de entrada y salida u otros recursos, con lógica de la FPGA (Huerta, 2010).

Existen varios proyectos elaborados utilizando PicoBlaze y que se han implementado en tarjetas como la Spartan 3 ó 3E. A continuación se mencionan algunos de ellos.

En la Universidad de Extremadura, en España, se diseña por software un contador ascendente-descendente utilizando el microcontrolador empotrado PicoBlaze. A partir de su repertorio de instrucciones se crea un código ensamblador que programa un contador (Gómez, 2011).

Ken Chapman, ingeniero de desarrollo de la empresa Xilinx y creador de PicoBlaze, ha desarrollado los siguientes proyectos con este microcontrolador, implementándolos en la tarjeta de desarrollo Spartan 3E Starter Board, manufacturada por Digilent:

- Reloj capaz de visualizar en un display horas, minutos y segundos y que permite programar una alarma. El diseño contiene un módulo UART para establecer comunicación serial y se utiliza para observar y/o programar el tiempo y la alarma a través de comandos simples y mensajes; para ello se utiliza el HyperTerminal de Windows (Chapman, 2003a).
- Diseño de una aplicación que proporciona una introducción general a los puertos de entrada analógicos y resalta las funciones básicas del convertidor analógico-digital LTC1407A-1 y del amplificador programable LTC6912-1. Los resultados son mostrados en el display LCD (Chapman, 2006a).
- Proyecto para implementar las funciones básicas del convertidor digital-analógico LTC2624. El dispositivo es controlado por PicoBlaze para proveer una introducción general a los convertidores analógico-digital (Chapman, 2006b).
- Contador de frecuencia, capaz de medir frecuencias de hasta 200MHz. Puede ser utilizado para realizar pruebas de equipos o simplemente como práctica introductoria a los osciladores que dispone la tarjeta, los resultados son mostrados en el display LCD (Chapman, 2006c).
- Módulo para mostrar un mensaje rotatorio en el display LCD y que permite controlar los LEDs utilizando switches y push buttons (Chapman, 2006d).
- Implementación por software de la modulación de ancho de pulso. Permite manejar 12 canales, ocho canales controlan la intensidad de los LEDs, los cuatro canales restantes pueden ser enviados al conector J4 para ser observadas a través del osciloscopio. El PWM implementado tiene una resolución de 8 bits y el ciclo de trabajo puede ser determinado a través del HyperTerminal (Chapman, 2006e).

1.2 Características de las FPGAs de Xilinx

Desde su fundación en el año 1984 por los ingenieros en semiconductores Ross Freeman y Bernard Vonderschmitt, la corporación Xilinx ha sido puntera en el desarrollo de FPGAs.

Las líneas de productos de Xilinx incluyen las series Virtex, Kintex, Artix y Spartan, cada una incluye configuraciones y modelos optimizados para diferentes aplicaciones.

Cada chip FPGA está compuesto de un número finito de recursos predefinidos con interconexiones programables para implementar un circuito digital reconfigurable. Poseen una estructura interna formada por: matriz de bloques lógicos configurables CLB (*Configurable Logic Block*) que se comunican entre sí y con los bloques de Entrada/Salida (*Input/Output Block*, I/OB) a través de canales de ruteo o interconexión horizontales y verticales. Estos bloques pueden observarse en la Figura 1.1.

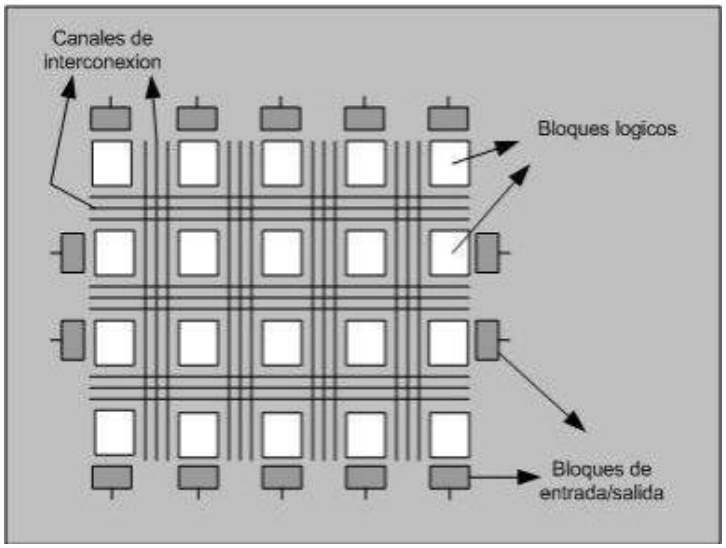

Figura 1.1. Diagrama general de una FPGA de Xilinx (Güichal, 2005).

Las FPGAs Spartan 3E de Xilinx poseen además nuevas características que mejoran el rendimiento del sistema y reducen el costo de configuración. Estas mejoras combinadas con los avances de la tecnología de 90nm dan mayor funcionalidad a las tarjetas. Por su bajo costo son ideales para un amplio rango de consumidores de aplicaciones electrónicas (Arias, 2010).

Dentro de la amplia gama de productos de Xilinx, en este trabajo se profundizará en la serie Spartan 3E, específicamente la FPGA XC3S500E, utilizada en el Kit Nexys2, hardware del que se dispone en la Facultad, y en el cuál está empotrado el microcontrolador PicoBlaze.

1.2.1 Características principales del Kit de desarrollo Nexys2

La tarjeta Nexys2 es una plataforma desarrolladora de circuitos basada en una FPGA Spartan 3E de Xilinx. El puerto USB2 que posee proporciona a la tarjeta una entrada de energía y la interface de programación, por lo que el Kit Nexys2 se puede utilizar con una PC para crear una estación de diseño portátil (Digilent, 2008).

La Figura 1.2 muestra el diagrama en bloques de esta tarjeta.

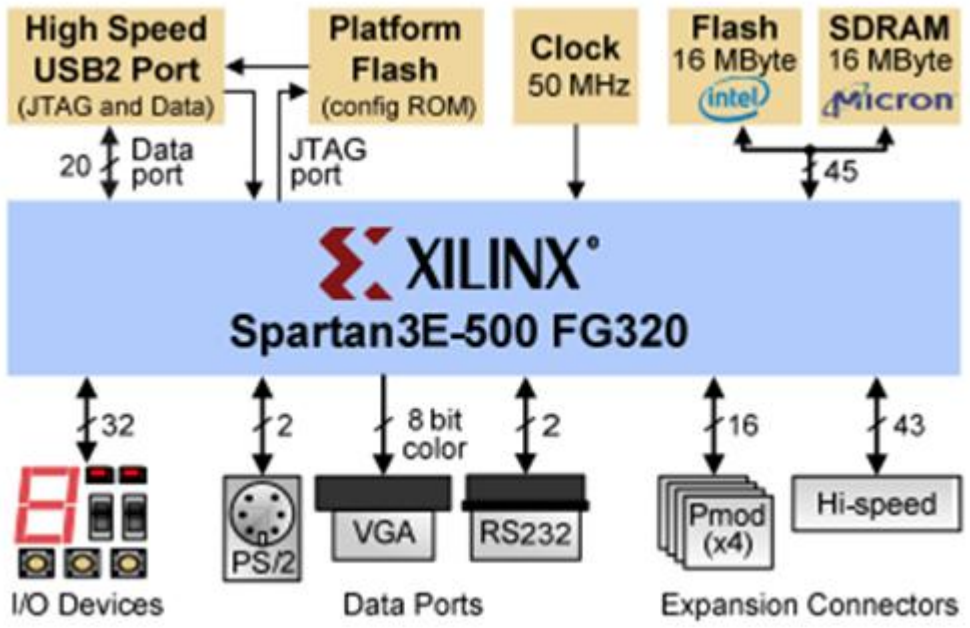

Figura 1.2. Diagrama en bloques de la tarjeta Nexys2 (Digilent, 2008).

A continuación se realiza una descripción de los principales bloques de la tarjeta Nexys2 que son usados en el desarrollo de este trabajo.

- **Relojes**

El tablero Nexys2 incluye un oscilador de 50MHz y un socket para un segundo oscilador como se muestra en la Figura 1.3. Las señales de reloj de los osciladores se conectan con los pines de entrada del reloj global que conduce a los bloques sintetizadores del reloj disponibles en la FPGA. Los sintetizadores del reloj llamados DLLs (*Delay Locked Loops*), proporcionan las capacidades de control del reloj que incluyen la posibilidad de doblar o

cuadruplicar y dividir por cualquier múltiplo entero la frecuencia de entrada, así como definir su fase exacta y retrasar las relaciones entre las señales del reloj.

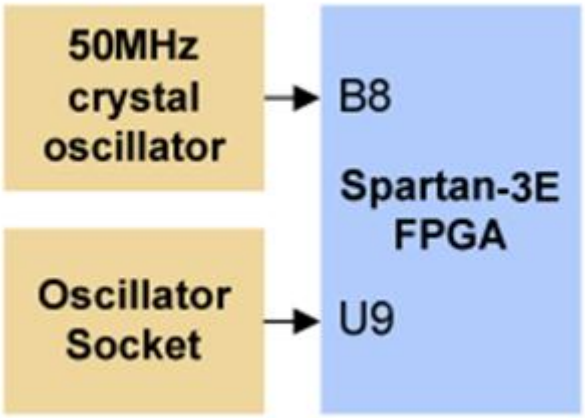

Figura 1.3. Relojes del Kit Nexys2 (Digilent, 2008).

- **I/O de usuario**

La tarjeta Nexys2 incluye varios dispositivos de entrada, salida y puertos de datos, permitiendo que muchos diseños sean implementados sin la necesidad de cualquier otro componente. En la Figura 1.4 se muestran los dispositivos de entrada/salida y su circuito se muestra en la Figura 1.5.

Figura 1.4. Dispositivos I/O (Digilent, 2008).

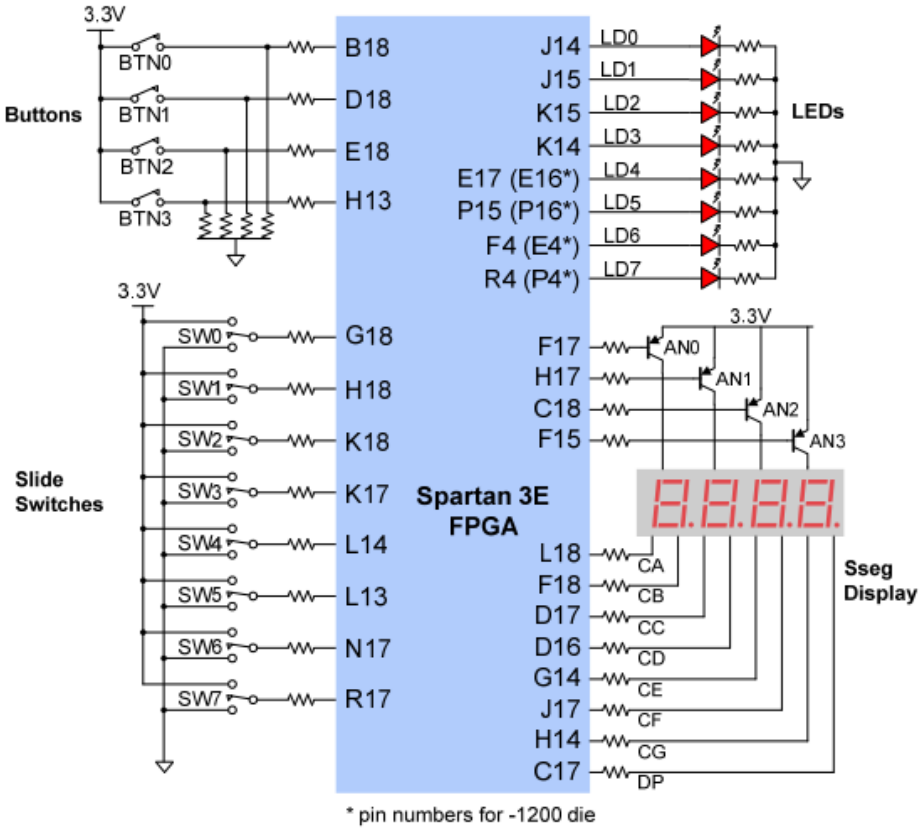

Figura 1.5. Circuito de los dispositivos I/O (Digilent, 2008).

La tarjeta tiene cuatro botones y ocho interruptores deslizantes. Los botones generan normalmente un nivel bajo (0V) y al ser presionados pasan a nivel alto (3.3V). Los interruptores deslizantes generan niveles constantes, ya sea alto o bajo, dependiendo de su posición. Las entradas de botón y de interruptor deslizante utilizan un resistor serie para la protección contra cortocircuitos.

Los ocho LEDs proporcionados tienen sus ánodos conectados a la FPGA mediante resistores de 390-ohmios, por lo que una salida de un '1' lógico provocará que los LEDs se iluminen.

El display siete segmentos que contiene el kit posee cuatro dígitos de nueve LEDs con ánodo común en cada segmento. Las señales del ánodo común están disponibles como cuatro señales de entrada "habilitar dígitos" lo cual permite iluminar cada dígito a la vez. Los cátodos de los segmentos están conectados a siete nodos y permiten representar en cada

dígito hasta 128 patrones además del punto (.) llamado DP. En la Figura 1.6 se muestra el circuito de este display.

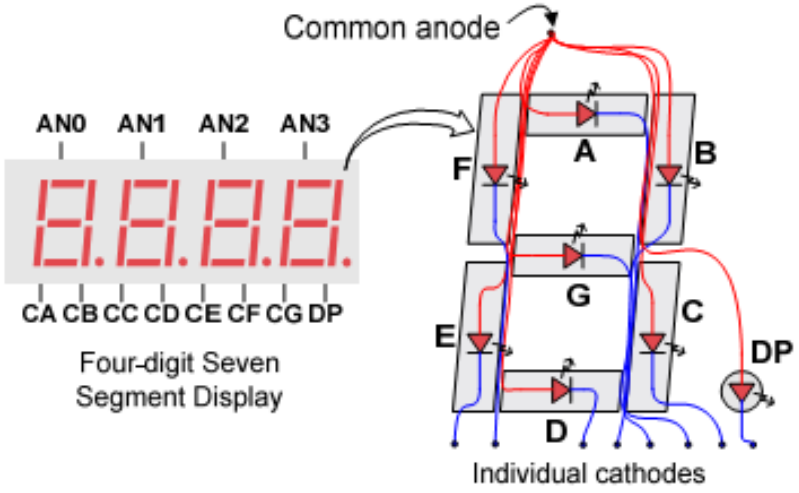

Figura 1.6. Circuito del display siete segmentos (Digilent, 2008).

Para que los dígitos parezcan iluminado continuamente, cada uno de ellos tienen que permanecer encendido un tiempo que varía de 1 a 16ms.

- **Puerto Serie**

La tarjeta Nexys2 contiene un puerto serie de dos hilos basado en un convertidor de voltaje ST Microelectronics ST3232, el cual convierte los niveles de señal usados por las comunicaciones RS-232 (-12V a -3V para '1' lógico y 12V a 3V para '0' lógico) a las señales de 3.3V usadas por la FPGA. Puesto que solamente dos señales están conectadas (RxD y TxD) como se muestra en la Figura 1.7, un controlador FPGA basado en puerto serie puede utilizar solamente protocolos de software *hand-shaking* (XON/XOFF) para el control de errores.

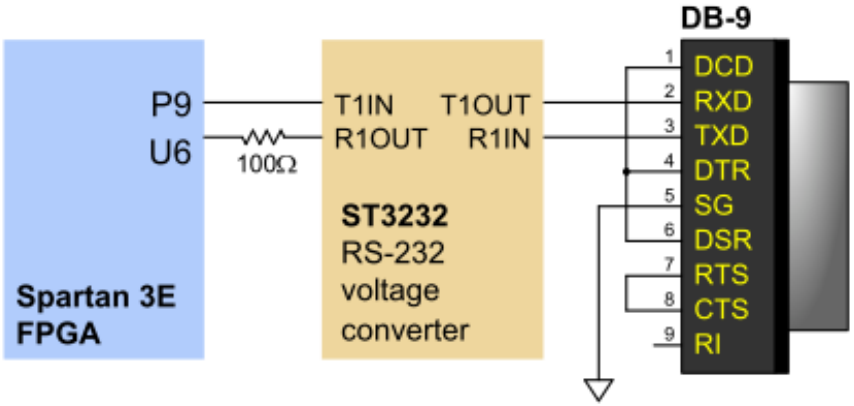

Figura 1.7. Circuito del puerto Serie (Digilent, 2008).

- **Conectores periféricos**

El kit proporciona cuatro conectores de 6 pines dispuestos en dos filas que juntos pueden acomodar hasta ocho módulos periféricos Pmod (*Peripheral Module*). Estos cuatro conectores, de 12 pines en total cada uno, tienen ocho señales de datos, dos pines de tierra, y dos de Vdd respectivamente. Todas las señales de los datos incluyen los resistores de protección contra cortocircuitos y diodos de protección ESD. En la Figura 1.8 puede observarse el circuito de estos conectores. En el Anexo 1.1 se muestra una tabla con la asignación de pines de estos conectores.

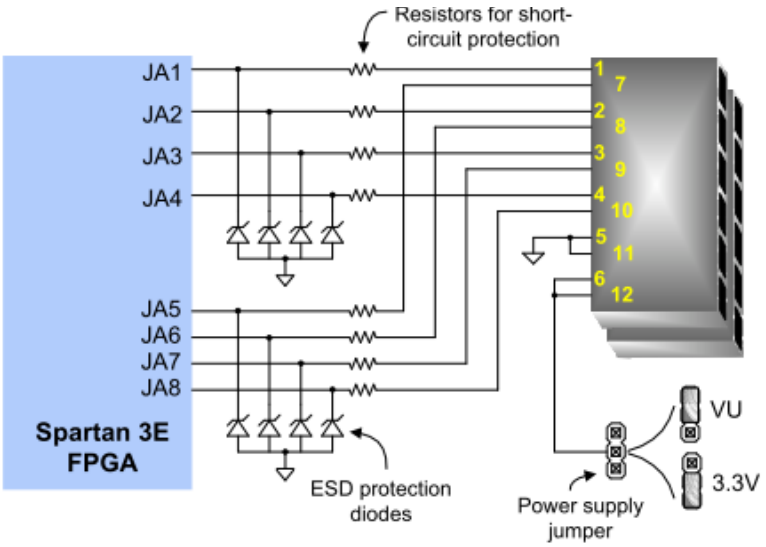

Figura 1.8. Circuito de los conectores Pmod (Digilent, 2008).

1.3 Características de los softwares Matlab/Simulink, Xilinx ISE y System Generator

Simulink es un entorno de programación visual, que forma parte del software Matlab y que permite construir y simular modelos de sistemas físicos y de control mediante diagramas de bloques. El comportamiento de dichos sistemas se define mediante funciones de transferencia, operaciones matemáticas, elementos de Matlab y señales predefinidas (MathWorks, 2009).

Simulink proporciona una interface gráfica de usuario GUI (*Graphical User Interface*) para realizar modelos como diagramas en bloques, permitiendo dibujarlos de la misma forma que se realizarían con lápiz y papel. Además incluye una librería de bloques de fuentes, componentes lineales y no lineales, conectores; aunque también permite crear bloques definidos por los usuarios. El GUI simplifica el proceso de modelado, eliminando la necesidad de realizar complicadas tareas como por ejemplo la elaboración de ecuaciones diferenciales en lenguaje de programación (MathWorks, 2009).

Por otra parte el entorno de diseño ISE (*Integrated Software Environment*) de Xilinx consiste en una herramienta de software que permite realizar un diseño digital basado en lógica programable. Permite recorrer fácilmente las diferentes fases del proceso de desarrollo de un circuito lógico, desde el diseño hasta la implementación sobre una arquitectura reconfigurable. Cada una de las etapas del flujo de diseño puede realizarse dentro del entorno integrado (Castelló y Valido, 2010).

ISE combina diferentes técnicas de diseño para facilitar la labor de descripción del diseño, mediante un conjunto de herramientas que permiten el diseño de circuitos digitales como esquemas lógicos, máquinas de estado o utilizando lenguajes de descripción de hardware como VHDL o Verilog (Rosado y Bataller, 2003).

Es posible llamar de manera automática y desde su propio entorno de trabajo a las diferentes utilidades y herramientas suministradas dependiendo de la etapa de diseño en la que se encuentre el proyecto, lo cual hace el trabajo más sencillo. El entorno posee un aspecto similar al de los entornos de programación actuales como Visual Basic o Visual C,

con diversas ventanas para visualizar tareas específicas sobre cada una de ellas (Arias, 2010).

System Generator se integra a Simulink como un toolbox al instalarse en la computadora Xilinx ISE. Esta herramienta de modelación facilita el diseño del hardware en una FPGA, ampliando de esta forma las posibilidades de Simulink. Además permite un alto nivel de abstracción que puede ser compilado automáticamente en la FPGA.

Las librerías de propiedad intelectual (IP) que posee System Generator permiten la implementación eficiente de funciones. En este sentido, no es necesario tener siempre conocimiento de los detalles de lo recursos internos de la FPGA, y aun así poder realizar un diseño eficiente incluyendo bloques como FFTs (*Fast Fourier Transform*), etc.

1.4 Conclusiones del capítulo

Tanto System Generator como Xilinx ISE permiten realizar diseños para ser implementados en una FPGA, no obstante, el primero brinda la posibilidad de generar el código HDL, y el fichero de configuración para la programación de la FPGA y definir los puertos a usar en la tarjeta; todo desde una interfaz amigable y sencilla para el usuario.

CAPÍTULO 2. ARQUITECTURA DEL MICROCONTROLADOR PICOBLAZE. FUNDAMENTO TEÓRICO DE LAS APLICACIONES DESARROLLADAS

En este capítulo se abordan las principales características de la arquitectura del microcontrolador PicoBlaze, en las cuales se basan las aplicaciones que se desarrollan en esta investigación. Además se describen los pasos fundamentales para iniciar el trabajo con este microcontrolador en el entorno de System Generator y se presentan los fundamentos teóricos de las aplicaciones confeccionadas.

2.1 Características de la arquitectura del microcontrolador PicoBlaze

El núcleo de PicoBlaze se encuentra empotrado en la FPGA, por lo que no requiere recursos externos, si no que hace uso de las abundantes y flexibles conexiones de entrada y salida que provee la tarjeta, todo a un bajo costo. Además, por ser un código fuente VHDL sintetizable, el núcleo puede ser migrado a futuras arquitecturas FPGAs, lo cual le permite no quedar obsoleto.

En la Figura 2.1 se puede ver la organización interna del núcleo PicoBlaze, el cual se distingue por las siguientes características fundamentales:

- Ancho de datos: 8 bit.
- 16 registros de propósito general.
- 1024 instrucciones de 18 bit que son cargadas durante la programación en la memoria ROM externa.
- ALU con banderas CARRY y ZERO.
- 64 bytes de memoria RAM interna.

- 256 puertos de entrada y 256 de salida.
- 31 *stacks call/return* de localización automática.
- Dos ciclos de reloj por instrucción. Soporta hasta 200 MHz o 100 MIPS.
- Rápida respuesta en la atención a las interrupciones, 5 ciclos de reloj en el peor de los casos.

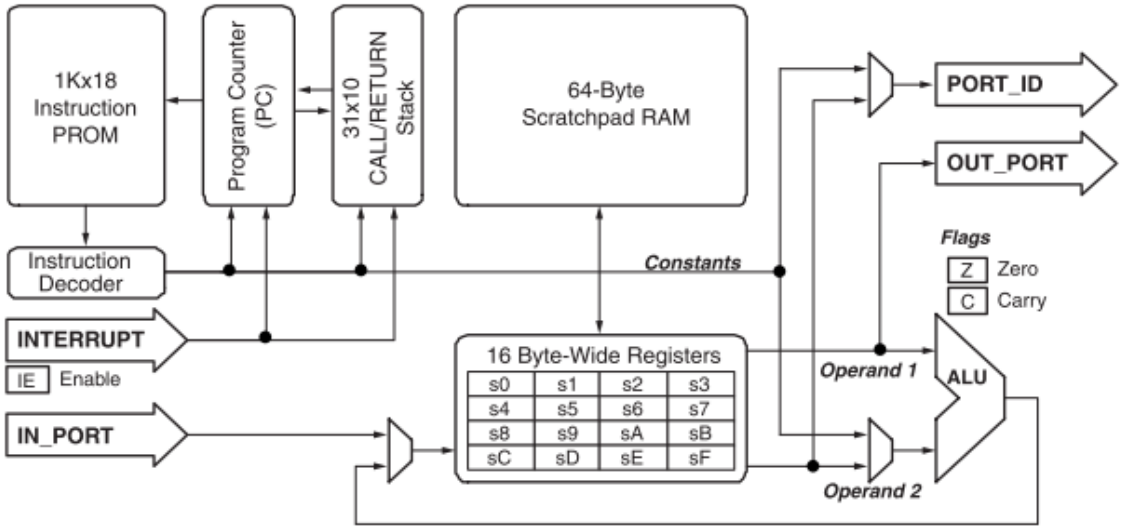

Figura 2.1. Diagrama en bloques de PicoBlaze (Chapman, 2005).

2.2 Bloques funcionales de PicoBlaze

El microcontrolador PicoBlaze posee 16 registros de propósitos generales (s0-sF), que pueden ser renombrados en el lenguaje ensamblador para una mejor claridad. Todas las operaciones de los registros son completamente flexibles y no se reservan registros para tareas especiales ni se les da prioridad sobre otros registros. No existen acumuladores, por lo tanto ningún registro puede ser adoptado para ser usado como acumulador. Además, él es capaz de ejecutar 1024 instrucciones de 18 bit, las cuales requieren 2 ciclos de reloj para su ejecución (Chapman, 2005).

El set de instrucciones de PicoBlaze se encuentra en el Anexo II.

PicoBlaze posee una unidad aritmética lógica (ALU) de 8 bits. La cual puede ejecutar las operaciones siguientes:

- Operaciones aritméticas básicas, como adición y substracción.
- Operaciones lógicas tales como AND, OR, XOR entre dos registros.
- Comparación aritmética.
- Rotación y corrimiento de bits.

Las operaciones aritméticas y lógicas se realizan entre registros, o entre un registro y una constante y afectan las banderas ZERO y CARRY. La bandera ZERO indica que el resultado de la última operación es cero y la bandera CARRY puede indicar varias condiciones, dependiendo de la última instrucción realizada. La bandera INTERRUPT_ENABLE habilita la interrupción (Chapman, 2005).

PicoBlaze posee una memoria interna de datos de 64 bytes de propósito general que puede ser direccionada de forma directa o indirecta usando las instrucciones STORE y FETCH. También puede manejar hasta 256 puertos de entrada y 256 puertos de salida, permitiendo conectarse a dispositivos periféricos o lógica dentro de la FPGA (Chapman, 2005).

El contador de programa (PC) contiene la dirección de la instrucción que va a ser ejecutada y se incrementa automáticamente para apuntar a la dirección de memoria de la siguiente instrucción a ejecutarse. El núcleo de PicoBlaze contiene una pila que almacena hasta 31 posiciones de memoria, habilitando secuencias de CALL de hasta 31 niveles de profundidad. Se debe considerar que la pila reserva un nivel cuando se usa una interrupción (Chapman, 2005).

La única fuente de interrupción que posee este microcontrolador es denominada INTERRUPT. Para poder manejar más de una interrupción es necesario acondicionar el hardware externo. Además, contiene un circuito de control de reset interno para asegurar su correcto inicio. El reset está sincronizado con la señal de reloj y dura cuatro ciclos de la misma. Se ocupa de formar una señal de control que reinicie el sistema (Peña, 2008).

2.3 PicoBlaze en el ambiente de System Generator

El bloque "*PicoBlaze Microcontroller*" disponible en System Generator implementa un microcontrolador de 8 bit con las características descritas anteriormente; pero este bloque por sí solo es inútil para el desarrollo de alguna aplicación. Unido a este bloque se

encuentra el bloque "*ROM*", en el cual es posible almacenar hasta 1024 instrucciones. En la Figura 2.2 puede observarse la interconexión de estos bloques.

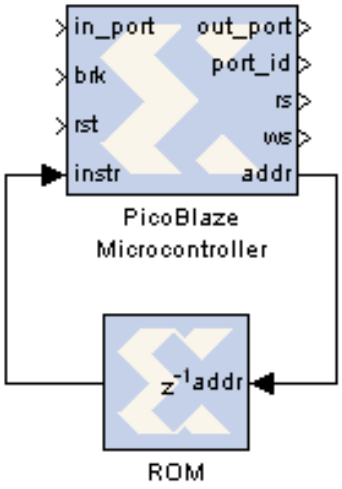

Figura 2.2. Microcontrolador PicoBlaze y memoria ROM (Xilinx, 2011).

El bloque "*ROM*" tiene solo dos terminales, entrada y salida, permitiendo obtener una instrucción específica en su puerto de salida al ser indicada por el microcontrolador una posición de memoria como una palabra digital de 10 bit en su puerto de entrada.

Por otra parte, el bloque "*PicoBlaze Microcontroller*" tiene nueve terminales. En la Tabla 2.1 se describe la función de cada uno de ellos.

Tabla 2.1. Función de las señales de PicoBlaze (Xilinx, 2011).

Señal	Dirección	Descripción
in_port[7:0]	Entrada	Puerto de entrada de datos
brk	Entrada	Interrupción
rst	Entrada	Reset
instr[17:0]	Entrada	Instrucciones
out_port[7:0]	Salida	Puerto de salida de datos
port_id[7:0]	Salida	Dirección del puerto
rs	Salida	Operación de lectura
ws	Salida	Operación de escritura
addr[9:0]	Salida	Dirección de la siguiente instrucción

2.3.1 Ensamblado del código KCPSM3 en Matlab

Para poder cargar en la memoria ROM el programa a ejecutar por el microcontrolador, es necesario ensamblar previamente el código fuente, llamado también código KCPSM3. Existen varias formas de realizar este proceso, pero este trabajo describe solo una, por ser simple y realizable desde Matlab, lo que trae como ventaja que no sea necesario usar un software adicional. Para lograr lo antes expuesto se deben seguir los siguientes pasos:

1. Crear el fichero que contendrá el código fuente de la aplicación a desarrollar. Para esto se debe crear un documento de texto con el Notepad de Windows, asignándole el nombre del proyecto. Seguidamente es necesario cambiar la extensión del fichero creado anteriormente a ".psm", quedando así el fichero listo para ser escrito en él el código fuente de la aplicación, proceso que puede ser realizado desde el propio Notepad o desde Matlab.
2. Ejecutar el Matlab y en él la instrucción ***cd c:\assembler***, donde *c:assembler* se debe sustituir por la ruta donde se encuentra el proyecto. Posteriormente se ejecuta la instrucción ***xlpb_as -p 'xxxx.psm',*** sustituyendo la palabra "xxxx" por el nombre del fichero creado en el paso 1. Si todo se hizo correctamente se obtiene el mensaje *"Program sucessfully assembled", g*enerándose un fichero con extensión ".m" en la carpeta del proyecto. La causa más común para no obtener este mensaje es algún error de sintaxis en el código fuente.
3. Ejecutar System Generator para configuran los parámetros del bloque de la memoria ROM y agregarle el código fuente. La Figura 2.3 muestra la configuración de la memoria.

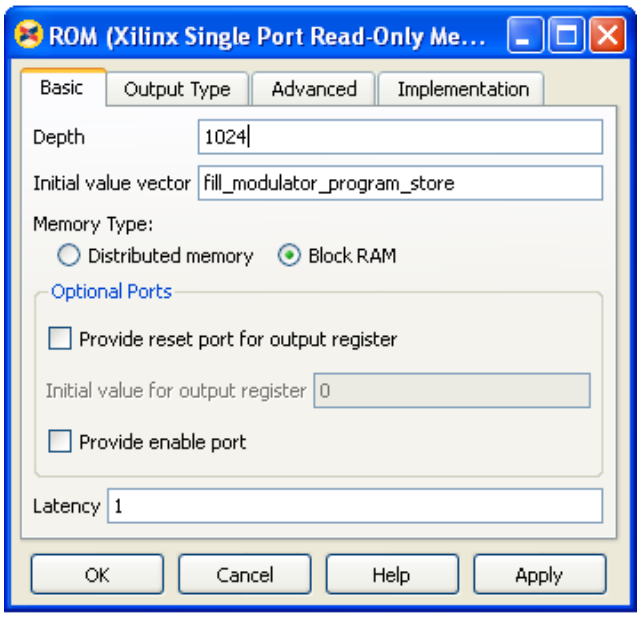

Figura 2.3. Configuración de la memoria ROM.

2.4 Análisis teórico de las aplicaciones confeccionadas

A continuación se presentan las principales consideraciones teóricas de las tres aplicaciones confeccionadas en este trabajo.

2.4.1 Función multiplicación con microcontroladores

Un algoritmo de multiplicación es un método para multiplicar dos números. Dependiendo del tamaño y naturaleza de los números, existen diferentes tipos de algoritmos. Usualmente microcontroladores como PicoBlaze no implementan funciones de multiplicación y división en sus unidades aritméticas. Esto se debe, en parte, a que pueden ser realizadas por el programador con las restantes operaciones que trae implementada la ALU.

La operación de multiplicación puede estudiarse como la suma respectiva del multiplicando las veces que indique el multiplicador. Por ejemplo la operación 7*3, en sistema binario puede realizarse de la siguiente forma: 0111*0011 = 0111+0111+0111 = 010101. Es decir, sumar tres veces el número siete.

Si los factores son de N cifras, el producto se expresa con 2*N dígitos. De manera que para N = 3 en sistema decimal se tiene que con operandos sin signo, el mayor factor es 999, por tanto 999*999 = 998001. Como se aprecia el resultado tiene seis cifras.

Debido a que PicoBlaze es un microcontrolador de 8 bit, si se desea realizar la ecuación A*A + B*B, donde A y B son números de cuatro bit, los resultados de las multiplicaciones de A*A y B*B se expresaran en 8 bit respectivamente, siendo el mayor valor para el resultado 225. Sin embargo, al realizar la ecuación completa para este caso en binario 1111*1111+1111*1111 = 111000010. El resultado decimal es 450, valor que supera el límite posible a representar con 8 bit, por tanto al realizar una operación, cuando el resultado exceda el 255, en el registro quedara almacenado el resultado de la operación restado 255.

2.4.2 Modulación BPSK

Una señal se puede transmitir por un canal que permita la propagación de ese tipo de señales. Sin embargo, no basta con esta correspondencia de la naturaleza de la señal y el canal, sino que además la señal debe tener determinados parámetros adecuados a las características del canal para una transmisión óptima en cuanto a velocidad y razón de errores. Por ejemplo: un canal transmite bien las señales a una determinada frecuencia y mal a otras. La modulación intenta conseguir esta relación entre señal y canal, de modo que en las transmisiones se usen aquellas frecuencias en las que el canal proporciona la mejor respuesta. Existen diferentes tipos de modulación, entre ellas la modulación BPSK (Carrión, 2011).

La modulación por desplazamiento de fase o PSK (*Phase Shift Keying*) es una forma de modulación angular que consiste en hacer variar la fase de la portadora entre un número de valores discretos M. Cuando este número de valores es dos, se conoce como modulación BPSK. A mayor número de posibles fases, mayor es la cantidad de información que se puede transmitir utilizando el mismo ancho de banda, pero mayor es también su sensibilidad frente a ruidos e interferencias. Es por ello que BPSK es la más inmune al ruido, teniendo la mínima cantidad de variaciones de fase posibles, siendo la diferencia de fase entre símbolos máxima (180°). Dichos símbolos suelen tener un valor de salto de fase de 0º para el ‘1’ lógico y 180º para el ‘0’ lógico según muestra la Figura 2.4.

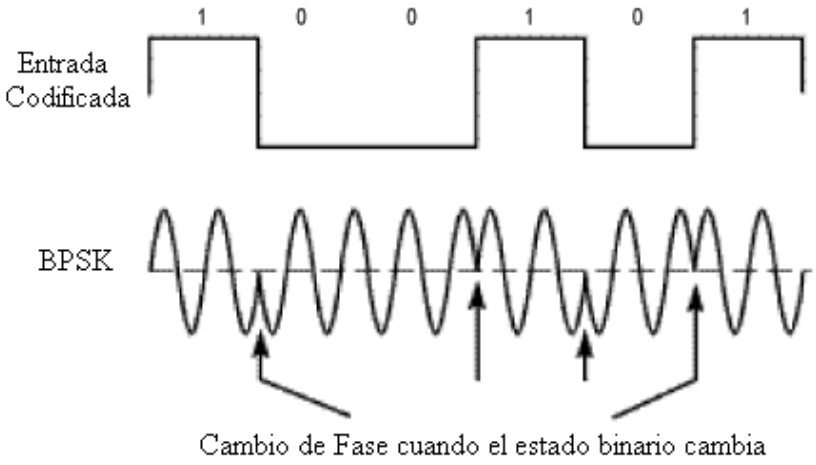

Figura 2.4. Modulación BPSK (Carrión, 2011).

2.4.3 Comunicación Serie

Un puerto serie es una interfaz de comunicaciones entre ordenadores y periféricos en donde la información es transmitida bit a bit enviando un solo bit a la vez (en contraste con el puerto paralelo que envía varios bits a la vez).

Los dos dispositivos conectados a cada extremo de un cable serie son conocidos como DTE (*Data Terminal Equipment*) y DCE (*Data Communication Equipment*). El DCE estaba originalmente concebido como modem, pero posteriormente surgió la idea de usar una computadora como DCE, de modo que se tendría una conexión DTE-DTE. Este tipo de conexión puede ser realizada únicamente cruzando las líneas 2 y 3 (RXD y TXD) de un cable serie, produciendo lo que se conoce como "*null modem*". Para conexiones DTE-DCE se usa el cable denominado "*straight-through*", en el cual cada terminal en uno de los conectores coincide con su semejante en el otro conector. La tarjeta Nexys2 está configurada como dispositivo DCE, pues se asume que será mayormente conectado a una computadora (Digilent, 2008).

En la Figura 1.7 del capítulo anterior puede verse el circuito del puerto serie de la tarjeta Nexys2.

El puerto serie es conocido comúnmente como UART (*Universal Asynchronous Receiver and Transmitter*) o Receptor y Transmisor Asíncrono Universal. Esto significa que las

señales TXD y RXD no están sincronizadas, por lo que es usada una referencia o base de tiempo tolerable para permitir la transferencia serie de cada byte.

La comunicación se realiza transmitiendo como primer bit el menos significativo de la palabra digital y a una razón de bit (BAUD *rate*) determinada y configurable. Como el transmisor puede iniciar la transmisión de datos en cualquier momento, el receptor necesita una forma de identificar cuando el LSB (*Less Significative Bit*) está siendo enviado. Esto se logra enviando una señal activa de inicio en nivel bajo con la duración de un bit (Chapman, 2003b).

En la Figura 2.5 puede verse la estructura de un byte.

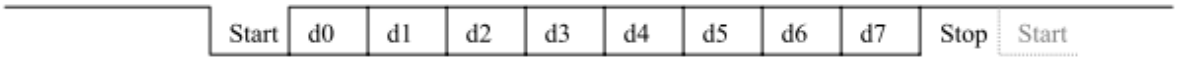

Figura 2.5. Estructura de un byte para el puerto serie (Chapman, 2003b).

El receptor usa el borde de bajada del bit de inicio para activar su circuito de temporización interno, el cual es usado para muestrear los valores que llegan al puerto de entrada justamente a la mitad de cada tiempo de bit, momento en el que el dato debe ser más estable (Figura 2.6).

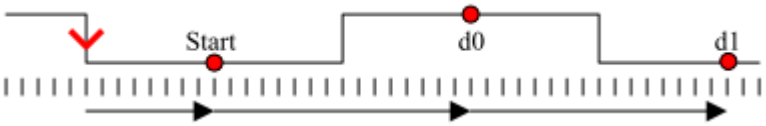

Figura 2.6. Muestreo del puerto serie (Chapman, 2003b).

2.5 Conclusiones del capítulo

Dada las características de la arquitectura del microcontrolador PicoBlaze, es posible realizar diseños de baja y mediana complejidad que no requieran de altas velocidades de ejecución. Este microcontrolador puede ser usado en aplicaciones que mezclen puertos de entrada y salida con procesamiento de alguna información, típico en moduladores, comunicación con la PC, calculadoras, sistemas de alarmas, relojes, etc.

CAPÍTULO 3. IMPLEMENTACIÓN Y COMPROBACIÓN DE LAS APLICACIONES DESARROLLADAS

En este capítulo se aborda la implementación en System Generator de las aplicaciones descritas anteriormente. Se presentan las simulaciones realizadas, las cuales confirman el correcto funcionamiento de las aplicaciones desarrolladas. Además se muestran las aplicaciones corriendo en el Kit Nexys2.

3.1 Configuración de la tarjeta Nexys2

La FPGA en la tarjeta Nexys2 debe ser configurada (o programada) por el usuario antes de que pueda realizar cualquier función. Durante la configuración, un fichero con extensión ".bit " se transfiere a las celdas de memoria dentro de la FPGA para definir las funciones lógicas y las interconexiones del circuito.

Se puede programar de varias maneras: de una PC, usando el puerto USB de la tarjeta o el conector JTAG, o desde una "*Platform Flash ROM*" en la tarjeta (esta última es también programable vía puerto USB). Un jumper determina qué fuente (PC o ROM) se utilizará para cargar la configuración. La Figura 3.1 muestra el circuito de programación de la tarjeta.

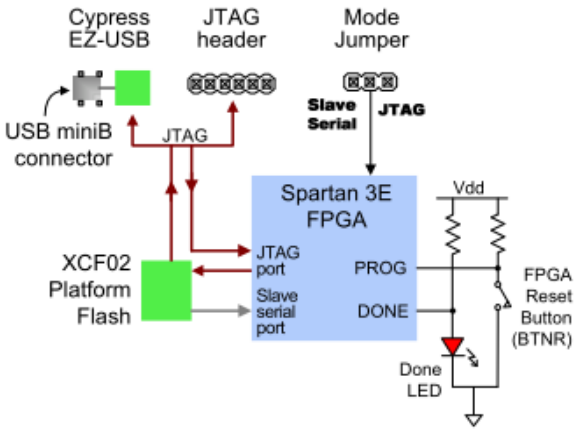

Figura 3.1. Circuito de programación del Kit Nexys2 (Digilent, 2008).

Para la configuración de la FPGA en este trabajo se escogió el primer método por ser sencillo, solo demanda de un cable USB del ordenador a la tarjeta y un software para cargar el fichero. No se utilizó el conector JTAG debido a que no existe en la Facultad. El software usado fue el Digilent Adept, cuyo ambiente se muestra en la Figura 3.2.

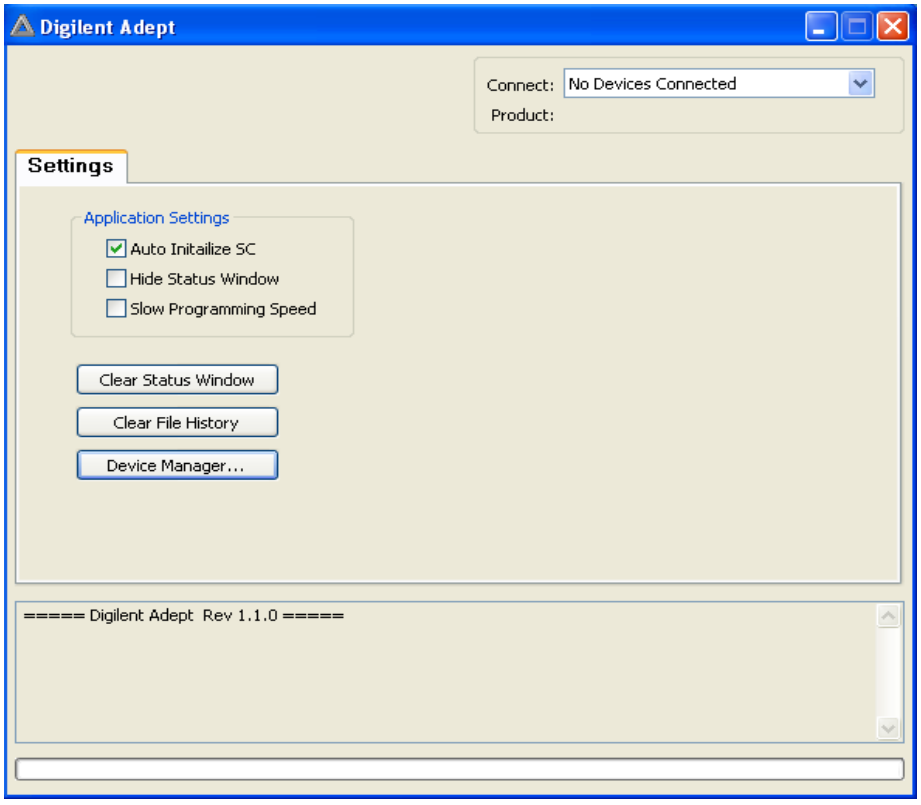

Figura 3.2. Ambiente de trabajo del software "Digilent Adept".

Luego de conectar la FPGA al ordenador, el software la reconoce y permite buscar el fichero de configuración (con extensión ".bit") generado por System Generator. Posteriormente, se hace clic en "*Initialize Chain*", iniciándose el proceso de programación de la FPGA.

3.2 Simulación e implementación de la función multiplicación

Esta aplicación recibe dos números A y B introducidos por el usuario en formato binario, eleva al cuadrado cada uno de ellos y luego los suma. El resultado se muestra en el display siete segmentos que posee la tarjeta Nexys2. Los números se reciben con los ocho interruptores deslizantes.

En la Figura 3.3 se muestra el circuito implementado en System Generator. El "*Gateway In*" llamado **SW_In** es configurado para recibir las señales de los ocho interruptores, las cuales van al microcontrolador y al "*Gateway Out*" **LEDs**. Este último envía las señales a los ocho LEDs que posee la tarjeta con el objetivo de visualizar los números introducidos en sistema binario.

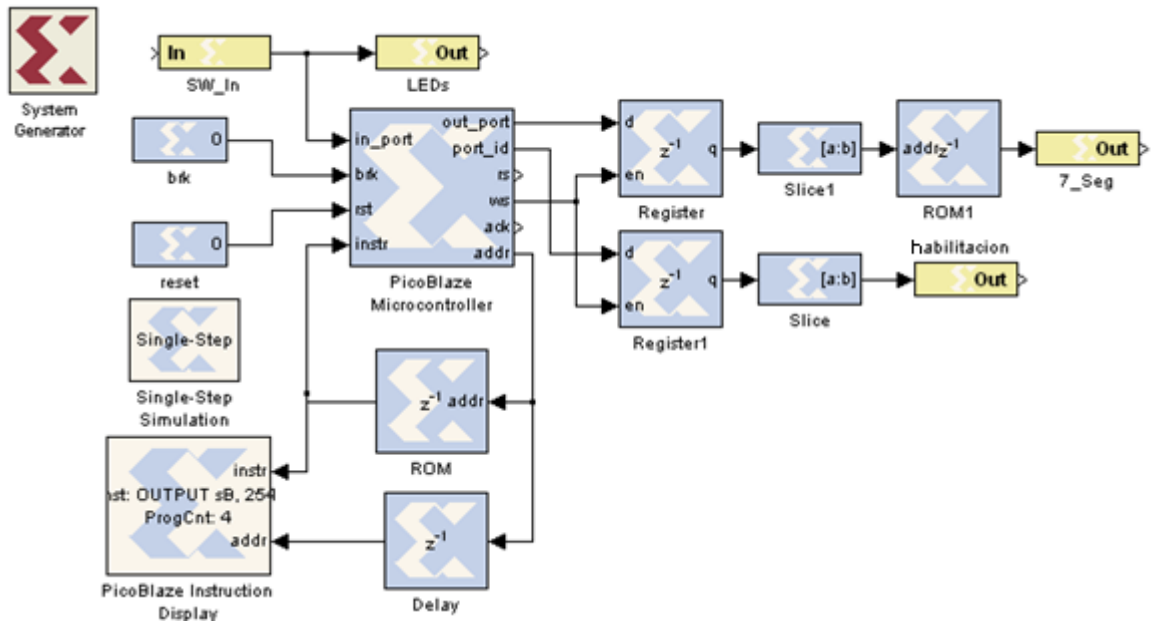

Figura 3.3. Circuito de la función $A^2 + B^2$ en System Generator.

Los registros permiten almacenar la información del puerto de salida y la dirección del puerto usado por el microcontrolador. Con el objetivo de realizar la conversión a BCD

(Binario Codificado Decimal) se usa la memoria **ROM1**, y el "*Gateway Out*" **7_Seg** se configura para enviar cada dígito del resultado en BCD a cada uno de los cuatro displays siete segmentos. Por otra parte, el "*Gateway Out*" llamado **habilitación** permite habilitar cada uno de los cuatro displays independientes y así mostrar cuatro dígitos del resultado de la operación.

El código fuente del programa implementado para esta aplicación ejecuta 5 subrutinas básicas, y se muestra en el Anexo III.

Para realizar la simulación de esta aplicación fue necesario hacer algunas modificaciones al circuito que se presenta en la Figura 3.3 pues System Generator no tiene entre sus bloques las interfaces de entrada/salida que posee la tarjeta como por ejemplo el display siete segmentos.

El circuito que se observa en la Figura 3.4 corresponde a la simulación realizada en System Generator de la aplicación "Función $A^2 + B^2$". En esta figura puede apreciarse como la aplicación recibe un valor constante de '49' (decimal), en binario 00110001. Si se divide el número en dos valores de cuatro bit, se tiene que A = 0011 ('3' en decimal) y B = 0001 ('1' en decimal). El resultado de la operación será entonces '10' en decimal. Este valor puede apreciarse en el display que brinda Simulink y que se usó en la simulación.

Si se observa la Figura 3.5 puede apreciarse el mismo resultado de la simulación, utilizándose el bloque "*WaveScope*" de System Generator. Este bloque permite visualizar, de forma simultánea, varias señales así como la señal del reloj global del sistema. Esta característica posibilita conocer el tiempo que demora el microcontrolador en efectuar la operación (momento en el cual el valor de *"Result"* cambia de '000' a '010') ya que debe ejecutar un número determinado de instrucciones que demandan dos períodos de reloj cada una.

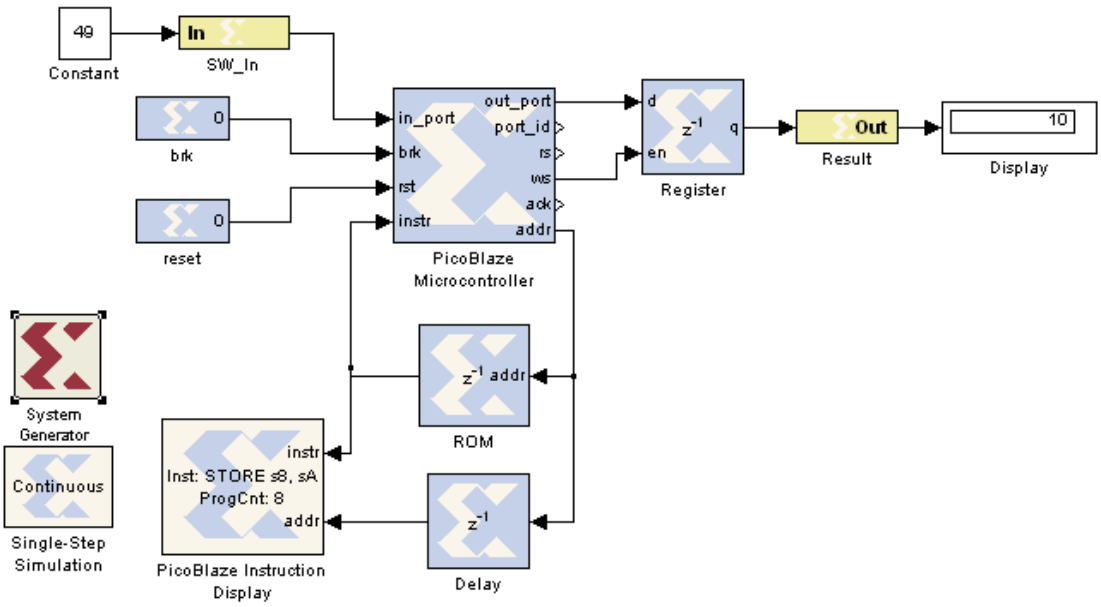

Figura 3.4. Circuito de la función $A^2 + B^2$ modificado para la simulación en System Generator.

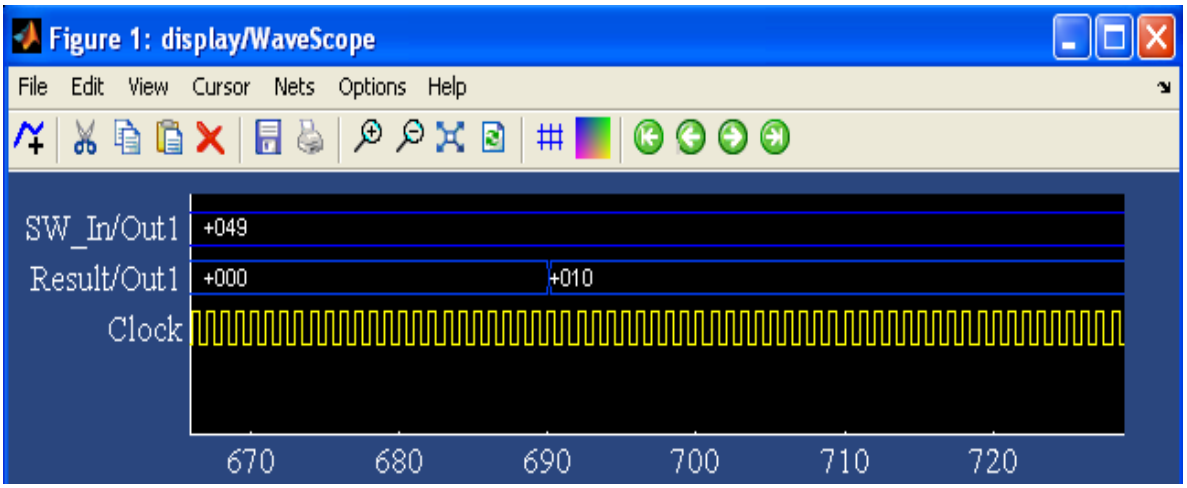

Figura 3.5. Señales de entrada y salida.

En la Figura 3.6 se puede apreciar la aplicación ejecutándose en la tarjeta FPGA. Los valores con los cuales se trabajó corresponden con los de la simulación, es decir A = 3 y B = 1, de manera que el resultado de la operación será '10' en decimal, valor que se puede apreciar en el display.

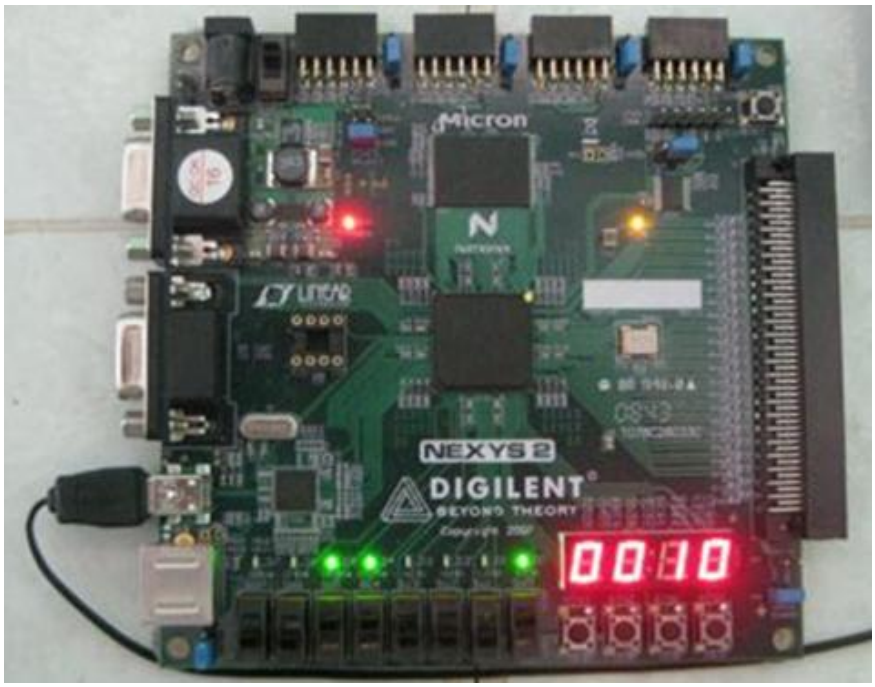

Figura 3.6. Función multiplicación en la FPGA.

El consumo de recursos en la tarjeta FPGA para esta aplicación se muestra en la Figura 3.7.

Device Utilization Summary			
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice Flip Flops	77	9,312	1%
Number of 4 input LUTs	171	9,312	1%
Number of occupied Slices	103	4,656	2%
Number of Slices containing only related logic	103	103	100%
Number of Slices containing unrelated logic	0	103	0%
Total Number of 4 input LUTs	173	9,312	1%
Number used as logic	103		
Number used as a route-thru	2		
Number used for Dual Port RAMs	16		
Number used for 32x1 RAMs	52		
Number of bonded IOBs	29	232	12%
Number of RAMB16s	2	20	10%
Number of BUFGMUXs	1	24	4%
Average Fanout of Non-Clock Nets	3.83		

Figura 3.7. Consumo de recursos en la FPGA.

3.3 Simulación e implementación del modulador BPSK

La modulación BPSK se realiza a partir de dos señales digitales: portadora (*CarryLED*) y moduladora (*SignalLED*). Estas señales son generadas por los contadores **Counter1** y **Counter2** respectivamente, debido a que el Kit Nexys2 no posee ningún convertidor análogo-digital. Seguidamente, se envían a dos conectores Pmod de la tarjeta y además se multiplexan mediante el multiplexor **Mux** para darles entrada a PicoBlaze por dos puertos diferentes.

En la Figura 3.8 se puede apreciar el circuito implementado en System Generator. El "*Gateway Out*" **BPSK** es configurado para enviar la señal ya modulada a uno de los conectores periféricos de la tarjeta para su visualización en un osciloscopio.

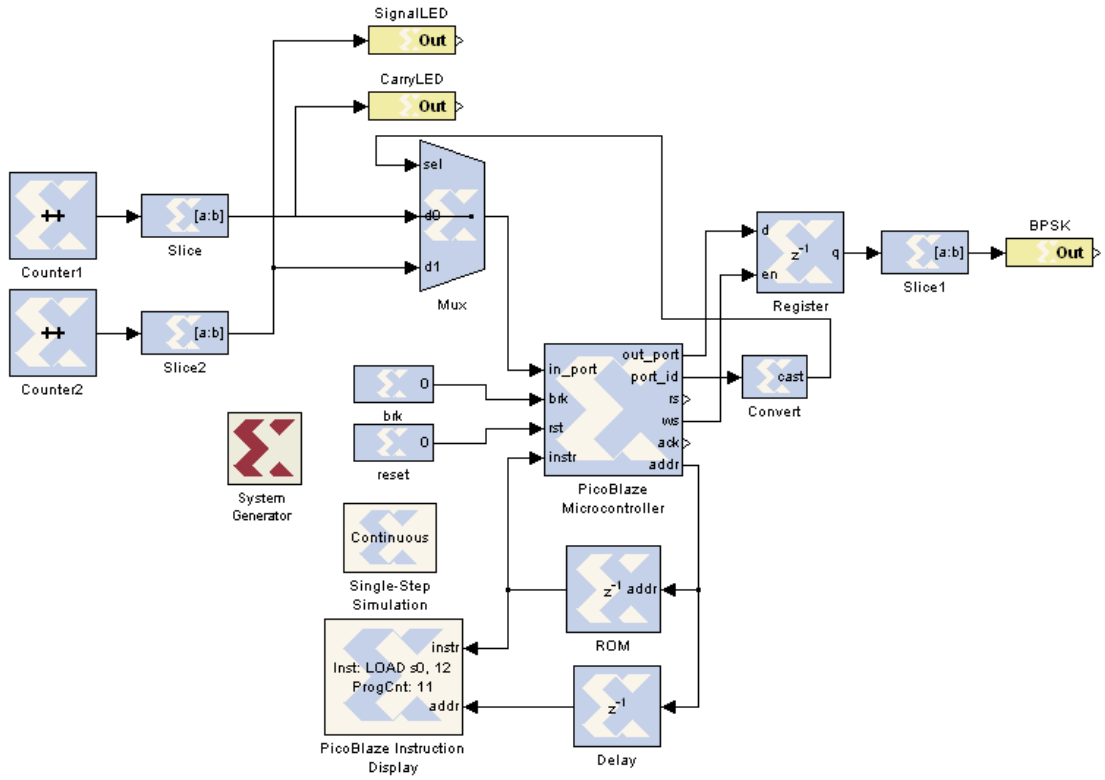

Figura 3.8. Circuito del modulador BPSK en System Generator.

La programación del código fuente de PicoBlaze para esta aplicación se basa en la determinación de los cambios de estado en la señal moduladora encontrando bordes de bajada o subida en la señal. En el momento en que un cambio de nivel es detectado

inmediatamente se cambia la fase de la señal portadora. De esta forma se logra la modulación BPSK.

En aplicaciones como los moduladores y otras donde se necesitan velocidades altas de ejecución del microcontrolador, PicoBlaze no es capaz de ejecutar las instrucciones del código fuente en correspondencia con el reloj de la FPGA en la simulación. Por esta causa fue necesario implementar tiempos prolongados en las corridas de simulación en Simulink para poder obtener resultados como los que muestran la Figura 3.9, en la cual aparecen las tres señales que intervienen en el modulador BPSK. Primeramente está la señal modulada con sus cambios de fase en los instantes en que la señal moduladora (tercera en la figura) cambia su estado, ya sea de '1' a '0' o de '0' a '1'. La segunda señal que se observa es la onda portadora.

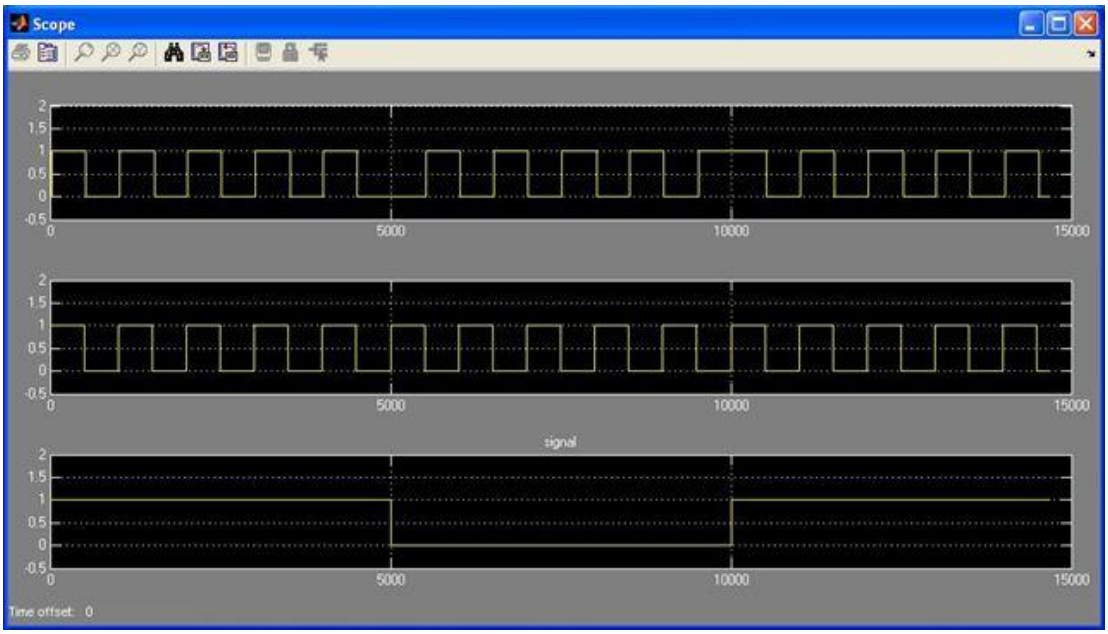

Figura 3.9. Señales de entrada y salida.

En la Figura 3.10 se aprecia el montaje del modulador en la FPGA, así como las señales en el osciloscopio. En la Figura 3.11 se presentan las mismas señales, pero en este caso a la señal modulada se le ha aplicado un filtro digital.

Figura 3.10. Modulador BPSK implementado en el Kit Nexys2.

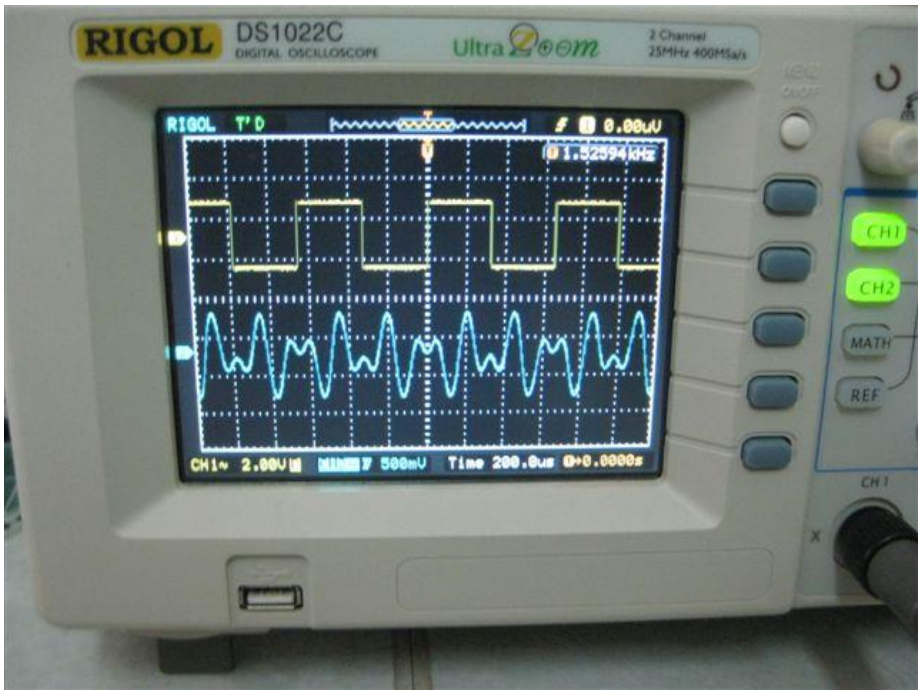

Figura 3.11. Señales del modulador BPSK.

Los recursos utilizados en la FPGA para esta aplicación pueden apreciarse en la Figura 3.12

Device Utilization Summary			
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice Flip Flops	98	9,312	1%
Number of 4 input LUTs	174	9,312	1%
Number of occupied Slices	108	4,656	2%
Number of Slices containing only related logic	108	108	100%
Number of Slices containing unrelated logic	0	108	0%
Total Number of 4 input LUTs	202	9,312	2%
Number used as logic	106		
Number used as a route-thru	28		
Number used for Dual Port RAMs	16		
Number used for 32x1 RAMs	52		
Number of bonded IOBs	4	232	1%
Number of RAMB16s	1	20	5%
Number of BUFGMUXs	1	24	4%
Average Fanout of Non-Clock Nets	3.54		

Figura 3.12. Consumo de recursos en la FPGA.

3.4 Simulación e implementación de una comunicación serie

Esta aplicación implementa una comunicación serie entre un ordenador y la tarjeta FPGA de forma unidireccional, transmitiendo de la PC a la tarjeta un caracter numérico a la vez. El número recibido se muestra en el display siete segmentos que posee la tarjeta.

En la Figura 3.13 se muestra el circuito implementado en System Generator. El bloque "*Gateway In*" llamado **Gateway In** es configurado para recibir la palabra binaria del puerto RS232 de la PC. Dado que la comunicación se inicia cuando el puerto de la PC pasa de nivel alto a bajo se usa un inversor para usar esta señal como fuente de interrupción a PicoBlaze y de esta forma iniciar la captura de la palabra digital.

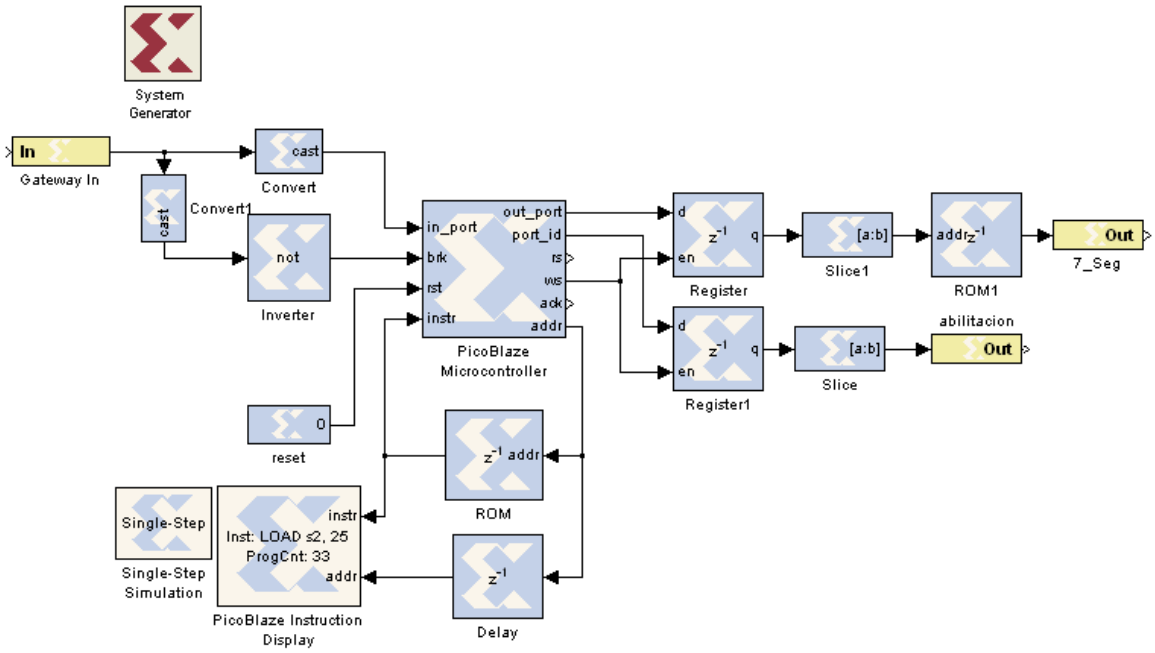

Figura 3.13. Circuito de la aplicación SerieCom en System Generator.

Los registros, así como la memoria ROM tienen como fin visualizar en el display siete segmentos el valor del carácter numérico que se recibe desde la PC.

El código fuente basa su funcionamiento en tomar las muestras de la palabra digital recibida de la PC en la mitad de cada tiempo de bit. Con tal fin fue necesario implementar una subrutina de demora de medio tiempo de bit. Este tiempo se definió para una velocidad de transmisión entre la PC y la FPGA de 1200 baudios.

En la Figura 3.14 se puede apreciar la aplicación ejecutándose en la tarjeta FPGA mientras recibe un ´1´ lógico de la PC.

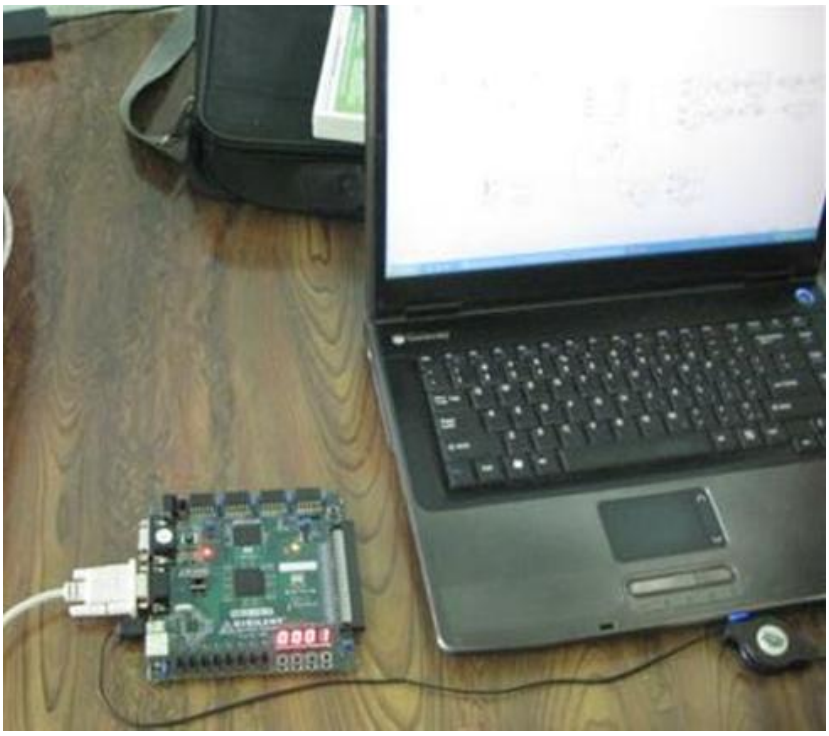

Figura 3.14. Montaje en la FPGA de la aplicación SerieCom.

La Figura 3.15 muestra los recursos utilizados en la FPGA por esta aplicación.

Device Utilization Summary			
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice Flip Flops	83	9,312	1%
Number of 4 input LUTs	175	9,312	1%
Number of occupied Slices	106	4,656	2%
Number of Slices containing only related logic	106	106	100%
Number of Slices containing unrelated logic	0	106	0%
Total Number of 4 input LUTs	177	9,312	1%
Number used as logic	107		
Number used as a route-thru	2		
Number used for Dual Port RAMs	16		
Number used for 32x1 RAMs	52		
Number of bonded IOBs	14	232	6%
Number of RAMB16s	2	20	10%
Number of BUFGMUXs	1	24	4%
Average Fanout of Non-Clock Nets	3.92		

Figura 3.15. Consumo de recursos en la FPGA.

3.5 Conclusiones del capítulo

En este capítulo se presentó la implementación de las aplicaciones en System Generator y en el Kit Nexys2. Se observó que tanto el desarrollo de las aplicaciones en System Generator, como el montaje en la FPGA, resultan sencillos debido al alto nivel de interacción con el usuario que permiten. Además, el Kit Nexys2 no requiere recursos externos para la implementación de aplicaciones como las desarrolladas.

CONCLUSIONES Y RECOMENDACIONES

Conclusiones

Con la realización de este trabajo se desarrollaron tres aplicaciones con el microcontrolador PicoBlaze en System Generator y se comprobó el funcionamiento de las mismas en una FPGA Spartan 3E de Xilinx. Esto ha permitido arribar a las siguientes conclusiones y recomendaciones:

1. PicoBlaze es un microcontrolador versátil, flexible y de uso libre, con una interfaz amistosa, lo cual justifica su uso en la confección de aplicaciones en distintas esferas de investigación.
2. System Generator brinda la posibilidad de realizar el diseño de aplicaciones, generar su código HDL y el fichero de configuración para la FPGA, así como definir los puertos a usar en la tarjeta, todo desde una interfaz amigable y sencilla para el usuario. Además permite realizar la simulación del sistema.
3. En aplicaciones que demandan altas velocidades, System Generator no es capaz de simular la ejecución de las instrucciones implementadas en PicoBlaze en tiempo real, a diferencia de otros bloques como los contadores.
4. Matlab constituye una herramienta útil para el diseño de sistemas con PicoBlaze puesto que además de incluir a System Generator como un toolbox de Simulink, permite ensamblar el código fuente de PicoBlaze. De este modo no es necesario el uso de otros softwares adicionales en el desarrollo de aplicaciones.

Recomendaciones

1. Investigar sobre el empleo de microcontroladores empotrados para la implementación de aplicaciones en Cuba.
2. Profundizar en el estudio de System Generator para el desarrollo de aplicaciones con las FPGA de Xilinx usando el microcontrolador PicoBlaze.
3. Usar a PicoBlaze para la docencia en instituciones que impartan asignaturas relacionadas con el diseño digital y los procesadores.

REFERENCIAS BIBLIOGRÁFICAS

ARACIL, J. & GÓMEZ, F. 2007. Introducción a Matlab y Simulink. [en línea]. Disponible en: http://www.esi2.us.es/~fabio/apuntes_matlab.pdf [Consultado 23 mayo 2012].

ARANTXA. 2010. Sistemas embebidos. Implementación en un FPGA del microcontrolador PicoBlaze. [en línea]. Universidad Autónoma de Madrid, España. Disponible en: http://arantxa.ii.uam.es/~edcd/lab/Practica2.pdf [Consultado 24 febrero 2012].

ARIAS, R. O. 2010. Metodología para el diseño de aplicaciones medianas en FPGAs de Xilinx., Universidad Central "Marta Abreus" de las Villas.

BORENSZTEIN, P. 2010. Diseño de sistemas con FPGA. [en línea]. Disponible en: http://www.xilinx.com/ipcenter/processor_central/picoblaze/picoblaze_userresources.html [Consultado 15 febrero 2012].

BORENSZTEJN, P. 2009. PicoBlaze. Diseño de sistemas con FPGA. [en línea]. Disponible en: http://www.dc.uba.ar/materias/disfpga/2010/c1/descargas/PicoBlaze.pdf [Consultado 15 febrero 2012].

CARRIÓN, C. 2011. Simulación e implementación de la modulación BPSK en una FPGA Xilinx Spartan 3 xcs200-4ftp256. [en línea]. Disponible en: http://www.monografias.com/trabajos33/implementacion-bpsk/implementacion-bpsk.shtml [Consultado 14 mayo 2012].

CHAPMAN, K. 2003. UART Real Time Clock. [en línea]. Disponible en: http://www.xilinx.com/bvdocs/appnotes/uart-clock.pdf [Consultado 3 abril 2012].

CHAPMAN, K. 2003. UART Transmitter and Receiver Macros. [en línea]. Disponible en: http://www.xilinx.com/bvdocs/appnotes/uart-manual.pdf [Consultado 3 abril 2012].

CHAPMAN, K. 2005. PicoBlaze 8-bit Embedded Microcontroller User Guide. [en línea]. Disponible en: http://www.eng.auburn.edu/~strouce/class/elec4200/ug129.pdf [Consultado 3 abril 2012].

CHAPMAN, K. 2006. Amplifier and A/D Converter Control for Spartan-3E Starter Kit. [en línea]. Disponible en: http://www.eng.auburn.edu/~strouce/class/elec4200/ug130.pdf [Consultado 3 abril 2012].

CHAPMAN, K. 2006. D/A Converter Control for Spartan-3E Starter Kit. [en línea]. Disponible en: http://www.eng.auburn.edu/~strouce/class/elec4200/ug117.pdf [Consultado 3 abril 2012].

CHAPMAN, K. 2006. Frequency Counter for Spartan-3E Starter Kit (with test oscillators). [en línea]. Disponible en: http://www.eng.auburn.edu/~strouce/class/elec4200/ug137.pdf [Consultado 3 abril 2012].

CHAPMAN, K. 2006. Initial Design for Spartan-3E Starter Kit (LCD Display Control). [en línea]. Disponible en: http://www.eng.auburn.edu/~strouce/class/elec4200/ug111.pdf [Consultado 3 abril 2012].

CHAPMAN, K. 2006. Software Implementation of Pulse Width Modulation (PWM). [en línea]. Disponible en: http://www.eng.auburn.edu/~strouce/class/elec4200/ug130.pdf [Consultado 3 abril 2012].

DIGILENT. 2008. Digilent Nexys2 Board Reference Manual. [en línea]. Disponible en: http://www.digilentinc.com/Data/Products/NEXYS2/Nexys2rm.pdf [Consultado 19 febrero 2012].

GÓMEZ, J. A. 2011. Práctica de diseño y prototipado sobre circuito FPGA. Contador modelado mediante el procesador Picoblaze. [en línea]. Universidad de Extremadura, España. Disponible en: http://arco.unex.es/ise6_xsv/contadorpico.htm [Consultado 24 febrero 2012].

GÜICHAL, G. 2005. Diseño Digital Utilizando Lógicas Programables. [en línea]. Universidad Tecnológica Nacional, Facultad Regional Bahía Blanca. Disponible en:

http://laboratorios.fi.uba.ar/lse/sase/2010/slides/SASE-2010-FPGA-Guichal.pdf [Consultado 25 abril 2012].

GUTIÉRREZ, M. 2009. Evaluación de herramientas de alto nivel para diseño hardware. [en línea]. Disponible en: http://ccc.inaoep.mx/fpgacentral/pdf/herramienthardw.pdf [Consultado 12 mayo 2012].

HUERTA, P. 2010. Sistemas MPSoC en FPGAs. [en línea]. Departamento de Arquitectura y Tecnología de Computadoras, Universidad Rey Juan Carlos, España. Disponible en: http://www.escet.urjc.es/~phuerta/pdf/442.pdf [Consultado 21 febrero 2012].

MAHMOUD, W. H. Incorporating System-Level Design Tools into Upper-Level Electrical Engineering Courses 2008 Universidad del Distrito Colombia.

MALPICA, N. Principios de los microcontroladores. 2006 Universidad Rey Juan Carlos.

MATHWORKS, T. 2009. Simulink Getting Started Guide. [en línea]. Disponible en: http://www.mathworks.com/access/helpdesk/help/pdf_doc/simulink/sl_gs.pdf [Consultado 3 mayo 2012].

MATPIC. 2011. Microprocesador PicoBlaze. [en línea]. Disponible en: http://www.matpic.com/esp/vhdl/picoblaze.html [Consultado 14 enero 2012].

PEÑA, J. 2008. Diseño de módulos para el manejo de Puertos, Temporización e Interrupciones para el Núcleo KCPSM3 e Implementación en el FPGA XC3S500 de Xilinx. [En línea]. Universidad Tecnológica de la Mixteca. Disponible en: http://jupiter.utm.mx/~tesis_dig/10621.pdf [Consultado 15 mayo 2012].

RODRÍGUEZ, J. 2009. Creación de los experimentos del curso IE0524. Universidad de Costa Rica.

ROSADO, A. & BATALLER, M. 2003. Práctica 1. Introducción al software Xilinx ISE Version 6. [en línea]. Disponible en: http://www.uv.es/rosado/dcse/prac1XilinxISE introd.pdf [Consultado 15 marzo 2012].

SANTANA, Y. B. 2011. Metodología para desarrollar aplicaciones con el PicoBlaze de Xilinx. Universidad Central "Marta Abreus" de las Villas.

XILINX, I. 2009. MicroBlaze Processor Reference Guide, Embedded Development Kit EDK 9.2i. [en línea]. Disponible en: http://dce.felk.cvut.cz/msy/files/fpga/MicroBlaze/micorblazeproductbrief.pdf [Consultado 14 marzo 2012].

XILINX, I. 2011. System Generator for DSP User Guide. [en línea]. Disponible en: http://www.xilinx.com/support/systgeneratordsp.htm [Consultado 7 enero 2012].

ANEXOS

Anexo I Asignación de pines de los conectores Pmods (Digilent, 2008).

Pmod JA		Pmod JB		Pmod JC		Pmod JD	
JA1: L15	JA7: K13	JB1: M13	JB7: P17	JC1: G15	JC7: H15	JD1: J13	JD7: K14[1]
JA2: K12	JA8: L16	JB2: R18	JB8: R16	JC2: J16	JC8: F14	JD2: M18	JD8: K15[2]
JA3: L17	JA9: M14	JB3: R15	JB9: T18	JC3: G13	JC9: G16	JD3: N18	JD9: J15[3]
JA4: M15	JA10: M16	JB4: T17	JB10: U18	JC4: H16	JC10: J12	JD4:; P18	JD10: J14[4]

Anexo II Set de instrucciones de PicoBlaze (Xilinx, 2011).

GRUPO	INSTRUCCIÓN
Control de Programa	JUMP aaa
	JUMP Z, aaa
	JUMP NZ, aaa
	JUMP C, aaa
	JUMP NC, aaa
	CALL aaa
	CALL Z, aaa
	CALL NZ, aaa
	CALL C, aaa
	CALL NC, aaa
	RETURN
	RETURN Z
	RETURN NZ
	RETURN C
	RETURN NC
Aritméticas	ADD sX, kk
	ADDCY sX, kk
	SUB sX, kk
	SUBCY sX, kk
	COMPARE sX, kk
	ADD sX, sY
	ADDCY sX, sY
	SUB sX, sY
	SUBCY sX, sY

	COMPARE	sX, sY
Interrupciones	RETURNI	ENABLE
	RETURNI	DISABLE
	ENABLE	INTERRUPT
	DISABLE	INTERRUPT
Lógicas	LOAD	sX, kk
	AND	sX, kk
	OR	sX, kk
	XOR	sX, kk
	TEST	sX, kk
	LOAD	sX, sY
	AND	sX, sY
	OR	sX, sY
	XOR	sX, sY
	TEST	sX, sY
Acceso a Memoria	STORE	sX, ss
	STORE	sX, (sY)
	FETCH	sX, ss
	FETCH	X, (sY)
Desplazamiento y Rotaciones	SR0	sX
	SR1	sX
	SRX	sX
	SRA	sX
	RR	sX
	SL0	sX
	SL1	sX
	SLX	sX
	SLA	sX
	RL	sX
Entrada y Salida	INPUT	sX, pp
	INPUT	sX, (sY)
	OUTPUT	sX, pp
	OUTPUT	sX, (sY)

La notación empleada para los operandos es:

- sX Uno de los 16 registros entre el rango de s0 a sF.
- sY Uno de los 16 registros entre el rango de s0 a sF.
- 'kk' Valor constante en el rango de 00 a FF.
- 'aaa' Dirección en el rango de 000 a 3FF.
- 'pp' Dirección de un puerto en el rango de 00 a FF
- 'ss' Dirección de almacenamiento interno en el rango de 00 a 3F.

Anexo III Código KCPSM3 de la función $A^2 + B^2$.

```
CONSTANT delay_1us_constant, 0B
constant UP_NIBBLE_MASK, 0F ; 0000 1111
constant a_lsb, 00
constant b_lsb, 02
constant aa_lsb, 04
constant aa_msb, 05
constant bb_lsb, 06
constant bb_msb, 07
constant aabb_lsb, 08
constant aabb_msb, 09
constant aabb_cout, 0A
namereg s8, data
namereg s9, addr
nameregsa, i
nameregsf, sw_in
constant sw_port, 01

  ADDRESS 000
cont: call clr_data_mem
callread_switch
call square
callwrite_led
calldisplay_mues
jumpcont

display_mues:
      LOAD s7, se
      LOAD sb, 00
      OUTPUT sb, FE
      CALL delay_1ms
      CALL delay_1ms
      LOAD sc, 64
call num_3
      OUTPUT sb, FD
      CALL delay_1ms
      CALL delay_1ms
      LOAD sc, 0A
call num_3
      OUTPUT sb, FB
      CALL delay_1ms
      CALL delay_1ms
      LOAD sb, s7
      OUTPUT sb, F7
      CALL delay_1ms
      CALL delay_1ms
```

```
        RETURN

num_3:  LOAD sb, 00
       COMPARE s7, sc
       JUMP NC, mayor_que
       JUMP fin3

mayor_que:  LOAD sd, s7
     otro3:SUBsd,  sc
COMPARE sd, sc
          JUMP c, fin2
          ADD sb, 01
          LOAD s7, sd
          JUMP otro3
 fin2:  ADD sb, 01
      LOAD s7, sd
 fin3:  RETURN

delay_1us:  LOAD s0, delay_1us_constant
      wait_1us:  SUB s0, 01
                 JUMP NZ, wait_1us
                 RETURN

    delay_40us:  LOAD s1, 28
     wait_40us:  CALL delay_1us
                 SUB s1, 01
                 JUMP NZ, wait_40us
                 RETURN

     delay_1ms:  LOAD s2, 19
      wait_1ms:  CALL delay_40us
                 SUB s2, 01
                 JUMP NZ, wait_1ms
                 RETURN

clr_data_mem   :
load   i, 40
load   data, 00
clr_mem_loop:
store  data, (i)
sub   i, 01
jump   nz, clr_mem_loop
return

read_switch:
input   sw_in,  sw_port
load   data, sw_in
```

```
call  get_lower_nibble
store  data, a_lsb
load  data, sw_in
call  get_upper_nibble
store  data,  b_lsb

get_lower_nibble:
and  data, UP_NIBBLE_MASK
return

get_upper_nibble:
sr0  data ; right  shift  4  times
sr0  data
sr0  data
sr0  data
return

write_led:
fetch  data, aabb_lsb
 LOAD se,  data
 RETURN

square  :
fetch  s3, a_lsb
fetch  s4, a_lsb
call  mult_soft
store  s6,  aa_lsb
store  s5,  aa_msb

fetch  s3,  b_lsb
fetch  s4,  b_lsb
call  mult_soft
store  s6,  bb_lsb
store  s5,  07

fetch  data,  aa_lsb
add  data,  s6
store  data,  aabb_lsb
fetch  data,  aa_msb
addcy  data,  s5
store  data,  aabb_msb
load  data,  00
addcy  data,  00
store  data,  aabb_cout
return

mult_soft:
```

```
load  s5,  00
load  i, 08
mult_loop:
sr0  s4
jumpnc,   shift_prod
add  s5,  s3
shift_prod:
sra s5
sra  s6
sub  i, 01
jumpnz,   mult_loop
RETURN
```

Anexo IV Código KCPSM3 de la aplicación SerieCom.

```
CONSTANT delay_1us_constant,  0B

ADDRESS  000
ENABLE  INTERRUPT

cont:CALLdisplay_mues
    JUMP cont

interrup:  CALL delay_mitad_bit
        CALL delay_mitad_bit
        CALL delay_mitad_bit
        LOAD s8,08
        LOAD s6,00
  cont2: INPUT s9, 01
        AND s9, 01
        OR s6, s9
        RR s6
        CALL delay_mitad_bit
        CALL delay_mitad_bit
        SUB s8, 01
        JUMP NZ, cont2
        OUTPUT s6, 01
        CALL delay_1ms
        RETURNI  ENABLE

delay_1us:  LOAD s0, delay_1us_constant
      wait_1us:  SUB s0, 01
            JUMP NZ, wait_1us
            RETURN

    delay_40us:  LOAD s1, 26
```

```
    wait_40us: CALL delay_1us
               SUB s1, 01
               JUMP NZ, wait_40us
               RETURN

delay_mitad_bit: LOAD s2, 0A
wait_ms: CALL delay_40us
               SUB s2, 01
               JUMP NZ, wait_ms
               CALL delay_1us
               CALL delay_1us
               CALL delay_1us
               CALL delay_1us
               RETURN

     delay_1ms: LOAD s2, 19
     wait_1ms: CALL delay_40us
               SUB s2, 01
JUMP NZ, wait_1ms
               RETURN
display_mues:
       LOAD sb, 00
       OUTPUT sb, FE
       CALL delay_1ms
       CALL delay_1ms
       OUTPUT sb, FD
       CALL delay_1ms
       CALL delay_1ms
       OUTPUT sb, FB
       CALL delay_1ms
       CALL delay_1ms
       LOAD sb, s6
       OUTPUT sb, F7
       CALL delay_1ms
       CALL delay_1ms
       RETURN

  ADDRESS 3FF

JUMP interrup
```