UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

Diseño y síntesis de una tarjeta electrónica, de propósito general, didáctica, basada en tecnología de dispositivos lógicos programables (PLD's) de hardware abierto

T E S I S

QUE PARA OBTENER EL TÍTULO DE:

INGENIERO ELÉCTRICO ELECTRÓNICO

PRESENTAN:

María Lucelde Fernández Herrera

DIRECTOR DE TESIS:

M.I. Ricardo Mota Marzano

CIUDAD UNIVERSITARIA, 2009

Agradecimientos

Primeramente agradezco a Dios por todos aquellos acontecimientos que marcaron momentos felices y satisfactorios en mi vida al lado de las personas que amo y por tener la oportunidad de conocer a personas que me han ayudado a ser mejor.

A mis padres que siempre me brindaron su amor, cariño, comprensión y apoyo incondicional y que para mi han sido un ejemplo de constante esfuerzo y dedicación.

A mis hermanas y hermano por compatir la dicha de ser una familia unida.

A mi asesor de tesis el M.I. Ricardo Mota Marzano por todo el apoyo, dedicación, atención, esmero, paciencia y conocimientos que me bindó.

Por todos mis amigos y amigas Avril, Zaira, Elizabeth, José Guadalupe, Natanael, Jorge Addi, que me ofrecieron incondicionalmente su amistad.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

Introducción

La tecnología de Dispositivos Lógicos Programables (PLD's), y en especial del tipo de Arreglo de Compuertas Programables en Campo (FPGA's), ha logrado colocarse como una de las opciones tecnológicas ventajosas para el diseño de circuitos digitales a la medida y para aplicaciones que exigen establecer etapas de procesamiento paralelo en un mismo dispositivo; así como para aquellas que requieran características de operación muy especiales

En las últimas dos décadas, el desarrollo de los Dispositivos Lógicos Programables (PLD's) y en particular de los denominados FPGA's ha sido más que relevante, esto se ve reflejado en las avanzadas arquitecturas que ,actualmente, poseen estos dispositivos, ya que cuentan con módulos embebidos o etapas fijas (*hardwired*) de operación específica, como son bloques de memoria, PLL's, multiplicadores, núcleos de procesamiento, osciladores, entre otros; y que en conjunto con las herramientas integradas de diseño (Ambiente Integrado de Diseño, IDE) y los lenguajes descriptivos de hardware (VHLD, Verilog, System C) representan una de las mejores opciones para llevar a cabo el diseño, simulación y síntesis de circuitos digitales a la medida .

Las ventajas de usar la tecnología de PLD's, son el conocimiento por defecto de la arquitectura sintetizada, la posibilidad de configurar o programar al dispositivo en sistema (*jtag & isp*), y la opción de realizar pruebas de operación en sistema (*jtag & boundary scan*), aunado a la tendencia hacia la búsqueda de un menor costo de desarrollo y fabricación, un menor tiempo para su venta en el mercado (*time to market*, TTM) y la mayor utilización en el desarrollo de nuevos productos.

De ahí, la importancia que tiene el incorporar en la enseñanza y formación de estudiantes, en el área ingeniería aplicada de la Licenciatura en Electrónica, este tipo de herramientas para dotarlos de conocimiento suficiente y al egresar sean competitivos y tengan la capacidad de innovar en la generación de productos (comerciales o de investigación) en el menor tiempo posible, aprovechando la rápida transición entre la tecnología de PLD's

Por lo cual, se considera que la realización de este proyecto de tesis que es el desarrollo y la síntesis de una tarjeta electrónica didáctica, de hardware abierto, basada en un FPGA traerá múltiples beneficios, como abatir los costos que representa adquirir tarjetas de desarrollo comerciales basadas en este tipo de tecnología del extranjero, la no dependencia de estas, el propiciar que los alumnos inicien sus propios diseños de tarjetas, etc.; puesto que la documentación subsecuente, presenta paso a paso todo el proceso que se realizó hasta llegar su obtención.

Con esta tarjeta electrónica los alumnos podrán diseñar, simular, y finalmente sintetizar sus diseños, pudiendo comprobar físicamente su correcto funcionamiento, tomando medidas y conectando circuitos externos.

Por ser una tarjeta de hardware libre, están disponibles todos los esquemas, de manera que cualquiera la pueda fabricar, usar, modificar, y redistribuir las modificaciones, pretendiendo así, que los alumnos, una vez que hayan aprendido la funcionalidad de la tecnología de PLD's, la puedan explotar en el diseño de circuitos digitales a la medida.

Objetivo General:

Contar con hardware e información adecuados para que los estudiantes de la carrera de Ingeniería Eléctrica y Electrónica, inclinados al área de Sistemas Digitales, puedan desarrollar aplicaciones de Electrónica Digital, empleando la tecnología de Dispositivos Lógicos Programables (PLD's), específicamente un FPGA, y el Lenguaje Descriptivo de Hardware VHDL, de forma práctica, sencilla y al más bajo costo posible.

Objetivo Específico:

Diseñar una tarjeta electrónica didáctica, basada en tecnología PLD, de hardware abierto, que reúna las siguientes características:

- Basada en un FPGA de alta densidad de integración (superior a 1000 elementos lógicos).
- Acceso a todos los puertos del dispositivo.
- Interfaz electrónica de programación incluida en la misma tarjeta.
- Display de cristal líquido y displays de 7 segmentos.
- Interruptores de presión de contacto momentáneo (push).
- Interruptores deslizables (switch).
- Teclado matricial.
- Conversión analógica – digital, digital – analógica.

CAPÍTULO I: ESTADO DEL ARTE DE LOS DISPOSITIVOS LÓGICOS PROGRAMABLES

1.1. Panorama General

Los dispositivos lógicos programables (PLD's) fueron introducidos en la década de los setenta. La idea de estos dispositivos era la de construir circuitos de lógica combinacional con la característica de ser *programables.* Contrario a los microprocesadores, los cuales son capaces de ejecutar varios programas pero poseen un *hardware* fijo, los PLD's fueron pensados para programarse a nivel *hardware*. En otras palabras, un PLD es un circuito integrado, de *propósito general*, cuyo *hardware* puede reconfigurarse una y otra vez para cumplir con especificaciones particulares.

Los primeros PLD's recibieron el nombre de PAL (*Programmable Array Logic*, Arreglos Lógicos Programables) o bien, PLA (*Programmable Logic Array*, Arreglos de lógica programable), dependiendo uno u otro nombre del esquema de programación que utilizan (discutido más adelante). Los PAL / PLA utilizaban únicamente compuertas lógicas (no *flip-flops*), por lo tanto, solo era posible la síntesis de circuitos combinacionales. Para resolver este problema, se incluyeron registros a los dispositivos, llamándolos entonces PLD's con registros, los cuáles ya contaban con un *flip-flop* en cada salida del circuito, lo que permitió entonces la realización también de funciones secuenciales simples.

Al comienzo de la década de los ochentas, se agregó más circuitería lógica en las salidas de los PLD's. La nueva *celda* de salida, llamada *Macrocelda,* contenía (además del *flip-flop* adicionado) multiplexores y compuertas lógicas. De tal modo que, cada celda podía ser programada, permitiendo así una gama posible de modos de operación. Adicionalmente, cada celda era capaz de proveer una señal de realimentación desde la salida hacia el arreglo programable, lo cual le proporcionó al PLD una gran flexibilidad. Esta nueva estructura de PLD fue llamada PAL *genérico* (GAL). Una arquitectura similar a los dispositivos tipo GAL fue conocida como PALCE (*PAL CMOS Electrically erasable/programmable,* PAL de tecnología CMOS, borrable y programable eléctricamente).

A este conjunto de circuitos integrados (PAL, PLA, PLD con registros y GAL / PALCE) se le conoce actualmente como SPLDs (*Simple PLD's*, PLD's simples). Los dispositivos GAL / PALCE, son los únicos que aun se fabrican en un encapsulado autónomo.

Posteriormente, llegó la fabricación de varios dispositivos GAL en un mismo circuito integrado, empleando un esquema de interconexión más sofisticado, tecnología de silicio más avanzada y otras características más (tales como la integración de la interfaz de comunicación *JTAG – Joint Test Action Group-* y la interacción con distintos estándares lógicos). A este último desarrollo se le denominó CPLD (*Complex PLD*, PLD complejo). Actualmente los CPLD's son muy populares debido a su alta densidad, alto rendimiento y bajo costo.

Finalmente, a midiados de los ochenta, se introdujeron los dispositivos llamados FPGA's (*Field Programmable Gate Array*, Arreglo de compuertas programable en campo). Los FPGA's difieren de los CPLD's en cuanto a arquitectura, tecnología de almacenamiento de información y costo, además de ciertas funciones de alto nivel que facilitan la labor del diseñador y que se encuentran embebidas en los FPGA's, por lo que los estos últimos suelen utilizarse principalmente en circuitos cuya síntesis es de gran tamaño y exige un alto rendimiento.

En la figura 1.1 se muestra un resumen de la evolución de los PLD's.

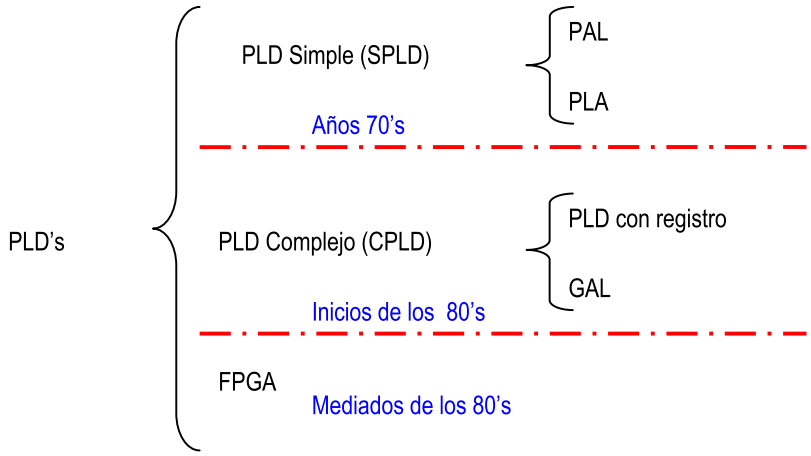


Figura 1.1. Evolución de los PLD's.

Un comentario importante es que todos los SPLD's y CPLD's son no-volátiles. Por lo que pueden ser circuitos OTP (*one-time programmable,* programables una sola vez), en cuyo caso se emplearon fusibles o antifusibles al momento de su fabricación; o bien pueden ser re-programables, fabricados con memorias EEPROM o Flash (en dispositivos nuevos la tendencia se inclina hacia tecnología Flash). Los FPGA's, por otro lado, son en su mayoría volátiles. Su fabricación se basa en memorias RAM estáticas (SRAM) para el almacenamiento de las conexiones, lo que permite un ahorro de espacio y a la vez un número muy elevado de interconexiones programables.

Los FPGA's basados en tecnología SRAM requieren de la inclusión de memorias ROM externas para su configuración cada vez que se enciende; más cabe mencionar que en la actualidad ya existen opciones comerciales de FPGA's con tecnología no-volátil, lo que permite prescindir del uso de etapas de memoria externa para su reconfiguración.

1.2. Dispositivos lógicos programables simples (SPLD's)

Como se mencionó anteriormente, los dispositivos PAL, PLA, GAL, son llamados en su conjunto, Dispositivos Lógicos Programables Simples (SPLD's). A continuación se describe brevemente cada tipo de arquitectura.

1.2.1. PAL

Estos dispositivos fueron introducidos por Monolithic Memories a mediados de los setentas. Su arquitectura básica se muestra en la figura 1.2, donde los círculos pequeños representan conexiones programables. Como se puede observar, el circuito consta de un arreglo *programable* de compuertas AND, seguidas de un arreglo de compuertas OR *fijo* o *no-programable*.

La implementación descrita en la figura 1.2 se basa en el hecho de que cualquier función combinacional se puede representar como una Suma de Productos (SOP, *Sum Of Products*); esto es: si $a_1, a_2, ..., a_N$ son las entradas lógicas, entonces cualquier salida combinacional x, puede escribirse como:

$x = m_1 + m_2 + ... + m_M$,

Donde $m_i = f_i(a_1, a_2, ..., a_N)$ son los productos de la función x. Por ejemplo:

$x = a_1\overline{a_2} + a_2 a_3 \overline{a_4} + \overline{a_1}\overline{a_2} a_3 a_4 \overline{a_5}$

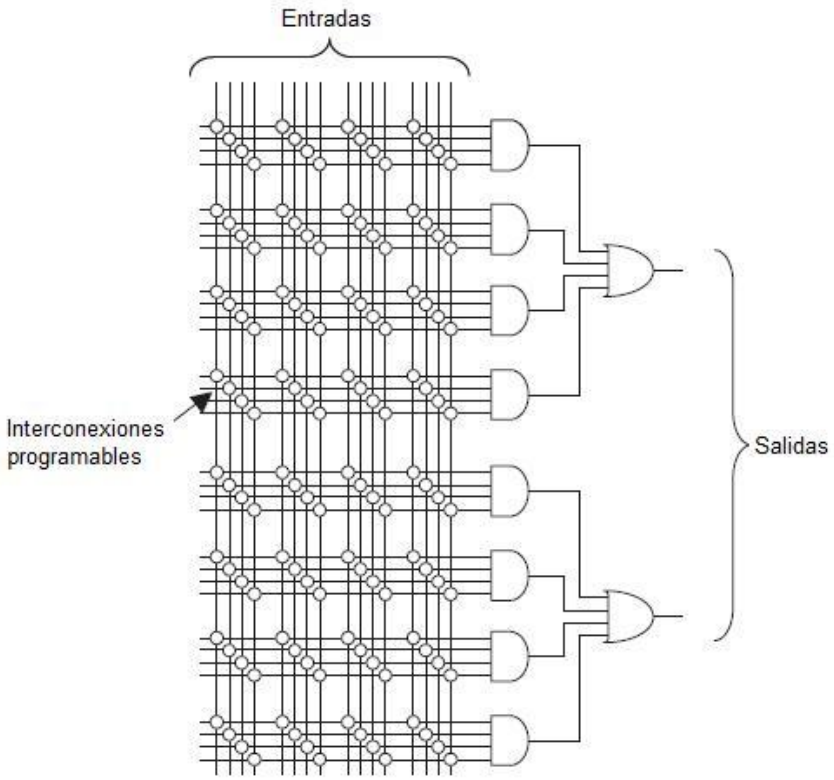


Figura 1.2. Ilustración de la arquitectura de un PAL.

Por lo tanto, estos productos representan funciones AND, cuyas salidas son posteriormente conectadas a una compuerta OR para obtener la suma, logrando así la implementación de la ecuación SOP antes descrita.

La limitación principal de estos dispositivos era que solo permitía la implementación de funciones combinacionales. Para solucionar este inconveniente, se desarrollaron dispositivos PAL con registros hacia finales del año de 1970, los cuáles incluían un *flip-flop* en cada salida (después de cada compuerta OR en la figura 1.2), lo que permitió entonces la implementación de funciones secuenciales.

La tecnología de fabricación empleada en los primeros PAL fue bipolar, con alimentación de 5 V y un consumo de corriente (con las salidas en circuito abierto) de aproximadamente 200 mA. La frecuencia máxima de operación era del orden de 100 MHz y las celdas programables eran tipo PROM (fusibles) o EPROM (tiempo de borrado de 20 minutos con rayos UV).

1.2.2. PLA

Estos dispositivos también se introdujeron en los años 1970's, por *Signetics.* Su arquitectura básica se muestra simbólicamente en la figura 1.3. Como podemos observar, a diferencia de la arquitectura de los PAL's, mostrada en la figura 1.2, la cuál está constituida por conexiones AND programables y conexiones OR 'fijas', en este caso, ambos tipos de conexiones (ANDs y ORs) son programables. Por lo que los PLAs ofrecían una mayor flexibilidad, aunque con la desventaja de ser circuitos más lentos debido a la elevación de las constantes de tiempo en los nodos internos.

Cabe mencionar que, a pesar de que actualmente los PLA's son obsoletos, éstos reaparecieron recientemente como bloques básicos dentro de la primera familia de CPLD's de bajo consumo, la familia CoolRunner de Xilinx.

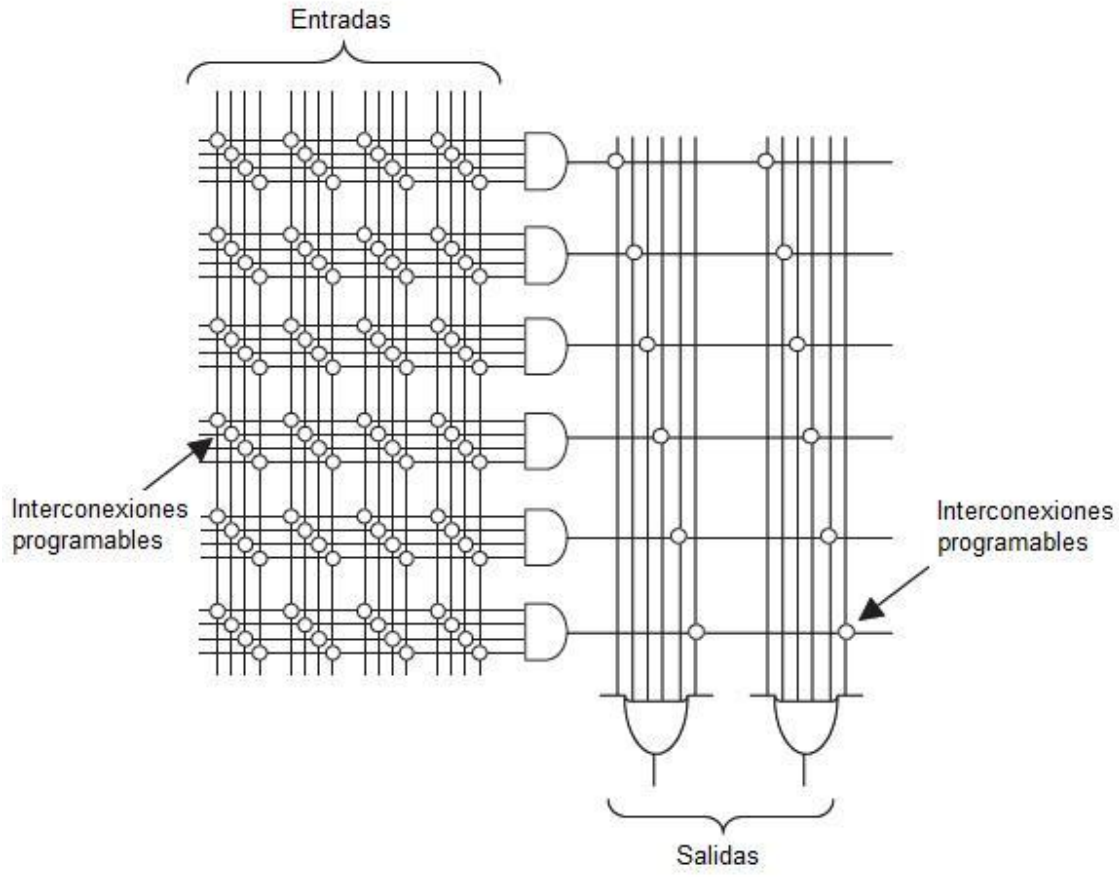


Figura 1.3. Ilustración de la arquitectura de un PLA.

1.2.3. GAL (PAL Genérico)

Esta arquitectura fue presentada por Lattice a inicios de los 1980's. Contenía bastantes mejoras con respecto a los PAL: primera, celdas de salida mucho más sofisticadas (Macroceldas), las cuáles, además de un f*lip-flop,* contenían también numerosas compuertas lógicas, así como multiplexores; segunda, dichas Macroceldas eran programables, permitiendo varios modos de operación; tercera, se incluyó también una señal de 'retorno' desde la salida de la Macrocelda, hacia el arreglo programable, brindando al circuito mayor versatilidad; cuarta, se empleó tecnología EEPROM en su fabricación, en lugar de PROM o EPROM. Además, se incluyó una firma electrónica para su identificación.

Como se mencionó anteriormente, los GAL son los únicos dispositivos SPLD (PLD Simples) que continúan manufacturándose en un encapsulado autónomo. Adicionalmente, son los bloques básicos que constituyen a la mayoría de los CPLD's actuales (con sus excepciones, como la familia CoolRunner de Xilinx, mencionada anteriormente, ya que emplea PLA's como bloques básicos).

La figura 1.4 muestra un ejemplo de un dispositivo GAL, el GAL16V8, circuito con 16-entradas, 8-salidas, en un encapsulado de 20-pines. Podemos observar una Macrocelda en cada salida (después de las compuertas OR), donde se encuentran el *flip-flop,* una serie de compuertas y multiplexores, así como la señal de realimentación de la Macrocelda hacia el arreglo programable. Las interconexiones programables se representan con círculos pequeños.

En la actualidad, los dispositivos GAL están basados en tecnología CMOS, requieren 3.3 V de alimentación, fabricados con tecnología Flash o EEPROM, y operan con una frecuencia máxima alrededor de 250 MHz. Son fabricados por compañías tales como Lattice, Atmel, TI, etc.

1.3. Dispositivos lógicos programables complejos (CPLD's)

En la figura 1.5 se muestra una aproximación de la construcción básica de un CPLD. En general, ésta consiste de numerosos SPLD's (de tipo GAL generalmente) fabricados en un solo circuito integrado, además de una matriz de interconexión programable, que sirve para conectar entre sí los SPLD's, así como hacia los pines de entrada/salida. Por otro lado, los CPLD's normalmente presentan algunos rasgos adicionales, como la integración de la interfaz de comunicación *JTAG* (*Joint Test Action Group*) y la interacción con diversos estándares lógicos (1.8 V, 2.5 V, 5 V, etc.).

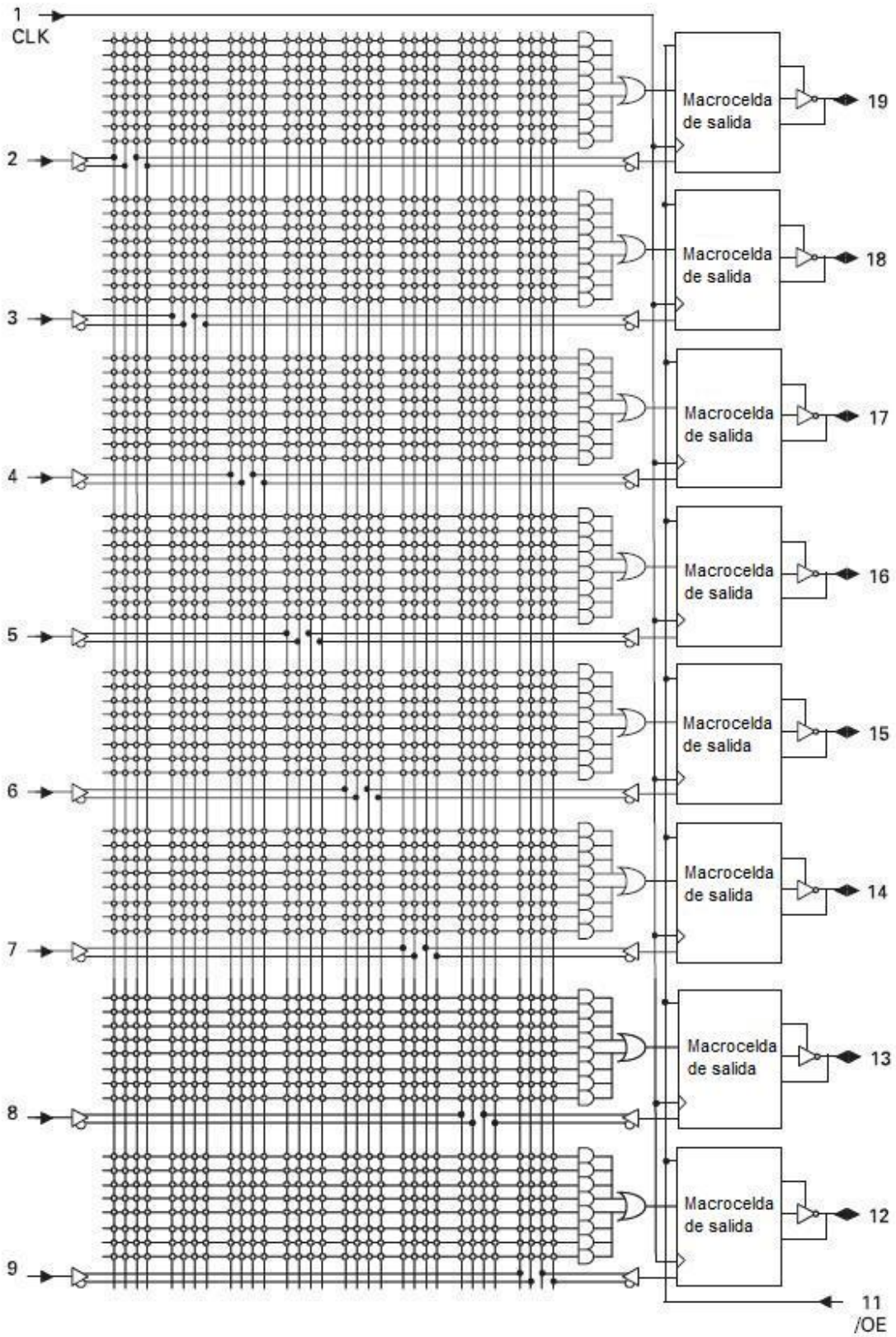


Figura 1.4. GAL16V8.

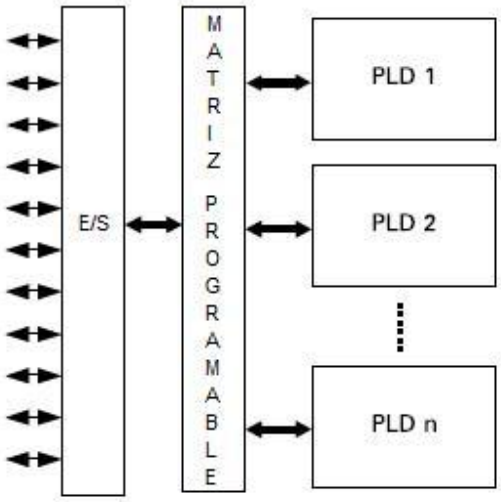


Figura 1.5. Arquitectura básica de un CPLD.

Muchas son las compañías que fabrican CPLD's, principalmente son: Altera, Xilinx, Lattice Semiconductor y Actel. En las tablas 1.1 y 1.2, se muestran ejemplos de los circuitos CPLD's fabricados por las compañías más importantes en este rubro (Altera y Xilinx), donde se encuentran dispositivos con más de 500 Macrocledas y más de 10,000 compuertas lógicas.

Familia	MAX7000 (B, AE, S)	MAX3000 (A)	MAX II (G)
Macroceldas / LUTs	32-512 macroceldas	32-512 macroceldas	240-2,210 LUTs (equivalente a 192-1,700 macroceldas)
Compuertas del sistema	600-10,000	600-10,000	
Terminales E/S	32-512	34-208	80-272
Freq. máx. de reloj interno	303 MHz	227 MHz	304 MHz
Voltaje de alimentación	2.5 V (B), 3.3 V (AE), 5 V (S)	3.3 V	1.8 V (G), 2.5 V, 3.3 V
Interconexiones	EEPROM	EEPROM	Flash + SRAM
Corriente estática	9 mA-450 mA	9 mA-150 mA	2 mA-50 mA
Tecnología	0.22 u CMOS EEPROM 4-capas de metal (7000 B)	0.3 u, 4-capas de metal	0.18 u, 6-capas de metal

Tabla 1.1. CPLD's de Altera.

Familia	XC9500 (XV, XL, -)	CoolRunner XPLA3	CoolRunner II
Macroceldas	36-288	32-512	32-512
Compuertas del sistema	800-6,400	750-12,000	750-12,000
Terminales E/S	34-192	36-260	33-270
Freq. máx. de reloj interno	222 MHz	213 MHz	385 MHz
Bloque básico de construcción	GAL54V18 (XV, XL) GAL36V18 (-)	Bloque PLA	Bloque PLA
Voltaje de alimentación	2.5 V (XV), 3.3 V (XL), 5 V	3.3 V	1.8 V (G), 2.5 V, 3.3 V
Interconexiones	Flash	EEPROM	
Corriente estática	11 mA-500 mA	< 0.1 mA	22 uA-1 mA
Tecnología	0.35 u CMOS	0.35 u CMOS	0.18 u CMOS

Tabla 1.2. CPLDs de Xilinx.

1.4. Introducción a los Arreglos de compuertas programables en campo (FPGA's)

Estos dispositivos fueron introducidos por Xilinx a mediados de la década de los ochenta. Difieren de los CPLD's en cuanto a arquitectura, tecnología de almacenamiento de interconexiones, número de funciones embebidas de alto nivel y costo. Son utilizados, principalmente, para implementar circuitos de gran tamaño y que exigen un alto desempeño.

Es importante notar que al realizar un diseño con FPGA se presentan los mismos inconvenientes que al realizar un sistema con componentes discretos, es decir toman relevancia los fenómenos de retardo de propagación y los relacionados con las señales de reloj. (Jitter, etc.).

La figura 1.6 ilustra la arquitectura básica de un FPGA, la cuál consiste en una matriz de Bloques Lógicos Configurables (CLB's, *Configurable Logic Blocks*), interconectada mediante un arreglo de interconexiones intercambiables.

Los Bloques Lógicos Configurables son distintos a las Macroceldas; básicamente, porque en lugar de implementar expresiones lógicas de suma de productos mediante el uso de compuertas AND, seguidas de compuertas OR (como en los SPLD's) para la síntesis de lógica combinacional, su operación se basa normalmente en la utilización de Tablas de Búsqueda (LUT, *Look-Up Table*).

Además de que, en un FPGA, el número de *flip-flops* es mucho mayor que en un CPLD, lo que permite la construcción de circuitos secuenciales mucho más sofisticados.

Los FPGA's también soportan la configuración mediante el puerto *JTAG* y también interactúan con distintos niveles lógicos; pero además de eso, cuentan con características adicionales, tales como multiplicación del reloj (PLL), interfaz PCI, etc. Algunos FPGA's incluso cuentan con bloques dedicados embebidos, como bloques de memoria, multiplicadores y núcleos de procesamiento.

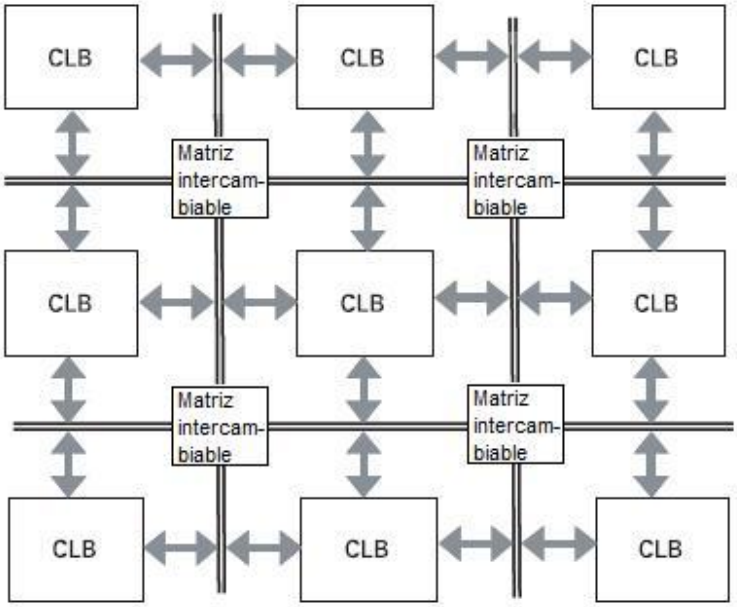


Figura 1.6. Arquitectura de un FPGA.

1.5. ¿Por qué emplear un FPGA para la tarjeta?

En los puntos 3, 4 y 5 se han explicado brevemente las principales características de los PLD's, tanto simples (SPLD's) como complejos (CPLD's), así como una pequeña introducción acerca de FPGA's.

Es de resaltar que los FPGA's son los dispositivos lógicos programables más versátiles que se pueden encontrar en la actualidad, lo cuál, en el campo de la docencia, investigación e industria es una característica muy importante. Además de eso, cabe señalar también, que dicha versatilidad representa una gran ventaja a la hora de diseñar circuitos a la medida, porque las avanzadas arquitecturas de estos dispositivos permite realizar el ruteo y procesamiento de datos de forma paralela, a velocidades muy altas, siendo posible alcanzar tasas de transferencia muy altas a mucho más bajas frecuencias, reduciendo significativamente el consumo de potencia.

Añadiendo a lo anterior, que los módulos embebidos, dedicados a funciones específicas que poseen los FPGA's, nos brindan gran flexibilidad para desarrollar aplicaciones de muy alto nivel de rendimiento, que interactúen con distintos tipos de periféricos, sin que esto represente una complejidad mayor en cuanto a diseño o utilización de hardware adicional, ya que los FPGA's son compatibles con distintos estándares lógicos y eléctricos de E/S, diferenciales y de una sola terminal (LVDS, RSDS, LVPECL, HSTL, SSTL, LVCMOS, LVTTL, etc.), además de contar con interfaces de apoyo para interactuar con distintos protocolos de comunicación (PCI, PCI-E, UARTE, I2C, etc.).

Por todo lo anterior, se justifica que un FPGA es la mejor opción para llevar a cabo el desarrollo de la tarjeta electrónica, la cual tiene fines didácticos y que se espera funja como herramienta importante en la docencia dentro de la Facultad de Ingeniería, ya que permitirá, que el personal docente y los estudiantes, realicen prácticas y/o proyectos de diferentes niveles de complejidad que vayan desarrollando sus conocimientos y capacidades entorno a la tecnología de PLD's.

CAPÍTULO II: LENGUAJES DE DESCRICPCIÓN DE *HARDWARE* (HDL'S) Y FPGA'S DISPONIBLES EN EL MERCADO

En el capítulo I se ha dado una introdución a los dispositivos lógicos programables, señalando la evolución que han sufrido hasta hoy en día. Así mismo, se han dado las razones del uso de un FPGA en el desarrollo de este trabajo.

De acuerdo a lo anterior, durante el presente capítulo se abordará la arquitectura de un FPGA, así como las opciones comerciales al momento de realizar la investigación para elegir el FPGA empleado en este diseño; destacando en este punto que los criterios en los que se basó la decisión por un dispositivo en específico son los siguientes:

- Densidad de elementos lógicos.
- Funciones de alto nivel embebidas en el dispositivo.
- Tecnología de almacenamiento de interconexiones.
- Interfaz de configuración.
- Tipo de encapsulado.
- Costo.

2.1. Arquitectura de un FPGA

La arquitectura de un FPGA varía de fabricante a fabricante. En general, la arquitectura de un FPGA cuenta con:

- Módulos embebidos, dedicados a funciones específicas.
- Recursos lógicos para la operación en modo lógico o aritmético, que son los denominados Bloques Lógicos Configurables (CLB's) o Bloques de Arreglos Lógicos (LAB's).
- Bloques de Entrada/Salida que proporcionan flexibilidad para interactuar con diferentes estándares lógicos y eléctricos.

Algunos de los módulos embebidos más comunes son: bloques de memoria, núcleos de procesamiento (microprocesadores embebidos), *Phase- Locked Loop* (PLL's), y multiplicadores.
La estructura de los bloques de memoria embebida puede ser configurada para funcionar en modos como RAM, FIFOs (*first-in first-out*) y ROM.

Los PLL's se utilizan para llevar a cabo la síntesis de señales de reloj. Se puede realizar multiplicación, división, corrimiento de fase y cambio de ciclo de trabajo de una señal de reloj.

Los multiplicadores sirven para el procesamiento digital de señales, incluyendo la implementación de filtros FIR, y la Transformada Rápida de Fourier (FFT), principalmente.
Las figuras 2.1, 2.2 y 2.3 muestran las arquitecturas de tres FPGA's de diferentes fabricantes.

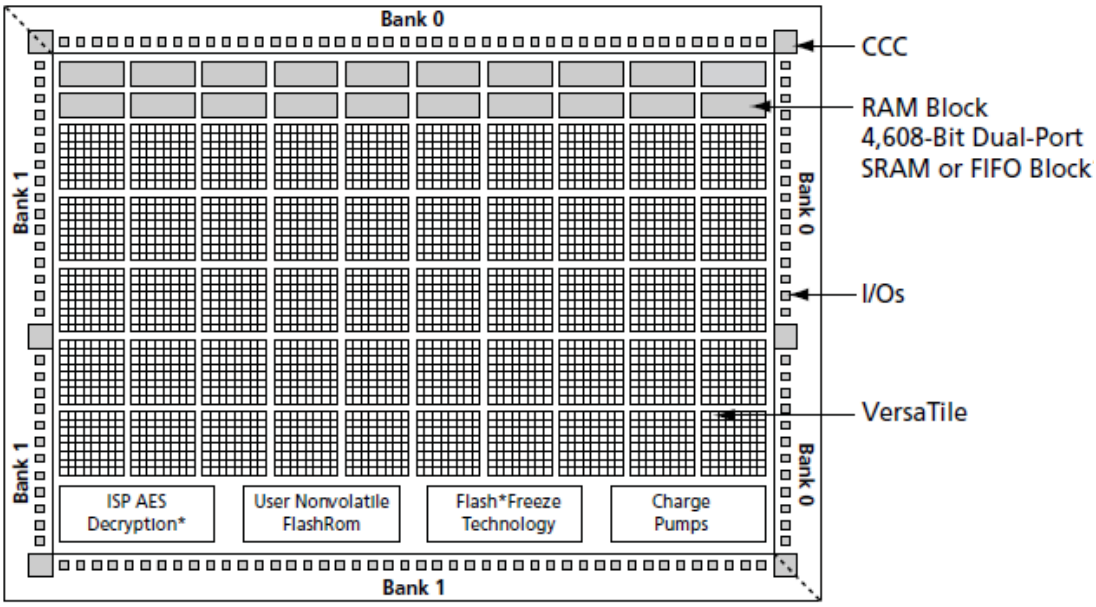


Figura 2.1. Arquitectura de la familia de FPGA's IGLOO de ACTEL (AGL015, AGL030, AGL060 y AGL125).

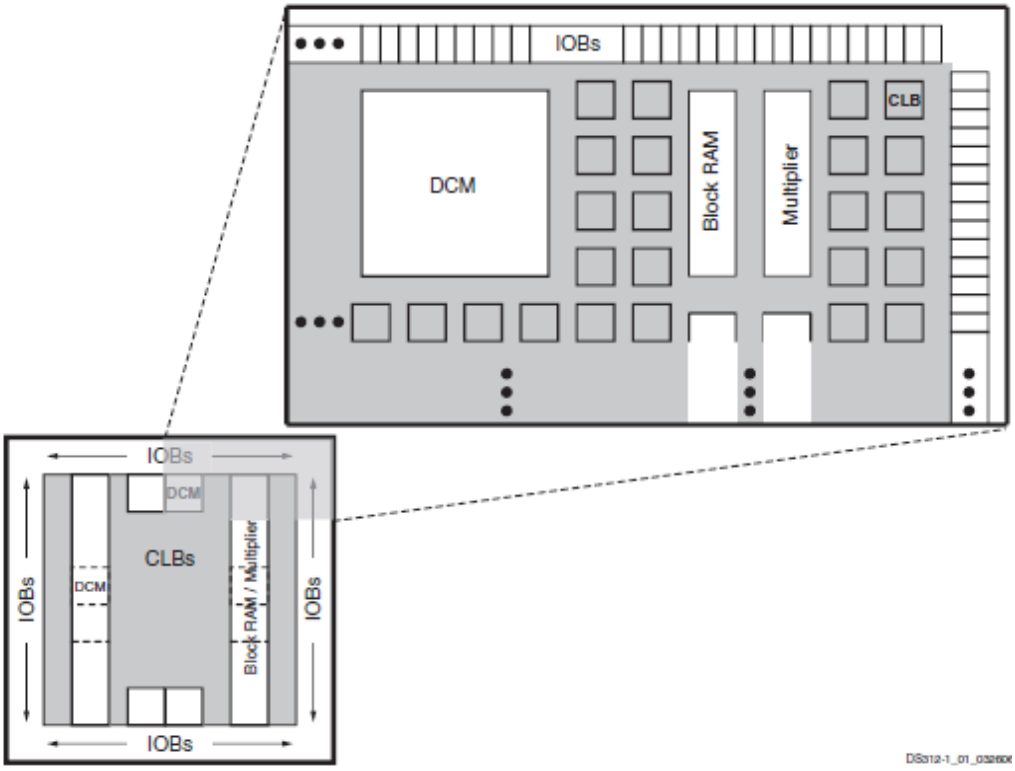


Figura 2.2. Arquitectura del FPGA Spartan-3A de XILINX.

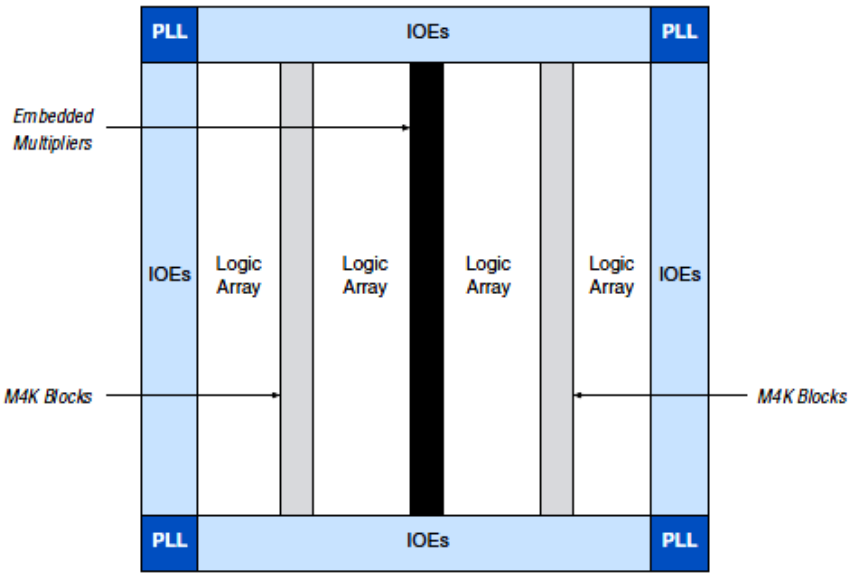


Figura 2.3. Arquitectura del FPGA EP2C20 de la familia *Cyclone II* de ALTERA.

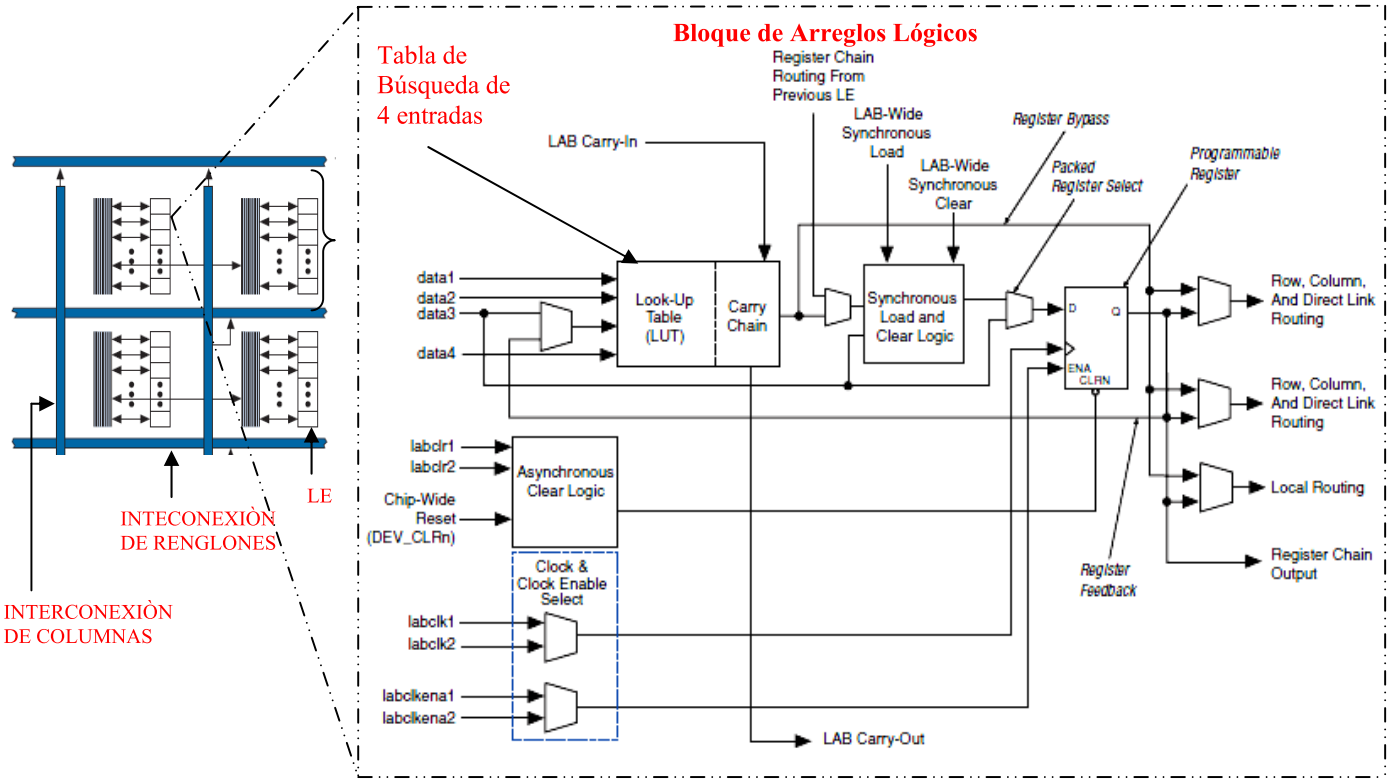


Figura 2.4. Bloque Lógico de un FPGA de la familia *CycloneII* de ALTERA.

Cada CLB o LAB, como los mostrados en las figuras 2.4 y 2.5, contiene elementos de almacenamiento, que pueden ser usados como *flip-flops* o *latches*, tablas de búsqueda (LUT's *Look up Tables*), multiplexores, compuertas, señales de control, etc.

Una LUT es un arreglo dinámico configurable para la síntesis de funciones lógicas. Las LUT's más comunes, en las arquitecturas de algunos FPGA's, son de 4 entradas las cuales pueden implementar cualquier función lógica Booleana de cuatro variables.

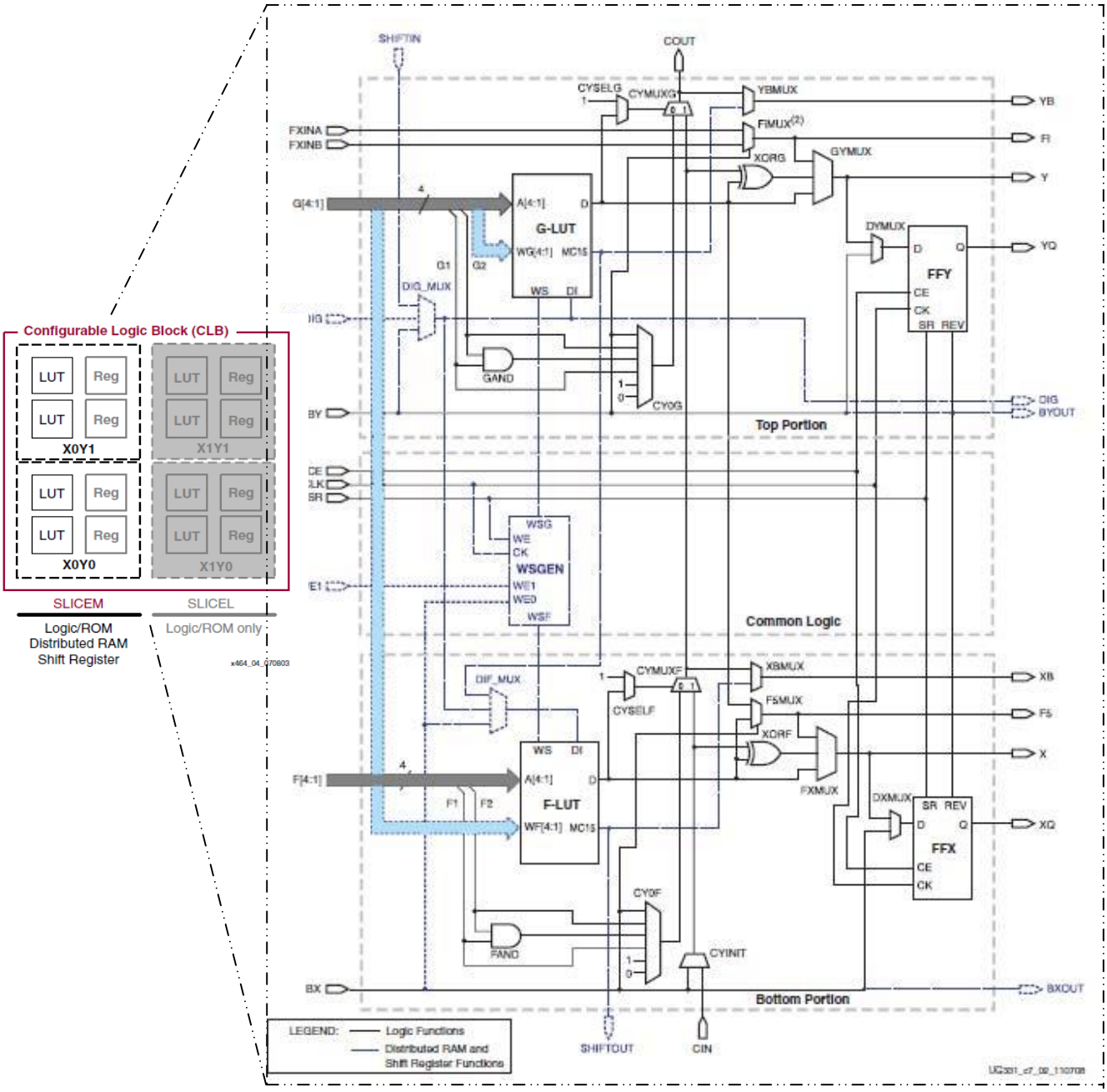


Figura 2.5. Bloque Lógico configurable de un FPGA Spartan3 de XILINX.

La mayoría de los FPGA's actuales nos brindan opciones para interactuar con distintos estándares lógicos y eléctricos de E/S diferenciales y de una sola terminal, como los LVDS, RSDS, LVPECL, HSTL, SSTL, LCVMOS, LVTTL, etc. Además, cuentan con interfaces de apoyo que nos proporcionan flexibilidad para interactuar con distintos protocolos de comunicación tales como el PCI, PCI-E, UARTE, I2C, etc.

La comunicación entre estos bloques que mencionamos depende de su arquitectura.

Las arquitecturas de algunos FPGA's están basadas en arreglos de renglones y columnas, que interconectan los módulos embebidos, los CLB's o LAB's y los bloques de Entrada/Salida.

Las interconexiones de estos bloques pueden ser directas, para unir bloques adyacentes, o globales, para comunicarse con el resto de los bloques y se llevan a cabo mediante circuitería de conmutación. En la figura 2.6 mostramos la interconexión de los bloques de un FPGA Spartan 3 del fabricante XILINX.

.

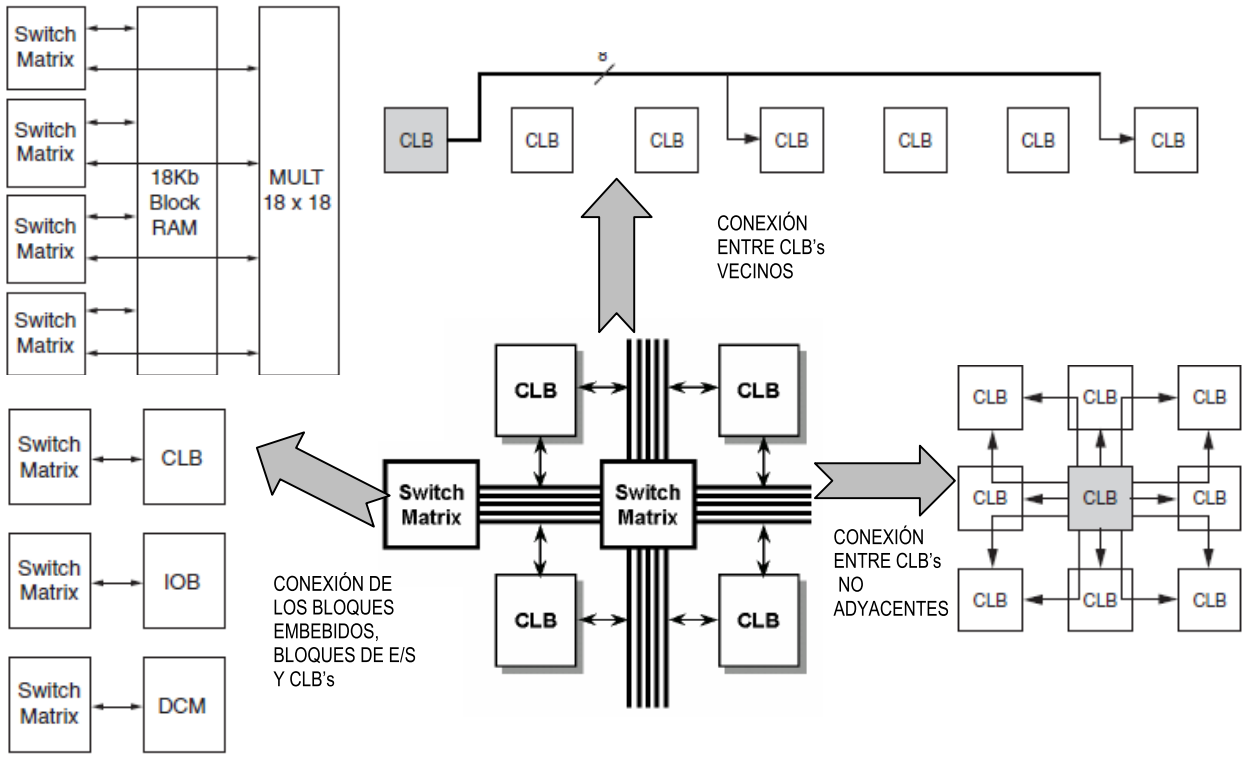


Figura 2.6.Interconexión entre los módulos internos de un FPGA.

De acuerdo a lo anterior, a continuación se proporciona de manera más detallada las características particulares de los dispositivos FPGA's fabricados por los líderes en el mercado de los dispositivos lógicos programables en la actualidad.

2.2. Estudio de los FPGA's disponibles en el mercado

Actualmente, en el mercado de los FPGA's de propósito general encontramos fabricantes importantes como Xilinx, Altera, Actel, y Lattice Semiconductor. Sus familias de dispositivos, en su mayoría, incluyen módulos embebidos o etapas fijas (*hard-wired*) de operación específica como bloques de memoria, PLL's, multiplicadores, núcleos de procesamiento, osciladores, entre otros, que son adicionales a la lógica de aplicación general integrada en el mismo dispositivo, así como también proporcionan soporte para diferentes interfaces como PCIe (*Peripheral Component Interconnect Express*), o I2C (*Inter-Integrated Circuits*), etc.

Xilinx. Ofrece las generaciones de FPGA's de alta densidad y encapsulados de gran tamaño, tales como las familias Spartan, Spartan-2, y cinco versiones de la familia Spartan-3 basadas en una tecnología SRAM de 65-nm, con múltiples densidades y grados de velocidad en cada versión. Por ejemplo, los FPGA's Spartan-3A proporcionan aplicaciones E/S de alta velocidad y los FPGA's Spartan-3A DSP proporcionan aplicaciones para el procesamiento digital de señales. La nueva familia son los FPGA's Spartan-3AN, los cuales son esencialmente un Spartan-3E, con la característica adicional de que están basados en tecnología flash para aplicaciones en sistemas no volátiles.

Altera. Ofrece las familias Cyclone, Cyclone II y Cyclone III basados en tecnología SRAM de 130-nm, 90-nm y 65-nm respectivamente.
Las densidades que integran estos dispositivos van de 3,000 a 120,000 elementos lógicos, poseen memoria embebida de 288 Kbits, 1.1 Mbits y 4Mbits además de multiplicadores, operando a 250MHz y PLL's.
Recientemente incorporan la familia de FPGA's Arria GX, con una característica adicional con respecto a la familia Cyclone, que soporta tres tipos de protocolos: PCIe, GbE (*Gigabit Ethernet*), y SRIO (*Serial RapidIO*).

Lattice Semiconductor lanzó al mercado dispositivos FPGA's con tecnología de proceso de 90-nm y 130-nm. En adición, Lattice es un proveedor en tecnología no volátil, sus FPGA's están basados en tecnología Flash. Las familias que ofrece son la EPC (*Economy Plus*) en una tecnología de proceso de 130-nm, la EPCP2 de 90-nm y posteriormente la ECP2M con características como: frecuencia de operación a 3.2 Gbps, SERDES y 5.3 Mbits de memoria y multiplicadores embebidos en el circuito integrado, soportando estándares de I/O para interactuar con memorias de tipo DDR y DDR2 y la Interfaz periférica serial (SPI4.2).

Las siguientes familias que ofrece Lattice son la XP y la XP2.

Actel .Sus FPGA's están basados en tecnología Flash y se orientan hacia aplicaciones que requieren un bajo consumo de potencia.
Entre las familias de FPGA's ofrecidas por este fabricante están la IGLOO y ProASIC3, las cuales manejan dispositivos con densidades que integran de 30,000 a 3 millón de sistemas de compuertas. La familia IGLOO posee modos adicionados para reducir aún más la potencia, el dispositivo más pequeño de esta familia, el AGLO15, puede operar a 5mW, el cual es del orden de dos a tres veces menor que el consumo en los FPGA's basados en SRAM. Actel además tiene un acuerdo con ARM que permite a los usuarios poner un core ARM7 en los FPGA's de Actel para un uso de muy baja potencia.

Las características de los FPGA's que se acaban de mencionar se muestran en el apéndice A en tablas, de acuerdo a cada fabricante.[1]

Estas características reflejan de manera contundente la evolución que los dispositivos lógicos programables han sufrido desde su surgimiento, en cuanto a su fabricación. Es de resaltar, que también la metodología de diseño de los circuitos digitales ha evolucionado con igual importancia y que en seguida se abordará.

2.3. Lenguajes de Descripción de *Hardware* (HDL's)

La evolución en la metodología de diseño de circuitos digitales se ha encaminado hacia la utilización de Lenguajes de Descripción de Hardware (*HDL's –Hardware Description Languages*-). Esto se debe a que en la metodología de diseño de circuitos, la especificación y descripción del circuito son fundamentales. La captura de esquemas, por ejemplo, ya no es adecuada si se pretende abordar un sistema con miles de elementos, por lo que se vuelve indispensable la utilización de un lenguaje para la especificación del *hardware*; así como el uso y aprovechamiento de herramientas de diseño electrónico automatizado (EDA -*Electronics Design Automation* -) y el diseño asistido por computadora (CAD –*Computer Aided Design*-).

Los lenguajes de descripción de *hardware* constituyen parte de las herramientas CAD empleadas para el diseño de circuitos. Son lenguajes mediante los cuales se generan arquitecturas electrónicas digitales con base en la descripción de su comportamiento lógico. La descripción puede llevarse a cabo de manera estructural, o bien, de manera comportamental o algorítmica:

[1] El estudio que se realizó sobre los FPGA's ofrecidos por los principales fabricantes fue hecho en el año 2008.

- **Descripción estructural**. Consiste en enumerar los componentes de un circuito y sus interconexiones. Dependiendo de la herramienta que se utilice, hay dos formas de hacerlo:
    - **Esquemas**. Es la forma tradicional en que los circuitos han sido diseñados desde los inicios de la electrónica. Consiste en la descripción gráfica de los componentes de un circuito.
    - **Lenguaje**. Se realiza una enumeración de los componentes de un circuito así como su conexión, tal es el caso del *NETLIST* que fue la primera forma de describir un circuito mediante un lenguaje.

- **Descripción comportamental o algorítmica**. Es posible describir un circuito electrónico (generalmente digital) simplemente describiendo cómo se comporta de manera lógica. Para este tipo de descripción también se utiliza un lenguaje de descripción de *hardware*.

Los HDL's, en primera instancia, son generados por cada fabricante para realizar diseños únicamente con los dispositivos que fabrican. Dada la cantidad de circuitos y herramientas de *software* actuales, se intentó establecer lenguajes estandarizados y que por lo tanto, sean reconocidos por la industria en general, independientemente de la marca o tecnología programable empleada.

Los lenguajes más importantes, y que continúan desarrollando su alcance son el VHDL (IEEE 1076), Verilog y System C.

VHDL. El significado de las siglas es VHSIC (*Very High Speed Integrated Circuits*) *Hardware Descripción Language*, es decir, lenguaje de descripción de *hardware* de circuitos integrados de muy alta velocidad. Nace como proyecto del Departamento de Defensa de EEUU en los años 80's para disponer de una herramienta estándar, independiente de los fabricantes, para la especificación (modelado y/o descripción) y documentación de los sistemas electrónicos.
La primera versión fue VHDL 87, después mejoró a la versión VHDL93. Este lenguaje de descripción de *hardware* fue estandarizado por el IEEE (*Institute of Electrical and Electronics Engineers*) a través del estándar IEEE 1076. Después se desarrolló un segundo estándar, vigente hasta la fecha, que es el IEEE1164.
VHDL está planteado para la síntesis de circuitos así como para la simulación de estos.

VERILOG. Fue inventado por Phil Moorby en 1985 mientras trabajaba en *Automated Integrated Design Systems*, más tarde renombrada *Gateway Design Automation*. El objetivo de Verilog era ser un lenguaje de modelado de *hardware*. *Gateway Design Automation* fue comprada por *Cadence Design Systems* en 1990. *Cadence* ahora tiene todos los derechos sobre los simuladores lógicos de Verilog y Verilog-XL hechos por Gateway.
Con el incremento en el éxito de VHDL, *Cadence* decidió hacer el lenguaje abierto y disponible para estandarización. *Cadence* transfirió Verilog al dominio público a través de Open Verilog International, actualmente conocida como Accellera. Verilog fue después enviado a la IEEE que lo convirtió en el estándar IEEE 1364-1995, habitualmente referido como Verilog 95.
Extensiones a Verilog 95 fueron enviadas a la IEEE para cubrir las deficiencias que los usuarios habían encontrado en el estándar original de Verilog. Estas extensiones se volvieron el estándar IEEE 1364-2001 conocido como Verilog 2001.

SystemC .Es un conjunto de librerías y macros implementadas en C++ que hacen posible una simulación de procesos concurrentes con la sintaxis del lenguaje C++ ordinario. Así los objetos descritos pueden comunicarse durante una simulación de tiempo real usando señales de cualquier tipo ofrecido por C++, además algunas otras ofrecidas por las librerías de SystemC y también otras definidas por el usuario. La metodología de diseño es comenzar con un modelo de alto nivel escrito en C++ y aplicar un proceso iterativo consistente en transformar el código para usar sólo los elementos que tengan su equivalente en un lenguaje de descripción de *hardware*.

La consolidación en la década de los noventa de los HDL's ha incorporado la implantación progresiva de la metodología de diseño electrónico denominada metodología descendente (*Top Down)* descrita a continuación:

Metodología Descendente (*Top-Down*). En la creación de diseños electrónicos, esta metodología inicia con la captura de una idea a un alto nivel de abstracción. Partiendo de esta descripción abstracta, los niveles posteriores serán más detallados según sea necesario. En cada nivel la abstracción irá disminuyendo hasta llegar a determinar los elementos discretos que sean requeridos; la figura 2.7 ilustra un ejemplo de estos niveles de abstracción.

Así por ejemplo, un circuito se divide en módulos, cada uno de los cuales representa una funcionalidad determinada. A su vez, estos módulos, dependiendo siempre de la complejidad del circuito inicial, pueden dividirse en otros módulos hasta llegar a los componentes básicos.

Las ventajas del uso de esta metodología son, principalmente, que permite la jerarquización de un diseño, por lo que al seccionarse puede ser abordado por diferentes grupos de trabajo; y que incrementa la reutilización del diseño, esto es, que la tecnología a utilizar no se fija hasta pasos posteriores en el proceso. Esto permite reutilizar los datos del diseño únicamente cambiando la tecnología de implementación.

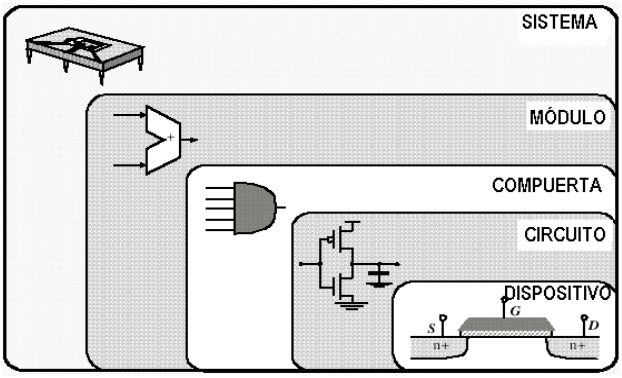


Figura 2.7. Diagrama a bloques de la metodología descendente.

En contraste, la metodología tradicional, (*Button Up),* que se había venido usando por años, cada vez resultaba menos adecuada para el manejo de sistemas complejos, por la manera en como los abordaba.

Metodología Ascendente (*Button Up*). Esta metodología plantea un diseño empezando por describir los componentes más pequeños del sistema para, posteriormente, agruparlos en diferentes módulos y estos a su vez en otros módulos hasta llegar a uno solo que represente el sistema completo que se pretende realizar, como se puede observar en la figura 2.8.

Se empieza por crear una descripción sencilla a bajo nivel, con esquemas por ejemplo, de los elementos básicos como pueden ser condensadores, chips, resistores, etc., y a partir de ahí, mediante otras herramientas hacer su implementación. Sin embargo, para diseños grandes, unir miles de componentes a bajo nivel provoca dificultades a la hora de detectar fallas en el circuito además de que su análisis se vuelve más complejo.

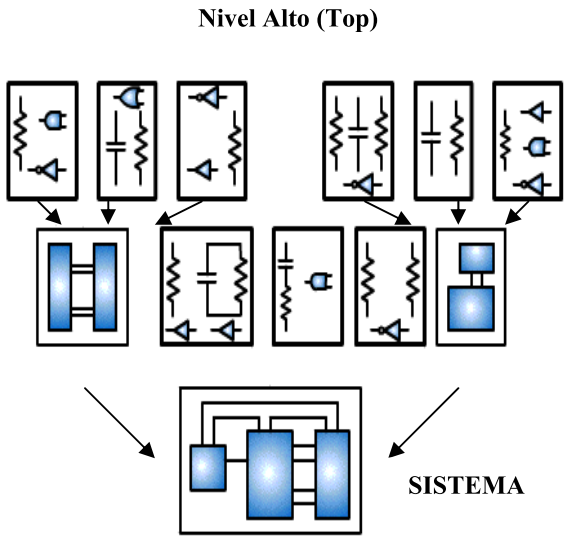


Figura 2.8. Diagrama a bloques de la metodología ascendente.

Frente a estos inconvenientes, la metodología descendente brinda una mayor flexibilidad a la hora de abordar diseños robustos utilizando algún HDL con la ventaja inmediata de garantizar la plena compatibilidad, portabilidad, reutilización e independencia tecnológica.

2.4. Elección del FPGA y del HDL a emplear en el diseño

Una vez que se indagó en el mercado actual de los FPGA's y sus principales características, así como las distintas opciones en cuanto a los lenguajes de descripción de *hardware*, fue posible, llevar a cabo la elección del FPGA más adecuado conforme a ciertos criterios que se establecieron, al igual que el HDL y la metodología de diseño a emplearse en el diseño de la tarjeta electrónica.

Como se mencionó al inicio del presente capítulo, las características que tomaron relevancia en la elección del FPGA son:

- Densidad de elementos lógicos.
- Funciones de alto nivel embebidas en el dispositivo.
- Tecnología de almacenamiento de interconexiones.
- Interfaz de configuración.
- Tipo de encapsulado.
- Costo.

Cada una de las características anteriores influenció significativamente en la elección dados los objetivos que se pretenden alcanzar con este trabajo.

Densidad de elementos lógicos: El incremento de la complejidad en las arquitecturas de los FPGA's permite que estos dispositivos integren desde 3,000 hasta 20 millones de elementos lógicos configurables.

Cantidad de funciones de alto nivel embebidas en el dispositivo: los módulos embebidos entre familias de FPGA's de diversos fabricantes comúnmente son bloques de memoria, núcleos de procesamiento, PLL's y multiplicadores.

Tecnología de almacenamiento e Interfaz de configuración: Dependiendo del tipo de la tecnología del dispositivo FPGA (Flash o SRAM), se tienen las siguientes formas de configuración:

- *In-System Programming* (ISP). Características que soportan los dispositivos lógicos programables como los FPGA's para ser configurados en sistema, es decir, se pueden configurar una vez que ya estén montados en una tarjeta de circuito impreso. Comúnmente se configura por medio del puerto *JTAG*. La programación *in-system* puede ser de dos maneras:
    - Externa. Utilizando un programador. Para el caso de los dispositivos basados en tecnología flash, estos almacenan los datos de configuración en celdas *flash–on chip*. Una vez configurado el dispositivo, los datos almacenados son parte inherente de su estructura y aún cuando se remueva la polarización del dispositivo este conserva la información cargada. En contraste, los dispositivos basados en SRAM, necesitan de una memoria no volátil externa adicional para configurarlos, ya que los datos de configuración guardados en las celdas SRAM se pierden cada vez que se interrumpe la polarización.

La interfaz electrónica que se comunica vía el puerto JTAG con el dispositivo a configurar se le conoce como programador.

- Interna. Usando un microprocesador y una memoria externa, se puede almacenar el algoritmo de configuración así como el archivo que contiene la descripción de un circuito en la memoria y usar al microprocesador para que ejecute el proceso de configuración.

Los fines que se persiguen con este trabajo están orientados al campo de la docencia, investigación e industria; de esta manera se considera necesario contar con un FPGA de un número de elementos lógicos suficientes para las posibles aplicaciones que estudiantes, investigadores y desarrolladores lleven a cabo con la tarjeta.

Se contempla la opción de que el FPGA incluya bloques embebidos como los que se han venido mencionando para aquellas aplicaciones que exijan característica de operación muy especiales.

Los fabricantes de FPGA's ofrecen dispositivos basados en dos tipos de tecnologías, Flash y SRAM, en un principio se decidió que la mejor opción para el diseño fuera un FPGA de tecnología Flash como los que Actel y Lattice Semiconductor ofrecen ya que no requieren de *hardware* adicional a la interfaz compatible con el estándar JTAG para su configuración; sin embargo, el inconveniente inmediato a esta posibilidad fue que Actel no proporciona suficiente información técnica relativa a su interfaz de configuración para poder desarrollarla de manera autónoma, lo que implica tener que comprarla a un costo que se iguala en precio a tarjetas de desarrollo comerciales competitivas basadas en FPGA's. Motivo por el cual los FPGA's de Actel fueron descartados. En cuanto a Lattice Semiconductor, los costos de sus dispositivos en general, son altos, por lo que ya no se consideraron.

Dado lo anterior, fue necesario considerar adicionalmente la disponibilidad de información concerniente a la interfaz de configuración para emplear un FPGA de los fabricantes restantes. Encontramos así, que tanto Xilinx como Altera son los fabricantes que tienen disponible en sus sitios WEB la información técnica para sintetizar la interfaz de configuración además de guardar ciertas similitudes entre sus dispositivos.

La decisión final fue inclinarse por emplear un FPGA de Altera, por el hecho haber trabajado anteriormente con CPLD's de este fabricante y en consecuencia estar familiarizadas con sus herramientas de diseño (*software)*, de las cuales, actualmente, hay versiones que ya no requieren de licencia.

En este punto, fue necesario retomar todas las características previamente expuestas (densidad de elementos lógicos, cantidad de funciones de alto nivel embebidas, tecnología de almacenamiento de interconexiones) aplicadas a familias de FPGA's de Altera, además de considerar el tipo de encapsulado en los que están disponibles por la relativa complejidad que pudiera representar la elaboración de su tarjeta de circuito impreso así como su ensamblado y la disponibilidad de herramientas necesarias para tal fin, por lo cual el encapsulado no debe ser de aquellos que requieran de un PCB (*Print Circuit Board*) multicapas o bien de bases especiales difíciles de adquirir y probablemente de costos elevados.

El factor costo no dejó en ningún momento de ser un aspecto importante para descartar los FPGA's de cualquier fabricante, pues la mayoría de las veces se encontró semejanzas en cuanto a las especificaciones técnicas del dispositivo pero diferencias significativas en costos.

En conclusión el dispositivo a emplearse en el diseño es el FPGA **EP2C5T144C8N** con las siguientes características:

Fabricante: Altera.
Familia: *Cyclone II*.
Elementos Lógicos: 4,608.
Funciones de alto nivel embebidas: Bloques de memoria, multiplicadores y PLL's.
Modos de configuración: *JTAG* (volátil), *Pasive Serial* y *Active Serial* (no volátil).
Interfaz de configuración: *ByteBlaster II*.
Encapsulado: TQFP de 144 terminales de montaje superficial y libre de plomo.
Software para compilación, simulación y configuración: Quartus II.
Costo: USD $12.80(al momento de adquirirlo).

Y el HDL a utilizarse es VHDL (VHSIC *Very High Speed Integrated Circuit)*, ya que es un lenguaje jerárquico que aplica plenamente la metodología de diseño descendente (*Top-Down*); permite la declaración de tipos propios de "señales" además de manejar diferentes niveles de abstracción al abordar un diseño, bajo algún estilo de descripción soportado por VHDL tales como: el de transferencia entre registros (RTL), algorítmico o estructura.

CAPÍTULO III: DISEÑO DE LA TARJETA ELECTRÓNICA

Una vez que se hizo la elección del FPGA, el EP2C5T144C8N de la familia *Cyclone II*, de acuerdo a los criterios que se plantearon en el capitulo anterior, el paso siguiente es explicar en el presente capitulo la manera en cómo se lleva a cabo la comunicación entre la PC y el EP2C5T144C8N a través de una interfaz electrónica para el proceso de configuración del FPGA conforme al protocolo establecido por el fabricante; analizar las etapas electrónicas periféricas al EP2C5T144C8N como la de configuración, entradas/salidas, conversión, ingreso y despliegue de datos; y posteriormente proceder a conjuntar todo en lo que será el diseño final de la tarjeta electrónica.

3.1. Modos de configuración soportados por el FPGA seleccionado

Los FPGA's *Cyclone II*, guardan los datos de su configuración en celdas SRAM. Para los FPGA's basados en este tipo de tecnología, SRAM, al momento en que se interrumpe la polarización, el contenido de sus celda se pierde, por lo que es necesario contar con una etapa de memoria no volátil para reconfigurarlos.

En particular, la serie de FPGA's *Cyclone II* soportan tres modos de configuración, *Active Serial (AS), Passive Serial (PS)* o *Joint Test Accion Group (JTAG).* La elección entre un modo de configuración y otro se lleva a cabo conduciendo a un nivel lógico alto o bajo, según sea el caso, dos terminales del FPGA EP2C5T144C8N llamadas MSEL[1..0]. La tabla 3.1 nos muestra los niveles lógicos requeridos para cada modo de configuración.

Modos de configuración	**MSEL 1**	**MSEL 0**
Active serial (20MHz)	0	0
Passive Serial	0	1
Fast Active Serial (40MHz)	1	0
JTAG	1/0	

Tabla 3.1. Modos de configuración soportados por el FPGA *Cyclone II.*

Cabe resaltar que para el modo de configuración *JTAG* las terminales MSEL1 y MSEL0 (ambas), deben conectarse a Vccio o GND.

Las opciones de selección de elementos o dispositivos adicionales para transferir los datos de configuración hacia el FPGA bajo alguno de estos modos y que son compatibles con la familia *Cyclone II* se señalan a continuación en la tabla 3.2.

Modos de configuración	Descripción
Active Serial	Utilizando dispositivos de configuración serie (EPCS1, EPSC4, EPCS16, EPCS64).
Passive Serial	Utilizando dispositivos de configuración avanzados (EPC4, EPC8, EPC16), dispositivos de configuración (EPC1 y EPC2), un microprocesador o una interfaz electrónica.
JTAG	Utilizando una interfaz electrónica a través de Las terminales destinados para el puerto JTAG, o el Estándar *Jam* (*Test and Programming Language STAPL*)

Tabla 3.2. Opciones disponibles para la descarga de datos de configuración de la familia de FPGA's *Cyclone II.*

De acuerdo al modo de configuración a utilizarse, la interfaz electrónica que el fabricante pone a disposición y que es compatible con la familia *Cyclone II* es conocida como *USB Blaster, EthernetBlaster, ByteBlaster MV*o *ByteBlaster II.*

Bajo cualquier de estos modos el proceso de configuración se lleva a cabo en tres etapas: Etapa de *reset,* Etapa de configuración y Etapa de inicio.

- **Etapa de *reset***

  En el momento que el FPGA se polariza, éste pasa a un estado de *Power on Reset*, en el cual sus terminales de E/S se mantienen en alta impedancia por un tiempo de 100ms. Pasado este intervalo de tiempo, las terminales del FPGA denominadas nCONFIG y nSTATUS transitan a un nivel lógico alto, indicando el inicio de la siguiente etapa, la de configuración.

- **Etapa de Configuración**

  Cuando la terminal nSTATUS se encuentra en un nivel lógico alto, el FPGA está listo para recibir los datos de configuración en su terminal DATA al mismo tiempo que genera una señal de reloj (*DCLK*), con un oscilador interno, a una frecuencia de 20MHz o 40MHz. Durante el envío de datos de configuración, el FPGA los retiene en cada flanco positivo de la señal de reloj *DCLK*, y al final de la recepción, hay una transición de un nivel lógico bajo a un nivel lógico alto en la terminal CONF_DONE indicando que la etapa de configuración ha sido completada y que se puede dar comienzo a la siguiente etapa.

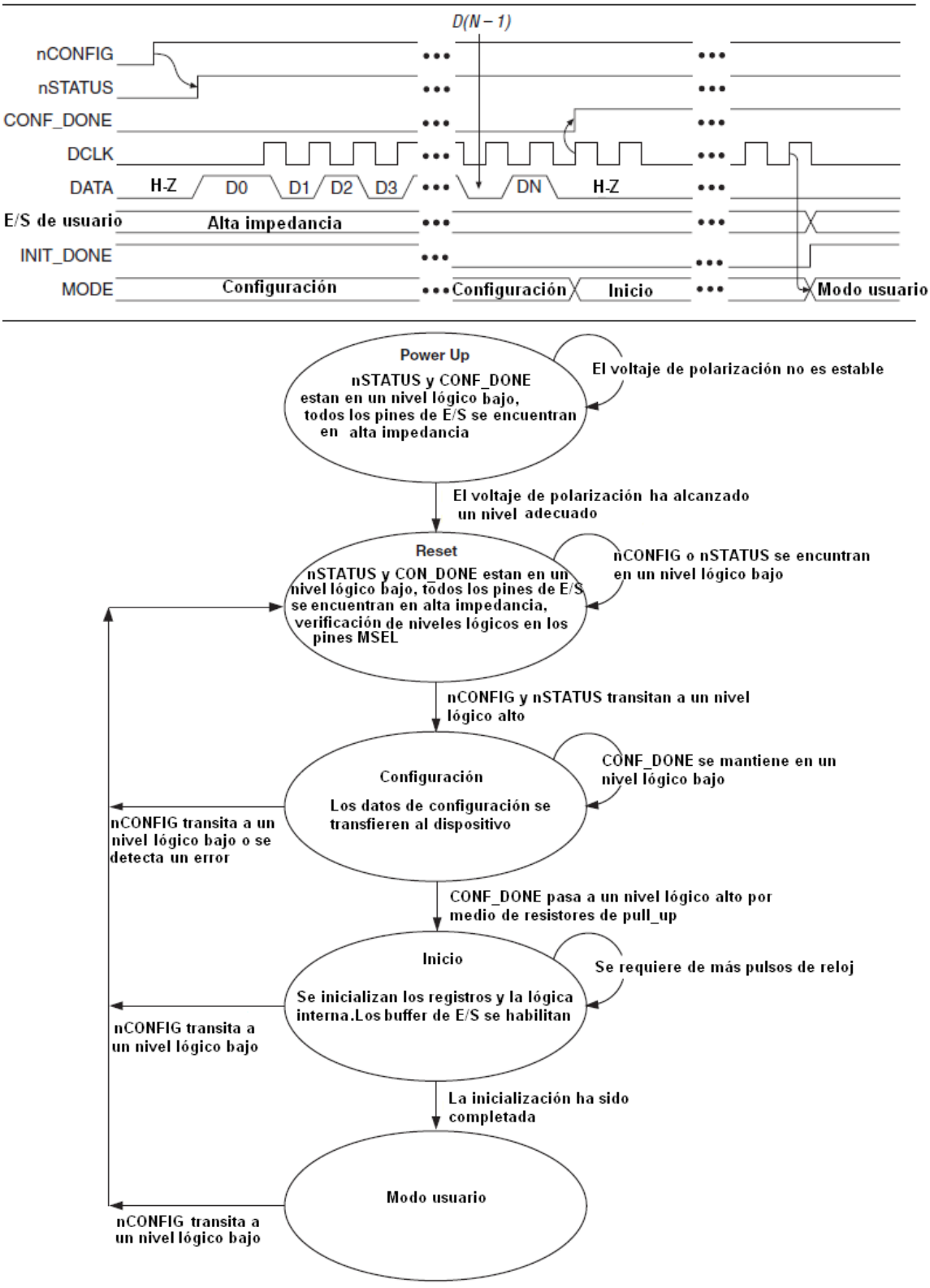


Figura3.1. Máquina de estados y diagrama de tiempos del proceso de configuración.

- **Etapa de inicio**

  En cuanto la terminal CONF_DONE transita a un nivel lógico alto, el FPGA requiere de 299 ciclos de la señal de reloj para inicializar correctamente. En esta etapa, la lógica interna y los registros de E/S son inicializados y los *buffers* de E/S son habilitados. El final de esta etapa la indica una terminal del FPGA denominada INIT_DONE al transitar a un nivel lógico alto. Al final de esta etapa, el FPGA pasa a modo de usuario en el que sus terminales dedicadas como E/S tienen asignada ya una función conforme a la descripción del comportamiento lógico de la arquitectura electrónica digital sintetizada en el dispositivo.

  La figura 3.1 muestra la máquina de estados que resume el proceso de configuración así como el diagrama de tiempos para las señales presentes en este.

3.1.1. Modo *Active Serial*

Los FPAG's *Cyclone II* son configurados utilizando un dispositivo de configuración serie, que es básicamente una memoria de tecnología Flash de 1, 4, 16, 64, o 128 Mbts.

En los dispositivo de configuración serie hay cuatro señales *serial data output (DATA), serial clock input (DCLK), AS data input (ASDI) y active-low chip select (nCS)* que interactúan directamente con las señales de control *DATAO, DCLK, ASDO*, y *nCSO* del FPGA respectivamente. La forma en cómo son conectadas estas cuatro terminales hacia el FPGA *Cyclone II* se muestra en la figura 3.2.

Estos dispositivos de configuración proporcionan una interfaz serie de acceso para los datos de configuración.

El FPGA controla el proceso de configuración, proporcionando los pulsos de reloj al dispositivo de configuración serie, lo habilita poniendo la señal *nCS* a un nivel lógico bajo a través de la señal *nCSO.* En seguida el FPGA envía las instrucciones y las direcciones de las localidades de memoria al dispositivo de configuración serie por medio del pin *ASDO.* El dispositivo de configuración serie responde a las instrucciones enviando los datos de configuración a la terminal *DATAO* del FPGA en un flanco de negativo del la señal de reloj *DCLK.*

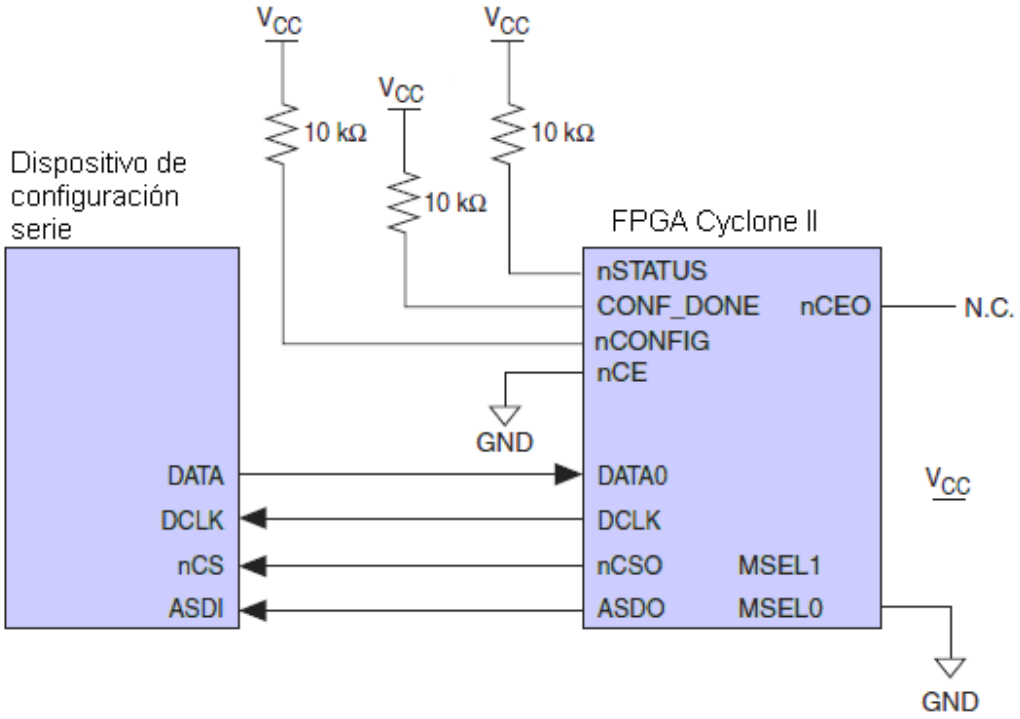


Figura 3.2. Modo de Configuración *Active Serial* con un solo dispositivo de configuración serie.

3.1.2. Modo *Passive Serial*

En este modo se puede hacer uso de un dispositivo de configuración, una interfaz electrónica, un microprocesador o un dispositivo CPLD, los cuales controlan el proceso de configuración proporcionando los datos de configuración desde un dispositivo de almacenamiento (el área de memoria del propio dispositivo de configuración, un disco duro, RAM u otro sistema de memoria) hacia el FPGA *Cyclone II*. En la figura 3.3 se muestra la interconexión del FPGA con un microprocesador para el modo de configuración en cuestión.

Los datos de configuración pueden ser almacenados en un formato RBF, HEX o TTF.

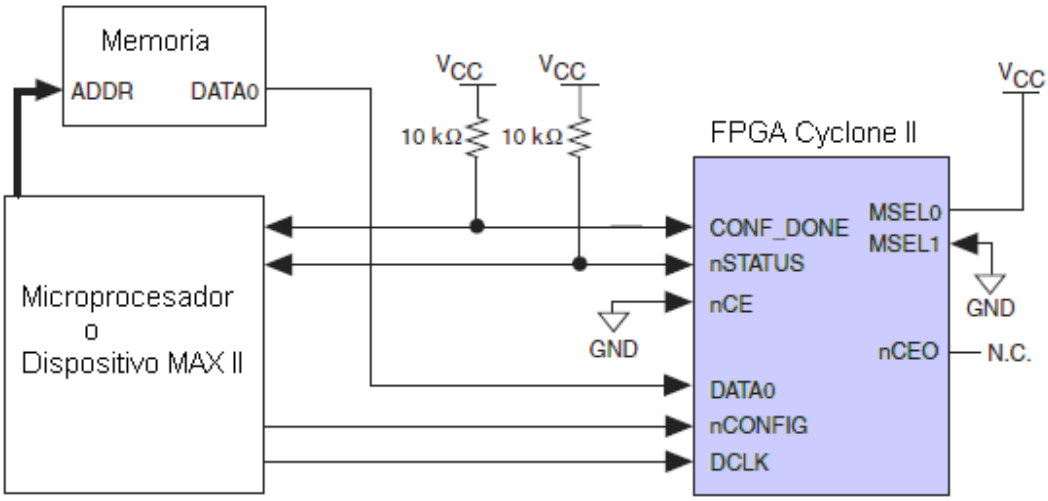


Figura 3.3. Modo de configuración *Passive Serial* utilizando un dispositivo MAXII o un microprocesador.

3.1.3. Modo basado en el estándar *JTAG*

En el modo de configuración *JTAG* cabe mencionar que está basado en el estándar 1149.1 de la IEEE conocido como *JTAG BST (Join Tests Accion Group Boundary Scan Test),* el propósito fundamental de este estándar es proveer un medio para el análisis de operación de circuitos impresos complejos, dispositivos con un difícil acceso a sus terminales y su interconexión hacia otros dispositivos, además de algoritmos de programación para dispositivos reconfigurables por medio de un puerto consistente de cuatro líneas (*TDI, TDO, TMS, TCK*).

Las líneas *TCK (Test Clock Input), TMS (Test Mode Select), TDO (Test Data Output), TDI (Test Data Input)* y una extra que es opcional *TRST (Test Reset Opcional)* forman el puerto de acceso de prueba (*Test Access Port*, *TAP*), por el cual cada función lógica de prueba accede para realizar lo anteriormente dicho y son utilizadas para configurar los *Cyclone II* en modo *JTAG.*

Bajo este modo de configuración se puede hacer uso de una interfaz electrónica. La figura 3.4 muestra la configuración en modo *JTAG* de un solo dispositivo *Cycllone II* utilizando la interfaz electrónica.

Al final de la configuración el *software* checa el estado lógico de la terminal *CONF_DONE* a través del puerto *JTAG*; si dicha terminal está en un estado lógico bajo, el *software* de programación indica que la configuración ha fallado, en caso contrario, es decir, *CONF_DONE* se encuentra en un nivel lógico alto, se indica que la configuración fue exitosa.

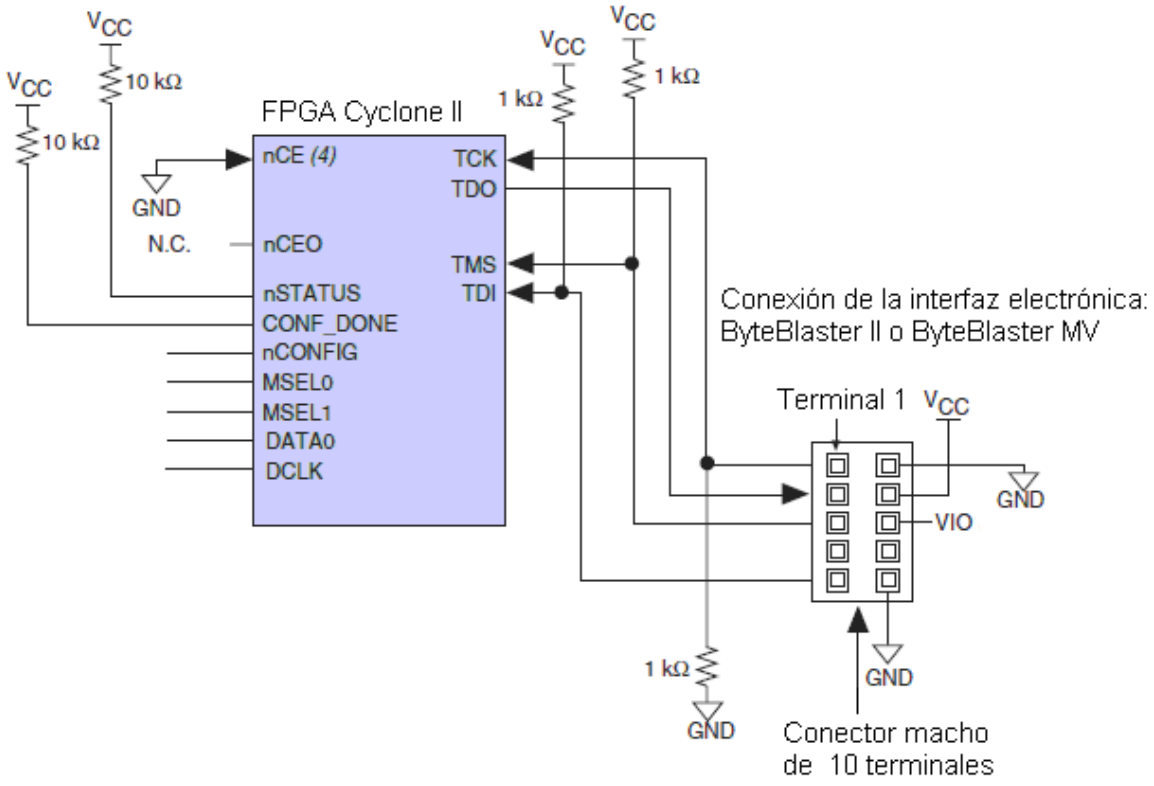


Figura 3.4. Configuración en modo *JTAG* de un solo *Cyclone II*.

Es importante señalar que los dispositivos *Cyclone II* están diseñados de tal manera que las instrucciones *JTAG* toman prioridad sobre cualquier otro modo de configuración. Esto quiere decir, que si se intenta configurar al FPGA, por ejemplo, en modo *Passive Serial* y en modo *JTAG* simultáneamente, la configuración *JTAG* toma lugar sin esperar a que el otro modo de configuración se complete.

Adicionalmente la serie de FPGA's *Cyclone II* tienen la capacidad de soportar la combinación de modos de configuración tales como *Passive Serial – JTAG* o *Active Serial- JTAG*.
Las figuras 3.5 y 3.6 muestran el diagrama de conexiones para la combinación de estos modos de configuración.

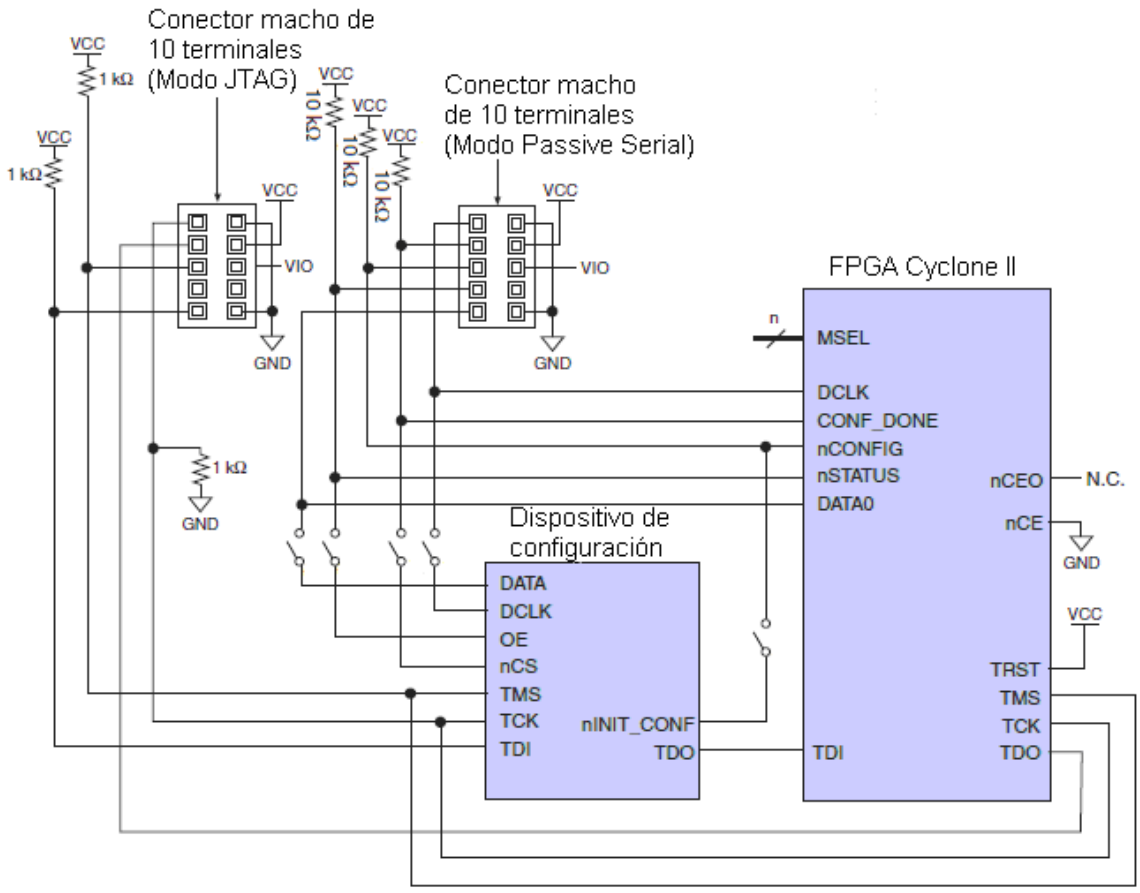


Figura 3.5. Combinación del modo de configuración *Passive Serial* con *JTAG* a través de una interfaz electrónica.

La figura 3.5 muestra el diagrama de conexiones para llevar a cabo la configuración en modo *Passive Serial* o *JTAG* mediante una interfaz electrónica. Se observa que hay dos conectores hembra de 10 terminales los cuales corresponden a cada modo.
En modo *JTAG* se programa al FPGA y al dispositivo de configuración, en *Passive Serial* este último transfiere los datos de configuración hacia el FPGA.

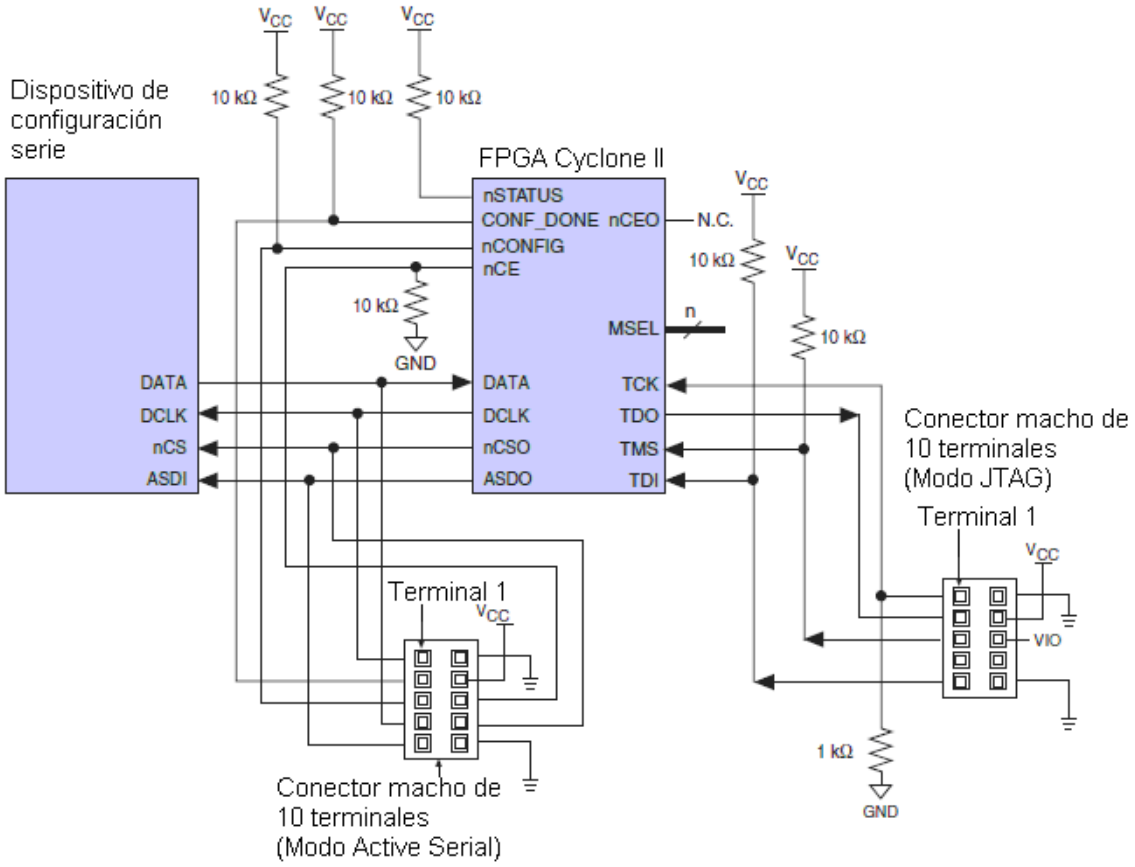


Figura 3.6. Combinación del modo de configuración *Active Serial* con *JTAG* a través de una interfaz electrónica.

Al igual que en la combinación de esquemas de configuración *Passive Serial – JTAG*, en la *Active Serial – JTAG* se requiere de dos conectores hembra de 10 terminales.
En modo *JTAG* se configura directamente al FPGA, en tanto que en *Active Serial* se lleva a cabo por medio del dispositivo de configuración serie, en sistema, vía la interfaz que proporciona este mismo.

En la explicación de cada modo de configuración se ha venido mencionando que la interfaz electrónica es una opción para transferir los datos de configuración hacia el FPGA. Actualmente el fabricante Altera pone a disposición del usuario cuatro de ellas, comercialmente conocidas como *USB Blaster, EthernetBlaster, ByteBlaster MV* y *ByteBlaster II,* las cuales conducen los datos de configuración de un puerto paralelo, un puerto universal serie (*USB*) o un puerto *Ethernet* de una PC hacia el FPGA que se encuentra montado en una placa de circuito impreso tal como se ilustra en la figura 3.7. De estas cuatro opciones se seleccionó la Interfaz *ByteBlaster II* por ser compatible con los modos de configuración soportados por el EP2C5T144C8N y porque se puede fabricar de manera autónoma, ya que la información tecnica para ello se encuentra disponible en el sitio WEB del fabricante, en consecuencia se considera importante el hecho de abordar su síntesis.

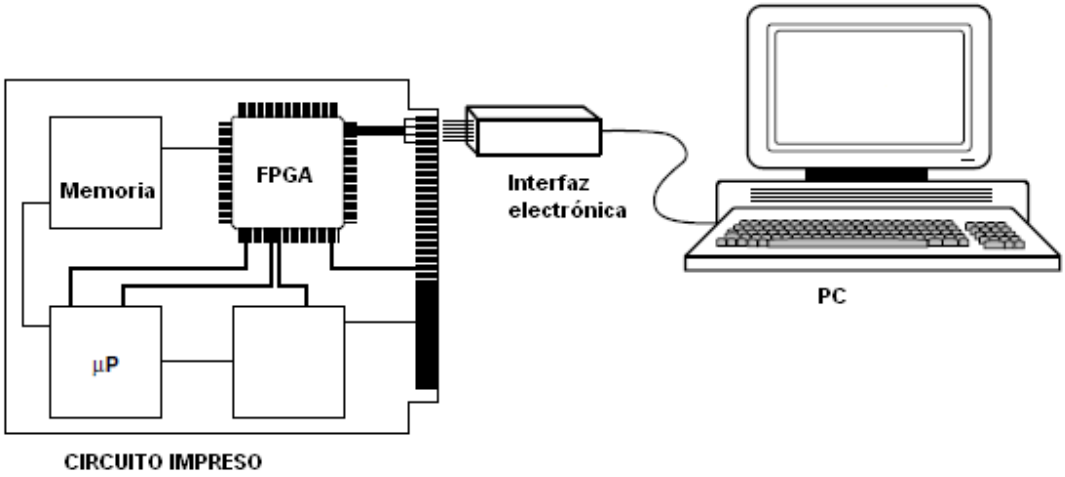


Figura 3.7. Comunicación entre la PC y el FPGA vía la interfaz electrónica.

Interfaz *ByteBlaster II*

La interfaz *ByteBlaster II* consiste de un conector DB25 macho, un conector hembra de 10 terminales y el circuito electrónico que transfiere los datos de configuración del puerto paralelo estándar de una PC hacia el FPGA. Dicho circuito electrónico consiste básicamente de *buffers* 3 estados no inversores orientado para un uso de transmisión/ recepción de datos, su diagrama eléctrico es mostrado en la figura 3.8.

A la conexión de cada *buffer* con su correspondiente terminal del conector DB25 o el conector hembra se les asigna una señal dependiendo del modo de configuración que se vaya a utilizar, las tablas 3.3 y 3.4 indican con exactitud qué señales van hacia qué terminales de ambos conectores

Adicionalmente la interfaz *ByteBlaster II* puede configurar en sistema dispositivos de configuración serie (EPCS1, EPCS4, EPCS16, EPCS64, EPCS128) y dispositivos de configuración avanzados (EPC2, EPC4, EPC8, EPC16, EPC1441).

Terminal	Modo Active serial		Modo Passive serial		Modo JTAG	
	Nombre de la señal	Descripción	Nombre de la señal	Descripción	Nombre de la señal	Descripción
2	DCLK	Clock signal	DCLK	Clock signal	TCK	Clock signal
3	nCONFIG	Configuration control	nCONFIG	Configuration control	TMS	JTAG state machine control
4	nCS	Serial configuration device chip select	-	No connect	-	No connect
5	nCE	Cyclone Chip Enable	-	No connect	-	No connect
8	ASDI	Active serial data in	DATA O	Data to device	TDI	Data to device
11	CONF_DONE	Configuration done	CONF_DONE	Configuration done	TDO	Data from device
13	DATAOUT	Active serial data out	nSTATUS	Signal status	-	No connect
15	nVCC Detect	-	nVCC Detect	-	nVCC Detect	-
18 a 25	GND	Signal ground	GND	Signal ground	GND	Signal ground

Tabla 3.3. Asignación de las terminales de salida del conector DB25.

Terminal	Modo Active serial		Modo Passive serial		Modo JTAG	
	Nombre de la señal	Descripción	Nombre de la señal	Descripción	Nombre de la señal	Descripción
1	DCLK	Clock signal	DCLK	Clock signal	TCK	Clock signal
2	GND	Signal ground	GND	Signal ground	GND	Signal ground
3	CONF_DONE	Configuration done	CONF_DONE	Configuration done	TDO	Data from device
4	VCC(TRGT)	Target power suplí	VCC(TRGT)	Target power supply	VCC(TRGT)	Target power supply
5	nCONFIG	Configuration control	nCONFIG	Configuration control	TMS	JTAG state machine control
6	nCE	Cyclone chip enable	-	No connect	-	No connect
7	DATAOUT	Active serial data out	nSTATUS	Configuration status	-	No connect
8	nCS	Serial configuration device chip select	-	No connect	-	No connect
9	ASDI	Active serial data in	DATAO	Data to device	TDI	Data to device
10	GND	Signal ground	GND	Signal ground	GND	Signal ground

Tabla 3.4. Asignación de las terminales de salida del conector hembra de 10 pines.

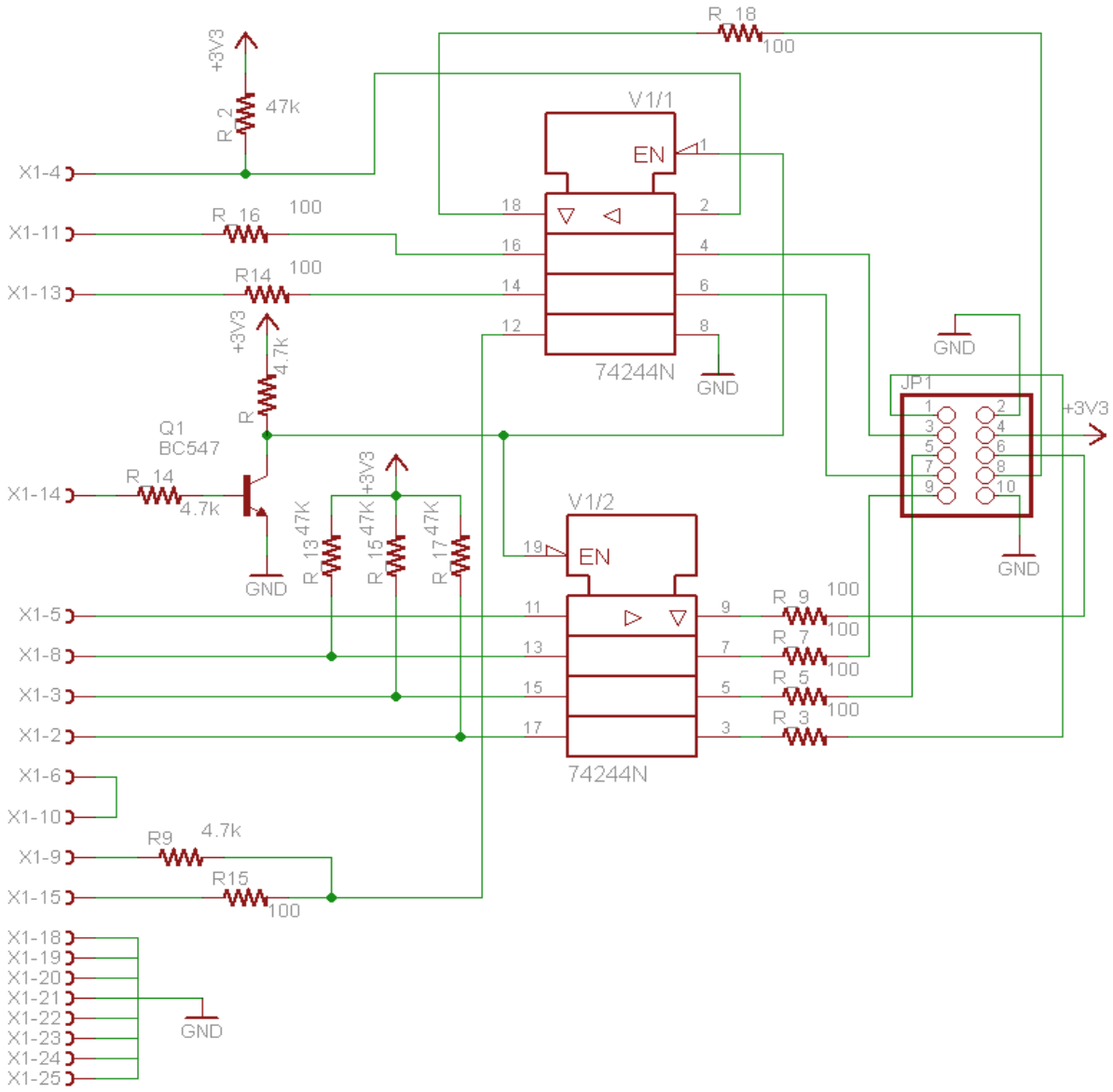


Figura 3.8 .Diagrama eléctrico de la interfaz *ByteBlaster II*.

3.2. Elección de las etapas electrónicas periféricas al FPGA

La implementación de las etapas periféricas está pensada con base a los diseños que los estudiantes en electrónica de la facultad de Ingeniería han elaborado en sus cursos de diseño digital y electrónica digital.

En sus diseños, frecuentemente las salidas son visualizadas por medio de LED's y/o displays de siete segmentos y como entradas se recurre a interruptores deslizables o de contacto momentáneo,

de manera adicional son utilizados los teclados matriciales y displays de cristal líquido si la aplicación es un poco mas elaborada.

Pero las entradas no solo se limitan a ser tomadas de interruptores, sino también del medio que nos rodea, sean de naturaleza analógica o digital, por ende implica cierta dificultad para manipularlas, ya que previamente necesitan de un acondicionamiento para su manejo en etapas posteriores de un diseño, es decir, se puede recurrir a un proceso de conversión ya sea una conversión analógica-digital a viceversa digital –analógica.

Contemplando principalmente estas particularidades, se considera necesario que sean parte indispensable de las etapas periféricas que conformarán la tarjeta electrónica bosquejada a manera de un diagrama a bloques en la figura 3. 9.

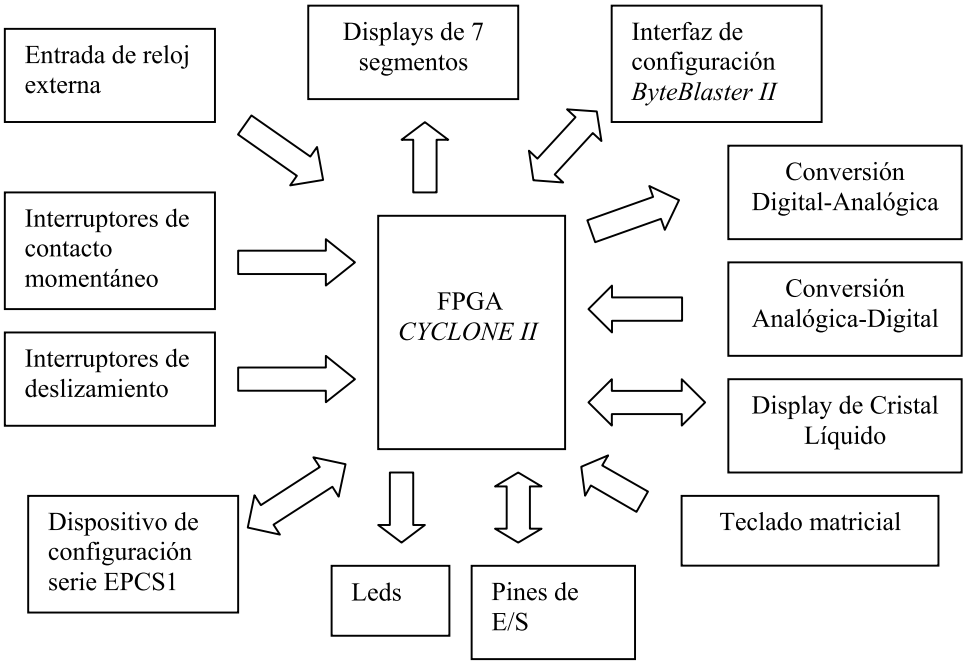


Figura 3.9. Etapas periféricas al *FPGA Cyclone II.*

De la cual sus características principales se mencionan en seguida.

FPGA EP2C5T144C8N de la familia Cyclone II

- 4608 elementos lógicos.
- 26 bloques de 4kbites de memoria RAM.
- 13 multiplicadores embebidos.
- 2 PLL's.
- Encapsulado TQPF de 144 terminales.

Entrada de reloj

- Oscilador con una frecuencia de salida de hasta 4MHz.

Ingreso de entradas

- Cuatro interruptores de contacto momentáneo. Normalmente se encuentran en un nivel lógico bajo, pero cuando se presionan se genera un nivel alto lógico.
- Cuatro interruptores deslizables. Se presenta un nivel lógico bajo cuado el interruptor está abierto, y un nivel lógico alto cuando está cerrado.
- Teclado matricial de 4 columnas por 4 renglones.

Dispositivo de configuración Serie EPCS1 e interfaz de configuración *ByteBlaster II*

- Soporte para tres modos de configuración, *Active Serial*, *Passive Serial* o *JTAG*.

Terminales de E/S

- 48 terminales de E/S disponible.

Visualización de salidas

- Ocho diodos emisores de luz.
- Dos displays de 7 segmentos de cátodo común.
- Display de cristal líquido de 1 línea de 8 caracteres basado en el controlador SED1278F.

Conversión de datos

- Convertidor analógico-digital, basado en el circuito integrado ADC0802.
- Convertidor digital -analógico, basado en el circuito integrado DAC0800.

3.3. Diseño de etapas electrónicas periféricas

3.3.1. Etapa de configuración del FPGA

Los *Cyclone II* pueden ser configurados en un modo *Active Serial*, *Passive Serial* o *JTAG* o bien utilizar la combinación *Passive Serial-JTAG* o *Active Serial- JTAG*. La ventaja de usar un modo de configuración combinado es que ya se cuenta con una etapa de memoria para el almacenamiento de los datos de configuración, es por eso que la etapa de configuración del EP2C5T144C8N se hará con un modo de configuración combinado, específicamente por *Active Serial- JTAG*, puesto que los dispositivos de configuración serie ,EPCS1, representan una opción de configuración de bajo costo en comparación con los dispositivos de configuración requeridos para una modo *Passive Serial-JTAG*.

Basándonos en el diagrama eléctrico de conexiones de la figura 3.6 para el modo *Active Serial-JTAG,* el dispositivo de configuración serie utilizado es un EPCS1 que es una memoria de tecnología flash de 1,048576 bits, el cual se encuentra disponible en un encapsulado SOIC de 8 terminales, y requiere de un voltaje de polarización de 3.3 V. La figura 3.10 muestra las terminales de este dispositivo

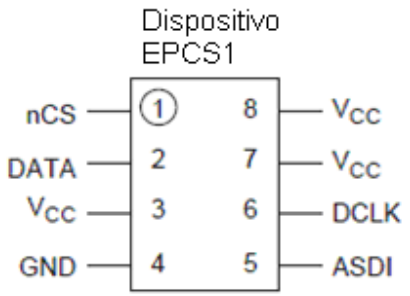


Figura 3.10. Diagrama de las terminales del dispositivo de configuración serie EPCS1.

3.3.2. Etapa de entradas y salidas básicas al FPGA (Interruptores deslizables, interruptores de contacto momentáneo, *LED's*, *display*s de siete segmentos)

Como entradas básicas disponemos de cuatro interruptores deslizables y cuatro interruptores de contacto momentáneo, los cuales se encuentran inicialmente en un nivel lógico bajo y proporcionan un nivel alto lógico de 3.3V al cerrarse o presionarse respectivamente.

Las salidas se pueden apreciar en los ocho LED's y en dos displays de siete segmentos de cátodo común, un nivel lógico alto en las terminales del FPGA donde se encuentran conectados los LED's provocará que estos se enciendan, para los segmentos de los displays es el mismo caso. La figura 3.11 muestra las conexiones tanto de las entradas como de las salidas básicas hacia las terminales del FPGA.

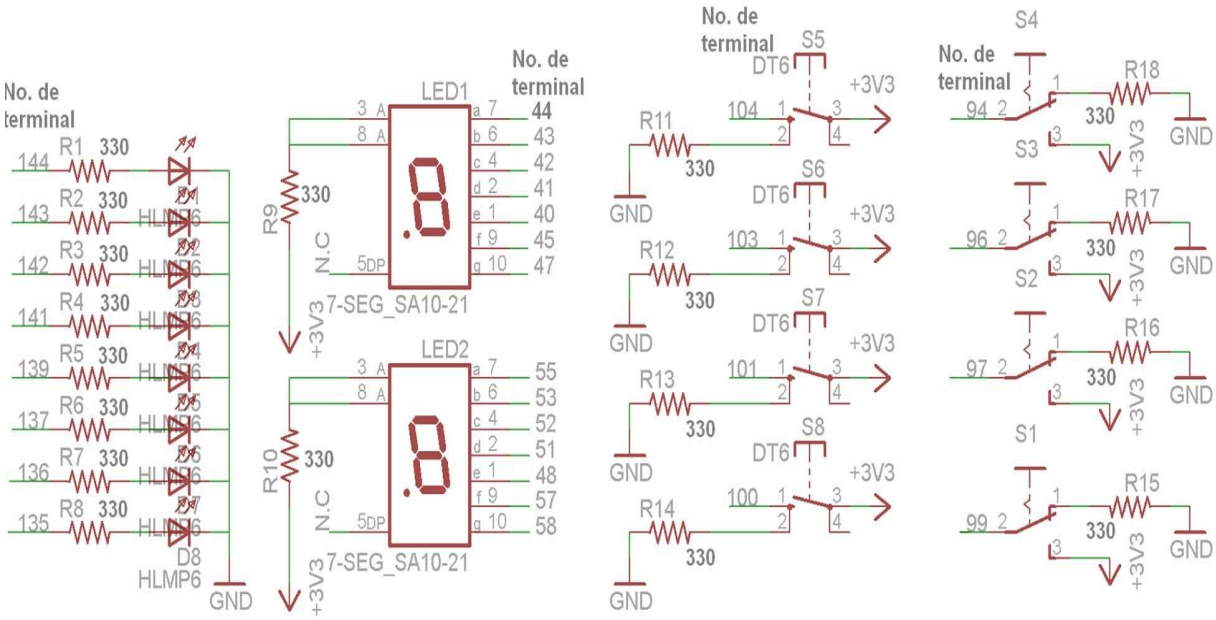


Figura 3.11. Conexión hacia las terminales del EP2C5T144C8N de entradas y salidas básicas.

3.3.3. Etapa de conversión analógica-digital y digital-analógica

El circuito electrónico para el proceso de conversión analógica-digital se baso en el circuito integrado ADC0802, el cual es un convertidor A/D de 8 bits basado en el principio de aproximaciones sucesivas. Las características principales de este integrado son sus entradas analógicas diferenciales que permiten incrementar el rechazo en modo común así como el reconocimiento de entradas y salidas MOS y TTL; puede producir una señal de reloj con un generador que tiene internamente o bien se cuenta con la opción de proveer una señal de reloj externa; opera a un voltaje de 5V, permitiendo así en sus entradas una señal analógica en un rango de 0-5V, trabaja con un voltaje de referencia de 2.5V que además puede ajustarse para permitir codificar señales analógicas de amplitudes pequeñas.

La manera en cómo se operará el ADC0802 es bajo un modo de funcionamiento denominado *free running*, lo que significa que el circuito integrado estará convirtiendo continuamente. En modo *free running* las señales de control, /INTR, /WR y /CS que provee el dispositivo deben conectarse como lo indica la figura 3.12.

Además, nótese que en esta figura, hay un condensador y un resistor con valores establecidos para obtener una frecuencia de operación de 640KHz, la cual recomienda el fabricante, para garantizar una mayor exactitud del proceso de conversión.
De acuerdo a las especificaciones del fabricante el ADC0802 entrega un voltaje de salida mínimo de 4.5V para nivel lógico alto, y el EP2C5T144C8N acepta un voltaje de entrada máximo de 4V lo que significa que no podrá haber interacción directamente si se requiere en algún momento manipular las salidas del ADC0802 para cierta aplicación; dado este inconveniente, como una posible solución a esta incompatibilidad se planteó utilizar un circuito convertidor de niveles lógicos de tensión, específicamente el MC14504B, el cual permite cambiar una señal TTL a niveles lógico CMOS o entre niveles CMOS diferentes por medio de una entrada de control (MODE).

El cambio de un nivel alto se logra seleccionando los niveles de tensión en las terminales VCC y VDD del MC14504B operando en un rango de 3 a 18 V. El nivel de tensión aplicado a la terminal VCC es el que se requiere convertir al nivel de tensión suministrado en la terminal VDD, tal como se muestra en la figura 3.13.

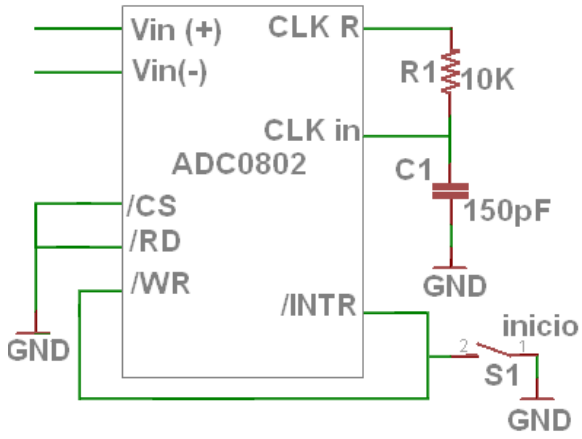
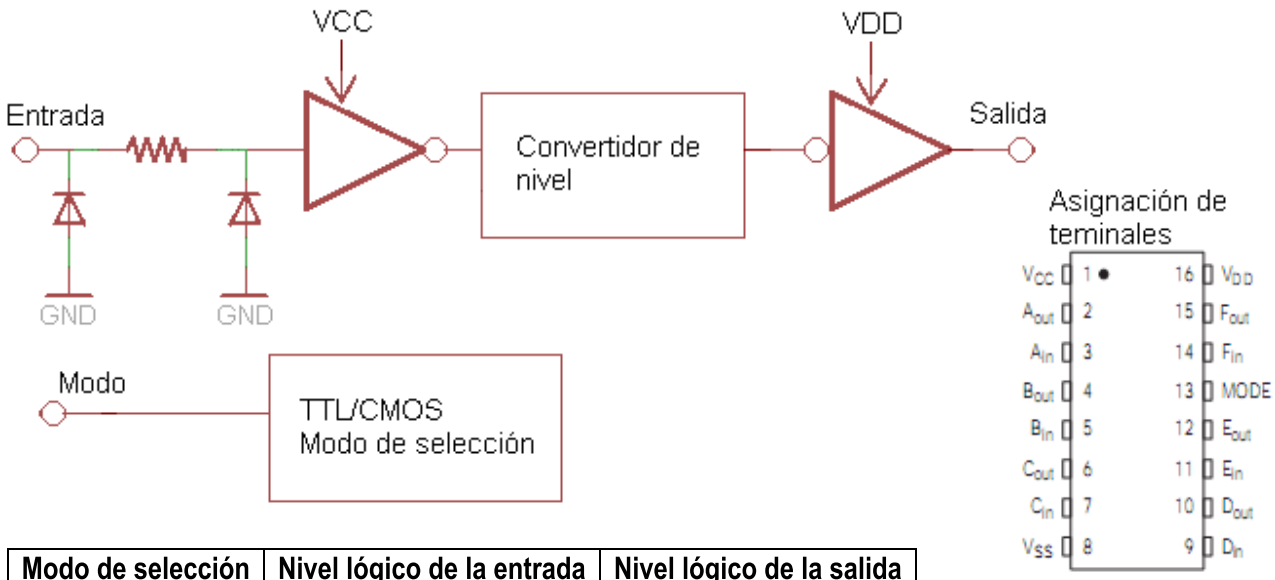


Figura 3.12. ADC0802 bajo modo *free running.*



Modo de selección	Nivel lógico de la entrada	Nivel lógico de la salida
1 (VCC)	TTL	CMOS
0 (VSS)	CMOS	CMOS

Figura 3.13. Diagrama lógico del MC14504B.

Con esta adecuación que se hizo para la etapa de conversión A/D, su diagrama eléctrico final se muestra en la figura 3.14.

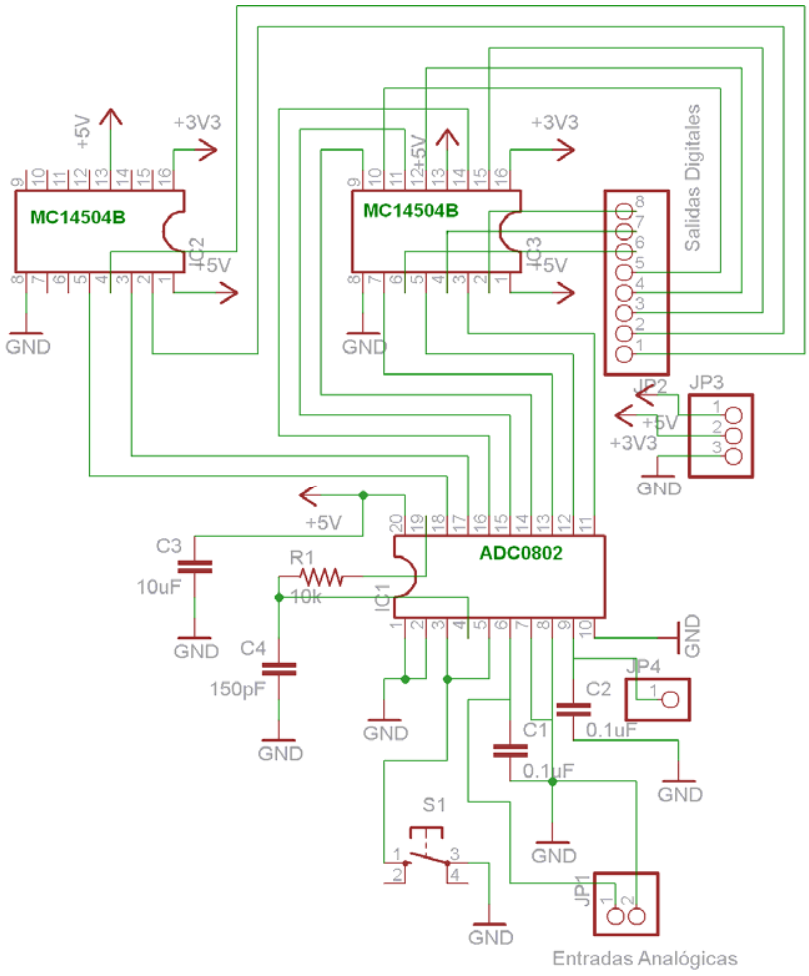


Figura 3.14. Diagrama eléctrico de la etapa de conversión A/D.

Para el proceso inverso de conversión se empleo el circuito integrado DAC0800 el cual es un convertidor D/A de 8 bits con salidas complementarias de corriente que pueden alcanzar una amplitud de hasta 20Vpp al colocarse una resistencia de carga en las salidas del integrado.

La inmunidad al ruido en las entradas del DAC0800 permite manejar niveles TTL o bien interactuar con otras familias lógicas, simplemente manejando cierto potencial en la terminal VLC del integrado.

Las características y el desempeño del dispositivo no cambian si se opera en un rango de +/-4.5 a +/-18V. Ofrece un bajo consumo de potencia cuando se manejan +/-5V de polarización, disipando hasta 33mW.

El diagrama eléctrico para esta etapa se basó en las aplicaciones típicas que el fabricante sugiere para este integrado, mostrado en la figura 3.15, con la variante de que el voltaje de polarización será

de +/-5V y en consecuencia la salida será de una amplitud de 5Vpp, con opción a reducirla o ampliarla mediante un resistor variable, disponible en el circuito electrónico del convertidor.

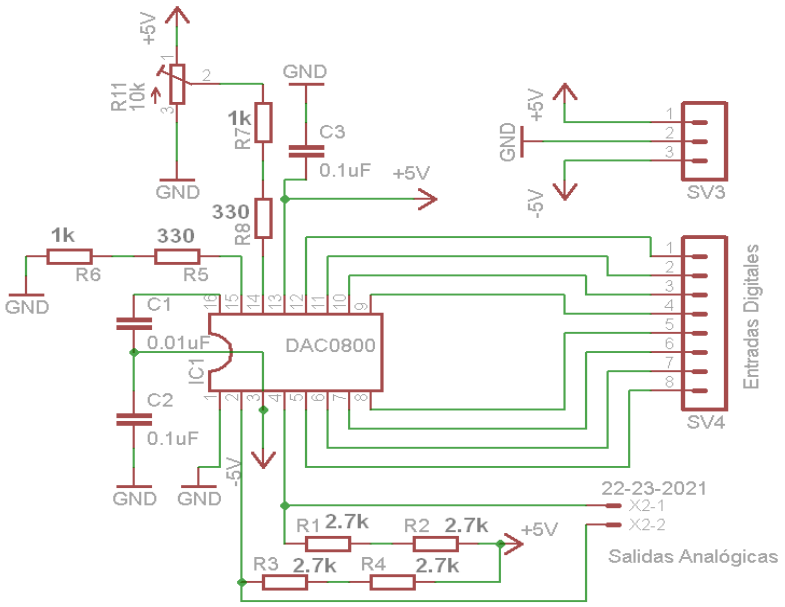


Figura3.15. Diagrama eléctrico de la etapa de conversión D/A.

3.3.4. Etapa de ingreso y despliegue de datos (Teclado matricial y *display* de cristal líquido (LCD))

Una opción adicional a las entradas básicas como interruptores para el ingreso de datos son los teclados matriciales, de los cuales, entre los más comunes podemos encontrar los que son de 3x4 o de 4x4 (renglones x columnas).

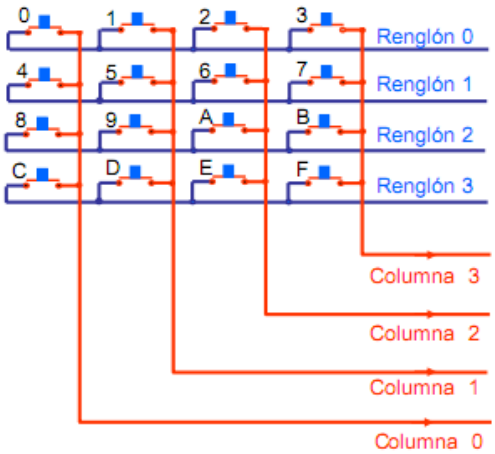


Figura 3.16. Teclado matricial de 4x4.

Los teclados matriciales son arreglos de interruptores de contacto momentáneo en forma de matriz, tal como se ilustra en la figura 3.16; cada interruptor se conecta a una columna y a un renglón de modo que al oprimirse una tecla estos hacen conexión.

Los teclados matriciales comerciales pueden traer consigo decodificadores para identificar las teclas que se presionan, pero en este caso la etapa de ingreso de datos no lo contendrá ya que se puede sintetizar en el mismo FPGA.
Se aclara que el teclado utilizado será uno de 4x4.

Así mismo para las salidas, cuando haya la necesidad de desplegar caracteres alfanuméricos, un display de cristal líquido es una opción muy adecuada. Los displays de cristal líquido se diferencian entre ellos por la cantidad de caracteres que puedan desplegar, los hay de 1 x 8 o de 2 x 16 (líneas de caracteres), a pesar de esto en su mayoría son estándar y se controlan de formas semejantes.

Anteriormente mencionamos que se empleará un LCD basado en el controlador SED1278F. Este LCD tiene un conector de 14 terminales, de las cuales 8 de ellas son un bus de datos de E/S que se puede manejar a 8 o 4 bits; 3 son señales de control, destinadas para la habilitación del LCD, la indicación de una operación de lectura o escritura y la selección del registro sobre el cual se va a leer o escribir; una es para manipular el contraste de la pantalla y las terminales restantes son para la polarización del LCD, que opera a una tensión de 5V. En la tabla 3.5 listamos estas terminales así como su función.

Terminal	Nombre	Función
1	VSS	GND
2	VDD	Alimentación
3	VO	Contraste
4	RS	Selección del registro
4	R/W	Operación de lectura/escritura
6	E	Habilitación del LCD
7	D0	Dato0
8	D1	Dato1
9	D2	Dato2
10	D3	Dato3
11	D4	Dato4
12	D5	Dato5
13	D6	Dato6
14	D7	Dato7 o bandera de “ocupado”

Tabla 3.5. Terminales del display de cristal líquido basado en el controlador SED1278F.

El problema de incompatibilidad que existe entre el EP2C5T144C8N y el ADC0802, explicado en párrafos anteriores, lo hay también entre el LCD y el FPGA, sin embargo la solución empleada fue exactamente la misma, se utilizaron convertidores de niveles lógicos de tensión para llevar a cabo la interacción entre estos dispositivos. En la figura 3.17 se puede ver el diagrama eléctrico para esta etapa.

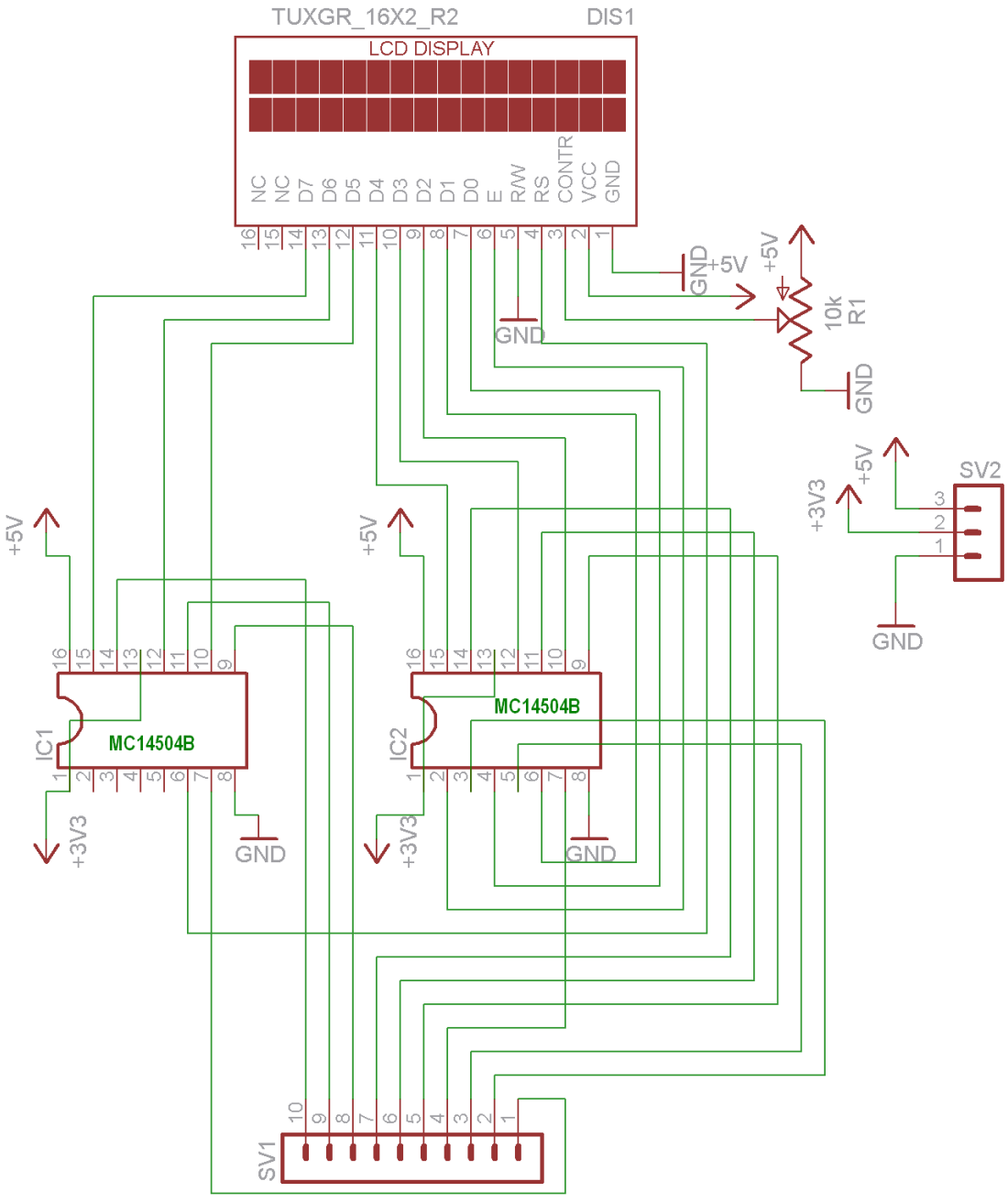


Figura 3.17. Etapa de despliegue mediante un display de cristal líquido.

3.4. Distribución e integración de todas las etapas diseñadas

La integración parcial o total de cada una de las etapas periféricas que se han descrito estará determinada únicamente por la complejidad que resulte de elaborar el circuito impreso.

Tratamos de evitar el uso de un circuito impreso multicapas, por los inconvenientes que conlleva; se pretende abordarlo de manera que se adjunten la mayor parte de las etapas en una sola cara de una placa fenólica, y aquellas que no se puedan incorporar a la tarjeta electrónica manejarlas por separado a manera de pequeñas tarjetas de expansión.

Durante la elaboración del circuito impreso las etapas que tomaron prioridad fueron principalmente la etapa de configuración, E/S básicas, además de la circuitería adicional para el manejo de los diferentes voltajes para la alimentación en general de los dispositivos, el oscilador externo y los pines E/S libres para el usuario, por lo que las etapas restantes se harán en circuitos impresos por separado acondicionándoles un conector macho de cierto numero pines, según lo requiera la etapa, para que cuando se necesite de su uso se incorpore a la tarjeta conectándolo al conector hembra que habrá en esta.

La figura 3.18 muestra las principales etapas integradas a la tarjeta, nótese la presencia de los conectores hembra para conectar las etapas restantes.

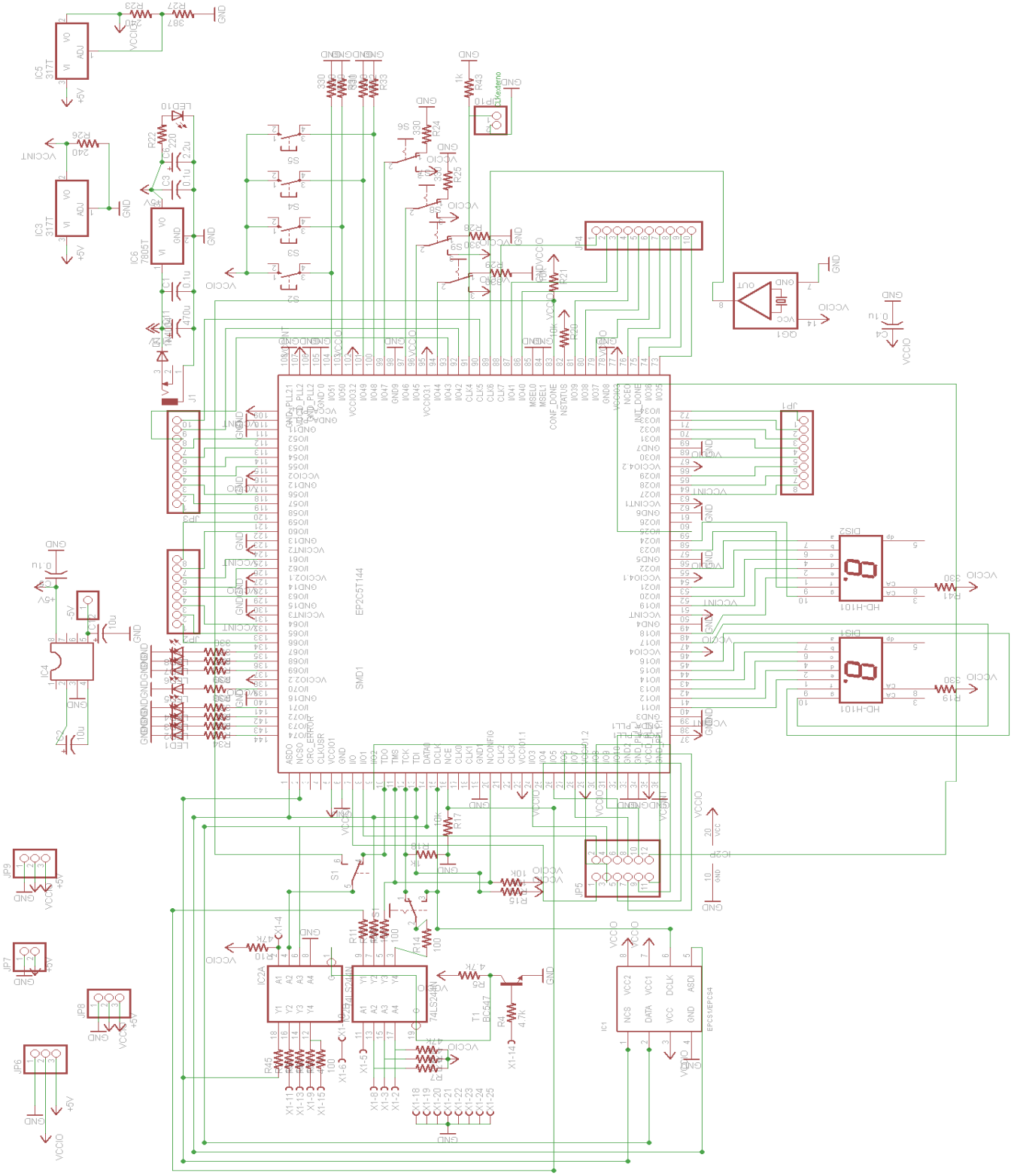


Figura 3.18. Esquema eléctrico de la tarjeta electrónica principal.

CAPÍTULO IV: IMPLEMENTACIÓN DE LA TARJETA ELECTRÓNICA

Al final del capítulo anterior se mencionó cuales fueron las etapas electrónicas periféricas al FPGA prioritarias que estarán presentes en el circuito impreso principal así como las que se manejaran en tarjetas de circuito impreso por separado (tarjetas de expansión).

Bajo esas consideraciones, lo siguiente es abordar el proceso de fabricación del circuito impreso que contendrá el FPGA, donde se explicará la técnica aplicada incluyendo la que se empleo en el soldado del EPC2C5T114C8N finalizando con la explicación de las pruebas aplicadas al sistema de tarjetas electrónicas, así como el análisis de resultados que nos lleven a la exposición de las observaciones y conclusiones correspondientes.
.

4.1. Fabricación del circuito impreso

Con la automatización de los proceso de diseño y fabricación de circuitos electrónicos hoy en día se dispone de una variedad de herramientas de Diseño Asistido por Computadora (CAD) que nos permiten abordar los diseños de circuitos impresos de una manera relativamente sencilla. Los paquetes de *software* que constituyen estas herramientas incluyen la captura de esquemas eléctricos a partir de los cuales se pueden generar circuitos impresos. Una de estas herramientas es el *software* para la edición de circuitos impresos llamado *Eagle (Easily Apliccable Graphical Layaout Editor)*, el cual consta de tres módulos integrados, un editor de capas, un editor de esquemas y un Autorouter.

El *software* viene con una variedad de librerías para diversos componentes con la posibilidad de crear nuevas librerías y modificar las ya existentes.
Con esta herramienta se abordó el diseño de la tarjeta electrónica principal, el cual se puede observar en las figura 4.1, así como el resto de las etapas periféricas, mostradas en las figuras 4.2 y 4.3.

Siendo coherentes con el objetivo de lograr que un alumno pueda desarrollar un sistema electrónico que utilice tecnología y dispositivos del tipo aquí seleccionado (FPGA de encapsulado tipo TQFP de 144 pines), no solamente se limitó a abordar el aspecto del diseño de la tarjeta de circuito impreso, sino que se abordó la fabricación de la misma empleando para su elaboración productos al alcance de cualquier persona, refinando la técnica que aunque de carácter artesanal ofreció resultados satisfactorios superando la complejidad que representa en primer lugar las dimensiones de pistas y en segundo el soldado de dispositivos, siendo el FPGA el que representaba en primera instancia el mayor reto debido a sus dimensiones. Por lo tanto se justifica el exponer la técnica aplicada y que complementa el diseño de la tarjeta.

La técnica empleada para la fabricación de la tarjeta de circuito impreso (PCB), consiste básicamente en dos fases:

1. Impresión y transferencia del diseño.
    a. Se imprimió el negativo del diseño en papel *couche* por medio de impresión láser.
    b. Se transfirió el diseño impreso en el papel a la placa fenólica de una cara (previamente limpiada de todo exceso de grasa) por medio de aplicación de calor (planchado). La técnica se fue refinando a prueba y error para establecer el nivel de temperatura y presión adecuadas para una correcta transferencia.
    c. Se retira el papel previamente humedecido resultando la transferencia del tóner (diseño PCB) a la placa fenólica.

2. Desbaste de cobre y barrenado.
    a. Se sumerge la tarjeta en la sustancia o mezcla química corrosiva (comúnmente cloruro férrico y agua) que retire las áreas de cobre que no están cubiertas por tóner.
    b. Finalmente se barrena en aquellos puntos en donde se requiera.

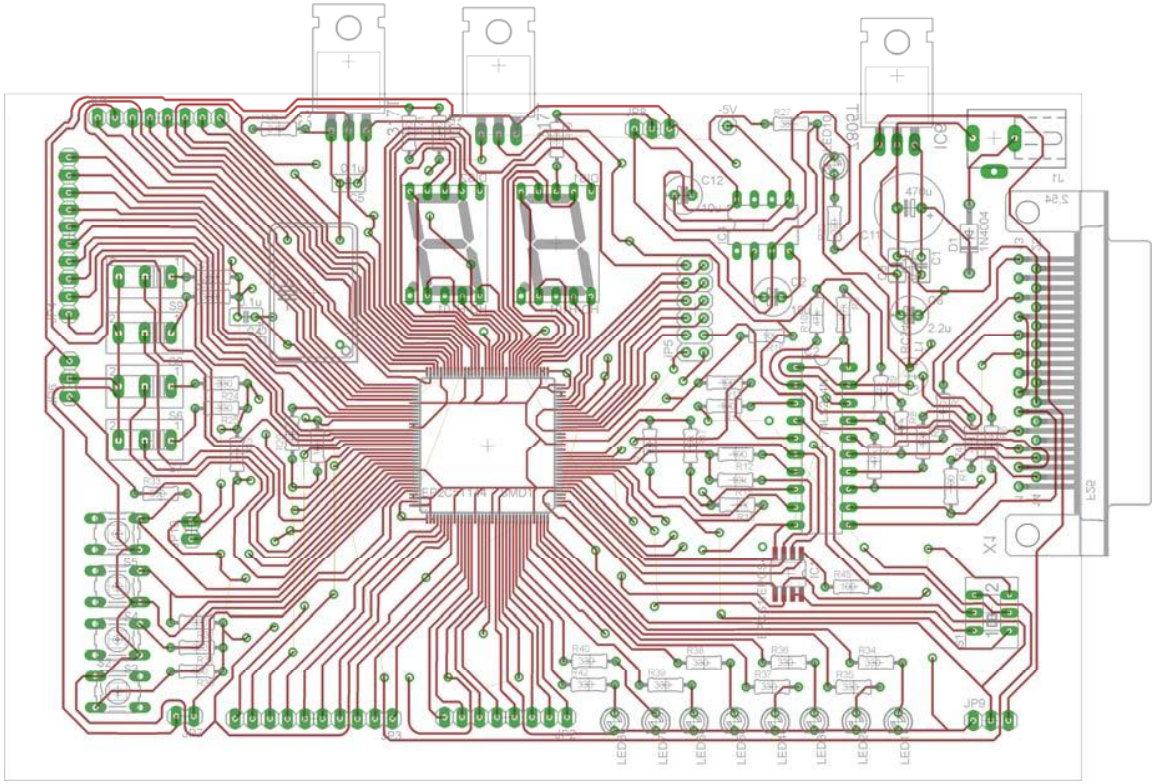


Figura 4.1. Circuito impreso generado en EAGLE PCB de la tarjeta electrónica principal.

La técnica aplicada para el soldado de los dispositivos de encapsulado de montaje superficial (el caso del FPGA del tipo TQFP de 144 terminales) consistió en:

1. Estañar previamente y de forma superficial las zonas de contacto de las terminales del circuito, denominados comúnmente como *pad's*, Se puede utilizar malla para desoldar para dejar una superficie de soldadura delgada y homogénea en los *pad's.*

2. Se coloca el FPGA sobre los *pad's* de soldado centrándolo lo mejor posible.

3. Se aplica el denominado flux (limpiador espumoso para soldar) de forma abundante para que de este modo se evite en lo posible uniones entre terminales con la soldadura.

4. Se procede a soldar al menos las terminales que se encuentran en los extremos de una cara del dispositivo para permitir corregir el centrado del mismo.

5. Si el centrado es correcto, se procede a pasar la punta del cautín (que dispone de muy poca soldadura) sobre todas las terminales de cada lado del dispositivo en movimientos rápidos.

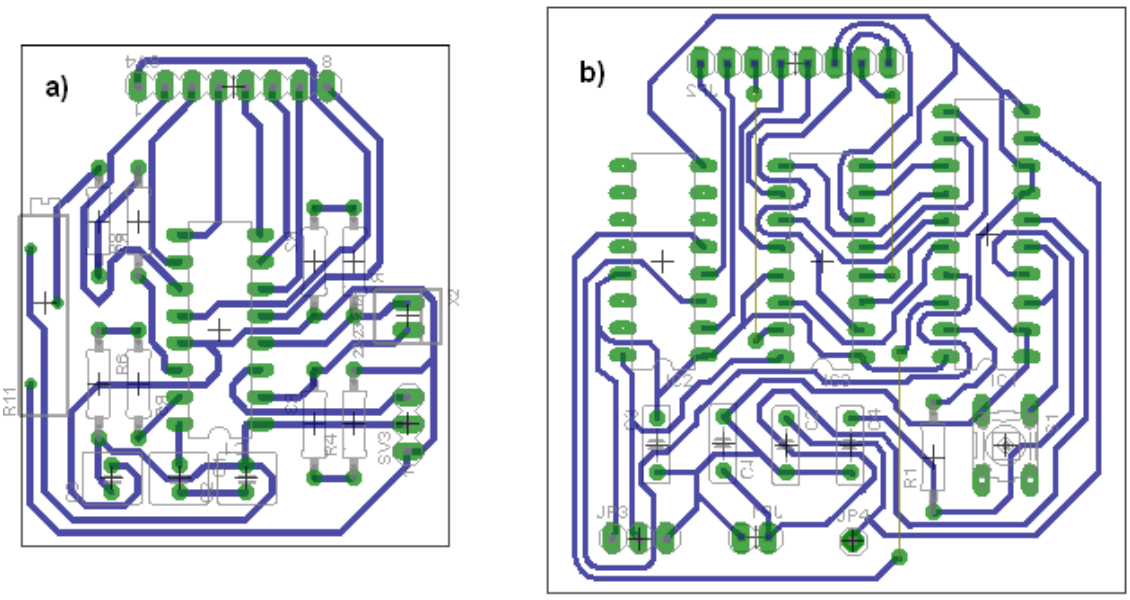


Figura 4.2.Circuitos impresos de a) la etapa de conversión D/A y b) etapa de conversión A/D.

La misma técnica se aplicó para todas las tarjetas de circuito impreso elaboradas.

En las figuras 4.4, 4.5 y 4.7 se muestran los resultados del proceso de fabricación del PCB y del soldado de dispositivos.

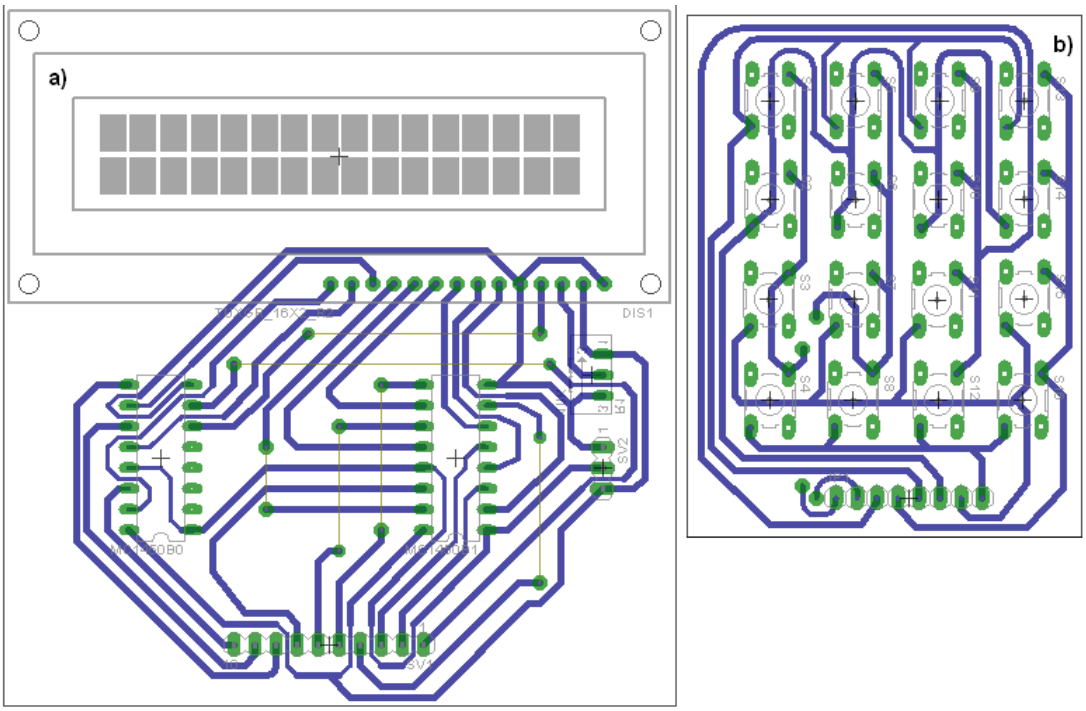


Figura 4.3. Circuitos impresos de a) la etapa de despliegue y b) etapa de ingreso de datos.

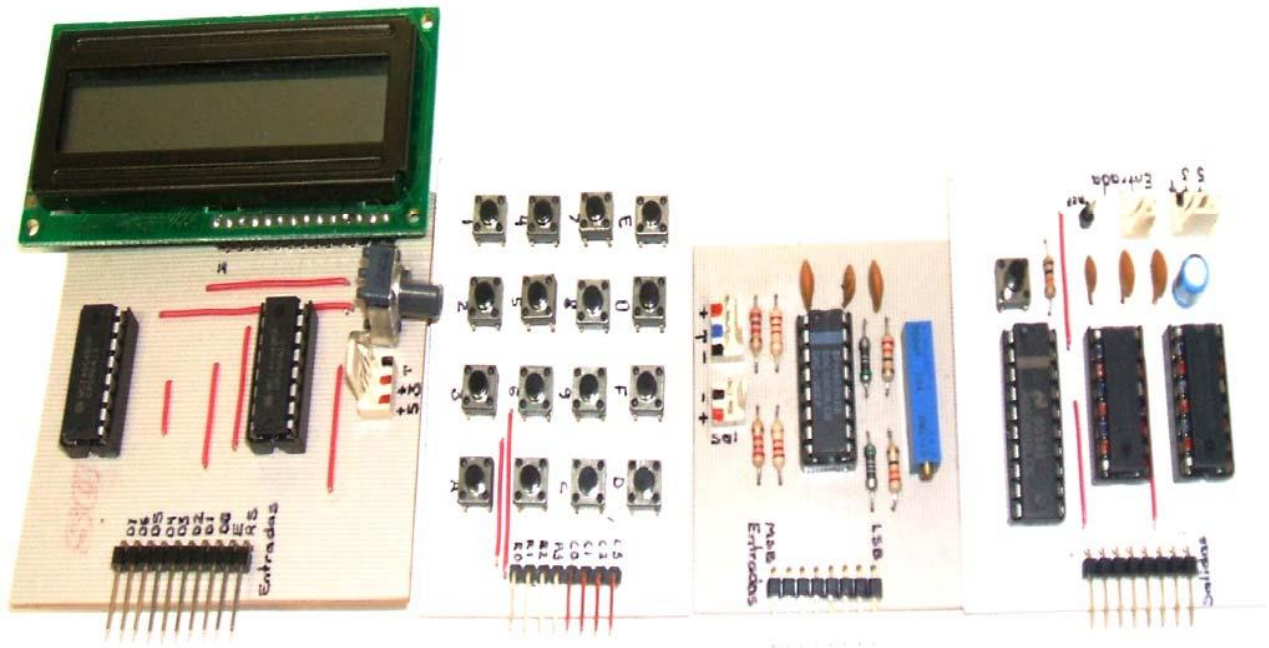


Figura 4.4. Etapas periféricas: despliegue e introducción de datos y conversión D/A y A/D.

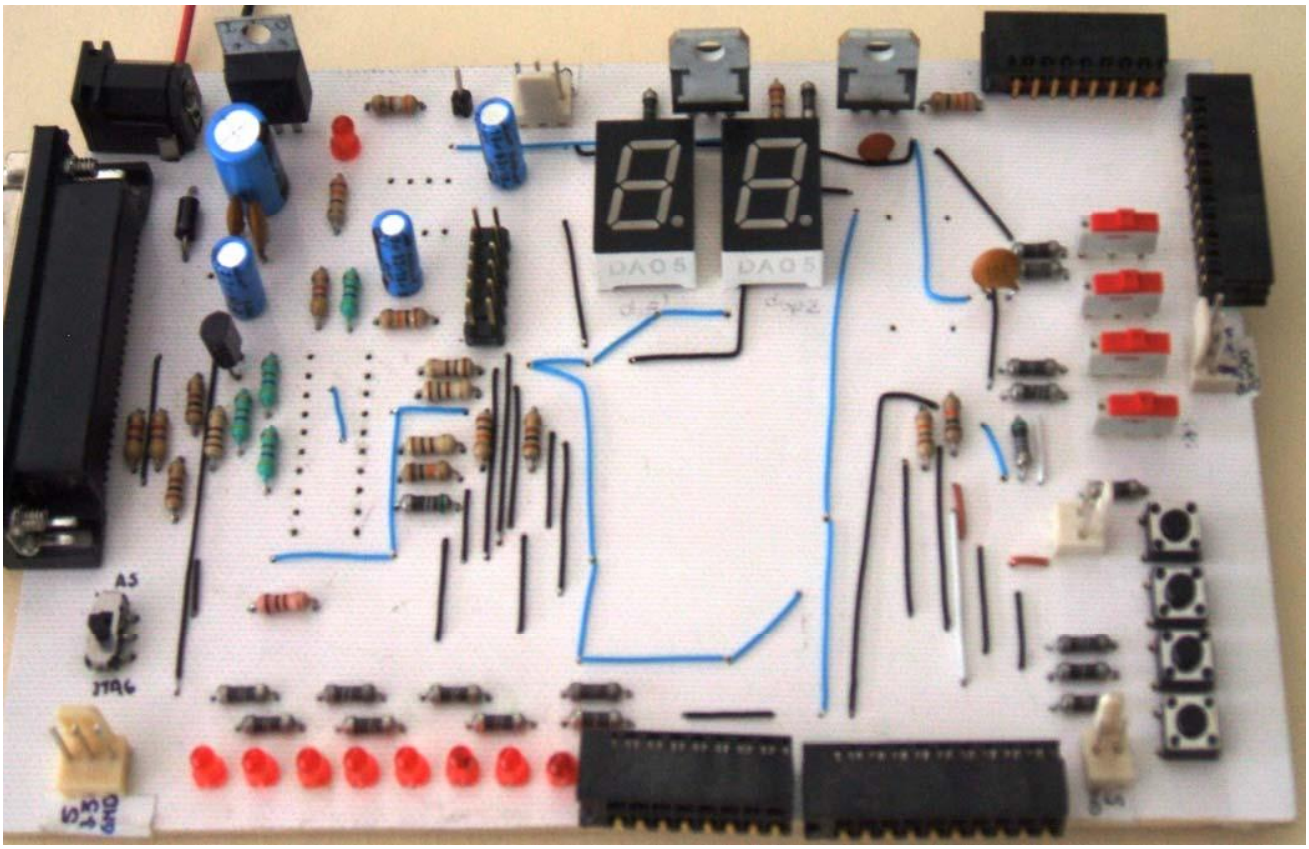


Figura 4.5. Tarjeta Electrónica principal.

4.2. Capacidades de la tarjeta electrónica

Para la implementación y síntesis de diseños de circuitos digitales a la mediad, la tarjeta electrónica principal, esta provista de:

- Un FPGA de la familia *Cyclone II* EP2C5T144C8N del fabricante Altera.
- El dispositivo EPSC1 para configurar al EP2C5T144C8N en modo *Active serial*.
- La interfaz electrónica de configuración *ByteBlaster II* que soporta los modos de configuración *Active Serial o JTAG* del EP2C5T144C8N.
- Ocho interruptores: cuatro de deslizamiento y cuatro de contacto momentáneo.
- Ocho LED's.
- Dos displays de siete segmentos de cátodo común
- Una fuente de señal de reloj externa mediante un oscilador a una frecuencia de 4MHz
- 48 terminales disponibles para ser usados como entradas o salidas.
- Salidas de voltaje a 5V, -5V y 3V.
- Etapas electrónicas periféricas por separado tales: convertidor analógico- digital y digital- analógico, teclado matricial de 4X4 y display de cristal Líquido de 1 línea por 8 caracteres.

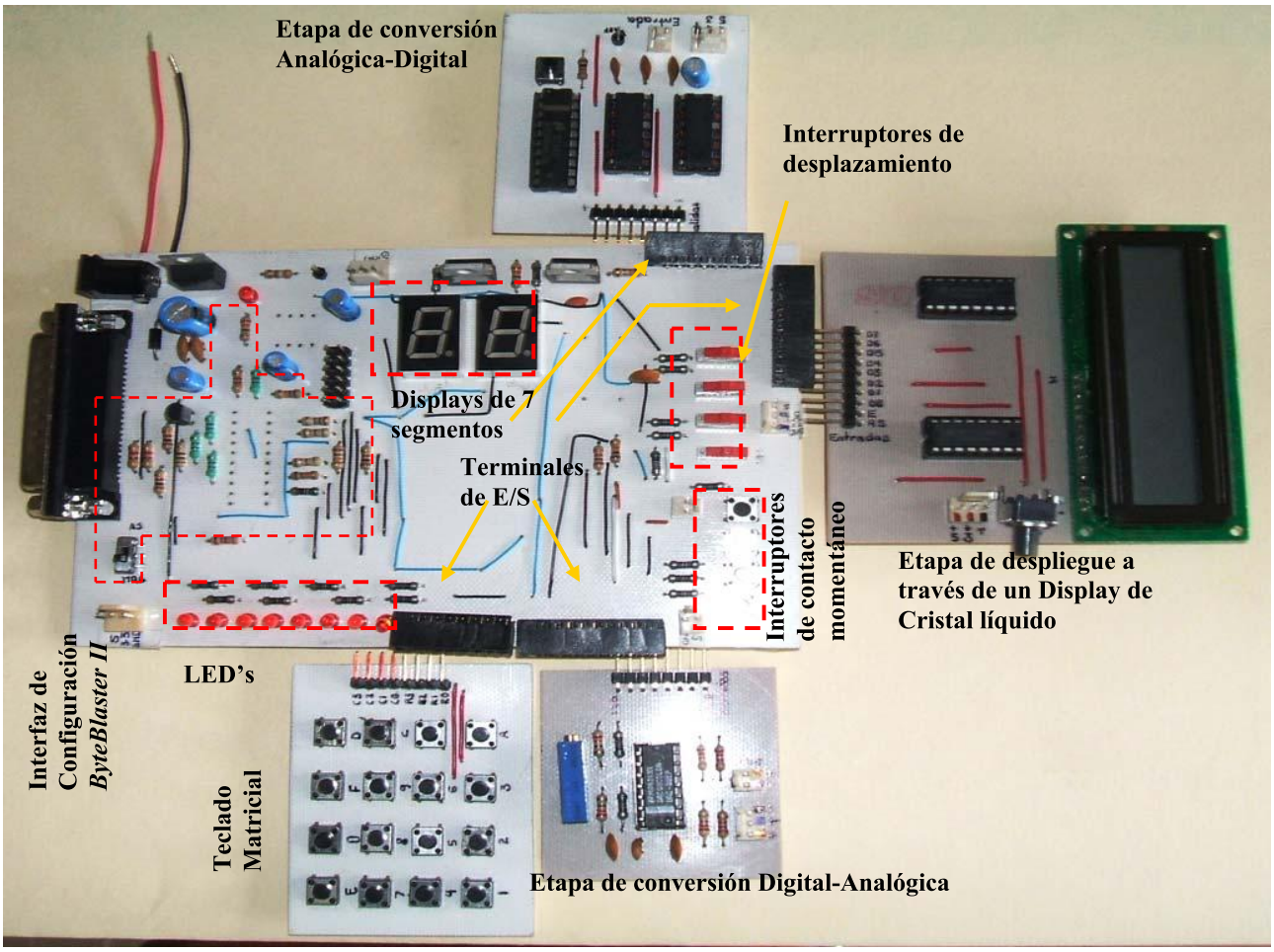


Figura 4.6. Integración de las etapas periféricas y la tarjeta electrónica principal.

Añadiendo a lo anterior, la inclusión de módulos embebidos que son adicionales a la lógica de aplicación general integrada en el mismo EP2C5T144C8N así como sus características, entre las cuales se encuentran:

- Veintiséis bloques de 4Kbits de memoria.
  Cada bloque de estos puede implementarse como una memoria RAM o ROM que a su vez pueden ser configurables en diversos modos.

- Trece Multiplicadores.
  Los cuales están optimizados para funciones de procesamiento digital de señales, filtros de respuesta al impulso finita (FIR), funciones para la transformada rápida de Fourier (FFT) y funciones de la transformada discreta coseno (DCT); y que se pueden operar bajo dos modos ya sea como un multiplicador de 18 bits o bien como dos independientes de 9 bits a una frecuencia de hasta 250MHz.

- Dos PLL's.
  Se utilizan, principalmente, para llevar a cabo síntesis de señales de reloj. Se puede realizar multiplicación, división, corrimiento de fase y cambio de ciclo de trabajo de una señal de reloj.
  Estan diseñados para operar bajo cuatro modos de realimentación que el fabricante denomina: *normal mode, zero delay buffer mode, no compensation mode* y *source synchronous mode.*

- Flexibilidad para interactuar con estándares lógicos y eléctricos de E/S, diferenciales y de una sola terminal como LVDS, RSDS, LVPECL, HSTL, SSTL, LVCMOS, LVTTL (1.5V, 1.8V y 3.3V).

- Flexibilidad para trabajar con memoria externa de tipo DDR, DDR2, SDRAM y SRAM.

- Salidas en colector abierto.

- Terminales de E/S de propósito dual.

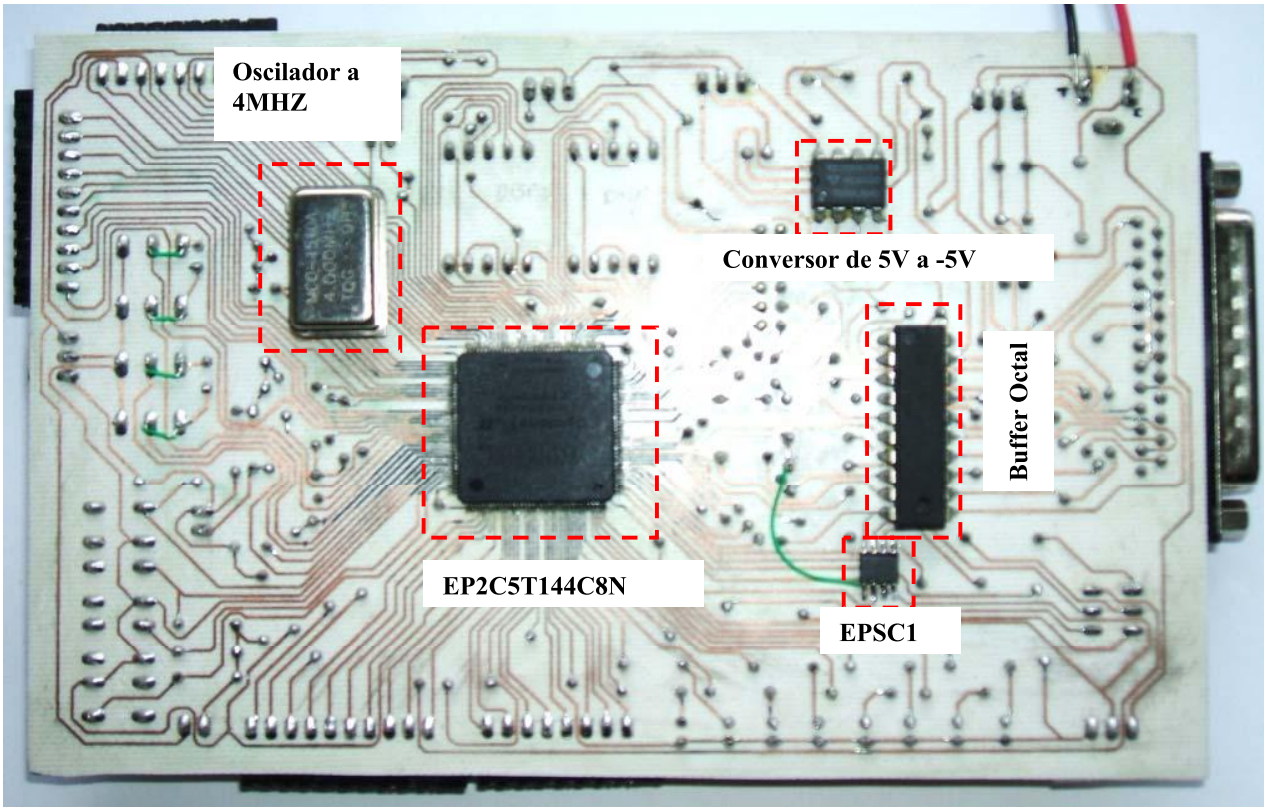


Figura 4.7. Parte posterior de la tarjeta electrónica principal.

Adicionalmente a las características anteriores, la capacidad del diseño se ve incrementada por las características de cada etapa periférica, listadas a continuación.

- Etapa de despliegue: permite la visualización de salidas en un display de cristal líquido de una línea de ocho caracteres. Esta basado en el microcontrolador SED1278F. Se cuenta con los circuitos integrados MC14504B que son convertidores de niveles lógicos de tensión para permitir la interacción entre el controlador SED1278F y el EP2C5T144C8N.

- Etapa de ingreso de datos: se dispone de un teclado matricial de cuatro líneas de entrada (renglones) y cuatro de salida (columnas).
  El teclado matricial no cuenta con un circuito decodificador ya que este se puede sintetizar en el propio FPGA, evitando el uso de circuitos adicionales.

- Etapa de conversión Digital –Analógica: basada en el integrado DAC0800 de 8 bits, que por su salida en colector abierto permite alcanzar amplitudes de hasta 20Vpp, además de que se tiene un bajo consumo de potencia, 30mW cuando se trabaja a +/-5V.

- Etapa de conversión Analógica- Digital: basada en el integrado ADC0802 de 8 bits, el cual ya no necesita de una señal de reloj adicional para el proceso de conversión ya que por medio de un resistor y un condensador, se establece la frecuencia de operación a 640KHz para la cual el fabricante garantiza una mayor exactitud.

En las figuras 4.6 y 4.7 se indican las ubicaciones de los componentes mencionados para cada tarjeta electrónica y que en conjunto constituyen el diseño final de la tarjeta electrónica de desarrollo basada en tecnología FPGA.

4.3. Pruebas finales

Este apartado tiene por objetivo principal comprobar la comunicación y configuración del FPGA EP2C5T144C8N además de verificar la correcta operación de todas las etapas periférica.

4.3.1. Prueba 1

Diseñar e implementar un circuito decodificador BCD (4 bits) a 7 segmentos, teniendo como entradas los cuatro interruptores deslizables, como la figura 4.8 muestra.

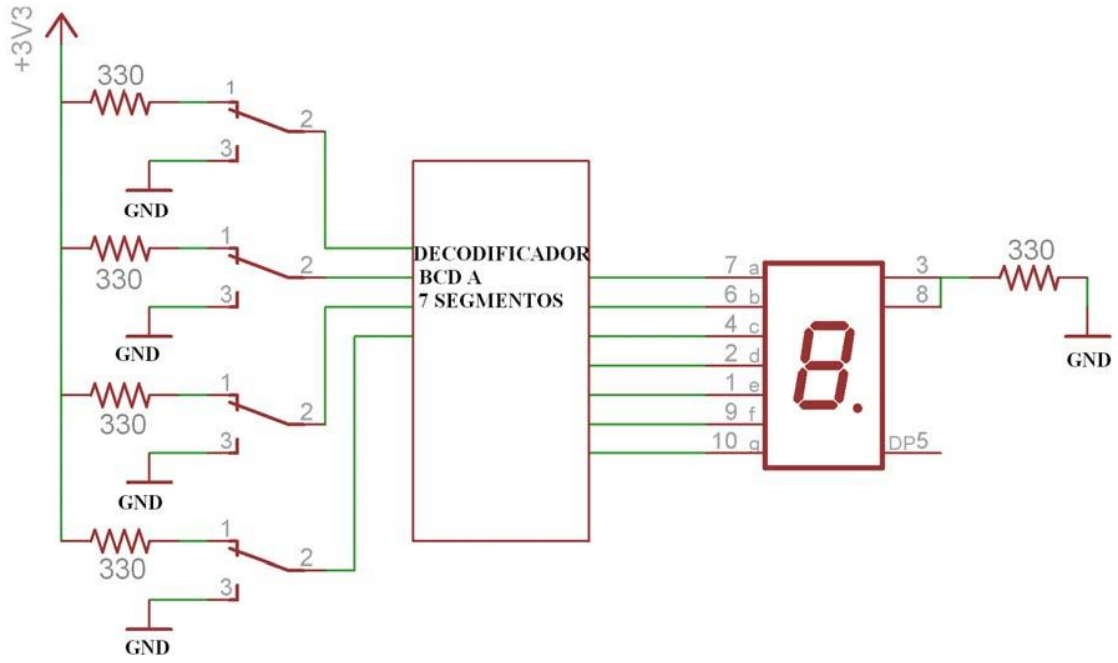


Figura 4.8. Diagrama eléctrico de un circuito decodificador BCD a 7 segmentos.

4.3.2. Prueba 2

Diseñar e implementar un generador de funciones digital, utilizando la tarjeta electrónica juntamente con la etapa de conversión Digital-Analógica.

Las funciones que se contemplan son una señal senoidal, cuadrada, triangular y diente de sierra.
La selección de cada una de las señales será por medio de interruptores de contacto momentáneo, como se indica en la figura 4.9, las cuales podrán visualizarse en un osciloscopio.

Se requiere que cada señal este constituida por un mínimo de 128 valores discretos; en el rango de valores establecidos el valor máximo de la señal se debe escalar a que sea FF(Hex) y para el valor mínimo 00(Hex) tal como la figura 4.10 lo señala.

La información binaria correspondiente a cada señal se puede generar con alguna herramienta de cálculo o con algún lenguaje de programación o hasta en el propio VHDL empleado para la descripción de hardware.

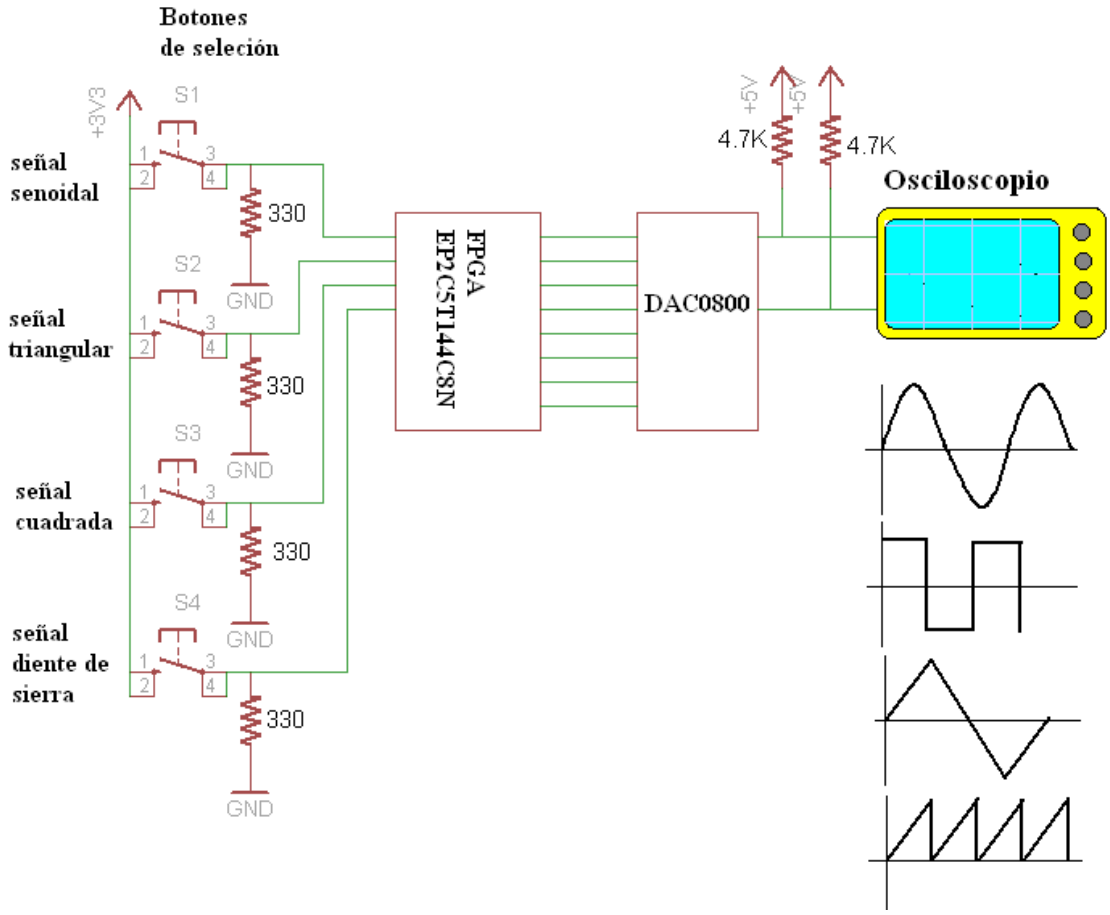


Figura 4.9. Esquema del generador de funciones digital.

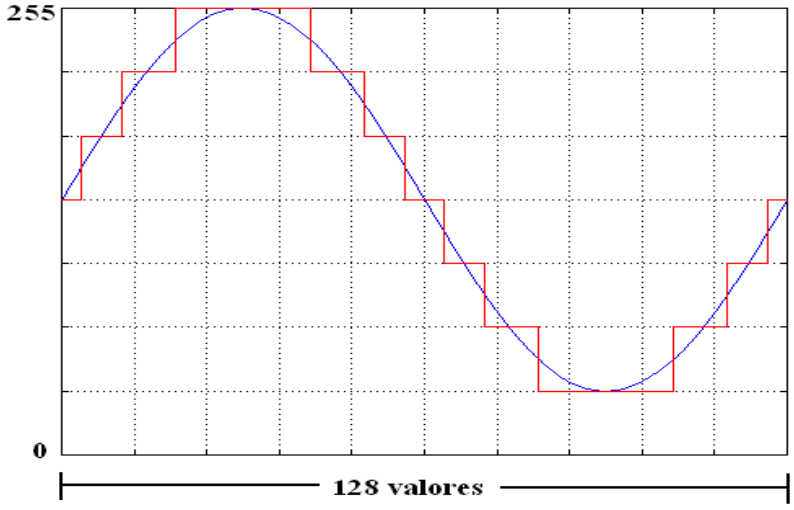


Figura 4.10. Valores discretos de la señal senoidal.

4.3.3. Prueba 3

Convertir la señal de voltaje proporcionada por la variación de un potenciómetro en un rango de 0 a 5V a su correspondiente valor binario en una resolución de 8 bits, utilizando la etapa de conversión Analógica-Digital. La salida debe mostrarse en LED's como se ve en la figura 4.11.

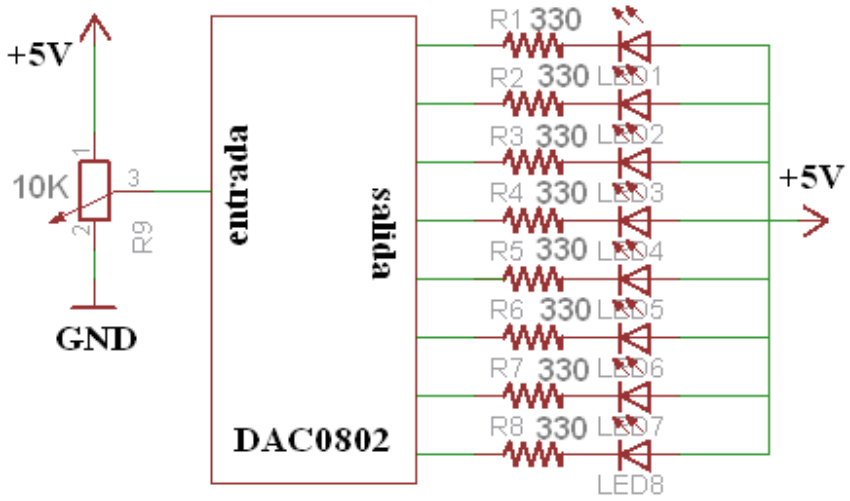


Figura 4.11. Circuito de prueba para la conversión A/D.

4.3.4 Prueba 4

Diseñar un circuito que permita ingresar datos a través del teclado matricial, los cuales deberán mostrarse en el display de cristal líquido.

Se debe considerar la rutina de inicialización del display, que consiste de una serie de comandos para configurar el modo de operación de este, atendiendo a las especificaciones de tiempo requeridas para cada instrucción.

La figura 4.12 muestra el diagrama de lo anteriormente especifcado.

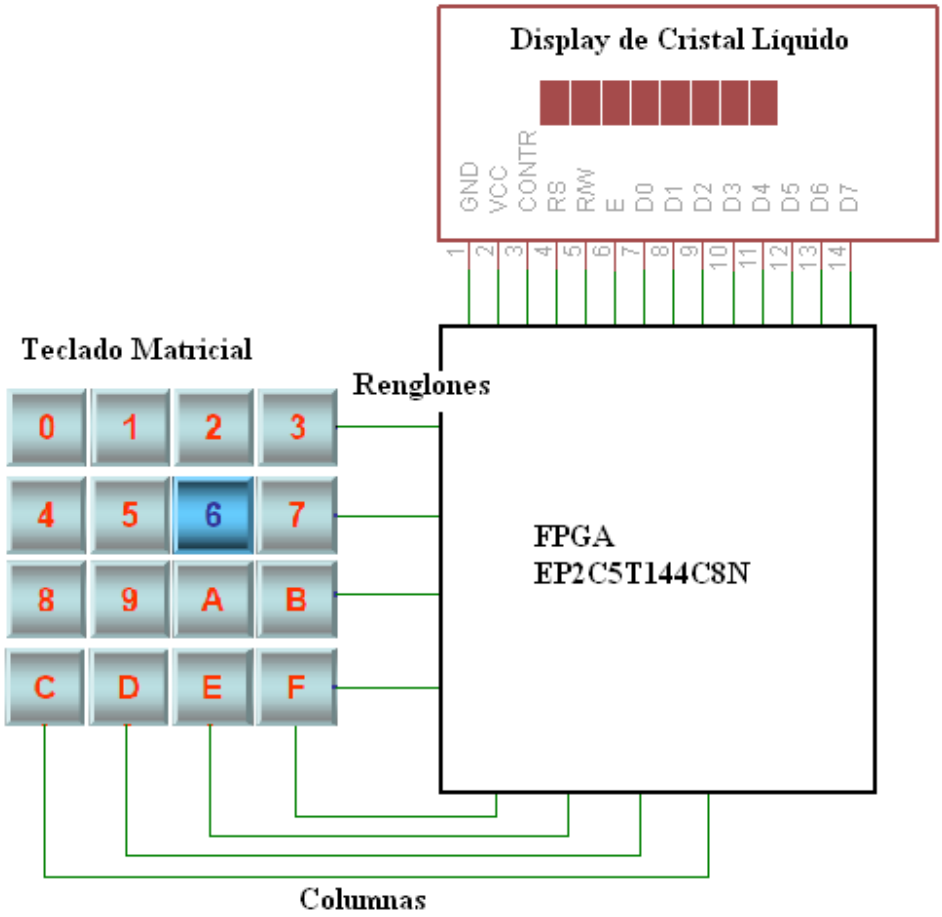


Figura 4.12. Diagrama para el ingreso y despliegue de datos.

El comportamiento lógico de cada circuito, correspondiente a las pruebas1, 2, y 4, es descrito en VHDL utilizando la herramienta *Quartus II*. El compilador de *Quartus II* se encarga de analizar el código así como de sintetizar el circuito y generar la implementación en el FPGA EP2C5T144C8N. Si no se detecta ningún error en este proceso, lo siguiente es hacer la asignación de E/S del circuito diseñado a las terminales disponibles en el FPGA.

Posterior a esto se procede con la configuración, ya sea en modo *JTAG* o *Active Serial*; en la primera los datos de configuración serán cargados directamente al FPGA y estos permanecerán mientras la polarización exista, en tanto que en el otro modo la información de la configuración es almacenada en la memoria flash del dispositivo EPCS1 para después transferirla al FPGA prescindiendo de configurar nuevamente al FPGA con el *software Quartus II* si se interrumpe la polarización, es decir, bastará con encender al sistema para que en un corto tiempo inicial se configure al FPGA por medio de la rutina e información que se encuentra dispuesta a ser transferida desde el dispositivo de configuración, para finalmente pasar a un estado de operación normal.

En las pruebas 2 y 4 el diseño se seccionó en módulos para abordarlos de manera más sencilla.

Por ejemplo en la prueba 2 se dividió en dos partes; la primera se encargaba de identificar el botón que había sido presionado para así determinar qué señal iba a desplegarse. Posterior a la identificación se necesitaba acceder a la información binaria de la señal en cuestión por lo que la segunda parte consistió en generar en VHDL una memoria ROM con los datos binarios de todas las señales que serían la salida a acondicionar para su visualización en el osciloscopio. Esta salida de se ingresó a la etapa de conversión digital-analógica donde se ajusto a su amplitud máxima, la cual fue de 5V antes de que se distorsionara.

La prueba 4 se abordo en tres partes, en una de ellas se describió el comportamiento lógico del circuito de captura de datos ingresados a través del teclado matricial, la siguiente se encargó de la decodificación de dichos datos al correspondiente código que maneja el display de cristal líquido y en la última se hizo la rutina de inicialización requerida para la correcta operación del display, configurándolo a que trabajara a 8 bits. En la operación del circuito los ajustes necesarios fueron identificar correctamente las entradas y salidas del teclado ya que al estar intercambiados los datos desplegados no correspondían a la tecla presionada.

Las pruebas restante se implementaron sin ningún por menor, únicamente tomando en cuenta la correcta conexión de los displays de 7 segmento y LED's

Resultados y conclusiones.

Se ha demostrado que este tipo de tecnología (PLD's) puede estar a nuestro alcance desarrollando arquitecturas de tarjetas electrónicas a la medida, como se ha expuesto en el presente trabajo

Para esto fue necesario indagar en las características de esta tecnología, es decir, conocer el estado del arte de los dispositivos lógicos programables y en especial de los denominados FPGA's tal como se introdujo en el capítulo 1 y se enfatizó en el capítulo 2, de manera que al familiarizarse con aspectos como las arquitecturas de las diferentes familias de FPGA's que ofrece cada fabricante y las herramientas de *software* y *hardware* para su manejo en el diseño de circuitos digitales; los criterios de selección fueran los apropiados para adquirir un dispositivo FPGA conveniente para el diseño y síntesis de la tarjeta electrónica.

El conocimiento pleno de las características del FPGA seleccionado dio pie a considerar las etapas electrónicas periféricas a la tarjeta electrónica, enfatizando que su elección está basada en algunos diseños de circuitos digitales que se realizan en la materia de electrónica digital, que adicional a estas se pueden implementar otras más por la flexibilidad que la tarjeta obtenida tiene para interactuar con diferentes estándares eléctricos y tener acceso a las terminales de entrada/salida del FPGA.

Llevar esto a algo tangible como lo fue la fabricación de la tarjeta electrónica y de las etapas periféricas requirió de adquirir experiencia en la elaboración de circuitos impresos con las técnicas

En las pruebas 2 y 4 el diseño se seccionó en módulos para abordarlos de manera más sencilla.

Por ejemplo en la prueba 2 se dividió en dos partes; la primera se encargaba de identificar el botón que había sido presionado para así determinar qué señal iba a desplegarse. Posterior a la identificación se necesitaba acceder a la información binaria de la señal en cuestión por lo que la segunda parte consistió en generar en VHDL una memoria ROM con los datos binarios de todas las señales que serían la salida a acondicionar para su visualización en el osciloscopio. Esta salida de se ingresó a la etapa de conversión digital-analógica donde se ajusto a su amplitud máxima, la cual fue de 5V antes de que se distorsionara.

La prueba 4 se abordo en tres partes, en una de ellas se describió el comportamiento lógico del circuito de captura de datos ingresados a través del teclado matricial, la siguiente se encargó de la decodificación de dichos datos al correspondiente código que maneja el display de cristal líquido y en la última se hizo la rutina de inicialización requerida para la correcta operación del display, configurándolo a que trabajara a 8 bits. En la operación del circuito los ajustes necesarios fueron identificar correctamente las entradas y salidas del teclado ya que al estar intercambiados los datos desplegados no correspondían a la tecla presionada.

Las pruebas restante se implementaron sin ningún por menor, únicamente tomando en cuenta la correcta conexión de los displays de 7 segmento y LED's

Resultados y conclusiones.

Se ha demostrado que este tipo de tecnología (PLD's) puede estar a nuestro alcance desarrollando arquitecturas de tarjetas electrónicas a la medida, como se ha expuesto en el presente trabajo

Para esto fue necesario indagar en las características de esta tecnología, es decir, conocer el estado del arte de los dispositivos lógicos programables y en especial de los denominados FPGA's tal como se introdujo en el capítulo 1 y se enfatizó en el capítulo 2, de manera que al familiarizarse con aspectos como las arquitecturas de las diferentes familias de FPGA's que ofrece cada fabricante y las herramientas de *software* y *hardware* para su manejo en el diseño de circuitos digitales; los criterios de selección fueran los apropiados para adquirir un dispositivo FPGA conveniente para el diseño y síntesis de la tarjeta electrónica.

El conocimiento pleno de las características del FPGA seleccionado dio pie a considerar las etapas electrónicas periféricas a la tarjeta electrónica, enfatizando que su elección está basada en algunos diseños de circuitos digitales que se realizan en la materia de electrónica digital, que adicional a estas se pueden implementar otras más por la flexibilidad que la tarjeta obtenida tiene para interactuar con diferentes estándares eléctricos y tener acceso a las terminales de entrada/salida del FPGA.

Llevar esto a algo tangible como lo fue la fabricación de la tarjeta electrónica y de las etapas periféricas requirió de adquirir experiencia en la elaboración de circuitos impresos con las técnicas

artesanales como la de planchado y en técnicas para soldar dispositivos de montaje superficial además del manejo de una herramienta de *software* para la edición de circuitos impresos para estos diseños, lo que viene a demostrar que la complejidad que se aparentaba en un principio de iniciar un diseño propio queda reducida.

Los beneficios primarios que se logran al publicar de forma abierta la información acerca del diseño y síntesis de una tarjeta electrónica de desarrollo de tecnología FPGA, son la no dependencia del uso de tarjetas de desarrollo comerciales de algún fabricante (en las que por razones comerciales obvias siempre quedan ocultas configuraciones de etapas electrónicas vitales para la comunicación, configuración y operación de sus diseños), propiciar el desarrollo y síntesis de sistemas embebidos (sistemas electrónicos de aplicación específica, cuya operación es independiente y totalmente funcional y están confinados en un área determinada), y en consecuencia la visualización de la posible aplicación de los PLD's en general evaluando sus ventajas para un determinado fin.

Tradicionalmente por la misma naturaleza de la tecnología abordada (FPGA, VHDL y metodología descendente) se tiene pleno conocimiento de la o las arquitecturas electrónicas sintetizadas en la tarjeta y/o en el FPGA.

Por otra parte, en el ámbito docente la tarjeta electrónica basada en el FPGA EP2C5T144C8N viene a ser un complemento a las tecnologías que se instruyen al alumnado del área de electrónica y de esta forma se logre enriquecer sus conocimientos sobre la tecnología de PLD's.

Finalmente y de forma general, se concluye que el presente trabajo de tesis logra poner a disposición de todas aquellas persona interesadas en el tema que quieran iniciar sus propios diseños de tarjetas electrónicas basadas en FPGA's conforme a sus necesidades y requerimientos, la documentación así como los diseños de los circuitos de la tarjeta electrónica y las etapas electrónicas periféricas, con la ventaja de que lo aquí expuesto es útil para cualquier dispositivo FPGA comercial y no necesariamente quedar limitados al EP2C5T144C8N empleado, ya que los diferentes fabricantes de este tipo de circuitos presentan características similares.

Apéndices A: Tablas de los FPGA's disponibles en el mercado

Family FPGA	Application/ feature focus	Density range	Top performance (logic, I/O)	Hard-wired logic	On-chip memory	Material	Power requirement
Igloo/IglooE	The Actel Igloo family of reprogrammable, full-featured flash FPGAs meets the demanding power and area requirements of portable electronics.	30,000 to 3 million system gates	Maximum 616 I/Os	PLL	As many as 112 blocks of 4608-bit embedded SRAM (variable configurations); 1024 bits of on chip, user-accessible, nonvolatile flash ROM	Flash	5 to 294 mW
ProASIC3/ ProASIC3E	The ProASIC3/E families of flash FPGAs offer performance, density, and features for consumer automotive and industrial applications.	30,000 to 3 million system gates	Maximum 616 I/Os	PLL	As many as 112 blocks of 4608-bit embedded SRAM (variable configurations), 1024 bits of on-chip flash ROM	Flash	3 to 37 mW

Tabla A.1. FPGA's de ACTEL.

Family FPGA	Application/ feature focus	Density range	Top performance (logic, I/O)	Hard-wired logic	On-chip memory	Material	Power requirement
Cyclone	Altera builds the Cyclone series of FPGAs from the ground up for cost-sensitive, highvolume applications. These low-cost devices provide application-focused features, such as embedded memory, external-memory interfaces, and clock-management circuitry.	3000 to 20,000 LEs (logic elements), as many as 250,000 equivalent ASIC gates	LVDS at 640 Mbps, DDR at 167 MHz, maximum clock frequency at 200 MHz	As many as two PLLs	As much as 288 kbits of RAM	SRAM	20 to 500 mA, depending on design
CycloneII	Cyclone II devices include customer defined FPGA features for low-cost applications, including a wide range of density, memory, embedded-multiplier, and packaging options.	5000 to 70,000 LEs (logic elements), as many as 875,000 equivalent	LVDS at 805 Mbps, DDR at 167 MHz, DDR2 at 167 MHz, maximum clock frequency at 260 MHz	As many as 150 18x18 multipliers, as many as four PLLs	1.1 Mbits of RAM	SRAM	Power-up current requirement of 20 to 500 mA, depending on design; static

		ASIC gates					power of 35 to 250 mA
CycloneIII	Cyclone III FPGAs combine low power, high functionality, and low cost.	5000 to 120,000 LEs (logic elements), as many as 1.5 million equivalent ASIC gates	LVDS at 840 Mbps, DDR at 167 MHz, DDR2 at 200 MHz, maximum clock frequency at 280 MHz	As many as four PLLs, as many as 288 18x18 multipliers	As much as 4 Mbits of RAM	SRAM	Power-up current requirement of 10 to 500 mA, depending on design; static power of 35 to 160 mA
Arria GX	Arria GX devices target high-volume, costsensitive, transceiver-based markets, such as wireless communications, video processing, military, industrial, and computation and storage.	21,500 to 90,200 LEs (logic elements), as many as 1.2 million equivalent ASIC gates	Performance varies based on design, I/O LVDS at 840 Mbps, support for DDR2 at 233 MHz	Four to 12 SERDES (serializer/ deserializer) blocks at speeds as high as 2.5 Gbps, 10 to 44 DSP blocks, 230 to 538 high-speed user-I/O blocks	1.2 to 4.5 Mbits	SRAM	Design-dependent power-up-current requirement

Tabla A.2. FPGA's de ALTERA.

Family FPAG	Application/ feature focus	Density range	Top performance	Hard-wired Logia	On-Chip memory	Matrial	Power requirement
LatticeECP2	Lattice ECP2 devices target high-volume applications in which I/O and DSP performance previously required the use of highperformance and high-cost FPGAs.	300,000 to 5 million ASIC gates	350-MHz logic, 840-Mbps I/O	sysDSP blocks, pre-engineered logic supporting 533-Mbps DDR2, 400-Mbps DDR, 750- Mbps SPI4.2, 840-Mbps generis interfaces	55 kbits to 1 Mbit	SRAM	130- to 237-mW static power plus design's switching power
LatticeECP2M	Lattice ECP2M devices target high-volume applications in which I/O and DSP performance, SERDES (serializer/deserializer), or memory capacity previously required the use of high-performance and high-cost FPGAs.	5 million to 20 million ASIC gates	350-MHz logic, 840-Mbps I/O, and 3.2-Gbps SERDES (serializer/deserializer)	3.2-Gbps SERDES (serializer/ deserializer), sysDSP blocks, pre-engineered logic supporting 533-Mbps DDR2, 400-Mbps DDR, 750-Mbps SPI4.2, 840- Mbps generic interfaces	1.2 to 5.3 Mbits	SRAM	150- to 285-mW static power plus design's switching power
LatticeXP2	Lattice XP2 devices target high-volume applications requiring the smallest board area, instant availability of logic at power-up, and FPGA security.	725,000 to 4 million ASIC gates	320-MHz logic, 840-Mbps I/O	sysDSP blocks, pre-engineered logic supporting 400-Mbps DDR2, 400-Mbps DDR, 840-Mbps generic interfaces	166 to 885 kbits	Hybrid SRAM/ flash on single die	90- to 195-mW static power plus design's switching power

Tabla A.3. FPGA's de LATTICE SEMICONDUCTOR CORPORATION.

Family FPAG	Application/ feature focus	Density range	Top performance	Hard-wired Logia	On-Chip memory	Material	Power requirement
Spartan-3 generation	The Spartan-3 generation targets highdensity,high-pin-count, and highly integrated data-processing applications.	50,000 to 5 million system gates	622-psec data-transfer rate per differential I/O, 33-MHz PCI, DDR and DDR2 support as fast as 333 Mbps, 18- to 280-MHz DCM (digital-clock-manager) input frequency, as many as 74,880 logic cells, as many as 784 single-ended I/Os, as many as 344 differential-I/O pairs, as many as 104 multipliers	18x18-bit multipliers, 18-kbit block RAM, DCMs (digital-clock managers), advanced-I/O block with DDR registers, 24 differential- and single-ended standards, single-ended and differential termination	As much as 1872 kbits of block RAM, as much as 520 kbits of distributed RAM	SRAM	13.5- to 193.5-mA typical quiescent current
Spartan-3E platform	The Spartan-3E platform is logic-optimized for logic integration, DSP coprocessing, and embedded control.	100,000 to 1.6 million system gates	270-MHz block RAM, 270-MHz multiplier, more-than-622-Mbps data-transfer rate per differential I/O, DDR and DDR2 support as fast as 333 Mbps, as many as 33,192 logic cells, as many as 376 single-ended I/Os, as many as 156 differential-I/O pairs, as many as 36 multipliers	18x18-bit multipliers, 18-kbit block RAM, DCMs (digital-clock managers), advanced-I/O block with DDR registers, 18 differential- and single-ended standards, differential termination, multibootconfiguration logia	As much as 648 kbits of block RAM, as much as 231 kbits of distributed RAM	SRAM	16.8- to 111.5-mA typical quiescent current
Spartan-3A platform	The Spartan-3A platform targets applications requiring extensive I/O, such as bridging, differential signaling, and memory interfacing.	50,000 to 1.4 million system gates	270-MHz block RAM, 280-MHz multiplier, more-than-622-Mbps data-transfer rate per differential I/O, DDR and DDR2 support as fast as 400 Mbps, as many as 25,344 logic cells, as many as 502 single-ended I/Os, as many as 227 differential-I/O pairs , as many as 32 multipliers	Device DNA, 18x18-bit multipliers, 18-kbit block RAM, DCMs (digital-clock managers), advanced-I/O block with DDR registers, 26 differential- and single-ended standards, differential termination, power-management modes, multiboot-configuration logic with watchdog timer	As much as 576 kbits of block RAM, as much as 176 kbits of distributed RAM	SRAM	0-mA quiescent current in hibernation mode, 7.7- to 48.3-mA typical quiescent current, 4.6- to 29-mA typical quiescent current in suspend mode
Spartan-3AN platform	The nonvolatile Spartan-3AN platform targets applications requiring nonvolatile system integration, security, or large amounts of user-accessible flash.	50,000 to 1.4 million system gates	270-MHz block RAM, 280-MHz multiplier, more-than-622-Mbps data-transfer rate per differential I/O, 66-MHz PCI, DDR and DDR2 support as fast as 400 Mbps, 5- to 320-MHz DCM (digital-clock-manager) input frequency, as many as 25,344 logic cells, as many as 502 single ended I/Os, as many as 227 differential-I/O pairs, as many as 32 multipliers	Device DNA, 18x18-bit multipliers, 18-kbit block RAM, DCMs (digital-clock managers), advanced-I/O block with DDR registers, 26 differential- and single- ended standards, differential termination, power-management modes, multiboot-configuration logic with watchdog timer	As much as 11 Mbits of nonvolatile, user-accessible flash; as much as 576 kbits of block RAM; as much as 176 kbits of distributed RAM	SRAM plus flash on system in package	0-mA quiescent current in hibernation mode, 7.8- to 48.4-mA typical quiescent current, 4.6- to 29-mA typical quiescent current in suspend mode

Spartan-3A DSP platform	The Spartan-3A DSP platform targets digitalsignal processing in applications requiring integrated DSP MAC (multiply/accumulate) operations and expanded memory.	1.8 million to 3.4 million system gates	270-MHz block RAM, 287-MHz DSP48A/multiplier, more-than-622-Mbps data-transfer rate per differential I/O, DDR and DDR2 support as fast as 333 Mbps, as many as 53,712 logic cells, as many as 519 single-ended I/Os, as many as 227 differential-I/O pairs, as many as 126 DSP48A slices (advanced multiply/ accumulate elements) or 126 18x18-bit multipliers	Device DNA, DSP48A slices (advanced multiply/accumulate elements), 18-kbit block RAM, DCMs (digital-clock managers), advanced-I/O block with DDR registers, 24 differential- and single- ended standards, differential termination, power-management modes, multiboot-configuration logic with watchdog timer	As much as 2268 kbits of block RAM, as much as 373 kbits of distributed RAM	SRAM	0-mA quiescent current in hibernation mode, 83.2- to 135.2- mA typical quiescent current, 49.8- to 81.1-mA typical quiescent current in suspend mode

Tabla A.4. FPGA's de XILINX.

Apéndice B: Tablas con la asignación de E/S a las terminales del EP2C5T144C8N

Nombre de la señal	No de terminal en el FPGA
Interruptor[0]	97
Interruptor[1]	99
Interruptor[2]	96
Interruptor[3]	94

Tabla B.1. Asignación de interruptores de deslizamiento.

Nombre de la señal	No de terminal en el FPGA
Push bottom[0]	104
Push bottom[1]	103
Push bottom[2]	101
Push bottom[3]	100

Tabla B.2. Asignación de los interruptores de contacto momentáneo.

Nombre de la señal	No de Terminal en el FPGA
Led[0]	144
Led[1]	143
Led[2]	142
Led[3]	141
Led[4]	139
Led[5]	137
Led[6]	136
Led[7]	135

Tabla B.3. Asignación de LED's.

Nombre de la señal	No de terminal en el FPGA	Nombre de la señal	No de terminal en el FPGA
SegD1[a]	44	SegD2[a]	55
SegD1[b]	43	SegD2[b]	53
SegD1[c]	42	SegD2[c]	52
SegD1[d]	41	SegD2[d]	51
SegD1[e]	40	SegD2[e]	48
SegD1[f]	45	SegD2[f]	57
SegD1[g]	47	SegD2[g]	58

Tabla B.4. Asignación de los segmentos de los dos displays.

Nombre de la señal	No de Terminal en el FPGA
Reloj_4M	89
Reloj_externo	91

Tabla B.5. Asignación de las entradas de reloj.

Terminales de E/S de propósito general:
134,133,132,129,126,125,122,121,120,119,118,115,114,113,112,93,92,90,72,71,70,69,67,65,64,63, 88,87,86,81,80,79,76,75,74,73,59,32,31,30,28,27,26,25,24,9,8,7.

Apéndice C: Códigos generados en VHDL para las pruebas 2 y 4.

Prueba2

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity estados_generador is
port(clk: in std_logic;
        botones: in std_logic_vector(3 downto 0);
        salida:out std_logic_vector(9 downto 0));
end estados_generador;

architecture algo of estados_generador is

type estados is(inicio, sen, esp_sen, tri, esp_tri, cuad, esp_cuad, diente, esp_diente);
signal presente:estados:=inicio;
signal cuenta:std_logic_vector(7 downto 0);
signal senal: std_logic_vector(1 downto 0);

begin

process(clk)
begin
        if clk='1' and clk 'event then
                if cuenta="11111111" then cuenta<="00000000"; else cuenta<=cuenta+1; end if;
        end if;
end process;

process(clk)
begin
        if clk='1' and clk 'event then
        case presente is
                when inicio => if botones="1000" then presente<=esp_sen; end if;
                                                        if botones="0100" then presente<=esp_tri; end if;
                                                        if botones="0010" then presente<=esp_cuad; end if;
                                                        if botones="0001" then presente<=esp_diente; end if;

                when esp_sen => if botones="0000" then presente<=sen; end if;
                when esp_tri => if botones="0000" then presente<=tri; end if;
                when esp_cuad => if botones="0000" then presente<=cuad; end if;
                when esp_diente => if botones="0000" then presente<=diente; end if;

                when sen => if botones/="0000" then presente<=inicio; end if;

                when tri => if botones/="0000" then presente<=inicio; end if;

                when cuad => if botones/="0000" then presente<=inicio; end if;

                when diente => if botones/="0000" then presente<=inicio; end if;

                end case;
        end if;
```

```
end process;

process(presente)
begin
        case presente is
                when inicio=> null;
                when esp_sen=> senal<="00";
                when esp_tri=> senal<="01";
                when esp_cuad=> senal<="10";
                when esp_diente=> senal<="11";
                when sen=> senal<="00";
                when tri=> senal<="01";
                when cuad=> senal<="10";
                when diente=> senal<="11";
        end case;
end process;

        salida(9)<=senal(1);
        salida(8)<=senal(0);
        salida(7)<=cuenta(7);
        salida(6)<=cuenta(6);
        salida(5)<=cuenta(5);
        salida(4)<=cuenta(4);
        salida(3)<=cuenta(3);
        salida(2)<=cuenta(2);
        salida(1)<=cuenta(1);
        salida(0)<=cuenta(0);

end algo;
```

Prueba 4

```
Library ieee;
use ieee.std_logic_1164.all;

entity TecladoLCD is
port(clk,hab_lcd:in std_logic;
        renglones:in std_logic_vector(3 downto 0);
        columnas:out std_logic_vector(3 downto 0);
        bit_sinc,rs,ena:out std_logic;
        valor:out std_logic_vector(7 downto 0);
        display:out std_logic_vector(6 downto 0));

end TecladoLCD;

architecture algo of TecladoLCD is

type estados is(inicio, espera1, tecla1, espera2, tecla2, espera3, tecla3,
 espera4, tecla4, espera5,tecla5, espera6, tecla6, espera7, tecla7, espera8,
```

```
 tecla8, espera9, tecla9, espera0, tecla0, esperaA, teclaA, esperaB, teclaB,
 esperaC, teclaC, esperaD, teclaD, esperaE, teclaE, esperaF, teclaF);
signal presente:estados:=inicio;

signal cuenta_tec:integer range 0 to 15:=0;
signal bit_s:std_logic:='0';

type estados1 is(inicio1, inicializa,envia_dato);
signal presente1: estados1:=inicio1;

type estados_com is(inicio_com,set_com1,set_com2,set_com3,
 env_com1,env_com2,env_com3,valida_com1,valida_com2,valida_com3);
signal presente_com: estados_com:=inicio_com;

type estados_dat is(inicio_dat,set_dat1,env_dat1,valida_dat1);
signal presente_dat: estados_dat:=inicio_dat;

signal aux_com,dato: std_logic_vector(7 downto 0);
signal rs_dat,rs_com,ena_dat,ena_com,hab_com,hab_dat,hab_ini,hab_env: std_logic;

begin

process(clk)
begin
if clk='1' and clk 'event then
        if bit_s='0' then
                if cuenta_tec=15 then cuenta_tec<=0; else cuenta_tec<= cuenta_tec+1; end if;
        end if;
end if;
end process;

mux:
process(cuenta_tec,renglones)
begin
case cuenta_tec is
        when 0|4|8|12 => bit_s <=renglones(0);
        when 1|5|9|13 => bit_s <=renglones(1);
        when 2|6|10|14 => bit_s <=renglones(2);
        when others=> bit_s <= renglones(3);
end case;
end process;

bit_sinc<=bit_s;

deco:
process(cuenta_tec)
begin
case cuenta_tec is
        when 0 to 3 => columnas <= "1000";
        when 4 to 7 => columnas <= "0100";
        when 8 to 11 => columnas <= "0010";
        when others => columnas<= "0001";
end case;
end process;

process(clk)
```

```
begin
if clk='1' and clk'event then
        case presente is
                when inicio => if bit_s='1' then
                        case cuenta_tec is
                                when 0 => presente<=espera1;
                                when 1 => presente<=espera4;
                                when 2 => presente<=espera7;
                                when 3 => presente<=esperaE;
                                when 4 => presente<=espera2;
                                when 5 => presente<=espera5;
                                when 6 => presente<=espera8;
                                when 7 => presente<=espera0;
                                when 8 => presente<=espera3;
                                when 9 => presente<=espera6;
                                when 10 => presente<=espera9;
                                when 11 => presente<=esperaF;
                                when 12 => presente<=esperaA;
                                when 13 => presente<=esperaB;
                                when 14 => presente<=esperaC;
                                when others=> presente<=esperaD;
                        end case; end if;
                when espera0=> if bit_s='0' then presente<=tecla0; end if;
                when espera1=> if bit_s='0' then presente<=tecla1; end if;
                when espera2=> if bit_s='0' then presente<=tecla2; end if;
                when espera3=> if bit_s='0' then presente<=tecla3; end if;
                when espera4=> if bit_s='0' then presente<=tecla4; end if;
                when espera5=> if bit_s='0' then presente<=tecla5; end if;
                when espera6=> if bit_s='0' then presente<=tecla6; end if;
                when espera7=> if bit_s='0' then presente<=tecla7; end if;
                when espera8=> if bit_s='0' then presente<=tecla8; end if;
                when espera9=> if bit_s='0' then presente<=tecla9; end if;
                when esperaA=> if bit_s='0' then presente<=teclaA; end if;
                when esperaB=> if bit_s='0' then presente<=teclaB; end if;
                when esperaC=> if bit_s='0' then presente<=teclaC; end if;
                when esperaD=> if bit_s='0' then presente<=teclaD; end if;
                when esperaE=> if bit_s='0' then presente<=teclaE; end if;
                when esperaF=> if bit_s='0' then presente<=teclaF; end if;
                when others=>if bit_s='1' then presente<=inicio; end if;
                end case;
end if;
end process;

process(presente)
begin
case presente is
        when espera0 | tecla0 | inicio => display<="0000001"; dato<="00110000";
        when espera1 | tecla1 => display<="1001111"; dato<="00110001";
        when espera2 | tecla2 => display<="0010010"; dato<="00110010";
        when espera3 | tecla3 => display<="0000110"; dato<="00110011";
        when espera4 | tecla4 => display<="1001100"; dato<="00110100";
        when espera5 | tecla5 => display<="0100100"; dato<="00110101";
        when espera6 | tecla6 => display<="0100000"; dato<="00110110";
        when espera7 | tecla7 => display<="0001111"; dato<="00110111";
        when espera8 | tecla8 => display<="0000000"; dato<="00111000";
        when espera9 | tecla9 => display<="0000100"; dato<="00111001";
```

```
		when esperaA | teclaA => display<="1100010"; dato<="01000001";
		when esperaB | teclaB => display<="1100000"; dato<="01000010";
		when esperaC | teclaC => display<="1110010"; dato<="01000011";
		when esperaD | teclaD => display<="1000010"; dato<="01000100";
		when esperaE | teclaE => display<="0110000"; dato<="01000101";
		when others => display<="0111000"; dato<="01000110";
end case;
end process;

process(clk,hab_lcd)
begin
		if hab_lcd='0' then
			elsif clk='1' and clk 'event then
				case presente1 is
					when inicio1 => if hab_ini='0' then presente1<=envia_dato; else
presente1<=inicializa; end if;
					when inicializa=> if hab_ini='0' then presente1<=envia_dato; end if;
					when envia_dato=> if hab_env='0' then presente1<=inicio1; end if;
				end case;
			end if;
end process;

process(presente1)
begin
		case presente1 is
			when inicio1=> hab_com<='0'; hab_dat<='0';
			when inicializa=> hab_com<='1'; hab_dat<='0';
			when envia_dato=> hab_com<='0'; hab_dat<='1';
		end case;
end process;

process(clk, hab_com)
begin
		if hab_com='0' then
		elsif clk='1' and clk 'event then
			case presente_com is
				when inicio_com => presente_com <= set_com1;
				when set_com1 => presente_com<= env_com1;
				when env_com1 => presente_com<= valida_com1;
				when valida_com1 => presente_com<=set_com2;
				when set_com2 => presente_com<= env_com2;
				when env_com2 => presente_com<= valida_com2;
				when valida_com2 => presente_com<=set_com3;
				when set_com3 => presente_com<= env_com3;
				when others => presente_com<= valida_com3;
				- - when valida_com3 => presente_com<= valida_com3;
			end case;
		end if;
end process;

process(presente_com)
begin
		case presente_com is
			when inicio_com => aux_com<="00000000";rs_com<='0'; ena_com<='0'; hab_ini<='1';
			when set_com1 => aux_com<="00111000"; rs_com<='0'; ena_com<='0'; hab_ini<='1';
			when env_com1 => aux_com<="00111000"; rs_com<='0'; ena_com<='1'; hab_ini<='1';
```

```
                when valida_com1 => aux_com<="00111000"; rs_com<='0'; ena_com<='0'; hab_ini<='1';
                when set_com2 => aux_com<="00001110"; rs_com<='0'; ena_com<='0'; hab_ini<='1';
                when env_com2 => aux_com<="00001110"; rs_com<='0'; ena_com<='1'; hab_ini<='1';
                when valida_com2 => aux_com<="00001110"; rs_com<='0'; ena_com<='0'; hab_ini<='1';
                when set_com3 => aux_com<="00000001"; rs_com<='0'; ena_com<='0'; hab_ini<='1';
                when env_com3 => aux_com<="00000001"; rs_com<='0'; ena_com<='1'; hab_ini<='1';
                when valida_com3 => aux_com<="00000001"; rs_com<='0'; ena_com<='0'; hab_ini<='0';
        end case;
end process;

process(clk, hab_dat)
begin
        if hab_dat='0' then
        elsif clk='1' and clk 'event then
                case presente_dat is
                        when inicio_dat => presente_dat <= set_dat1;
                        when set_dat1 => presente_dat<= env_dat1;
                        when env_dat1=> presente_dat<= valida_dat1;
                        when valida_dat1 => presente_dat<=inicio_dat;
                end case;
        end if;
end process;

process(presente_dat,dato)
begin
        case presente_dat is
                when inicio_dat => rs_dat<='1'; ena_dat<='0'; hab_env<='1';
                when set_dat1 =>  rs_dat<='1'; ena_dat<='0'; hab_env<='1';
                when env_dat1 =>  rs_dat<='1'; ena_dat<='1'; hab_env<='1';
                when valida_dat1 =>  rs_dat<='1'; ena_dat<='0'; hab_env<='0';
        end case;
end process;

process(dato,aux_com,rs_dat,rs_com,ena_dat,ena_com,hab_com,hab_dat)
begin
        if (hab_com='1' and hab_dat='0') then
                valor<=aux_com; rs<=rs_com; ena<=ena_com;
        elsif (hab_com='0' and hab_dat='1') then
                valor<=dato; rs<=rs_dat; ena<=ena_dat;
        end if;
end process;

end algo;
```

Bibliografía

1. Fernando Pardo Carpio y José A. Boluda Grau. *VHDL Lenguaje para síntesis y modelado de circuitos*. 2da edición. Departamento de Informática. Universidad de Valencia.

2. Volnei A. Pedroni. *Circuit Design with VHDL*. Cambridge, Massachussets y London, England.

3. John W. Lockwood. *Logic síntesis for Field Programmable Gate Array (FPGA) tecnology*. 2003
4. MIchael Santarini. *High noon for FPGAs: Low-cost-versushigh- end showdown*. Noviembre 8 del 2007.
   http://www.edn.com/contents/images/6495296.pdf

5. *Programming Actel Devices*
   http://avmaster.bnx.homelinux.net/datasheets/Actel-ProgrammingGuide.pdf

6. Altera Programming Hardware data sheet

7. ByteBlaster II Download Cable User Guide

8. Cyclone II Device family Handbook
   http://www.altera.com/literature/hb/cyc2/cyc2_cii5v1.pdf

9. Cyclone II FPGA Starter Development Kit User Guide
   http://www.altera.com/literature/ug/ug_cii_starter_board.pdf

10. EASILY APPLICABLE GRAPHICAL LAYOUT EDITOR (EAGLE) Manual Version 5.

11. Epson: SED1278FData sheet

12. LatticeECP3 Family Data Sheet

13. Nacional Semiconductor: DAC0800 y ADC0802 Data sheet

14. On Semiconductor: MC14504B Data sheet

15. Quartus II Introduction Using VHDL Design

16. Serial Configuration Devices (EPCS1, EPCS4, EPCS16, EPCS64, and EPCS128) Data Sheet

17. Telcom Semiconductor, Inc: TC7660CPA Data sheet