PLACA IIE-CYCLONE

Proyecto de Fin de Carrera

Andrés Bergeret
Gabriel Colombo
Guillermo Di Maio

Tutor
Juan Pablo Oliver

Universidad de la Republica
Facultad de Ingeniería
Montevideo, Uruguay
Abril de 2007

Agradecimientos:

Al apoyo incondicional de nuestras familias, Juan Pablo Oliver, Sebastián Fernández, Ignacio Ramírez, Julio Pérez , Mario Vignolo, Fiorella Haim, Alberto Bergeret, Lucía Rodríguez, Maria Eugenia Rodríguez, David A Clunie, Magdalena Scoffone, Alfonsina Daguerre, Isabella Ceraolo, la Comisión de Enseñanza del IIE, Altera, Texas Instruments, Instituto de Ingeniería Eléctrica.

Muchas Gracias!!

1 TABLA DE CONTENIDOS

1.1 Tabla de Contenidos:

1.2 Indice de figuras:

1.3 Indice de tablas:

1.4 Contenido del CD:

Requerimientos del sistema:
- Procesador: Pentium 466 MHz o superior
- Memoria RAM: 128 MB o superior
- Video: 800x600 o superior
- CD-ROM: 16X o superior
- Sistema Operativo: Windows 98 o superior para utilitarios
- Software: Acrobat Reader 5.0 o superior para archivos pdf, MS Office 97 o superior u OpenOffice 1.1 o superior para archivos .doc .xls y .ppt
- Otros: GCC para compilar el software de compresión de imágenes Protel DXP o posterior para editar los esquemáticos

Contenidos del CD-ROM:

```
--> Diseños VHDL
      |
      --> leds.rar
      |
      --> loco-i_iie-cyclone.rar
      |
      --> salida_232.rar
      |
      --> segundero.rar

--> Documentación General
      |
      --> Altera
      |     |
      |     --> Configuration - Software Settings.pdf
      |     |
      |     --> Cyclone II Handbook 2006.pdf
      |     |
      |     --> EP2C20Q240 pinout.xls
      |     |
      |     --> Serial Configuration Devices.pdf
      |
      --> Camera Link
      |     |
      |     --> CameraLink - Español.pdf
      |     |
      |     --> CameraLink v1_13.pdf
      |
      --> Loco-I
      |     |
      |     --> JPEG-LS - Software Implementation.pdf
      |     |
      |     --> Loco-I - Introduction.pdf
      |     |
      |     --> Loco-I - NASA Implementation.pdf
      |     |
      |     --> Loco-I into JPEG-LS.pdf
      |
      --> Micron
            |
            --> Micron - 128MbSDRAMx32.pdf
```

```
--> Documentación Placa IIE-Cyclone
      |
      --> Diseño Esqueleto
      |     (carpeta con archivos de Quartus II de un diseño
      |     esqueleto para la placa IIE-PCI)
      |
      --> Esquemáticos
      |     (carpeta con todos los esquemáticos para Protel DXP)
      |
      --> Gerbers
      |     (carpeta con todos los archivo gerber)
      |
      --> Artículo Proyecto Placa IIE-Cyclone.pdf
      |     (breve artículo con las características principales)
      |
      --> Asignación de pines Placa IIE-Cyclone.xls
      |     (planilla con asignación de los pines del FPGA)
      |
      --> Documentación Proyecto Placa IIE-Cyclone.pdf
      |     (documentación completa de la placa)
      |
      --> Manual Placa IIE-Cyclone.pdf
      |     (manual de uso de la Placa IIE-Cyclone)
      |
      --> Presentación Proyecto Placa IIE-Cyclone.ppt
            (diapositivas con la presentación del Proyecto)


Software
      |
      --> jpeg-ls - UBC v1.1
      |     (carpeta con el código C para codificación y
      |     decodificación de JPEG-LS desarrollado por la University
      |     of British Columbia)
      |
      --> loco-i_iie-cyclone
            (carpeta con el código C de los programas para
            codificación y decodificación del algoritmo LOCO-I
            implementado en el Proyecto Placa IIE-Cyclone)


Utilitarios (freeware)
      |
      --> HxDBetaen.zip
      |     (HxD: software para editar y comparar archivos
      |     hexadecimales)
      |
      --> iview400_setup.exe
      |     (IrfanView: software para visualizar imágenes de varios
      |     formatos)
      |
      --> Terminal.exe
      |     (Terminal: software para comunicación por Puerto Serie)
      |
      --> viewmate9_681.exe
            (ViewMate: software de visualización de archivos gerber)
```

2 INTRODUCCIÓN

2.1 Resumen:

Nuestro Proyecto consistió en el desarrollo de una plataforma basada en lógica reconfigurable y de un algoritmo de compresión de imágenes sin pérdida para implementar en ella.

La placa IIE-Cyclone fue diseñada para ser utilizada como una plataforma de desarrollo genérica. El procesamiento se basa en un FPGA Cyclone II de Altera, y cuenta también con una memoria SDRAM, una interfaz serial RS232 y otra del tipo Camera Link para comunicarse con otros dispositivos, así como pulsadores, dip switches y LEDs.

El algoritmo de compresión de imágenes sin pérdida que implementamos está basado en el algoritmo LOCO-I y fue desarrollado en lenguaje VHDL de diseño de hardware. Al LOCO-I le realizamos varias modificaciones para mejorar su desempeño en un FPGA y optimizar los recursos del mismo. Trabajamos con imágenes de 100 x 100 píxeles en escala de grises, utilizando solamente un 5% del chip.

Los resultados logrados fueron similares a los de otras implementaciones en hardware e incluso en software. Alcanzamos una tasa de compresión en el entorno de 35 %, pudiendo procesar 1.88 Mpíxeles/seg.

2.2 Introducción:

El Proyecto Placa IIE-Cyclone se divide en dos etapas; la primera consistió en el diseño y desarrollo de una plataforma de lógica programable y una segunda etapa basada en el desarrollo de una aplicación sobre la misma.
Nuestro desarrollo es una plataforma genérica, lo que permite su reutilización en un sinnúmero de aplicaciones. Con respecto a la aplicación implementamos un algoritmo de compresión sin pérdida para imágenes en escala de grises de 8 bpp (bits/píxel).

Fue nuestro interés realizar el Proyecto de Fin de Carrera tomando parte en todas las etapas del desarrollo de la plataforma, lo cual implica el diseño, elección de componentes y el montaje de los mismos.
Elegimos este enfoque frente a la posibilidad de desarrollar una aplicación de diseño de hardware en una plataforma ya existente, propuesta inicial de nuestro tutor Juan Pablo Oliver, a pesar del costo económico que tuvimos que enfrentar.

Nuestra plataforma cuenta con un FPGA Cyclone II de la familia Altera y puede operar en forma stand-alone o en forma integrada con la Placa IIE-PCI. En esta configuración se utiliza la Placa IIE-PCI para la comunicación con la PC a través del bus PCI y la Placa IIE-Cyclone para el procesamiento

Para esta etapa del Proyecto contamos con el apoyo del grupo de Electrónica Aplicada del IIE quienes cuentan con una amplia experiencia en el diseño de placas de lógica programable y en la implementación de aplicaciones de lógica programable. En este marco en el año 2003 se realizó como Proyecto de Fin de Carrera la Placa IIE-PCI[14], la misma fue diseñada por Ciro Mondueri y Sebastián Fernández. Esta cuenta con un FPGA ACEX 1K100 de Altera y tiene una interfaz PCI. Fue diseñada como plataforma de pruebas para desarrollar aplicaciones sobre dicho bus.
Desde el momento de su diseño la Placa IIE-PCI viene siendo utilizado con muy buen desempeño tanto por docentes como por estudiantes del IIE para probar distintos tipos de aplicaciones sobre su FPGA.

Para compatibilizar ambas placas fue necesario adaptar nuestro diseño a las características de la Placa IIE-PCI. La conexión se realizó mediante los 2 conectores de expansión con los que cuenta la placa IIE-PCI, esto nos determinó no sólo la distancia entre los conectores de expansión en nuestra placa, sino también el posicionamiento de todos los componentes y la estructura general de la misma. Además de los requisitos de estructura previamente mencionados la interconexión entre ambas placas también nos influyó en la determinación de los niveles de voltaje de operación, el acceso a memoria y el manejo de la señal de reloj. Con respecto a este modo de operación dentro del marco de nuestro proyecto probamos la correcta interconexión física entre ambas placas.

También nos pareció importante que la placa IIE-Cyclone pudiera operar en modo stand-alone. Debido a esto fue necesario dotar a la placa de

conexiones de alimentación, de entrada y salida de datos y de una RAM propia.

Con respecto a la entrada y salida de datos, optamos por dotar a la placa con 2 interfaces distintas. La primera, una interfaz serial RS-232 la que nos permite una fácil comunicación con un PC. La segunda una interfaz Camera Link [26] que utiliza señales del tipo LVDS [27] (Low Voltage Differential Signaling). Estas son señales diferenciales de bajo voltaje, lo que permite buena inmunidad al ruido y poder alcanzar altas tasas de transferencia. El propósito de esta interfaz es brindarle la capacidad de desarrollar sobre la placa aplicaciones de procesamiento de imágenes y video.

Fue interés del grupo de Electrónica Aplicada implementar un algoritmo de compresión de imágenes sin pérdida como aplicación para implementar sobre nuestra plataforma. Como algoritmo seleccionamos el LOCO-I [30], dada la experiencia existente dentro del grupo de Procesamiento de Imágenes del IIE trabajando sobre el mismo y la interacción de ellos con los creadores del algoritmo. Nos decidimos por una implementación de este algoritmo realizada por el Jet Propulsion Laboratory del California Institute of Technology a pedido de la NASA [35], pensado para comprimir imágenes tomadas por sondas espaciales. Elegimos esta variante del algoritmo dado que tiene características específicamente pensadas para su implementación en lógica programable.

Resumiendo, las principales características de la plataforma desarrollada son:

- FPGA EP2C20Q240C8 Cyclone II de Altera.
- Memoria SDRAM de 128 Mbit.
- Memoria de configuración EPCS4.
- Buses de expansión compatibles con los de la Placa IIE-PCI.
- Interfaces Camera Link y serial RS-232.

3 HARDWARE

3.1 Descripción general de la placa:

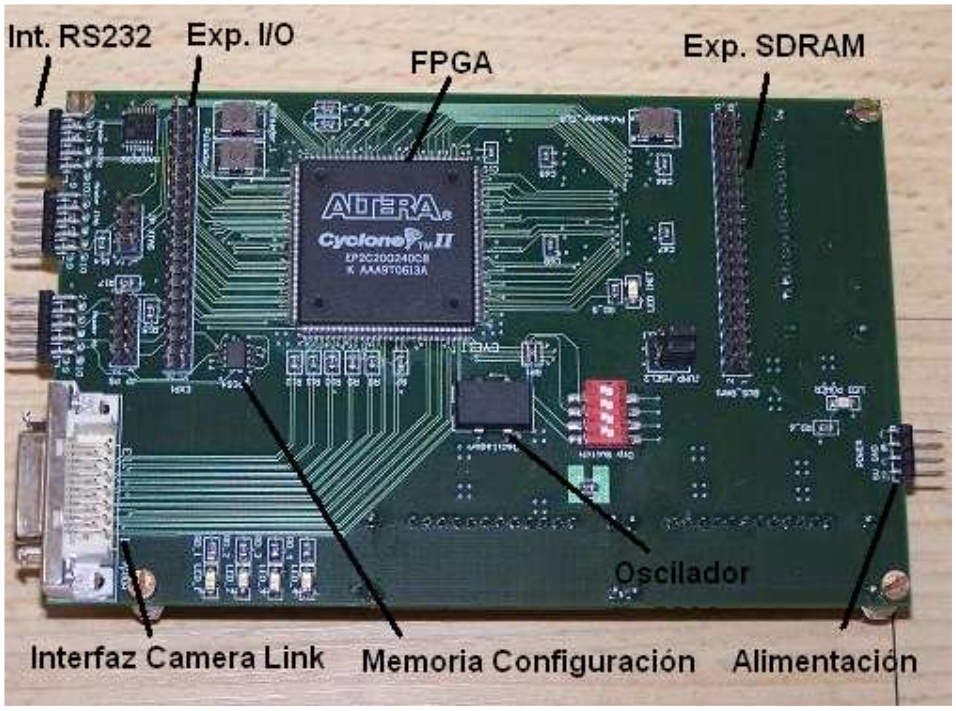


Figura 1: Foto lado superior Placa IIE-Cyclone

La Placa IIE-Cyclone es una placa pensada y fabricada para cumplir la función de plataforma de pruebas para aplicaciones que se monten sobre su FPGA. Asimismo se diseñó para que fuera compatible con la Placa IIE-PCI [14] y de esta manera poder aumentar la capacidad de procesamiento de la misma, ya que cuenta con un FPGA ACEX, también de Altera, de menor cantidad de elementos lógicos.

Está fuertemente orientada a aplicaciones que trabajen con video digital ya que posee una interfaz del tipo Camera Link [26]. Dicha interfaz ha sido desarrollada especialmente para la comunicación con cámaras de video digital.

El FPGA utilizado en la placa es un Cyclone II EP2C20Q240C8 [3] de Altera; el mismo dentro de sus principales características cuenta con 120 pines de entrada/salida multipropósito, 18752 elementos lógicos y cuatro PLLs internos.

La Placa IIE-Cyclone también cuenta con una memoria on-board del tipo SDRAM de 128 Mbits [19]. Se tomó la decisión de dotar a la placa con dicha memoria ya que el tamaño de la memoria interna del FPGA puede llegar a

quedar chica (ver capítulo 3.3.1) si se quiere trabajar con aplicaciones de imagen o video, o cualquier otro tipo de aplicación que requiera mucha capacidad de almacenamiento de datos y accesos rápidos a memoria.

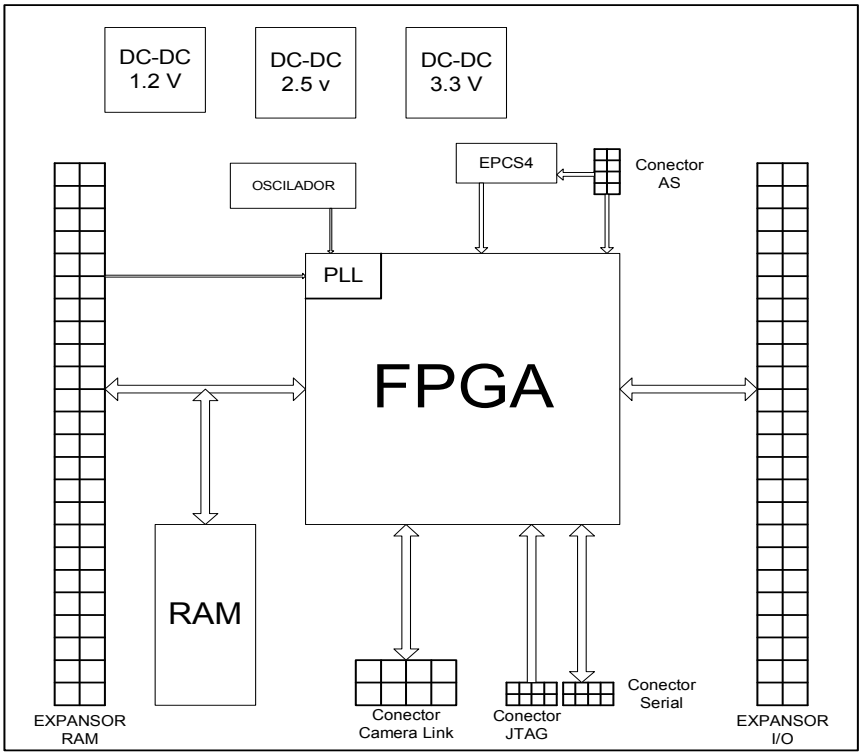


Figura 2: Diagrama de bloques Placa IIE-Cyclone

Para que la Placa IIE-Cyclone pudiera trabajar conectada con la Placa IIE-PCI fue necesario colocarle 2 buses de expansión que fueran compatibles con los que ya cuenta la placa IIE-PCI. Dichos buses se encuentran en la cara superior de la Placa y cada uno de los mismos tiene 40 pines.
Además de señales de configuración y reloj estos buses interconectan entre sí pines I/O de los FPGA de ambas placas, y las señales de datos de las SDRAM también de ambas placas.

La señal de reloj de la placa puede ser generada por el oscilador de la misma u obtenida desde la Placa IIE-PCI a través de un pin del bus de expansión. La frecuencia de éstas puede ser multiplicada o dividida por uno de los 4 PLL con que cuenta el Cyclone II [5]. Luego esta señal, generada por el PLL, es utilizada como señal de reloj para el resto de la placa, particularmente para las aplicaciones del FPGA y la SDRAM.

Como los FPGA no cuentan con una memoria no volátil se le agregó a la placa un dispositivo de configuración serial modelo EPCS4 de Altera [17,18] que cuenta con una memoria de 4Mb, con la misma se evita tener que configurar el FPGA cada vez que se enciende.

Para la programación de la memoria de configuración se utiliza un protocolo propietario de Altera denominado Active Serial (AS). En este protocolo se descarga desde la PC a la memoria el archivo de configuración y luego el FPGA controla la recepción desde el dispositivo de configuración de los datos. En el caso que se quiera configurar solamente el FPGA se puede realizar utilizando el protocolo JTAG [2].

Además se puede configurar al FPGA mediante protocolo Passive Serial. En este protocolo, también propietario del fabricante, un Dispositivo Externo (ej.: un microprocesador o CPLD) controla el envío de los datos de configuración al FPGA. Esta opción es solo válida en nuestra placa cuando ésta se encuentra conectada a la Placa IIE-PCI, ya que se utiliza al FPGA ACEX como inteligencia exterior para que controle la configuración. Dentro del marco de nuestro proyecto no implementamos esta solución pero el diseño se realizo para que este método de configuración fuera viable.

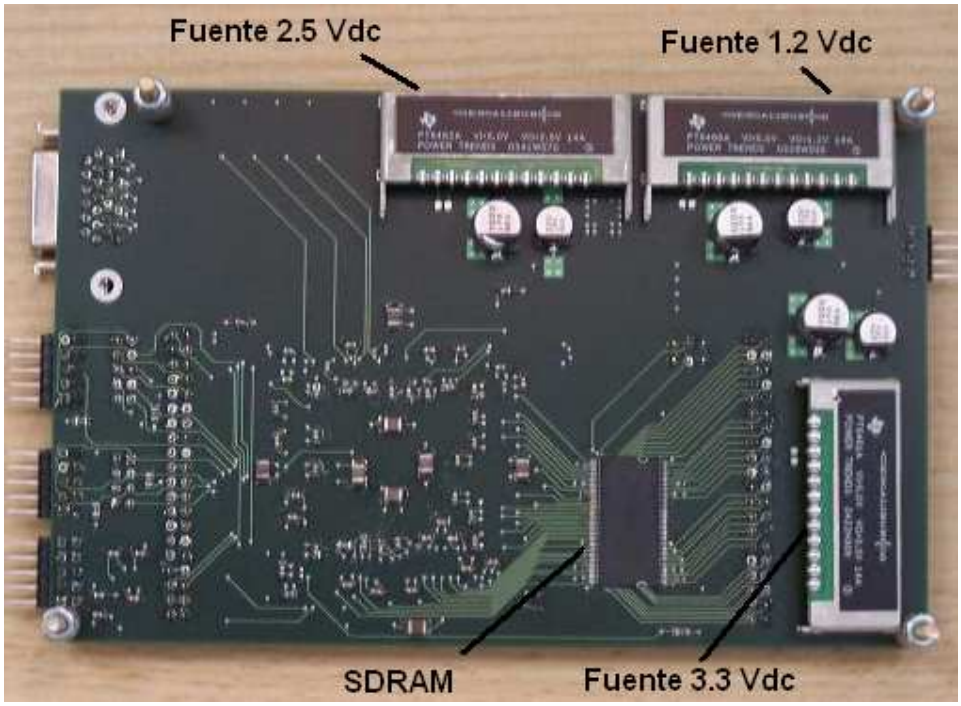


Figura 3: Foto lado inferior Placa IIE-Cyclone

Además de los buses de expansión previamente mencionados se le agregó a la Placa IIE-Cyclone dos interfaces para ampliar las opciones de conexión con el exterior.

La primera es una interfaz Camera Link que utiliza un conector del tipo MDR-26 de 3M y señales del tipo LVDS. Es una interfaz diferencial de alta velocidad diseñada especialmente para comunicarse con cámaras de video digital. Utilizamos la configuración básica de Camera Link; la misma posee 4 señales para control de la cámara, 4 señales de datos desde la cámara al

FPGA, una señal de reloj y dos señales seriales, una desde la cámara al FPGA y la otra en sentido contrario.

La segunda es una interfaz serial RS-232. Para implementarla utilizamos un chip MAX3232 que se encarga principalmente de adaptar los niveles de voltaje a la salida de los pines del FPGA a los niveles de voltaje especificados en el estándar RS-232.

Para poder probar aplicaciones sencillas decidimos conectar a pines de entrada y salida del FPGA una serie de pulsadores, dip switches y LEDs ya que mediante los mismos se puede realizar dichas pruebas de una manera sencilla y práctica.

Con respecto a la estructura física de la placa al diseñarla decidimos que la misma tuviera 4 capas; las dos exteriores de señal, una de las interiores de tierra y la restante que contuviera los 4 planos distintos de voltaje (5 Vdc, 1.2 Vdc, 3.3 Vdc y 2.5 Vdc).

Optamos por dotar a la placa de dos planos de señal ya que teniendo en cuenta la cantidad de componentes y las distintas conexiones entre los mismos (pistas) se iba a tornar imposible rutearlas todas en una sola capa.
Nos basamos en el diseño de otras placas y en bibliografía especializada en el tema [8, 9,10, 11,12, 13] para definir qué posiciones iba a ocupar cada una de las capas. Se recomienda que las capas exteriores sean siempre de señal, que una de las interiores sea de tierra (para que siempre se tenga un camino lo más corto posible hacia dicho plano) y que en la medida de lo posible se tenga un plano completo para los distintos niveles de voltaje.

La placa se alimenta mediante un conector del tipo Mini 2.54 mm. El mismo debe ser conectado tanto cuando la placa trabaja de forma independiente o cuando trabaja conectada a la Placa IIE-PCI. La alimentación general recibida es de 5 Vdc. Luego, a partir de la alimentación de 5 Vdc y utilizando 3 conversores DC-DC se logran los niveles de voltaje necesarios para alimentar los distintos componentes de la placa.

3.2 Principales características de la placa:

- FPGA Cyclone II EP2C20Q240C8 de ALTERA.
- SDRAM 128 Mb MT48LC4M32B2 de MICRON.
- Dispositivo de configuración serial de 4 Mb EPCS4 de ALTERA.
- Oscilador de 25 MHZ CMX309FBC25.000MTR de CITIZEN AMERICA CORPORATION.
- PLL: se utiliza uno de los cuatro PLL internos del FPGA.
- Conversor 5 Vdc/3.3 Vdc PT6461 de TEXAS INSTRUMENTS.
- Conversor 5 Vdc/2.5 Vdc PT6462 de TEXAS INSTRUMENTS.
- Conversor 5 Vdc/1.2 Vdc PT6466 de TEXAS INSTRUMENTS.
- 4 LEDs multipropósito, 1 LED indicador de alimentación y un LED indicador de configuración.
- 3 Pulsadores.
- DIP SWITCH 4 llaves.
- Interfaz Camera Link.
- Interfaz RS-232.
- Buses de expansión compatibles con los de la Placa IIE-PCI:
    - 26 señales I/O del Cyclone II.
    - 32 señales de datos de la SDRAM de nuestra placa.
    - Señales de configuración JTAG y Passive Serial.
    - Señal de reloj.
- Configuración del FPGA mediante:
    - Cadena JTAG con la Placa IIE-PCI.
    - JTAG utilizando conector en la placa.
    - Passive Serial utilizando el FPGA ACEX de la Placa IIE-PCI como external host.
    - Active Serial utilizando el EPCS4 que configura al FPGA al alimentar la placa.
- Conector de alimentación externa.

3.3 Descripción de los bloques:

A continuación se detallan los distintos componentes y bloques anteriormente mencionados que fueron implementados en la placa. Se muestra una descripción general de cada uno, la motivación para su implementación, las distintas opciones que manejamos previo a su desarrollo y un breve análisis de los mismos.

3.3.1 FPGA:

Para elegir qué FPGA utilizaríamos en nuestra placa tomamos en cuenta los siguientes factores:

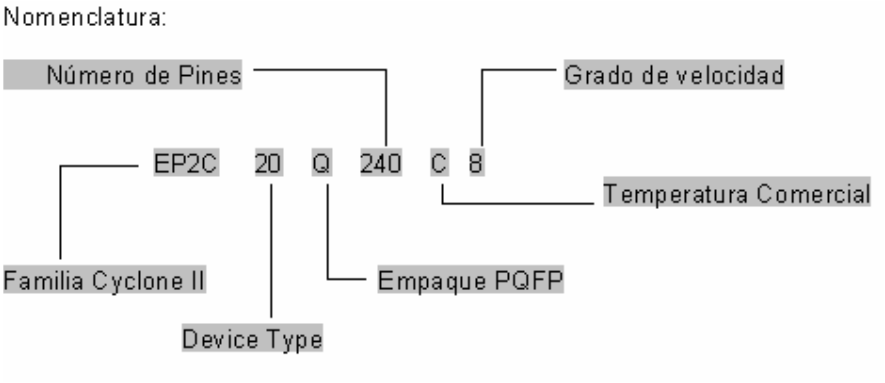
1) Que fuera de la marca Altera, ya que el FPGA presente en la Placa IIE-PCI es de la misma marca y de esta manera tratamos evitar posibles problemas de compatibilidad. Además se contaba con experiencia en el manejo del software de diseño Quartus II [41] y en trabajar con FPGAs de dicha compañía.

2) Tenía que ser un modelo de bajo costo pero que a su vez fuera de los últimos en salir al mercado. Ya que se iba a solicitar su donación pero se quería contar con un FPGA con buena capacidad.

3) Que tuviera un encapsulado que permitiera poder soldarlo a la placa sin la necesidad de herramientas muy sofisticadas.

4) Que pudiera manejar señales del tipo LVDS, ya que se quería dotar a la placa de una interfaz Camera Link para cámaras de video digital.

Nos terminamos decidiendo por utilizar un FPGA de Altera de la familia Cyclone II, específicamente, se utilizó el modelo EP2C20Q240C8.
La donación de los mismos fue tramitada por Juan Pablo Oliver y la empresa Altera nos donó 6 unidades.
La familia de FPGAs Cyclone II de Altera es la segunda generación de la familia Cyclone y fue desarrollada para brindarle al mercado una opción de FPGAs con una gran cantidad de elementos lógicos pero que a su vez tenga bajo costo y bajo consumo de potencia. Es compatible con el Software de diseño Quartus II de Altera.



El rango de temperatura comercial es de: Tj 0°C a 85°C.
El grado de velocidad hace referencia al retardo en nanosegundo en una macrocelda del integrado (8ns → 125MHz).

Figura 4: Especificación nomenclatura familia Cyclone II

Esta familia presenta diversas variedades de encapsulado, dependiendo éstas de la cantidad de elementos lógicos del mismo y de la cantidad de pines. La línea EP2C20, que cuenta con cerca de 20,000 elementos lógicos (LE) presenta como posibilidades de encapsulado: PQFP de 240 pines, FBGA de 256 y FBGA de 484 pines.
En el encapsulado FBGA (Fine Ball Grid Array) los pines se encuentran distribuidos en una grilla ocupando toda la cara inferior del integrado. En cambio, en el PQFP (Plastic Quad Flat Pack), los 240 pines se encuentran en el perímetro del integrado, permitiéndonos esto soldarlo nosotros con los instrumentos de los que disponíamos.

Modelo	LEs (*)	PLLs	Bloques de Memoria (**)
EP2C5	4608	2	26
EP2C8	8256	2	36
EP2C20	**18752**	**4**	**52**
EP2C35	33216	4	105
EP2C50	50528	4	129
EP2C70	68416	4	250

(*) LEs: Elementos lógicos, unidad lógica más pequeña dentro de la estructura del FPGA
(**) Bloques de Memoria: Cada bloque de memoria contiene 4608 RAM bits (incluyendo bits de paridad).

Tabla 1: Principales Características de la Familia Cyclone II

Los pines de entrada/salida del Cyclone II están agrupados en 8 bancos distintos. Cada uno de estos bancos tiene pines de alimentación de entrada y salida independientes (VCCIO). Esto quiere decir que cada banco puede manejar distintos niveles de voltaje de entrada y salida. En la tabla 2 se especifican los distintos estándares soportados por cada uno de los bancos del Cyclone II.
Los niveles de voltaje que puede manejar son: 1.5V, 1.8V, 2.5V, 3.3V.

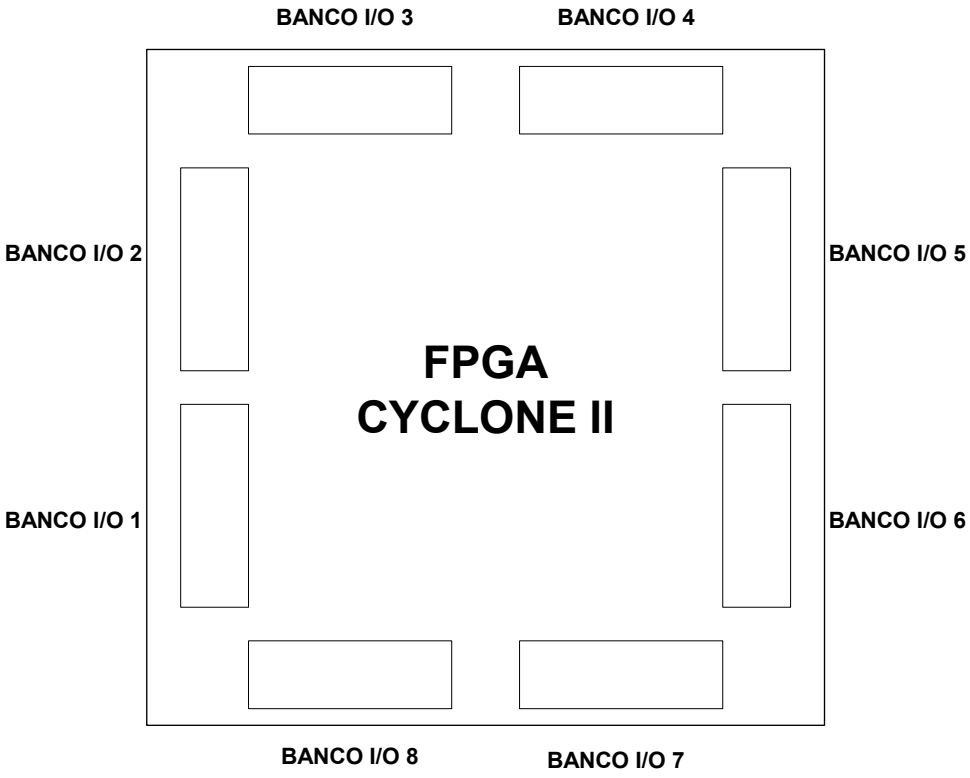


Figura 5: Representación gráfica de la ubicación de los bancos

	BANCOS							
Estándar I/O	1	2	3	4	5	6	7	8
LVTTL	√	√	√	√	√	√	√	√
LVCMOS	√	√	√	√	√	√	√	√
2.5 V	√	√	√	√	√	√	√	√
1.8 V	√	√	√	√	√	√	√	√
1.5 V	√	√	√	√	√	√	√	√
3.3 V PCI	√	√			√	√		
3.3 V PCI X	√	√			√	√		
SSTL-2 class I	√	√	√	√	√	√	√	√
SSTL-2 class II	√	√	√	√	√	√	√	√
SSTL-18 class I	√	√	√	√	√	√	√	√
SSTL-18 class II			√	√			√	√
1.8 V HSTL Class I	√	√	√	√	√	√	√	√
1.8 V HSTL Class II			√	√			√	√
1.5 V HSTL Class I	√	√	√	√	√	√	√	√
1.5 V HSTL Class II			√	√			√	√
LVDS	√	√	√	√	√	√	√	√

Tabla 2: Estándares soportadas por los bancos del Cyclone II

3.3.2 Memoria:

La memoria utilizada en la placa es una SDRAM de 128 Mbit modelo MT48LC4M32B2 de la empresa MICRON [19].

Se quiso dotar a la placa de una memoria On Board para poder brindarle independencia con respecto a la Placa IIE-PCI y que pudiera ser operativa sin necesidad de estar conectada a la misma.

Con respecto a la elección de la memoria optamos por la MT48LC4M32B2 ya que fue utilizada tanto en la placa IIE-PCI (ya mencionada y presentada anteriormente) como en la Placa Virtex, también desarrollada como Proyecto de Fin de Carrera en el IIE por Silvia Gómez, Jimena Saporiti y Agustín Villavedra.
Además se cuenta con un Controlador de SDRAM Wishbone Compatible desarrollado como Proyecto de Diseño Lógico 2 en el año 2003 por Jimena Saporiti y Agustín Villavedra[20], el cual fue pensado para interactuar con esta memoria.

Sumadas a las ventajas previamente mencionadas, ya existían en el IIE tres memorias MT48LC4M32B2 donadas previamente por la empresa MICRON.

Las principales características de la memoria utilizada son:

1) Encapsulado TSOP de 86 pines.
2) Capacidad: 128 Mbits.
3) Cuatro bancos de 33.554.432 bits, cada uno organizado en 256 columnas por 4.096 filas por 32 bits.
4) Alimentación: 3.3 V.
5) Entradas y salidas LVTTL compatibles.
6) Velocidad máxima 166 MHz.

Luego de la elección de la memoria faltaba decidir como se resolvería el acceso a la misma cuando se estuviera trabajando conectado a la placa IIE-PCI.
Inicialmente surgieron dos opciones distintas:

Opción 1:

El bus de datos es compartido por ambos FPGA's y las RAM's. Dicho bus está accesible en el expansor de datos (ver figura 6).
El direccionamiento de las memorias es exclusivo de cada FPGA. Si se quiere escribir en la RAM de la placa IIE-Cyclone desde el FPGA de la placa IIE-PCI o viceversa, se debe implementar un multiplexor en cada FPGA que permita que los pines que comparte con el otro chip se mapeen en su bus de direcciones de RAM.

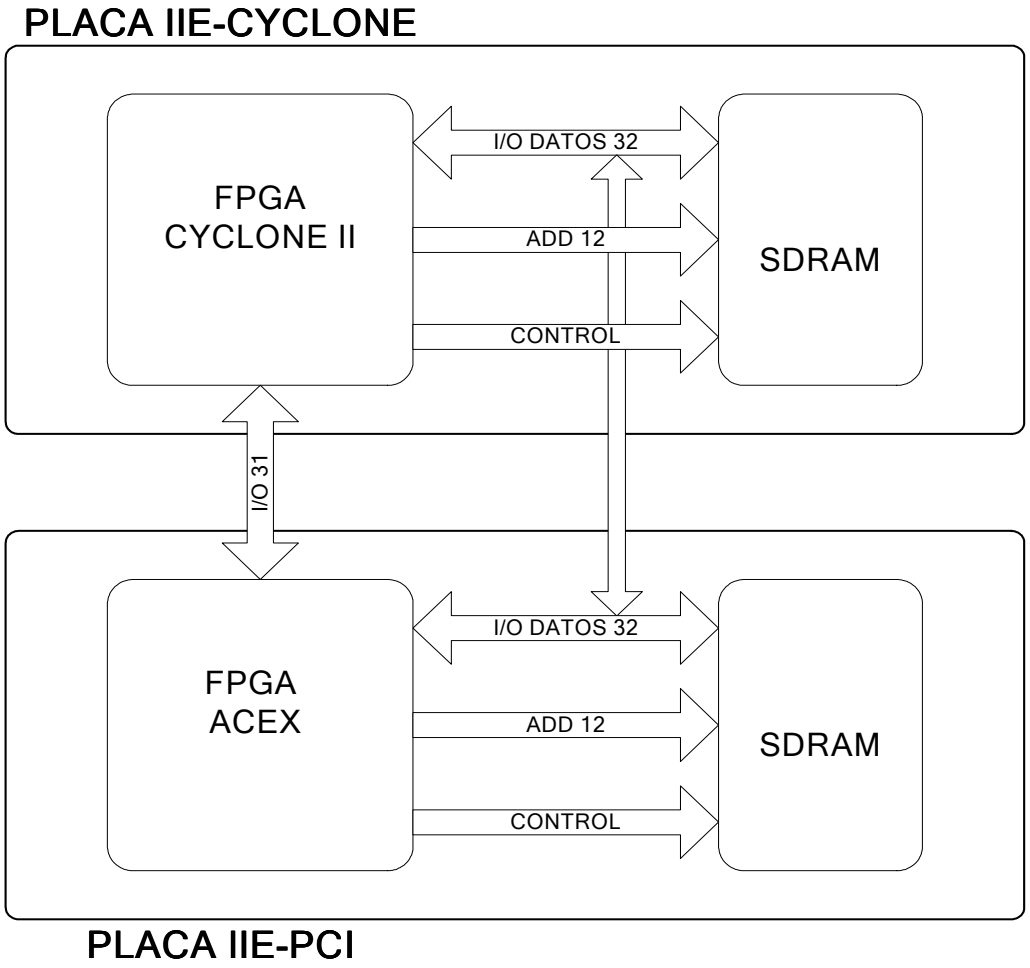


Figura 6: Opción 1, implementada para el acceso a memorias

Ventajas:

- Con las dos RAM funcionando nos ofrece la mayor cantidad posible de pines de comunicación entre ambos FPGA.

Opción 2:

En esta configuración también se comparte el bus de datos de las memorias. La RAM de la placa IIE-PCI solo puede ser direccionada por el ACEX.
Mientras que la RAM de la Placa IIE-Cyclone puede ser direccionada por ambos FPGAs.
Para la transmisión de datos entre los dos FPGA se deshabilitaría la RAM de la placa IIE-Cyclone y quedarían todos los pines del expansor de pines I/O como pines de intercomunicación, o se pueden comunicar pasándose los datos a través de las memorias.

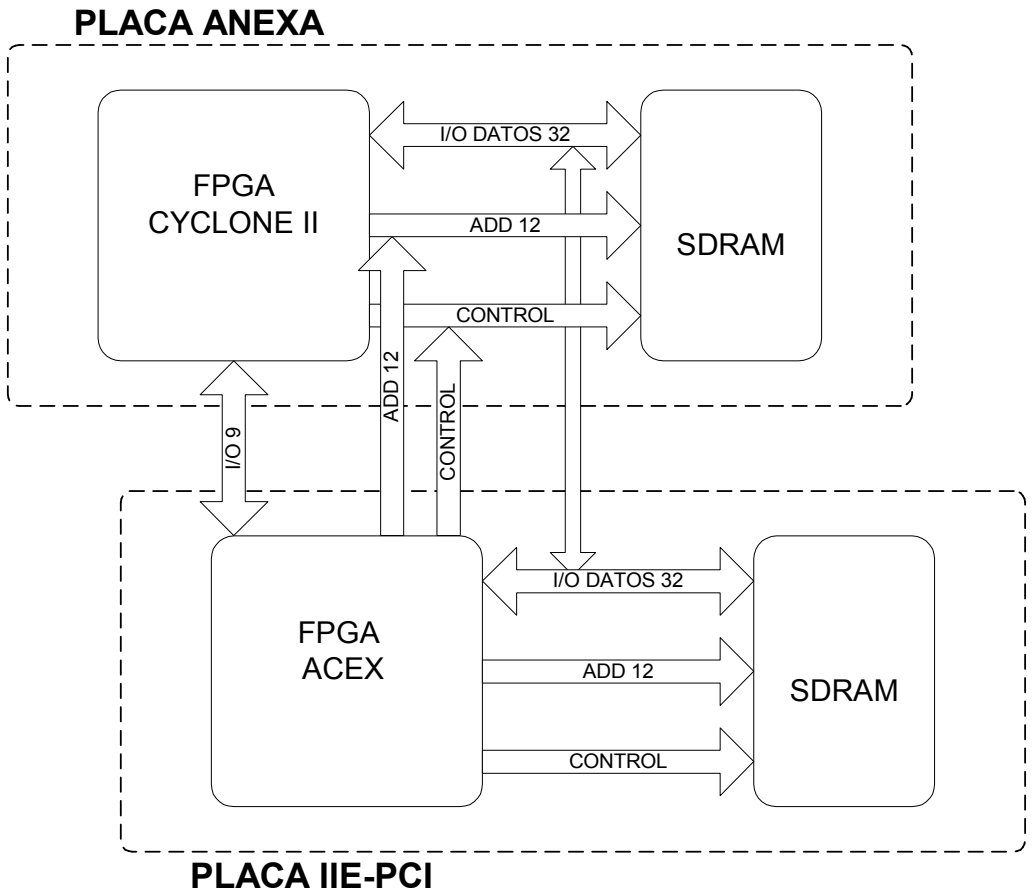


Figura 7: Opción 2 (no implementada) para el acceso a memorias

Ventajas:

- En esta configuración tenemos más pines de I/O del Cyclone II disponibles.
- El ACEX puede acceder a la RAM superior sin estar configurado el Cyclone. Gran utilidad en pruebas o para aumentar la capacidad de memoria de la placa IIE-PCI.

Tomando en cuenta las ventajas y desventajas de cada una se terminó optando por la opción número 1.

3.3.3 Expansores:

Para poder trabajar en conjunto con la Placa IIE-PCI fue necesario colocarle a nuestra placa 2 expansores Macho de 2.54 mm, de 40 pines cada uno, distribuidos en 2 filas de 20 pines.

La distancia entre ambos expansores y el lugar donde fueron colocados en la placa quedaron determinados por el posicionamiento de los expansores de la Placa IIE-PCI.

Inicialmente se había decidido que los expansores de nuestra placa fueran hembra e implementaríamos la interconexión utilizando pines doble macho. Esta opción aumentaba la distancia entre ambas placas lo que facilitaría el posicionamiento de los componentes en nuestra placa, ya que no tendríamos que tener en cuenta la altura de los componentes en la Placa IIE-PCI.

Terminamos desechando esta opción ya que quisimos evitar posibles falsos contactos debido al doble contacto que se implementaría al utilizar pines doble macho para la conexión.

En su lugar decidimos implementar la conexión utilizando pines macho lo que nos da un mejor contacto pero nos obliga a tener mucho más cuidado en la etapa de posicionamiento de los componentes.

Se conectaron al expansor de I/O los pines del Cyclone II que se utilizan para la configuración Passive Serial. Esto se realizó para poder configurar al Cyclone II desde la Placa IIE-PCI utilizando el FPGA ACEX como Dispositivo Externo de configuración, es decir, quien controla el envío de datos en forma activa.
También se conectaron los pines de configuración JTAG para poder cerrar la cadena JTAG y habilitar la opción de poder configurar ambos FPGA vía dicho protocolo. Por más información al respecto consultar el Manual del Usuario Placa IIE-Cyclone.

En los expansores de la Placa IIE-PCI hay dos pines de alimentación: uno de 5V y el otro de 3.3V. Se decidió no conectarlos a los planos de voltaje para evitar posibles problemas térmicos en los pines en caso de circulación de corrientes muy altas (mayores a 1 A). En el caso de que ambas placas estén trabajando juntas simplemente hay que utilizar el conector de alimentación con el que cuenta la Placa IIE-Cyclone.

Uno de los pines del expansor de la Placa IIE-PCI transmite la señal de reloj proveniente del PLL de la misma; dicho pin se conectó a una de las dos entradas del PLL interno del Cyclone II que utilizamos.

Luego, para decidir qué señales conectar al resto de los pines de los expansores se tomó en cuenta la forma en la que se implementó la solución para el manejo de las memorias; dicha solución se especifica en el Capítulo 3.3.2 Memoria.

Llamamos BUS_RAM1, al que tiene conectados los pines de la RAM (se encuentra claramente identificado en la placa) y EXP1 al que tiene conectadas en su mayoría pines I/O del Cyclone II (también se encuentra identificado en la placa). Para ver los pines correspondientes en los expansores de la Placa IIE-PCI ver documentación de la Placa IIE-PCI [14].
Hay que tener en cuenta que la numeración de los pines se encuentra espejada con la numeración de los pines de la Placa IIE-PCI, es decir pin 1 de la Placa IIE-Cyclone corresponde al pin 2 de la Placa IIE-PCI, y así sucesivamente.

PIN Expansor	Señal (Pin Cyclone II)	PIN Expansor	Señal (Pin Cyclone II)
1	DQ0 (125)	21	DQ31(195)
2	GND	22	GND
3	DQ2 (134)	23	DQ29(203)
4	DQ1 (131)	24	DQ30(199)
5	DQ4(139)	25	DQ27(218)
6	DQ3(136)	26	DQ28(214)
7	DQ6(150)	27	DQ25(226)
8	DQ5(141)	28	DQ26(223)
9	GND	29	GND
10	DQ7(156)	30	DQ24(230)
11	DQ15(130)	31	DQ16(194)
12	GND	32	GND
13	DQ13(135)	33	DQ18(200)
14	DQ14(132)	34	DQ17(197)
15	DQ11(140)	35	DQ20(216)
16	DQ12(137)	36	DQ19(208)
17	DQ9(155)	37	DQ22(228)
18	DQ10(149)	38	DQ21(222)
19	GND	39	GND
20	DQ8(157)	40	DQ23(231)

Tabla 3: Pinout del expansor BUS_RAM1

PIN Expansor	Señal (Pin Cyclone II)	PIN Expansor	Señal (Pin Cyclone II)
1	GND	21	I/O(41)
2	No Conectada	22	I/O(42)
3	CLK_EXP(95)	23	I/O(38)
4	GND	24	I/O(39)
5	DCLK	25	I/O(21)
6	No Conectada	26	I/O(37)
7	DATA	27	I/O(18)
8	NCSO(5)	28	I/O(20)
9	OE	29	I/O(15)
10	NINIT	30	I/O(16)
11	I/O(57)	31	I/O(13)
12	I/O(58)	32	I/O(14)
13	I/O(55)	33	I/O(9)
14	I/O(56)	34	I/O(11)
15	I/O(51)	35	I/O(7)
16	I/O(52)	36	I/O(8)
17	I/O(47)	37	JTAG_TDO(37)
18	I/O(50)	38	JTAG_TDI(38)
19	I/O(44)	39	JTAG_TCK(39)
20	I/O(46)	40	JTAG_TMS(40)

Tabla 4: Pinout del expansor EXP1

3.3.4 Interfaces:

A continuación se detallan las dos interfaces que se desarrollaron en la placa, vale la pena volver a mencionar que las mismas fueron agregadas principalmente para darle la opción a la placa de tener contacto con PCs y otros dispositivos sin necesidad de estar conectada a la IIE-PCI.

3.3.4.1 Interfaz RS-232:

La Placa IIE-Cyclone cuenta con una interfaz del tipo serial RS-232. A través de la misma se logra tener un método sencillo y rápido tanto para el envió de datos para ser procesados por el FPGA como para, luego de su procesamiento, poder analizarlos.

Para su implementación se utilizó el chip MAX3232 [25]. Al mismo se le conectan 4 pines I/O del Cyclone II y 4 pines del conector serial macho acodado de 10 pines con el que cuenta la placa (identificado como Header RS232).

La función del MAX3232 es la de adaptar los niveles de voltaje de los pines I/O del FPGA con los niveles de voltaje especificados en la norma RS-232.

Para poder utilizar esta interfaz va a ser necesario desarrollar y cargar en el FPGA una aplicación que cumpla la función de UART (Universal Asynchronous Communications Element), es decir convertir datos en paralelo a datos en serie.

Dentro de la variedad de señales especificadas en el Protocolo RS-232, vamos a manejar las siguientes:

Serial_Tx: transmisión de datos en forma serial del FPGA

Serial_Rx: recepción de datos en forma serial hacia el FPGA

Serial_RTS (Request to Send): señal utilizada para el control de flujo

Serial_CTS (Clear to Send): señal utilizada para el control de flujo

Señal	Pin FPGA	Pin Conector
Serial _Tx	235	2
Serial_Rx	234	3
Serial_CTS	237	8
Serial_RTS	236	7
GND		5

Tabla 5: Pinout del FPGA y del Conector Serial

3.3.4.2 Interfaz Camera Link:

Camera Link es una interfaz de comunicación especialmente desarrollada para imagen y video digital [26]. Se desarrolló para simplificar y uniformizar la comunicación entre Cámaras de Video Digital y Frame Grabbers (placas de captura digital, que se encargan de la adquisición de la imagen para que luego sea procesada por un ordenador o por otro tipo de inteligencia).

Dentro de las especificaciones Camera Link fueron estandarizados los siguientes puntos:

- Conectores, tanto para cámaras como para los Frame Grabbers.
- El cable de conexión entre cámara y Frame Grabber
- Formato de imagen a ser trasmitido por la cámara
- Señales de control de la cámara
- Método de comunicación serial desde y hacia la cámara
- Chip Channel Link de recepción y transmisión de señales

Los chips Channel Link de transmisión especificados por el estándar cumplen con la característica de convertir 28 bits CMOS/TTL en 4 señales LVDS (Low Voltaje Differential Signaling) [28], mientras que los de recepción cumplen con la función inversa. Ambos chips manejan a su vez una señal de reloj.

LVDS, como su nombre lo indica, maneja señales de baja potencia, diferenciales y de alta velocidad. Al manejar señales de bajo voltaje se mejora el consumo de potencia, mientras que se utilizan señales del tipo diferencial para mejorar la inmunidad a interferencias electromagnéticas y de esta manera poder aumentar las tasas de transferencia.
Por especificaciones de la norma LVDS se colocó una resistencia de 100 ohm entre cada par diferencial en recepción. La misma tiene que ir lo más cerca posible del receptor, en nuestro caso el FPGA.

Específicamente el Cyclone II puede transmitir y recibir señales LVDS a una tasa de 640 Mbps y 805 Mbps respectivamente[6].

Agrupando los chips de transmisión y recepción previamente mencionados se generan las distintas configuraciones posibles dentro del estándar Camera Link. Las mismas son: Base, Medium y Full.

En la tabla 6 se especifican sus principales características.

Configuración	Señales diferenciales de datos	Cantidad de conectores
BASE	4	1
MEDIUM	8	2
FULL	12	2

Tabla 6: Configuraciones Camera Link

En la placa implementamos la configuración del tipo BASE; es la que brinda menor cantidad de señales de datos y por lo tanto la que tiene menor tasa de transferencia. Nos decidimos por esta configuración ya que es la utilizada por la mayoría de las cámaras digitales, es la que requiere menor cantidad de pines I/O del FPGA y además es la única de las tres que requiere un solo conector.

Sumadas a las señales de datos en la configuración BASE se determinan cuatro señales diferenciales para el control de la cámara, una para comunicación serial desde la cámara, otra para comunicación serial hacia la cámara y por último una señal diferencial para el reloj. En la tabla 7 se especifican claramente todas las señales involucradas dentro de la configuración básica.

Los pines del conector Camera Link fueron ruteados directamente a pines de entrada/salida del Cyclone II por lo que en caso de querer utilizar dicha interfaz sería necesario configurar estos pines como pines LVDS en la asignación de pines en el software del fabricante (Quartus II) e implementar dentro del FPGA un bloque para convertir datos paralelos a seriales y viceversa, sustituyendo la funcionalidad del chip Channel Link. Al pie de la tabla 7 se especifica cada señal y si la misma tiene que ser implementada como de recepción o de transmisión desde el punto de vista del FPGA.

Cabe recordar que nuestra placa cumpliría la función de Frame Grabber, o sea la de adquirir los datos desde la cámara. Una de las ventajas que posee es que al contar con un FPGA como adquisidor de los datos se le puede sumar la posibilidad de procesar los mismos de manera inmediata dentro del mismo integrado.

Con respecto al conector utilizamos el MDR-26 fabricado por la empresa 3M. Optamos por el mismo ya que fue diseñado específicamente para trabajar con señales del tipo LVDS, cumple con el estándar definido por Camera Link y además es el más utilizado en la industria de cámaras digitales.

En la tabla 7 se especifica el pinout del conector (el mismo cumple con el estándar establecido para Camera Link) y los pines que se utilizaron del FPGA.

Pin FPGA	Pin Conector	Señal
	1	GND
	14	GND
66	25	X0-
65	12	X0+
68	24	X1-
67	11	X1+
72	23	X2-
70	10	X2+
80	22	Xclk-
79	9	Xclk+
87	21	X3-
86	8	X3+
88	20	SerTC+
90	7	SerTC-
97	19	SerTFG-
96	6	SerTFG+
106	18	CC1-
105	5	CC1+
110	17	CC2-
109	4	CC2+
114	16	CC3-
113	3	CC3+
116	15	CC4+
117	2	CC4-
	13	GND
	26	GND

Xi: señales de datos, unidireccionales, hacia el FPGA
Cci: señales de control de cámara, unidireccionales desde el FPGA.
Xclk: señal de reloj, unidireccional, hacia el FPGA.
SerTC :señal serial de comunicación hacia la cámara.
SerTFG: señal serial de comunicación desde la cámara.

Tabla 7: Pinout FPGA y del conector Camera Link

3.3.5 Alimentación:

3.3.5.1 Descripción:

La Placa IIE-Cyclone cuenta con un plano enteramente dedicado a suministrar los distintos niveles de voltaje requeridos por los componentes de la placa.
Para cumplir con dicho objetivo tuvimos que dividir el plano en 5 zonas distintas:

- 5V
- 3.3 V
- 2.5 V
- 1.2 V analógico
- 1.2 V digital

La zona de 5V esta en contacto con uno de los pines del conector de alimentación Mini 2.54 mm, mediante esta zona se alimentan todos los conversores DC-DC de la placa los que a su vez alimentan a los distintos componentes de la misma.
Como se decidió no conectar los pines de alimentación del expansor, siempre es necesario proveer de alimentación a la placa mediante el conector Mini 2.54 mm. Dicha decisión fue tomada para evitar posibles problemas de térmicos en los pines del expansor y posibles problemas de diferencia de potencial.

Mediante la zona de 3.3V se alimenta:

1) SDRAM
2) La alimentación I/O de los bancos 1, 2, 5, 6, 7 y 8 del Cyclone II
3) Max3232
4) EPCS4
5) Oscilador

Mediante la zona de 2.5V se alimenta los pines I/O de los bancos 3 y 4 del Cyclone II, dichos bancos manejan las señales LVDS de la interfaz Camera Link. Este plano fue necesario para cumplir con el estándar LVDS.

Mediante la zona de 1.2V digital se le brinda la alimentación general al Cyclone II.

Mediante la zona de 1.2V analógico se alimento los cuatro PLL internos del Cyclone II.

Fue necesario dividir el plano de 1.2V en dos, uno digital y otro analógico porque el ruido que generan las señales digitales puede llegar a causar errores en la señal de reloj.

Para llegar a los niveles de voltaje necesario a partir de la fuente de 5V utilizamos 3 conversores DC-DC:

1) PT6461 de Texas Instruments (5V a 3.3V)
2) PT6462 de Texas Instruments (5V a 2.5V)
3) PT6466 de Texas Instruments (5V a 1.2V)

Todos tienen una corriente de salida máxima de 14 A, eficiencia mayor al 90 %, pines para el ajuste fino (mediante resistencias) del nivel de voltaje de salida y vienen en encapsulado SIP (Single In-line Package) de 12 pines.
Decidimos utilizar dichos conversores ya que los mismos resuelven casi por completo la alimentación de la placa; sólo fue necesario agregarles un par de capacitores a la entrada y a la salida.
Además, unos módulos similares, pero de 12V de entrada, fueron utilizados en la Placa Virtex [15] con buen desempeño, lo que marcaba un buen antecedente.
Se terminó utilizando la línea PT646x [24], ya que entre los de 5V de entrada, eran los más pequeños de tamaño. Además, tienen su mejor eficiencia (96%) a corrientes de entre 2A y 4A.
La placa cuenta con un LED indicador de alimentación que se enciende cuando está presente la alimentación de 5 VDC.

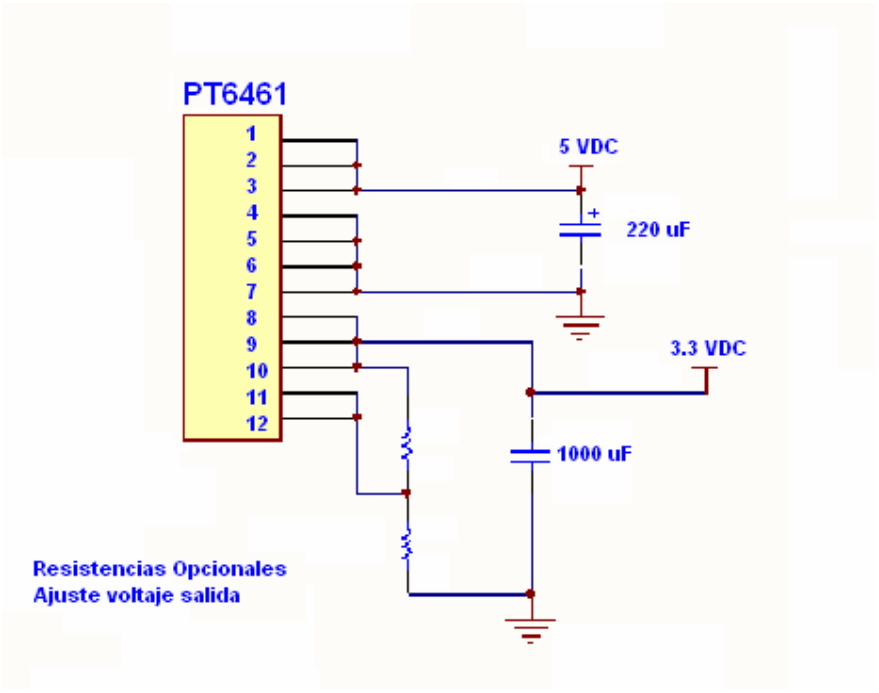


Figura 8: Fuente de 3.3 VDC

3.3.5.2 Consumo de potencia:

Fue necesario realizar un análisis aproximado del consumo de potencia de cada uno de los integrados de la placa para de esta manera poder determinar cual seria la eficiencia de nuestras fuentes de alimentación.
Para obtener un consumo estimado de los distintos bloques del Cyclone II utilizamos el Early Power Estimator [7]. El mismo es una planilla Excel desarrollada por Altera que permite calcular la potencia de consumo del FPGA tomando en cuenta las características del diseño que se quiere implementar en el mismo. En nuestro caso calculamos la potencia tomando como hipótesis la utilización de un 70 % de los recursos del FPGA.

1) Componentes que se alimentan con 3.3 Vdc:

- Oscilador: consumo máximo 0.009 A, utilizamos este valor para la estimación (fuente información: hoja de datos del componente)

- SDRAM: consumo en régimen 0.130 A (fuente información: documentación placa IIE-PCI)

- EPCS4: consumo máximo 0.015 A (fuente información: hoja de datos del componente)

- Max3232: consumo máximo 0.001 A (fuente información: hoja de datos del componente)

- Bancos I/O 1, 2, 5, 6, 7, 8 del Cyclone II: consumo estimado 0.351 A (fuente información: Early Power Estimator).

Consumo total fuente 3.3 Vdc: 0.506 A

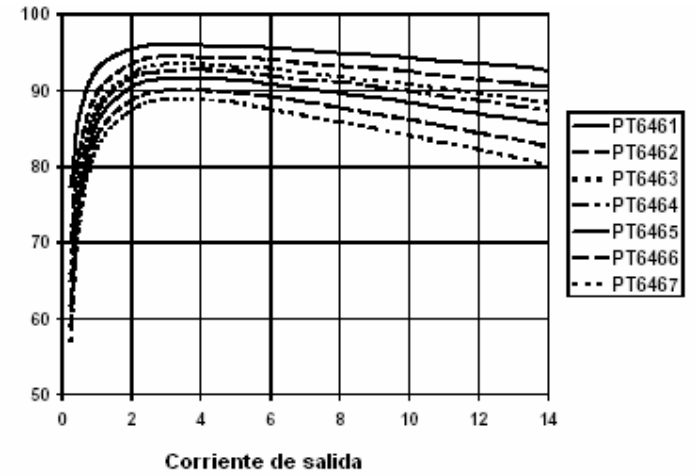


Figura 9: Eficiencia (%) en función de corriente de salida para los PT646x

En la figura 9 se detalla la eficiencia en función de la corriente para los PT646x. De la misma se puede inferir que nuestra fuente de 3.3 Vdc (PT6461) tendrá una eficiencia aproximada del 75 %.

2) Componentes que se alimentan con 1.2 Vdc:

- PLL interno del FPGA: consumo estimado 0.011 A (fuente de información: Early Power Estimator)
- Alimentación general Cyclone II: consumo estimado 0.663 A (fuente información: Early Power Estimator)

Consumo total fuente 1.2 Vdc: 0.674 A

A partir de la figura 9 inferimos que la eficiencia para nuestra fuente de 1.2 Vdc será del 80 %.

3) Componentes que se alimentan con 2.5 Vdc:

- Bancos I/O 3 y 4 del Cyclone II: consumo estimado: 0.101 A (fuente información: Early Power Estimator)

Nuevamente a partir de la figura 9 inferimos que la eficiencia para nuestra fuente de 2.5 Vdc será en el entorno del 60 %.

3.3.6 Switches, LEDs y pulsadores:

Como la placa fue pensada desde un principio como plataforma para pruebas de diseños de aplicaciones sobre su FPGA, nos pareció muy práctico y útil dotar a la misma de llaves, pulsadores y LEDs para facilitar las pruebas de las mismas.

La placa cuenta con un Dip Switch de cuatro llaves conectadas mediante pull-ups a pines I/O del Cyclone II. Cuando las llaves se encuentran en posición ON los pines del Cyclone II quedan a GND y cuando están en posición OFF quedan conectados a 3.3V. Tener la precaución al realizar la asignación de pines de que si no se utiliza el Dip Switch las mismas queden asignadas como entradas tri-estate.

Se agregaron 4 LEDs de propósito general conectados a pines I/O del Cyclone II, los mismos se encienden cuando se asigna un nivel alto de voltaje a la salida correspondiente (un “1” lógico).
Además de estos LEDs se conectó un LED al pin INIT_DONE del Cyclone II. Dicho pin se activa cuando el FPGA entra en user mode. Sirve como testigo para saber si el Cyclone II quedó bien configurado. Esta opción tiene que ser habilitada en el Quartus II (en la sección “Device & Pin Options” de los Settings del Proyecto). De lo contrario, se comporta como un pin I/O estándar.

También se agregaron tres pulsadores, dos de uso múltiple conectados a pines I/O del Cyclone II y el otro conectado al pin DEV_CLR. Este último borra todos los registros internos del FPGA. Nuevamente para que cumpla con esta función se tiene que habilitar mediante el Quartus II, si no se comporta como un pin I/O estándar.

.

Componente	Pin FPGA
Switch 1	119
Switch 2	118
Switch 3	111
Switch 4	100
LED 1	64
LED 2	73
LED 3	78
LED 4	84
LED_INIT	126
Pulsador_1	232
Pulsador_2	233
Pulsador_CLR	238

Tabla 8: Pines del FPGA asignados.

Componente	ON	OFF
Switch 1	0	1
Switch 2	0	1
Switch 3	0	1
Switch 4	0	1

Tabla 9: Valor lógico de las posiciones del DipSwitch

3.3.7 Señal de reloj:

En el momento del diseño de la placa manejamos dos opciones distintas para proveer de señal de reloj a la misma:

- Utilización de un PLL implementado con un integrado
- Utilización de uno de los PLL internos del Cyclone II.

Implementación del PLL con un integrado:

La señal de reloj que llega al FPGA y a la SDRAM es generada por un PLL externo (ej.: IDT5V925, utilizado en la placa IIE-PCI y en la placa Virtex).
El mismo, a partir de la señal de un cristal o de la señal de reloj recibida por el expansor genera por uno de sus pines de salida un pulso de reloj cuyo período es un múltiplo o un submúltiplo de las entradas.

Implementación utilizando uno de los PLL internos del Cyclone II:

El Cyclone II cuenta con 4 PLLs, los mismos están físicamente colocados en cada una de las esquinas del integrado. Son habilitados y configurados mediante el Quartus II. Para su funcionamiento es necesario conectar a los pines de entrada de los mismos una fuente de reloj.

Cada uno de los PLL cuenta con las siguientes características:

- Multiplicación y división de la señal de entrada
- Corrimiento de fase programable
- Ciclo de trabajo programable
- Hasta tres salidas internas de reloj
- Una salida externa
- Opción de salida diferencial
- Hasta cuatro entradas de reloj
- Señales de control internas
- Cambio de la referencia de reloj mediante lógica interna

Teniendo en cuenta las ventajas anteriormente descriptas y para aprovechar al máximo las posibilidades que nos brinda el Cyclone II optamos por utilizar uno de los PLL internos con los que cuenta el FPGA.

Decidimos implementar la solución utilizando el PLL número cuatro del Cyclone II. A estos PLL se les puede conectar como entrada para fuente de sincronismo dos señales distintas. En nuestra placa una de sus entradas es la señal de reloj proveniente de la Placa IIE-PCI; la misma llega a nuestra placa a través de uno de los pines del expansor.

La restante fuente de reloj proviene de un oscilador de 25 MHz con el que cuenta la Placa IIE-Cyclone. Dicho oscilador provee a la placa de una señal de reloj independiente, de forma de no necesitar estar conectados a la placa IIE-PCI para poder trabajar en forma síncrona.
La salida externa con la que cuenta el PLL se conectó al pin de reloj de la SDRAM.

Señal	Pin FPGA	Tipo
Clk_Expansor	95	entrada
Salida Osc.	94	entrada
Clk_RAM	128	salida

Tabla 10: Pinout señales reloj FPGA

Teniendo en cuenta el rango de valores por los cuales se puede multiplicar y dividir la señal a la entrada del PLL interno optamos por un oscilador de 25 MHz, ya que con el mismo tenemos un amplio espectro de frecuencias a la salida del PLL y en un rango válido para el resto de los componentes de la placa.
Utilizamos el oscilador CMX309FBC25.000MTR de Citizen America Corporation por su bajo costo y porque el mismo se encontraba dentro del catálogo de la empresa a la cual se realizó la compra general de componentes para la placa.

Como se mencionó anteriormente, con el software Quartus II se pueden habilitar los PLLs del Cyclone II. Para esto se puede utilizar la mega función altpll. Mediante la misma se selecciona la señal a ser utilizada como referencia (oscilador o señal reloj proveniente del expansor) y entre otros parámetros, los factores de multiplicación y división de la señal de entrada.

3.3.8 Capacitores de desacople:

Cuando cambia el estado de una señal en un pin de salida de un circuito integrado, por ejemplo de un nivel lógico alto a uno bajo, se requiere una corriente de manera inmediata en la carga a la salida para alcanzar el nivel de voltaje requerido.
La función de los capacitores de desacople es la de almacenar carga eléctrica, la que es liberada hacia la línea cada vez que una transición ocurre. De esta manera se minimiza el ruido que generaría en la fuente esta necesidad de corriente inmediata.

Este sistema de almacenamiento de energía debe cubrir un amplio espectro de frecuencias por lo que una serie de capacitores de distinto valor se requiere para cumplir esta función. Es necesario que los mismos se coloquen entre tierra y el plano de alimentación.
Capacitores pequeños con una baja impedancia en serie proveen corrientes rápidas para transiciones de alta frecuencia.
Capacitores grandes continúan brindando corriente después de que los capacitores pequeños se han descargado.

Altera recomienda utilizar tres tipos distintos de capacitores de desacople: de baja frecuencia, de frecuencia media y de alta frecuencia.

Es de vital importancia el posicionamiento para los capacitores de alta frecuencia, ya que se debe minimizar el largo de las pistas entre los terminales de los capacitores y los pines de alimentación del FPGA, para de esta manera disminuir lo más que se pueda la inductancia del camino entre los capacitores y los pines del FPGA.
Es por lo anteriormente mencionado, que se recomienda posicionar los capacitores de desacople de alta frecuencia en la cara contraria a la que se coloca el FPGA. Así se logra minimizar la distancia entre el capacitor y el FPGA.

Los restantes tipos de capacitores, los de frecuencia media y los de baja frecuencia, pueden ser colocados en cualquier lugar de la placa. De todas maneras Altera recomienda que se coloquen los capacitores de frecuencia media a menos de 3 cm del FPGA.

En la figura 10 se detallan las recomendaciones de Altera para el posicionamiento, el tipo y el valor que deben de tener los distintos tipos de capacitores de desacople.
Con respecto a las cantidades Altera recomienda colocar un capacitor de alta frecuencia por cada pin de alimentación (tanto para alimentación VCCINT como para alimentación VCCIO), un capacitor de frecuencia media cada dos bancos de I/O (uno para VCCIO y otro para VCCINT) y un capacitor de baja frecuencia para cada nivel de voltaje con el que se alimente el FPGA.

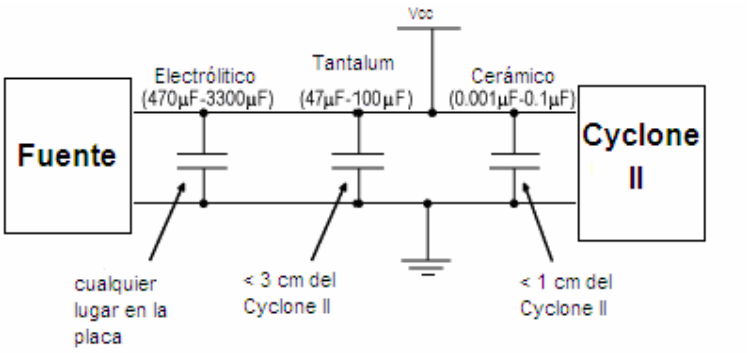


Figura 10: Capacitores desacople para Cyclone II

En nuestro caso decidimos utilizar los siguientes capacitores:

1) Capacitores de frecuencia alta para VCCIO: cerámicos de 2.2 nF

2) Capacitores de frecuencia alta para VCCINT: cerámicos de 33 nF

3) Capacitores de frecuencia intermedia para VCCIO y VCCINT: 47 uF de Tantalum

4) Capacitores de frecuencia baja para VCCIO y VCCINT: Electrolíticos de 1000 uF

Para los restantes circuitos integrados de la placa se colocaron la cantidad y tipos de capacitores de desacople según las recomendaciones del fabricante.

3.4 Elección, listado y costo de materiales

Luego de un diseño inicial de la placa, en el cual ya se tenía diagramados los distintos bloques, nos encontramos frente a la necesidad de definir con qué componentes electrónicos se implementarían dichos bloques.

Con respecto al FPGA, por ser el componente más costoso de la placa, siempre manejamos la opción de solicitar su donación al fabricante.
También teníamos decidido que el mismo fuera de la empresa Altera, ya que se tenia experiencia en el trabajo con FPGAs de dicha compañía, la cuál siempre ha cooperado con el Instituto, incluso donando los FPGA Acex con que cuenta la placa IIE-PCI.

Finalmente nos decidimos por la línea Cyclone II ya que la misma es una de las últimas generaciones de FPGAs de Altera y cuentan con una cantidad importante de elementos lógicos. Además, es una línea de productos diseñados para ser vendidos a bajo costo, lo que facilitaría su donación.
Dentro de los distintos sabores de Cyclone II optamos por el EP2C20Q240C8 ya que el mismo es el más grande de los que tienen sus pines en su perímetro, lo que nos permitiría poder soldarlo a la placa con las herramientas con las que contábamos. Los modelos que cuentan con más celdas lógicas son del tipo BGA (Ball Grid Array) lo que nos exigía enviarlos a soldar a otro país, con los costos y tiempos que esto implica. Por mas detalles al respecto de la soldadura de componentes ver capítulo 2.9.

La donación fue tramitada por Juan Pablo Oliver, solicitando además de los FPGA los dispositivos de configuración. Los Cyclone II utilizan memorias de la línea EPCS, de las cuáles la que tiene la capacidad adecuada para nuestro FPGA es la de 4Mbit (EPCS4).
Como resultado del pedido la empresa Altera nos donó 6 Cyclone II del modelo solicitado y 100 memorias de configuración del tipo EPCS4.

Luego de la llegada de los componentes a Montevideo fue necesario realizar los trámites para poder retirarlos de Aduana sin costo de importación. Para lograr esto contamos con la ayuda de Sebastián Fernández que gestionó los trámites en la Facultad.

El siguiente bloque a resolver era el de la memoria, se decidió utilizar las SDRAM con las que contaba el IIE, las mismas fueron fabricadas por Micron y su modelo es MT48LC4M32B2.
Estas fueron donadas por la empresa Micron y dicha donación fue inicialmente solicitada por el grupo de la Placa IIE-PCI, el cual utilizo 2 memorias, una por placa. Luego el grupo de la Placa Virtex también las utilizó, quedando disponibles para nuestro proyecto las últimas 3.

Con respecto al bloque de alimentación optamos por resolver los distintos conversores DC-DC necesarios para nuestra placa, utilizando la línea de productos de Texas Instruments (T.I.) PT646x. En el capítulo 2.2.5 se detallan sus características y los motivos de su elección.

Se solicitó su donación a la empresa Texas Instruments vía online. Como la misma ya no envía componentes de muestra a Uruguay debido a pedidos abusivos de varios compatriotas tuvimos que solicitar que enviara las muestras a una dirección en Estados Unidos. Afortunadamente contamos con la colaboración del tío de Andrés, Alberto Bergeret, quien no solo permitió que solicitáramos la muestra poniendo como destinatario la dirección de su casa en Estados Unidos, sino que también se encargó de traernos las muestras en Diciembre de 2005.
Resumiendo, T.I. nos donó 2 PT6461, 2 PT6462 y 2 PT6466 con los que pudimos resolver todos los niveles de voltaje requeridos en el diseño de nuestra placa.

Para poder implementar la interfaz serial RS-232 utilizamos el integrado MAX3232 de Texas Instruments. El mismo fue donado por la empresa Texas y la donación fue solicitada por los integrantes del proyecto de la placa Virtex.

El resto de los componentes de la placa fueron comprados en DigiKey vía online; el costo de los mismos fue financiado por los miembros del grupo.
En la tabla 11 se detallan los materiales utilizados para fabricar una placa y el costo de los mismos. Para el proyecto se fabricaron dos placas idénticas.

COSTO TOTAL DE COMPONENTES Y FABRICACIÓN DE UNA PLACA: 190.39 US$

Componente	Marca	Modelo	VALOR	Cant. X Placa	Costo Unitario (US$)
Placa	N/A	N/A	N/A	1	145
Pulsador	Panasonic	EVQ-PHP03T	N/A	3	0.86
Capacitor	Panasonic	ECJ-4YF0J476Z	47uF	8	1.609
Capacitor	Panasonic	ECJ-1VB1H333K	33nF	20	0.04
Capacitor	Panasonic	ECJ-1VB1H222K	2.2nF	17	0.02
Capacitor	Panasonic	ECJ-1VB1H102K	1nF	4	0.02
Capacitor	Rohm	TCP0J106M8R	10uF	1	0.31
Capacitor	Panasonic	ECJ-1VB1C104K	0.1uF	15	0.02
Capacitor	Panasonic	EEV-FK0J221P	220uF	3	0.28
Capacitor	Panasonic	EEV-FK0J102P	1000uF	3	1.92
Oscilador	Citizen Amer. Corp	CMX309FBC25.MTR	N/A	1	2.81
Con. Alimentación	N/A	N/A	N/A	1	1.82 (1)
Con. Camera Link	3M	MDR-26	N/A	1	4.04
Cyclone II	Altera	EP2C20Q240C8	N/A	1	Donado por Altera
Mem. Configuración	Altera	EPCS4	N/A	1	Donado por Altera
Expansores	N/A	N/A	Tira pines M 20X2 rectos	2	1.5
Header	N/A	N/A	Tira pines M 1X3 rectos	2	(3)
Header	N/A	N/A	Tira pines M 2X4 rectos	1	(3)
Header	N/A	N/A	Tira pines M 2X5 acodados	3	3.76 (2)
Header	N/A	N/A	Tira pines M 2X5 rectos	1	(3)
LED	Lite-On inc	LTST-C150EKT	N/A	6	0.088
RS-232 Line driver	Texas	MAX3232	N/A	1	Donado por Texas
SDRAM	Micron	MT48LC4M32B2	N/A	1	Donado por Micron
Conv. DC-DC	Texas	PT6461	N/A	1	Donado por Texas
Conv. DC-DC	Texas	PT6462	N/A	1	Donado por Texas
Conv. DC-DC	Texas	PT6466	N/A	1	Donado por Texas
Resistencia	Panasonic	ERJ-6GEYJ680V	68ohm	4	0.077
Resistencia	Panasonic	ERJ-6ENF1001V	1Kohm	7	0.048
Resistencia	Panasonic	ERJ-6ENF1690V	169ohm	1	0.091
Resistencia	Panasonic	ERJ-6GEYJ121V	120ohm	1	0.077
Resistencia	Panasonic	ERJ-6ENF1002V	10Kohm	4	0.048
Resistencia	Panasonic	ERJ-6ENF1000V	100ohm	6	0.048
Switch	ITT Industries	SDA04H1SKD	4 llaves	1	1.74

(1) Costo de una tira de 36 posiciones
(2) Costo de tira de 72 posiciones
(3) Para todas se utilizo una tira de 72 posiciones, costo 4.02 US$

Tabla 11: Listado de componentes Placa IIE-Cyclone

3.5 Diseño de la placa:

3.5.1 Introducción:

Luego de haber elegido el tipo de Placa, sus principales bloques y los componentes con los que pensamos implementar los mismos, llegamos al momento de diseñar su interconexión y el formato final de la Placa.
La primera decisión que tuvimos que tomar fue la elección de la herramienta de diseño a utilizar. Buscamos un software que sirviera para todas las etapas del diseño, es decir que nos sirviera para diseñar los esquemáticos de la placa y que a partir de los mismos se pudiera diseñar el Layout de la placa impresa.
Decidimos utilizar el Protel DXP de Altium [39,42,43] (ofrecen una versión Demo por 30 días). El mismo presenta una interfaz de usuario muy sencilla y a su vez muy potente, cuenta con bibliotecas de componentes de varias marcas y como mencionamos previamente cumple con el requisito de servirnos para todas las etapas del diseño. Además, tuvimos muy buenas referencias de versiones anteriores del mismo, las que fueron utilizadas en proyectos previos dentro del IIE.

El primer paso dentro del diseño es el de la realización de los esquemáticos de la placa. Los mismos fueron realizados de acuerdo a las distintas funcionalidades de la placa y especifican de forma gráfica los distintos componentes y sus interconexiones.
Para nuestro diseño realizamos ocho esquemáticos distintos: dos presentan los distintos pines I/O del FPGA y sus conexiones, uno especifica la alimentación del FPGA y el conexionado de sus PLL, otro el conexionado de los pines de configuración del FPGA, el quinto muestra los conversores DC/DC y los capacitores de desacople, el sexto los conexionados de la RAM, el séptimo las distintas interfaces de la placa y el ultimo es un diagrama general que interconecta los distintos bloques mencionados.

El siguiente paso fue asignarle un footprint a cada uno de los distintos componentes de la placa. Éstos contienen toda la información del componente para la fabricación de la placa, incluyendo pads (lugar donde serán soldados) si son de montaje superficial o thru-hole, el área que se debe reservar alrededor del mismo para su correcta colocación, su silkscreen, etc.
Los mismos fueron generados por nosotros o tomados de las distintas bibliotecas con las que cuenta el Protel DXP. En la tabla 12 se especifican los distintos componentes y su footprint asociado.

Previo al posicionamiento de los componentes y al ruteo de las pistas fue necesario definir tamaño, forma de la placa y cantidad de capas con los que iba a contar.
Para optimizar superficie de impreso decidimos que la misma fuera rectangular. Con respecto al tamaño, partimos de un tamaño inicial de 20 cm. x 12 cm. para luego de varias optimizaciones terminar con un tamaño final de 15.2 cm. x 9.6 cm. Para llegar a dicho tamaño final se tomó en cuenta no sólo

que entraran todos los componentes sino que también se tuviera espacio para un ruteo acorde de las pistas.

Componente	VALOR	FOOTPRINT
Placa	N/A	N/A
Pulsador	N/A	push botton (1)
Capacitor	47uF	1210
Capacitor	33nF	0603
Capacitor	2.2nF	0603
Capacitor	1nF	0603
Capacitor	10uF	0805
Capacitor	0.1uF	0603
Capacitor	220uF	Cap220cd (1)
Capacitor	1000uF	Cap1000cf (1)
Oscilador	N/A	Oscillator (1)
Con. Alimentación	N/A	HDR1X4H
Con. Camera Link	N/A	MDR-26
Cyclone II	N/A	PQFP240
Mem. Configuración	N/A	SOIC8
Expansores	Tira pines M 20X2 rectos	HDR2X20
Header	Tira pines M 1X3 rectos	HDR1X3
Header	Tira pines M 2X4 rectos	HDR2X4
Header	Tira pines M 2X5 acodados	HDR2X5H
Header	Tira pines M 2X5 rectos	HDR2X5H
LED	N/A	1206
RS-232 Line driver	N/A	SSO-G16/X.4
SDRAM	N/A	TSOP86
Conv. DC-DC	N/A	PT6460 (1)
Conv. DC-DC	N/A	PT6460 (1)
Conv. DC-DC	N/A	PT6460 (1)
Resistencia	68ohm	0805
Resistencia	1Kohm	0805
Resistencia	169ohm	0805
Resistencia	120ohm	0805
Resistencia	10Kohm	0805
Resistencia	100ohm	0805
Switch	4 llaves	BPA08

(1) footprints realizados por nosotros

Tabla 12: Componentes y footprint asociados

Con respecto a los capas, tomando en cuenta todas las pistas que teníamos que interconectar entre los distintos componentes llegamos a la conclusión que con una sola capa de señal no nos alcanzaría para rutearlas.

Terminamos definiendo una placa con cuatro capas, las dos exteriores tienen todas las pistas de señal, una de las interiores posee los distintos planos de voltaje con los que trabaja la placa y el restante es de tierra.
Para definir la ubicación ya mencionada de cada una de las capas nos basamos en placas anteriormente diseñadas en el IIE y en bibliografía especializada en el tema.

3.5.2 Posicionamiento de los componentes:

Como queríamos que nuestra placa fuera compatible con la Placa IIE-PCI, lo primero que tuvimos que tomar en cuenta para el posicionamiento de los componentes fue la distancia entre los buses de expansión; dicha distancia estaría determinada por la distancia que hay entre los buses de expansión de la Placa IIE-PCI.
Luego, a partir de dicha distancia, posicionamos los buses en nuestra placa de manera tal que al estar conectadas ambas placas y la IIE-PCI conectada al bus PCI, el borde externo derecho de nuestra placa quedara accesible desde el exterior de un PC con gabinete estándar.

En dicho borde derecho posicionamos todos los conectores externos (Camera Link, Serial RS232, JTAG y Active Serial) para que se pudiera tener acceso a los mismos cuando se estuviera trabajando con ambas placas conectadas entre sí y al bus PCI.

Al principio pensábamos implementar nuestros buses con tiras de pines hembras y solucionar la interconexión con la Placa IIE-PCI, que también tiene pines hembra en sus expansores, utilizando tira de pines doble macho.
Finalmente para lograr una mejor conexión física con la Placa IIE-PCI decidimos utilizar pines macho como expansores de nuestra placa; esto nos terminó determinando una distancia mucho menor que la pensada inicialmente entre las placas cuando están conectadas entre sí.
Debido a lo anterior tuvimos que tener en cuenta para el posicionado de los componentes en la placa la altura de cada uno de los componentes de la cara superior de la placa IIE-PCI. Por este motivo colocamos los conversores DC-DC en la cara inferior de nuestra placa y colocamos los LEDs en el borde superior derecho de la placa ya que coincide con los componentes más altos de la Placa IIE-PCI (jumpers y zócalo de la memoria de configuración).

De todos modos ambas placas en conjunto no cumplen con el máximo tamaño permitido para una placa que se conecte a un slot PCI. Por lo tanto es necesario dejar libre el slot siguiente al que vaya a ser conectada la placa IIE-PCI y acceder a las interfaces de nuestra placa mediante la ranura de dicho slot. Como nuestro diseño es un prototipo preferimos darle mayor importancia a lograr las características que queríamos que nuestra placa tuviera más que a cumplir con una norma de distancia entre slots.

El primer componente que posicionamos en la placa fue el FPGA; como el mismo tiene pistas que lo unen a cada uno de los restantes componentes de

la placa, decidimos colocarlo en el centro de la misma para evitar que alguna pista quedara muy larga o que no fuera viable su ruteo.
Fue orientado para lograr que los bancos con mayor cantidad de pines I/O que manejan señales del tipo LVDS tuvieran un camino lo más corto posible al conector Camera Link, el cual fue ubicado en el borde derecho de la placa para cumplir con las características de acceso previamente mencionadas.

El posicionamiento de la RAM fue el más discutido y el que generó mayor dificultad. Ya que teníamos que rutear las pistas de datos de la RAM no sólo hacia el FPGA sino que también hacia el bus de expansión (ver capítulo 3.3.3 por una explicación más detallada de esta configuración).
Lo primero que surgió fue posicionar la RAM entre el FPGA y el bus de expansión para acortar distancias en las pistas de datos.
Inicialmente colocamos la RAM en el lado superior de la placa, pero luego de varios intentos de ruteo de las pistas de datos, nos dimos cuenta que con sólo dos capas de señal las rutas entre el FPGA y la RAM quedaban o muy largas o tenían varias vías (cambios de capa) para una misma señal.
Como no era viable desde el punto de vista económico agregarle más capas a nuestro diseño decidimos intentar el ruteo posicionando la RAM en la otra cara de la placa.
Luego de varios días de ruteo, en los cuales cambiamos varias veces la posición de la RAM, subiéndola, bajándola, girándola y en los cuales cambiamos un sin número de veces los pines del FPGA asignados a conectarse a la RAM, logramos encontrar una posición para la RAM en la cual cumpliéramos con un ruteo acorde a las características buscadas para el mismo (ver capítulo 3.5.3 por más detalles de ruteo).

Con respecto a los conversores DC-DC decidimos colocarlos en el perímetro de la placa para evitar interferencias con las pistas de señal.
El conector de alimentación fue colocado en el borde izquierdo de la placa para que cuando la misma sea conectada a la Placa IIE-PCI el conector quede mirando hacia el interior del PC y de esta manera facilitar su conexión.
También decidimos colocar los conversores DC-DC lo más cerca posible del conector de alimentación, de esta manera se logra que el plano de 5 VDC sea lo más chico posible ya que el mismo solo se utiliza para tomar la alimentación general a partir del conector y de ahí alimentar a los tres conversores.

Para posicionar el oscilador tuvimos en cuenta que el largo de la pista de señal de reloj hasta el pin de entrada del PLL interno del FPGA sea lo más corto posible.
Asimismo se colocó la memoria de configuración en la esquina derecha superior del FPGA para lograr que las pistas que lo unen hacia el FPGA y las que lo unen al conector Header Active Serial fueran lo más cortas posible.

El MAX3232 se colocó lo más cerca posible del conector Serial y en una posición tal que se pudiera pasar las pistas hacia el FPGA por la parte inferior del expansor I/O.

El resto de los componentes: pulsadores, dip switch, LEDs y jumpers fueron colocados en los espacios existentes entre los componentes principales inicialmente colocados, tratando de posicionarlos de manera tal que el ruteo de las pistas hacia el FPGA fuera lo más directa posible.

Los capacitores de desacople se colocaron lo más cerca posible de los pines de alimentación de los componentes. Se realizó de esta manera para cumplir con las recomendaciones de los fabricantes de los integrados.
Para lograr dicho objetivo fue necesario que quedaran posicionados del lado opuesto al que se colocó el componte; esto fue necesario para no interferir con las pistas de señal que salen de cada componente.

Cabe mencionar que el software de diseño Protel DXP cuenta con una función de posicionamiento automático de los componentes. Hicimos una prueba de la misma, primero posicionando los buses de expansión y luego dejamos que el programa hiciera su trabajo. El programa para posicionar los componentes tiene en cuenta no solo su área sino que también el conexionado con el resto de los componentes.

Luego de ver el diseño final del posicionamiento hecho por el programa desistimos de esta metodología de trabajo porque en nuestro diseño también teníamos que tener en cuenta otros factores extra: la altura de los componentes, el fácil acceso de los conectores desde el exterior y el mantener los conversores lejos de las pistas de señal, puntos que el programa no tenía en cuenta para el posicionamiento.

3.5.3 Ruteo de pistas:

Para el ruteo de las pistas se tomaron en cuenta los siguientes puntos[8]:

- Evitar cambios bruscos de dirección (90 grados) y los ángulos agudos en las pistas de señal (genera cambios de impedancia y por lo tanto señales reflejadas).
- Si es necesario conectar pistas a 90 grados suavizar los ángulos utilizando 2 giros de 45 grados.
- Ancho de pistas de GND lo más grande posible, ya que por las mismas pasa mucha corriente y si las pistas son angostas se generan altas resistencias, generando altas caídas de tensión.
- Trazado lo más corto posible para las señales de reloj y rodearlas de pistas de tierra.
- Realizar las pista de señal lo más anchas posible (evita diafonía).
- Distancia entre pistas paralelas lo más grande posible (evita diafonía capacitiva).
- Largo de las pistas lo más corto posible para evitar transformarlas en líneas de transmisión.

Líneas de transmisión:

Cuando trabajamos con señales de baja frecuencia podemos suponer que las mismas se comportan cumpliendo las leyes de la teoría de circuitos (Thevenin, Kirchhoff, etc.); en realidad para obtener una descripción exacta de las mismas tendríamos que remitirnos a las leyes del electromagnetismo, pero a bajas frecuencias el modelo de la teoría de circuitos se acerca mucho al comportamiento real.

A medida que la frecuencia de las señales del circuito va aumentando, el comportamiento descrito por la teoría de circuitos se va alejando del comportamiento real; una de las características de dicho alejamiento es que las pistas de un PCB se empiezan a comportar como líneas de transmisión.
A partir de las ecuaciones de Maxwell se pueden describir las líneas de transmisión como circuitos de constantes distribuidas, es decir como una serie de cuadripolos infinitesimales compuestos por inductancias, capacidades, resistencias y conductancias.

En las líneas de transmisión los valores de voltaje y corriente dependen tanto del tiempo como de la posición en la línea, el parámetro que utilizamos para describir a las mismas es la impedancia característica que se define a partir de la geometría y del material de la misma; es un parámetro independiente del largo de la línea.

Cuando utilizamos el modelo de línea de transmisión la señal es recibida sin distorsión sólo si la impedancia del transmisor, el receptor y la impedancia característica de la línea coinciden (líneas adaptadas).
Cuando trabajamos con líneas adaptadas el único efecto que tenemos es un retardo en la transmisión de la señal, si las mismas no se encuentran adaptadas se genera una onda reflejada en la recepción que luego se vuelve a reflejar en el transmisor y así sucesivamente generándose una onda ringing, la amplitud de la misma depende enteramente del grado de desapareo de las impedancias y su período depende del tiempo de transición entre el transmisor y el receptor; por lo tanto termina dependiendo del largo de la línea.

Ringing en circuitos digitales es siempre un efecto no deseado ya que genera cambios espurios en la señal pero puede llegar a ser tolerado si las amplitudes involucradas están dentro la banda de inmunidad de los circuitos integrados.

Cuando trabajamos con circuitos digitales se utiliza la siguiente regla para determinar si una pista debe de ser tomada como línea de transmisión o si no:

$$T_{rise} < 3 \; X \; T_{travelling}$$

Siendo:

Trise: menor tiempo de subida de la señal presente en la línea (depende de la carga)

Ttravelling: tiempo que tarda la señal en viajar a lo largo de la línea (función de la constante dieléctrica del material y del largo de la pista).

Si estamos en presencia de una línea de transmisión se puede reducir el efecto ringing terminando las pistas con una resistencia de valor aproximado a 10 Kohm que se colocara a V+ o a tierra (terminación shunt). Otra opción es la de colocar una resistencia en serie en la fuente de la señal, el valor de dicha resistencia sumado al valor de la impedancia de salida debe de ser igual a la impedancia característica de la línea; de esta manera si hay señal reflejada en la carga, la misma es absorbida en la fuente.

Como mencionamos previamente en el listado de puntos, siempre intentamos al rutear las pistas, que las mismas fueran del menor largo posible.
Para las pistas más largas de la placa realizamos el cálculo para ver si teníamos que tomarlas como líneas de transmisión o no.

Luego de realizar los cálculos concluimos que ninguna de las pistas de la placa debían de ser tomadas como línea de transmisión y por lo tanto no fue necesario utilizar resistencias de adaptación en ningún caso.

El Protel DXP también cuenta con una herramienta para el autoruteo de las pistas. Luego de la etapa de posicionamiento hicimos varias pruebas con esta herramienta.
En el primer PC que la probamos, luego de un tiempo de haberse iniciado la función de autoruteo, la misma aumentaba el consumo de CPU hasta llegar al punto de que la máquina se reiniciaba.
Luego intentamos con un PC con mejor procesador y después de varias horas de autoruteo decidimos detener dicha función ya que notábamos que no generaba los mejores caminos para las pistas.

El ruteo de la totalidad de las pistas fue realizado por los miembros del grupo teniendo en cuenta todos los puntos ya mencionados y nos consumió un tiempo aproximado de un mes.

3.6 Fabricación de la placa:

3.6.1 Introducción

A medida que realizábamos el diseño del Layout de la placa mediante el software Protel DXP fuimos buscando distintas opciones de lugares para enviar a fabricar la placa.

Nos pareció útil este enfoque ya que al realizar las primeras búsquedas de fabricantes nos dimos cuenta que no todos podían cumplir con las mismas características de fabricación, por ejemplo no todos tenían el mismo valor para el ancho mínimo de pista, del mismo modo no todos los fabricantes tenían los mismos tamaños para las vías, ni todos podían realizar placas con cuatro capas. La suma de estos factores terminaban incidiendo en el diseño que estábamos realizando.

Luego de buscar durante varios días en Internet llegamos a la conclusión de que en la región no había ningún fabricante que pudiera realizar impresos con cuatro capas. Esto restringía nuestra búsqueda a empresas en Norteamérica, Europa o Asia.
Por temas de logística vinculados principalmente al costo de envío y a la facilidad de poder ahorrar el mismo mediante la ayuda de alguien que viajara al exterior y nos pudiera traer la placa, nos decidimos a buscar empresas que fabricaran nuestra placa dentro de Estados Unidos.

Finalmente llegamos a dos posibles opciones:

- PCBExpress (www.pcbexpress.com)
- Imagineering Inc. (www.pcbnet.com)

Ambas ofrecían placas de cuatro capas con similares características técnicas, la principal diferencia entre ambas era de precio.
Mientras que Imagineering ofrecía 2 placas por 100 Dólares, PCBExpress ofrecía lo mismo por 290 Dólares. Otro punto a tener en cuenta era que en el IIE ya se tenía muy buenas experiencias al trabajar con PCBExpress.
Teniendo en cuenta el precio, el primer contacto lo establecimos con Imagineerig; luego de realizarle varias consultas vía mail y de no obtener ninguna respuesta de su parte, optamos por enviar a fabricar la placa a PCBExpress.

3.6.2 Características de la fabricación:

La Placa IIE-Cyclone cuenta con:

- 805 pads
- 2665 segmentos de pista
- 321 vías
- 127 componentes

Las características de su fabricación son:

- 4 capas
- ancho mínimo de pista: 6 mils
- ancho mínimo entre pistas: 6 mils
- tamaño mínimo de vía: 8 mils
- mascara antisoldante entre pads
- silkscreen solo del lado superior de la placa (nomenclatura y contorno de los componentes en la placa)
- dieléctrico utilizado FR-4
- espesor del dieléctrico: entre capas 1 y 2: 10-14 mils
  entre capas 2 y 3: 26-28 mils
  entre capas 3 y 4: 10-14 mils
- espesor del cobre : capas 1 y 4 : 07 mils
  capas 2 y 3: 1.4 mils

3.6.3 Información requerida por el fabricante para realizar la placa:

La empresa elegida para fabricar nuestra placa no realiza ningún tipo de chequeo de diseño con respecto a la información que se le envía, por lo tanto es necesario enviar los archivos de diseño de la placa respetando las características que el fabricante detalla en su pagina web.

Primero se estipula que es necesario utilizar el visor de archivos gerber View Mate de la empresa Pentalogic. Además es necesario que las característicos de la placa (tamaño, drill size, etc) se adecuen de manera exacta a las opciones brindadas por el fabricante para el tipo de placa que se contrata el servicio. Sino puede suceder que si diseñamos nuestra placa con un tamaño mínimo de drill distinto al estipulado, luego la misma se fabrica con el tamaño de drill más cercano, lo que generaría problemas a la hora de montar los componentes en la placa.
Tampoco toman en cuenta ningún tipo de archivo de texto en el que se detalle alguna característica de la placa.

Resumiendo toda la información necesaria para fabricar la placa tiene que estar contenida en los siguientes archivos:

Silkscreen: determina las leyendas (texto en la parte superior de la placa) de la placa.

Top Layer: especifica la disposición de las pistas en la cara superior.

Bottom Layer: especifica la disposición de las pistas en la cara inferior.

Top Soldermask: especifica la máscara antisoldadura para los pads de la cara superior.

Bottom Soldermask: especifica la máscara antisoldadura para los pads de la cara inferior.

Nc Drill: coordenadas y tamaño de vias para la placa.

Top Internal Plane: especifica las características de la cara interna superior.

Botton Internal Plane: especifica las características de la cara interna inferior.

3.7 Montaje de componentes:

Inmediatamente después de recibir las dos placas le realizamos una serie de pruebas previas para ver si coincidían con nuestro diseño.

Primero corroboramos tamaño de la placa y distancias entre los principales componentes; en nuestro caso el punto más sensible era la distancia entre los buses de expansión ya que si la misma no se correspondía con la distancia que especificamos en nuestro diseño, podríamos llegar a tener problemas de interconexión con la Placa IIE-PCI.

Luego verificamos que no existiera cortocircuito entre ninguno de los distintos planos de voltaje y que además existiera continuidad entre los distintos pads de alimentación que correspondían al mismo nivel de voltaje.
Por ultimo presentamos los componentes principales sobre la placa para verificar que los pines de los mismos coincidieran de manera exacta con los pads de la placa.

Todas las pruebas anteriormente mencionadas dieron como resultado que la placa enviada por PCBExpress se ajustaba de manera correcta al diseño que les enviamos.

Ya terminada la etapa de testeo inicial nos vimos frente a la necesidad de definir las herramientas que utilizaríamos para el montaje de los componentes en la placa.

De las primeras reuniones con Juan Pablo Oliver ya había quedado claro que necesitaríamos las siguientes herramientas para montar los componentes en la placa:

- Soldador de punta fina
- Algún tipo de aumento bifocal (fundamental para tener noción de profundidad)
- Estaño, lo más fino posible
- Flux
- Cinta desoldadora (malla de cobre para absorber estaño)

El soldador nos los prestó el IIE; lo único que tuvimos que conseguir fue un par de puntas finas, ya que las mismas se deterioran con mucha facilidad.

Con respecto a la lupa de aumento, logramos conseguir un microscopio de pie del tipo que se utiliza para trabajos en Laboratorios de Biología.
El mismo tiene un aumento fijo de x2 y mediante un juego de lentes se le puede aumentar hasta un total de x 30. Luego de probar con los distintos tipos de aumento concluimos que la mejor opción para la soldadura de los componentes con muchos pines era utilizarlo con un aumento total de por 10.

El estaño más fino que pudimos conseguir fue de 0.55 mm y lo compramos en Eneka.

El Flux fue de gran ayuda, ya que el mismo permite que el estaño corra con mayor facilidad entre los pads y los pines de los componentes. Lo conseguimos también en Eneka y la mínima cantidad que vendían era medio litro, por lo que luego de soldar las dos placas nos sobró aproximadamente 490 ml.

Con las placas ya en nuestras manos y con las herramientas ya definidas, solo nos faltaba, para comenzar a soldar, decidirnos por una estrategia de montaje de componentes en la placa.

Nos reunimos con Sebastián Fernández quien ya tenía experiencia en el tema, y nos planteó distintas formas de encarar la soldadura.
Hay dos enfoques distintos: uno que recomienda comenzar soldando por la cara principal de la placa y por los componentes más pequeños y difíciles de soldar (FPGA, Memoria); de esta manera se puede trabajar tranquilos sobre los mismos sin ningún otro componente cercano que nos dificulte el acceso con el soldador.
El otro método recomienda comenzar soldando la parte de alimentación, para luego poder probar con la ayuda de un tester si los planos de voltaje tienen los niveles de tensión adecuados.

Tomando en cuenta las recomendaciones de Sebastián y que ninguno de los miembros del grupo tenía experiencia previa en soldadura de componentes de montaje superficial, decidimos comenzar soldando todos los capacitores de desacople de la placa, que se encuentran en su mayoría en la cara inferior de la placa. De esta manera adquirimos un poco de experiencia para luego comenzar con la soldadura de los componentes de montaje superficial con más pines.

Estos fueron los más difíciles de soldar, principalmente el FPGA y la SDRAM, no sólo por la gran cantidad de pines con los que cuentan sino porque al estar tan cerca, más de una vez al querer soldar un pin se termina haciendo una pequeña pelota de estaño, la que deja 2 o 3 pines del componente cortocircuitados.
En estos casos fue necesario acudir al uso de la cinta para quitar el estaño; la misma se coloca sobre los excesos de éste, se calienta con el soldador, y la malla absorbe el estaño eliminando el cortocircuito entre pines.
Luego de quitar la mayor parte usando la cinta es necesario retirar las gotas pequeñas de estaño de la zona con la ayuda del flux y el soldador, la idea es acercar el soldador lo más que se pueda al pin y tratar de que el estaño quede adherido al soldador.

El Flux también terminó siendo muy efectivo para ayudar a soldar los pines de los componentes. En el caso de los componentes con muchos pines optamos por humedecer con Flux los pads y los pines del componente, de esta manera se facilita mucho la soldadura del pin ya que se evita que al acercar el soldador se deposite demasiado estaño sobre el mismo.

Al finalizar la soldadura de todos los componentes de montaje superficial, seguimos el montaje de la placa soldando los PT646x y por último soldamos el resto de los componentes Through Hole.

La totalidad de los componentes de ambas placas fueron soldados por los miembros del grupo y nos tomó 4 semanas de trabajo.

Como conclusiones de esta etapa de montaje se puede decir que es totalmente viable con herramientas poco sofisticadas como las que manejamos poder soldar una placa con más de 100 componentes de distinto tipo y tamaño en ambas caras de la misma.
Es una actividad que conviene realizar en jornadas lo más largas posibles ya que la preparación de la mesa de trabajo lleva mucho tiempo, además es muy importante trabajar siempre con luz natural y realizar la tarea de soldar en postas, ya que más de una hora soldando puede resultar muy agotador.

Con respecto al orden a seguir para soldar los componentes, es altamente recomendable para gente que no tiene experiencia en soldar componentes de montaje superficial seguir los pasos que realizamos nosotros. Ya que trabajar primero con los capacitores de desacople no sólo nos dio más habilidad para poder afrontar la soldadura de los componentes grandes, sino que también nos dio confianza para soldar los mismos.

3.8 Pruebas a la placa:

La primer prueba que realizamos sobre la placa con los componentes ya montados fue el de alimentarla y verificar el correcto funcionamiento de los tres conversores de voltaje.

Luego realizamos un diseño sencillo que manipula los LEDs a partir de la posición de las llaves del DipSwitch. Mediante este diseño probamos el FPGA, la memoria de configuración EPCS4 y dos métodos de configuración, JTAG y Active Serial.

El segundo diseño que le cargamos al FPGA fue un contador binario que utiliza los LEDs de la placa y toma la señal de reloj del oscilador.
Mediante este diseño probamos el FPGA, el oscilador de la placa y el PLL interno del Cyclone II.

Por último, previo a cargarle al FPGA el algoritmo de compresión de imágenes sin pérdida, le cargamos al FPGA un diseño que implementa salida de datos utilizando el puerto serie de la placa. Mediante el mismo se prueba el FPGA, el PLL y el MAX3232.

3.9 Placas utilizadas como referencia:

En este capítulo detallamos las características de una serie de placas que utilizamos como referencia a la hora de definir el diseño de nuestra placa.

1) IIE-PCI:

- FPGA ACEX EP1K100PQ208
- Memoria SDRAM 128 Mbit
- Compatible con bus PCI de 32 bits
- Expansores con señales provenientes del FPGA
- Conversor DC-DC para generar fuentes de 3.3v y 2.5v a partir de 5v
- PLL
- Llaves y LEDs de propósito general

Por más información: www.mondueri.com/iiepci

2) Sendero:

- FPGA Cyclone II de Altera
- Memoria SDRAM 256 Mbit
- Compatible con PCI-Express
- Interfaz USB 2.0
- Interfaces de video digital
- Memoria Flash de 64 Mbit

Por más información: www.alearep.com

3) DE2:

- FPGA Cyclone II de Altera
- Memoria SRAM de 512 Kbytes
- Memoria SDRAM de 8 MBytes
- Interfaz USB
- Interfaz serial RS232
- Llaves, pulsadores y LEDs de propósito general
- Memoria de configuración EPCS16 de 16 Mbit
- Memoria Flash de 4-Mbyte
- 2 expansores de 40 pines

Por más información: www.terasic.com/english/discuss

4) DBC2C20:

- FPGA Cyclone II de Altera
- Interfaces seriales RS232 y RS485
- Interfaz USB
- LEDs y pulsadores de propósito general
- Interfaces LVDS
- Memoria de configuración EPCS16 de 16 Mbit
- Memoria SDRAM de 16Mbyte
- Memoria SRAM de 1 Mbyte
- Memoria Flash de 8 Mbyte
- Display de 7 segmentos

Por más información: www.devboards.de

5) Nios II:

- FPGA Cyclone II de Altera
- Memoria Flash de 8 Mbytes
- Memoria RAM de 1 Mbyte
- Memoria SDRAM de 16 Mbytes
- Memoria de configuración EPCS4 de 4 Mbit
- Dos Interfaces seriales RS232
- LEDs, llaves y pulsadores de propósito general
- Display de 7 segmentos
- Expansores con 41 acceso a 41 pines I/O del FPGA

Por más información: www.altera.com

6) Cyclone Video Demostration Borrad

- FPGA Cyclone de Altera
- 6 entradas distintas de reloj para el PLL del Cyclone
- LEDs, pulsadores y llaves de propósito general
- 2 entradas ASI y 1 SDI
- 1 salida ASI y una salida SDI
- Memoria de configuración EPCS4

Por más información: www.altera.com

4 CONFIGURACIÓN

4.1 Introducción:

Existen varias formas de configurar el FPGA [2], entre ellas JTAG(Joint Test Action Group) cumpliendo con el standard IEEE1149.1. El Cyclone II usa celdas SRAM para guardar la configuración y como estas son volátiles el Cyclone II debe ser configurado cada vez que se alimenta. La placa cuenta con una memoria EPCS4 con el fin de no tener que configurar el Cyclone II en cada encendido.

En este capítulo se detallan los distintos modos de configuración implementados.

La Placa IIE-Cyclone puede ser configurada de cinco maneras distintas:

1) Active Serial (AS), 20 MHz
2) Fast Active Serial, 40 MHz
3) JTAG placa aislada
4) Cadena JTAG con Placa IIE-PCI
5) Passive Serial (PS)

Para poder determinar qué modo de configuración va a ser utilizado es necesario ajustar los jumpers MSEL1 y MSEL2, los mismos se encuentran claramente identificados entre el Dip Switch y el bus expansor de la RAM.
En la tabla 13 se especifica para cada tipo de configuración el posicionamiento correcto de los jumpers.

Esquema de Configuración	JUMP_MSEL2	JUMP_MSEL1
AS (20 MHz)	0	0
PS	0	1
Fast AS (40 MHz)	1	0
JTAG placa aislada(1)	0(2)	0(2)
Cadena JTAG(3)	0(2)	0(2)

Tabla 13: Selección de jumpers de configuración

(1) Configuración JTAG toma precedencia sobre el resto de las configuraciones.
(2) No se deben dejar los pines de MSEL libres, se recomienda conectarlos a VCCIO o GND, si se usa sólo JTAG se deben conectar a GND.
(3) Es necesario colocar jumpers en conector JP_JTAG.

Pin 1: 3.3V
Pin 2: Pin Cyclone II
Pin 3: GND

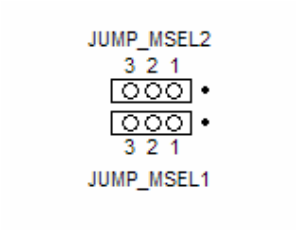
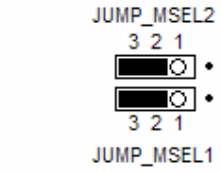


Figura 11: Esquema de conexionado de jumpers configuración

Ejemplo 1: Para configurar AS o JTAG realizar el siguiente esquema



Ejemplo 2: Para configurar PS realizar el siguiente esquema

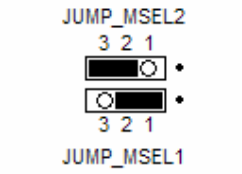


Figura 12: Ejemplos de posicionamiento de jumpers de configuración

4.2 JTAG:

4.2.1 JTAG placa aislada:

Primero es necesario ajustar los jumpers de configuración como se detalla en la tabla 13 y dejar sin jumpers el conector JP_JTAG, el mismo se encuentra entre el bus de expansión y el Header JTAG.
El conector JP_JTAG cumple la función de cerrar la cadena JTAG mediante la utilización de jumpers. Se utiliza cuando se encuentra la Placa IIE-Cyclone conectada a la Placa IIE-PCI y se quiere configurar los FPGA de ambas placas mediante protocolo JTAG.

Luego es necesario utilizar el cable ByteBlaster II conectándolo al Header JTAG de la placa (Se encuentra claramente identificado en el borde izquierdo de la placa) y al puerto paralelo (LPT1) del PC.

Al conectar el cable, el pin 1 del Header debe coincidir con la marca de color rojo en el cable del ByteBlaster II. En la figura 13 se detalla el orden de los pines del Header JTAG.

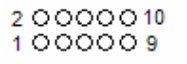


Figura 13: Header JTAG

Pin Header JTAG	Señal JTAG
1	TCK
2	GND
3	TDO
4	VCCIO
5	TMS
6	N.C.
7	N.C.
8	N.C.
9	TDI
10	GND

Tabla 14: Pinout Header JTAG

Configuración del Quartus II (Ver. 6.1 Web Edition)**:**

1) Luego de realizar la compilación completa se generan los archivos de programación para el FPGA (.sof) y elementos de memoria (.pof)

2) Seleccionar para que los pines sin usar queden en estado tri-state, recordar que los pines que no usamos pueden estar conectados a la RAM u otro dispositivo y si no los reservamos podemos terminar quemando los drivers de línea de dichos dispositivos.
   Para realizar esto ir a Settings → Device → Device & Pins Options → Unused Pins y en el campo “Reserve all unused pins” poner la opción “As input tri-state”.

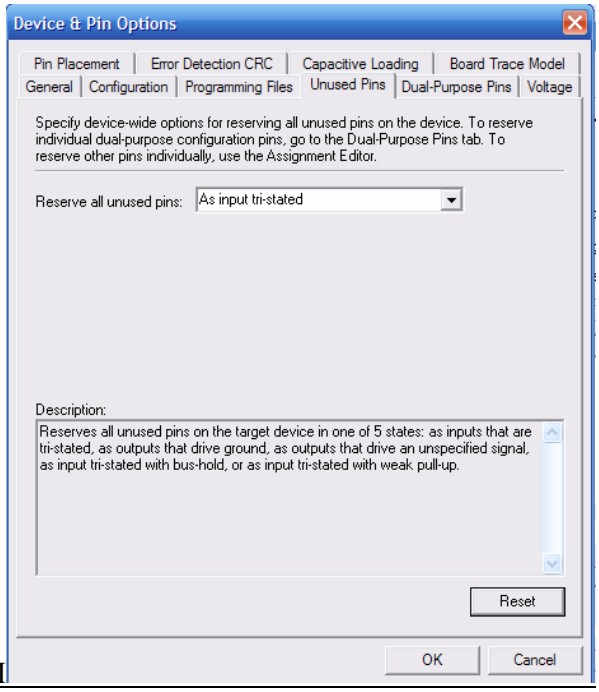
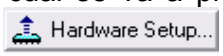


Figura 14: Captura pantalla unused pins Quartus

3) Realizar la asignación de pines en la sección Assignments Pins → Pin Planner

Figura 15: Captura pantalla asignación pines Quartus II

4) Para programar seleccionar Tools → Programmer

4-1) Seleccionar el Hardware por el cual se va a programar el FPGA, seleccionando Hardware Setup 

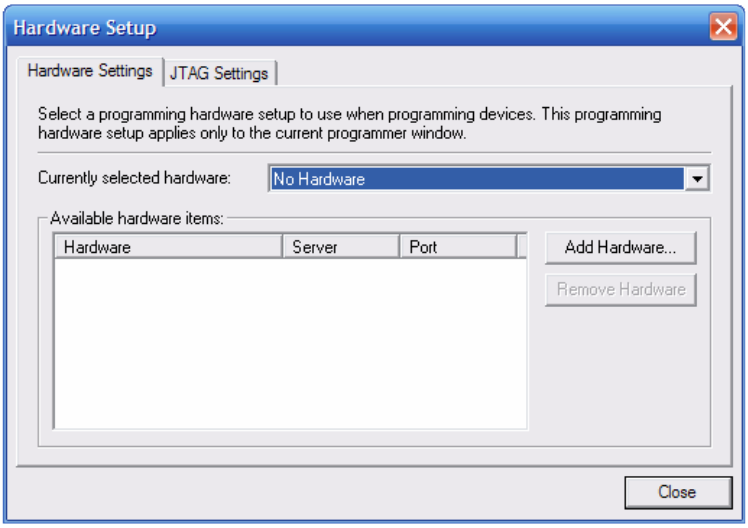


Figura 16: Captura pantalla Hardware Setup Quartus II

4-2) Luego *Add Hardware* y agregar la interfaz de configuración, en nuestro caso usamos el puerto paralelo LPT1.

4-3) Luego de elegido el puerto paralelo se puede elegir el archivo de configuración seleccionando en *Add File* el archivo correspondiente.

4-4) Seleccionar el modo de programación en modo JTAG seleccionando en Mode: JTAG

5) Por ultimo Presionar *Start* para configurar el dispositivo.

Mediante este método solo se configura el FPGA Cyclone II y la configuración se pierde cuando se le quita la alimentación al FPGA.

4.2.2 Cadena JTAG:

Se utiliza esta forma de configuración cuando la Placa IIE-Cyclone se encuentra conectada mediante los buses de expansión a la Placa IIE-PCI.
Primero es necesario ajustar los jumpers de configuración como se detalla en la tabla 13 y colocar los jumpers en todas las posiciones del conector JP_JTAG como se especifica en la figura 17.

Luego en la Placa IIE-PCI es necesario manipular los jumpers JTAG_EXP y JTAG_BRIDGE_EPC2 para lograr cerrar la cadena JTAG. En el documento DocIIEPCI.pdf (ver referencias) se detalla el procedimiento completo.
Por ultimo es necesario conectar el cable ByteBlasterMV entre el puerto paralelo del PC y el conector Header JTAG de la Placa IIE-PCI.

Figura 17: Jumpers conector JP_JTAG para cerrar la cadena JTAG

4.3 Active Serial:

Primero es necesario ajustar los jumpers de configuración como se detalla en la tabla 13, luego es necesario conectar el cable ByteBlaster II entre el puerto paralelo del PC y el conector Header_AS (que se encuentra claramente identificado en el borde izquierdo de la placa).

Al conectar el cable el pin 1 del Header debe coincidir con la marca de color rojo en el cable. En la figura 18 se detalla el orden de los pines del Header_AS.

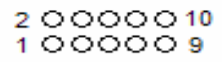


Figura 18: Header_AS

Header_AS	Señal AS
1	DCKL
2	GND
3	CONF_DONE
4	VCCIO
5	nCONFIG
6	nCE
7	DATA
8	nCS
9	ASDI
10	GND

Tabla 15: Pinout Header_AS

En el Quartus II es necesario seguir los pasos previamente detallados para configuración JTAG salvo que en el punto 4.4) hay que seleccionar el modo de programación: Active Serial Programming.
Mediante este método se carga la configuración en la memoria flash del dispositivo de configuración serie EPCS4. De esta forma al quitar la alimentación de la placa la configuración queda guardada en la memoria y al alimentar la placa nuevamente el FPGA carga la configuración de ésta.
El esquema de configuración para Fast Active Serial es igual al esquema para Active Serial, la diferencia es que se debe seleccionar los jumpers de configuración según la tabla 13.

4.4 Passive Serial:

En el esquema Passive Serial una inteligencia externa controla la configuración, la idea es que el Cyclone II es configurado vía Passive Serial por el FPGA ACEX de la Placa IIE-PCI.
El mismo puede recibir la configuración a través del bus PCI mediante el Core PCITWBM desarrollado en el Proyecto de Fin de Carrera IIE-PCI y luego mediante una máquina de estados a ser desarrollada se lo cargaría al Cyclone II.
Para permitir lo previamente detallado se conectaron pines del expansor a los pines de configuración Passive Serial del Cyclone II, en la tabla 16 se especifica dichos conexionados. Es necesario colocar jumpers en todas las posiciones del conector JP_PS1 (el mismo se encuentra entre el expansor y el Header AS).

Señal AS	Pin FPGA	Pin Expansor
DCLK	26	5
DATA	27	7
OE	143	9
nINIT	33	10
nCS	144	8

Tabla 16: Pines del expansor para configuración PS

5 APLICACIÓN

5.1 Introducción:

La aplicación consiste en el diseño, desarrollo e implementación de un algoritmo de compresión de imágenes sin pérdida en hardware.
Para diseñar nuestra aplicación nos basamos en el algoritmo LOCO-I [30], en su implementación realizada por el Jet Propulsion Laboratory del California Institute of Technology a pedido de la NASA [35] y en la estandarización del LOCO-I realizada por la ISO/ITU para la compresión de imágenes sin pérdida, conocida como JPEG-LS [31].

El LOCO-I alcanza tasas de compresión similares a las obtenidas por algoritmos basados en codificación aritmética, con un nivel de complejidad mucho menor que los anteriores. Esta característica lo hace ideal para implementarlo en hardware, siendo este el motivo principal de nuestra elección. En la tabla 17 se muestra el desempeño del LOCO con respecto a otros algoritmos de compresión sin pérdida.

IMAGEN	LOCO-I	JPEG-LS	FELICS	Lossless HUFFMAN	Lossless arithm.	CALIC arithm.	LOCO-A	PNG
Bicicleta	3.59	3.63	4.06	4.34	3.92	3.50	3.54	4.06
Café	4.80	4.83	5.31	5.74	5.35	4.69	4.75	5.28
Mujer	4.17	4.20	4.58	4.86	4.47	4.05	4.11	4.68
Herramientas	5.07	5.08	5.42	5.71	5.47	4.95	5.01	5.38
Bicicleta 3	4.37	4.38	4.67	5.18	4.78	4.23	4.33	4.84
gatos	2.59	2.61	3.32	3.73	2.74	2.51	2.54	2.82
agua	1.79	1.81	2.36	2.63	1.87	1.74	1.75	1.89
dedos	5.63	5.66	6.11	5.95	5.85	5.47	5.50	5.81

JPEG-LS es la estandarización del algoritmo LOCO-I con modificaciones menores.

Tabla 17: Resultados de Compresión (bits / muestra)

El algoritmo se basa en un esquema de dos componentes independientes, el modelado y la codificación.
El modelado infiere el comportamiento estadístico de la imagen. Se logra mediante el escaneo píxel a píxel de la imagen en un orden pre-definido (raster scan).
La etapa de codificación se hace codificando el error entre la predicción de cada píxel y su valor real, ajustado por una desviación (bias) regulada por el comportamiento estadístico de la imagen que se obtiene del modelado. La codificación que se usa es la de Golomb, que luego de ciertos procesamientos se ajusta de muy buena manera a la distribución que presenta la diferencia entre el valor real y la predicción del píxel.

En nuestra implementación no utilizamos la codificación de RLE (Run Length Encoding), que codifica de manera distinta y más eficiente las secuencias de píxeles iguales. Esta simplificación no le quita eficiencia al algoritmo ya que el mismo está pensado para trabajar con imágenes naturales, en las cuales es

muy poco probable encontrar regiones de tonalidades uniformes. Nos basamos en el trabajo realizado para la NASA [35] para tomar esta decisión.

Además, realizamos otras modificaciones sobre el algoritmo basados en el estándar JPEG-LS, estos cambios fueron introducidos para poder comparar los resultados de nuestra aplicación utilizando un programa en C que implementa JPEG-LS.
El mismo fue diseñado en el Departamento de Ingeniería Eléctrica y Computación de la Universidad de British Columbia por Faouzi Kossentini y Ismail R. Ismail. Cuenta con las opciones de comprimir y descomprimir imágenes utilizando el estándar JPEG-LS con y sin modo Run Length [47].

En los siguientes capítulos se detallará el algoritmo LOCO-I. Por más información al respecto del mismo consultar el texto “LOCO-I: A Low Complexity, Context-Based, Lossless Image Compression Algorithm” [30].

5.2 Algoritmo LOCO-I:

5.2.1 Introducción:

En este capítulo presentaremos los bloques principales del algoritmo LOCO-I e introduciremos los principales conceptos e ideas en las que se basa el mismo. En la figura 19 se muestra un esquema general compuesto por los distintos bloques que conforman el algoritmo.

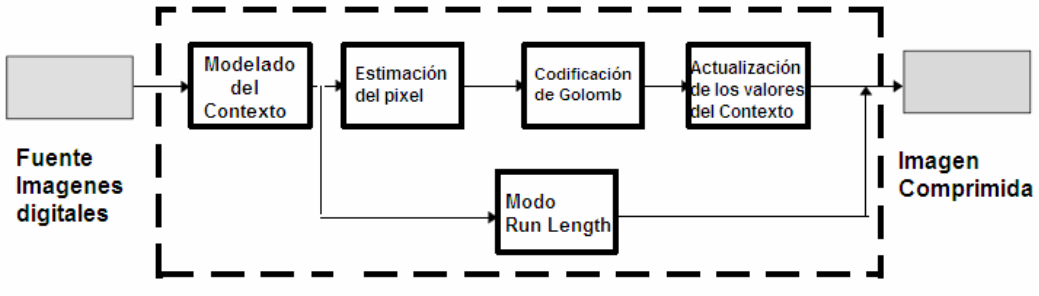


Figura 19: Bloques del algoritmo LOCO-I

El algoritmo conceptualmente se puede dividir en cinco bloques diferentes:

1) **Modelado del Contexto:** clasificación de los píxeles en distintos contextos, acorde a los valores de los píxeles previamente codificados.

2) **Estimación del píxel:** se realiza una estimación del píxel a codificar mediante el predictor fijo, luego se ajusta el mismo mediante una corrección (de aquí en adelante llamada bias) y finalmente se calcula la diferencia con respecto al valor real del píxel (error o residuo).

3) **Codificación de Golomb:** mapeo del error a un valor entero no negativo y posterior codificación usando códigos de Golomb de largo variable.

4) **Actualización de los valores del Contexto:** tomando en cuenta el valor de la muestra actual se actualiza la información almacenada para cada uno de los contextos.

5) **Modo Run Length:** modo de procesamiento en el cual se entra si las muestras que vienen siendo analizadas son iguales. Se utiliza esta característica para lograr un procesamiento más eficiente de la información.

5.2.1.1 Bloque 1:

En esta etapa se le asigna un contexto a cada uno de los píxeles de la imagen. Estos contextos están basados en el valor de 5 píxeles previamente clasificados. En la figura 20, se especifica claramente la posición de estos 5 píxeles (a, b, c, d, e), con respecto al píxel que se esta actualmente clasificando (p). Luego, estos contextos son utilizados para estimar una distribución de probabilidad condicional para el siguiente píxel a analizar.

	c	b	d
e	a	***p***	

Figura 20 : Etiquetado de los píxeles relativos al píxel (p)

El contexto se determina por los valores de Q_9 (d-b), Q_9 (b-c) y Q_9 (c-a), siendo Q_9 la siguiente función:

$$Q_9(n) = \begin{cases} q_1 \text{ si } n \geq 15 \\ q_2 \text{ si } 7 \leq n \leq 14 \\ q_3 \text{ si } 3 \leq n \leq 6 \\ q_4 \text{ si } 1 \leq n \leq 2 \\ q_5 \text{ si } n = 0 \\ q_6 \text{ si } -1 \leq n \leq -2 \\ q_7 \text{ si } -3 \leq n \leq -6 \\ q_8 \text{ si } -7 \leq -n \leq -14 \\ q_9 \text{ si } n \leq -15 \end{cases}$$

Y por los valores de Q_3(a-e) siendo Q_3 la siguiente función:

$$Q_3(n) = \begin{cases} q_1 \text{ si } n \geq 5 \\ q_2 \text{ si } n < \lfloor 5 \rfloor \\ q_3 \text{ si } n \leq -5 \end{cases}$$

Siendo los q_i los distintos niveles de cuantización, los que dependen de la cantidad de bits que se utilicen. Después de cuantizar se define el contexto, el que es generado a partir de la concatenación de los valores de Q_9 (d-b), Q_9 (b-c), Q_9 (c-a) y Q_3(a-e).

5.2.1.2 Bloque 2:

En esta etapa se realiza una estimación del píxel a ser codificado (píxel p). Cuanto más exacta sea la estimación, menor será la diferencia entre ésta y el valor real del píxel, disminuyendo de esta manera la diferencia entre ambos, que es el valor que va a ser codificado.

La estimación es generada a partir de los valores de a, b y c mediante la siguiente función:

$$\text{Estimación de p:} \begin{cases} \text{mínimo (a, b)} & \text{si c} \geq \text{máximo (a, b)} \\ \text{máximo (a, b)} & \text{si c} \leq \text{mínimo (a, b)} \\ \text{a + b - c} & \text{en otros casos} \end{cases}$$

Esta función es conocida como “predictor fijo”. La estimación final del píxel se genera sumándole el valor del bias a la salida del predictor fijo. El bias es una corrección que se utiliza para lograr un valor más exacto de p, el valor del mismo es generado en la etapa de actualización de datos.

5.2.1.3 Bloque 3:

En esta etapa se codifica sin pérdida la diferencia entre el valor real del píxel y la estimación realizada en el bloque 2 (llamamos ϵ a esta diferencia), utilizando la codificación de Golomb.
Los valores de ϵ tienen una distribución aproximada a la del tipo “two-sided geometric”. Luego se realiza un mapeo del error para lograr que el mismo pase a tener una distribución aproximada del tipo “one sided geometric”.
El mapeo se realiza para adaptar el error a las características del código de Golomb, ya que el mismo es altamente eficiente codificando señales que tienen distribuciones del tipo “one sided geometric”. Esta modificación se puede lograr de manera eficiente mediante operaciones de manipulación sobre los bits. La estructura de la codificación de Golomb permite un cálculo simple de la palabra código, sin la necesidad de recurrir a tablas de codificación como es usual para los códigos no estructurados (Ej.: código de Huffman).

5.2.1.4 Bloque 4:

En esta etapa se actualizan los datos pertenecientes al contexto al cual quedó asociado el píxel actualmente analizado (p), quedando la información asociada a cada uno de los restantes contextos sin cambio.
A modo de ejemplo, en el estándar JPEG-LS la información que se va guardando para cada uno de los contextos es la siguiente:

1) Cantidad de ocurrencias de un contexto (N).

2) Magnitud de la suma de los errores (msum).

3) Suma de los errores (rsum).

4) Valor de bias (bias).

El valor de bias se calcula mediante la siguiente fórmula:

bias = redondeo (msum/N).

Como realizar una división es una operación costosa en tiempo y recursos en el caso de JPEG-LS se opta por calcular el valor de bias utilizando un algoritmo alternativo. Para utilizar este algoritmo es necesario inicializar a cero los valores de rsum y bias al comienzo del análisis de la imagen.

5.2.1.5 Bloque 5:

En el caso en el que en la etapa 1 las diferencias en las restas de los píxeles vecinos sean igual a cero (d-b = b-c = c-a = 0) se entra en el modo de procesamiento “Run Mode”, y no se siguen las etapas previamente detalladas. Este modo procesa de manera mucho más eficiente una “corrida” de píxeles iguales.

5.2.2 Descripción detallada:

En este capítulo se detalla la forma en la cual se implementan los bloques especificados en el capítulo anterior. Se muestran los distintos sub-bloques que los componen y se especifican las relaciones entre los mismos. En la figura 21 se muestran dichas relaciones de forma gráfica.
Vale la pena mencionar previamente que la codificación de la imagen se realiza analizando la misma de izquierda a derecha y en forma descendente. Además, cada píxel es codificado de forma independiente del resto, y los parámetros de codificación para cada píxel dependen del contexto al que sean asociados y a los parámetros estadísticos de estos contextos.
Estos parámetros estadísticos son basados en los píxeles previamente codificados que pertenecen al mismo contexto. Los mismos se van actualizando cada vez que un píxel es asociado a un contexto.

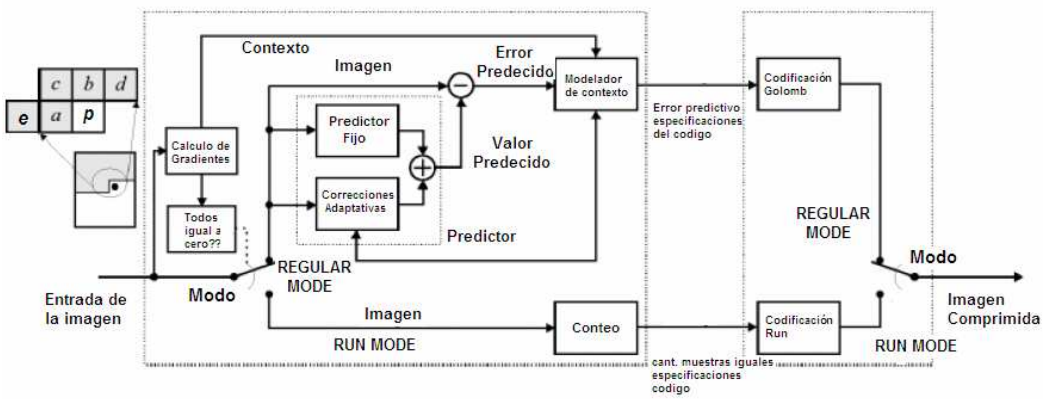


Figura 21: Algoritmo Loco-I

Lo primero que se calcula es el valor de los gradientes para el píxel que se está analizando. A partir de los mismos no sólo se determina en qué modo de procesamiento voy a entrar, sino que también en el caso del procesamiento regular se le asocia a cada píxel un contexto.
Tomando en cuenta el valor de los píxeles vecinos a, b, c, d y e (en la figura 21 se muestra la posición relativa de los mismos con respecto al píxel actualmente analizado *p*), se calculan los gradientes de la siguiente manera:

$$G_1 = d - b \qquad G_2 = b - c \qquad G_3 = c - a \qquad G_4 = a - e$$

Estos cuatro gradientes representan el nivel de actividad en torno a una muestra. Con nivel de actividad nos referimos a qué tan cerca de un borde entre tonalidades distintas se encuentra la muestra o cómo se viene degradando una tonalidad.
Si el valor de los cuatro gradientes es igual a cero, entra en el modo Run, si no, se entra en el modo Regular.

5.2.2.1 Modo Run:

Al entrar al modo Run se van contando la cantidad de muestras iguales consecutivas. Luego, esta cantidad (que puede ser igual a cero) es codificada. Dentro de este modo se deja de calcular el valor de los gradientes y por ende tampoco se determina el contexto asociado para cada píxel que caiga dentro de este modo. Si los píxeles consecutivos son interrumpidos por un fin de línea, se sale del modo Run, volviéndose a calcular el gradiente del primer píxel de la siguiente línea para determinar en qué modo se entra.

Si los píxeles consecutivos son interrumpidos por un píxel distinto (x), se sale del modo Run, y se codifica la diferencia $\epsilon = x - b \bmod a$, siendo *b* el píxel por encima del píxel *x* y *a* el píxel anterior (o sea el píxel que se viene repitiendo hasta la aparición de *x*).

En este modo se utilizan dos codificaciones distintas, una para la cantidad de muestras iguales y otra para la diferencia $\epsilon = x - b \bmod a$. La codificación de muestras iguales se realiza utilizando el código *elementary Golomb* de orden m, siendo m la cantidad de muestras iguales, mientras que la diferencia $\epsilon = x - b \bmod a$ se codifica de igual manera que en el modo regular.

Este modo logra que imágenes que contengan regiones planas (mismo valor para cada uno de los píxeles en la región) se codifiquen de forma mucho más eficiente. De todos modos este tipo de regiones sólo se presentan en imágenes que no sean naturales, por ejemplo imágenes generadas por programas de diseño.

5.2.2.2 Modo Regular:

A partir del valor de los gradientes ya calculados se calcula el contexto asociado al píxel siguiendo los siguientes pasos:

1) Se calcula los valores de $Q_9(d-b)$, $Q_9(b-c)$ y $Q_9(c-a)$ y $Q_3(b-e)$, siendo Q_9 la siguiente función:

$$Q_9(n) = \begin{cases} q_1 \text{ si } n \geq 15 \\ q_2 \text{ si } 7 \leq n \leq 14 \\ q_3 \text{ si } 3 \leq n \leq 6 \\ q_4 \text{ si } 1 \leq n \leq 2 \\ q_5 \text{ si } n = 0 \\ q_6 \text{ si } -1 \leq n \leq -2 \\ q_7 \text{ si } -3 \leq n \leq -6 \\ q_8 \text{ si } -7 \leq -n \leq -14 \\ q_9 \text{ si } n \leq -15 \end{cases}$$

y Q_3 la siguiente función:

$$Q_3(n) = \begin{cases} q_1 \text{ si } n \geq 5 \\ q_2 \text{ si } n < |5| \\ q_3 \text{ si } n \leq -5 \end{cases}$$

Los q_i son los distintos niveles de cuantización, que dependen de la cantidad de bits que se utilice.

2) Se concatenan los gradientes, quedando: $Q_9(b-c)$, $Q_9(d-b)$, $Q_9(c-a)$, $Q_3(b-e)$

3) Si el primer valor distinto de cero es negativo, se invierten los signos de $Q_9(b-c)$, $Q_9(d-b)$, $Q_9(c-a)$, $Q_3(b-e)$ logrando de esta manera el valor del contexto. En estos casos se levanta una bandera para especificar que se invirtieron los valores que generan el contexto, de esta forma los siguientes cálculos a realizar en el algoritmo pueden ser modificados teniendo en cuenta el valor de esta bandera. Esta reducción de contextos está basada en la simetría de los errores y de esta forma se logra reducir la cantidad de contextos aumentando la información de los mismos.

Luego de calcular el contexto se genera una predicción del valor del píxel a ser codificado (*p*), dicha predicción se compone de dos partes.
La primera se llama predictor fijo y se basa en los valores de los píxeles vecinos a la muestra para predecir el valor de la misma. Para hallar dicho valor se utiliza la siguiente función:

$$p_{predictor\_fijo} = \begin{cases} \text{mínimo (a, b)} & \text{si c} \geq \text{máximo (a, b)} \\ \text{máximo (a, b)} & \text{si c} \leq \text{mínimo (a, b)} \\ \text{a + b - c} & \text{en otros casos} \end{cases}$$

La motivación para utilizar este predictor es que cumple la función de detector de bordes. El mismo tiende a elegir *b* en caso de que un borde vertical exista a la izquierda del píxel *p*. Tiende a elegir *a* en caso de que exista un borde horizontal por encima del píxel *p* y elige un valor promedio en el caso de que no se detecte ningún borde.

La segunda parte del estimador es conocido como bias y su valor depende del contexto al que haya sido asignado el píxel *p.* A partir de observaciones empíricas se determinó que el residuo de un predictor fijo sobre una imagen de tonos continuos se puede modelar por una distribución del tipo *Two Side Geometric Distribution* (TSGD) centrada en cero. Estudios posteriores demostraron que al utilizar un offset con respecto al cero se consiguen resultados más precisos. Este offset se puede dividir en dos partes, una entera (bias) y otra fraccional.

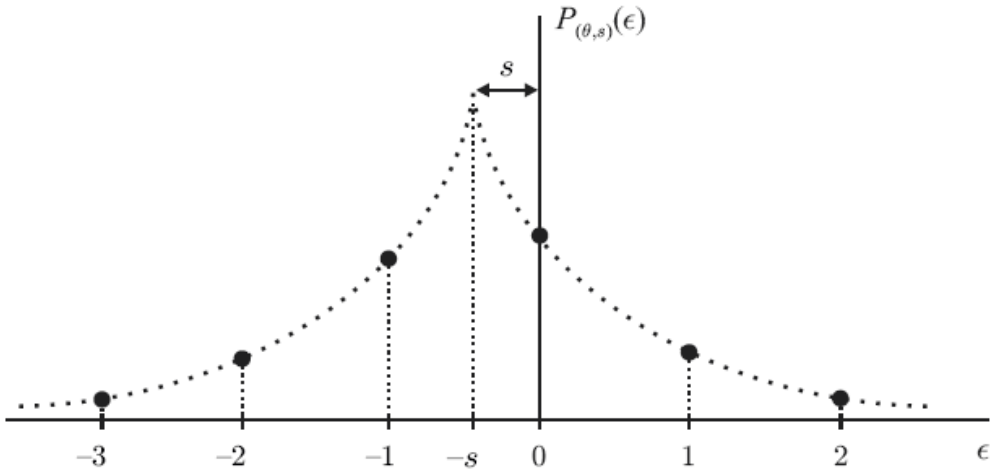


Figura 22: Distribución del tipo TSGD

Tomando *u* como el offset, *R* como su parte entera (bias) y *s* como la parte fraccional, las relacionamos entre si mediante: *u = R – s.*
Luego, la distribución de probabilidad del error asumida por el estándar LOCO-I/JPEG-LS para el residuo de un predictor fijo para cada contexto está dada por la siguiente fórmula:

$$P_{(\theta, u)}(\epsilon) = C(\theta, s)\, \theta^{|\epsilon - R + s|}$$

Siendo $C(\theta, s) = (1 - \theta) / (\theta^{1-s} + \theta^{s})$ el factor de normalización.

Luego del cálculo del predictor fijo y del bias, se calcula la estimación para el píxel p como la suma de ambos. Por lo tanto se puede concluir que el error entre el valor del píxel y nuestra estimación va a tener una distribución de probabilidad del siguiente tipo:

$$P_{(\theta, u)}(\epsilon) = C(\theta, s)\, \theta^{|\epsilon + s|}$$

Los parámetros θ y s dependen del contexto al que se le asignó el píxel, de igual manera que el valor del bias (R) depende del contexto.

Para el cálculo del valor del bias, se realiza una estimación basada en el promedio de los errores ocurridos para las muestras previamente analizadas que pertenezcan al mismo contexto de la muestra que se está analizando.

Para realizar dicho cálculo es necesario guardar la siguiente información para cada uno de los contextos existentes:

N = número de ocurrencias del contexto.

D = suma acumulativa de los errores de la predicción fija.

Luego, a partir de estos dos valores se calcula el bias como:

R = redondeo (D / N)

El redondeo es necesario porque el valor de R representa la parte entera del offset. Como realizar una división es una operación costosa en recursos y tiempo en el estándar JPEG-LS el cálculo del bias se realiza utilizando el siguiente algoritmo:

```
rsum = rsum + ε
N = N +1
if (rsum ≤ -N) {
        bias = bias - 1; rsum = rsum + N;
        si (rsum ≤ -N) rsum = -N + 1 ;
}
else if (rsum > 0) {
        bias = bias + 1 ; rsum = rsum – N ;
        si (rsum > 0) rsum = 0;
}
```

Al finalizar la etapa de cálculo de la predicción se genera el residuo de la predicción. Este valor es la resta entre el valor real del píxel y nuestra predicción. En el caso de que la bandera de signo negativo esté asociada a este píxel se toma el valor inverso del *Residuo*. Como se mencionó anteriormente este *Residuo* (ε) va a tener una distribución del tipo TSGD.

Luego, el valor del *Residuo* (ε) es codificado utilizando los códigos de Golomb, estos toman un valor entero *m* como parámetro de codificación y codifican un número *n* en dos partes:

1) una representación binaria de *n* mod *m.*
2) una representación no-aria de *n/m.*

Los códigos de Golomb son óptimos para codificar valores positivos que tengan una distribución del tipo one-side geometric (OSGD). En estos casos para cada distribución de este tipo existe un valor de *m* que genera que el código de Golomb de orden *m* (G_m) tenga en promedio el largo más corto posible de código.

El valor óptimo de *m* esta dado por:

$$m = [\log (1+ \theta) / \log (\theta^{-1})]$$

El *Residuo* (ϵ) a codificar tiene una distribución TSGD y el código que se utiliza es óptimo para codificar valores que tengan una distribución del tipo OSGD, por lo tanto es necesario realizar un mapeo para llevar el *Residuo* (ϵ) a una distribución del tipo OSGD. Esto se logra utilizando la siguiente función para el *Residuo:*

$$M(\epsilon) = \begin{cases} 2\epsilon & \text{si } \epsilon \geq 0 \\ -2\epsilon - 1 & \text{si } \epsilon < 0 \end{cases}$$

Si se toma el valor de *m* como una potencia de 2 ($m = 2^k$) se logra un procedimiento más simple de codificación y decodificación. En este caso el código para un valor *n* consiste en los k menos significativos bits de n, seguidos por el número formado por los restantes bits en representación no-aria. El largo de estos códigos pasa a ser:

$$\text{Largo código} = k + 1 + [n/2^k]$$

Estos códigos son conocidos como Golomb-Rice.

Por último, se actualiza la información referente al contexto al que fue asociado el píxel que se codificó. A modo de ejemplo, para cada contexto en el estándar JPEG-LS se almacena la siguiente información:

1) Cantidad de ocurrencias de un contexto (N).

2) Magnitud de la suma de los errores (msum).

3) Suma de los errores (rsum).

4) Valor de bias (bias).

A partir de esta información se van a calcular los parámetros estadísticos para el próximo píxel que sea asociado al contexto. Dependiendo del tamaño de la imagen y del enfoque que se le quiera dar a la compresión; se puede tanto mantener la misma tabla desde el principio del análisis de la imagen, o se puede reiniciar los valores de la misma cada *x* (variable a determinar) píxeles codificados. De esta manera no solo se reduce el tamaño de la tabla, sino que también se logra que solo las regiones de la imagen cercanas al píxel a codificar incidan en la característica de la codificación.

5.3 Implementación de LOCO-I en VHDL

5.3.1 Introducción:

Al algoritmo LOCO-I le realizamos una serie de modificaciones. Las mismas tienen como objetivo simplificar la implementación del algoritmo en VHDL y permitir la decodificación de la imagen comprimida en el FPGA. Para esto utilizamos un decodificador desarrollado en C, que implementa la norma JPEG-LS sin el modo RUN.

El primer cambio realizado es que no implementamos el bloque de Run Length, la función del mismo es la de codificar regiones planas (mismo valor de píxel en toda la región) de manera más eficiente que si se utilizara el modo Regular. El fundamento para prescindir del bloque Run Length es que el mismo complicaría la implementación en Hardware, y que no aprovecharíamos sus beneficios ya que en imágenes naturales no es común la aparición de regiones con el mismo valor de píxel.
El segundo cambio fue modificar la cantidad de regiones del cuantizador. En nuestra implementación utilizamos los siguientes niveles:

$$Q_7(n) = \begin{cases} q_1 \text{ si } n \geq 13 \\ q_2 \text{ si } 5 \leq n \leq 12 \\ q_3 \text{ si } 2 \leq n \leq 4 \\ q_4 \text{ si } -1 \leq n \leq 1 \\ q_5 \text{ si } -2 \leq n \leq -4 \\ q_6 \text{ si } -5 \leq n \leq -12 \\ q_7 \text{ si } n \leq -13 \end{cases}$$

Este cambio se realizó para poder representar todos los niveles de cuantización con solo 3 bits y simplificar de esta manera el procesamiento del algoritmo. Además, se disminuye la cantidad de contextos de 1094 a 515 lo que determina una menor necesidad de memoria para guardar la información asociada a los contextos.

Para realizar estos dos primeros cambios nos basamos en la implementación del LOCO-I realizada para la NASA [35].
Los siguientes cambios mencionados se realizaron para compatibilizar nuestra implementación con el decodificador utilizado:

1) No utilizamos el píxel *e* para determinar el contexto asociado al píxel a ser codificado. Esto determina que para determinar el contexto utilicemos solamente: Q_7 (b-c), Q_7 (d-b), Q_7 (c-a). Este cambio termina llevando la cantidad de contextos que utilizamos a 172.

2) Utilizamos el siguiente mapeo para llevar el *Residuo* de una distribución TSGD a una OSGD:

if (k=0) && (2B <= -N) {
    if (error >= 0)
        M(ϵ)= 2ϵ + 1
    else
        M(ϵ) = -2ϵ – 2
    }
else {
    if (error >= 0)
        M(ϵ)= 2ϵ
    else
        M(ϵ) = -2ϵ – 2
    }

3) Para poder calcular el gradiente para los bordes de la imagen, agregamos una columna de píxeles a la izquierda de la imagen, una columna de píxeles a la derecha de la imagen y una fila de píxeles por encima de la imagen. En la figura 23 se detallan dichos agregados. El valor de dichos píxeles queda determinado por:

Fila superior: I(0, j) = 0
Columna izquierda: I(i, 0) = I(i -1,1)
Columna derecha: I(i, j) = I(i-1,,j)

Tomamos a la imagen como una matriz I de *i* filas y *j* columnas.

0	**0**	**0**	**0**	**0**	**0**	**0**
0	15	23	22	8	23	**23**
15	5	31	21	8	5	**5**
5	32	5	7	7	15	**15**
32	11	14	18	21	17	**17**
11	26	8	11	14	14	**14**
26	8	23	19	12	15	**15**
8	9	26	22	21	25	**25**

En celeste se muestran los píxeles agregados a la imagen (en blanco).

Figura 23: Agregado a la imagen para el procesamiento de bordes.

5.3.2 Características de nuestra implementación:

Implementa: algoritmo de compresión sin pérdida LOCO-I modificado.

Lenguaje: VHDL.

Características de la imagen a procesar: Imagen en escala de grises, de tamaño 100 x 100 y 8 bits/píxel.

Tasa de compresión: 35.15 %

Tiempo de compresión: 5.3 ms

Throughput: 1.88 Mpíxeles/seg

Frecuencia máxima de funcionamiento: 24 MHz

Cantidad de elementos lógicos utilizados: 600

Cantidad de registros: 208

Memoria utilizada: 8192 bits

5.3.3 Diagrama de bloques:

Nuestra implementación del LOCO-I, toma como señales de entrada:

- los 5 píxeles necesarios para la codificación: a, b, c, d, p.
- *we_ram* que activa por alto la actualización de los valores para cada contexto (context_ram).
- *comienzo* que activa por alto el comienzo de la codificación de Golomb.

Y las señales de salida son:

- *error_cod*: salida del codificador.
- *start*: bandera que indica durante un período de reloj el comienzo de la codificación (activa por alto)
- *end:* bandera que indica durante un período de reloj el fin de la codificación (activa por alto).

En la figura 24 se muestran los principales bloques de nuestra implementación y las señales que los relacionan.

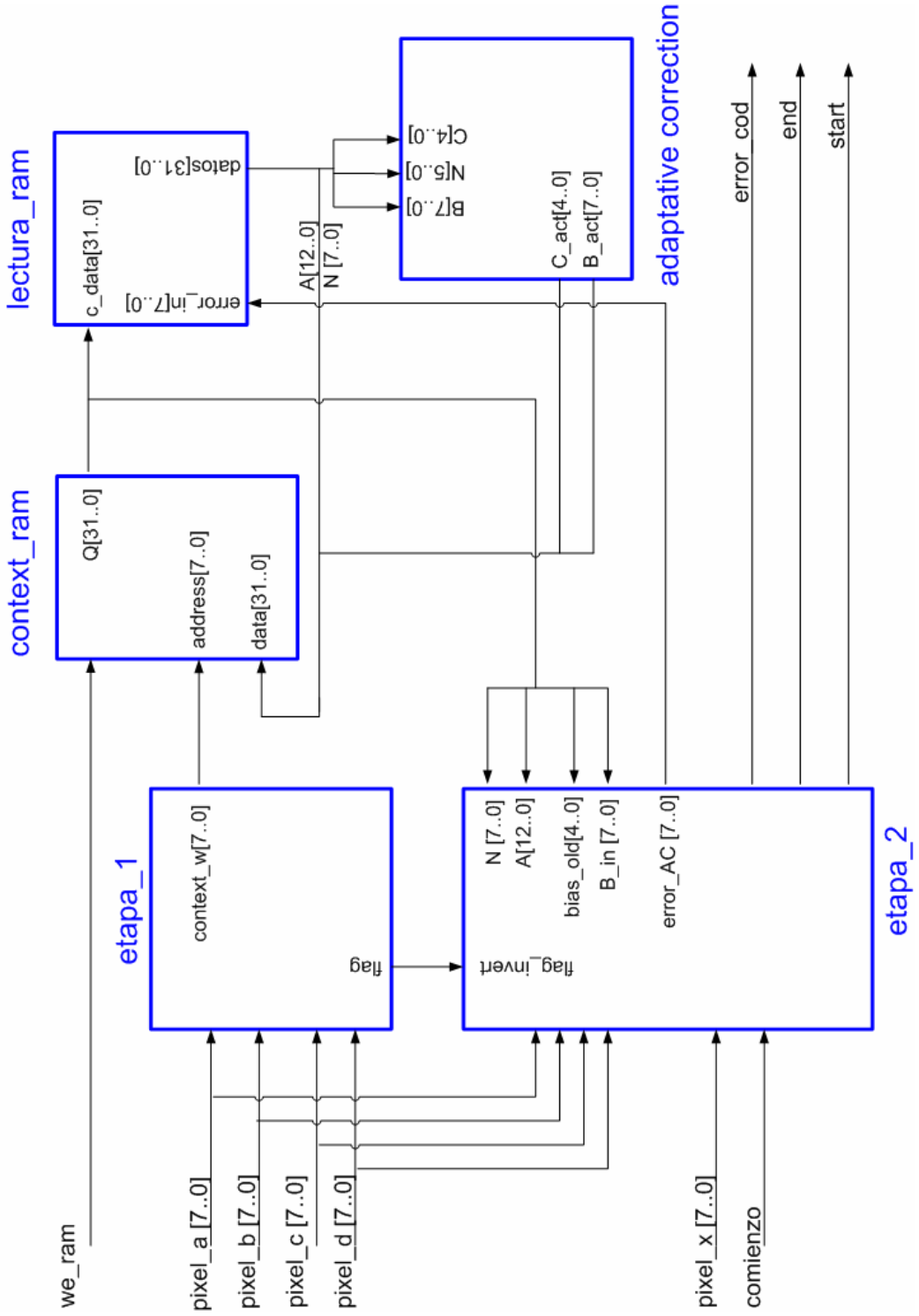


Figura 24: Diagrama de bloques de la implementación.

5.3.3.1 etapa_1:

Señales de entrada:

nombre: pixel_a
tipo: std_logic_vector [7..0]
descripción: valor del píxel a

nombre: pixel_b
tipo: std_logic_vector [7..0]
descripción: valor del píxel b

nombre: pixel_c
tipo: std_logic_vector [7..0]
descripción: valor del píxel c

nombre: pixel_d
tipo: std_logic_vector [7..0]
descripción: valor del píxel d

Señales de salida:

nombre: context_w
tipo: std_logic_vector [7..0]
descripción: contexto asociado al píxel a codificar

nombre: flag
tipo: std_logic
descripción: bandera que indica que se invirtió el signo del contexto

Funciones:

- Calcula los gradientes.
- Cuantiza el valor de los gradientes.
- Arma la palabra contexto.

Diagrama de bloques:

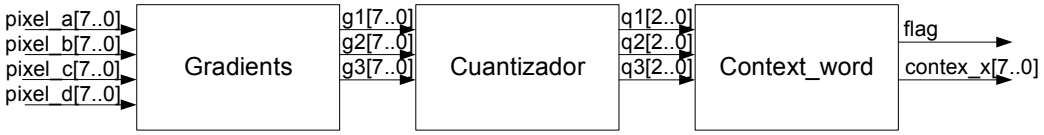


Figura 25: Diagrama de bloques de la etapa_1

Simulación:

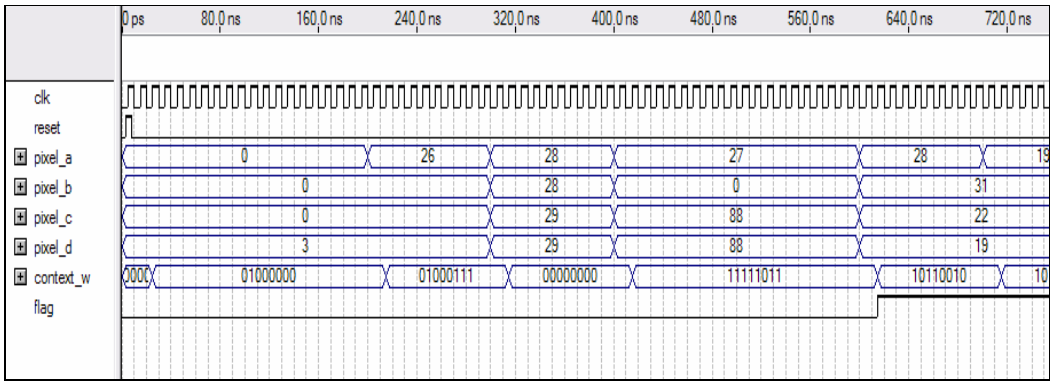


Figura 26: Simulación etapa 1

En la figura 26 se observa un ejemplo del comportamiento de la etapa_1, en el mismo se calcula y cuantiza el gradiente para luego formar la palabra context_w.
En el ejemplo en t=160ns se tiene que g1=d-b= 3 que se cuantiza con 010b, g2=b-c=0, g3=c-a=0 formando el vector context_w=010000000b.

5.3.3.2 etapa_2:

Señales de entrada:

nombre: pixel_a
tipo: std_logic_vector [7..0]
descripción: valor del píxel a

nombre: pixel_b
tipo: std_logic_vector [7..0]
descripción: valor del píxel b

nombre: pixel_c
tipo: std_logic_vector [7..0]
descripción: valor del píxel c

nombre: pixel_x
tipo: std_logic_vector [7..0]
descripción: valor del píxel a ser codificado

nombre: bias_old
tipo: std_logic_vector [4..0]
descripción: valor del bias actual, sin actualizar

nombre: B_in
tipo: std_logic_vector [7..0]
descripción: suma de los errores actuales, sin actualizar

nombre: comienzo
tipo: std_logic
descripción: determina el comienzo de la codificación de Golomb, activa por alto

nombre: flag_invert
tipo: std_logic
descripción: determina que se invirtió el signo de la palabra contexto, activa por alto

nombre: N
tipo: std_logic_vector [7..0]
descripción: cantidad de ocurrencias del contexto

nombre: A
tipo: std_logic_vector [7..0]
descripción: suma acumulada de los errores en valor absoluto, sin actualizar

Señales de salida:

nombre: error_cod
tipo: std_logic
descripción: salida del codificador

nombre: error_AC
tipo: std_logic_vector [7..0]
descripción: error previo al mapeo

nombre: start
tipo: std_logic
descripción: bandera que indica durante un período de reloj el comienzo de la codificación, activa por alto

nombre: end
tipo: std_logic
descripción: bandera que indica durante un período de reloj el fin de la codificación, activa por alto

Funciones:

- Calcula el valor de la predicción fija.
- Calcula el valor final de la predicción sumándole a la predicción fija el valor del bias.
- Realiza el mapeo del error.
- Codifica el residuo, utilizando codificación de Golomb.

Diagrama de bloques:

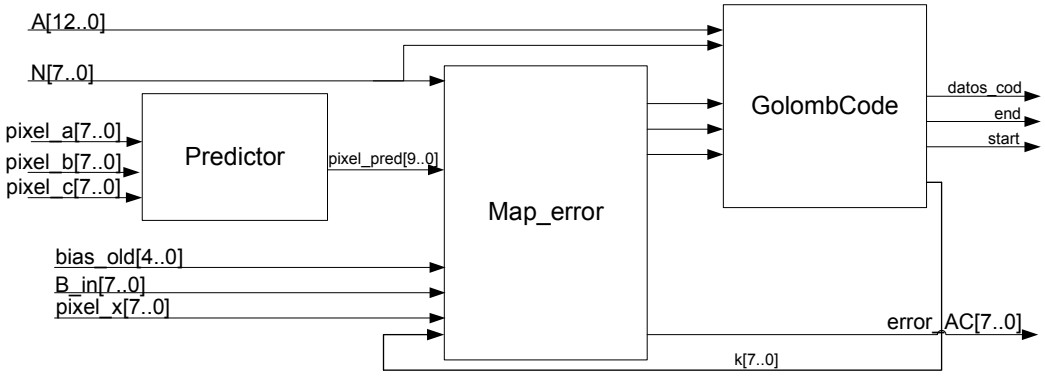


Figura 27: Diagrama de bloques de la etapa_2

Simulación:

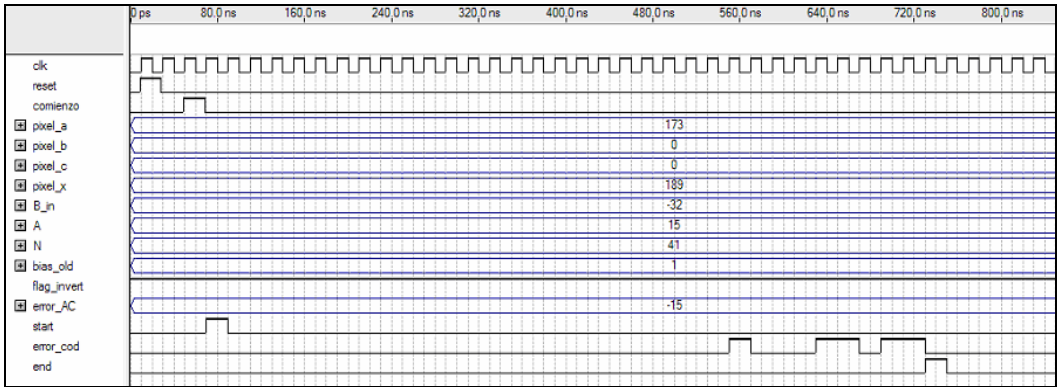


Figura 28: Simulación de la etapa 2

En la figura 28 se observa la simulación de la etapa 2. Los valores de los píxeles de entrada están expresados en decimal.
En primera instancia la etapa 2 calcula el valor del predictor fijo (pixel_pred = pixel_a + pixel_b - pixel_c = 173). Luego se genera el valor de la predicción final, el mismo sale de la suma del predictor fijo y el bias, luego dicho valor se le resta al valor real del píxel, quedando dicha diferencia en –17 (signo de menos debido a que la bandera de flag_invert está en 1).
El error es mapeado en el bloque Map_error para adaptar las características del mismo a las de la codificación de Golomb. De aquí obtenemos como resultado el valor 32. De los valores de A y N se desprende que k=0.
Ya calculados los valores de k y la salida del mapeo se genera la salida codificada por Golomb (error_cod), la misma es serial y está compuesta por 23 ”ceros” + 100011011.

5.3.3.3 context_ram

Memoria RAM interna del FPGA creada a partir del MegaWizard Plug-in Manager del Quartus II.

Características:

- Palabras: 256
- Ancho de palabra: 32 bits
- Entradas:
    - data: std_logic_vector [31..0]
    - address: std_logic_vector [7..0]
    - wren: std_logic
    - clock: std_logic

- Salidas:
    - q: std_logic_vector [31..0]

5.3.3.4 lectura_ram

Señales de entrada:

nombre: c_data
tipo: std_logic_vector [31..0]
descripción: datos provenientes de la RAM

nombre: error_in
tipo: std_logic_vector [7..0]
descripción: error entre el valor actual del píxel y la predicción

Señales de salida:

nombre: datos
tipo: std_logic_vector [31..0]
descripción: salida de datos hacia la RAM y el bloque adaptative correction

Funciones:

Bloque cuya función es leer de la memoria (context_ram) los datos asociados a cada contexto, esperar por la actualización de los mismos y escribirlos nuevamente en la memoria. Los datos que son actualizados en este bloque son:

- count es incrementado en uno
- msum se le suma el valor absoluto del error_in
- rsum se le suma error_in

En el caso de que count sea igual a 64, los valores de count, msum y rsum son divididos entre dos. Esto es necesario para que los valores previamente mencionados no superen el máximo posible que se puede representar.

5.3.3.5 adaptative correction

Señales de entrada:

nombre: B
tipo: std_logic_vector [7..0]
descripción: suma de los errores

nombre: N
tipo: std_logic_vector [5..0]
descripción: cantidad de ocurrencias del contexto

nombre: C
tipo: std_logic_vector [4..0]
descripción: valor de bias

Señales de salida:

nombre: C_act
tipo: std_logic_vector [4..0]
descripción: valor actualizado de bias

nombre: B_act
tipo: std_logic_vector [7..0]
descripción: valor actualizado de la suma de los errores

Funciones:

Calcula los nuevos valores de bias y rsum.

5.3.4 Prueba de la implementación

Para poder probar nuestra implementación fue necesario desarrollar los siguientes bloques en VHDL:

- mem_pixeles_bordes.vhd
- MaquinaDeControl.vhd
- Mem_Rom_image.vhd
- Conformador.vhd
- Cont_imagen.vhd

1) mem_pixeles_bordes:

Señales de entrada:

nombre: dato_in
tipo: std_logic_vector [7..0]
descripción: datos de entrada provenientes de la ROM

nombre: enable
tipo: std_logic
descripción: si está activa (“1”) hay un dato válido en dato_in

nombre: fin
tipo: std_logic
descripción: se activa por alto cuando se terminó de leer toda la ROM

Señales de salida:

nombre: pixel_a
tipo: std_logic_vector [7..0]
descripción: valor del píxel a

nombre: pixel_b
tipo: std_logic_vector [7..0]
descripción: valor del píxel b

nombre: pixel_c
tipo: std_logic_vector [7..0]
descripción: valor del píxel c

nombre: pixel_d
tipo: std_logic_vector [7..0]
descripción: valor del píxel d

nombre: pixel_x
tipo: std_logic_vector [7..0]
descripción: valor del píxel x

Funciones:

Registros más FIFO para armar los píxeles a, b, c, d de la imagen a comprimir, adicionalmente se encarga de agregar las condiciones de bordes que se utilizan en JPEG-LS.

2) MaquinaDeControl:

Señales de entrada:

nombre: cont_image
tipo: std_logic
descripción: se activa por alto cuando se finaliza de leer la ROM

nombre: start
tipo: std_logic
descripción: señal proveniente del bloque *UnaPasadaLoco* que indica el comienzo de la codificación

nombre: end
tipo: std_logic
descripción: bandera que indica durante un período de reloj el fin de la codificación, activa por alto

Señales de salida:

nombre: ena_pixel
tipo: std_logic
descripción: se activa por alto para leer un nuevo píxel

nombre: ena_contador
tipo: std_logic
descripción: habilita el contador que recorre las direcciones de la ROM

nombre: comienzo
tipo: std_logic
descripción: se activa por alto para comenzar la codificación de Golomb

nombre: reg_salida_ena
tipo: std_logic
descripción: activa por alto cuando hay datos válidos en la salida error_cod del bloque *UnaPasadaLoco*

nombre: we_ram
tipo: std_logic
descripción: se activa por alto para escribir en context_ram cuando hay datos actualizados nuevos.

Funciones:

Máquina de Control que lee de ROM el contenido de la imagen y pasa los píxeles al bloque *UnaPasadaLoco*

3) Mem_Rom_image:

Señales de entrada:

nombre: address
tipo: std_logic_vector [13..0]
descripción: direccionamiento de la ROM

Señales de salida:

nombre: q
tipo: std_logic_vector [7..0]
descripción: datos de la ROM

Funciones:

Memoria ROM con el contenido de la imagen.

4) Conformador:

Señales de entrada:

nombre: dato_in
tipo: std_logic
descripción: datos seriales de la salida del codificador

nombre: enable
tipo: std_logic
descripción: activa por alto cuando se tienen datos válidos en dato_in

Señales de salida:

nombre: dato_out
tipo: std_logic_vector [7..0]
descripción: datos de la salida del codificador agrupados en octetos

nombre: dato_val
tipo: std_logic
descripción: activa por alto cuando se tiene un dato válido en dato_out

Funciones:

Bloque en el cual se forman palabras de 8 bits con los datos válidos de la codificación. En el mismo se insertan marcadores especificados por el estándar JPEG-LS, el mismo consiste en agregar un 0 luego de detectar un byte formado por X'FF

5) Cont_imagen

Señales de entrada:

nombre: cnt_en
tipo: std_logic
descripción: habilita el contador, activa por alto

nombre: sclr
tipo: std_logic
descripción: borra el contador en forma sincrónica

Señales de salida:

nombre: q
tipo: std_logic_vector [13..0]
descripción: salida del contador que maneja las direcciones de la ROM

Funciones:

Contador que recorre las direcciones de la ROM.

En la figura 29 se presentan los bloques anteriormente mencionados y las señales que los relacionan.

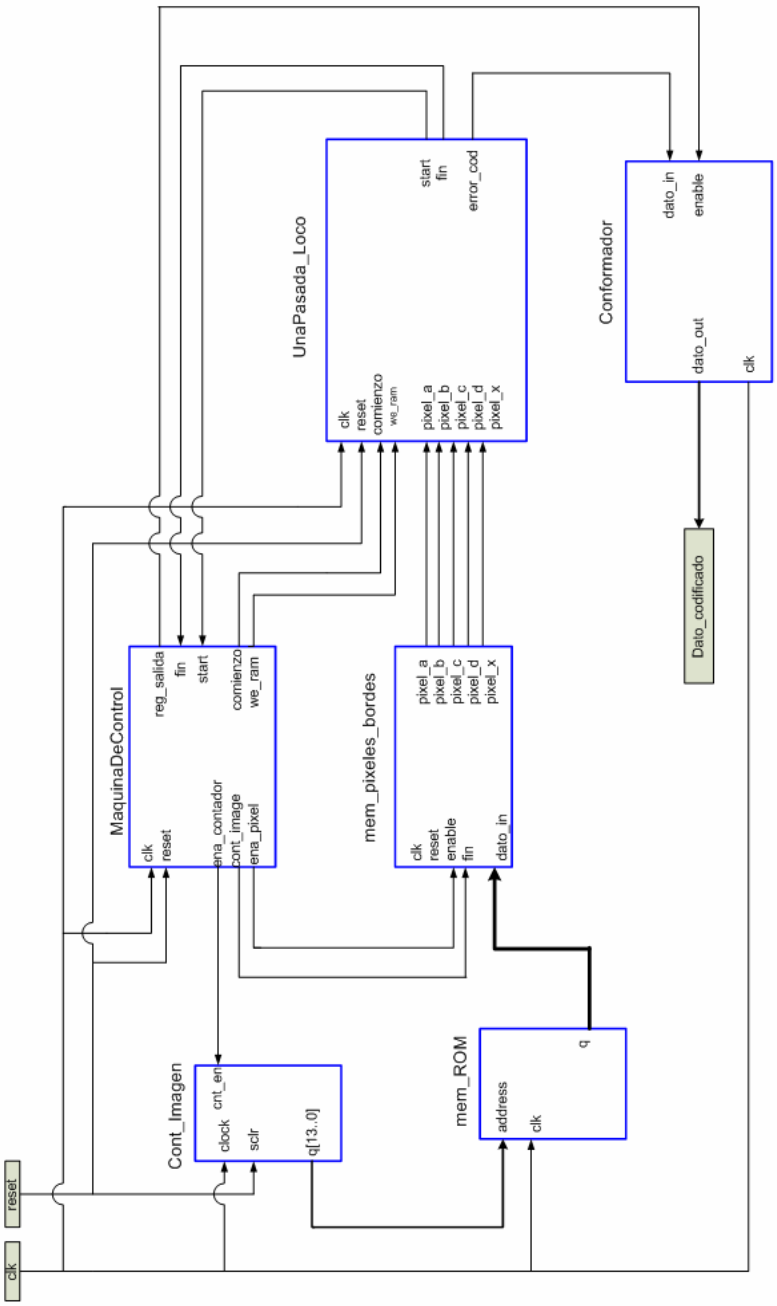


Figura 29: Diagrama de bloques de la prueba para la implementación

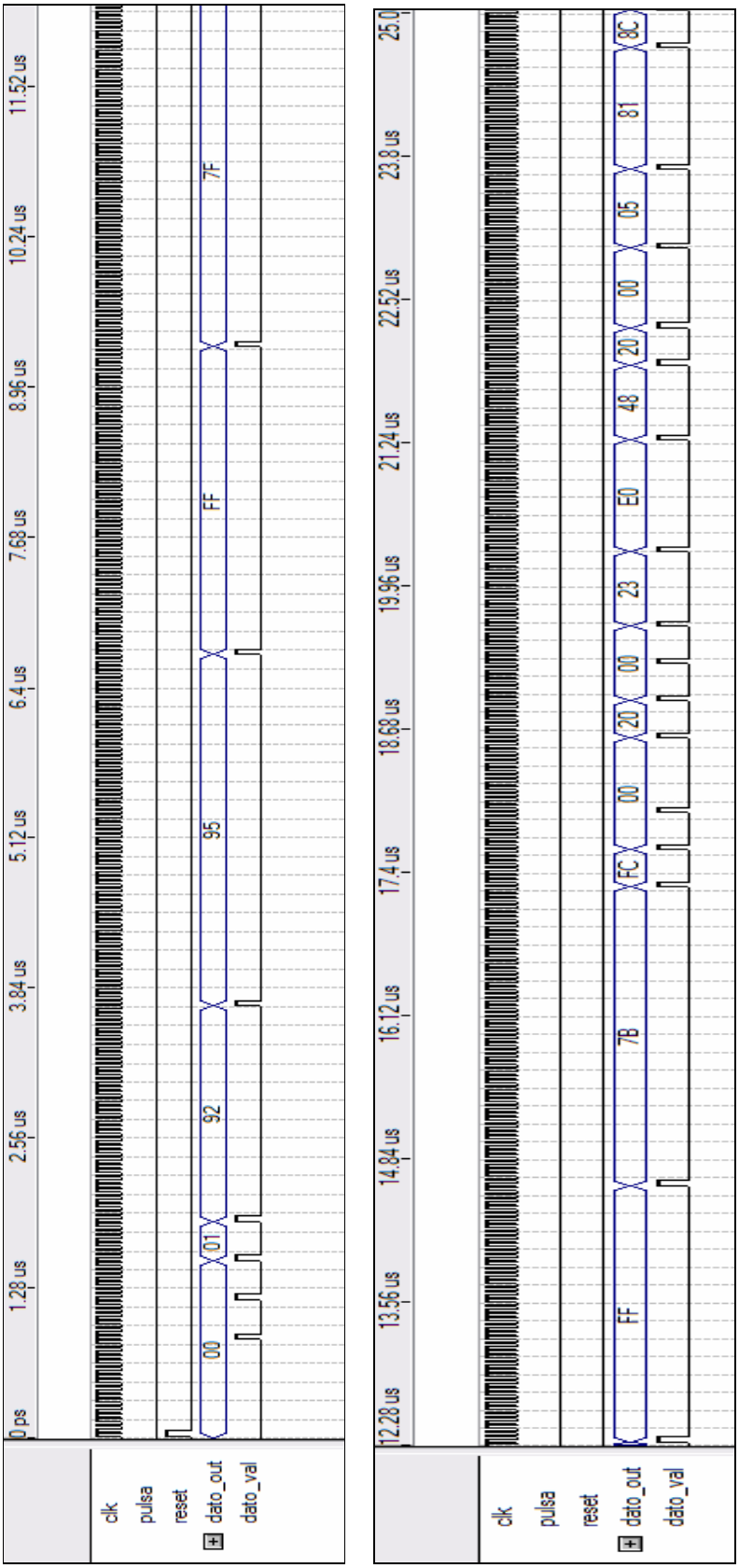


Figura 30: Diagrama de tiempos de la prueba de la implementación.

En la figura 30 se observa en la señal dato_out la salida en hexadecimal del compresor, teniéndose un dato válido cada vez que dato_val está en uno. Se muestran los primeros 25 us, donde se obtienen los primeros 23 Bytes de la compresión, siendo los mismos: *00 00 01 92 95 FF 7F FF 7B FC 00 00 20 00 00 23 E0 48 20 00 05 81 BC*.
Para la simulación tomamos una imagen en formato pgm, de tamaño 100 píxeles por 100 píxeles y resolución de 8 bits/píxel. En la figura 31 se muestra dicha imagen.

Figura 31: Imagen original: marte.pgm

El tamaño de la imagen original es de 10000 Bytes y luego de ser comprimida por nuestra implementación del algoritmo pasa a ser de 6514 Bytes.
Para corroborar nuestro algoritmo utilizamos una implementación de JPEG-LS diseñada en el Departamento de Ingeniería Eléctrica y Computación de la Universidad de British Columbia. El mismo fue desarrollado en C y le realizamos una serie de modificaciones para que se adapte a nuestra implementación; por más detalles ver Anexo 5.
Comprimimos la imagen con la etapa de compresión de dicho software y la comparamos con la salida de nuestro compresor. Verificamos Byte a Byte y comprobamos el correcto funcionamiento de nuestra implementación.
También verificamos que al utilizar el descompresor con los datos obtenidos se obtiene nuevamente la imagen original.
En la Tabla 18 se muestra los resultados comparados entre nuestra implementación del LOCO-I y el estándar JPEG-LS. Las imágenes utilizadas son de 512 x 512 píxeles y con 8 bits/píxel.

Imagen	Tamaño(*)	JPEG-LS (*)	Compresión (**)	Imp. LOCO (*)	Compresión(**)
Fish	262.182	113.748	3,471	114.197	3,485
Vacaciones	262.182	186.231	5,683	186.571	5,694
Jardin	262.182	162.569	4,961	162.707	4,965
IIE-Cyclone	262.182	117.323	3,580	117.603	3,589

(*) bytes
(**) bits/píxel

Tabla 18: Comparación JPEG-LS vs Implementación LOCO-I

En la figura 32 se muestran las imágenes utilizadas para la comparación entre la compresión realizada con el estándar JPEG-LS y la realizada por nuestra implementación.

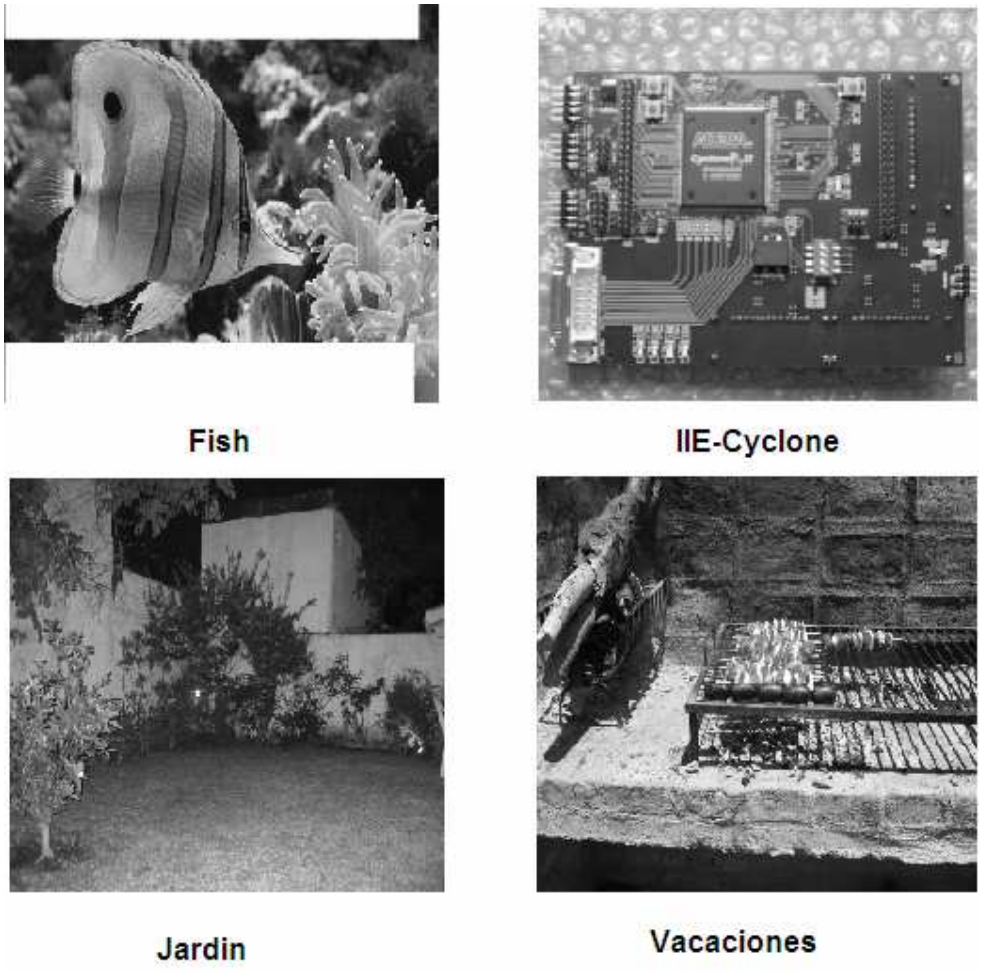


Figura 32: Imágenes utilizadas para la comparación

6 TRABAJOS FUTUROS

Para seguir ampliando el trabajo sobre la Plataforma IIE-Cyclone recomendamos los siguientes pasos a seguir:

1. Comenzar los trabajos de interconexión con la Placa IIE-PCI probando configurar ambos FPGA mediante la cadena JTAG (en el capitulo 4 Configuración se detallan los pasos necesarios para realizar dicha tarea).

2. Probar la conexión entre los pines I/O de ambos FPGA. Para realizar dicha tarea conviene montar un diseño de prueba en el Cyclone II que se encargue de enviar datos y otro sobre el ACEX que se encargue de recibir dichos datos.

3. Montar una aplicación en el Cyclone II que permita mediante los pines I/O de conexión con el ACEX recibir el direccionamiento de la SDRAM de la Placa IIE-Cyclone.

4. Configurar el Cyclone II vía Passive Serial utilizando el ACEX como inteligencia externa encargada de la configuración. El mismo recibiría la configuración del Cyclone II a través del bus PCI y mediante una maquina de estados a ser desarrollada se lo cargaría al Cyclone II (por más información al respecto ver capitulo 4 Configuración).

7 CONCLUSIONES

Con respecto a la etapa de Hardware se logró el objetivo inicial de diseñar y fabricar una plataforma basada en lógica programable.
La elección del FPGA resultó ser muy buena, ya que el modelo elegido de la familia Cyclone II terminó siendo un chip con muy buena capacidad, facilidad de configuración, facilidad de montaje y el hecho de que contenga PLLs internos facilitó mucho el diseño de la placa.
La etapa de soldadura se cumplió sin inconvenientes mayores y fue posible de realizar sin la necesidad de herramientas sofisticadas.
La elección de las interfaces nos permitió realizar una serie de pruebas muy útiles para la implementación.
El modo de trabajo standalone brinda la opción de una plataforma de pruebas muy sencilla de utilizar y fácil de configurar mientras que el modo de trabajo con la Placa IIE-PCI da la opción de contar con una plataforma conjunta de trabajo con interfaz PCI.

Se logró desarrollar una variante del algoritmo LOCO-I adaptado a las características del hardware. Esta aplicación requiere pocos recursos del FPGA y alcanza tasas de compresión de 3.51 bits/píxel, similares a las obtenidas por el estándar JPEG-LS. Con respecto a la velocidad de procesamiento la misma es de 1.88 Mpíxeles/seg, cercana a la alcanzada por la implementación en hardware del LOCO-I realizada para la NASA.

8 REFERENCIAS

CYCLONE II

[1]Altera, “Low Power Design Techniques”, versión 1, Agosto 2005. Disponible: http://www.altera.com/literature/lit-cyc2.jsp

[2] Altera, “Configuration Handbook, Volume 2”, versión 2.2, 2005.
Disponible: http://www.altera.com/literature/lit-cyc2.jsp

[3] Altera, “Cyclone II Device Handbook, Volume 1”, versión 2.2 2005.
Disponible: http://www.altera.com/literature/lit-cyc2.jsp

[4] Altera, “Altera Device Package Information”,versión 14.1, Octubre 2005.
Disponible: http://www.altera.com/literature/lit-cyc2.jsp

[5] Altera, “altpll Megafunction”, versión 3.0, Diciembre 2004.
Disponible: http://www.altera.com/literature/lit-cyc2.jsp

[6] Altera, “High-Speed Differential Interfaces in Cyclone II Devices”, versión 2.1, Noviembre 2005.
Disponible: http://www.altera.com/literature/lit-cyc2.jsp

[7] Altera, “Power Play Early Power Estimator User Guide for Cyclone II FPGA´s”, versión 1.0, Diciembre 2005.
Disponible: http://www.altera.com/literature/lit-cyc2.jsp

DISEÑO DE PCB

[8] Howard Johnson, Martín Graham, “High Speed Digital Design: A Handbook of Black Magic”, Prentice Hall, 1993.

[9] Christopher T. Robertson, “Printed Circuit Boards”, Prentice Hall, Octubre 2003.

[10] Douglas Brooks, “Signal Integrity Issues and Printed Circuit Board Design”, Prentice Hall, Junio 2003.

[11] Howard Johnson, Martin Graham, “High Speed Signal Propagation: Advanced Black Magic”, Prentice Hall, Febrero 2003.

[12] Eric Bogatin, “Signal Integrity-Simplified”, Prentice Hall, Septiembre 2003.

[13] Tim Williams, “The Circuit Designer´s Companion”, Elsevier, 1991

[14] S. Fernandez, C. Mondueri, “IIE-PCI: Una plataforma de desarrollo para el bus PCI”, Facultad de Ingeniería-Universidad de la Republica, Diciembre 2003.

[15] S. Gomez, J. Saporiti, A. Villavedra, “PGVIRTEX: PCI-BOARD, Facultad de Ingeniería-Universidad de la Republica.

[16] M. Groeneveld, “Sendero Evaluation Kit User Manual”, versión 1.2, Enero 2006. Disponible en http://www.alearep.com/images/Sendero_Flyer_RB.pdf

MEMORIA DE CONFIGURACIÓN

[17] Altera, “Altera Configuration Devices”, versión 2.2, Agosto 2005. Disponible: http://www.altera.com/literature/lit-cyc2.jsp

[18] Altera, “Serial Configuration Devices (EPCS1, EPCS4,EPCS16 & EPCS64)”, versión 1.5, Agosto 2005.
Disponible: http://www.altera.com/literature/lit-cyc2.jsp

SDRAM

[19] Micron, “ Synchronous Dram MT48LC4M32B2- 1 MEG x 32 x 4 Banks”, 2001.Disponible:download.micron.com/pdf/datasheets/dram/sdram/128MbSDRAMx32.pdf

[20] J. Saporiti, A. Villavedra, “Controlador de SDRAM”, Instituto de Ingeniería Eléctrica, Universidad de la Republica, 2003.

CABLE BYTEBLASTER

[21] Altera, “ByteBlaster Paralel Port Download Cable”, versión 2.01, Febrero 1998. Disponible: www.altera.com/literature/ug/ug_bbii.pdf

[22] Altera, “ByteBlaster II Download Cable User Guide”, versión 1.1, Diciembre 2004. Disponible: www.altera.com/literature/ug/ug_bbii.pdf

FUENTES DC

[23] Texas Instruments,“Power Management Reference Guide for Altera FPGAs and CPLDs”, 2005.
Disponible:
www.national.com/appinfo/power/files/NationalAlteraDesignGuide.pdf

[24] Texas Instruments,“14-A 5-V Input Adjustable Integrated Switching Regulator”, Junio 2003. Disponible:
http://focus.ti.com/lit/ds/symlink/pt6461.pdf

RS232

[25] Texas Instruments,“Low-Voltage, Single-Supply 232-Standard Interface Solutions”, Septiembre 2000.
Disponible:http://focus.ti.com/lit/an/slla083a/slla083a.pdf

CAMERA LINK

[26] “Specifications of the Camera Link Interface Standard for digital Cameras and Frame Grabbers”, versión 1.1, Enero 2004. Disponible: www.alacron.com/downloads/vncl98076xz/CameraLinkSPEC.pdf

[27] Jon Brunetti, Brian Von Herzen, “The LVDS I/O Standard”, versión 1.1, Noviembre 1999. Disponible: www.xilinx.com/bvdocs/appnotes/xapp230.pdf

[28] Jon Brunetti, Brian Von Herzen, “Virtex-E LVDS Drivers & Receivers: Interface Guidelines”, versión 1.0, Octubre 1999. Disponible: www.xilinx.com/bvdocs/appnotes/xapp232.pdf

[29] “LVDS System Data Framing”, versión 1.0, Diciembre 2000. Disponible: www.xilinx.com/bvdocs/appnotes/xapp238.pdf

ALGORITMO LOCO-I

[30] Marcelo J. Weinberger, G. Seroussi, G Sapiro, “LOCO-I A low complexity, context-based, lossless image compression algorithm”, Proceedings DCC’96, IEEE Comput. Soc. Press., Los Alamitos, California, 1996, pág. 140-9.

[31] Marcelo J. Weinberger, G. Seroussi, G Sapiro, “The LOCO-I lossless Image Compression Algorithm: Principles and Standardization into JPEG-LS”, IEEE Transactions on Image Processing, Agosto 2000, Vol. 9(8), pág. 1309-24.

[32] M. Ferretti, M Boffadossi, “A Parallel Pipelined Implementation of LOCO-I for JPEG-LS”. 17th International Conference on Pattern Recognition (ICPR'04) – Vol.1 pág. 769-772.

[33] A. Savakis, M. Piorun, “Benchmarking and hardware implementation of JPEGLS”.Disponible:www.ce.rit.edu/~savakis/papers/ICIP02_savakis_piorun.pdf

[34] ISO/IEC JTC1/SC 29/WG 1, “FCD 14495, Lossless and near-lossless coding of continuous tone still images (JPEG-LS), 1997.

[35] M. Klimesh, V. Stanton, D. Watola,"Hardware Implementation of a Lossless Image Compression Algorithm Using a Field Programmable Gate Array", Febrero 2001.

[36] B. Schoner, J. Villasenor, S. Molloy, R. Jain, "Techniques for FPGA Implemetation of Video Compression Systems".

[37] S. Wakasugi, "Hardware Design of a Lossless Image Compressor".

[38] C. Topal, O. Gerek, "PDF Sharpening for Multichannel Predictive Coders".

HERRAMIENTAS DE DISEÑO

[39] M. Torres, M. A. Torres, "Diseño e ingeniería electrónica asistida con Protel DXP", Alfaomega, 2005.

[40] Altera, "Introduction to Quartus II", versión 5.0, Abril 2005. Disponible : http://www.altera.com

[41] Altera, "Quartus II Handbook", versión 5.0, 2005. Disponible: http://www.altera.com

[42] Altium, "Introducing Protel DXP", 2003.

[43] Altium, "Protel DXP Training Manual", 2003.

[44] Altium, "Getting Started with PCB Design", versión 1.4 Noviembre 2005.

CODIGOS DE GOLOMB

[45] J. Liang, S. Fraser, "ENSC 424 – Multimedia Communications Engineering Topic 5: Golomb Code.", Setiembre 2005.

[46] J. Meany, "Golomb Coding Notes", Octubre 2005. Disponible: http://decsai.ugr.es/ccd/complementario/teoria/Golomb_Notes.pdf

SOFTWARE JPEG-LS

[47] Ismail R. Ismail, Faouzi Kossentini,Implementación en C del estándar JPEG-LS, 9/30/97, distribución libre.

9 ANEXOS

Anexo 1: Códigos de Golomb

Anexo 2: Jerarquías de archivos VHDL

Anexo 3: Layout de la Placa IIE-Cyclone

Anexo 4: Esquemáticos de la Placa IIE-Cyclone

Anexo 5: JPEG-LS Software

Anexo 6: Pinout FPGA

Anexo 1: Codificación Golomb

¿Cómo represento un número utilizando códigos de Golomb?

Dados:

n: número a representar.
m: entero que representa el orden del código.

La palabra código esta formada por dos partes distintas, un prefijo que es la representación no-aria de n DIV m y un sufijo que es la representación binaria de n MOD m.

Palabra código (n)= repr. no-aria (n DIV m) + repr. binaria (n MOD m)

Los códigos no-arios representan un entero positivo mediante la utilización de ceros y unos, concatenando los ceros, hasta que la cantidad de los mismos sea igual al número a representar, luego se determina el fin de la representación colocando un uno. En la siguiente tabla se muestran las palabras código para los enteros del cero al 9:

número	código
0	1
1	01
2	001
3	0001
4	00001
5	000001
6	0000001
7	00000001
8	000000001
9	0000000001

Los códigos no-arios son óptimos cuando el universo de posibles eventos a representar tienen probabilidades del tipo: 1/2, 1/4, 1/8, 1/16, 1/32....., ya que la mayor cantidad de palabras a ser codificadas van a ser representadas por una palabra código corta. Queda claro de la tabla que los mismos son de largo variable.
Con respecto al sufijo de la representación, se genera un tipo especial de código cuando el parámetro $m = 2^k$, en estos casos el sufijo se convierte en los k menos significativos bits de n. Este sub-tipo de código es conocido como códigos de Golomb-Rice.
El la siguiente tabla se muestran las palabras código para los enteros del cero al 15, utilizando un código Golomb-Rice con k = 3.

número	código
0	1000
1	1001
2	1010
3	1011
4	1100
5	1101
6	1110
7	1111

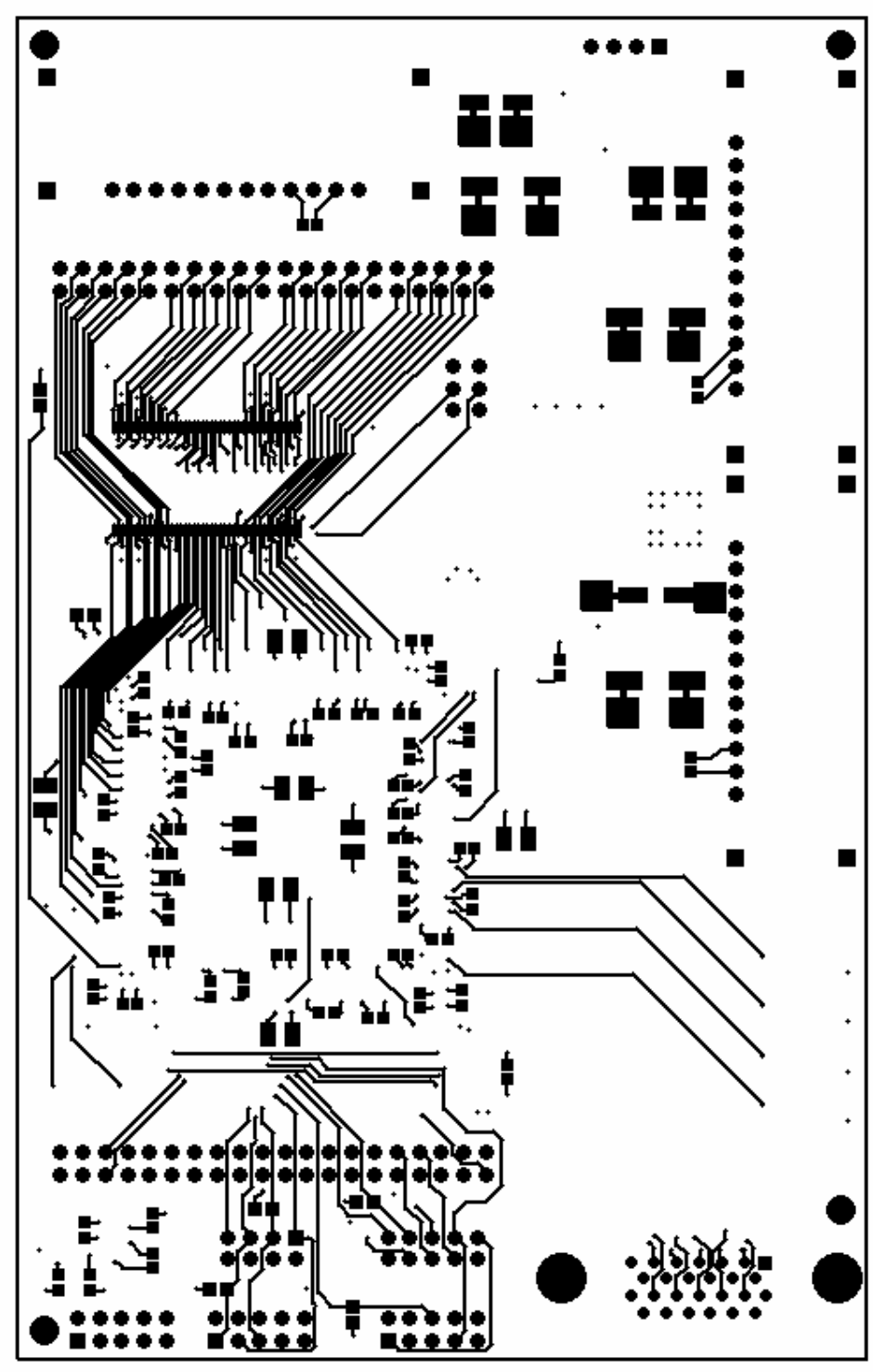
número	código
8	01000
9	01001
10	01010
11	01011
12	01100
13	01101
14	01110
15	01111

Anexo 2: Jerarquías de archivos VHDL

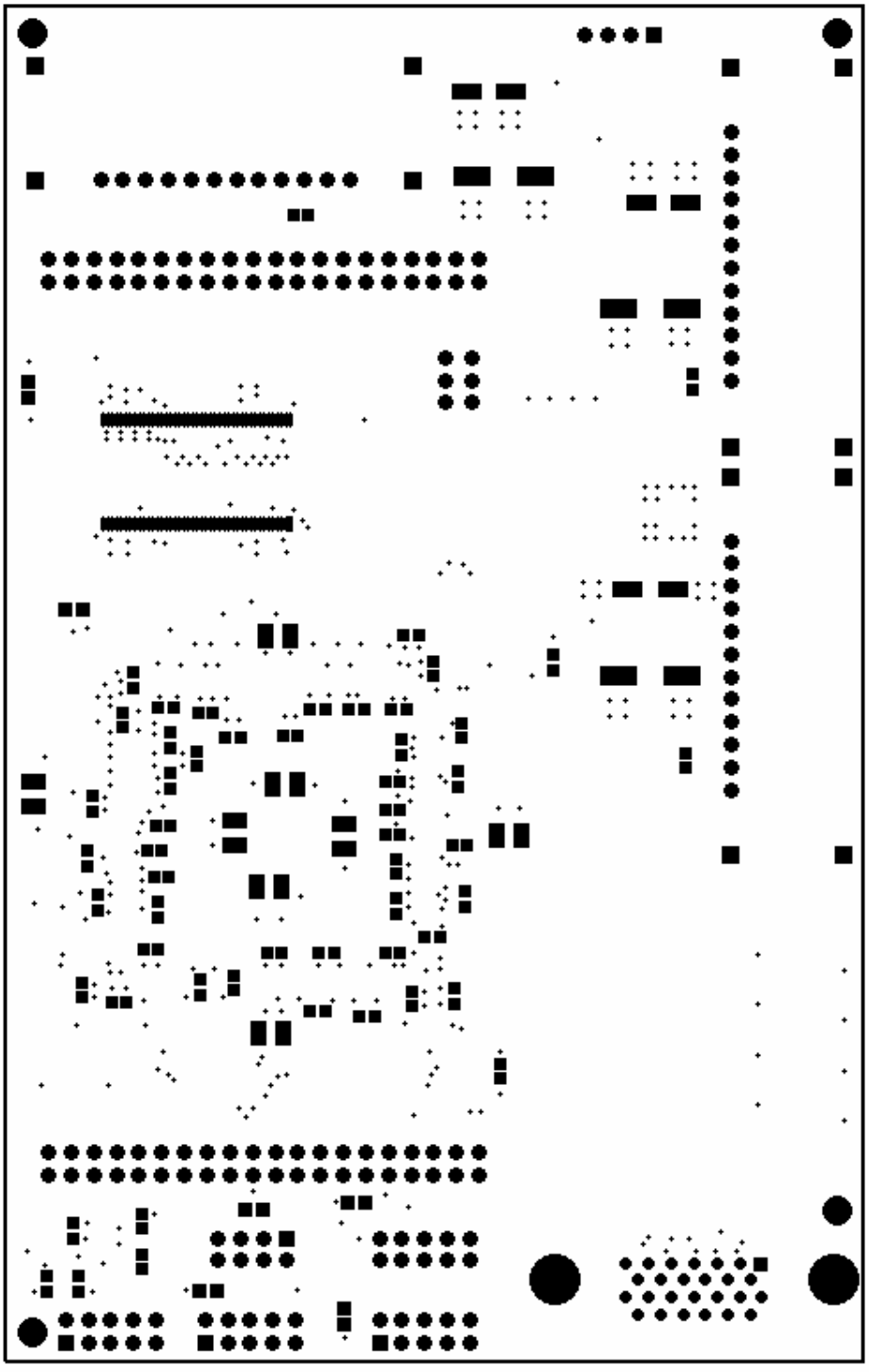
- UnaPasadaLoco.vhd
  - Etapa_1.vhd
    - Gradients.vhd
      - Trunc.vhd
        - Resta.vhd
    - Cuantizador.vhd
      - Quant_7.vhd
    - Context_word.vhd
  - Etapa_2.vhd
    - Predictor.vhd
      - SumaPredictor.vhd
    - Map_error.vhd
      - Mapeo.vhd
        - Multiplicador.vhd
        - v_abs.vhd
      - ModRange.vhd
      - Trunc_r.vhd
        - Resta.vhd
    - GolombCode.vhd
      - calculo_k.vhd
        - Multiplexor.vhd
        - PorDos.vhd
      - Contador.vhd
      - err_div_k.vhd
    - context_ram.vhd
    - AdaptativeCorrection.vhd
      - Negador.vhd
    - lectura_ram.vhd
      - divido_signed.vhd
        - división_signed.vhd
      - Suma8.vhd
      - Suma13.vhd
      - v_abs.vhd

Anexo 3: Layout de la Placa IIE-Cyclone

Placa IIE-Cyclone Bottom Layer

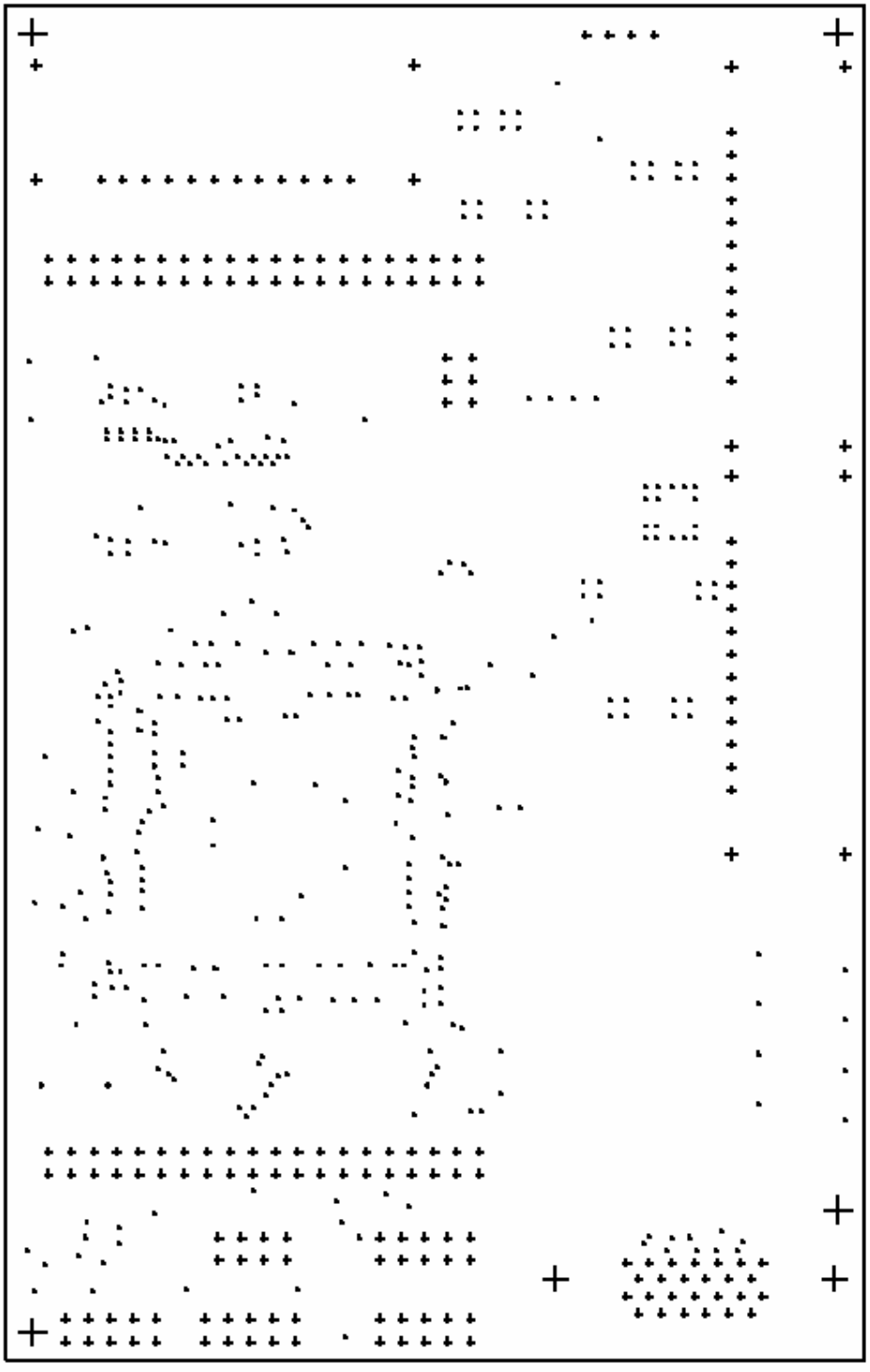


Placa IIE-Cyclone Bottom Soldermask

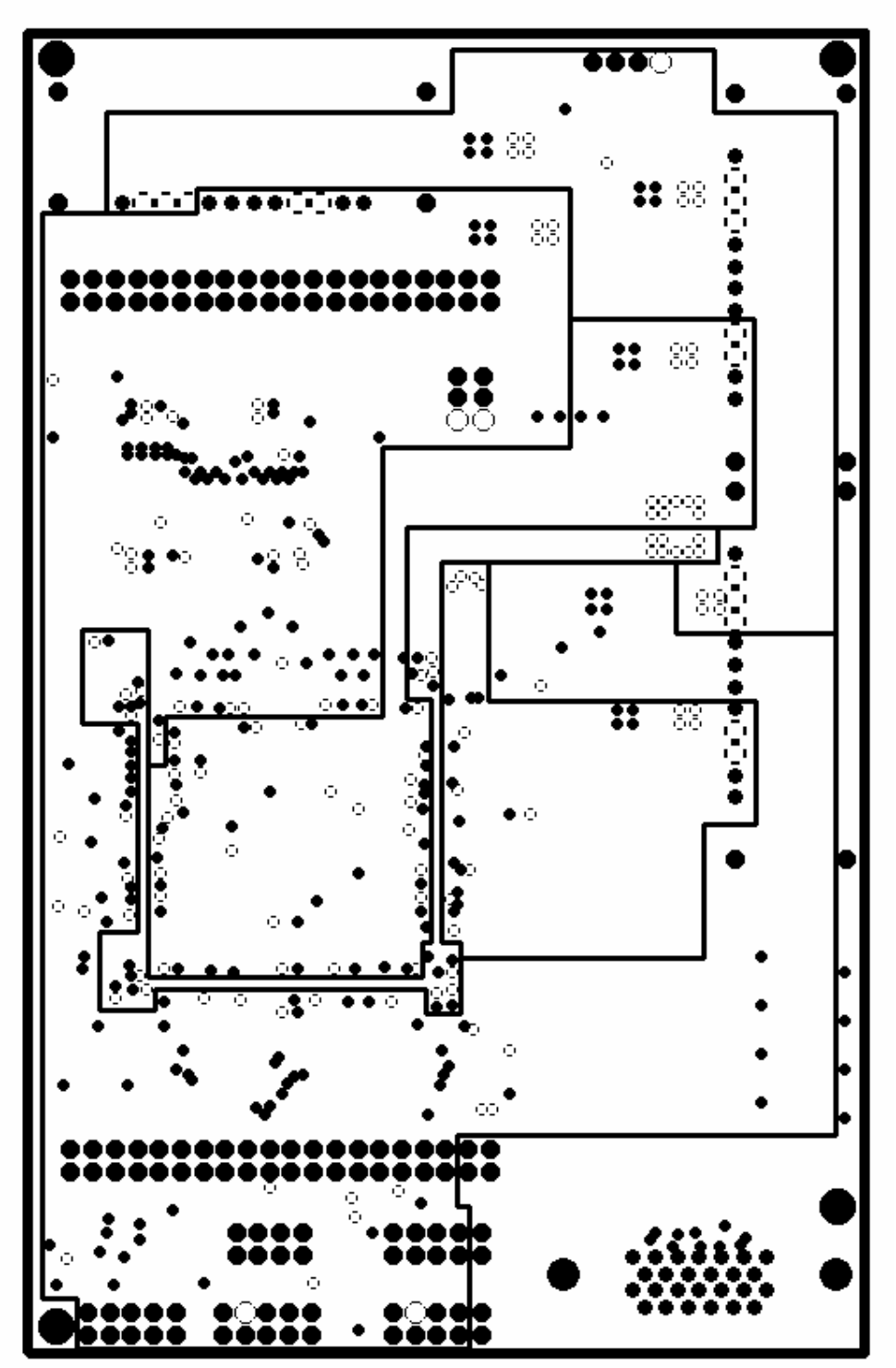


Placa IIE-Cyclone Drill Size

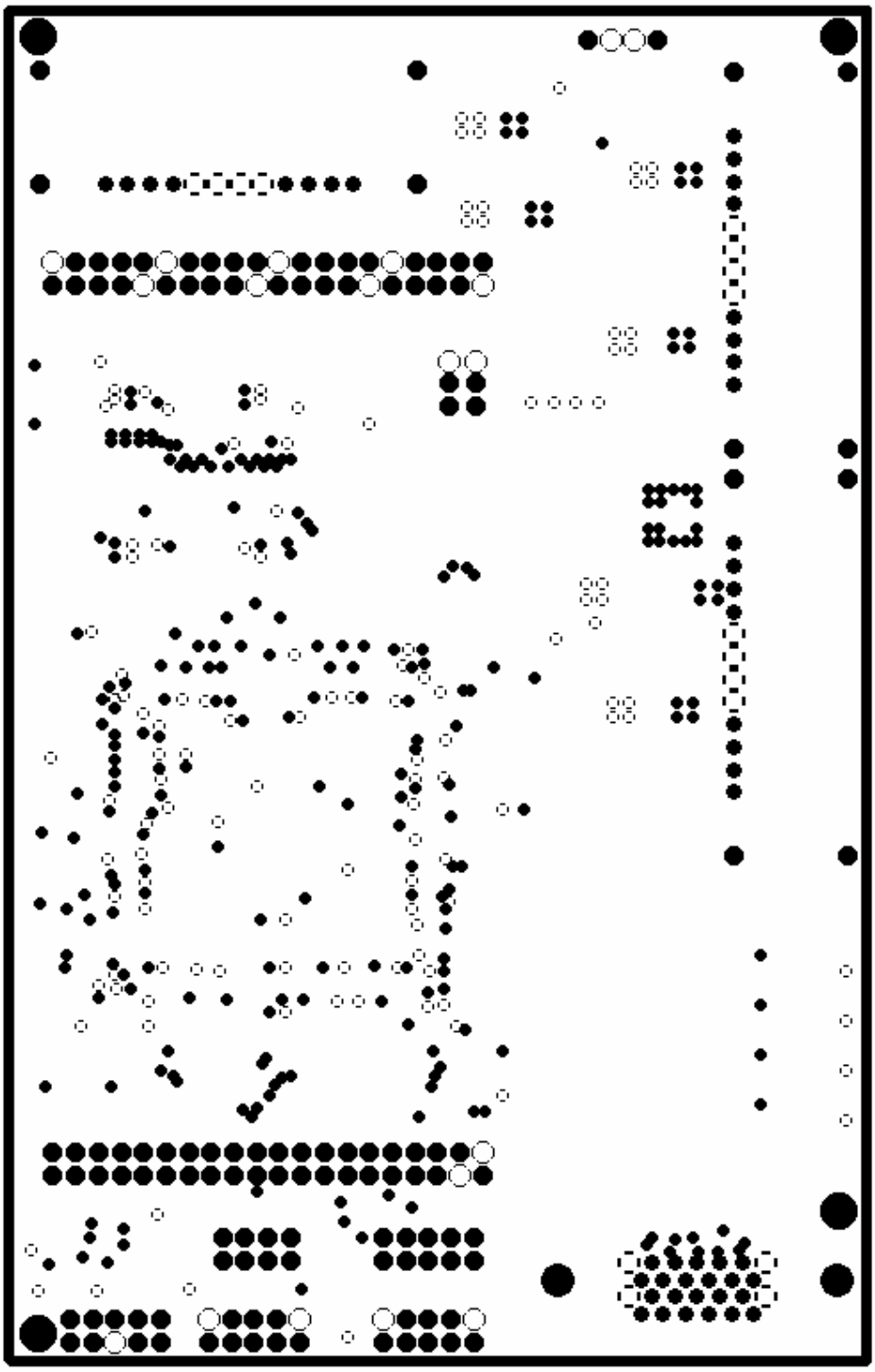
Placa IIE-Cyclone Drill Guide



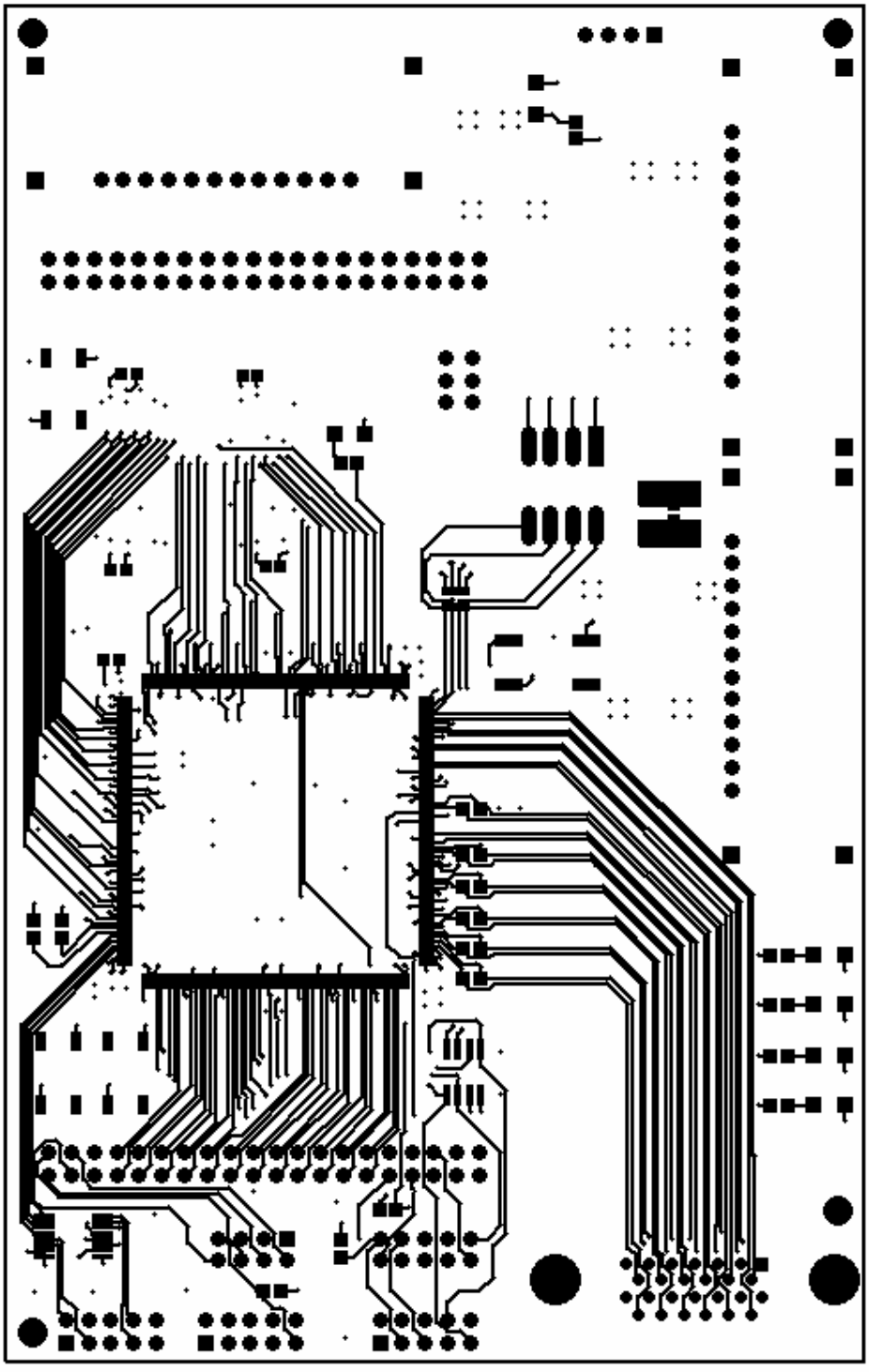
Placa IIE-Cyclone Power Plane



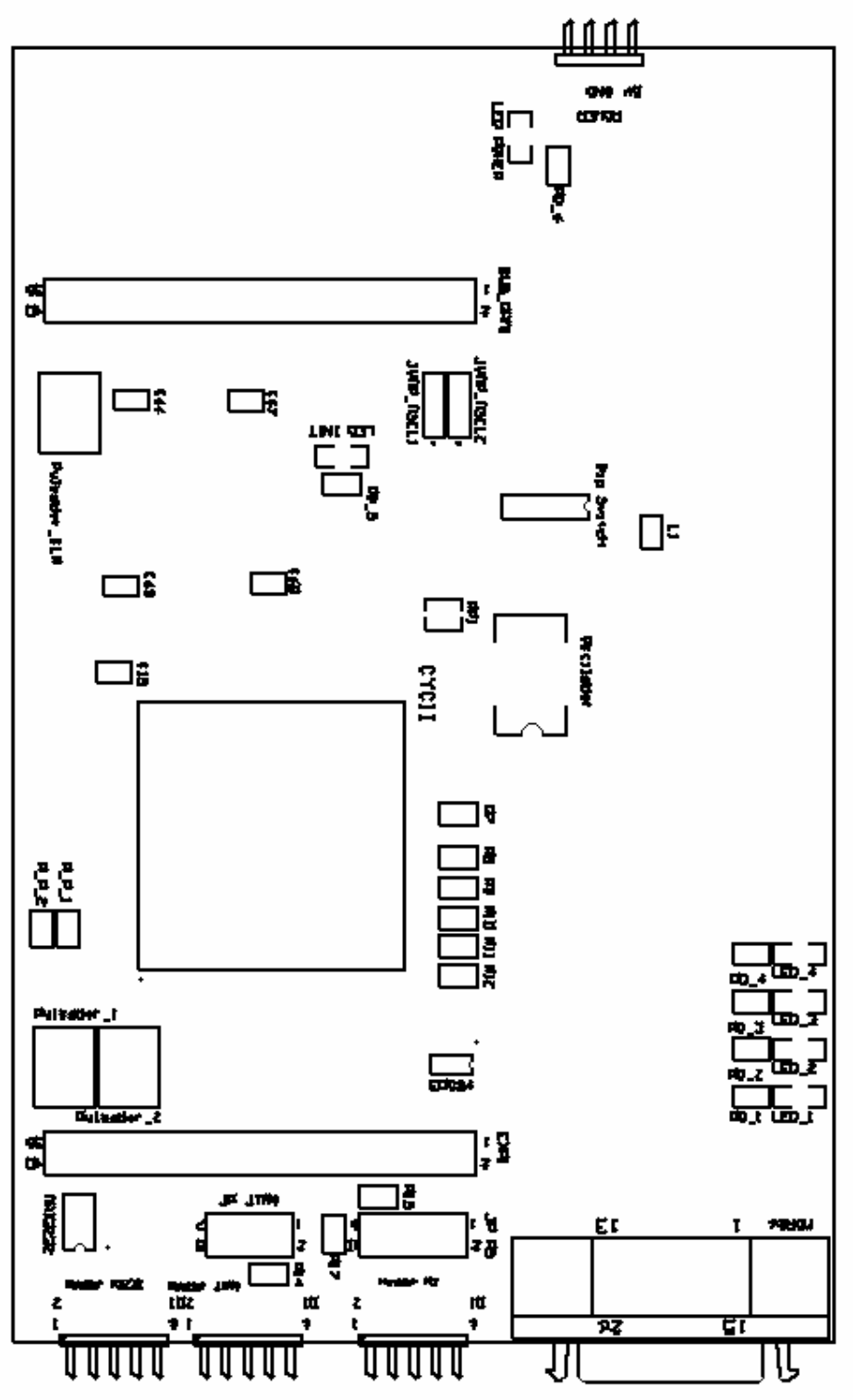
Placa IIE-Cyclone Ground Plane



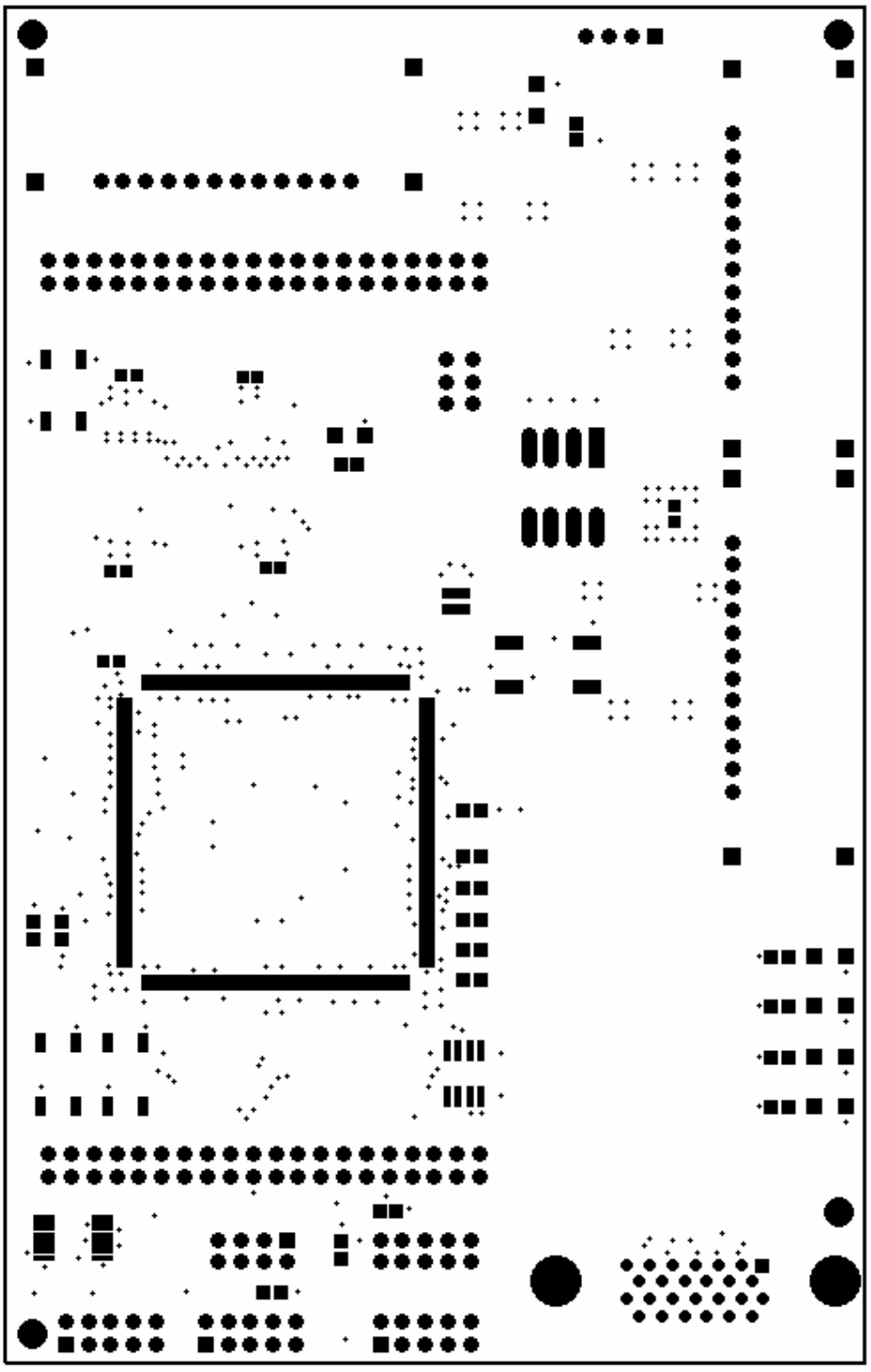
Placa IIE-Cyclone Top Layer



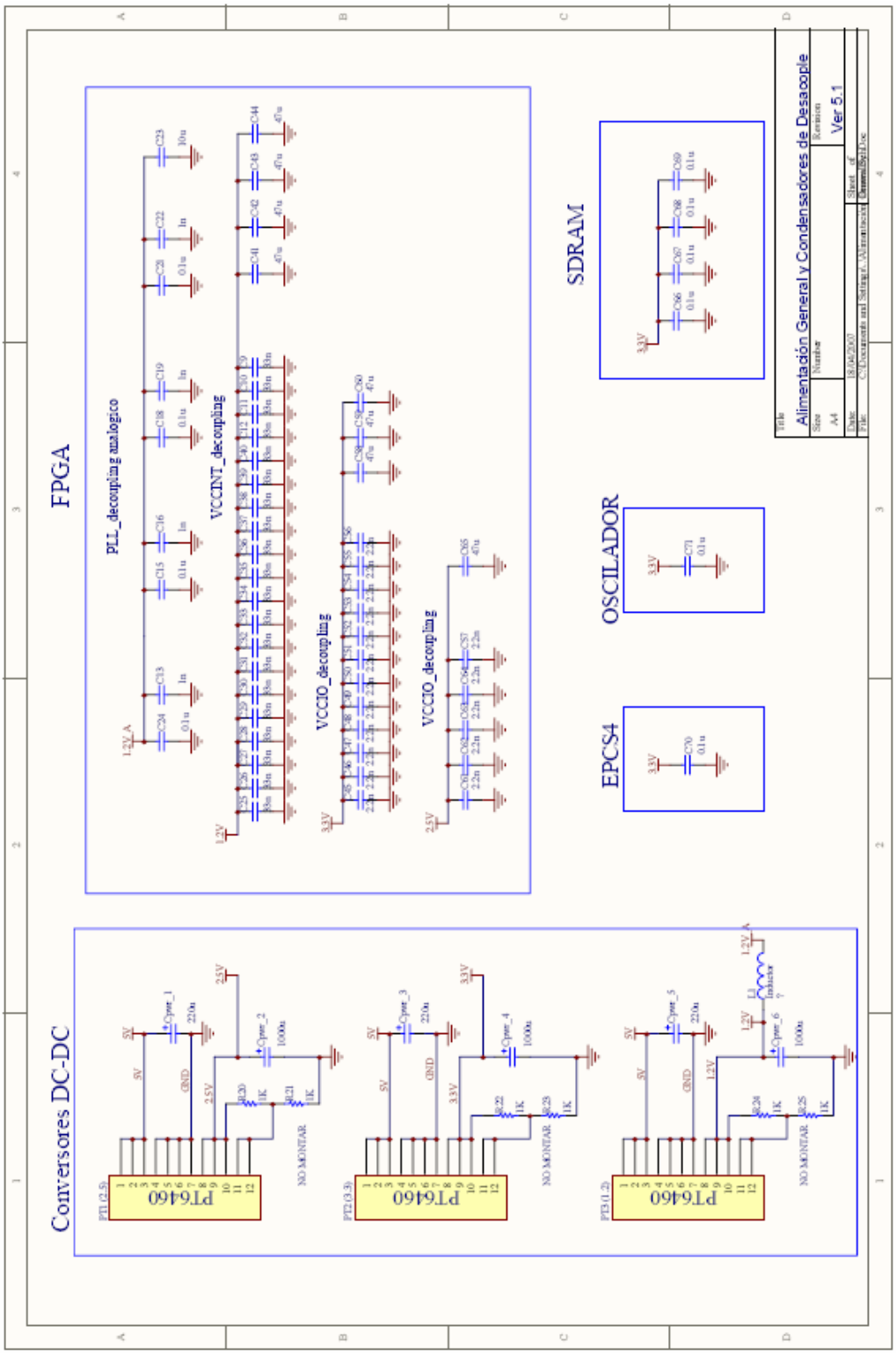
Placa IIE-Cyclone Top Overlay

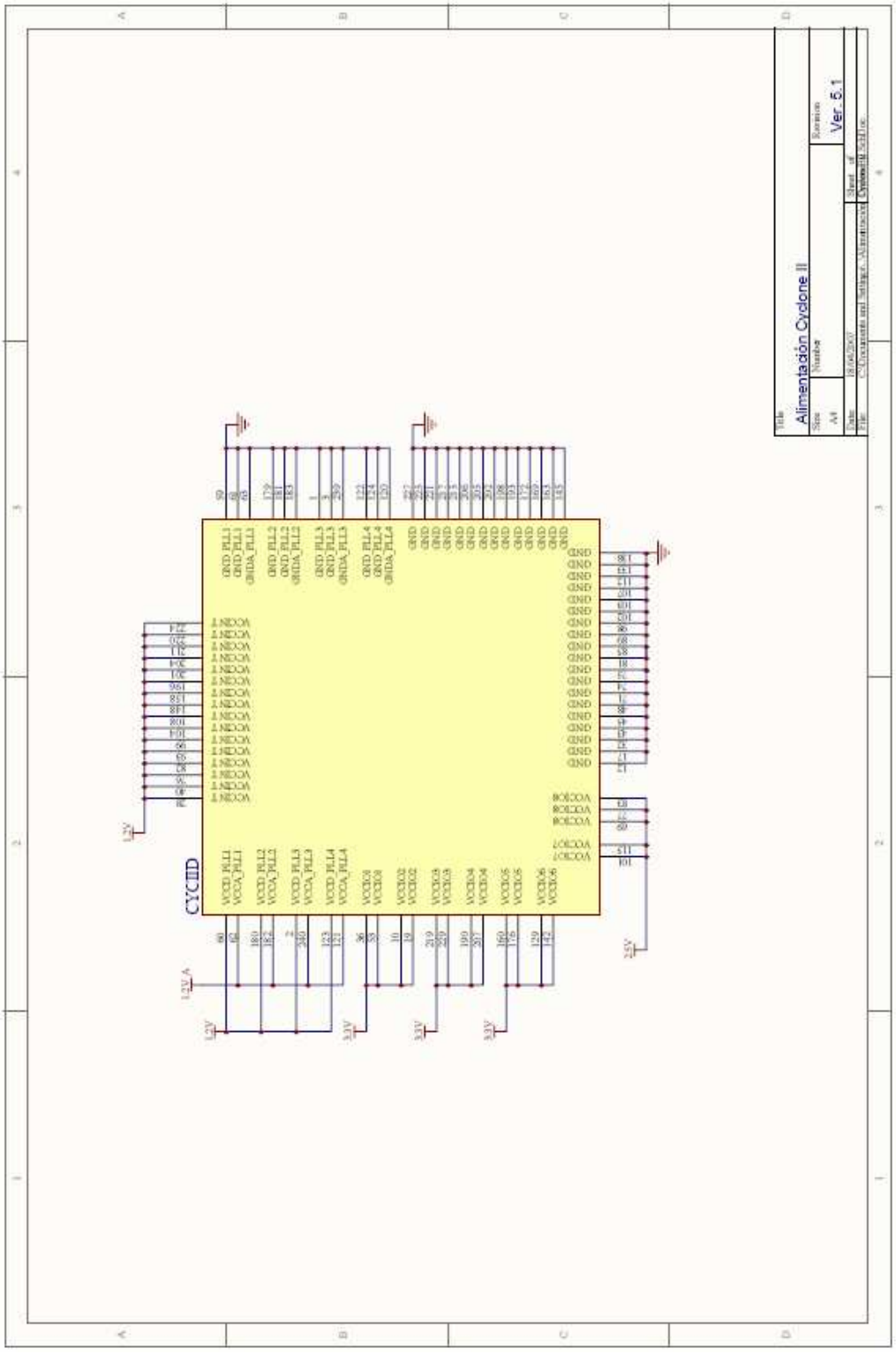


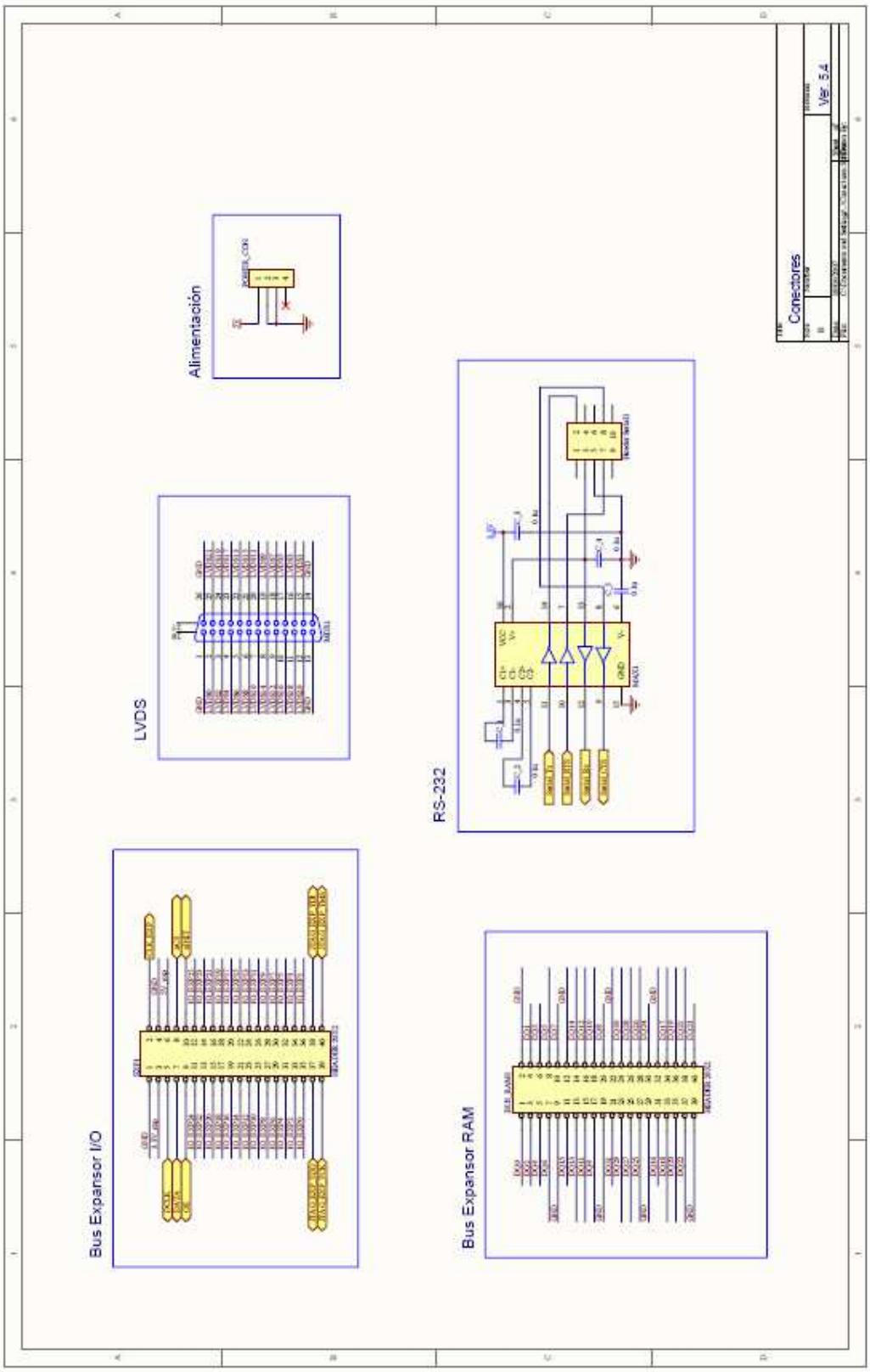
Placa IIE-Cyclone Top Soldermask

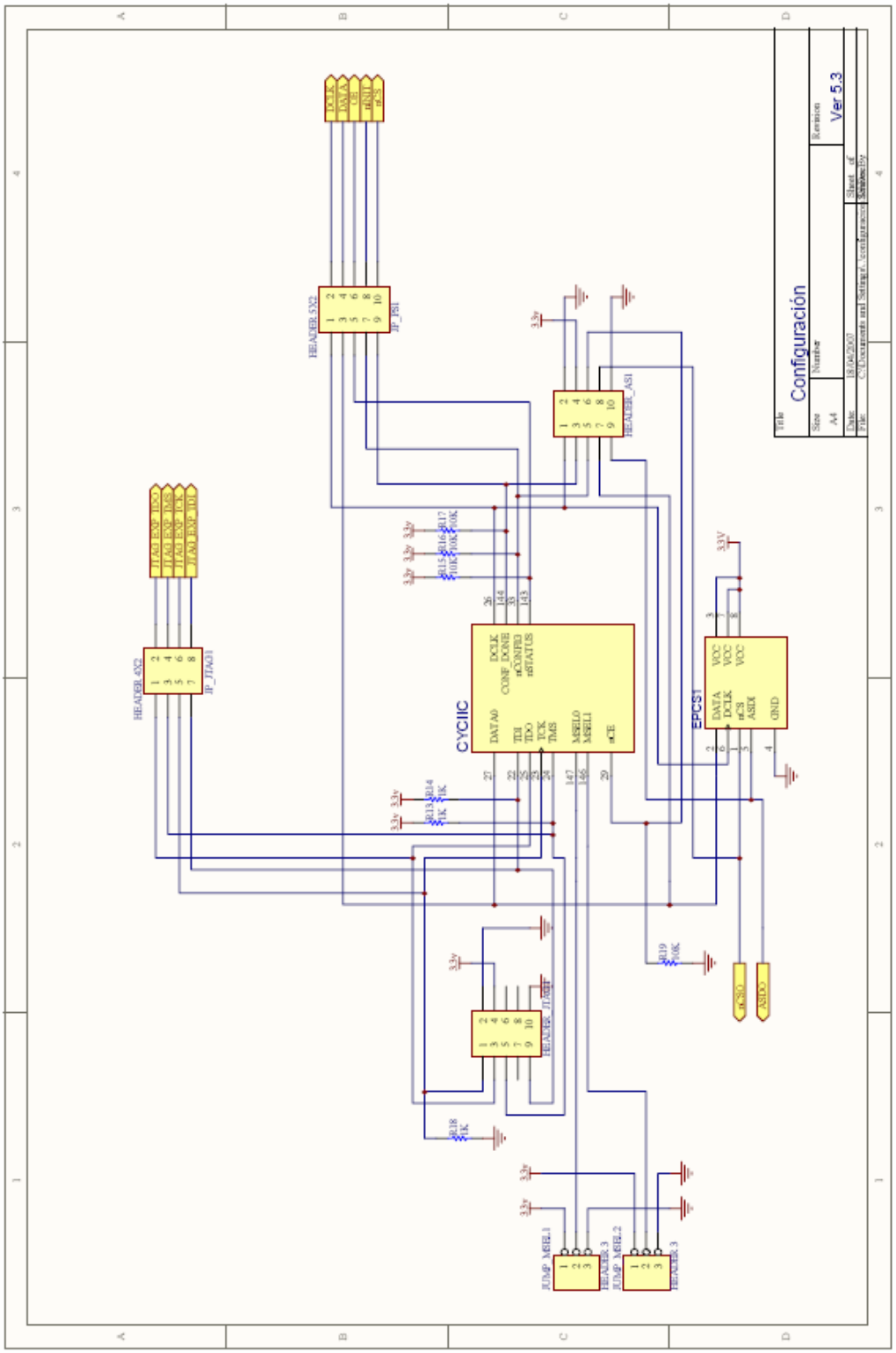


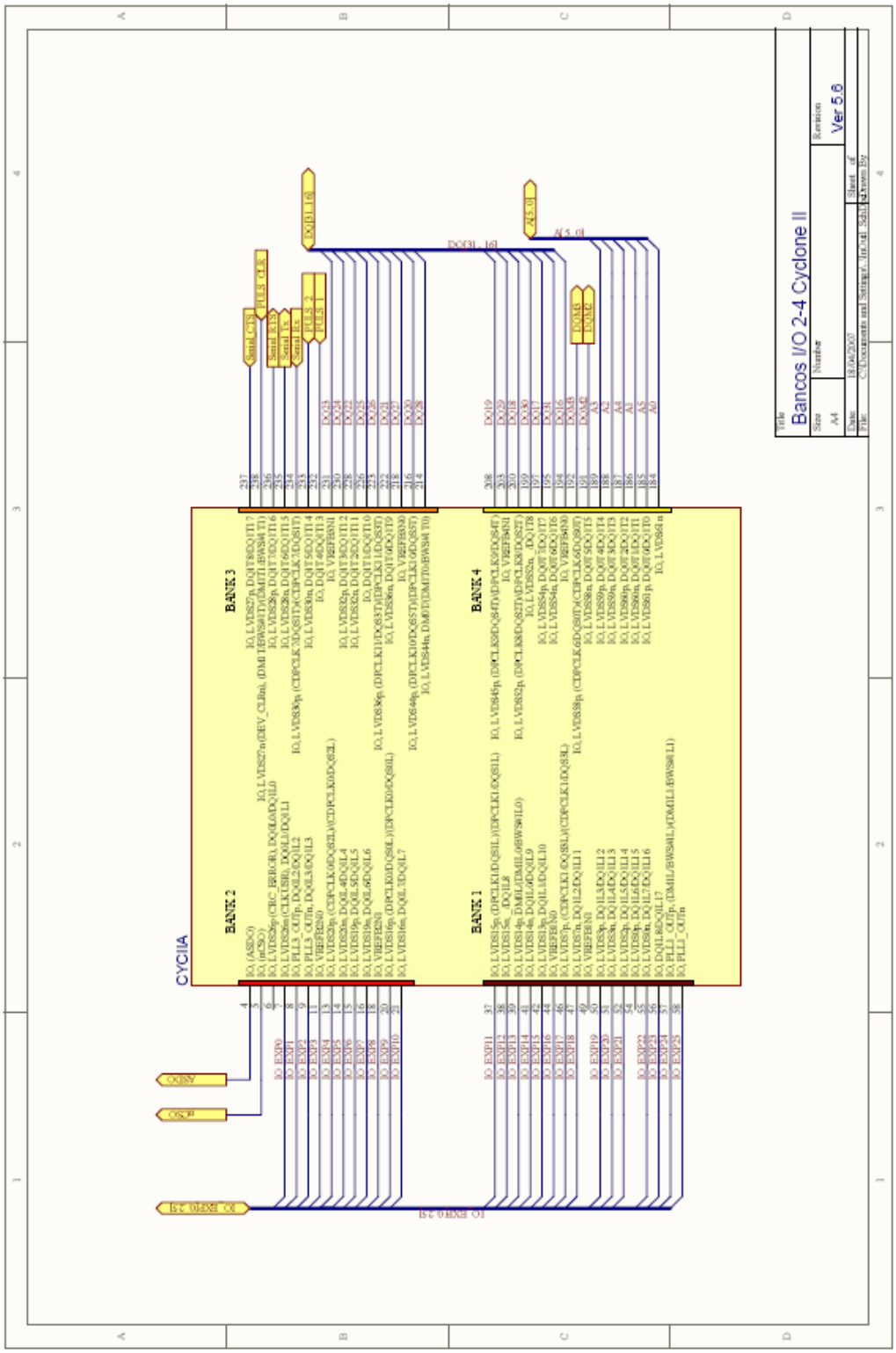
Anexo 4: Esquemáticos Placa IIE-Cyclone

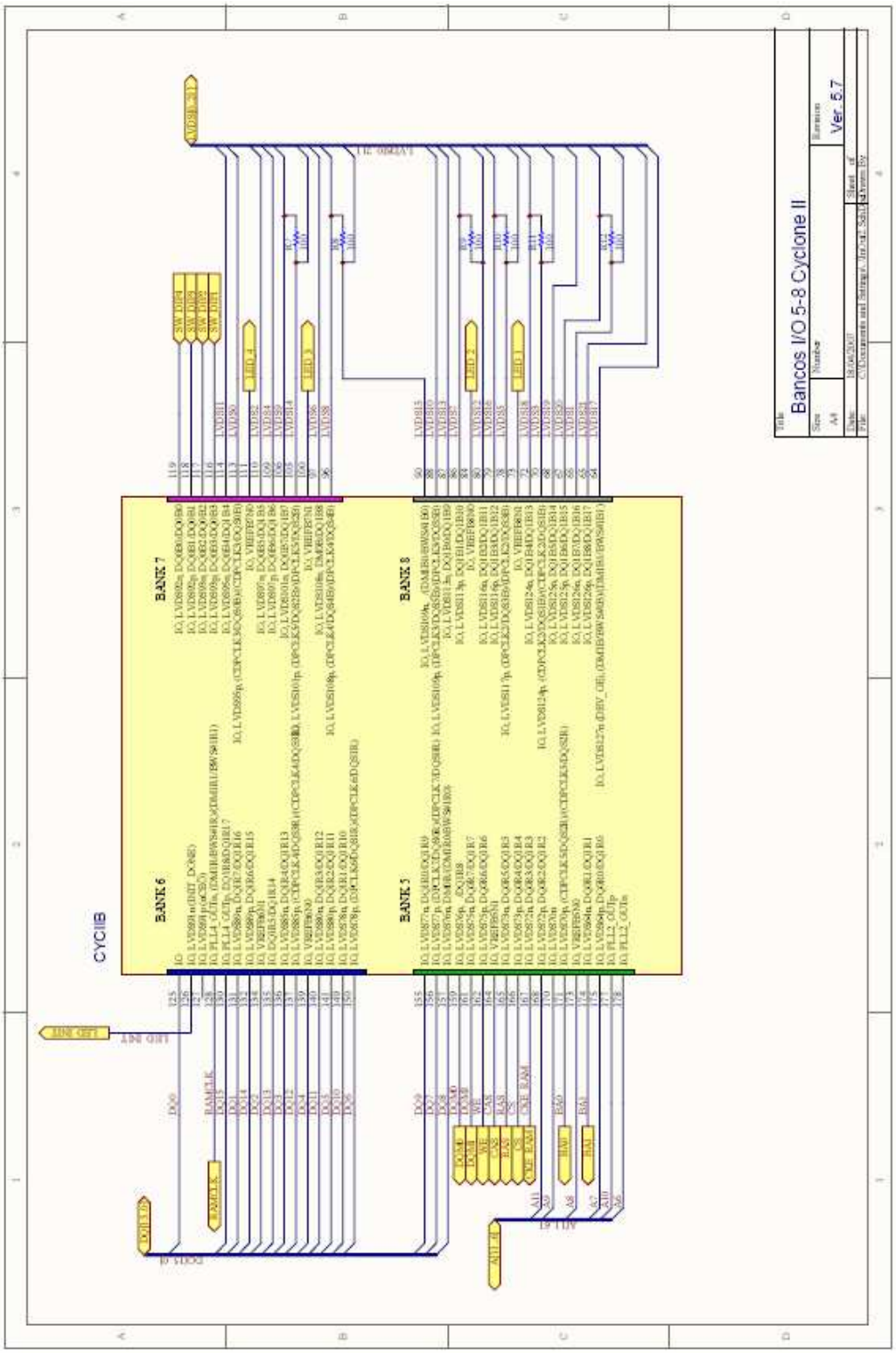

FPGA
PLL_decoupling analogico
VCCINT_decoupling
VCCIO_decoupling
VCCIO_decoupling
SDRAM
OSCILADOR
EPCS4
Conversores DC-DC
PT6460
PT6460
PT6460
Alimentación General y Condensadores de Desacople
Ver 5.1

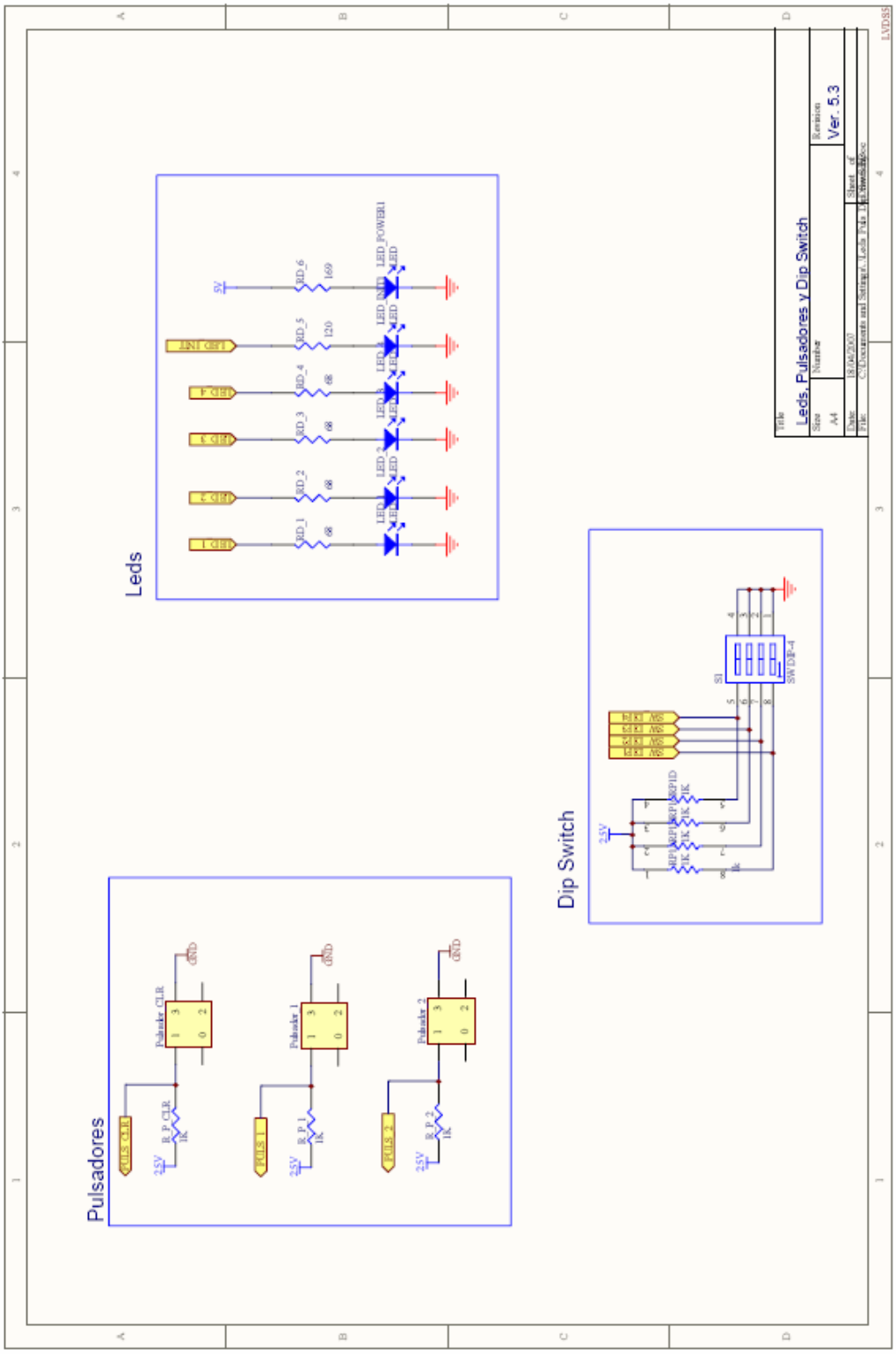

Alimentación Cyclone II
Ver. 5.1
CYCIID

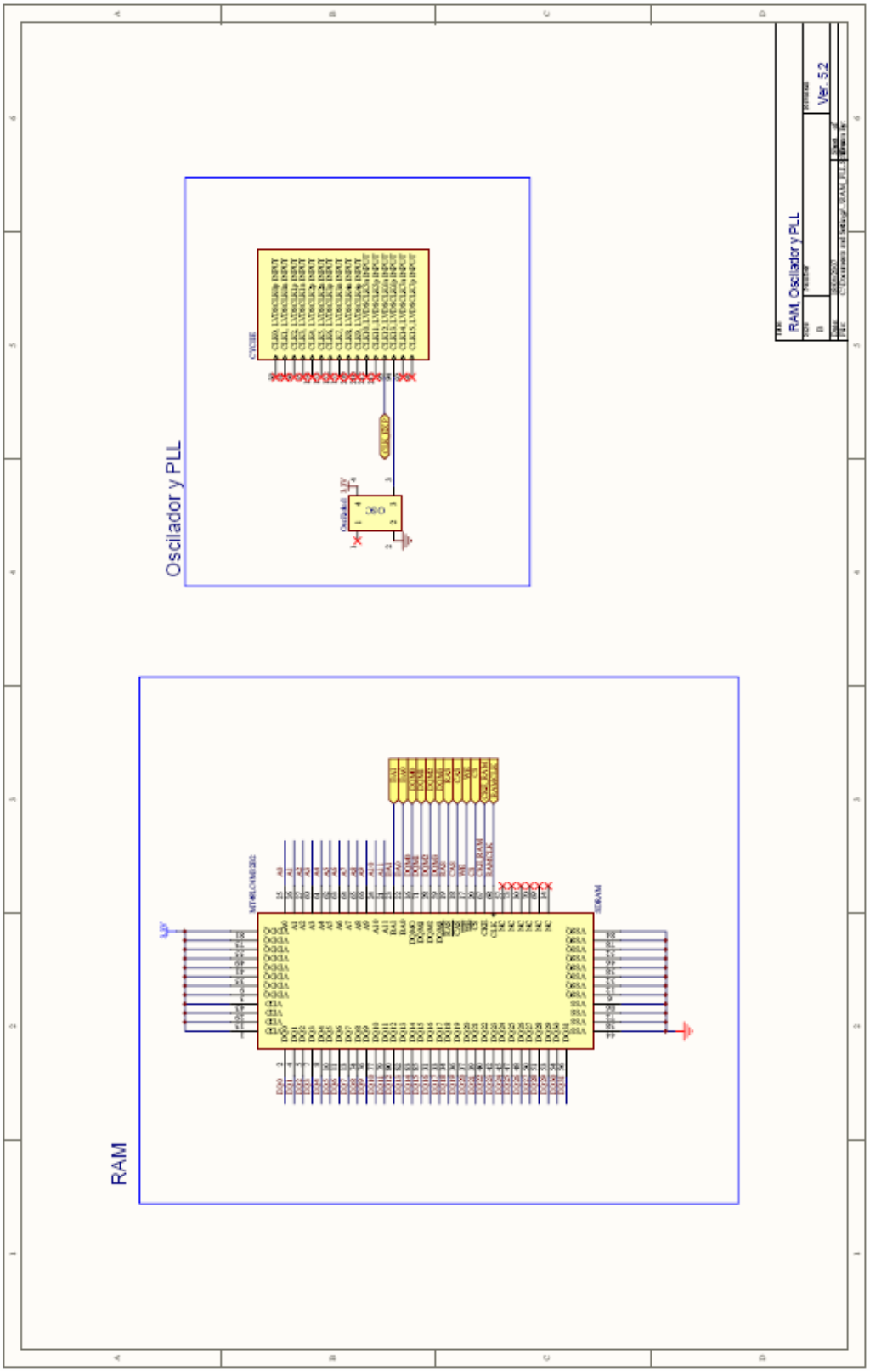

Alimentación
LVDS
RS-232
Bus Expansor I/O
Bus Expansor RAM
Conectores
Ver 5.4


Configuración
Ver 5.3
CYCIIC
EPCS1
HEADER 5X2
JP_JS1
HEADER_AS1
HEADER_4X2
JP_JTAG1
HEADER_JTAG1
JUMP_MSEL1
JUMP_MSEL2
HEADER_3
DCLK
DATA0
nCE
nCONFIG
nSTATUS
CONF_DONE
TDI
TDO
TCK
TMS
MSEL0
MSEL1
DATA
ASDI
nCS
GND
VCC
3.3v


Bancos I/O 2-4 Cyclone III
Ver 5.6
CYCIIA
BANK 1
BANK 2
BANK 3
BANK 4


CYCIIB
BANK 6
BANK 7
BANK 5
BANK 8
Bancos I/O 5-8 Cyclone II
Ver. 5.7


Leds
Dip Switch
Pulsadores
Leds, Pulsadores y Dip Switch
Ver. 5.3


RAM, Oscilador y PLL
Ver 5.2
Oscilador y PLL
RAM

Anexo 5: JPEG-LS Software

El software utilizado fue diseñado por Ismaeil R. Ismaeil y Faouzi Kossintini del Departamento de Ingeniería Eléctrica y Computación de la Universidad de British Columbia para comprimir y descomprimir imágenes bajo el estándar JPEG-LS. La versión utilizada fue la 1.1, fue creada en lenguaje C y el compilador usado es el *gcc* de Linux.

Este software fue utilizado como patrón para comparar los resultados obtenidos de nuestra implementación, para lograr este objetivo fue necesario realizarle algunas modificaciones.

El Codificador está compuesto de los siguientes archivos:

- globals.h
- marker.h
- prototypes.h
- read_write.h
- encoder.c
- general.c
- marker.c
- regular_mode.c
- run_mode.c
- makefile

Para compilar el codificador se debe ejecutar el siguiente comando:

#make jpeglse

El Decodificador está compuesto por los siguientes archivos:

- globals.h
- marker.h
- prototypes.h
- read_write.h
- decoder.c
- general.c
- marker.c
- regular_mode.c
- run_mode.c
- makefile

Para compilar el decodificador se debe ejecutar el siguiente comando:

#make jpeglsd

Para codificar se debe ejecutar el siguiente comando:

 # jpeglse imagen_entrada imagen_salida filas columnas bpp

Ej.: # jpeglse prueba.pgm prueba.jls 100 100 8
En este ejemplo se comprime la imagen prueba.pgm de 100 píxeles por 100 píxeles con 8 bit/píxel de resolución, obteniendo el archivo prueba.jls con la imagen comprimida.

Para decodificar se debe ejecutar el siguiente comando:

 # jpeglsd imagen_comprimida imagen_reconstruida

Modificaciones realizadas en el código:

Se codificó solo utilizando el modo Regular Mode y se cambiaron las regiones y niveles de cuantización. Los cambios se muestran en **negrita**.

En encoder.c :

/* set the encoding parameters based on precision value */

```
  if(P<=8){
        T1 = 2;
        T2 = 5;
        T3 = 13;
     }


// if (D[0] == 0 && D[1] == 0 && D[2] == 0)
//     RunModeProcessing();
//   else
     RegularModeProcessing();
//   }
```

En regular_mode.c :

```
/* Quantization of the gradients */
   for(i=0; i<3; i++){
        if(D[i] <= -T3) Q[i] = -3;
        else if(D[i] <= -T2) Q[i] = -2;
        else if(D[i] <= -T1) Q[i] = -1;
        else if(D[i] < T1) Q[i] = 0;
        else if(D[i] < T2) Q[i] = 1;
        else if(D[i] < T3) Q[i] = 2;
        else Q[i] = 3;
    }
```

Anexo 6: Pinout FPGA

PIN Cyclone	Nombre	Función
1	GND = 0V	Alimentación
2	VCC = 1.2V	Alimentación
3	GND = 0V	Alimentación
4	ASDO	Configuración
5	nCSO	Configuración
6	N/C	
7	IO_EXP0	I/O
8	IO_EXP1	I/O
9	IO_EXP2	I/O
10	VCCIO_LVTTL/LVCMOS = 3.3V	Alimentación
11	IO_EXP3	I/O
12	GND = 0V	Alimentación
13	IO_EXP4	I/O
14	IO_EXP5	I/O
15	IO_EXP6	I/O
16	IO_EXP7	I/O
17	GND = 0V	Alimentación
18	IO_EXP8	I/O
19	VCCIO_LVTTL/LVCMOS = 3.3V	Alimentación
20	IO_EXP9	I/O
21	IO_EXP10	I/O
22	TDI	Configuración
23	TCK	Configuración
24	TMS	Configuración
25	TDO	Configuración
26	DCLK	Configuración
27	DATA0	Configuración
28	VCC = 1.2V	Alimentación
29	nCE	Configuración
30	N/C	
31	N/C	
32	GND = 0V	Alimentación
33	nCONFIG	Configuración
34	N/C	
35	N/C	
36	VCCIO_LVTTL/LVCMOS = 3.3V	Alimentación
37	IO_EXP11	I/O
38	IO_EXP12	I/O
39	IO_EXP13	I/O
40	VCC = 1.2V	Alimentación
41	IO_EXP14	I/O
42	IO_EXP15	I/O
43	GND = 0V	Alimentación
44	IO_EXP16	I/O
45	GND = 0V	Alimentación
46	IO_EXP17	I/O
47	IO_EXP18	I/O
48	GND = 0V	Alimentación
49	N/C	

PIN Cyclone	Nombre	Función
50	IO_EXP19	I/O
51	IO_EXP20	I/O
52	IO_EXP21	I/O
53	VCCIO_LVTTL/LVCMOS = 3.3V	Alimentación
54	N/C	
55	IO_EXP22	I/O
56	IO_EXP23	I/O
57	IO_EXP24	I/O
58	IO_EXP25	I/O
59	GND = 0V	Alimentación
60	VCC = 1.2V	Alimentación
61	GND = 0V	Alimentación
62	VCC = 1.2V_Analógico	Alimentación
63	GND = 0V	Alimentación
64	LED_1	I/O
65	X0+	LVDS
66	X0-	LVDS
67	X1+	LVDS
68	X1-	LVDS
69	VCCIO_LVTTL/LVCMOS = 2.5V	Alimentación
70	X2+	LVDS
71	GND = 0V	Alimentación
72	X2-	LVDS
73	LED_2	I/O
74	GND = 0V	Alimentación
75	GND = 0V	Alimentación
76	VCC = 1.2V	Alimentación
77	VCCIO_LVTTL/LVCMOS = 2.5V	Alimentación
78	LED_3	I/O
79	Xclk+	LVDS
80	Xclk-	LVDS
81	GND = 0V	Alimentación
82	VCC = 1.2V	Alimentación
83	VCCIO_LVTTL/LVCMOS = 2.5V	Alimentación
84	LED_4	I/O
85	GND = 0V	Alimentación
86	X3+	LVDS
87	X3-	LVDS
88	SerTC+	LVDS
89	GND = 0V	Alimentación
90	SerTC-	LVDS
91	N/C	
92	N/C	
93	VCC = 1.2V	Alimentación
94	Oscilador	Reloj
95	CLK_EXP	Reloj
96	SerTFG+	LVDS
97	SerTFG-	LVDS
98	GND = 0V	Alimentación
99	VCC = 1.2V	Alimentación

PIN Cyclone	Nombre	Función
100	SW_DIP1	I/O
101	VCCIO_LVTTL/LVCMOS = 2.5V	Alimentación
102	GND = 0V	Alimentación
103	GND = 0V	Alimentación
104	VCC = 1.2V	Alimentación
105	CC1+	LVDS
106	CC1-	LVDS
107	GND = 0V	Alimentación
108	VCC = 1.2V	Alimentación
109	CC2+	LVDS
110	CC2-	LVDS
111	SW_DIP2	I/O
112	GND = 0V	Alimentación
113	CC3+	LVDS
114	CC3-	LVDS
115	VCCIO_LVTTL/LVCMOS = 2.5V	Alimentación
116	CC4+	LVDS
117	CC4-	LVDS
118	SW_DIP3	I/O
119	SW_DIP4	I/O
120	GND = 0V	Alimentación
121	VCC = 1.2V_Analógico	Alimentación
122	GND = 0V	Alimentación
123	VCC = 1.2V	Alimentación
124	GND = 0V	Alimentación
125	DQ0	RAM
126	LED_INIT	I/O
127	N/C	
128	RAMCLK	RAM
129	VCCIO_LVTTL/LVCMOS = 3.3V	Alimentación
130	DQ15	RAM
131	DQ1	RAM
132	DQ14	RAM
133	GND = 0V	Alimentación
134	DQ2	RAM
135	DQ13	RAM
136	DQ3	RAM
137	DQ12	RAM
138	GND = 0V	Alimentación
139	DQ4	RAM
140	DQ11	RAM
141	DQ5	RAM
142	VCCIO_LVTTL/LVCMOS = 3.3V	Alimentación
143	nSTATUS	Configuración
144	CONF_DONE	I/O
145	GND = 0V	Alimentación
146	MSEL1	Configuración
147	MSEL0	Configuración
148	VCC = 1.2V	Alimentación
149	DQ10	RAM

PIN Cyclone	Nombre	Función
150	DQ6	RAM
151	N/C	
152	N/C	
153	N/C	
154	N/C	
155	DQ9	RAM
156	DQ7	RAM
157	DQ8	RAM
158	VCC = 1.2V	Alimentación
159	DQM0	RAM
160	VCCIO_LVTTL/LVCMOS = 3.3V	Alimentación
161	DQM1	RAM
162	WE	RAM
163	GND = 0V	Alimentación
164	CAS	RAM
165	RAS	RAM
166	CS	RAM
167	CKE_RAM	RAM
168	A11	RAM
169	GND = 0V	Alimentación
170	A9	RAM
171	BA0	RAM
172	GND = 0V	Alimentación
173	A8	RAM
174	BA1	RAM
175	A7	RAM
176	VCCIO_LVTTL/LVCMOS = 3.3V	Alimentación
177	A10	RAM
178	A6	RAM
179	GND = 0V	Alimentación
180	VCC = 1.2V	Alimentación
181	GND = 0V	Alimentación
182	VCC = 1.2V_Analógico	Alimentación
183	GND = 0V	Alimentación
184	A0	RAM
185	A5	RAM
186	A1	RAM
187	A4	RAM
188	A2	RAM
189	A3	RAM
190	VCCIO_LVDS = 3.3V	Alimentación
191	DQM2	RAM
192	DQM3	RAM
193	GND = 0V	Alimentación
194	DQ16	RAM
195	DQ31	RAM
196	VCC = 1.2V	Alimentación
197	DQ17	RAM
198	GND = 0V	Alimentación
199	DQ30	RAM
200	DQ18	RAM

PIN Cyclone	Nombre	Función
201	VCC = 1.2V	Alimentación
202	GND = 0V	Alimentación
203	DQ29	RAM
204	VCC = 1.2V	Alimentación
205	GND = 0V	Alimentación
206	GND = 0V	Alimentación
207	VCCIO_LVDS = 3.3V	Alimentación
208	DQ19	RAM
209	N/C	
210	N/C	
211	VCC = 1.2V	Alimentación
212	N/C	
213	N/C	
214	DQ28	RAM
215	GND = 0V	Alimentación
216	DQ20	RAM
217	GND = 0V	Alimentación
218	DQ27	RAM
219	VCCIO_LVDS = 3.3V	Alimentación
220	VCC = 1.2V	Alimentación
221	GND = 0V	Alimentación
222	DQ21	RAM
223	DQ26	RAM
224	VCC = 1.2V	Alimentación
225	GND = 0V	Alimentación
226	DQ25	RAM
227	GND = 0V	Alimentación
228	DQ22	RAM
229	VCCIO_LVDS = 3.3V	Alimentación
230	DQ24	RAM
231	DQ23	RAM
232	PULS_1	I/O
233	PULS_2	I/O
234	Serial_Rx	RS-232
235	Serial_Tx	RS-232
236	Serial_RTS	RS-232
237	Serial_CTS	RS-232
238	PULS_CLR	I/O
239	GND = 0V	Alimentación
240	VCC = 1.2V_Analógico	Alimentación

MANUAL DEL

USUARIO

PLACA IIE-CYCLONE

TABLA DE CONTENIDOS

1) ACERCA DEL MANUAL

El siguiente manual brinda la información necesaria para un correcto uso de la Placa IIE-Cyclone. Se describen sus características principales de hardware, los distintos métodos de configuración, el pinout de los principales componentes y tiene una sección con aplicaciones sencillas las que permiten realizar las primeras pruebas sobre la plataforma.

La placa IIE-Cyclone fue diseñada para ser utilizada como una plataforma de desarrollo genérica. El procesamiento se basa en un FPGA Cyclone II de Altera, y cuenta también con una memoria SDRAM, interfaces I/O para comunicarse con otros dispositivos, así como pulsadores, dip switches y LEDs.

2) CARACT. PRINCIPALES DE LA PLACA

- FPGA Cyclone II EP2C20Q240C8 de ALTERA.
- SDRAM 128 Mb MT48LC4M32B2 de MICRON.
- Dispositivo de configuración serial de 4 Mb EPCS4 de ALTERA.
- Oscilador de 25 MHZ CMX309FBC25.000MTR de CITIZEN.
- PLL: se utiliza uno de los cuatro PLL internos del FPGA.
- Conversor 5 Vdc/3.3 Vdc PT6461 de TEXAS INSTRUMENTS.
- Conversor 5 Vdc/2.5 Vdc PT6462 de TEXAS INSTRUMENTS.
- Conversor 5 Vdc/1.2 Vdc PT6466 de TEXAS INSTRUMENTS.
- 4 LEDs multipropósito, 1 LED indicador de alimentación y un LED indicador de configuración finalizada.
- 3 Pulsadores.
- DIP SWITCH 4 llaves.
- Interfaz Camera Link.
- Interfaz RS-232.
- Buses de expansión compatibles con los de la Placa IIE-PCI:
    - 26 señales I/O provenientes del FPGA.
    - 32 señales de datos de la SDRAM.
    - Señales de configuración JTAG y Passive Serial.
    - Señal de reloj.
- Posibles métodos de configuración del FPGA:
    - Cadena JTAG con la Placa IIE-PCI. (a implementar)
    - JTAG utilizando conector en la placa.
    - Passive Serial utilizando el FPGA ACEX de la Placa IIE-PCI como external host. (a implementar)
    - Active Serial utilizando el EPCS4 que configura al FPGA al alimentar la placa.

3) BREVE DESCRIPCIÓN FUNCIONAL

La Placa IIE-Cyclone es una placa pensada y fabricada para cumplir la función de plataforma de pruebas para aplicaciones que se monten sobre su FPGA. Asimismo se diseñó para que fuera compatible con la Placa IIE-PCI y de esta manera poder aumentar la capacidad de procesamiento de la misma, ya que esta cuenta con un FPGA ACEX, también de Altera, de menor cantidad de elementos lógicos.

El FPGA utilizado en la placa es un Cyclone II de Altera; el mismo dentro de sus principales características cuenta con 120 pines de entrada/salida multipropósito, 18752 elementos lógicos y cuatro PLLs internos.

La Placa IIE-Cyclone también cuenta con una memoria on-board del tipo SDRAM de 128 Mbits.

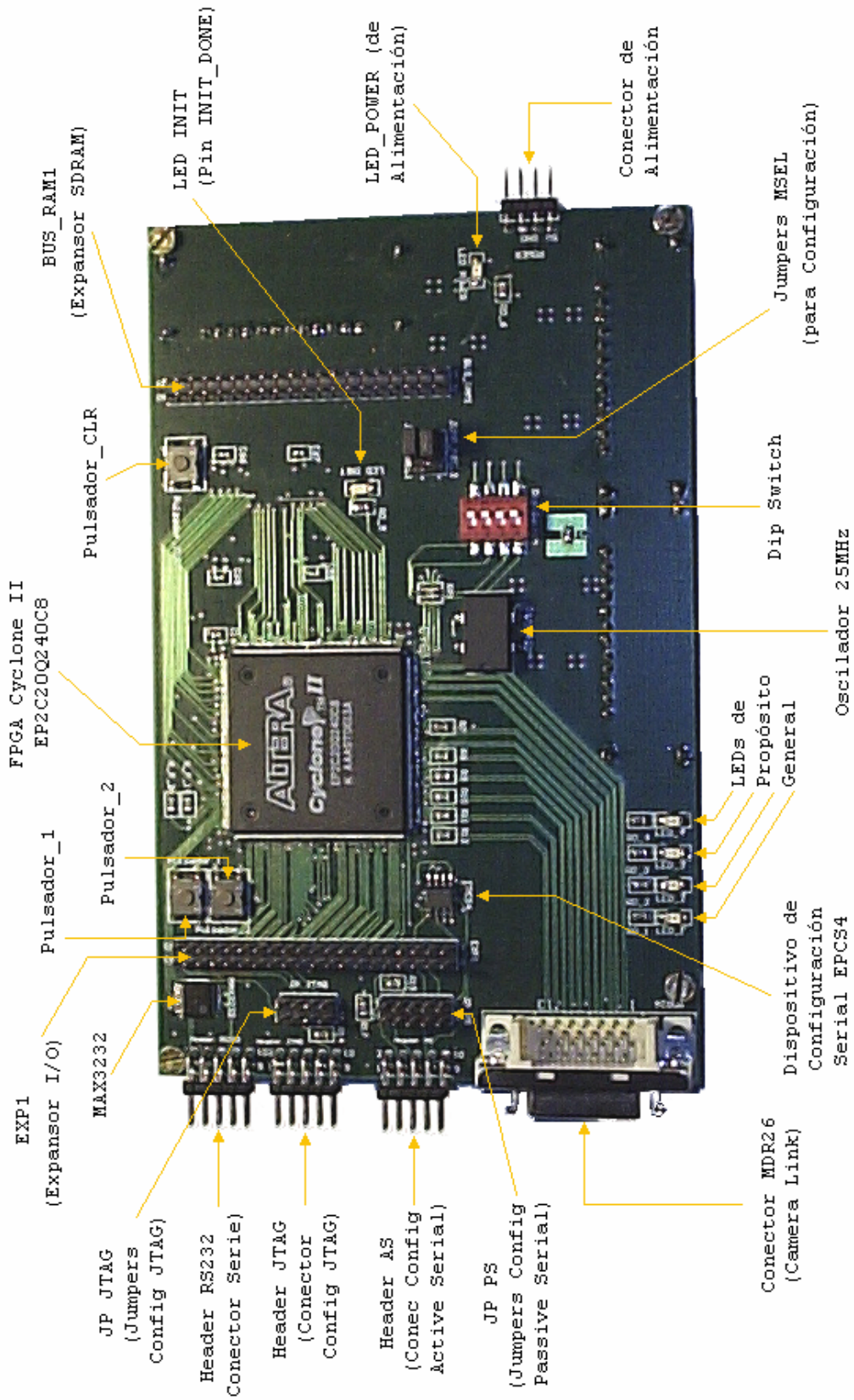
Para que la Placa IIE-Cyclone pudiera trabajar conectada con la Placa IIE-PCI fue necesario colocarle 2 buses de expansión que fueran compatibles con los que ya cuenta la Placa IIE-PCI. Además de estos buses de expansión se le agregó a la Placa IIE-Cyclone dos interfaces para ampliar las opciones de conexión con el exterior. Las mismas son una interfaz del tipo RS232 y una interfaz Camera Link.

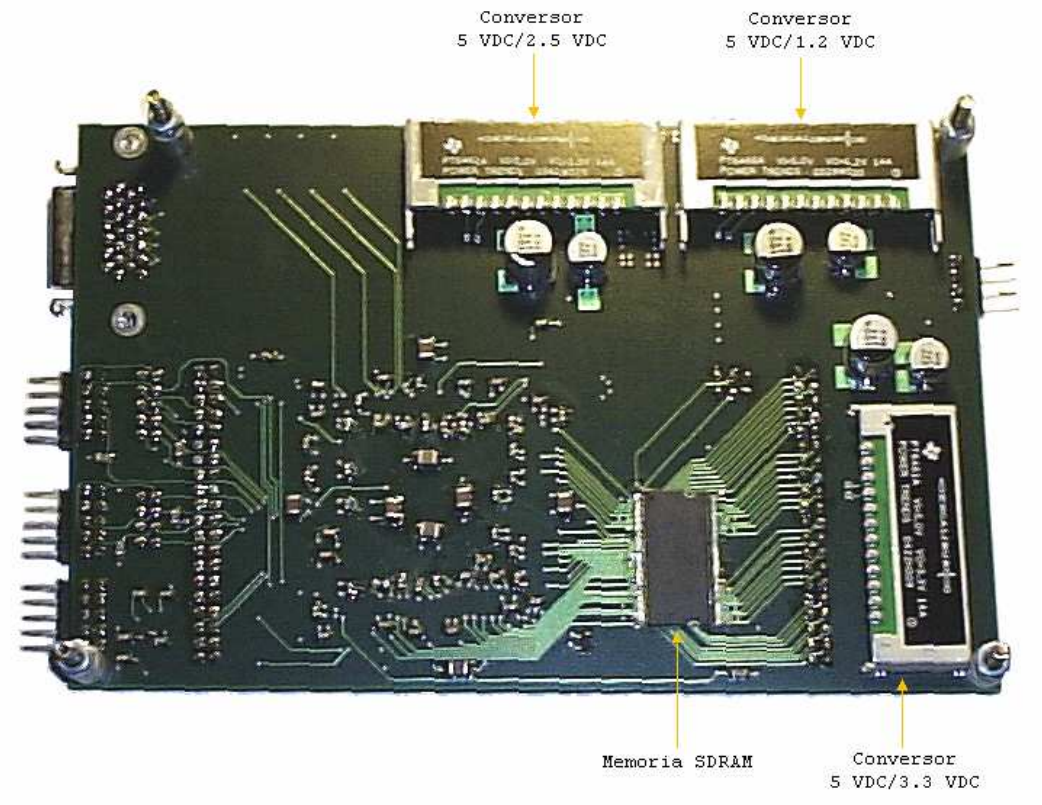
Para poder testear y probar aplicaciones sencillas decidimos conectar a pines de entrada y salida del FPGA una serie de pulsadores, dip switches y LEDs ya que mediante los mismos se puede realizar dichas pruebas de una manera sencilla y práctica.

Con respecto a la configuración del FPGA, se le agregó a la placa un dispositivo de configuración serial que cuenta con una memoria de 4Mb, con la misma se evita tener que configurar el FPGA cada vez que se enciende.

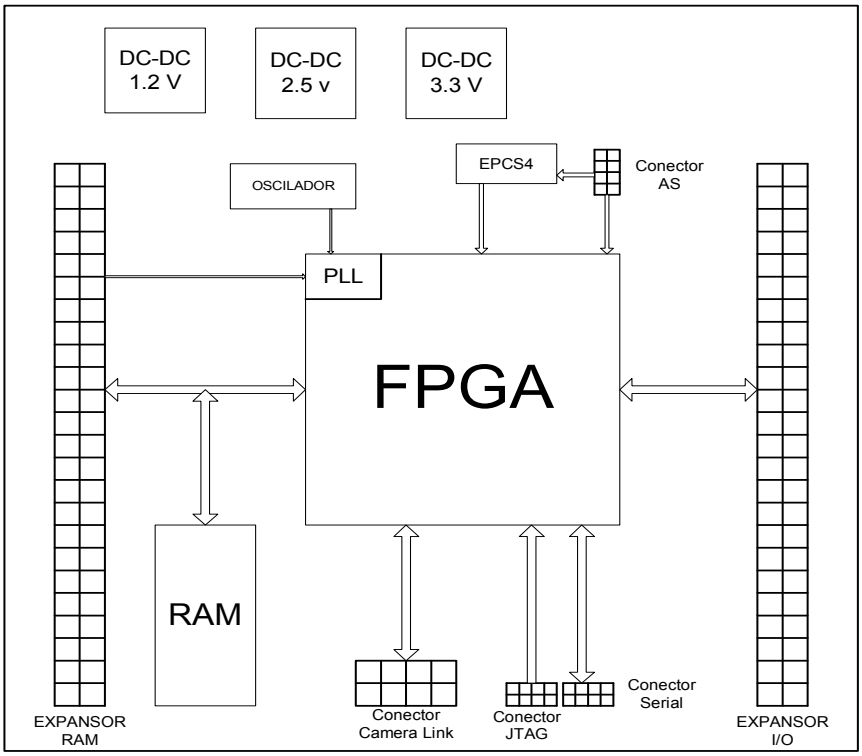
La placa se alimenta mediante un conector del tipo Mini 2.54 mm. El mismo debe ser conectado tanto cuando la placa trabaja de forma independiente o cuando trabaja conectada a la Placa IIE-PCI. La alimentación general recibida es de 5 Vdc. Luego, a partir de la alimentación de 5 Vdc y utilizando 3 conversores DC-DC se logran los niveles de voltaje necesarios para alimentar los distintos componentes de la placa.

En las siguientes figuras se muestran la vista superior e inferior de la placa con los distintos componentes señalados:


BUS_RAM1
(Expansor SDRAM)
LED INIT
(Pin INIT_DONE)
LED_POWER (de
Alimentación)
Conector de
Alimentación
Jumpers MSEL
(para Configuración)
Pulsador_CLR
Dip Switch
FPGA Cyclone II
EP2C20Q240C8
Oscilador 25MHz
Pulsador_1
Pulsador_2
LEDs de
Propósito
General
EXP1
(Expansor I/O)
MAX3232
Dispositivo de
Configuración
Serial EPCS4
JP JTAG
(Jumpers
Config JTAG)
Header RS232
(Conector Serie)
Header JTAG
(Conector
Config JTAG)
Header AS
(Conec Config
Active Serial)
JP PS
(Jumpers Config
Passive Serial)
Conector MDR26
(Camera Link)



En la siguiente figura se muestra un diagrama de bloques de los distintos componentes de la placa.



4) QUARTUS II - INSTALACIÓN Y LICENCIA

Descarga e Instalación del Quartus II Web Edition:

El software Quartus II es la principal herramienta para el desarrollo y programación de FPGAs de la empresa Altera. Este documento hace referencia a varias de sus funcionalidades.

Para su descarga e instalación se deben seguir los siguientes pasos:

1. En la página de Altera (www.altera.com) clickear en Download
2. Seguir las instrucciones de la página para completar el proceso de descarga. Ante cualquier dificultad, leer el documento *Installing the Quartus II Software* en el *Quartus II Installation & Licensing Manual for PCs* que se encuentra en la sección Documentación de la página
3. Ejecutar el archivo descargado y seguir los pasos para la instalación

Licenciamiento del Quartus II

Previo a usar el software Quartus II, se debe solicitar on-line en la página del fabricante una licencia e instalarla en la PC.
Cuando se solicita una licencia, Altera envía por e-mail un archivo license.dat que permite activar el software.
Para esto, se debe seguir los siguientes pasos:

1. Ir a la página www.altera.com/licensing
2. Clickear en “Quartus II Web Edition Software”
3. Seguir las instrucciones, incluyendo el llenado de un formulario con datos personales (e-mail entre ellos.)
4. Un archivo de licencia (license.dat) será enviado luego de algunos minutos a la dirección de e-mail proporcionada. Salvar este archivo.
5. Ejecutar el software Quartus II.
6. Ir a Tools → License Setup
7. En el campo “License File”, indicar el archivo recibido y guardado previamente.

5) ALIMENTACIÓN DE LA PLACA IIE-CYCLONE

Para la alimentación de la placa debe usarse un conector del tipo Mini 2.54 mm de 4 pines.
El pin al que se deben conectar 5VDC está debidamente identificado, así como los dos siguientes que deben conectarse a GND. El cuarto pin no está conectado.
Para lograr los 5VDC con un conector adecuado se sugiere usar una fuente de PC, usando el conector de Mini 2.54 mm de la diskettera (el cable rojo que indica los 5VDC debe conectarse al pin adecuado.)
El LED azul indica la presencia de alimentación en la placa.

6) SEÑAL DE RELOJ

La señal de reloj proviene de un oscilador de 25 MHz con el que cuenta la placa.
También se cuenta con una señal de reloj proveniente de la Placa IIE-PCI, cuando ésta está conectada. La misma llega a la Placa IIE-PCI a través de uno de los pines del expansor.
La salida externa con la que cuenta el PLL se conectó al pin de reloj de la SDRAM.

Señal	Pin FPGA	Tipo
Oscilador	94	entrada
Clk_Expansor	95	entrada
Clk_RAM	128	salida

Pinout señales reloj FPGA

Con el software Quartus II se pueden habilitar los PLLs del Cyclone II. Para esto se puede utilizar la mega función altpll. Mediante la misma se selecciona la señal a ser utilizada como referencia (oscilador o señal reloj proveniente del expansor) y entre otros parámetros, los factores de multiplicación y división de la señal de entrada.

7) INTERFACES DE ENTRADA Y SALIDA

Expansores

La placa cuenta con 2 expansores Macho de 2.54 mm, de 40 pines cada uno, distribuidos en 2 filas de 20 pines. La distancia entre ambos expansores y el lugar donde fueron colocados en la placa permiten la conexión con la Placa IIE-PCI.

Llamamos BUS_RAM1, al que tiene conectados los pines de la RAM (se encuentra claramente identificado en la placa) y EXP1 al que tiene conectadas en su mayoría pines I/O del Cyclone II (también se encuentra identificado en la placa).

PIN Expansor	Señal (Pin Cyclone II)	PIN Expansor	Señal (Pin Cyclone II)
1	DQ0 (125)	2	GND
3	DQ2 (134)	4	DQ1 (131)
5	DQ4(139)	6	DQ3(136)
7	DQ6(150)	8	DQ5(141)
9	GND	10	DQ7(156)
11	DQ15(130)	12	GND
13	DQ13(135)	14	DQ14(132)
15	DQ11(140)	16	DQ12(137)
17	DQ9(155)	18	DQ10(149)
19	GND	20	DQ8(157)
21	DQ31(195)	22	GND
23	DQ29(203)	24	DQ30(199)
25	DQ27(218)	26	DQ28(214)
27	DQ25(226)	28	DQ26(223)
29	GND	30	DQ24(230)
31	DQ16(194)	32	GND
33	DQ18(200)	34	DQ17(197)
35	DQ20(216)	36	DQ19(208)
37	DQ22(228)	38	DQ21(222)
39	GND	40	DQ23(231)

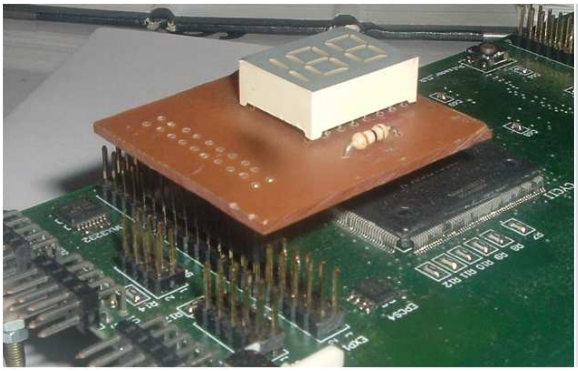
Pinout del expansor BUS_RAM1

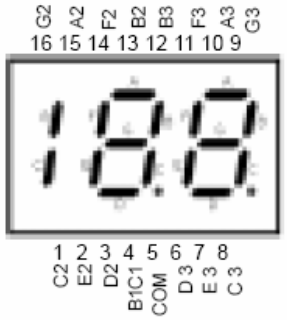
PIN Expansor	Señal (Pin Cyclone II)	PIN Expansor	Señal (Pin Cyclone II)
1	GND	2	CLK_EXP (95)
3	N/C	4	GND
5	DCLK	6	N/C
7	DATA	8	CONF_DONE
9	NSTATUS	10	NCONFIG
11	I/O (57)	12	I/O (58)
13	I/O (55)	14	I/O (56)
15	I/O (51)	16	I/O (52)
17	I/O (47)	18	I/O (50)
19	I/O (44)	20	I/O (46)
21	I/O (41)	22	I/O (42)
23	I/O (38)	24	I/O (39)
25	I/O (21)	26	I/O (37)
27	I/O (18)	28	I/O (20)
29	I/O (15)	30	I/O (16)
31	I/O (13)	32	I/O (14)
33	I/O (9)	34	I/O (11)
35	I/O (7)	36	I/O (8)
37	JTAG_TDO	38	JTAG_TDI
39	JTAG_TCK	40	JTAG_TMS

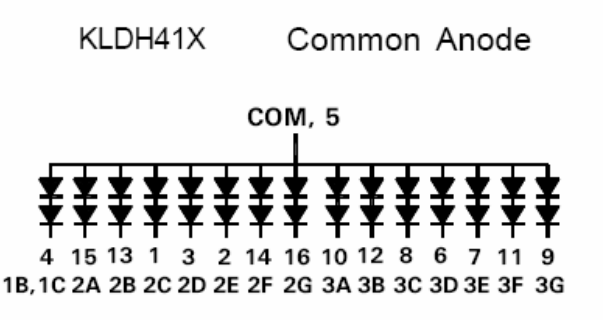
Pinout del expansor EXP1

Periférico: Display de 2 ½ dígitos

Para contar con una interfaz de salida adicional se creó una placa con un display de 2 ½ dígitos con un conector mini 2.54 mm hembra de 20 pines.
La misma se conecta al expansor EXP1 ocupando los pines 11 al 30 del mismo como se muestra en las siguientes imágenes y tabla.







PIN_DISP	SEGMENTO	PIN_EXP	PIN FPGA
5	A.C.*	11	I/O(57)
4	1B,1C	13	I/O(55)
1	2C	15	I/O(51)
2	2E	17	I/O(47)
3	2D	19	I/O(44)
14	2F	21	I/O(41)
15	2A	23	I/O(38)
16	2G	25	I/O(21)
13	2B	27	I/O(18)
12	3B	29	I/O(15)

PIN_DISP	SEGMENTO	PIN_EXP	PIN FPGA
6	3D	12	I/O(58)
7	3E	14	I/O(56)
8	3C	16	I/O(52)
		18	I/O(50)
		20	I/O(46)
		22	I/O(42)
		24	I/O(39)
9	3G	26	I/O(37)
10	3A	28	I/O(20)
11	3F	30	I/O(16)

Tabla con conexión entre pines del display, a qué segmento corresponden y a qué pines del expansor y del FPGA están conectados (* : Ánodo Común)

RS-232

La Placa IIE-Cyclone cuenta con una interfaz del tipo serial RS-232 que utiliza el chip MAX3232 para adaptar los niveles de voltaje de los pines I/O del FPGA con los niveles de voltaje especificados en la norma RS-232.

Al mismo se le conectan 4 pines I/O del Cyclone II y 4 pines del conector serial macho acodado de 10 pines con el que cuenta la placa (identificado como Header Serial).

Para utilizar esta interfaz es necesario configurar en el FPGA un bloque que cumpla la función de UART, como por ejemplo el core miniUART que se encuentra en Opencores.org

Señales:

Serial_Tx: transmisión de datos en forma serial desde el FPGA
Serial_Rx: recepción de datos en forma serial hacia el FPGA
Serial_RTS (Ready to Send): señal utilizada para el control de flujo
Serial_CTS (Clear to Send): señal utilizada para el control de flujo

Señal	Pin FPGA	Pin Conector
Serial _Tx	235	2
Serial_Rx	234	3
Serial_CTS	237	8
Serial_RTS	236	7

Pinout del FPGA y del Conector Serial

8) DIP SWITCH, LEDS Y PULSADORES

Dip Switch:

La placa cuenta con un dip switch de 4 llaves conectadas a pines I/O del Cyclone II mediante pull-ups.
Cuando los dip switch están en posición ON los pines del FPGA quedan conectados a GND y en posición OFF a 3.3V.

Componente	Pin FPGA
Switch 1	119
Switch 2	118
Switch 3	111
Switch 4	100

LEDs:

La placa cuenta con 4 LEDs de propósito general, conectados a pines I/O del FPGA que se encienden cuando se le asigna un "1" lógico a la salida correspondiente.
También cuenta con un LED llamado LED_INIT correspondiente al pin INIT_DONE del FPGA. El mismo es activado ("1") cuando el FPGA entra en modo usuario (user mode).

Componente	Pin FPGA
LED 1	64
LED 2	73
LED 3	78
LED 4	84
LED_INIT *	126

(*) Esta opción se habilita en el Quartus, de lo contrario puede ser usado como un pin de I/O.

Pulsadores:

La placa Anexa tiene tres pulsadores, dos de ellos de propósito general (Pulsador_1, Pulsador_2) y uno de ellos destinado a reiniciar el Cyclone a un estado inicial (Pulsador_CLR).

Componente	Pin FPGA
Pulsador_1	232
Pulsador_2	233
Pulsador_CLR *	238

(*) Este pin del FPGA es el correspondiente a DEV_CLRn. Si es habilitado en el Quartus, es la entrada de reinicio; de lo contrario es un pin de I/O.

9) CONFIGURACIÓN

La Placa IIE-Cyclone puede ser configurada de cinco maneras distintas:

Active Serial (AS), 20 MHz
Fast Active Serial, 40 MHz
JTAG placa aislada
Cadena JTAG con Placa IIE-PCI
Passive Serial (PS)

Para poder determinar qué modo de configuración va a ser utilizado es necesario ajustar los jumpers MSEL1 y MSEL2, los mismos se encuentran claramente identificados entre el Dip Switch y el bus expansor de la RAM.

Los métodos probados e implementados a la finalización del proyecto son:

1) Active Serial
2) JTAG placa aislada

En la siguiente tabla se especifica para cada tipo de configuración el posicionamiento correcto de los jumpers.

ESQUEMA DE CONFIGURACIÓN	JUMP_MSEL2	JUMP_MSEL1
AS (20 MHz)	0	0
PS	0	1
Fast AS (40 MHz)	1	0
JTAG placa aislada (1)	0(2)	0(2)
Cadena JTAG (3)	0(2)	0(2)

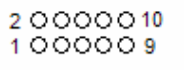
(1) Configuración JTAG toma precedencia sobre el resto de las configuraciones.
(2) No se deben dejar los pines de MSEL libres, se recomienda conectarlos a VCCIO o GND, si se usa sólo JTAG se deben conectar a GND.

A continuación detallaremos los dos métodos de configuración probados e implementados a lo largo del proyecto.

JTAG placa aislada:

Primero es necesario ajustar los jumpers de configuración como se detalla en la tabla anterior y dejar sin jumpers el conector JP_JTAG. El mismo se encuentra entre el bus de expansión y el Header JTAG.
El conector JP_JTAG cumple la función de cerrar la cadena JTAG mediante la utilización de jumpers. Se utiliza cuando se encuentra la Placa IIE-Cyclone conectada a la Placa IIE-PCI y se quiere configurar los FPGA de ambas placas mediante protocolo JTAG.

Luego es necesario utilizar el cable ByteBlaster II o algún otro cable para configuración JTAG (e.g.: ByteBlaster MV) conectándolo al Header JTAG de la placa (se encuentra claramente identificado en el borde izquierdo de la placa) y al puerto paralelo (e.g.: LPT1) del PC.
Al conectar el cable, el pin 1 del Header debe coincidir con la marca de color rojo en el cable del ByteBlaster II. En la siguiente figura se detalla el orden de los pines del Header JTAG.



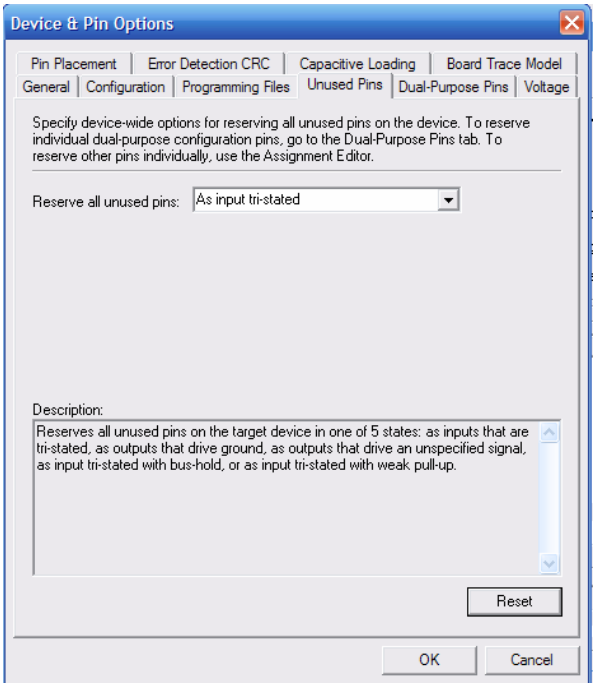
En la siguiente tabla se detalla el pinout del conector.

Pin Header JTAG	Señal JTAG
1	TCK
2	GND
3	TDO
4	VCCIO
5	TMS
6	N.C.
7	N.C.
8	N.C.
9	TDI
10	GND

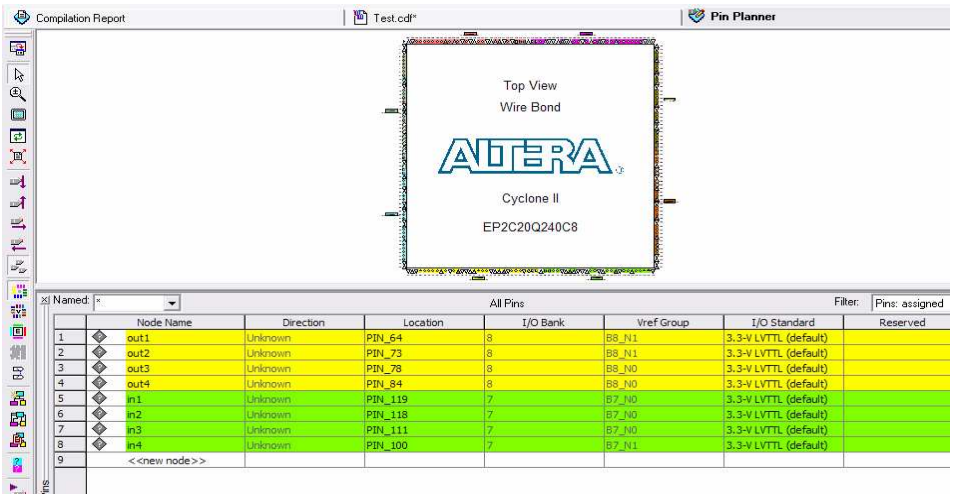
Configuración del Quartus II (Ver. 6.1 Web Edition)**:**

1) Luego de realizar la compilación completa se generan los archivos de programación para el FPGA (.sof) y elementos de memoria (.pof)

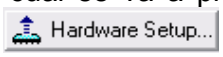
2) Seleccionar para que los pines sin usar queden en estado tri-state. Recordar que los pines que no usamos pueden estar conectados a la RAM u otro dispositivo y si no los reservamos podemos terminar quemando los drivers de línea de dichos dispositivos.
   Para realizar esto ir a Settings → Device → Device & Pins Options → Unused Pins y en el campo “Reserve all unused pins” seleccionar la opción “As input tri-stated”.

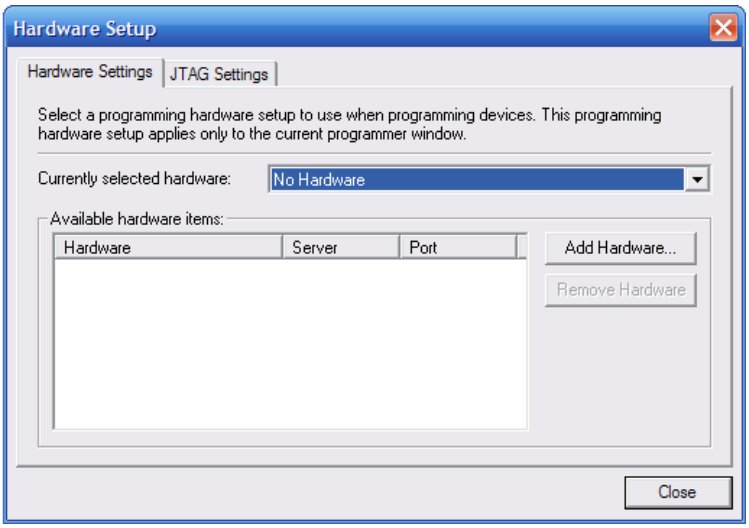


3) Realizar la asignación de pines en la sección Assignments → Pin Planner



4) Para programar seleccionar Tools → Programmer

4-1) Seleccionar el Hardware por el cual se va a programar el FPGA, seleccionando Hardware Setup 



4-2) Luego *Add Hardware* y agregar la interfaz de configuración, en nuestro caso usamos el puerto paralelo LPT1.

4-3) Luego de elegido el puerto paralelo se puede elegir el archivo de configuración seleccionando en *Add File* el archivo correspondiente (normalmente se usa .sof para configuración JTAG).

4-4) Seleccionar el modo de programación en modo JTAG seleccionando en Mode: JTAG

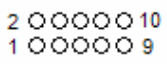
5) Por ultimo Presionar *Start* para configurar el dispositivo.

Mediante este método solo se configura el FPGA Cyclone II y la configuración se pierde cuando se le quita la alimentación al FPGA.

Configuración Active Serial:

Primero es necesario ajustar los jumpers de configuración como se especifica al comienzo del capitulo, luego es necesario conectar el cable ByteBlaster II (sólo este cable permite la configuración AS) entre el puerto paralelo del PC y el conector Header_AS (que se encuentra claramente identificado en el borde izquierdo de la placa).

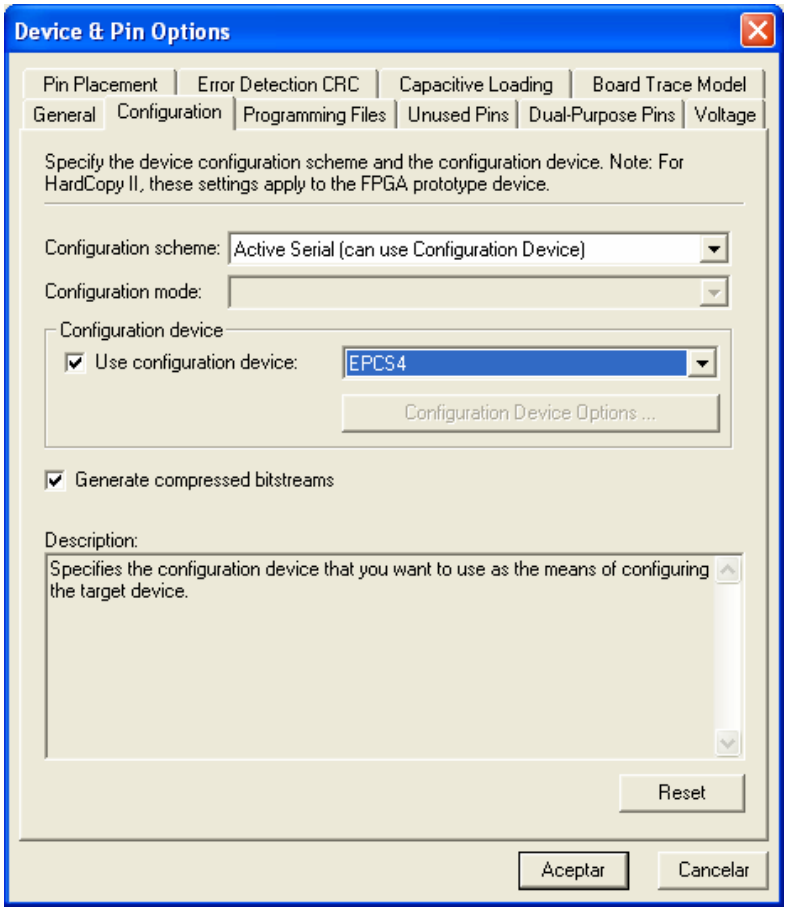
Al conectar el cable, el pin 1 del Header debe coincidir con la marca de color rojo en el cable. En la siguiente figura se detalla el orden de los pines del Header_AS.



En la siguiente tabla se especifica el pinout del conector Active Serial:

Header_AS	Señal AS
1	DCLK
2	GND
3	CONF_DONE
4	VCC (TRGT)
5	nCONFIG
6	nCE
7	DATA
8	nCS
9	ASDI
10	GND

En el Quartus II es necesario definir qué Dispositivo de Configuración Serie estamos usando. Para hacer esto debemos ir a Settings → Device → Device & Pins Options → Configuration y en el campo "Use configuration device:" seleccionar la opción "EPCS4".



Luego debemos seguir los pasos previamente detallados para configuración JTAG salvo que en el punto 4.3) seleccionaremos un archivo .pof y en el punto 4.4) debemos seleccionar el modo de programación: Active Serial Programming.
(Nota: si elegimos el EPCS4 en forma adecuada como fue detallado el archivo de configuración .pof será de 513 KB)

Mediante este método se carga la configuración en la memoria no volátil del dispositivo de configuración serie EPCS4. De esta forma al quitar la alimentación de la placa, la configuración queda guardada en la memoria y al alimentar la placa nuevamente, el FPGA carga la configuración de la misma.

10) DISEÑOS VHDL (TUTORIAL)

En los distintos archivos comprimidos incluidos en el CD (leds.rar, segundero.rar y salida_232.rar) se incluyen los archivos necesarios para probar estos diseños en la Placa IIE-Cyclone.
En todos los casos se incluyen los archivos VHDL (.vhd) necesarios para generar los proyectos y compilación de los mismo.
Los archivos .qpf (Quartus II Project File) fueron creados con el Quartus II 6.1 por lo que otras versiones del programa pueden mostrar una advertencia indicando que éste será modificado.
Se incluyen los archivos .pof y .sof de programación para poder programar los diseños como se explica en el capítulo 10 sin necesidad de editarlos o compilarlos.
También se incluyen los archivos .pin con la asignación de pines y (salvo en el caso del segundero) los archivos .vwf de simulación de tiempos.

LEDS

Este diseño es una simple demostración del dip-switch, los LEDs y los pulsadores.
Implementa las funciones lógicas AND y OR entre los pines conectados al dip-switch y los pulsadores (especificadas en el archivo Leds.vhd) para controlar el encendido y apagado de los 4 LEDs.

SEGUNDERO

Este diseño implementa un segundero hexadecimal dividiendo la frecuencia del Oscilador de la placa (25 MHz) entre 25,000,000 (con el core freq_div.vhd usado en el curso de Diseño Lógico 2) e ingresándola a un contador de 4 bits (contador.vhd) que da salida a los LEDs.
Adicionalmente, el bloque hex27seg.vhd convierte esta salida de 4 bits a dígitos de 7 segmentos para ser usado con el periférico Display de 2 ½ dígitos. En caso de no usarse, el periférico deben eliminarse los pines que utiliza éste del Pin Assignment.

SALIDA_232

Este diseño muestra la funcionalidad de comunicación de la Placa IIE-Cyclone con una PC a través del Puerto Serie de la misma utilizando el protocolo RS-232.
En este diseño está implementado solamente el envío de datos desde una memoria RAM de 1024 KB a 115200 bps, 8 bits de datos, 1 bit de parada, sin paridad, sin handshaking.
Todos estos parámetros pueden ser cambiados en los archivos VHDL. Para hacerlo, consultar las especificaciones del core miniUART (www.opencores.org).
El archivo con el contenido inicial de la memoria es: init_mem_1K.hex. Éste puede ser fácilmente editado en el Quartus II y luego se lo debe especificar como memoria inicial de la memoria RAM_comp.vhd (editándola con el MegaWizard Plug-In Manager.)
Para el envío de datos, debe conectarse el cable serie en el Header Serial y el Puerto Serie del PC, abrir algún programa de comunicación (e.g.: HyperTerminal), setear los parámetros de comunicación y presionar el Pulsador 1.

Proyecto Placa IIE-Cyclone

Andrés Bergeret, Gabriel Colombo y Guillermo Di Maio

Universidad de la República, Instituto de Ingeniería Eléctrica, Montevideo Uruguay, Mayo 2007

Resumen—Este proyecto abarca la construcción de una plataforma de lógica reconfigurable basada en un FPGA Cyclone II de Altera. La misma cuenta con una interfaz LVDS (Low Voltage Diferential Signaling) con el fin de poder realizar adquisición de imágenes y una interfaz RS232 para debug. Esta plataforma es compatible con la Placa IIE-PCI realizada en el año 2003 por Sebastián Fernández y Ciro Mondueri dentro del IIE (Instituto de Ingeniería Electrica) de la Universidad de la Repúbila - Montevideo, Uruguay. Posteriormente se desarrolló una variante del algoritmo LOCO-I(Low Complexity Losless Compression for Images) para probar sobre la plataforma, el mismo fué realizado en lenguaje VHDL para imágenes de 8 bit/píxel. Se lograron tasas de compresión de promedio de 3.5 bits/píxel con un throughput promedio de 1.88 Mpíxel/seg.

I. INTRODUCCIÓN

El Proyecto Placa IIE-Cyclone se divide en dos etapas; la primera consistió en el diseño y desarrollo de una plataforma de lógica reprogramable y una segunda etapa basada en el desarrollo de una aplicación sobre la misma.
Se desarrolló una plataforma genérica, lo que permite su reutilización en un sinnúmero de aplicaciones. Con respecto a la aplicación se implementó un algoritmo de compresión sin pérdida para imágenes en escala de grises de 8 bpp (bits/píxel).
Esta plataforma cuenta con un FPGA Cyclone II de la familia Altera y puede operar en forma stand-alone o en forma integrada con la Placa IIE-PCI[1]. En esta configuración se utiliza la Placa IIE-PCI para la comunicación con la PC a través del bus PCI y la Placa IIE-Cyclone para el procesamiento. Con respecto a la entrada y salida de datos, se optó por dotar a la placa con 2 interfaces distintas. La primera, una interfaz serial RS-232 la que permite una fácil comunicación con un PC. La segunda una interfaz Camera Link [2] que utiliza señales del tipo LVDS [3] (Low Voltage Differential Signaling). Estas son señales diferenciales de bajo voltaje, lo que permite buena inmunidad al ruido y poder alcanzar altas tasas de transferencia. El propósito de esta interfaz es brindarle la capacidad de desarrollar sobre la placa aplicaciones de procesamiento de imágenes y video.
El algoritmo de compresión de imágenes sin pérdida que se implementó está basado en el algoritmo LOCO-I y fue desarrollado en lenguaje VHDL de diseño de hardware. Al LOCO-I le realizamos varias modificaciones para mejorar su desempeño en un FPGA y optimizar los recursos del mismo. Trabajamos con imágenes de 100 x 100 píxeles en escala de grises, utilizando solamente un 5% del chip.

II. DESCRIPCIÓN DEL HARDWARE

En la Figura 1 se observa una imagen de la cara superior de la placa IIE-Cyclone.

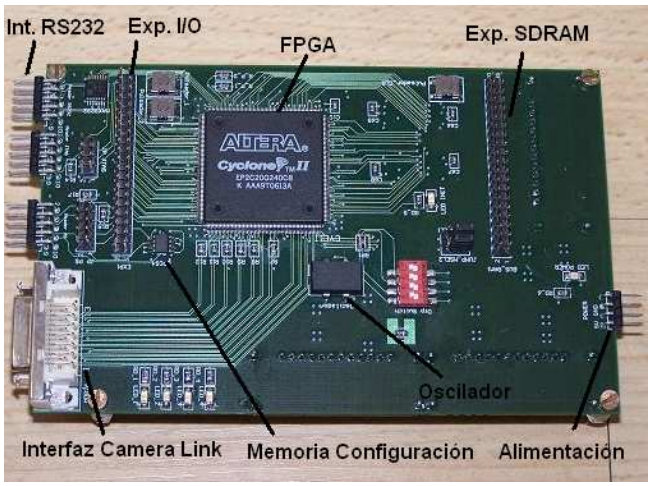


Figura 1: Foto cara superior Placa IIE-Cyclone.

En la Figura 2 se observa una imagen de la cara inferior de la placa IIE-Cyclone.

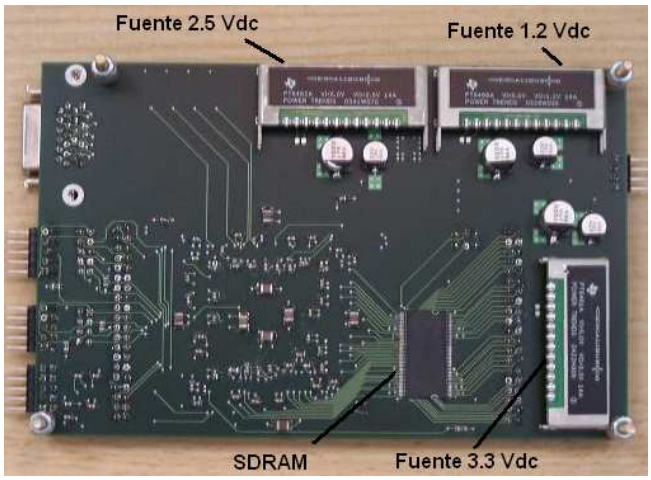


Figura 2: Foto cara inferior Placa IIE-Cyclone.

Andrés Bergeret, Gabriel Colombo y Guillermo Di Maio son estudiantes de Ingeniería Eléctrica de la Facultad de Ingeniería de la UDELAR

El FPGA utilizado en la placa es un Cyclone II EP2C20Q240C8 [4] de Altera; el mismo dentro de sus principales características cuenta con 120 pines de entrada/salida multipropósito, 18752 elementos lógicos y cuatro PLLs internos.
La Placa IIE-Cyclone también cuenta con una memoria on-board del tipo SDRAM de 128 Mbits [5].
Como los FPGA no cuentan con una memoria no volátil se le agregó a la placa un dispositivo de configuración serial modelo EPCS4 de Altera que cuenta con una memoria de 4Mb, con la misma se evita tener que configurar el FPGA cada vez que se enciende.
La Placa IIE-Cyclone cuenta con una interfaz Camera Link que utiliza un conector del tipo MDR-26 de 3M y señales del tipo LVDS. Es una interfaz diferencial de alta velocidad diseñada especialmente para comunicarse con cámaras de video digital. Se utilizó la configuración básica de Camera Link; la misma posee 4 señales para control de la cámara, 4 señales de datos desde la cámara al FPGA, una señal de reloj y dos señales seriales, una desde la cámara al FPGA y la otra en sentido contrario.

La otra es una interfaz serial RS-232, con un chip MAX3232 que se encarga principalmente de adaptar los niveles de voltaje a la salida de los pines del FPGA a los niveles de voltaje especificados en el estándar RS-232.

Con respecto a la estructura física de la placa al diseñarla se decició que la misma tuviera 4 capas; las dos exteriores de señal, una de las interiores de tierra y la restante que contuviera los 4 planos distintos de voltaje (5 Vdc, 1.2 Vdc, 3.3 Vdc y 2.5 Vdc).

Principales características de la placa:

a. FPGA Cyclone II EP2C20Q240C8 de ALTERA.
b. SDRAM 128 Mb MT48LC4M32B2 de MICRON.
c. Dispositivo de configuración serial de 4 Mb EPCS4 de ALTERA.
d. Oscilador de 25 MHZ CMX309FBC25.000MTR de CITIZEN AMERICA CORPORATION.
e. PLL: se utiliza uno de los cuatro PLL internos del FPGA.
f. Conversor 5 Vdc/3.3 Vdc PT6461 de TI(Texas Instruments).
   Conversor 5 Vdc/2.5 Vdc PT6462 de TI.
   Conversor 5 Vdc/1.2 Vdc PT6466 de TI.
g. 4 LEDs multipropósito, 1 LED indicador de alimentación y un LED indicador de configuración.
h. 3 Pulsadores,DIP SWITCH 4 llaves.
i. Interfaz Camera Link.
j. Interfaz RS-232.

III. APLICACIÓN

La aplicación consiste en el diseño, desarrollo e implementación de un algoritmo de compresión de imágenes sin pérdida en hardware.

Para diseñar nuestra aplicación nos basamos en el algoritmo LOCO-I [6], en su implementación realizada por el Jet Propulsion Laboratory del California Institute of Technology a pedido de la NASA [7] y en la estandarización del LOCO-I realizada por la ISO/ITU para la compresión de imágenes sin pérdida, conocida como JPEG-LS [8].

El algoritmo se basa en un esquema de dos componentes independientes, el modelado y la codificación.
El modelado infiere el comportamiento estadístico de la imagen. Se logra mediante el escaneo píxel a píxel de la imagen en un orden pre-definido (raster scan).
En nuestra implementación no utilizamos la codificación de RLE (Run Length Encoding), que codifica de manera distinta y más eficiente las secuencias de píxeles iguales. Esta simplificación no le quita eficiencia al algoritmo ya que el mismo está pensado para trabajar con imágenes naturales, en las cuales es muy poco probable encontrar regiones de tonalidades uniformes. Nos basamos en el trabajo realizado para la NASA para tomar esta decisión.

Además, realizamos otras modificaciones sobre el algoritmo basados en el estándar JPEG-LS, estos cambios fueron introducidos para poder comparar los resultados de nuestra aplicación utilizando un programa en C que implementa JPEG-LS.

El algoritmo conceptualmente se puede dividir en cinco bloques diferentes:

1) Modelado del Contexto: clasificación de los píxeles en distintos contextos, acorde a los valores de los píxeles previamente codificados.

2) Estimación del píxel: se realiza una estimación del píxel a codificar mediante el predictor fijo, luego se ajusta el mismo mediante una corrección (de aquí en adelante llamada bias) y finalmente se calcula la diferencia con respecto al valor real del píxel (error o residuo).

3) Codificación de Golomb: mapeo del error a un valor entero no negativo y posterior codificación usando códigos de Golomb de largo variable.

4) Actualización de los valores del Contexto: tomando en cuenta el valor de la muestra actual se actualiza la información almacenada para cada uno de los contextos.

5) Modo Run Length: modo de procesamiento en el cual se entra si las muestras que vienen siendo analizadas son iguales. Se utiliza esta característica para lograr un procesamiento más eficiente de la información.

El primer cambio realizado es que no implementamos el bloque de Run Length. El fundamento para prescindir del bloque Run Length es que el mismo complicaría la implementación en Hardware, y que no aprovecharíamos sus beneficios ya que en imágenes naturales no es común la aparición de regiones con el mismo valor de píxel.
El segundo cambio fue modificar la cantidad de regiones del cuantizador. En nuestra implementación utilizamos los siguientes niveles:

$$Q_7(n) = \begin{cases} q_1 \text{ si } n \geq 13 \\ q_2 \text{ si } 5 \leq n \leq 12 \\ q_3 \text{ si } 2 \leq n \leq 4 \\ q_4 \text{ si } -1 \leq n \leq 1 \\ q_5 \text{ si } -2 \leq n \leq -4 \\ q_6 \text{ si } -5 \leq n \leq -12 \\ q_7 \text{ si } n \leq -13 \end{cases}$$

Para realizar estos dos primeros cambios nos basamos en la implementación del LOCO-I realizada para la NASA [7].
Los siguientes cambios mencionados se realizaron para compatibilizar nuestra implementación con el decodificador utilizado:

No utilizamos el píxel *e* para determinar el contexto asociado al píxel a ser codificado. Esto determina que para determinar el contexto utilicemos solamente: Q_7 (b-c), Q_7 (d-b), Q_7 (c-a). Este cambio termina llevando la cantidad de contextos que utilizamos a 172.

Utilizamos el siguiente mapeo para llevar el *Residuo* de una distribución TSGD(Two Side Geometric Ditribution) a una OSGD(One Side Geometric Distribution):

```
if (k=0) && (2B <= -N) {
        if (error >= 0)
                M(ϵ)= 2ϵ + 1
        else
                M(ϵ) = -2ϵ – 2
        }
else {
        if (error >= 0)
                M(ϵ)= 2ϵ
        else
                M(ϵ) = -2ϵ – 2
        }
```

Para poder calcular el gradiente para los bordes de la imagen, agregamos una columna de píxeles a la izquierda de la imagen, una columna de píxeles a la derecha de la imagen y una fila de píxeles por encima de la imagen. En la figura 3 se detallan dichos agregados. El valor de dichos píxeles queda determinado por:

Fila superior: I(0, j) = 0
Columna izquierda: I(i, 0) = I(i -1,1)
Columna derecha: I(i, j) = I(i-1,,j)

Tomamos a la imagen como una matriz I de *i* filas y *j* columnas.

0	0	0	0	0	0
0	0	0	90	74	74
0	68	50	43	70	70
68	64	98	98	98	98
64	99	99	98	98	98

Figura 3: Agregado a la imagen para el procesamiento de bordes.

Características de nuestra implementación:

Implementa: algoritmo de compresión sin pérdida LOCO-I modificado.

Lenguaje: VHDL.

Características de la imagen a procesar: Imagen en escala de grises, de tamaño 100 x 100 y 8 bits/píxel.

Tasa de compresión: 35.15 %

Tiempo de compresión: 5.3 ms

Throughput: 1.88 Mpíxeles/seg

Frecuencia máxima de funcionamiento: 24 MHz

Cantidad de elementos lógicos utilizados: 600

Cantidad de registros: 208

Memoria utilizada: 8192 bits

Para la simulación tomamos una imagen en formato pgm, de tamaño 100 píxeles por 100 píxeles y resolución de 8 bits/píxel. En la figura 4 se muestra dicha imagen.

Figura 4: Imagen original: marte.pgm

El tamaño de la imagen original es de 10000 Bytes y luego de ser comprimida por nuestra implementación del algoritmo pasa a ser de 6514 Bytes.
Para corroborar nuestro algoritmo utilizamos una implementación de JPEG-LS diseñada en el Departamento de Ingeniería Eléctrica y Computación de la Universidad de British Columbia. El mismo fue desarrollado en C y le realizamos una serie de modificaciones para que se adapte a nuestra implementación.
Comprimimos la imagen con la etapa de compresión de dicho software y la comparamos con la salida de nuestro compresor. Verificamos Byte a Byte y comprobamos el correcto funcionamiento de nuestra implementación.
También verificamos que al utilizar el descompresor con los datos obtenidos se obtiene nuevamente la imagen original.
En la Tabla 1 se muestra los resultados comparados entre nuestra implementación del LOCO-I y el estándar JPEG-LS. Las imágenes utilizadas son de 512 x 512 píxeles y con 8 bits/píxel.

Imagen	JPEG-LS (*)	Compresión (**)	Imp. LOCO (*)	Compresión(**)
Fish	113.748	3,471	114.197	3,485
Vacaciones	186.231	5,683	186.571	5,694
Jardin	162.569	4,961	162.707	4,965
IIE-Cyclone	117.323	3,580	117.603	3,589

(*) bytes
(**) bits/píxel

Tabla 1: Comparación JPEG-LS vs Implementación LOCO-I

En la figura 4 se muestran las imágenes utilizadas para la comparación entre la compresión realizada con el estándar JPEG-LS y la realizada por nuestra implementación.

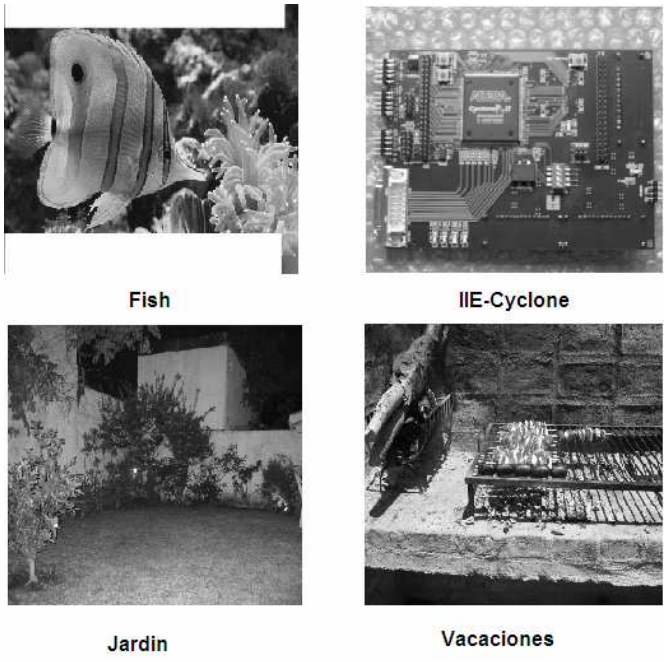


Figura 4: Imágenes utilizadas para la comparación.

IV. CONCLUSIONES

El modo de trabajo standalone brinda la opción de una plataforma de pruebas muy sencilla de utilizar y fácil de configurar mientras que el modo de trabajo con la Placa IIE-PCI da la opción de contar con una plataforma conjunta de trabajo con interfaz PCI.

Se logró desarrollar una variante del algoritmo LOCO-I adaptado a las características del hardware. Esta aplicación requiere pocos recursos del FPGA (1000 LEs) y alcanza tasas de compresión de 3.51 bits/píxel, similares a las obtenidas por el estándar JPEG-LS. Con respecto a la velocidad de procesamiento de la misma es de 1.88 Mpíxeles/seg, cercana a varias implementaciones en hardware que existen.

V. REFERENCIAS

[1] S. Fernández, C. Mondueri, "IIE-PCI: Una plataforma de desarrollo para el bus PCI", Facultad de Ingeniería-Universidad de la Republica, Montevideo Uruguay Diciembre 2003, Página Web: http://modueri/iiepci.

[2] "Specifications of the Camera Link Interface Standard for digital Cameras and Frame Grabbers", versión 1.1, Enero 2004.

[3] Jon Brunetti, Brian Von Herzen, "The LVDS I/O Standard", versión 1.1, Noviembre 1999.

[4] Altera, "Cyclone II Device Handbook, Volume 1", versión 2.2 2005.

[5] Micron, " Synchronous Dram MT48LC4M32B2 -1 MEG x 32 x 4 Banks", 2001.

[6] Marcelo J. Weinberger, G. Seroussi, G Sapiro, "LOCO-I A low complexity, context-based, lossless image compression algorithm", Marzo 1996.

[7] M. Klimesh, V. Stanton, D. Watola, "Hardware Implementation of a Lossless Image Compression Algorithm Using a Field Programmable Gate Array", Febrero 2001.

[8] FCD 14495 – Public Draft, ISO/IEC JTC1/SC29 WG1, "Lossless and near-lossless coding of continuous tone still images (JPEG-LS)", 1997/7/16.